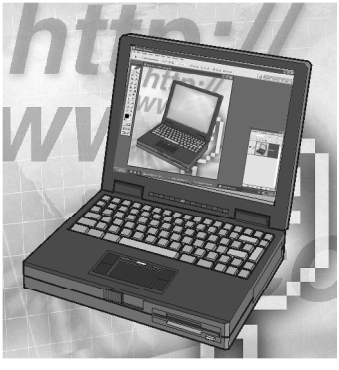
O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI
O‘RTA MAXSUS, KASB-HUNAR TA’LIMI MARKAZI

SH.I. RAZZOQOV, M.J. YUNUSOVA

DASTURLASH

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

9-nashri



Toshkent — «ILM ZIYO» — 2017

UO‘K: 004. 438 (075)
KBK 32.973-018
R18

Oliy va o‘rta maxsus, kasb-hunar ta’limi ilmiy-metodik birlashmalari faoliyatini muvofiqlashtiruvchi Kengash tomonidan nashrga tavsiya etilgan.

O‘quv qo‘llanmada zamonaviy va eng keng tarqalgan dasturlash tillaridan biri bo‘lgan *Turbo Paskal*ning asoslari beriladi. Kitob *Turbo Paskal*ning 7.0 versiyasi muhitida algoritmlash va dasturlash bo‘yicha ma’lumotlar asosida yozilgan bo‘lib, informatika va axborot texnologiyalari yo‘nalishlari bo‘yicha o‘quv dasturiga mos ravishda tuzilgan. Shuningdek, o‘quv qo‘llanma dasturlash bo‘yicha dastlabki ma’lumotlarni egallamoqchi bo‘lgan boshlovchilar uchun ham foydali bo‘lishi mumkin.

Taqrizchilar: **A. A. XOLYIGITOV** — O‘zbekiston Milliy universiteti «Dasturlash va tarmoq texnologiyalari» kafedrasi mudiri, professor; **K.Z. OBIDOV** — Buxoro oziq-ovqat va yengil sanoat texnologiyasi instituti «Informatika» kafedrasi mudiri, dotsent; **M.M. XUDOYQULOV**— Buxoro kommunal xo‘jaligi kasb-hunar kolleji direktori.

ISBN 978-9943-16-299-0

KIRISH

Axborot va telekommunikatsiya texnologiyalarining jadal sur'atlar bilan rivojlanishi natijasida insoniyat yangi informatsion jamiyat poydevorini qurishga erishdi. Informatsion jamiyatni shakllantirishda zamonaviy kompyuterlar va dasturiy vositalarning o'rni beqiyosdir.

Respublikamizda ham axborot va telekommunikatsiya texnologiyalarini rivojlantirish va ushbu soha uchun malakali mutaxassislar tayyorlash tizimini shakllantirishga alohida e'tibor berilmoqda. Bu borada axborot texnologiyalari dastur mahsulotlarining keng imkoniyatlarini talabalarga chuqur o'rgatish, ularning dasturlash madaniyatini shakllantirish muhim ahamiyat kasb etadi.

Respublikamizning ta'lim muassasalarida «Informatika», «Informatika va informatsion texnologiyalar» fanlarining o'quv rejalarida yuqori saviyali algoritmik tillardan birining imkoniyatlarini o'rgatish rejalashtirilgan. *Turbo Paskal* dasturlash tili mana shunday algoritmik tillardan biridir. Barcha dasturlash tillarida uchraydigan tushuncha va tuzilishlarni o'z ichiga olganligi, katta imkoniyatlarga egaligi, shuningdek, obyektga yo'naltirilgan dasturlash tillarini o'rganishga asos bo'lib xizmat qilganligi sababli, o'quv yurtlarida aynan shu tilni chuqur o'rganishga e'tibor qaratilgan. Shveysariyalik olim Niklas Virt tomonidan talabalarga dasturlashni o'rgatish vositasi sifatida o'ylab chiqilgan Paskal tili, Amerikaning Borland korporatsiyasi xodimi, *Turbo Paskal*ning yaratuvchisi va g'oyachisi Anders Xeylsberg hamda korporatsiyaning iste'dodli xodimlari tomonidan bugungi kunda har qanday oddiy dasturlar tuzishdan tortib, ma'lumotlar bazasini boshqarishning murakkab relatsion tizimlarini ishlab chiqishdek masalalarni hal qila oladigan quvvatli zamonaviy dasturlash tizimiga aylandi.

Windows va *Windows* muhitida dasturlar ishlab chiqish uchun *Borland Pascal with Object* va *Delphi* instrumental vositalarining paydo bo'lishi yana bir marta bu tilning cheksiz imkoniyatlarga ega ekanligini ko'rsatdi. *Borland Pascal* ham, *Delphi*da ishlatiladigan

Object Pascal ham *Turbo Paskal*ga asoslanadi va uning g‘oyasini rivojlantiradi.

Mazkur nashrni tayyorlashda kitobga «Turbo Paskalning grafik imkoniyatlari» deb nomlangan yangi bobi kiritildi, ammo kitobning katta qismi o‘zgarishsiz qoldirildi, chunki *Turbo Paskal* tilining o‘zida ham o‘zgarishlar bo‘lmagan.

Kitob kasb-hunar kollejlari talabalari uchun o‘quv qo‘llanma sifatida yozildi. Ammo qo‘llanma *Turbo Paskal* tilini o‘rganayotganlar, ishlatayotganlar uchun ham foydali bo‘lishi mumkin. Qo‘llanmada *Turbo Paskal*ning oxirgi 7-versiyasi imkoniyatlariga urg‘u berilgan. U dasturlash san’atining boshlang‘ich bilimlariga ega bo‘lish uchun kerakli bilimlarni beradigan hamma bo‘limlarni o‘z ichiga olgan.

I bob. PASKAL DASTURLASH TILI

1.1. Dasturlash haqida asosiy tushunchalar

1.1.1. Algoritm tushunchasi. Kundalik hayotda, ko‘pincha, oldindan ko‘zda tutilgan amallar qatori ko‘rsatilgan turli ko‘rsatmalarni uchratishga to‘g‘ri keladi. Ularni ketma-ket bajarib, kutilgan natijaga erishish mumkin. Ko‘rsatmalar ketma-ketligi matematik masalalarni yechganda ham tuziladi. Masala mazmuni turli sohalarga tegishli bo‘lishidan qat’i nazar, ularni yechishning umumiy tomonlari bor — yechish jarayonlari qisqa ko‘rsatmalar ketma-ketligi orqali tasvirlanadi. Bu ketma-ketliklar masalani yechish algoritmini tashkil qiladi. Ko‘rsatilgan amallarni bajara borib, talab etilgan natijaga erishish mumkin.

Berilgan ma’lumotlarni qo‘llab, talab etilgan natija hosil bo‘lishini ta’minlovchi aniq, bir ma’noli ko‘rsatmalar ketma-ketligi *algoritm* deb yuritiladi. Masalalarni yechishda ishtirok etuvchi barcha kattaliklar *berilgan ma’lumotlar*, algoritmning bajarilishidan oldin ma’lum bo‘lgan ma’lumotlar *dastlabki ma’lumotlar*, masala yechimining natijasi — so‘nggi *hosil bo‘luvchi ma’lumotlar* deyiladi.

Algoritmlarni turlicha yozish mumkin. Algoritmni ifodalash shakli, tarkibi va amallar miqdori mazkur algoritmning ijrochisi kim bo‘lishiga bog‘liq. Agar masala kompyuterda yechilsa, masalani yechish algoritmi mashinaga tushunarli shaklda, ya’ni *dastur* ko‘rinishida yozilishi kerak.

Hozirgi zamon EHMlari (kompyuterlari) inson qo‘llaydigan rus, o‘zbek, ingliz yoki boshqa shu kabi biron-bir tilda tuzilgan dasturni tushuna oladigan darajada takomillashgan emas. Shuning uchun, mashinaga mo‘ljallangan buyruqlar u tushunadigan shaklda yozilishi kerak. Bu maqsadda algoritmik yoki dasturlash tillari deb ataluvchi sun’iy tillar qo‘llaniladi.

1.1.2. Kompyuter va dastur. Kompyuter — murakkab elektron qurilma, uni mufassal o‘rganish uchun ancha vaqt zarur bo‘ladi.

Biroq, kompyuterdan foydalanuvchilar — dasturchilar uchun uning ishi haqida eng umumiy ma'lumotlarga ega bo'lish yetarlidir. Dasturchini, asosan, mashina dastur buyruqlarini qanday qilib amalga oshirishi, bunda u qanday amallarni bajarishi qiziqtiradi. Kompyuter dasturni bajara borib, turli ma'lumotlar (sonlar, mantiqiy qiymatlar, matnlar va h.k.) ustidan amallar bajaradi. Ma'lumotlar ustidagi barcha amallarni mashina *protsessori* bajaradi.

Har qanday kompyuterning eng asosiy qurilmalaridan biri uning *xotirasidir*. Mashina o'z xotirasida masalani yechish uchun zarur bo'lgan barcha ma'lumotlarni saqlash xususiyatiga ega. Oddiy bo'lishi uchun mashina xotirasini kataklarga bo'lingan ko'rinishda tasvirlash mumkin. Bu kataklar *mashina so'zi* yoki *yacheykalar,* deb ataladi. Odatda, ma'lumotlarning har biri alohida yacheykaga joylashtiriladi. Xotiraga yozilgan ma'lumotlarni bir necha marta o'qish va hisoblashlarda foydalanish mumkin. Biroq xotiraning ma'lum yacheykasiga yangi so'z kiritilsa, shu yacheykadagi oldingi saqlab turilgan ma'lumot o'chadi.

Kompyuter protsessori ma'lumotlarni xotiradan o'qiydi, ular ustida dasturda ko'rsatilgan zarur amallarni bajaradi va hisoblash natijalarini yana kompyuter xotirasiga yozadi. Mashina masala mazmunini tushunmaydi, chunki kompyuter dasturda yozilgan ko'rsatmalarni aniq bajaruvchi elektron robotdir. Shu sababli unga bajarish uchun aniq va bir ma'noli tavsiflangan ko'rsatmalar berilishi lozim.

Dastur ham dastlabki ma'lumotlar kabi EHM xotirasiga kiritiladi va uning har bir buyrug'i ma'lum usul bilan kodlanadi. Mashina protsessori dastur matnini buyruqma-buyruq, ketma-ket o'qiydi, ularning mazmunini yoritadi va ko'rsatilgan amallarni bajaradi.

1.1.3. Dasturlash tillari. Translatorlar. Dastlab birinchi avlod EHMlari uchun dastur mashina tilida tuzilar edi. *Mashina tili* aniq amallarni sonli ko'rinishda kodlash qoidalariga olib kelishdan iborat edi. Keyingi avlod EHMlarida dastur tuzilayotganda mashina kodlari bilan ish olib borish zaruriyati bo'lmaydi, chunki dastur dasturlash tillarining birontasida tuziladi. Bunday dastur matnlari bajarish uchun yaroqli bo'lishi uchun ularni mashina kodlariga aylantirish kerak. Mana shunday vaqtda *translatorlar* deb ataluvchi maxsus standart dasturlardan foydalaniladi. Translator algoritmik tilda

yozilgan dasturni ko‘rsatmalar ketma-ketligiga — mashina kodiga aylantiradi. Shaxsiy kompyuterning markaziy qurilmasi protsessor tezkor (operativ) xotirada yozilgan bu ko‘rsatmalar ketma-ketligini bajaradi.

Ishlash usuliga ko‘ra, translatorlar kompilatorlarga va interpretatorlarga bo‘linadi. *Kompilator*ning interpretatordan farqi shuki, kompilator foydalanuvchi yozgan dasturni EHM uchun tushunarli ko‘rinishga o‘tkazadi (u ichki ko‘rinish deyiladi), so‘ngra bu ko‘rinishdagi dastur bajariladi. *Interpretator* esa har bir ko‘rsatmani ichki ko‘rinishga o‘tkazib bajaradi. Paskal tilining translatorlari uning boshlang‘ich ko‘rinishlaridan so‘nggi *Turbo Pascal 7.0* versiyasigacha kompilirlovchi tamoyil asosida ishlaydi. Mashina tili *quyi darajadagi dasturlash tili* hisoblanib, mashinaga mo‘ljallangan tillar sinfiga kiradi. Bu tillarning asosida aniq bir hisoblash mashinasining buyruqlar tizimi yotadi. Quyi darajadagi tillarga, shuningdek, assembler, makroassembler va mashina tillari kiradi.

Hisoblash texnikasining rivojlanishi masalalarning xususiyatlariga butunlay mo‘ljallangan va aniq bir mashinaga bog‘liq bo‘lmagan *yuqori darajadagi dasturlash algoritmik tillari*ning paydo bo‘lishi hamda ularning rivojlanishiga olib keldi. Mazkur tillarning imlosi, so‘z boyligi dastur bilan ishlaydigan insonga (dasturchiga) ham, kompyuterga ham bir xilda qulay qilib tanlanishi kerak. Yuqori saviyadagi dasturlash tillariga algol, fortran, PL/1, Paskal, Simula va boshqa juda ko‘p tillar misol bo‘la oladi. Kompyuter dasturda beriladigan buyruqlar ketma-ketligini oson talqin qilishi va bajara olishi kerak. Demak, dasturlash tilini inson va mashinaning muloqot vositasi, deb hisoblash mumkin.

Dasturlash tili quyidagi afzalliklarga ega:

- u jonli tilimizga o‘xshash bo‘lib, uni o‘rganish oson;
- bu tilda yozilgan dastur mashina tilidagidan qisqaroq bo‘ladi;
- dastur yozishga kamroq vaqt sarflanadi va kam xatolikka yo‘l qo‘yiladi;
- yozilgan dasturni ixtiyoriy dasturchi o‘qiy oladi;
- dasturlash tili mashina turiga bog‘liq emas.

Demak, dasturlash (yuqori saviyadagi) tilida dastur tuzish qulayroq, osonroq va buning uchun mashina (quyi saviyadagi) tilini bilish shart emas ekan.

Hozirgi paytda dasturlash tillarining soni juda ko‘payib ketmoqda. Lekin, shuni aytish kerakki, har qanday dasturlash tili o‘zining darajasi va qo‘llash sohasiga ega. Ba’zi bir tillar bir necha xil soha masalalarini yechishda ishlatiladi. Bunday tillar universal tillar, deb ham ataladi. Paskal algoritmik tili ham universal til hisoblanadi. U dasturlashning asosiy tushunchalari va konstruksiyalarini o‘z ichiga olishi bilan birga, boshqa universal dasturlash tillariga qaraganda ancha sodda. Endi bu til bilan tanishishga o‘tamiz.

1.2. Paskal dasturlash tili bilan tanishish

Paskal tili shveysariyalik olim Niklas Virt tomonidan, talabalarga dasturlashni o‘rgatish vositasi sifatida 1969-yilda o‘ylab topilgan edi. Tilning nomi mashhur fransuz matematigi va faylasufi Blez Paskal (1623—1662) sharafiga qo‘yilgan.

Dastlab Paskal tili universitetlarda keng tarqaldi, ma’lum bir vaqtdan keyin har xil turdagi EHMlarda Paskal tili uchun o‘nlab translatorlar ishlab chiqildi. 1981-yilda Paskal tilining xalqaro standarti taklif etildi.

Paskalning 4.0 versiyasidayoq foydalanuvchilar o‘z qo‘l ostilariga, tizimdan chiqmasdan turib katta dasturlar tuzish va yaxshilash imkoniyatlari bo‘lgan, qulay tizim (dasturlar ishlab chiqishning yig‘ma muhiti)ga ega edilar. 5.5 versiyasining paydo bo‘lishi bilan *Turbo Paskal*da obyektli dasturlash imkoniyati ham paydo bo‘ldi. 6.0 versiyasida dastur matnlariga assemblerda yozilgan bo‘laklarni kiritish mumkin bo‘ldi. Bundan tashqari, yig‘ma muhit birmuncha o‘zgardi. Bu versiyaga, shuningdek, boshlang‘ich matnlar paketi (*Turbo Vision*) taklif etildi. Undan foydalanish natijasida tashqi jihatdan Borland firmasining dasturlar ishlab chiqarish yig‘ma muhitiga o‘xshash muloqot tizimlarini tez yaratish mumkin bo‘lib qoldi.

IBM PC turidagi shaxsiy kompyuterlarda ishlatiladigan Borland firmasining mahsuloti — *Turbo Paskal* dasturlash tizimi hozirgi kunga kelib eng keng tarqalgan dasturlash tizimlaridan biriga aylandi. Bunga, bir tomondan Paskal dasturlash tili asosining soddaligi sabab bo‘lgan bo‘lsa, ikkinchi tomondan, tilni mukammallashtirishga ko‘p kuch sarflagan *Turbo Paskal*ning yaratuvchisi Anders Xeylsberg rahbarligidagi Borland xodimlarining mehnati va mahorati sabab bo‘ldi. Til murakkab bo‘lmagan hisoblash

masalalarini yechishga mo‘ljallangan oddiy dasturlar tuzishdan tortib ma’lumotlar bazasini boshqaruvchi murakkab relatsion tizimlarni ishlab chiqarishgacha bo‘lgan har qanday masalani yecha oladigan qudratli zamonaviy professional dasturlash tizimiga aylandi.

*Windows*ning va *Windows* muhitida dasturlar ishlab chiqish uchun *Borland Pascal with Objects* va *Delphi* instrumental vositalarining paydo bo‘lishi yana bir marta *Turbo Paskal*ning bitmas-tuganmas imkoniyatlarga ega ekanligini ko‘rsatdi.

Borland Pascal ham, *Delphi*da ishlatiluvchi *Objects Pascal* ham *Turbo Paskal*ga asoslanishadi va uning g‘oyalarini rivojlantirishadi.

1.3. *Turbo Paskal* dasturlash tizimi

Turbo Paskal dasturlash tizimi ikkita, ma’lum ma’nodagi, mustaqil boshlang‘ichlarning yagona birligidan iborat. Ularga Paskal *dasturlash tili kompilatori* va dastur yaratish samaradorligini oshirishga imkon yaratuvchi instrumental *dastur qobig‘i* kiradi. Aniqlik uchun bundan keyin kompilator tomonidan amalga oshiriluvchi dasturlash tilini *Turbo Paskal tili* deb, dastur qobig‘i yordamida berilayotgan har xil servis xizmatlarini *Turbo Paskal muhiti* deb ataymiz.

Turbo Paskal muhiti tushunchasiga to‘xtalamiz. Dasturlashga yordam beruvchi har xil servis xizmatlar maxsus dasturlar yordamida amalga oshiriladi. Bu dasturlar nimalar qilishi kerak? Dastlab ular *Turbo Paskal*da tuzilgan dastur matnini kiritishga imkon berishi, vaqt-vaqti bilan diskka ishning navbatdagi natijasini yozib borishi, yo‘l qo‘yilishi mumkin bo‘lgan kichik imloviy xatoliklarni topish vositasiga ega bo‘lishi kerak. Dastur xatolarini tuzatish jarayonida, uning to‘g‘ri ishlayotganligini ko‘rib turishi uchun dasturni tez-tez ishga tushirib turishi zarur. Bu va shunga o‘xshash boshqa ishlar jamlovchi muhitda bajariladi.

1.4. *Borland Pascal with Objects 7.0* dastur yaratish muhitining tarkibi

Turbo Paskal tilida dastur yaratish uchun Borland firmasining oldingi mahsulotlari kompilator va standart protsedura hamda funksiyalar kutubxonasidan tuzilgan modullardan iborat edi. Kompilator ikki versiyaga ega, ulardan biri ishlab chiqishning

integral muhitida (TURBO.EXE fayli), ikkinchisi paket rejimda (TPC.EXE fayli) ishlagan. *Borland Pascal with Objects 7.0* kompilatorning yangi mahsuloti oldingilaridan keskin farq qiladi. U ishlab chiqishning uchta integral muhitiga (TURBO.EXE, BP.EXE, BPW.EXE) va ikkita paket versiyalariga (TPC.EXE va BPC.EXE) ega. Ularni guruhlarga quyidagicha ajratish mumkin:

- protsessorning real rejimida MS-DOS boshqaruvida ishlovchi kompilator versiyalari (TURBO.EXE, TPC.EXE);
- protsessorning himoya rejimida MS-DOS boshqaruvida ishlovchi kompilator versiyalari (BP.EXE va BPC.EXE);
- kompilatorning *Windows* boshqaruvida ishlovchi versiyalari (BPW.EXE).

Borland Pascal with Objects 7.0 kompilatori versiyalarining imkoniyatlarini taqqoslash uchun ularning ba'zi bir tavsiflarini keltiramiz.

Protsessorning real rejimida MS-DOS boshqaruvida ishlovchi kompilator versiyalari (TURBO.EXE, TPC.EXE).

1. Kompilator versiyalari:
   a) ishlab chiqishning integral muhitida (IM) ishlovchi kompilator versiyasi (TURBO.EXE);
   b) kompilatorning paketli versiyasi (TPC.EXE).
2. Kompilatorning operatsion ishchi muhiti: protsessorning real rejimidagi MS-DOS.
3. Kompilator chiqish kodi bilan quvvatlanuvchi operatsion muhitlar:
   protsessorning real rejimidagi MS-DOS.
4. EXE., TPU chiqish fayllari turlari.
5. Asosiy modullar kutubxonasi fayllari: TURBO.TPL.

Protsessorning himoya rejimida MS-DOS boshqaruvida ishlovchi kompilator versiyalari (BP.EXE, BPC.EXE).

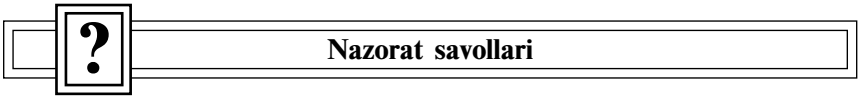
1. Kompilator versiyalari:
   a) ishlab chiqishning integral muhitida (IM) ishlovchi kompilator versiyasi (BP.EXE);
   b) kompilatorning paketli versiyasi (BPC.EXE).
2. Kompilatorning operatsion ishchi muhiti: protsessor himoya rejimida MS-DOS.
3. Kompilator chiqish kodi bilan quvvatlanuvchi operatsion muhitlar:

a) protsessor real rejimidagi MS-DOS;
b) protsessor himoya rejimidagi MS-DOS;
d) *Windows*.

4. Chiqish fayllari turlari: .EXE,.TPU,.TPP,.PPW,.DLL.
5. Asosiy modullar kutubxonasi fayllari: TPP.TPL.

Windows boshqaruvida ishlovchi kompilator versiyalari (BPW.EXE).

1. Kompilator versiyalari:
   a) ishlab chiqishning jamlovchi muhitida (IM) ishlovchi kompilator versiyasi (BPW.EXE);
   b) kompilatorning paketli versiyasi (mavjud emas).
2. Kompilatorning operatsion ishchi muhiti: *Windows*.
3. Kompilator chiqish kodi bilan quvvatlanuvchi operatsion muhitlar:
   a) protsessor real rejimidagi MS-DOS;
   b) protsessor himoya rejimidagi MS-DOS;
   d) *Windows*.
4. Chiqish fayllari turlari: .EXE,.TPU,.TPP,.PPW,.DLL.
5. Asosiy modullar kutubxonasi fayllari: TPW.TPL.



Nazorat savollari

1. Algoritm nima?
2. EHM protsessori va xotirasi qanday funksiyalarni bajaradi, ularning ishlash tamoyili qanday?
3. Dasturlash tillari qanday tillar?
4. Quyi va yuqori saviyadagi tillar o‘rtasida qanday farq bor?
5. Paskal tili qanday dunyoga kelgan?
6. Mashina kodi deb nimaga aytiladi?
7. Nega bevosita mashina kodida dastur tuzish maqsadga muvofiq emas?
8. Translatorlar (interpretatorlar) nima va ular qanday vazifani bajaradi?
9. Jamlovchi muhit deganda nima tushuniladi?
10. *Turbo Paskal* dasturlash tizimi qanday qismlardan iborat?
11. *Borland Pascal with Objects 7.0* dastur yaratish muhiti tarkibi nimalardan iborat?
12. Kompilatorning qanday versiyalari *Borland Pascal with Objects 7.0* paketi tarkibiga kiradi?

II bob. ***TURBO PASKAL*** **TILI ELEMENTLARI**

2.1. Imlo

Turbo Paskal imlosiga harflar, raqamlar, o‘n oltilik raqamlar, maxsus belgilar, bo‘shliqlar va rezerv so‘zlar kiradi.

Harflar—ularga *a* dan *z* gacha va *A* dan *Z* gacha bo‘lgan lotin harflari, shuningdek, ostiga chizish belgisi (ASCII dagi kodi 95) kiradi. *Turbo Paskal*da imloning yozma va bosma harflari o‘rtasida, agar ular ramziy va satrli ifodalarga kirmagan bo‘lsa, farq yo‘q.

Raqamlar—0 dan 9 gacha arab raqamlari.

O‘n oltilik raqamlar — dastlabki 10 ta qiymat 0...9 arab raqamlari bilan, qolgan oltitasi esa *A... F* yoki *a ... f* lotin harflari bilan belgilanadi.

*Maxsus belgilar—Turbo Paskal*da bular quyidagilar:

+ — * / = , ‘ . : ; < > [ ] (0) { } ^ @ $ #

Maxsus belgilarga, shuningdek, quyidagi belgi juftliklari kiradi:

<> <=>= := (* *) (. .)

Dasturda bu belgi juftliklarini, agar ular munosabat amallari yoki izoh belgilari sifatida ishlatilayotgan bo‘lsa, bo‘shliqlar (probel) bilan ajratish mumkin.

(. va .) belgilar mos ravishda [va] o‘rnida ishlatilishi mumkin.

Bo‘shliqlar—til imlosida alohida o‘rin tutadi. Ularga kodlarning 0 dan 32 gacha bo‘lgan sohasidagi ASCII ning ixtiyoriy belgisi kiradi. Bu belgilar identifikatorlar, o‘zgarmaslar, sonlar, rezerv so‘zlarning chekliklari sifatida ishlatiladi. Ketma-ket kelgan bir nechta bo‘shliq bitta bo‘shliq, deb hisoblanadi (oxirgisi satr o‘zgarmaslariga kirmaydi).

*Rezerv so‘zlar—Turbo Paskal*da quyidagi rezerv so‘zlar bor:

and	*end*	*nil*	*shr*
asm	*file*	*not*	*string*

array	*for*	*object*	*then*
begin	*function*	*of*	*to*
case	*goto*	*or*	*type*
const	*if*	*packed*	*unit*
constructor	*implementation*	*procedure*	*until*
destructor	*in*	*program*	*uses*
div	*inline*	*record*	*var*
do	*interface*	*repeat*	*while*
downto	*label*	*set*	*with*
else	*mod*	*shl*	*xor*

Rezerv so‘zlar identifikator sifatida ishlatilishi mumkin emas.

Standart direktivalar dasturdagi ba’zi bir standart e’lonlar bilan bog‘liq. Ularga quyidagilar kiradi:

absolute	*far*	*near*
assembler	*forward*	*private*
external	*Interrupt*	*virtual*

Rezerv so‘zlarga o‘xshab standart direktivalar *Turbo Paskal* tahrir oynasida rangi bilan ajralib turadi, shunga qaramasdan ixtiyoriy standart direktivani oldindan aniqlash, ya’ni bir ismli identifikatorni e’lon qilish mumkin, *private* va *virtual* standart direktivalar obyektlarni e’lon qilish chegarasida amal qilishadi.

2.2. Identifikatorlar

*Turbo Paskal*da identifikatorlar bu yozuvlardagi o‘zgarmaslar, o‘zgaruvchilar, nishonalar (metka), turlar, obyektlar, protseduralar, funksiyalar, modullar, dasturlar va maydonlarning ismlaridir. Identifikatorlar ixtiyoriy uzunlikka ega bo‘lishi mumkin, lekin uning faqat dastlabki 63 belgisigina ma’noga ega bo‘ladi.

Identifikator hamma vaqt harf bilan boshlanadi, undan keyin harflar va raqamlar kelishi mumkin. Ostiga chizish chizig‘i ham harfga kirishi uchun identifikator bu belgi bilan boshlanishi va, hatto, faqat shundan yoki bir nechta ostiga chizish chizig‘idan iborat bo‘lishi mumkin. Bo‘shliqlar va imloning maxsus belgilari identifikatorga kirmaydi.

To‘g‘ri identifikatorlarga misollar:

a	*My Variable*
ALPHA	*My_Variable*
beta	*Stop*
Program Notebook	*lab*_12
*date*_17_*dez*	_1_2_3_

Xato identifikatorlarga misollar:

1 *Program* {raqam bilan boshlangan};
block # 1 {maxsus # belgiga ega};
My Prog {bo‘shliqqa ega};
case {rezerv so‘z};
1_2_3_ {raqam bilan boshlangan}.

2.3. O‘zgarmaslar (konstantalar)

*Turbo Paskal*da o‘zgarmaslarga butun, haqiqiy, o‘n oltilik sonlar, mantiqiy o‘zgarmaslar, belgilar, belgilar satri, to‘plam konstruktorlari va noaniq NIL ko‘rsatkichi nishoni kiradi.

Butun sonlar odatdagi qoidalar bo‘yicha ishorali va ishorasiz yozilishi va —2147483648 dan +2147483648 gacha qiymatlarga ega bo‘lishi mumkin. Agar butun qiymatli o‘zgarmas ko‘rsatilgan chegaradan chetga chiqsa, kompilator xato to‘g‘risida axborot beradi. Bunday o‘zgarmaslar o‘nlik nuqta bilan yozilishi, ya’ni haqiqiy son sifatida aniqlanishi kerak.

Haqiqiy sonlar ishorali va ishorasiz o‘nlik nuqtadan va eksponensial qismdan foydalanib yoziladi. Eksponensial qism *e* yoki *E* belgi bilan boshlanadi, undan keyin «+» yoki «—» ishoralar va o‘nlik tartib keladi. *e (E)* belgi o‘nlik tartibni ko‘rsatilgan va darajali 10 ga ko‘paytirish ma’nosini bildiradi. Misol:

3,14E5 — 3.145 ni 5-darajali o‘nga ko‘paytirish;

—17e-2 — minus 17 ni minus 2-darajali o‘nga ko‘paytirish.

Agar haqiqiy sonning yozilishda o‘nlik nuqta bo‘lsa, nuqta va undan keyin hech bo‘lmaganda bittadan raqam bo‘lishi kerak. Agar eksponensial qismning *e (E)* belgisi ishlatilayotgan bo‘lsa, undan keyin hech bo‘lmaganda o‘ninchi tartibli bitta raqam kelishi kerak.

O‘n oltilik son dollar $ (ASCII da kodi 36) belgisi bilan keladigan o‘n oltilik raqamdan iborat. O‘n oltilik sonning o‘zgarish sohasi $00000000 dan $FFFFFFFF gacha.

Mantiqiy o‘zgarmas — FALSE (yolg‘on) yoki TRUE (haqiqat) so‘zi.

Ramzli o‘zgarmas — bu tuturiq (′) ichida yozilgan shaxsiy kompyuter ixtiyoriy belgisidir:

′X′ — X belgisi;

′M′ — M belgisi.

Agar tuturiqning o‘zini ramziy ko‘rinishda yozish zarur bo‘lsa, u ikkilantiriladi:

′′′′—′ (tuturiq) belgisi.

Ramz belgisini uning ichki kodini (35) ko‘rsatish bilan (koddan oldin # belgi qo‘yiladi) yozish mumkin, masalan,

97 — a belgi;

90 — z belgi;

39 — ′ belgi;

13 — CR belgi.

Satrli o‘zgarmas — tuturiq ichida yozilgan belgilarning (CR-karetkani qaytarish belgisidan boshqa) ixtiyoriy ketma-ketligi. Agar satrda tuturiq belgisining o‘zini ko‘rsatish kerak bo‘lsa, u ikkilantiriladi, masalan:

′Men dasturlashni o‘rganaman′;

′that ′′ s string′.

Belgilar satri bo‘sh bo‘lishi, ya’ni tuturiq ichida hech qanday belgi bo‘lmasligi mumkin. Satrni har biri # belgi bilan boshlanuvchi kerakli belgilar kodidan tuzish mumkin, masalan, #83 #121 #109 #98 #11 #108 satri *′Symbol′* satriga teng kuchli.

Nihoyat, tuturiq ichida kodlar yordamida yozilgan qismlarni navbat bilan yozish mumkin. Shunday yo‘l bilan satrlarga ixtiyoriy boshqaruvchi belgilarni, shu jumladan, CR (13-kod) belgini ham kiritish mumkin, masalan:

#7 ′Xato!′ # 13 ′Ixtiyoriy klavishni bosing ....′ # 7.

To‘plam konstruktori kvadrat qavslarga olingan to‘plam elementlari ro‘yxatidir, masalan,

[1,2,4 .. 7,12]

[blue, red]

[]

[true].

Standart Paskaldan farq qilib, *Turbo Paskal*da o‘zgarmaslarni e’lon qilishda, operandalari avval e’lon qilingan turdoshmas o‘zgarmaslardan, tur va obyekt ismlari bo‘lishi mumkin bo‘lgan ixtiyoriy ifodalardan, shuningdek, ularning quyida keltirilgan funksiyalaridan foydalanishga ruxsat etiladi:

abs	*lo*	*ptr*	*swap*
chr	*odd*	*round*	*trunc*

hi	*ord*	*sizeof*
length	*pred*	*suec*

Masalan,

```
const
        MaxReal = MaxInt div SizeOf (real);
        NumChars = ord ('z') — ord ('a') +1;
        Ln10=2.302585092994;
        Ln10R = 1/Ln10.
```

2.4. Ifodalar

Dasturning bajariluvchi qismini tashkil etuvchi asosiy elementlarga o‘zgarmaslar, o‘zgaruvchilar va funksiyalarga murojaat kiradi. Bu elementlardan har biri o‘zining qiymati bilan tavsiflanadi va berilganlarning qandaydir bir turiga tegishli bo‘ladi. Amal belgilari va qavslar yordamida yangi qiymatlar hosil qilish qoidalarini bayon etuvchi ifodalarni tuzish mumkin.

Birlamchi element, ya’ni o‘zgarmas, o‘zgaruvchi yoki funksiya va murojaat ifodaning xususiy holi bo‘lishi mumkin. Bunday ifodaning qiymati tabiiy hol, element qanday turda bo‘lsa, shu turda bo‘ladi. Umumiy holda ifoda bir nechta elementlardan (operandalardan) va amal belgilaridan tuzilgan bo‘ladi, uning qiymat turi esa operandalar turi va unga qo‘llangan amallar turi bilan bir xil bo‘ladi.

Ifodalarga misollar:

y	*a*>2
21	*not Flag and (a=b)*
*(a+b) * x*	*NIL*
sin(t)	[1,3 .. 7] * *set* 1

2.5. Amallar

*Turbo Paskal*da quyidagi amallar aniqlangan:

unar	*not, @*
multurlikativ	*, /, div, mod, and, shl, shr;*
additiv	+, —, *or, xor;*
munosabatlar	=, <>, <, >, <=, >=, *in.*

Amallar afzalligi ko‘rsatilgan tartibda kamayib boradi, ya’ni unar amallari eng yuqori, munosabat amallari eng quyi afzallikka

ega. Bir xil afzallikka ega bir qancha amallarning bajarish tartibi kompilator tomonidan dastur kodini optimallashtirish shartidan kelib chiqib o‘rnatiladi va ular chapdan o‘ngga tomon bajarilishi shart emas. Mantiqiy ifodalarni hisoblashda bir xil afzallikdagi amallar chapdan o‘ngga tomon hisoblanadi, bunda agar *Turbo Paskal* muhitida OPTIONS/COMPILER/ COMPLETE BOOLEAN EVAL opsiyasining qiymati o‘rnatilgan bo‘lsa, hamma munosabat amallari, agar opsiya o‘rnatilmagan bo‘lsa, faqat natijani olish uchun yetarli bo‘lganlarigina hisoblanadi.

Ismi bo‘yicha uzatiladigan global o‘zgaruvchilari yoki parametrlari o‘zgaradigan funksiyalar bilan munosabat amallarini ishlatganda, bu holni hisobga olish zarur, masalan,

```
Function Addi (var x: Integer): Integer;
begin {Addi}
      inc (x);
      Addi:=x
end {Addi};
var
     a,b: integer;
begin {main}
     if (a>b) or (Addi (a)>100) then b:=a;
  .....
```

Bu bo‘lakni bajarishda A o‘zgaruvchi qiymati opsiyaga bog‘liq: agar opsiya faollashtirilgan bo‘lsa, A ning qiymati hamma vaqt, faollashtirilmagan bo‘lsa, faqat A<=B holda 1 ga oshiriladi.

Har xil turdagi operandalar bilan amallardan foydalanish qoidalari 2.1-jadvalda keltirilgan.

2.1-jadval

Amal	Faoliyat	Operanda turi	Natija turi
not	Inkor	Mantiqiy	Mantiqiy
not	Inkor	Ixtiyoriy butun	Operanda turi
@	Adres	Ixtiyoriy	Ko‘rsatkich
*	Ko‘paytirish	Ixtiyoriy butun	Eng kichik butun
*	Ko‘paytirish	Ixtiyoriy haqiqiy	Extendet
*	To‘plam kesishuvi	To‘plamli	To‘plamli
/	Bo‘lish	Ixtiyoriy haqiqiy	Extendet
div	Butun sonli bo‘lish	Ixtiyoriy butun	Eng kichik butun

mod	Bo‘lishdan qoldiq	Ixtiyoriy butun	Eng kichik butun
and	Mantiqiy BA	Mantiqiy	Mantiqiy
and	Mantiqiy BA	Ixtiyoriy butun	Eng kichik butun
Shl	Chapga siljish	Ixtiyoriy butun	Eng kichik butun
Shr	O‘ngga siljish	Ixtiyoriy butun	Eng kichik butun
+	Qo‘shish	Ixtiyoriy butun	Eng kichik butun
+	Qo‘shish	Ixtiyoriy haqiqiy	Extendet
+	To‘plamlarni birlashtirish	To‘plamli	To‘plamli
+	Satrlarni ulash	Satrli	Satrli
—	Ayirish	Ixtiyoriy butun	Eng kichik butun
—	Ayirish	Ixtiyoriy haqiqiy	Extendet
or	Mantiqiy YOKI	Mantiqiy	Mantiqiy
or	Mantiqiy YOKI	Ixtiyoriy butun	Eng kichik butun
=	Teng	Ixtiyoriy oddiy yoki satrli	Mantiqiy
<>	Tengmas	Ixtiyoriy oddiy yoki satrli	Mantiqiy
<	Kichik	Mantiqiy	Mantiqiy
<=	Kichik yoki teng	Mantiqiy	Mantiqiy
>	Katta	Mantiqiy	Mantiqiy
>=	Katta yoki teng	Mantiqiy	Mantiqiy

Haqiqiy tur bilan amal bajarishda operandalardan bittasi ixtiyoriy butun turdagi qiymat bo‘lishi mumkin. Amal jadvalda ko‘rsatilgan EXTENDED turi natijasiga faqat *Turbo Paskal* muhitida o‘rnatilgan arifmetik soprotsessor yoki uning emulatsiyasiga tayangan kod generatsiyasi rejimi uchun ega bo‘ladi. Agar bu rejim o‘rnatilmagan bo‘lsa, natija REAL turidagi qiymatga ega bo‘ladi.

Unar @ amal ixtiyoriy turdagi operandaga qo‘llaniladi va operanda adresiga ega bo‘lgan POINTER turidagi natijani qaytaradi. Masalan, quyidagilar berilgan bo‘lsin:

```
type
        TwoChar = array [1...2] of char;
var
        Int : integer;
        TwoCharPtr :^  TwoChar.
```

Unda:

TwoCharPtr: @Int.

Operatori, *TwoCharPtr* da ikki belgi to‘plami sifatida tushuntiriladigan butun sonli INT o‘zgaruvchi adresining saqlanishiga olib keladi.

Shuning uchun, masalan,

if TwoCharPtr ^ [1] = ′c′ then ...

operator bo‘lishi mumkin.

Agar @ amal protsedura, funksiya yoki obyektdagi usulga qo‘llanilsa, bu protsedura (funksiya, usul)ga kirish nuqtasining adresi uning natijasi bo‘ladi. Bu adresni faqat assemblerda yozilgan ichki dasturda yoki *INLINE* bo‘laklarida ishlatish mumkin.

*Turbo Paskal*da quyidagi mantiqiy amallar aniqlangan:

not — mantiqiy inkor;

and — mantiqiy VA;

or — mantiqiy YOKI;

xor — faqat YOKI.

Mantiqiy amallar butun va mantiqiy turdagi operandalarga qo‘llaniladi. Agar operandalar — butun sonlar bo‘lsa, mantiqiy amal natijasi ham, bitlari (ikkilamchi razryadlari) 2.2-jadvalda ko‘rsatilgan qoidalar bo‘yicha operanda bitlaridan tashkil topgan, butun son bo‘ladi.

2.2-jadval

INTEGER turidagi berilganlar ustida mantiqiy amallar (xonalar bo‘yicha)					
1-operanda	2-operanda	not	and	or	xor
1	—	0	—	—	—
0	—	1	—	—	—
0	0	—	0	0	0
0	1	—	0	1	1
1	0	—	0	1	1
1	1	—	1	1	0

*Turbo Paskal*dagi mantiqiy amallarga, odatda, ikkita butun son ustida bajariladigan siljish amallari ham kiradi:

i sh j — *i* tarkibini *j* xona chapga siljitish, bo‘shagan kichik razryadlar nollar bilan to‘ldiriladi;

i shr j — *i* tarkibini *j* xona o‘ngga siljitish; bo‘shagan razryadlar nollar bilan to‘ldiriladi.

Bu amallarda *i* va *j* ixtiyoriy butun turdagi ifodalar.

Quyida keltirilgan misol dasturi yordamida ikkita butun songa qo‘llanilgan mantiqiy amallar natijasini ekranga chiqarish mumkin.

Dastur ikki butun sonni kiritadi va ularga mantiqiy amallarni qo‘llash natijasini chop etadi. Dasturdan chiqish uchun *Ctrl-Z* ni kiritish va *Enter*ni bosish kerak.

```
var
        n,m:integer;
begin
        while not EOF do
                begin
                        write ('n,m=');
                        Readln (n,m);
                        writeln ('not=', not n,' ', not m);
                        writeln ('and=', n and m);
                        writeln ('or=', n xor m);
                        writeln ('shl=', n shl m);
                        writeln ('shr=', n shr m);
                end
end.
```

Mantiqiy ma’lumotlar ustida bajarilgan mantiqiy amallar 2.3-jadvalda ko‘rsatilgan qoidalar bo‘yicha mantiqiy turdagi natijalarni beradi.

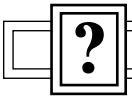
2.3-jadval

Boolean turidagi ma’lumotlar ustida mantiqiy amallar					
1-operanda	2-operanda	not	and	or	xor
True	—	False	—	—	—
False	—	True	—	—	—
False	False	—	False	False	False
False	True	—	False	True	True
True	False	—	False	True	True
True	True	—	True	True	False

IN munosabat amali ikkita operandaga qo‘llaniladi. Birinchi (o‘ng) operanda ixtiyoriy tartib turidagi ifoda, ikkinchisi — o‘sha

turdagi elementlardan iborat to‘plam yoki to‘plam turidagi identifikator bo‘lishi mumkin. Agar chap operanda to‘plamga tegishli bo‘lsa, amal *True*ni beradi, masalan,

```
var
        c:char;
type
        digit = set of '0' .. '9';
begin
        if c in digit then .....
```



Nazorat savollari

1. *Turbo Paskal* imlosiga nimalar kiradi?
2. Rezerv so‘zlar qanday so‘zlar?
3. Standart direktivalarga nimalar kiradi?
4. Identifikatorlar nima?
5. Identifikatorga qo‘yilgan talablar nimalardan iborat?
6. *Turbo Paskal*da o‘zgarmaslarga nimalar kiradi?
7. Ramzli o‘zgarmaslar qanday o‘zgarmaslar?
8. Satrli o‘zgarmas nima?
9. *Turbo Paskal*da o‘zgarmaslarni e’lon qilish standart Paskaldan nimasi bilan farq qiladi?
10. Ifodalarga nimalar kiradi?
11. *Turbo Paskal*da qanday amallar aniqlangan?
12. *Turbo Paskal*da amal afzalliklari tartibi qanday?
13. *Turbo Paskal*da har xil turdagi operandalar bilan amallar bajarishning o‘ziga xos qoidalari.
14. *Turbo Paskal*da qanday mantiqiy amallar bor va ular qanday turdagi operandalarga qo‘llaniladi?

III bob. ***TURBO PASKAL*** **DASTURLASH TILI. MA'LUMOTLAR TURLARI**

3.1. *Turbo Paskal* ma'lumotlar turlari bilan tanishish

Odatda, *Turbo Paskal*da dastur tarkibi quyidagi ko'rinishga ega bo'ladi:

Program <dastur nomi>;
    {Bayonlar bo'limi};
begin;
    {operatorlar bo'limi};
end.

Program, begin va *end* so'zlari dasturni ikki — bayon va operator qismlariga ajratadi. Har qanday dastur uchun bunday tasnif zarurdir. Bu tilning qat'iy talablaridan kelib chiqadi: bajariluvchi operatorlarda ishtirok etuvchi har qanday standartmas identifikator oldindan bayonlar bo'limida tavsiflanishi kerak. (Standart identifikatorlar oldindan e'lon qilingan obyektlar bilan bog'liq va *Turbo Paskal* standart kutubxonasiga kiradi. Masalan, *WriteLn* identifikatori shunday identifikator. Standart identifikatorlar, agar ular dasturga kirsa, bayon etilmaydi.) *Turbo Paskal*da identifikatorlarning oldindan bunday bayon etilishi, dasturni ba'zi bir xatoliklarning (identifikatorlarda belgilarning to'g'ri, to'la yozilishi) bo'lishidan himoya qiladi, uning ishonchliligini oshiradi.

Identifikatorni bayon etish — identifikator bilan bog'liq bo'lgan dastur obyekti turini ko'rsatish demakdir. Tur tushunchasi *Turbo Paskal*dagi asosiy tushunchalardan biridir. Hozircha, tilning quyida keltiriladigan xususiyatlarini tushuntirish uchun har xil turlarni batafsil ko'rib o'tirmaymiz va turga, birinchidan, kompyuter uchun obyektni ichki ifodalash usuli, ikkinchidan, uning ustidan bajarish mumkin bo'lgan amallarni aniqlovchi, deb izoh beramiz. Tur ko'rinishlarini VI (oddiy turi) va IX (murakkab turi) boblarida batafsil ko'ramiz.

Bundan keyingi dasturlarimizda ma'lumotlarning quyidagi turlari kerak bo'ladi:

- INTEGER — butun sonli ma'lumotlar, ichki tasavvurda 2 baytni egallaydi, qabul qilish qiymatlari sohasi — 32768 dan +32768 gacha, ma'lumotlar aniq ifodalanadi;
- REAL — haqiqiy ma'lumotlar, 6 baytni egallaydi, modulning qiymatlar qabul qilish sohasi — 2.9 E — 39 dan 1.7 E + 38 gacha, ma'lumotlarni 11 ... 12 xonalargacha aniqlikda ifodalaydi;
- CHAR — belgi, 1 baytni egallaydi;
- STRING — belgilar satri, MAX+1 baytni egallaydi, bu yerda MAX — satrdagi belgilarning maksimal soni;
- BOOLEAN — mantiqiy tur, 1 baytni egallaydi va ikkita FALSE (yolg'on) hamda TRUE (haqiqat) qiymatlariga ega bo'ladi.

O'zgarmas turi uning qiymatini yozish usuli bilan aniqlanadi. Masalan,

```
const
        A1 = 25;
        A2 = 2.27;
        A3 = 'C';
        A4 = '2.27';
        A5 = FALSE.
```

Dasturning bu bo'lagini tahlil qilishda kompilator birinchi o'zgarmasni INTEGER, ikkinchisini — REAL, uchinchisini — CHAR, to'rtinchisini — STRING va oxirgisini BOOLEAN turiga kiritadi. O'zgarmasni INTEGER yoki REAL turga kiritish belgisi son qiymatidagi o'nlik nuqtaning bor yoki yo'qligidir. A2 va A4 o'zgarmaslar har xil turlarga kiradi: A2 — REAL (o'zgarmasda o'nlik nuqta bor), A4 — STRING (o'zgarmas tuturiq ichida yozilgan). Tuturiq ichida yolg'iz yozilgan C o'zgarmas qiymatini kompilator CHAR turiga, bir necha belgini esa STRING turiga kiritadi.

O'zgarmasdan farq qilib, o'zgaruvchi dasturda hisob borishi bilan o'z qiymatini o'zgartiradi. O'zgaruvchilarni bayon etishda identifikatordan keyin ikki nuqta va tur nomi qo'yiladi. Bir nechta bir xil turli o'zgaruvchilarni bitta ro'yxatga oralariga vergul qo'yib birlashtirish mumkin. O'zgaruvchilarni bayon etish bo'limi boshida *VAR (VARiables* — o'zgaruvchilar) rezerv so'zi turishi kerak.

Masalan,

```
VAR
        alpha : real;
        a,b,c,d : char;
        text 1 : string [15];
        text 2 : string;
        gamma : boolean.
```

Yuqorida aytganimizdek, ma'lumotlar turi tegishli o'zgaruvchilarning ichki tasvirlash uzunligini aniqlaydi. Xususan, STRING (belgilar satri) turidagi o'zgaruvchilarni ichki tasvirlash uzunligi satrni tashkil qilishi mumkin bo'lgan belgilarning maksimal soniga bog'liq. Yuqorida keltirilgan misolda *text 1* o'zgaruvchi maksimal uzunligini (15 belgi) ko'rsatish bilan bayon etilgan, *text 2* o'zgaruvchining maksimal uzunligi esa ko'rsatilmagan, kompilator unga *Turbo Paskal*da, chegaraviy — 255 belgidan iborat, maksimal uzunlikni o'rnatadi.

Yana bir murakkab bo'lmagan dasturni ko'ramiz. Klaviaturadan ikki butun sonni kiritish, birinchi sonni ikkinchisiga bo'lish va hosil bo'lgan natijani ekranga chiqarish kerak:

```
Program Input_Output;
{Dastur ikkita butun sonni kiritadi va 1-sonni 2-songa
bo'lish natijasini chiqaradi}
var
        n1, n2: Integer;  {n1 va n2 — kiritiladigan butun sonlar}
        x : Real;     {x bo'linma}
begin
        write ('n1='); {n1 kiritilish to'g'risidagi axborot}
        readln (n1); {n1 kiritiladi}
        write ('n2='); {n2 kiritilish to'g'risidagi axborot}
        Readln (n2); {n2 kiritiladi}
        x:=n1/n2; {natija topiladi}
        writeln ('n1/n2=', x); {natija chiqariladi}
end.
```

Dasturda { va } figurali qavslar ichida izohlar ishlatilgan. Izohlar dasturning ixtiyoriy joyida { va } yoki (* *) belgilar ichida yozilishi mumkin. Misol,

```
{Bu izoh}
(*Bu ham izoh*)
```

Agar dasturdan qandaydir bir matn bo'lagini vaqtincha o'chirish kerak bo'lib qolsa, uni izoh belgilari ichiga olish mumkin.

Endi ma'lumotlarni kiritishga to'xtaymiz:

Write (..);
Readln (..).

Operatorlar jufti quyidagicha ishlaydi. Avval *Write* operatori satrni ekranga chiqaradi va kursorni hozirgina chiqarilgan matn satrining oxirida qoldiradi.

WriteLn (Text).

WriteLn operatori matn kiritilgach, satrni o‘tkazib yuborib, kursorni ekranning navbatdagi satr boshiga o‘rnatadi. *WriteLn* va *Write* protseduralar ishlaridagi asosiy farq ana shundan iborat.

Keyin *ReadLn* operatori bo‘yicha ma’lumotlarni kiritish protsedurasi chaqiriladi va dastur kiritishni kutib to‘xtaydi. Bu vaqtda klaviaturadan kerakli sonni terish va *Enter*ni bosish kerak. Shu zahotiyoq dastur ishini davom ettiradi: kiritilgan sonni tahlil qiladi, navbatdagi sonni kiritishga yoki natijani hisoblashga o‘tadi.

Kiritilgan sonlar nisbatini hisoblash uchun *Turbo Paskal*da asosiy operatorlaridan biri bo‘lgan o‘zlashtirish (hisoblash) operatori ishlatiladi. Uning chap qismida o‘zgaruvchi nomi, o‘ngda o‘zgaruvchi bilan bir xil turdagi ifoda yoziladi. O‘ng va chap qismlar «:=» belgilar bilan bog‘lanadi (o‘zgarmaslarni bayon qilishda «=» belgi ishlatiladi).

Matematik nuqtayi nazardan *x=x+1* ifoda ma’nosiz. *x:=x+1* ifoda esa *x* dagi qiymatga *1* sonini qo‘shib, natijani yana *x* da qoldirish kerakligini bildiradi.

Natijalarni chiqarish

writeln (′n1/n2=′, x).

operatorida parametrlardan bittasi *(′n1/n2=′)* belgilar satri o‘zgarmas turidadir. O‘zgarmaslarni esa o‘zgaruvchilardan farq qilib, bayonlar bo‘limida e’lon qilish shart emas, chunki kompilator tomonidan ularning turi o‘zgarmasning yozilishi shaklida oson aniqlanadi. Shuni hisobga olib, shaxsiy kompyuter ekraniga: «Men Paskalda dastur tuzaman», degan axborotni chiqaruvchi oddiy dasturni qisqa qilib quyidagicha yozish mumkin:

```
begin
        WriteLn (′Men Paskalda dastur tuzaman′);
end.
```

3.2. Turlarni o‘zgartirish va ular ustida amallar bajarish

Yuqorida bayon etilganidek, o‘zgaruvchi turi uni ichki tasvirlash uzunligini o‘rnatishgagina emas, balki ular ustida dasturda bajarila-

digan amallarni nazorat qilishga ham imkon beradi. Dasturni kompilatsiya qilish bosqichidayoq o‘zgaruvchilarning ishlatilishini nazorat qila olishi *Turbo Paskal*ning boshqa dasturlash tillaridan muhim afzalligidir. Nazorat natijasida turlar avtomatik tarzda o‘zgartiriladi. *Turbo Paskal*da turlarni noaniq o‘zgartirish deyarli mumkin emas. Bundan REAL turidagi ifodalarda ishlatish mumkin bo‘lgan INTEGER turidagi o‘zgarmas va o‘zgaruvchilar mustasno. Agar, masalan, *x* va *y* o‘zgaruvchilar quyidagicha bayon etilgan bo‘lsa:

```
var
        x: Integer;
        y: Real;
        y:=x+2.
```

y:=x+2; operatorda o‘zlashtirish belgisidan o‘ngda butun ifoda, chapdan esa haqiqiy o‘zgaruvchi turgan bo‘lsa ham, sintaksis nuqtayi nazardan to‘g‘ri bo‘ladi.

Kompilator zaruriy o‘zgartirishlarni avtomatik ravishda bajaradi. Bir vaqtning o‘zida x:=2.0; operator xato bo‘ladi, chunki *Turbo Paskal*da REAL (2.0 o‘zgarmas o‘nlik nuqtaga ega, demak, u haqiqiy) turni INTEGER turiga avtomatik ravishda o‘zgartirishi mumkin emas, lekin bu ma’lumotlar o‘zgartirish vositasining yo‘qligini bildirmaydi. Ular bor, lekin ularni aniq qilib ifodalash kerak. Ularga keyinchalik (VI va IX boblarda) to‘xtalamiz. Ma’lumotlarni o‘zgartirish uchun *Turbo Paskal*da o‘rnatilgan (встроенные) funksiyalar mavjud. Ular parametr sifatida bir turdagi qiymatlarni qabul qilib, natijani boshqa turdagi qiymat ko‘rinishida uzatadi. Masalan, REALni ROUND eng yaqin butun songa yaxlitlaydi, TRUNC esa REALni, kasr qismini tashlab, butun songa aylantiradi.

Masalan,

```
        x: = y/x;
```

operator xato, lekin

```
        x: = round (y/x)
```

to‘g‘ri bo‘ladi (o‘zgaruvchilar yuqorida e’lon qilingan).

*Turbo Paskal*da funksiya tushunchasi protsedura tushunchasiga yaqin. Protsedura kabi funksiya ham ismi bilan chaqiriladi va *Turbo Paskal*ning ixtiyoriy sondagi operatorlaridan va hatto, ichki protsedura hamda funksiyalaridan iborat bo‘lishi mumkin. Funksiyaning protseduradan farqi shuki, funksiya o‘zining xususiy qiymatlariga ega va, demak, o‘zgaruvchilar bilan bir qatorda, tegishli turdagi ifodalarda ishlatilishi mumkin.

CHAR (belgi) turidagi berilganlarni butun songa o‘tkazish uchun ORD funksiya, INTEGERdan CHARga o‘tish uchun CHR funksiya ishlatiladi.

Quyidagi murakkab bo‘lmagan dastur yordamida ixtiyoriy belgining ichki kodini bilish mumkin:

```
Program  Code_of_Char;
{Dastur klaviaturadan belgini o‘qiydi va ekranga bu bel-
gini va unga tegishli bo‘lgan ichki kodni chiqaradi}
var
        ch:Char;            {bu o‘zgaruvchida belgi o‘qiladi}
begin
        Write ('ixtiyoriy belgini kiriting:');
        ReadLn (ch);        {bitta belgi o‘qiladi}
        WriteLn (ch, '=', ord (ch));  {u butun songa o‘zgarti-
                                       riladi va ekranga chiqari-
                                       ladi}
end.
```

```
        WriteLn (ch, '=' , ord (ch)),
```

deb chaqirilganda, murojaatning uchinchi parametrida *ord (ch)* funksiyani chaqirish ko‘rsatilgan. Bu til nuqtayi nazaridan *ifoda* bo‘ladi.

Ko‘p hollarda protsedura va funksiyalarni chaqirishda chaqirish parametri sifatida faqat o‘zgaruvchi yoki o‘zgarmaslarni emas, balki ular ishtirokidagi ifodalarni ham ko‘rsatish mumkinligini biz keyinroq (VIII bobda) ko‘ramiz.

Zarur bo‘lishiga qarab, ma’lumotlar turini o‘zgartirishning boshqa funksiyalari bilan ham tanishib boramiz. Endi har xil turdagi ma’lumotlar ustida amallar bajarish bilan tanishamiz.

*Turbo Paskal*da REAL va INTEGER o‘zgaruvchilari ustida, albatta, hamma to‘rt arifmetik amal bor:

\+ — qo‘shish;

— — ayirish;

$\frac{35}{17}$ — ko‘paytirish;

/ — haqiqiy sonli bo‘lish;

div — butun sonli bo‘lish.

*Turbo Paskal*da ikkita bo‘lish amalining mavjudligi, kompilator turlarini almashtirishga tayyorligini ko‘rsatishdir. Masalan, agar Fortran tilida N butun son, X haqiqiy son bo‘lsa,

N = 1/2 va X=1/2

amallardan keyin N = 0 qiymatni, X=0,5 qiymatni o‘zlashtiradi. *Turbo Paskal*da esa bunday ikki ma’nolilik yo‘q: $^1/_2$ ifoda hamma vaqt 0,5 qiymatga ega. Shuning uchun

```
var
        N: Integer;
begin
        N:=1/2;
```

bo‘lishi mumkin emas. Bu hol *Turbo Paskal*da quyidagicha bayon etiladi:

```
var
        X: Real;
begin
        X:=1 div 2;
```

div natijaning kasr qismini tashlaydi.

INTEGER turidagi ma’lumotlar uchun *Turbo Paskal*da yana bitta MOD, butun sonli bo‘lishda qoldiq hosil qilish amali bor. Masalan,

5 *mod* 2 = 1;

31 *mod* 16 = 15;

18 *mod* 3 = 0.

*Turbo Paskal*da darajaga ko‘tarish amali yo‘q, bu hisoblashlarda ba’zi bir noqulayliklarni keltirib chiqaradi. SQR funksiyaning mavjudligi esa bu holatni oz bo‘lsa ham yengillashtirishga yordam beradi, bu funksiya parametr qiymatining kvadratini topishga imkon beradi, bunda natija turi parametr turi bilan aniqlanadi.

*Turbo Paskal*ning kamchiliklaridan yana biri shuki, unda mavhum (kompleks) tur va ular ustida tegishli amallar yo‘q. Umuman, har xil hisoblash jarayonlarini amalga oshirishda *Turbo Paskal* ba’zi bir dasturlash tillariga, jumladan, Fortranga yon bosadi. Xususan, unda o‘rnatilgan (встроенные) funksiyalar to‘plami ancha kam.

Butun sonlar bilan ishlashda ikkita protsedura foydali bo‘lishi mumkin:

DEC (X [,N]) — (kvadrat qavslar ichida shart bo‘lmagan parametrlar yoziladi) — X o‘zgaruvchi qiymatini N ifoda qiymatiga kamaytiradi (agar N berilmagan bo‘lsa, 1 ga); bu yerda X o‘zgaruvchi va N ifoda qiymatlari INTEGER;

INC (X [,N]) — X ning qiymatini N ga (agar u berilmagan bo‘lsa, 1 ga) oshiradi.

Belgilar va satrlar ustida yagona amal — ikki satrni ulash — aniqlangan. Masalan,

```
var
        st:string;
begin
        st:= 'Turbo' +  '—'+ 'Paskal';
        writeln (st);
end.
```

Bu dastur *Turbo Paskal* degan satrni chop etadi.

Satrlar va belgilar ustida qolgan amallar o‘rnatilgan protsedura va funksiyalar yordamida amalga oshiriladi (VI va IX boblar).

Endi munosabat va mantiqiy amallarga to‘xtalamiz.

REAL, INTEGER, CHAR, STRING turidagi ma’lumotlar ustida quyidagi munosabat (taqqoslash) amallari aniqlangan:

= — teng;

<> — tengmas;

< — kichik;

> — katta;

<= — kichik yoki teng;

>= — katta yoki teng.

Taqqoslash amallarida, albatta, bir turli operandalar ishtirok etishi kerak. Bundan, bir-biri bilan taqqoslanishi mumkin bo‘lgan, REAL va INTEGER mustasno. BOOLEAN turidagi operanda ixtiyoriy operandalar bilan munosabat amallariga kirishadi.

Ikki satrni taqqoslash quyidagi tarzda amalga oshiriladi. Satr belgilari boshqasi bilan juft-juft qilib taqqoslanadi: birinchi satrning 1-belgisi, ikkinchi satrning 1-belgisi bilan, birinchi satr ikkinchi belgisi, ikkinchi satr ikkinchi belgisi va h.k. Belgilar ularning ichki tasvirlanish kodlari bilan (VI va IX boblar) taqqoslanadi. Agar bir satr ikkinchisidan qisqa bo‘lsa, yetishmagan belgilar nollar bilan to‘ldiriladi. Birinchi bir-biri bilan mos kelmagan belgilar juftligining nisbati ikki satr nisbati deb qabul qilinadi.

BOOLEAN turidagi ma’lumotlarni taqqoslashda *Turbo Paskal*ning ichki kelishuvi hisobga olinadi, unga ko‘ra FALSE nolinchi bayt, TRUE esa kichik razryaddagi bir raqamli baytdir.

ORD funksiya faqat belgilarnigina emas, balki mantiqiy miqdorlarni ham butunga keltiradi, shuning uchun:

ord (false) = 0,

ord (true) = 1.

*Turbo Paskal*da quyidagi mantiqiy amallar aniqlangan:

not —mantiqiy YO‘Q; *or* — mantiqiy YOKI;

and — mantiqiy VA; *xor* — yo‘qotuvchi YOKI.

Mantiqiy amallar butun va mantiqiy turdagi operandalarga qo‘llaniladi. Agar operandalar butun sonlar bo‘lsa, mantiqiy amal natijasi ham butun son bo‘ladi (batafsil bu haqda VI va IX boblarda fikr yuritiladi). Mantiqiy ma’lumotlar ustida mantiqiy amallar mantiqiy turdagi natijani beradi.

Ixtiyoriy turdagi ifodani hisoblashda hisoblashlar afzalligi qavslar bilan, qavslar bo‘lmaganda esa quyidagi jadvalga ko‘ra (afzalligi pasayib borish tartibida) aniqlanadi:

3.1-jadval

Amallar afzalligi

Afzallik tartibi	Amallar
1	not, @
2	* , / , div, mod, and, shl, shr
3	+, —, or, xor
4	=, <>, >, >=, <, <=, in

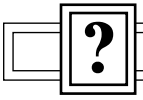
! *Izoh:* @ — adresni hosil qilish, *shl* — chapga siljish, *shr* — o‘ngga siljish va *in* — to‘plamga tegishlilik amallari.

Boshqa tillardan farq qilib, *Turbo Paskal*da dasturlashda mantiqiy amallar, munosabat amallariga qaraganda yuqori afzallikka ega. Shuning uchun murakkab mantiqiy ifodalarda, odatda, qavslar qo‘yish zarur. Agar, masalan, *b* va *c* INTEGER turiga ega bo‘lsa,

a=b and c<d

ifoda sintaksis xatoni ko‘rsatadi, chunki avval *b and c* amal bajariladi, yuqoridagi ifodani quyidagicha yozish to‘g‘ridir:

(a=b) and (c<d).



Nazorat savollari

1. *Turbo Paskal*da dastur, odatda, qanday ko‘rinishda bo‘ladi?
2. *Turbo Paskal* dasturida identifikatorlarni oldindan bayon etilishning qanday afzalliklari bor?
3. *Integer* turdagi ma’lumotlar qanday ma’lumot?
4. *Real* turdagi ma’lumotlar qanday ma’lumot?
5. *Char* turdagi ma’lumotlar qanday ma’lumot?
6. *String* turdagi ma’lumotlar qanday ma’lumot?
7. *Boolean* turdagi ma’lumotlar qanday ma’lumot?
8. O‘zgarmasning turi qanday aniqlanadi?
9. O‘zgaruvchi dasturda qanday bayon etiladi?
10. Dasturda izohlar qanday yoziladi?
11. O‘zlashtirish operatorining mohiyati va ishlatilishi.
12. Kompilator dastur turlari o‘rtasidagi o‘zgartirishlarni qanday olib boradi?
13. «O‘rnatilgan» funksiyalar qanday funksiyalar?
14. CHAR turidagi berilganlar butun songa va teskarisiga qanday o‘tkaziladi?
15. Ma’lumotlar turini o‘zgartirishning yana qanday funksiyalari bor?
16. *Turbo Paskal*da butun sonlar bilan ishlaganda qanday protseduralar mavjud?
17. Har xil turdagi ma’lumotlar ustida qanday munosabat amallari mavjud?
18. Ixtiyoriy turdagi hisoblashlar amallarning qanday afzalliklari bilan bajariladi?
19. *Turbo Paskal*da mantiqiy amallarning afzalligi qanday aniqlanadi?

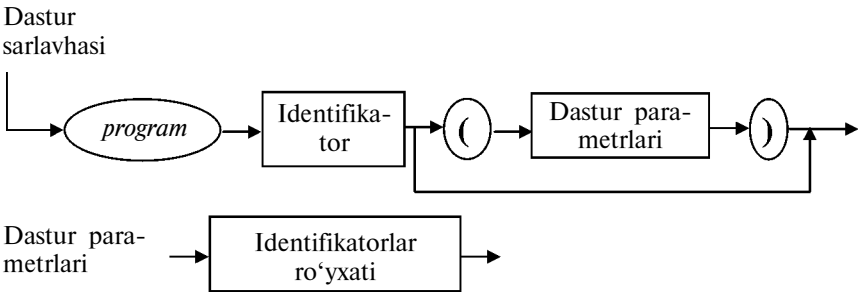
IV bob. ***TURBO PASKAL*** **DASTURI TARKIBI**

Dastur tarkibini norasmiy ravishda quyidagicha ko‘rsatish mumkin:

```
        {I. Dastur sarlavhasi}
Program
        {II. Ishlatiladigan modullarni ko‘rsatish}
uses
        {Ishlatiladigan modullar ro‘yxati}
        {III. Bayonlar bo‘limi}

label           nishonalar bayoni;
const           o‘zgarmaslar bayoni;
type            turlar bayoni;
var             o‘zgaruvchilar bayoni;

procedure  ┐
function   ┘─   protsedura va funksiyalar bayoni;
exports         eksport qilinadigan ismlar bayoni;

        {IV. Operatorlar bo‘limi (operator bloki)}
begin
                operatorlar;
end.
```

4.1. Dastur sarlavhasi bo‘limi

Dastur sarlavhasi, albatta, bo‘lishi shart emas, u, asosan, namoyish maqsadidagina ishlatiladi. Sintaktik nuqtayi nazaridan sarlavha quyidagicha yozilishi kerak:



Misollar:

Program Simple;
Program Print (Output);
Program GetPut (Input, Output);
Program Complex (Input, Output, MyFile).

Sarlavhaning zaruriy emasligidan kelib chiqib, odatda, *Turbo Paskal*da keltirilgan variantlardan faqat birinchisi ishlatiladi.

4.2. Modullarni ko‘rsatish bo‘limi

O‘z vaqtida *Turbo Paskal*ni yaratuvchilar IBM kompyuterlarining hamma (deyarli hamma) quvvatidan foydalanishga imkon beruvchi modullarni yaratdilar, ya’ni kerakli protsedura, funksiya, o‘zgarmaslar, ma’lumotlar turlarini modullar yordamida bayon etishdi. Modul bayonlar bo‘limidagi har xil tashkil etuvchilari va ba’zi bir bajariluvchi operatorlarni o‘z ichiga olgan mustaqil kompilatsiya qilinadigan dastur birligidir. Modullarning muhim xususiyati shundaki, *Turbo Paskal* kompilatori ularning dastur kodini xotiraning alohida segmentiga joylashtiradi. Segmentning maksimal uzunligi 64 Kbaytdan oshmaydi, lekin bir vaqtda ishlatiladigan modullar soni foydalanish mumkin bo‘lgan xotira bilan cheklanadi, bu esa juda yirik dasturlar tuzishga imkon beradi.

Turbo Paskal modullariga quyidagilar kiradi:

SYSTEM — bu modulga Paskalning va *Turbo Paskal*ning bir qator standart protsedura va funksiyalari kiradi. Hamma dasturlar avtomatik *System* modulida ishlay oladi.

DOS — bu modul MS-DOS operatsion tizimining vositalaridan foydalanishga imkon beruvchi protsedura va funksiyalaridan iborat.

Crt — u ekran, klaviatura, IBM kompyuteri dinamikadan foydalanish uchun kerakli protseduralar to‘plamidan iborat.

Graph — mazkur modul har xil grafik adapterlari yordamida kompyuterning grafik imkoniyatlaridan foydalanishga imkon beruvchi dasturlarning katta to‘plamidan iborat.

Printer — printer bilan ishlashni soddalashtiradigan kichkina modul.

Overlay — overley dasturlar yaratishda ishlatiladigan modul.

Graph3 — *Turbo Paskal*ning 3.0 versiyasi uchun grafikli dasturlarning to‘la to‘plamiga ega modul.

Turbo3 — *Graph3* kabi 3.0 versiya uchun yaratilgan modul.

Foydalaniladigan modullarni ko‘rsatish bo‘limi rezerv *uses* so‘zi bilan boshlanadi. Dasturda *uses* so‘zining bo‘lishi shart emas. Agar dasturda *System* modulidan boshqa *Turbo Paskal*ning standart modullarida yoki foydalanuvchi yaratgan modulda aniqlangan o‘zgarmaslar, turlar, o‘zgaruvchilar, protsedura va funksiyalar ishlatilsa, *uses* so‘zi yoziladi.

Misol,

uses Crt, Graph;

uses so‘zi har bir alohida dasturda faqat bir marta, dastur sarlavhasidan keyin yozilishi kerak.

SYSTEM standart moduli hamma vaqt sukut tarzida ishlatiladi va uni *uses* so‘zida ko‘rsatish kerak emas. Bu modul faylli kiritish-chiqarish, satrlarni ishlab chiqish, suzuvchi vergul bilan amallar bajarish, xotirani dinamik taqsimlash vositalarini quvvatlaydi. *Turbo Paskal*ning DOS, *Crt, Graph* va shunga o‘xshash boshqa standart modullari avtomatik ulanmaydi, ularni ishlatish zaruriyati bo‘lganda ular, albatta, *uses* so‘zida ko‘rsatiladi.

4.3. Bayonlar bo‘limi

Bayonlar bo‘limi ham oldingi bo‘limlar kabi majburiy emas. Lekin bu bo‘limni ishlatmasdan turib, faqat eng oddiy dasturlarni yozish mumkin.

O‘zgarmaslar (*const*), turlar (*type*), o‘zgaruvchilar (*var*), protseduralar (*procedure*), funksiyalar (*function*) va eksport (*exports*) ichki bo‘limlari, bayonlar bo‘limi doirasida, ixtiyoriy tartibda bir necha marta takrorlanishi mumkin. Faqat quyidagi qoidaning bajarilishiga rioya qilish kerak: qandaydir bir B element

(o‘zgarmas, tur, o‘zgaruvchi, protsedura, funksiya, eksport ro‘yxati) bayonida A element (o‘zgarmas, tur va h.k.) ishlatilsa, A element B elementdan oldin bayon etilishi kerak.

Eksport (*exports*) ichki bo‘limining bayoni faqat protsessorning himoya rejimini ishlatuvchi kompilator versiyalarida (ya’ni BP.EXE, BPC.EXE, BPW.EXE) bo‘ladi.

Ichki bir xil bo‘limlar bayonining bir necha marta ishlatilishi keltirilgan qoida talabini bajarish uchun zarur bo‘lgan hollarda, shuningdek, bayonlar tasnifini yaxshilash va dastur o‘qilishini oshirish maqsadida ishlatiladi.

Masalan,

```
type        {1-ichki masalani yechish}
var         {uchun}
procedure   {bayon}
(…….…)
label       {2-ichki masalani yechish}
const       {uchun}
var         {bayon}
(…….…)
(…….……..… )
const       {N-ichki masalani yechish}
type        {uchun}
var         {bayon}
function
```

4.3.1. Nishona (metka)lar bayoni. Nishona bo‘limi bayonining umumiy ko‘rinishi:

label vergul bilan bir-biridan ajratilgan nishonalar.

Nishona dasturning ixtiyoriy operatoridan oldin keladi va undan (:) belgi bilan ajratiladi. Nishonalar o‘tish (*goto*) operatori bilan birgalikda ishlatiladi.

Misol,

```
label 1, a;
        .....
        goto 1;
        .....
1:a:=1;
```

goto a;

.....

a: end.

O‘tish (*goto*) operatori bilan nishonalarga murojaat dasturda sodir bo‘lmasa, nishonalarni bayon etish, xato hisoblanmasa ham, ma’nosizdir. Nishonalarni va *goto* operatorlarini ishlatish, ko‘p hollarda tasnifiy dasturlash tamoyiliga qarama-qarshi ekanligini aytib o‘tamiz, shuning uchun bu konstruksiyalarni dasturda ishlatmaslik tavsiya etiladi.

4.3.2. Turlar bayoni. Turlar bayonining umumiy ko‘rinishi:

type

tur identifikatori = tur bayoni.

Turbo Paskal turlari to‘plamini ikki guruhga bo‘lish mumkin:

- standart turlar;
- foydalanuvchi tomonidan aniqlangan turlar.

*Turbo Paskal*ning *standart turlari*ga quyidagilar kiradi:

- butun turlar guruhi (*Shortint, Integer, Longint, Byte, Word*);
- haqiqiy turlar guruhi (*Single, Real, Double, Extended, Comp*);
- bulev turlar guruhi (*Boolean, ByteBool, WordBool, LongBool*);
- ramzli tur (*Char*);
- satrli tur (*String, Pchar*);
- ko‘rsatkichli tur (*Pointer*);
- matnli tur (*Text*).

ByteBool, WordBool, LongBool va *Pchar*lar *Turbo Paskal*ning 7-versiyasiga kiritilgan yangi turlardir.

Foydalanuvchi turlari qo‘shimcha mavhum (oddiy va tasnifli) turlardir, ularning tavsifini dasturchi o‘zi mustaqil aniqlaydi. Bunday turlardan foydalanish dasturga qo‘yilgan masalani oldinroq va aniqroq bayon etishga imkon beradi, kompilatorga esa sintaksis xatolarni tekshirishda va yana ham samarali mashina kodini hosil qilishda ko‘proq axborot yetkazadi.

*Foydalanuvchi turlari*ga quyidagilar kiradi:

— sanab o‘tiladigan turlar;

— interval turlar;

— ko‘rsatkich turlari (standart *Pointer* turidan boshqa);

— tasnifli turlar;
— protsedurali turlar.

(*Turbo Paskal* tilida ishlatiladigan barcha turlarning batafsil tavsifini VI va IX boblarda beramiz.)

4.3.3. O‘zgarmaslar bayoni. O‘zgarmaslar oddiy va turdoshlashtirilgan bo‘lishi mumkin.

Oddiy o‘zgarmaslar bayonining umumiy ko‘rinishi:

const identifikator = o‘zgarmas;
o‘zgarmas = ifoda.

Oddiy o‘zgarmaslarga misollar:

```
const
        {Sonli o‘zgarmaslar}
  Length = 100;
  MinNeg=-1; MaxNeg=-32678; Numb=7.87 e-3;
        {Bulev o‘zgarmaslari}
  Bool = True; Boo12=False;
        {Ramzli o‘zgarmaslar}
  Char7='7'; CharCR = #13;
        {Satrli o‘zgarmaslar}
  Str1='Turbo'; Str2='Pascal'.
```

Oddiy o‘zgarmaslardan tashqari *Turbo Paskal*da dastur kompilatsiyasi vaqtida hisoblanishi mumkin bo‘lgan o‘zgarmasli ifodalarni ishlatishga ham yo‘l qo‘yiladi. Agar yuqorida keltirilgan o‘zgarmaslarni bor, deb hisoblasak, quyidagi o‘zgarmaslarni e’lon qilish mumkin bo‘ladi:

```
const
        ChrLength = Chr (Length);
        Mean = (MaxNeg-MinNeg) div2;
        BoolAnd = Bool1 and Bool2;
        CodeofChar 7 = Ord (Char7);
        Name = Str1+Str2 + CharCR.
```

O‘zgarmas ifodalar odatdagi ifodalar bayon qilinadigan qoidalar bo‘yicha bayon qilinadi. Lekin o‘zgarmas ifodalarda mumkin bo‘lgan standart funksiyalar ro‘yxati quyidagi funksiyalar bilan chegaralanadi:

Abs, Chr, Hi, *Length, Lo, Odd, Ord, Pred, Ptr, Round, Sizeof, Succ, Swap, Trunc*

Turdoshlashtirilgan o‘zgarmaslar. Oddiy o‘zgarmaslardan farq qilib, o‘zgarmaslar bayonida o‘zgarmasning qiymati va uning turi ko‘rsatiladi.

Turdoshlashtirilgan o‘zgarmaslar, aslida, xotiraning statik sinfi o‘zgaruvchilaridir. Ya’ni ular o‘zlari uchun bayon etilgan qiymatni faqat bir marta dastur bajarilishining boshida qabul qilishadi, har safar o‘zlari e’lon qilingan protsedura (funksiya)ga yangi kirishlarida boshqatdan initsializatsiya qilinmaydi va protsedura (funksiya)ning oldingi chaqirilishida qabul qilingan qiymatlarini saqlaydilar. Turdoshlashtirilgan o‘zgarmaslarni xuddi shu turdagi o‘zgaruvchilar qanday ishlatilsa, shunday ishlatish mumkin va ular o‘zlashtirish operatorining chap tomonida paydo bo‘lishi mumkin.

Turdoshlashtirilgan o‘zgarmaslar bayoni:

Identifikator: tur = turdoshlashtirilgan o‘zgarmas.

Turdoshlashtirilgan o‘zgarmas bu o‘zgarmas yoki adres-o‘zgarmas, yoki massiv-o‘zgarmas, yoki yozuv-o‘zgarmas, yoki obyekt-o‘zgarmas, yoki to‘plam-o‘zgarmas bo‘lishi mumkin.

Turdoshlashtirilgan o‘zgarmas qiymatini berishda odatdagi o‘zgarmas ifodalardan tashqari o‘zgarmas adresli ifodalar ishlatiladi.

O‘zgarmas adresli ifoda — qiymati global o‘zgaruvchi, turdoshlashtirilgan o‘zgarmas, protsedura yoki funksiya adresi bo‘lgan ifodadir. O‘zgarmas adresli ifoda protseduralarning lokal o‘zgaruvchilariga yoki dinamik o‘zgaruvchilariga murojaat etilishi mumkin emas, chunki ularning adreslarini kompilatsiya vaqtida hisoblash mumkin emas.

Turdoshlashtirilgan o‘zgarmas aslida initsiallashtirilgan o‘zgaruvchini ifodalagani uchun, uni boshqa o‘zgarmaslar va turlarni e’lon qilishda ishlatish mumkin emas.

Standart turdagi turdoshlashtirilgan o‘zgarmaslar:

```
const
        Arr_Length: Integer=100;
        Step:Real=0.001;
        Flag:Boolean= False;
        LineFeed:Char=#10;
        NewLine:String [2] = #13 #10;
        Name:String [14] = 'Turbo Pascal';
var
        Buffer:array [0 .. 1023] of Byte;
const
        Buffer Ofs:Word = Ofs (Buffer);
        Buffer Seg:Word = Seg (Buffer);
```

Ptr:Pointer = @ Buffer.

Ko‘rsatkichli turdagi turdoshlashtirilgan o‘zgarmaslar:

type

Ptr=^Integer;

const

IntPtr:Ptr=nil;
Int1:Integer=0;
*Int1Ptr:Ptr=@Int*1.

Tasnifli turdagi turdoshlashtirilgan o‘zgarmaslar.

Turbo Paskal turdoshlashtirilgan o‘zgarmaslarning quyidagi tasnifli turlari bilan ish olib boradi:

- «massiv» (*array*) turida;
- «to‘plam» (*set*) turida;
- «yozuv» (*record*) turida;
- «obyekt» (*object*) turida.

Tasnifli turdagi o‘zgarmaslarni bayon qilishda uning har bir a’zosining qiymati ma’lum sintaktik qoidalarga mos ravishda ko‘rsatiladi.

«Massiv» turidagi turdoshlashtirilgan o‘zgarmaslar. Bu turdagi o‘zgarmaslar quyidagi sintaktik qoidalarga ko‘ra bayon etiladi: uni bayon etishda har bir o‘lchamli massiv a’zolari alohida qavslarda yoziladi va bir-biridan vergullar bilan ajratiladi. Eng ichki qavslarda joylashgan a’zolar massivning oxirgi (eng o‘ngdagi) o‘lchamiga to‘g‘ri keladi.

«Massiv» turidagi o‘zgarmaslarga misollar:

- bir o‘lchamli sonli massiv:

```
const
        DigVector : array [1 .. 7] of Real=
        (0.1, 3.25, 21.32, 55, 11.99, 78.1, 4.5);
```

- ikki o‘lchamli sonli massiv:

```
const
        DigMatrix : array [1..3,1..4] of Integer=
        (1,2,3,4), (2,3,4,5), (3,4,5,6).
```

Buning natijasida quyidagi ko‘rinishdagi matritsa tashkil bo‘ladi:

1 2 3 4
2 3 4 5
3 4 5 6

- uch o‘lchamli sonli massiv:

```
const
        Dig3D : array [1..4,1..3,1..2] of Byte=
        (— ((1,2), (1,2), (1,2)), ((1,2), (1,2), (1,2)),
        ((1,2), (1,2), (1,2)), ((1,2), (1,2), (1,2));
```

$\frac{35}{17}$belgilarning bir o‘lchovli massivi:

```
const
        charVector : array [1 .. 6] of Char =
        ('P', 'A', 'S', 'C', 'A', 'L');
```

yoki yana ham qisqaroq

```
        charVector : array [1 .. 6] of Char = 'Pascal'.
```

«To‘plam» turidagi turdoshlashtirilgan o‘zgarmaslar

«To‘plam» turidagi o‘zgarmasning har bir a’zosi tegishli turdagi alohida o‘zgarmasni yoki bir-biridan belgi bilan ajratilgan ikki o‘zgarmasdan iborat qiymatlar oralig‘ini ifodalaydi.

«To‘plam» turidagi o‘zgarmaslarga misol:

```
type
        Digits = set of 0..9;
        CharDig = set of '0' .. '9';
const
        DigSet1 : Digits = [0, 2, 4, 6, 8];
        DigSet2 : Digits = [1..3, 5..7];
        CharDigSet1 : CharDig = ['0', '2', '4', '6', '8'];
        CharDigSet2 : CharDig = ['0' .. '3', '5' .. '7'];
        CharSet     : set of char = ['a' .. 'z', 'A' .. 'Z'].
```

«Yozuv» turidagi turdoshlashtirilgan o‘zgarmaslar

«Yozuv» turidagi o‘zgarmaslar bayonida yozuvning hamma maydonlari qiymatlari ham, ularning identifikatorlari ham tegishli qoida bo‘yicha ko‘rsatiladi. Bunday turdagi o‘zgarmaslarda faylli turdagi maydonlarning bo‘lishiga yo‘l qo‘yilmaydi. Variantli o‘zgarmas — yozuvlarda oldindan o‘rnatilgan maydon — belgi o‘zgarmasiga faqat mos kelgan maydonning variantini ko‘rsatish mumkin. Maydonlar tur bayonida qanday kelgan bo‘lsa, shunday tartibda ko‘rsatiladi.

«Yozuv» turidagi turdoshlashtirilgan o‘zgarmaslarga misollar:

```
type
        Rec = record
                R:Real;
                B:Boolean;
```

```
            C:Char;
        end;
        ArrayOfRec=array [1 .. 3] of Rec;
        RecOfArray= record
            ArrInt : array [1 .. 3] of Integer;
            ArrChar : array [1 .. 2] of Char;
    end;
    RecOfRec = record
        I : Integer;
        S : String;
        Z : Rec;
    end;
const
    RecElem : Rec = (R: 3.1415; B: True; C: '*');
    ArrRec : ArrayOfREc =
        ((R: 3.1415; B: True; C: '*'),
        (R: 0.0; B: False; C: '$'),
        (R: 6.2832; B: True; C: '&'));
    RecArr : RecOfArray =
        (ArrInt: (1,2,3); ArrChar: ('1', '2'));
    RecRec : RecOfRec =
        (I: 32767; S: 'PASCAL';
        Z: (R:3.1415; B: True; C: '*')).
```

«Obyekt» turidagi turdoshlashtirilgan o‘zgarmaslar

«Obyekt» turidagi o‘zgarmaslar bayoni «yozuv» turidagi o‘zgarmaslar kabi bajariladi. Obyekt maydonlari xuddi shunday ko‘rsatiladi, usullar uchun qiymatlar ko‘rsatilmasligi mumkin.

```
type
    Point = object
        X, Y: Integer;
    end;
    Rect=object
        A, B: Point;
        procedure Init (XA, YA, XB, YB: Integer);
        procedure Copy (var R: Rect);
        procedure Move (Dx, Dy: Integer);
    end;
const
    ZeroPoint : Point = (X : 0; Y : 0);
```

```
ScreenRect : Rect = (A : (X : 0 ; Y : 0);
                     B : (X : 80; Y : 25)).
```

Virtual usullarga ega bo‘lgan obyekt turidagi o‘zgarmaslarning initsializatsiyasi kompilatsiya bosqichida avtomatik ravishda bajariladi.

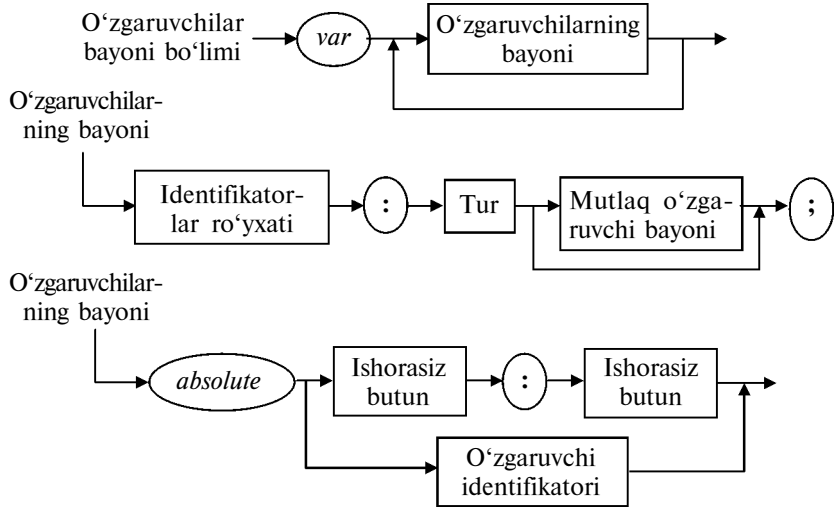
«Protsedura» turidagi turdoshlashtirilgan o‘zgarmaslar

«Protsedura» turidagi o‘zgarmas, o‘zgarmas turi bilan o‘zlashtirish bo‘yicha birgalikda bo‘lgan protsedura yoki funksiya identifikatorini ko‘rsatishi kerak:

```
type
        FuncType = Function (X : Real);
        function SH (X: Real) : Real; far;
        begin
              SH : = (Exp(X) — Exp(-X)) / 2;
        end;
        function CH (X: Real) : Real; far;
        begin
              CH : = (Exp(X) + Exp(-X)) / 2;
        end;
const
        CurrentFunc : FuncType = SH.
```

4.3.4. O‘zgaruvchilar bayoni

O‘zgaruvchilarni bayon etish sintaksisi quyidagi ko‘rinishga ega:



O‘zgaruvchilar bayonida tur sifatida III bobda bayon etilgan identifikator turini yoki mustaqil ravishda turlar sintaksisiga ko‘ra, mustaqil yangi turni olish mumkin:

```
type
        Colors = (Red, Blue, Green);
        Vector = array [1 .. 100] of integer;
var
        a, b, c      :Real;
        i, j         : Integer;
        Flag         : Boolean;
        Color        : Colors;
        Digit        : 0 .. 9;
        Season       : (Spring, Summer, Autumn, Winter);
        Vect1, Vect2 : Vector;
        Matrix : array [1..5, 1..10] of Byte.
```

Absolute direktivasi yordamida xotirada ko‘rsatilgan adres bo‘yicha qat’iy joylashadigan mutlaq o‘zgaruvchilar, deb ataladigan o‘zgaruvchilarni bayon qilish mumkin. Har bir mutlaq o‘zgaruvchi alohida bayon qilinishi, ya’ni e’lon qilishda ikki nuqtadan oldin identifikatorlar ro‘yxati faqat bitta identifikatordan iborat bo‘lishi kerak.

Mutlaq o‘zgaruvchilarni bayon qilishning ikki ko‘rinishi mavjud:

35/17 o‘zgaruvchi joylashtiriladigan aniq adresni ko‘rsatish bilan;

35/17 ikki o‘zgaruvchini bitta adres bo‘yicha joylashtirish bilan.

Birinchi ko‘rinishda bayon etiladigan o‘zgaruvchi joylashtiriladigan segmentning va siljishning aniq qiymatlari (ya’ni to‘liq adres) ko‘rsatiladi:

CrtMode : *Byte absolute* $ 0040: $ 0049.

Birinchisi o‘zgarmas segment qiymatini, ikkinchisi esa bu segment ichida siljish qiymatini aniqlaydi. Ikkala o‘zgarmaslar $0000 dan $FFFF (0 dan 65535) soha chegarasidan chiqmasligi kerak.

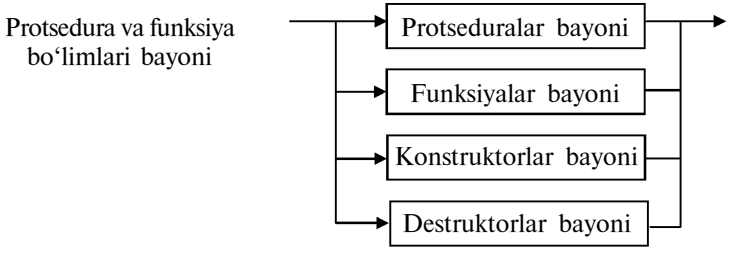
Absolute direktivali ikkinchi ko‘rinish ikkinchi o‘zgaruvchi «ustidan» (ya’ni o‘sha adres bo‘yicha) joylashtiriladigan o‘zgaruvchini bayon etish uchun ishlatiladi.

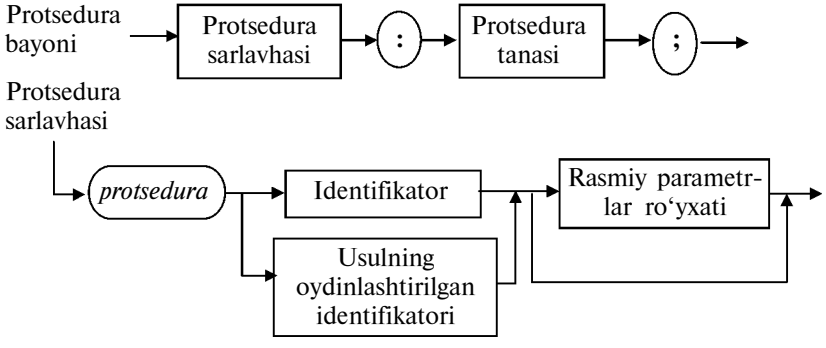
Quyida keltiriladigan dasturlarda *Digits* o‘zgaruvchi *ChDigits*

o‘zgaruvchining adresi bo‘yicha joylashtirish kerakligi ko‘rsatiladi. *Byte* turidagi massivni belgilar satriga qo‘yish natijasida massiv elementlari satr tashkil topgan belgi kodlari qiymatlarini o‘zlashtiradi:

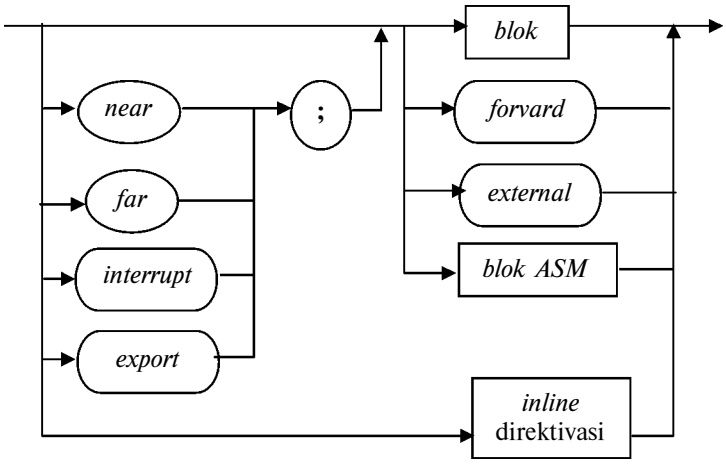
```
Program TestAbsolute;
Const
        ChDigits    : String [10] = '0123456789';
var
        Digits      : array [0..10] of Byte absolute ChDigits;
        i           : integer;
begin
        writeln ('«0» dan «9» gacha belgilar kodi');
        for i:=1 to 10 do Write (Digits [i], ' ');
        writeln
end.
```

4.3.5. Protsedura va funksiyalar bayoni. Bu bo‘limda quyida keltirilgan sintaksis diagrammalar bo‘yicha protsedura, funksiya, konstruktorlar va destruktorlar bayoni bajariladi. Ularning batafsil izohlari VIII bobda beriladi.

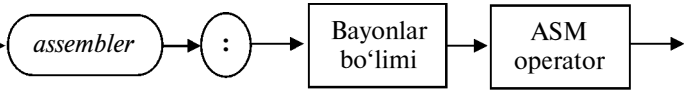




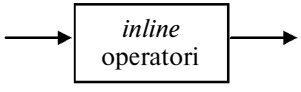
Protsedura tanasi



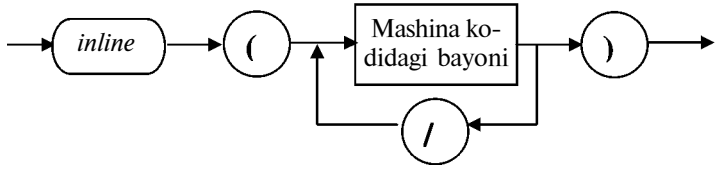
blok ASM



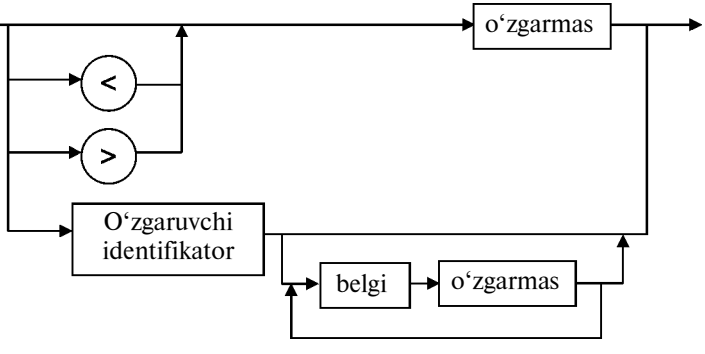
inline direktivasi

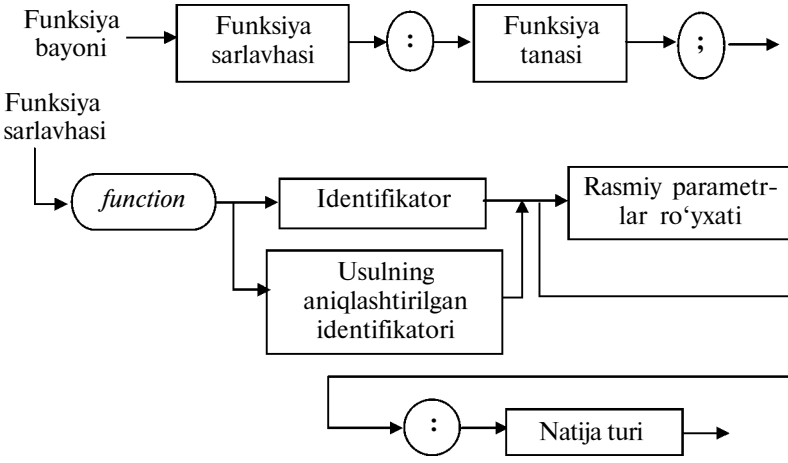


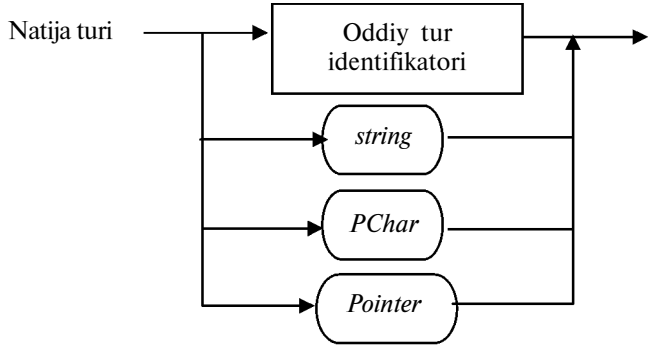
inline operatori

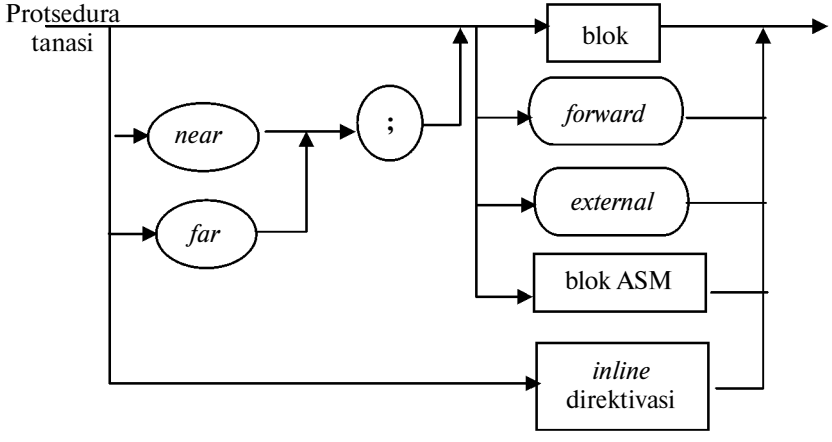


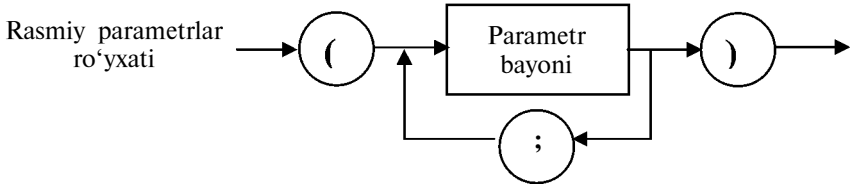
Mashina kodidagi bayoni

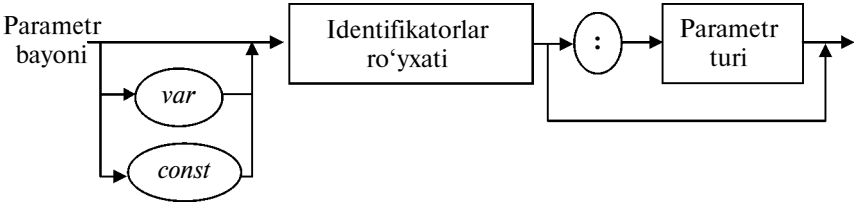


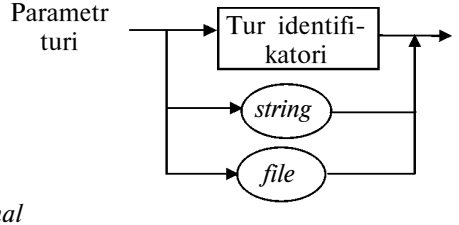

Funksiya bayoni
Funksiya sarlavhasi
:
Funksiya tanasi
;
Funksiya sarlavhasi
function
Identifikator
Usulning aniqlashtirilgan identifikatori
Rasmiy parametr-lar ro‘yxati
:
Natija turi

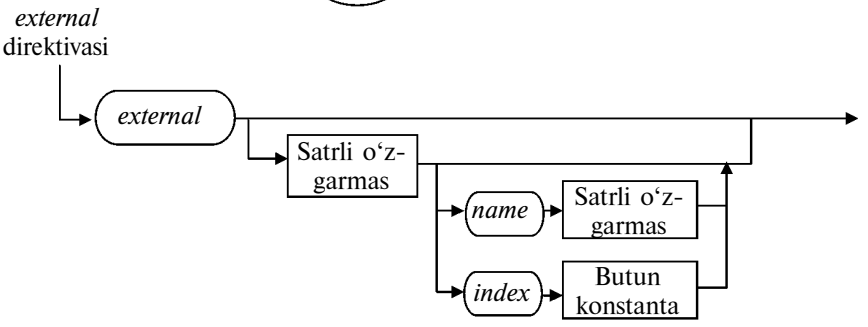

Natija turi
Oddiy tur identifikatori
string
PChar
Pointer

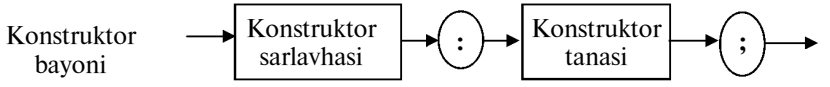

Protsedura tanasi
near
far
;
blok
forward
external
blok ASM
inline direktivasi

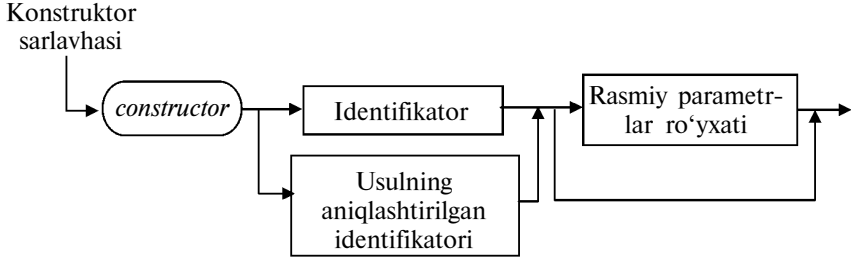

Rasmiy parametrlar ro‘yxati
(
Parametr bayoni
)
;

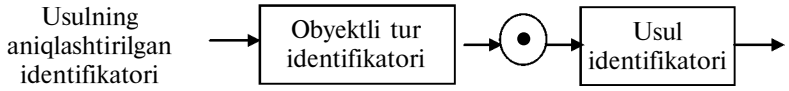

Parametr bayoni
var
const
Identifikatorlar ro‘yxati
:
Parametr turi


Parametr turi
Tur identifi-katori
string
file


external direktivasi
external
Satrli o‘z-garmas
name
Satrli o‘z-garmas
index
Butun konstanta


Konstruktor bayoni
Konstruktor sarlavhasi
:
Konstruktor tanasi
;


Konstruktor sarlavhasi
constructor
Identifikator
Usulning aniqlashtirilgan identifikatori
Rasmiy parametr-lar ro‘yxati

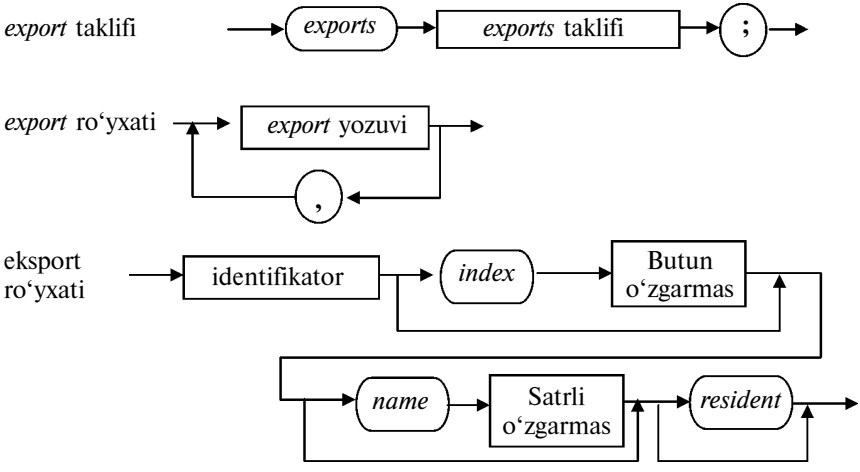


4.3.6. Eksport bayoni

! *Izoh:* kompilatorning BP.EXE, BPC.EXE va BPW.EXE versiyalari bilan tarjima qilinadigan dasturlarda *exports* bo‘limini ishlatish mumkin.

Eksport qilinadigan ismlar bayoni *exports* dastur bayonlar qismining istalgan joyida bir necha marta uchrashi mumkin. Eksport qilinadigan protsedura yoki funksiyaning identifikatori *exports*dagi har bir yozuvni beradi. Bunda bu protsedura yoki funksiya *exports* bo‘limida uning ismi ko‘rsatilguncha bayon qilinishini kuzatish zarur. Bundan tashqari, eksport qilinayotgan protsedura yoki funksiya export (oxirida «s» harfisiz) direktivasiga ega bo‘lishi kerak. Eksport qilinadigan ismlar sifatida aniqlangan identifikatorlar ishlatilishi mumkin.

Quyida *export* bo‘limi bayonining sintaksisi keltirilgan:



exports ichki bo‘limlari bayonida bu dastur tomonidan eksport qilinadigan hamma protsedura va funksiyalar sanab o‘tiladi. Dastur

*exports*ga ega bo‘lgan bo‘lsa ham, amaliyotda bu kam uchrashini ta’kidlaymiz. Odatda, u DLLda ko‘rsatiladi.

*Exports*ni dastur bayonining tashqi bo‘limida yoki DLLda yozish mumkin. Protseduralar, funksiyalar va modul bo‘limlari bayonida ular ishlatilmaydi.

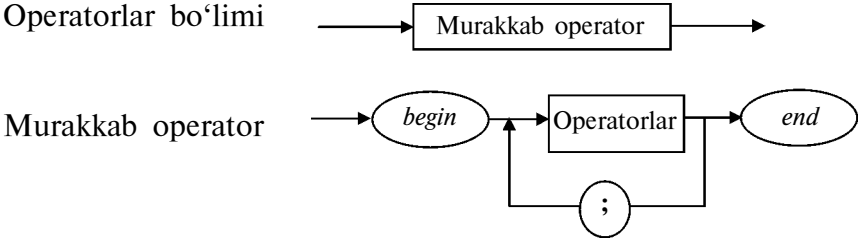
Eksportning har bir yozuvi o‘ziga *index* degan kalit so‘zni kiritishi mumkin, bu so‘zdan keyin 1 dan 32767 sohasida butun qiymatlar keladi. *Index*da ko‘rsatilgan son eksport qilinayotgan protsedura yoki funksiyaga, mos ravishda, maxsus tartib qiymatini qo‘yadi. Agar eksport yozuvida *index* bayoni bo‘lmasa, tartib qiymat avtomatik ravishda o‘zlashtiriladi.

Exports yozuvi *name* kalit so‘ziga ega bo‘lishi mumkin. Bu so‘zdan keyin satrli o‘zgarmas keladi. Bu o‘zgarmas protsedura yoki funksiyaga ism beradi. Bu ism bilan u eksport qilinadi. Agar bayonda *name* so‘zi bo‘lmasa, protsedura yoki funksiya o‘zining, belgilari yuqori registrda o‘zgartiriladigan, identifikatori bilan eksport qilinadi.

Bundan tashqari, eksport yozuvi *resident* kalit — so‘ziga ham ega bo‘lishi mumkin. *Resident* ko‘rsatmasiga ko‘ra eksport to‘g‘risidagi axborot tezkor xotirada, DLL yuklangan vaqtda qoladi. Bu dinamik yuklangan kutubxonada ismi bo‘yicha ichki dasturni izlash vaqtini sezilarli darajada qisqartirishga imkon beradi.

4.4. Operatorlar bo‘limi

Bu dastur tasnifida bo‘lishi shart bo‘lgan yagona bo‘lim. Operatorlar bo‘limining sintaksisi quyidagicha:



Bu bo‘limda ishlatiladigan operatorlar to‘g‘risida VII bobda so‘z yuritiladi.

Nazorat savollari

1. *Turbo Paskal* tilidagi dastur qanday bo‘limlardan iborat?
2. *Turbo Paskal* tilidagi dasturda sarlavha bo‘lishi shartmi?
3. Qaysi hollarda *uses* so‘zi ishlatiladi va dasturning qaysi joyida u yoziladi?
4. *Turbo Paskal*da modul nimani bildiradi?
5. Bayonlar bo‘limi qanday ichki bo‘limlardan iborat?
6. Nishona (metka)lar nima maqsadda ishlatiladi?
7. *Turbo Paskal* turlari to‘plami qanday guruhlarga bo‘linadi?
8. *Turbo Paskal*ning standart turlarini sanab o‘ting.
9. Foydalanuvchi turlari qanday turlar?
10. Foydalanuvchi turlariga qanday turlar kiradi?
11. O‘zgarmaslar qanday bayon qilinadi?
12. O‘zgarmaslarning qanday turlari bor?
13. Turdoshlashtirilgan o‘zgarmaslar oddiy o‘zgarmaslardan nimasi bilan farq qiladi?
14. Turdoshlashtirilgan o‘zgarmaslar qanday guruhlarga bo‘linadi?
15. Tasnifli turdagi turdoshlashtirilgan o‘zgarmaslarga qaysi guruhlar kiradi?
16. O‘zgaruvchilar qanday bayon etiladi?
17. Protsedura va funksiyalar bayoni qanday ketma-ketlikda amalga oshiriladi?
18. Eksport ichki bo‘limi vazifasi va bayon etish tartibi qanday?
19. Operatorlar bo‘limi qanday bayon etiladi, uning boshqa bo‘limlardan farqi nimada?

V bob. ***TURBO PASKAL*** **MA'LUMOTLARI TARKIBI**

Algoritmlar bilan bir qatorda, ma'lumotlar tarkibi ham yaratilayotgan dasturning asosiy tashkil etuvchi qismlari bo'ladi.

Ixtiyoriy ma'lumotlar, ya'ni o'zgarmaslar, o'zgaruvchilar, funksiya yoki ifodalarning qiymatlari *Turbo Paskal* turlari bilan tavsiflanadi. *Turbo Paskal* u yoki bu obyektning mumkin bo'lgan qiymatlari to'plami, shuningdek, unga nisbatan qo'llanilishi mumkin bo'lgan amallar to'plamini aniqlaydi. Bundan tashqari, tur ma'lumotlarni shaxsiy kompyuter xotirasida ichki ifodalash formatini aniqlaydi.

Dasturlashda ishlatilayotgan ma'lumotlarni ikki katta guruhga bo'lish mumkin:

- statik tasnifli ma'lumotlar;
- dinamik tasnifli ma'lumotlar.

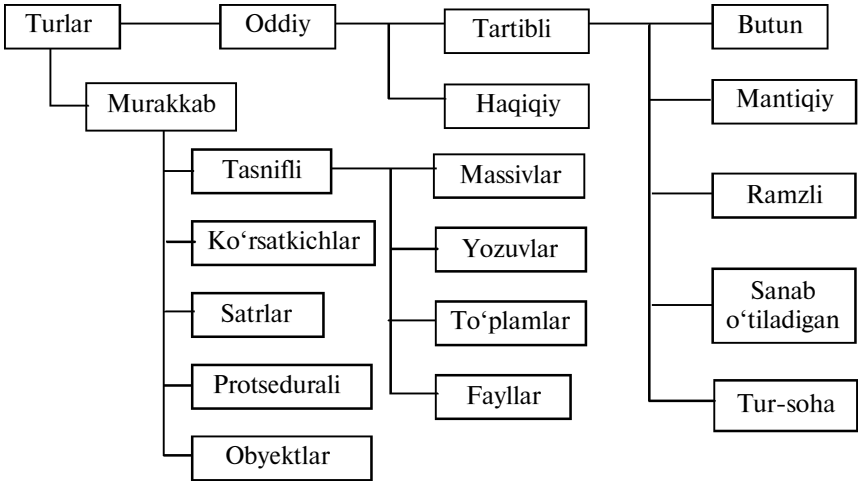
Elementlarning o'zaro joylashuvi va o'zaro aloqasi hamma vaqt o'zgarmas bo'lib qoluvchi ma'lumotlar *statik tasnifli* ma'lumotlar, deyiladi.

Ichki tuzilishi qandaydir bir qonuniyatga ko'ra shakllanadigan, lekin elementlari soni, ularning o'zaro joylashuvi va o'zaro aloqasi dasturni bajarish vaqtida, shakllanish qonuniyatiga ko'ra, dinamik o'zgarishi mumkin bo'lgan ma'lumotlar *dinamik tasnifli* ma'lumotlar deyiladi.

5.1. Statik tasnifli ma'lumotlar

Turbo Paskal statik tasnifli ma'lumotlar turlarining tarmoqlangan tuzilishiga ega (5.1-chizma).

*Turbo Paskal*da ma'lumotlarning yangi turlarini yaratish mexanizmi hisobga olingan. Unga ko'ra, dasturda ishlatiladigan turlarning umumiy soni xohlagancha katta bo'lishi mumkin.



5.1-chizma. Statik ma’lumot turlarining tasnifi.

Tasnifdan ko‘rinib turibdiki, statik tasnifli ma’lumotlar *oddiy (skalar)* va *murakkab (agregatli)* bo‘lar ekan.

Dasturlash tillarida *oddiy ma’lumotlarga* ma’lumotlarning standart (oldindan aniqlangan) turlari mos keladi, ularga, odatda, arifmetik (natural, butun, haqiqiy, kompleks), ramzli, bulev va ko‘rsatkichli (murojaatli) turlari kiradi. *Turbo Paskal*ga *Byte, Word* (natural), *Integer, Shortint, Longint* (butun), *Real, Single, Double, Extended, Satr* (haqiqiy), *Boolean, ByteBool, WordBool, LongBool* (bulev), *Char* (ramzli) va *Pointer* (ko‘rsatkichli) turlar kiritilgan. *Turbo Paskal*ning haqiqiy turlari o‘rnatilgan va suzuvchi nuqta shaklida ifodalanishi kerak.

Bundan tashqari, ba’zi bir dasturlash tillari (birinchi shunday til *Paskal* edi) dasturchiga xususiy skalar turlarini (ular uchun mumkin bo‘lgan barcha qiymatlarni sanab o‘tish yoki boshqa skalar tur qiymatlarining ichki sohasini ko‘rsatish yo‘li bilan) bayon etishga imkon beradi.

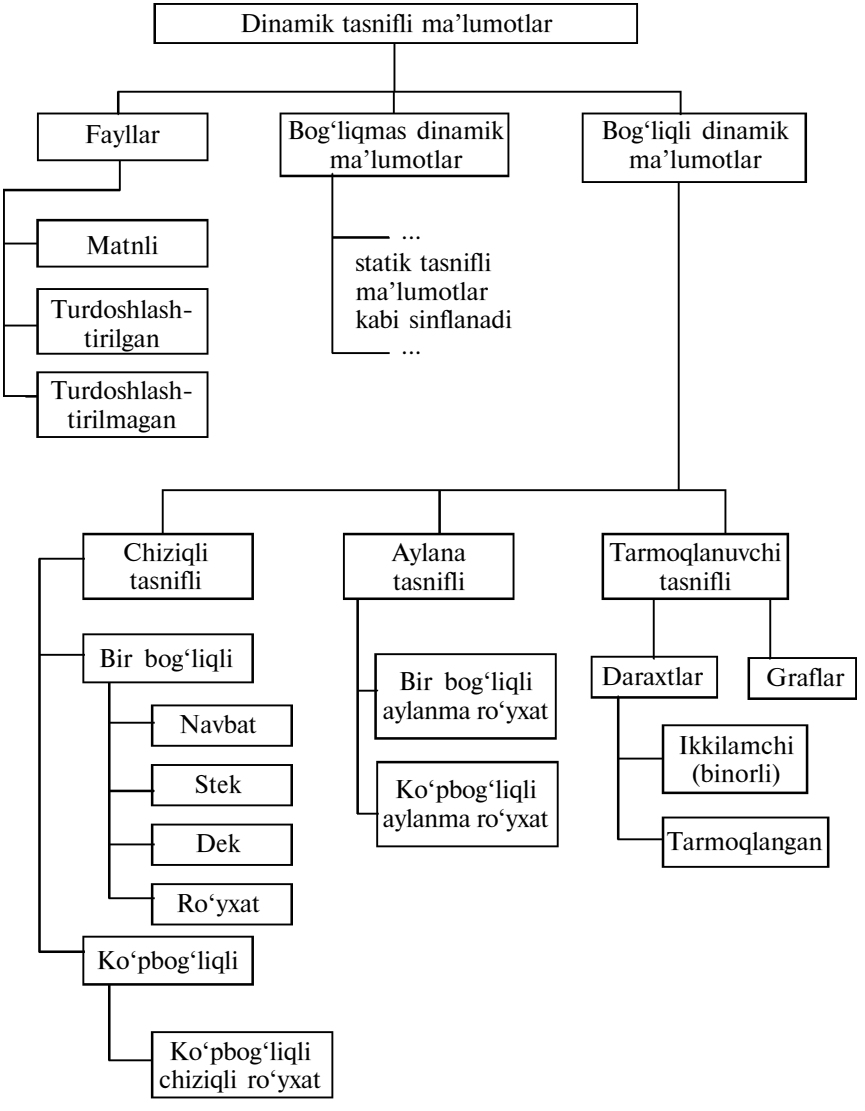
Murakkab tasnifli ma’lumotlar bir jinsli, ya’ni hamma elementlari bir xil va bir jinslimas (ular har xil turdagi elementlarni bir butun qilib birlashtiradi) bo‘ladi. *Bir jinsli tasnifli ma’lumotlarga* massivlar, satrlar va to‘plamlar, *bir jinslimas tasnifli ma’lumotlarga* esa oddiy yozuvlar, variantli yozuvlar, birlashmalar va obyektlar tegishli bo‘ladi.

Statik tasnifli ma’lumotlar bilan ishlash VIII bobda beriladi.

5.2. Dinamik tasnifli ma’lumotlar

Dinamik tasnifli ma’lumotlarga fayllar, bog‘liqmas va bog‘liqli dinamik ma’lumotlar kiradi.

Dinamik tasnifli ma’lumotlarning tarkibi quyidagi tarmoqlangan tuzilishga ega:



Nazorat savollari

1. Dasturning asosiy tashkil etuvchi qismlariga nimalar kiradi?
2. *Turbo Paskal*da turlar nimalar bilan tavsiflanadi?
3. Dasturlashda ishlatiladigan ma'lumotlar qanday guruhlarga bo'linadi?
4. Statik tasnifli ma'lumotlar qanday ma'lumotlar?
5. Dinamik tasnifli ma'lumotlar qanday ma'lumotlar?
6. Statik tasnifli ma'lumotlar qanday guruhlarga bo'linadi?
7. Statik oddiy ma'lumotlarga qanday turlar kiradi?
8. Statik murakkab ma'lumotlarga qanday turlar kiradi?
9. Dinamik tasnifli ma'lumotlarga qanday ma'lumotlar kiradi?
10. Dinamik tasnifli ma'lumotlar qanday tarmoqlangan tuzilishga ega?

VI bob. ***TURBO PASKAL*** MA'LUMOTLARINING ODDIY TURLARI

Oddiy turdagi ma'lumotlarga tartibli va haqiqiy turlar kiradi.

Tartibli turlarning har bir chekli sondagi mumkin bo'lgan qiymatlarga ega bo'lishi bilan farqlanadi. Bu qiymatlarni ma'lum tarzda tartibga solish (turning nomlanishi shundan kelib chiqadi) mumkin va, demak, ularning har biri bilan qandaydir butun sonni — qiymatning tartib raqamini taqqoslash mumkin.

Haqiqiy turlar ham sonni ichki ifodalash formati bilan aniqlanuvchi qiymatlarning chekli soniga ega. Lekin haqiqiy turlarning mumkin bo'lgan qiymatlar soni shunchalik kattaki, ularning har biri bilan butun son (uning raqami)ni taqqoslash mumkin emas.

6.1. Tartibli turlar

Tartibli turlarga *butun, mantiqiy, ramzli, sanab o'tiladigan* va *tur-sohalar* kiradi. Ularning ixtiyoriy biriga, X ifodaning tartib qiymati raqamini qaytaruvchi, *ORD(x)* funksiyani qo'llash mumkin. Butun turlar uchun ORD(x) funksiya ixtiyoriy butun turga tegishli *x* uchun *x* qiymatning o'zini qaytaradi, ya'ni *ORD(x)=x*. *ORD(x)* funksiyani mantiqiy, ramzli va sanab o'tiladigan turlarga qo'llash 0 dan 1 gacha (mantiqiy tur), 0 dan 255 gacha (ramzli), 0 dan 65535 gacha (sanab o'tiladigan) sohada bo'lgan musbat butun sonni beradi. Tur-soha tartibli turning hamma xossalarini saqlaydi, shuning uchun unga *ORD(x)* funksiyani qo'llash natijasi bu tur xossasiga bog'liq bo'ladi.

Tartibli turlarga, shuningdek, quyidagi funksiyalarni qo'llash mumkin.

PRED(x) — tartibli turning (qiymati *ORD(x)—1* tartib raqamiga mos keluvchi) oldingi qiymatini qaytaradi, ya'ni

$$ORD\ (PRED(x)) = ORD(x)-1;$$

SUCC(x) — *ORD(x)+1* tartib raqamiga mos keluvchi tartibli

turning keyingi qiymatini qaytaradi, ya'ni

$$ORD(SUCC(x)) = ORD(x) + 1.$$

Masalan, agar dasturda o'zgaruvchi aniqlangan bo'lsa,

```
var
        c:char;
begin
        c:='5'
end.
```

Unda *PRED(c)* funksiya '4' qiymatni, *SUCC(c)* funksiya esa '6' qiymatni qaytaradi.

Agar ixtiyoriy tartibli tur tartibga solingan, o'ngdan chapga tomon o'suvchi va sonlar o'qida qandaydir bir kesmani egallovchi to'plam, deb faraz qilinsa, *PRED(x)* funksiya kesmaning chapdagi, *SUCC(x)* funksiya esa kesmaning o'ngdagi oxiri uchun aniqlanmagan bo'ladi.

6.1.1. Butun turlar — butun turlarning qabul qilishi mumkin bo'lgan qiymatlar sohasi ularning bir, ikki yoki to'rt baytni egallashi mumkin bo'lgan ichki ifodalanishiga bog'liqdir. 6.1-jadvalda butun turlarning nomlari, ularning baytlardagi ichki ifodalanish uzunligi va qabul qilishi mumkin bo'lgan qiymatlar sohasi keltirilgan.

6.1-jadval

Butun turlar		
Ismi	Uzunligi, bayt	Qiymatlar sohasi
Byte	1	0...255
ShortInt	1	-128...+127
Word	2	0...65536
Integer	2	-32768...+32767
LongInt	4	-2147483648...+2147483647

Butun sonli parametrli protsedura va funksiyalarni ishlatishda turlarning «ichma-ichligi»ga tayanish mumkin, ya'ni *WORD* ishlatilishi mumkin bo'lgan hamma joyda *BYTE* ni ishlatish mumkin (lekin aksi emas), *LONGINT*ga *INTEGER* «kiradi», u, o'z navbatida, o'z ichiga *SHORTINT*ni oladi.

Butun turlarga qo'llaniladigan protsedura va funksiyalar ro'yxati 6.2-jadvalda keltirilgan. *b,s,w,i,l* harflari bilan, mos ravishda,

BYTE, SHORTINT, WORD, INTEGER va LONGINT turidagi ifodalar belgilangan; *vb,vs,vw,vi,vl,vx* — harflar tegishli turdagi o‘zgaruvchilarni bildiradi. Kvadrat qavslarda shart bo‘lmagan parametr ko‘rsatilgan.

6.2-jadval

Butun turlarga qo‘llaniluvchi standart protsedura va funksiyalar		
Murojaat	Natija turi	Faoliyati
abs(x)	x	X modulini qaytaradi
chr(b)	char	Ramzni kodi bo‘yicha qaytaradi
dec(vx[,i])	—	vx ning qiymatini i ga, agar i bo‘lmasa, 1 ga kamaytiradi
inc(vx[,i])	—	vx ning qiymatini i ga, agar i bo‘lmasa, 1 ga oshiradi
Hi(i)	Byte	Argumentning bosh baytini qaytaradi
Hi(w)	Byte	O‘shaning o‘zi
LO(i)	Byte	Argumentning kichik baytini qaytaradi
LO(w)	Byte	O‘shaning o‘zi
Odd(e)	Boolean	Agar argument toq son bo‘lsa, True ni qaytaradi
Random(w)	Parametrdagi kabi	0...(w-1) soha teng taqsimlangan mavhum tasodifiy sonni qaytaradi
sgr(x)	X	Argument kvadratini qaytaradi
swap(i)	Integer	So‘zdagi baytlar o‘rnini almashtiradi
swap(w)	Word	

Butun sonlar bilan ish ko‘rganda natija turi operandalar turiga, agar operandalar har xil butun turlarga tegishli bo‘lsa, maksimal quvvatga (qiymatlarning maksimal sohasiga) ega bo‘lgan operanda turiga mos bo‘ladi. Natijaning to‘lib qolish ehtimolligi nazorat qilinmaydi. Bu esa anglashilmovchiliklarga olib kelishi mumkin, masalan:

```
var
        a:Integer;
        x,y:Real;
begin
        a:=32767; {Integer turining mumkin bo‘lgan maksi-
```

```
                mal qiymati}
    x:=a+2;     {Bu ifodani hisoblashda to‘lishning sodir
                bo‘lishi!}
    y:=LongInt (a)+2; {O‘zgaruvchi quvvatliroq turga keltiril-
                        gandan keyin to‘lish yo‘qoldi}
    writeln(x:10:0,  y:10:0)

end.
```

Dasturning EHMdan o‘tkazilishi natijasida quyidagilar olindi:

—32767 32769

6.1.2. Mantiqiy tur. Mantiqiy turning qiymatlari oldindan e’lon qilingan *FALSE* (yolg‘on) yoki *TRUE* (haqiqat) ikkita o‘zgarmaslardan biri bo‘lishi mumkin. Ularga quyidagi qoidalar xos:

ORD(False) = 0;
ORD(True) = 1;
False < *True;*
Succ(False) = *True;*
Pred(True) = *False.*

Mantiqiy tur tartibli turga tegishli bo‘lgani uchun, uni hisob turidagi operatorda ishlatish mumkin, masalan,

```
var
    l: Boolean;
begin
    for l: = False to True do ...
```

Ramzli tur. Shaxsiy kompyuterning hamma belgilar to‘plami ramzli tur qiymatlari bo‘ladi. Har bir belgiga 0...255 sohadagi butun son yoziladi. Bu son belgini ichki ifodalash uchun kod bo‘lib xizmat qiladi, uni ORD funksiya qaytaradi.

Kodlashtirish uchun ASCII (*American Standart Code for Information Interchange* — axborotni almashtirish uchun Amerika standart kodi) kodi ishlatiladi. Bu 7 bitli kod, ya’ni uning yordamida faqat 128 ta belgini 0 dan 127 gacha sohada kodlashtirish mumkin. Shu vaqtning o‘zida *Turbo Paskal*ning belgilarini saqlash uchun ajratilgan 8 bitli baytda ikki marta ko‘p belgilarni 0 dan 255 sohada kodlash mumkin. ShK 0...127 kodlik belgilarining birinchi yarmi ASCII standartiga (6.3-jadval) mos keladi, 128 dan 255 gacha kodlarga ega bo‘lgan belgilarning ikkinchi yarmi standartning qat’iy doiralarida chegaralanmagan va har xil rusumli ShKlarda o‘zgarishi mumkin.

6.3-jadval

ASCII standarti bo‘yichabelgilarning kodlari							
Kod	Belgi	Kod	Belgi	Kod	Belgi	Kod	Belgi
0.	NUL	32.	BL	64.	@	96.	`
1.	SOH	33.	!	65.	A	97.	a
2.	STX	34.	"	66.	B	98.	b
3.	ETX	35.	#	67.	C	99.	c
4.	EOT	36.	$	68.	D	100.	d
5.	ENQ	37.	%	69.	E	101.	e
6.	ACK	38.	&	70.	F	102.	f
7.	BEL	39.	´	71.	G	103.	g
8.	BS	40.	(	72.	H	104.	h
9.	HT	41.	)	73.	I	105.	i
10.	LF	42.	*	74.	J	106.	j
11.	VT	43.	+	75.	K	107.	k
12.	FF	44.	,	76.	L	108.	l
13.	CR	45.	—	77.	M	109.	m
14.	SO	46.	.	78.	N	110.	n
15.	SI	47.	/	79.	O	111.	o
16.	DEL	48.	0	80.	P	112.	p
17.	DC1	49.	1	81.	Q	113.	q
18.	DC2	50.	2	82.	R	114.	r
19.	DC3	51.	3	83.	S	115.	s
20.	DC4	52.	4	84.	T	116.	t
21.	NAK	53.	5	85.	U	117.	u
22.	SYN	54.	6	86.	V	118.	v
23.	ETB	55.	7	87.	W	119.	w
24.	CAN	56.	8	88.	X	120.	x
25.	EM	57.	9	89.	Y	121.	y
26.	SUB	58.	:	90.	Z	122.	z
27.	ESC	59.	;	91.	[	123.	{
28.	FS	60.	<	92.	\	124.	|

29.	GS	61.	=	93.	]	125.	}
30.	RS	62.	>	94.	^	126.	~
31.	US	63.	?	95.	_	127.	□

0...31 kodlik belgilar xizmat kodlariga kiradi. Agar bu kodlar dasturning ramzli belgili matnida ishlatilsa, ular bo‘shliqlar deb hisoblanadi. Ular kiritish-chiqarish ishlarida ishlatilsa, bu belgilar quyidagi mustaqil ma’nolarga ega bo‘lishi mumkin:

Belgi	Kod	Ma’nosi
BEL	7	Qo‘ng‘iroq; bu belgi ekranga tovush signali hamkorligida chiqadi
HT	9	Gorizontal tabulatsiya; ekranga chiqarilganida kursorni 8 karra, plus 1 (9,17,25 va h.k.) xona keyinga siljitadi
LF	10	Satrni o‘tkazish; bu belgi ekranga chiqarilsa, keyingi belgilar shu xonadan boshlab, navbatdagi satrdan chiqariladi
VT	11	Vertikal tabulatsiya; ekranga chiqarilganda, maxsus belgi bilan almashtiriladi
FF	12	Betni o‘tkazish; printerga chiqarilganda betni rasmiylashtiradi, ekranga chiqarilganda maxsus belgi bilan almashtiriladi
CR	13	Karetkani qaytarish; Enter klavishini bosish bilan kiritiladi (READ yoki READLN yordamida kiritishda «kiritish» buyrug‘ini bildiradi va kiritish buferiga sig‘maydi; chiqarishda «chiqarishni joriy satrning boshidan davom et», degan buyruqni bildiradi)
SUB		Fayl oxiri; klaviaturadan Ctrl-Z klavishlarini bosish bilan kiritiladi, chiqarishda maxsus belgi bilan almashtiriladi
ESC		Ish oxiri; klaviaturadan ESC klavishini bosish bilan kiritiladi; chiqarishda maxsus belgi bilan almashtiriladi

CHAR turiga munosabat amallari, shuningdek, o‘rnatilgan funksiyalar qo‘llaniladi.

CHR(B) — CHAR turidagi funksiya; BYTE turidagi B ifodani ramziyga o‘zgartiradi va uni o‘z qiymati bilan qaytaradi.

UPCASE(CH) — CHAR turidagi funksiya; agar CH kichik lotin harfi bo‘lsa, bosh harfni qaytaradi, aks holda CH belgining o‘zini qaytaradi, masalan,

```
var
        c1, c2 : Char;
```

```
begin
        c1:= UpCase ('s');
        c2:= UpCase ('f');
        WriteLn (c1,'      ',c2)
end.
```

UPCASE funksiya kirill imlosini ishlamasligi uchun, bu dasturni o‘tkazish natijasida ekranga

s f

chiqariladi.

6.1.3. Sanab o‘tiladigan tur. Bu tur qabul qilishi mumkin bo‘lgan qiymatlarni sanash bilan beradi. Har bir qiymat qandaydir identifikator bilan nomlanadi va aylana qavslarga olingan ro‘yxatda yoziladi, masalan:

```
type
        colors = (red, white, blue).
```

Sanab o‘tiladigan turlarning qo‘llanilishi dasturlarni ko‘rgazmali qiladi. Masalan, yil oylari bilan bog‘langan ma’lumotlar ishlatilgan dasturning bo‘lagi quyidagi ko‘rinishda bo‘lsa, u kirill imlosida o‘quvchilar uchun ko‘rgazmali bo‘lar edi.

```
type
        TurOy =  (янв, фев, март, апр, май, июн, июл,
                  авг, сен, окт, ноя, дек)
var
        oy:TurOy;
begin
        .............
        if oy = avg then Writeln ('Салқин жойда дам олсак,
        яхши бўлар эди!');
        .............
end.
```

Lekin, afsuski, *Turbo Paskal*da kirill harfidan identifikatorlarda foydalanish mumkin emas, shuning uchun biz dastur bo‘lagini quyidagicha yozishga majburmiz:

```
type
        TypeMonth = (jan, feb, mar, apr, may, jun, jul, aug,
                     sep, oct, nov, dec);
var
        Month:TypeMonth;
begin
```

```
.....
if month=aug then Writeln ('Salqin joyda dam olsak,
                           yaxshi bo'lar edi!')
.....
end.
```

Sanab o‘tiladigan tur qiymatlari va bu qiymatlarning tartib raqamlari o‘rtasidagi moslik sanab o‘tish tartibi bilan o‘rnatiladi: ro‘yxatdagi birinchi qiymat — 0, ikkinchisi — 1 va h.k. tartib raqamlarini qabul qiladi. Sanab o‘tiladigan turning maksimal quvvati 65535 qiymatni tashkil qiladi, shuning uchun amalda sanab o‘tiladigan tur WORD butun turidagi qandaydir ichki to‘plamni beradi, unga 0,1 va h.k. qiymatli butun sonli o‘zgarmaslar guruhlarini birdaniga ixcham holda e’lon qilinishi, deb qarash mumkin.

Sanab o‘tiladigan turlarning ishlatilishi, tegishli o‘zgaruvchilar qabul qiladigan qiymatlarni nazorat qila olish imkoniyatining bo‘lishi bilan, dastur ishonchliligini oshiradi. Masalan, quyidagi sanab o‘tiladigan turlar berilgan bo‘lsin:

```
type
        colors = (black, red, white);
        ordenal = (one, two, three);
        days = (Monday, Tuesday, Wednesday).
```

Imkoniyati va ichki ifodalanishi nuqtayi nazaridan uchala tur teng kuchli:

ord (black)=0, ..., *ord (white)*=2,
ord (one)=0, ..., *ord (three)*=2,
ord (monday)=0, ..., *ord (wednesday)*=2

Lekin,

```
    var
        col:colors;
        num:ordenal;
        day:days;
```

o‘zgaruvchilar aniqlangan bo‘lsa,

```
        col:=black;
        num:=succ (two);
        day:=pred (tuesday);
```

operatorlar ishlatilishi mumkin,

```
        col:=one;
        day:=black;
```

operatorlarining ishlatilishiga esa yo‘l qo‘yilmaydi.

Yuqorida aytib o‘tganimizdek, sanab o‘tiladigan turlar qiymatlariga butun sonlar to‘plami o‘rtasida ORD(x) funksiya bilan beriluvchi bir qiymatli moslik mavjud. *Turbo Paskal*da teskari shakl almashtirishga ham yo‘l qo‘yiladi: WORD turidagi ixtiyoriy ifodani, agar butun sonli ifoda qiymati sanab o‘tiladigan tur quvvatidan oshmasa, sanab o‘tiladigan tur qiymatlariga o‘tkazish mumkin (IX bobning 4-bo‘limiga qaralsin). Masalan, yuqorida ko‘rilgan turlarni e’lon qilishga quyidagi o‘zlashtirishlar teng kuchli:

col:=one;
col:=colors (o).

Bundan

col:= 0;

o‘zlashtirishning bo‘lishi mumkin emasligi kelib chiqadi.

Ixtiyoriy sanab o‘tiladigan turning o‘zgaruvchilarini bu turni oldindan bayon etmasdan e’lon qilish mumkin, masalan,

var
col: (black, white, green).

6.1.4. Tur-soha. Tur-soha o‘zining asosiy turiga kiruvchi ichki to‘plamni bildiradi. Tur-sohadan boshqa ixtiyoriy sanab o‘tiladigan tur ichki to‘plam sifatida ishlatilishi mumkin.

Tur-soha ichki bazaviy turining qiymatlari chegarasi bilan beriladi:

<min.qiymat> .. <maks.qiymat>

Bu yerda,

<min.qiymat> — tur-sohaning minimal qiymati,
<maks.qiymat> — uning maksimal qiymati.

Misol:

type
digit = ′0′ .. ′9′;
dig2 = 48 .. 57.

Tur-sohani TYPE bo‘limida bayon etish shart emas, uni o‘zgaruvchini e’lon qilishda bevosita ko‘rsatish mumkin, masalan,

var
date :1 .. 31;
month : 1 .. 12;
lchr : ′A′ .. ′z′.

Tur sohani aniqlashda quyidagi qoidalarga rioya qilish kerak:

$\frac{35}{17}$ ikkita « .. » belgi bitta belgi deb qaraladi, shuning uchun ular o‘rtasiga bo‘shliqlar qo‘yish mumkin emas;

• sohaning chap chegarasi o‘ng chegarasidan oshmasligi kerak.

Tur-soha o‘zining bazaviy turining hamma xossalarini (uning kichik quvvati bilan bog‘liq bo‘lgan cheklanishlar bilan) saqlaydi. Xususiy holda agar o‘zgaruvchi aniqlangan bo‘lsa,

```
type
        days       = (mo, tu, we, th, fr, sa, su);
        WeekEnd    = sa .. su;
var
        w: weekEnd;
begin
        .....
        w:=sa;
        .....
end.
```

ORD(w) funksiya 5 qiymatni qaytargan vaqtda, PRED(w) xatolikka olib keladi.

Turbo Paskal standart kutubxonasiga tur-sohalar bilan ishlovchi ikkita funksiya kiritilgan:

HIGH(x) — *x* o‘zgaruvchi tegishli bo‘lgan tur-sohaga maksimal qiymatni qaytaradi;

LOW(x) — tur-sohaning minimal qiymatini qaytaradi.

Quyidagi qisqa dastur ekranga

—32768 ... 32767

satrni chiqaradi.

```
var
        K:Integer;
begin
        writeln(Low(K),  '...' , High (K))
end.
```

6.2. Haqiqiy turlar

Qiymatlari hamma vaqt butun son bilan taqqoslanadigan va, demak, shaxsiy kompyuterda mutlaq aniq ifodalanadigan, sanab o‘tiladigan turlardan farq qilib, haqiqiy turlarning qiymatlari ixtiyoriy sonni, haqiqiy sonning ichki formatiga bog‘liq bo‘lgan, faqat qandaydir chekli aniqlik bilan belgilaydi.

6.4-jadval

Bayt uzunligi	Nomi	Ma'noga ega raqamlar soni	O'nlik tartib sohasi
6	Real	11 ...12	-39 ...+38
8	Double	15 ...16	-324 ...+308
10	Extended	19 ...20	-4951 ...+4932
8	Comp	19 ...20	$-2\cdot10^{63}+1...+2\cdot10^{63}-1$

*Turbo Paskal*da haqiqiy son 4 dan 10 tagacha aralash bayt-larga ega va shaxsiy kompyuter xotirasida quyidagicha tasniflanadi:

s	e	m

Bu yerda, *s*—sonning ishorali razryadi; *e*—eksponensial qism, ikkilamchi tartibga ega; *m*—son mantissasi.

Mantissa *m* 23 dan (SINGLE uchun) 63 gacha (EXTENDED uchun) ikkilamchi xonali uzunlikka ega, bu esa SINGLE uchun 7...8 va EXTENDED uchun 19...20 o'nlik raqamli aniqlikni ta'minlaydi. O'nlik nuqta (vergul) mantissaning chap (bosh) xonasi oldida, deb tushuniladi, lekin sonlar bilan ishlaganda uning o'rni chapga yoki o'ngga tomon, eksponensial qismida saqlanuvchi sonning ikkilamchi tartibiga mos ravishda siljiydi, shuning uchun haqiqiy sonlar ustida bajariladigan amallar, suzuvchi nuqtali (vergulli) arifmetika deyiladi.

Turbo Paskal haqiqiy turlarning boy turiga ega. Lekin SINGLE, DOUBLE, EXTENDED turlarini faqat kompilatsiyaning alohida rejimlarida ishlatish mumkin. Gap shundaki, bu turlar suzuvchi nuqtali arifmetikaning apparatli quvvatlanishiga mo'ljallangan va ularning samarali ishlatilishi uchun shaxsiy kompyuter tarkibiga arifmetik soprotsessor kiritilishi kerak. *Turbo Paskal* kompilatori ixtiyoriy shaxsiy kompyuterda soprotsessorli yoki usiz ishlaydigan va ixtiyoriy haqiqiy sonlardan foydalaniladigan dasturlar yaratishga imkon beradi.

Bu uchun kompilatorni to'g'rilash kerak. Yuklash jarayonida *Turbo Paskal* apparatli vositalarning tarkibini tekshiradi va soprotses-sorning bor-yo'qligini aniqlaydi. Ba'zi bir hollarda avtonazoratni o'chirishga zaruriyat bo'ladi. Buning uchun *Turbo Paskal*ni ishga tushirishdan oldin DOSning quyidagi buyrug'ini berish kerak:

set 87 = *N*.

Aksincha, *set* 87 = *Y* buyruq avtonazoratni ishga tushiradi, bu buyruq sukut asosida ishlaydi.

Arifmetik soprotsessor hamma vaqt EXTENDED formatidagi sonlar bilan ishlaydi, bu holda, qolgan boshqa uch xildagi haqiqiy turlar esa natijani kerakli o‘lchamgacha oddiy qirqish bilan hosil qiladi va, asosan, xotirani tejashda ishlatiladi.

Masalan, agar «mashina epsiloni» quyidagi dastur asosida hisoblansa:

```
{$N+, E+}
type
        RealType = Real;
var
        epsilon: RealType;
begin
        epsilon:=1;
        while 1 + epsilon/2 > 1 do
                epsilon: = epsilon/2;
        WriteLn (epsilon)
end.
```

REALTYPE turining (u SINGLE, REAL, DOUBLE yoki EXTENDED bo‘lishi mumkin) e’lon qilinganiga bog‘liq bo‘lmagan holda chop etishga quyidagi natija uzatiladi:

1. 08420217248550E — 0019

Bu EXTENDED turiga mos keladi. Bu shu sababga ko‘ra sodir bo‘ladiki, haqiqiy ifodali hamma 1+ *epsilon*/2 operandalar WHILE operatorida hisoblashdan oldin avtomatik tarzda EXTENDED turiga aylantiriladi. To‘g‘ri natija hosil qilish uchun (masalan, REALTYPE turi uchun u 9.0949470177298E—0113) dasturni quyidagi tarzda o‘zgartirish kerak:

```
{$N+, E+}
type
        RealType = Real;
var
        epsilon, eps1: RealType;
begin
        epsilon:=1;
        repeat
                epsilon: = epsilon/2;
                eps1:=1+ epsilon
        until eps1=1;
        WriteLn (2*epsilon)
```

end.

REAL turi soprotsessorsiz ishlash uchun yaxshilanganligini hisobga olish kerak. Agar dasturni qisqa vaqt ichida ishlatish talabi qo‘yilgan bo‘lsa, REAL ni SINGLE yoki DOUBLE turlari bilan almashtirish kerak, bunday paytda soprotsessorli mashinalarda hisoblash tezligi 2...3 marta oshadi. Agar shaxsiy kompyuterda arifmetik soprotsessor bo‘lmasa, haqiqiy turning hamma turlaridagi ma’lumotlarni qayta ishlash tezligi bir xil bo‘ladi.

Haqiqiy ma’lumotlar bilan ishlashda moslik matematik funksiyalari ishlatilishi mumkin (6.5-jadval). Bu jadvalda REAL ixtiyoriy haqiqiy turni, INTEGER — ixtiyoriy butun turni bildiradi.

6.5-jadval

Turbo Paskalning matematik funksiyalari

Murojaat	Parametr turi	Natija turi	Izoh
Abs (x)	Real, Integer	Argument turi	Argument moduli
ArcTan (x)	Real	Real	Arktangens, radianlardagi qiymati
cos (x)	Real	Real	Kosinus, radianlardagi burchak
exp (x)	"	"	Eksponenta
frac (x)	"	"	Sonning kasr qismi
int (x)	"	"	Sonning butun qismi
ln (x)	"	"	Natural logarifm
pi	—	"	π=3.141592653...
Random	—	"	0 ... [1] sohada tekis taqsimlangan psevdotasodifiy son
Random (x)	Integer	Integer	0 … (x—1) sohada tekis taqsimlangan psevdotasodifiy son
Randomize	—	—	Tasodifiy sonlar generatorining initsiatsiyasi
sin (x)	Real	Real	Sinus, radianlardagi burchak
sqr (x)	Real	Real	Argument kvadrati
sqrt (x)	"	"	Kvadrat ildiz

Nazorat savollari

1. Oddiy turlarga qanday turlar kiradi?
2. Tartibli turlar bilan haqiqiy turlar o‘rtasida qanday farq bor?
3. Tartibli turlarga qanday turlar kiradi?
4. Tartibli turlarga qanday funksiyalarni qo‘llash mumkin?
5. Butun turlar qanday turlar, ularning qiymat sohalari qaysi chegaralarda bo‘ladi?
6. Butun turlarga qaysi protsedura va funksiyalar qo‘llaniladi?
7. Mantiqiy turlar qanday turlar, ularga qanday qoidalar xos?
8. Qanday turlar ramzli deyiladi?
9. Xizmat kodlari belgilari qanday belgilar?
10. CHAR turiga qanday munosabat amallari va funksiyalar qo‘llaniladi?
11. Sanab o‘tiladigan turlar qaysi tomoni bilan boshqa turlardan farqlanadi?
12. Sanab o‘tiladigan turlarni qo‘llashdan maqsad?
13. Tur-soha qanday tur, uning berilishi.
14. Tur-sohalar bilan qanday funksiyalar ishlaydi?
15. Haqiqiy turlar qanday turlar?
16. Haqiqiy turlar ShK xotirasida qanday tarkibga ega?
17. Haqiqiy turlarning qanday qiymat turlari bor?
18. Haqiqiy turlar bilan ishlaganda ShK tarkibiga nega arifmetik soprotsessor kiritilishi kerak?

VII bob. ***TURBO PASKAL*** **MUHITIDA ISHLASH**

7.1. ***Turbo Paskal*****da ishni boshlash tartibi**

Turbo Paskal tizimi bir nechta distributiv disketlarda beriladi va qattiq diskka o‘rnatiladi. Qattiq diskda tizim ochilganda, odatda, TP (yoki PAS, TURBO PAS, PASCAL va h.k.) nomli katalog hosil qilinadi, bu katalogga distributiv disketdagi hamma fayllar joylashtiriladi. *Turbo Paskal* muhitini ishga tushirish uchun shaxsiy kompyuterning kataloglar shajarasida bu katalogni va undagi TURBO.EXE faylini topish kerak. Bu fayl *Turbo Paskal*ning ishga tayyor dasturlash muloqot tizimiga ega. Unga *Turbo Paskal*ning minimal miqdordagi zaruriy qismlari (matn muharriri, kompilator, joylashtiruvchi, yuklovchi) kiradi. Shuningdek, muloqot muhitida normal ishlash uchun TURBO. TPL faylida asosiy kutubxona va yordam xizmati (TUPBO. HLP) ham bo‘lishi kerak. Ko‘pgina misollarining yozilishi, tarjima (kompilatsiya) qilinishi va bajarilishi uchun shu fayllar yetarli.

Sanab o‘tilgan fayllar D diskning TP katalogida joylashgan bo‘lsin. Unda *Turbo Paskal*ni chaqirish uchun:

D:\TP\TURBO buyrug‘ini berish kerak.

Bu buyruq bo‘yicha MS-DOS operatsion tizimi TURBO. EXE faylidan dasturni bajarishga qo‘yadi: dasturni tezkor xotiraga yuklaydi va unga boshqarishni uzatadi.

Yuqorida sanab o‘tilgan fayllarga ega katalog *tizim (sistema) katalogi* deyiladi. Tizim katalogini joriy katalog qilib ishlatish mumkin emas. Chunki, birinchidan, dasturlash tizimidagi qandaydir bir faylni o‘chirib qo‘yib, uning ish qobiliyatini buzish mumkin, ikkinchidan, bu katalog tezda *Turbo Paskal*ga tegishli bo‘lmagan boshqa fayllar bilan to‘lib qolishi mumkin. Yana bir sababi shuki, *Turbo Paskal* TURBO.TP va TURBO.PSK nomli fayllar orqali ishga tayyor bo‘lish xususiyatiga ega.

Tizim chaqirilganda u joriy katalogdan bu fayllarni izlay boshlaydi. Agar bu xususiy katalog bo‘lsa, tizim har safar foydalanuvchining xohishiga ko‘ra ishga tayyorlanadi. Agar bu fayllar katalogda topilmasa (*Turbo Paskal*ga birinchi murojaatda shunday bo‘ladi ham), sistema izlashni tizim katalogida davom ettiradi. Ularni u yerda topolmagach, standart tarzida ishga tayyorlanadi. Keyinchalik foydalanuvchi ishga tayyorlash fayllarini shaxsiy katalogga saqlab qo‘yib, shu bilan u har safar tizimga murojaat qilingandan keyin uni qayta ishga tayyorlash zaruriyatidan qutulishi mumkin.

Sistema muvaffaqiyat bilan chaqirilgach, shaxsiy kompyuter ekrani birinchi 7.1-rasmda ko‘rsatilgan ko‘rinishni oladi:

7.1-rasm. Ekranning *Turbo Paskal*ni chaqirgandan keyingi ko‘rinishi.

*Turbo Paskal*dan chiqish uchun Alt klavishini, uni qo‘yib yubormasdan turib, *X* lotin harfli klavishni bosish va har ikkala klavishni qo‘yib yuborish kerak.

Yuqori satr *Turbo Paskal*ning mumkin bo‘lgan ish rejimlari «menyu»sidan, quyi satr asosiy funksional klavishlarning qo‘llanilishi to‘g‘risidagi qisqacha ma’lumotdan iborat. Ekranning qolgan qismi, ikkilamchi ramka bilan chegaralangan va dastur matnini kiritish hamda tahrirlash uchun mo‘ljallangan, tahrir oynasiga tegishli. Uning yuqori satrida dastur matni o‘qilgan disk faylining nomi (yangi faylga NONAME.PAS ismi beriladi) va «sichqon» — kiritish qurilmasi bilan ishlashda foydalaniladigan maydon (bu

maydonlar kvadrat qavslar bilan belgilangan) hamda 1 raqami — oyna raqami keltiriladi. *Turbo Paskal*da bir vaqtda bir nechta dasturlar (yoki bitta yirik dasturning qismlari) bilan ishlash mumkin. Ularning har biri alohida tahrir oynasida bo‘lishi mumkin. Muhit bir vaqtda 9 tagacha tahrir oynasini ishlatishga imkon beradi.

*Turbo Paskal*da tahrir oynasi (oynalari)dan tashqari tahrir rejimi, dastur ishi natijasini chiqarish, yordam, registrlar xizmati oynalari ishlatiladi. Xohishga ko‘ra ular ekranga navbat bilan chaqiriladi yoki unda bir vaqtda mavjud bo‘ladi.

7.2. Funksional klavishlar

Funksional klavishlar *Turbo Paskal* muhitini boshqarish uchun ishlatiladi. Ular *F1, F2, ... F12* deb belgilanadi va klaviaturaning eng yuqori registrida joylashgan bo‘ladi. Bu klavishlarning har biri bilan menyuning qandaydir bir buyrug‘i bog‘lanadi. Deyarli hamma klavishlarning ishlashini uchta klavishlar bilan takomillashtirish mumkin. Bu klavishlar quyidagilar: *Alt* (*Alternative* — qo‘shimcha degan so‘zdan), *Ctrl* (*Control* — boshqaruvchi) va *Shift* (SHIFT — siljituvchi). Bu klavishlar yozuv mashinkasida registrni vaqtincha almashtirish klavishlari kabi ishlatiladi: ularning bittasini bosish va uni qo‘yib yubormasdan turib, funksional klavishni bosish kerak. Bundan keyin ikki klavishning bunday birgalikda bosilishini chiziq orqali ko‘rsatamiz. Masalan, *Alt-F3 Alt* klavishi bilan birgalikda *F3* klavishi ham bosilishi kerakligini bildiradi.

Quyida funksional klavishlar va ularning *Ctrl* hamda *Alt* klavishlari hamkorligida *Turbo Paskal* muhitida uzatiladigan buyruqlar keltiriladi:

F1 — yordam xizmatiga ma’lumot olish uchun murojaat qilish (*Help* — yordam);

F2 — tahrirlanadigan matnni diskli faylga yozish;

F3 — diskli fayldagi matnni muharrir oynasida o‘qish;

F4 — tuzatish (yaxshilash) rejimida ishlatiladi: dastur bajarilishini boshlash yoki davom ettirish va kursor turgan satrning bajarilishi oldidan to‘xtash;

F5 — faol oynani butun ekranga yoyish;

F6 — navbatdagi oynani faollashtirish;

F7 — tuzatish rejimida ishlaydi: dasturning keyingi satrini bajarishni; agar satrda protsedura (funksiya)ga murojaat bo‘lsa, bu

protsedura (funksiya)ga kirish va uning birinchi operatori bajarilishi oldidan to‘xtashni bildiradi.

F8 — tuzatish rejimida ishlatiladi: dasturning keyingi satrini bajarishni; agar satrda protsedura (funksiya)ga murojaat bo‘lsa, uni bajarish va uning ishini kuzatmaslik kerakligini bildiradi;

F9 — dasturni bajarmaslik, lekin uni kompilirlash kerakligini bildiradi;

F10 — bosh menyu yordamida ishning muloqot tanlov rejimiga o‘tishni kerakligini bildiradi;

Ctrl-F9 — dasturni o‘tkazishni bajarish: muharrirdagi dasturni kompilirlash, tezkor xotiraga yuklash va bajarish, shundan keyin *Turbo Paskal* muhitiga qaytishni bildiradi;

Alt-F5 — muharrir oynasini dastur ishi natijalarini chiqarish oynasi bilan almashtirishni bildiradi.

Qisqa izohlarni keltiramiz.

Dastur ishini tekshirish uchun *Ctrl-F9*, *Turbo Paskal*dan chiqish uchun *Alt-X* buyruqlari kerak bo‘ladi. *F2* va *F3* klavishlar shaxsiy kataloglar bilan ishlashda yordam beradi. *Alt-F5* buyrug‘i yordamida, ixtiyoriy daqiqada, dasturni ishlatish natijasida olingan ma’lumotlarni ekranda ko‘rish mumkin bo‘ladi.

7.3. Matn muharriri

Turbo Paskal matn muharriri foydalanuvchiga dastur matnlarini yaratish va tahrir qilish uchun qulay vositalar beradi. Tahrir oynasida katta bo‘lmagan imlovchi to‘rtburchak (kursor)ning mavjudligi muhit tahrirlash holatida ekanligi belgisidir. Tahrir rejimi *Turbo Paskal* yuklanishi bilan avtomatik tarzda o‘rnatiladi. Tahrirlash rejimidan *Turbo Paskal*ning ixtiyoriy boshqa rejimiga funksional klavishlar yordamida yoki bosh menyudan kerakli rejimini tanlash bilan o‘tish mumkin. Agar muhit menyudan tanlash holatida bo‘lsa, kursor yo‘qoladi, menyu satrida esa kodli so‘zlarda (menyu opsiyalarida) rangli to‘rtburchakli ko‘rsatkich paydo bo‘ladi.

Bosh menyuning rejim tanlash holatidan tahrir holatiga o‘tish uchun ESC (ESCape — qochish) klavishini, bosh menyudan tanlashga o‘tish uchun *F10* klavishini bosish kerak.

Matn muharrir bilan ishlashning asosiy yo‘llarini ko‘ramiz.

Dastur matnini yaratish uchun bu matnni shaxsiy kompyuter klaviaturasidan kiritish kerak. Navbatdagi satr to‘lishi bilan kursorni

keyingi satrga o‘tkazish uchun (kursor hamma vaqt ekranda dasturning navbatdagi belgisi yozilishi kerak bo‘lgan joyini ko‘rsatadi) *Enter* klavishi bosiladi.

Tahrir oynasi uzun, yetarli darajada keng, bir bo‘lagi ekranda ko‘rinib turgan qog‘oz varag‘ini ifodalaydi. Agar kursor quyi chegaraga yetgan bo‘lsa, tahrir oynasini o‘rash amalga oshiriladi: u bir satr yuqoriga ko‘tariladi va pastdan varaqning yangi satri paydo bo‘ladi. Agar kursor ekranning o‘ng chegarasiga yetsa, oyna, belgilarning kiritila borishi bilan, o‘ngga siljib boradi. Varaqning gorizontal va vertikal o‘lchamlari fayldagi belgilarning umumiy soni bilan (ular 64535 dan ko‘p bo‘lmasligi kerak) cheklanadi, lekin *Turbo Paskal* kompilatori uzunligi 126 belgidan katta bo‘lmagan dastur satrlarini qabul qiladi.

Oynani varaqqa nisbatan quyidagi klavishlar yordamida siljitish mumkin:

PageUp — bir bet yuqoriga;
PageDown — bir bet pastga;
Home — joriy satr boshiga;
End — joriy satr oxiriga;
Ctrl-PageUp — matn boshiga;
Ctrl-PageDown — matn oxiriga.

Kursorni o‘tkazish klavishlari (yo‘nalish klavishlari) bilan uni ekran bo‘yicha siljitish mumkin. Oyna chegarasiga yetishi bilan u satrga yoki belgiga siljiydi.

Xatoliklarni o‘chirish uchun *Backspace* (kursorning chap tomonidagi belgini o‘chiradi), *Delete* (kursor ko‘rsatayotgan belgini o‘chiradi) va *Ctrl-Y* (kursor turgan butun satrni o‘chiradi) klavishlari ishlatiladi.

Turbo Paskal muhiti har bir satr oxirida *Enter* klavishi bilan ko‘rinmas ajratuvchi — belgi qo‘yadi, bu belgi *Backspace* yoki *Delete* klavishlari yordamida o‘chiriladi. Ajratuvchi — belgini o‘chirish (yoki kiritish) bilan satrlarni ulash (yoki kesish) mumkin. Satrni kesish uchun kursorni kerakli joyga qo‘yish va *Enter*ni bosish, qo‘shni satrlarni ulash uchun kursorni birinchi satr oxiriga o‘rnatib (buning uchun *End* klavishdan foydalanish qulay), *Delete* yoki kursorni ikkinchi satr boshiga (*Home* klavishi yordamida) o‘rnatib, *Backspace* klavishlarini bosish kerak.

Muharrirning me’yoriy ish rejimi — oraga kiritish rejimidir, bu rejimda har bir yangi kiritiladigan belgi, ekranda matnni, «itarib» satr qoldig‘ini o‘ngga siljita borib, o‘zi uchun joy ochadi. Faqat shu

rejimdagina matnni kesish va tushirilgan satrlarni kiritish mumkin. Muharrir, shuningdek, mavjud eski matn ustiga yangi belgilarni yozish rejimida ham ishlashi mumkin; bu rejimda kursor ko‘rsatayotgan eski belgini yangi belgiga almashtiradi, satr qoldig‘i esa siljimaydi. Bir rejimdan ikkinchi rejimga o‘tish uchun *Insert* klavishini bosish kerak. Muharrirning qaysi rejimda ishlayotganini kursor shaklidan bilish mumkin: oraga kiritish rejimida kursor yorug‘ ostiga chizish belgisiga o‘xshaydi, ustiga yozish rejimida esa kursor belgini butunicha to‘sib turuvchi, yirik yorug‘ to‘rtburchakni ifodalaydi.

Odatda, muharrir avtochetlanish rejimida ishlaydi. Bu rejimda har bir yangi satr ekranda oldingi satr qaysi xonadan boshlangan bo‘lsa, xuddi shu xonadan boshlanadi. Bu rejim dastur matnini yaxshi ko‘rinishda rasmiylashtirishga ko‘maklashadi: chap chegaradan chetlanishlar shartli yoki murakkab operator tanasini ajratadi va dasturni ancha ko‘rgazmali qiladi. *Ctrl-O I* (*Ctrl* klavishi bosib turilib avval lotincha O harfli klavish bosiladi, keyin O qo‘yib yuborib, *I* harfli klavish bosiladi) buyrug‘i bilan avtochetlanishdan voz kechish mumkin, *Ctrl-OI* ni takroriy bosish avtochetlanish rejimini tiklaydi.

Quyida *Turbo Paskal*ning eng ko‘p ishlatiladigan buyruqlari keltirilgan:

Kursorni siljitish

PageUp — bir bet yuqoriga;
PageDown — bir bet pastga;
Home — joriy satr boshiga;
End — joriy satr oxiriga;
Ctrl-PageUp — matn boshiga;
Ctrl-PageDown — matn oxiriga.

Tahrir buyruqlari

Backspace — kursordan chapdagi belgini o‘chiradi;
Delete — kursor ko‘rsatgan belgini o‘chiradi;
Ctrl-Y — kursorli satrni o‘chiradi;
Ctrl-Q L — o‘zgartirilgan satrni tiklaydi (agar kursor satrdan, u o‘zgartirilgandan keyin ketmagan bo‘lsagina ta’sir ko‘rsatadi);
Enter — eski satrni qirqib, yangisini kiritadi.

Bloklar bilan ishlash

Ctrl-K B — blokni ajratishni boshlaydi;

Ctrl-K K — blokni ajratishni tugatadi;
Ctrl-K Y — ajratilgan blokni yo‘q qiladi;
Ctrl-K C — blokni nusxalaydi;
Ctrl-K W — blokni faylga yozadi;
Ctrl-K R — blokni fayldan o‘qiydi;
Ctrl-K P — blokni chop etadi.

7.4. *Turbo Paskal* muhitida ishlashning asosiy yo‘nalishlari

7.4.1. Fayllar bilan ishlash. Yuqorida bayon etilganidek, *Turbo Paskal* ishga tushirilishi bilan muhit matnni tahrirlash rejimiga o‘tadi. Bu rejimda yangi dasturni tayyorlash yoki mavjud dasturni o‘zgartirish mumkin.

Dastur matnlarini muhitdan tashqarida saqlashning asosiy ko‘rinishi fayllardir. *Turbo Paskal* bilan ish tugatilgach, yangi dastur matnini, yana keyinchalik ishlatish maqsadida, disk faylida saqlash mumkin. Disk fayllari va muhit tahriri o‘rtasida ma’lumotlarni almashtirish maqsadida *F2* (faylga yozish) va *F3* (fayldan o‘qish) klavishlari belgilangan. Yangi yaratilgan dastur joylashtiriladigan faylga, ism hali berilmagan bo‘lsa, muhit NONAMEOO.PAS (NONAME — ismi yo‘q) degan standart ism beradi. Faylda dastur matnini saqlab qolish uchun *F2* klavishini bosish kerak. Shu lahzada muhit dastur ismini tekshiradi, agar u standart NONAME ismi bo‘lsa, uni o‘zgartirish kerakligini so‘raydi, ekranda quyidagi yozuvli katta bo‘lmagan so‘rov oynasi paydo bo‘ladi:

SAVE File as
(faylda ... ism bilan saqlash kerakmi?)

Quyida esa fayl ismini yozish uchun maydon berilgan bo‘ladi, bu maydonga kerakli ismni yozish va *Enter* klavishini bosish (matn faylda saqlanadi) mumkin. Agar ismda kengaytma tushirib qoldirilgan bo‘lsa, muhit faylga .PAS — standart kengaytmani beradi. Agar foydalanuvchi *Turbo Paskal* bilan ishni tugatmoqchi bo‘lsa, tahrirda esa faylda saqlanmagan matn qolsa, ekranda savolli oyna paydo bo‘ladi:

NONAMEOO.PAS has been modifed. Save?
(*NONAMEOO.PAS* fayli o‘zgartirildi. Saqlaymi?)

Javob sifatida, agar matnni faylda saqlash kerak bo‘lsa, *Y* (*Yes* — Ha), aks holda *N* (*No* — Yo‘q)ni bosish kerak.

7.4.2. Dasturni o‘tkazish va tahrir qilish. Dastur matni tayyorlangach, uni bajarib ko‘rish, ya’ni dasturni kompilatsiya qilib, uni standart protsedura va funksiyalar kutubxonasi bilan bog‘lash (agar bu zarur bo‘lsa), dasturni tezkor xotiraga yuklash va uni boshqarishga uzatish kerak. Bunday ishlarning ketma-ketligi dasturni o‘tkazish deyiladi va *Ctrl-F9* buyrug‘i bilan amalga oshiriladi.

Agar dasturda sintaksis xatolar bo‘lmasa, hamma ishlar ketma-ket bajarib boriladi, bunda katta bo‘lmagan oynada kompilatsiya qilingan satrlar soni va tezkor xotiraning bo‘sh sig‘imi to‘g‘risida axborot beriladi. Boshqarishni yuklangan dasturga berishdan oldin muhit ekranni tozalaydi (aniqrog‘i, ekranga dasturni o‘tkazish oynasini chiqaradi), dastur ishini tugatib bo‘lgach, kompyuterni boshqarishni yana o‘z zimmasiga oladi va ekranda qayta tahrir oynasini tiklaydi.

Agar qandaydir bir bosqichda muhit xatolik topsa, u o‘zining keyingi ishini to‘xtatadi, tahrir oynasini tiklaydi va kursorni dasturning kompilatsiya yoki bajarish vaqtida xatosi topilgan satrga joylashtiradi. Bunda tahrirning yuqori satrida xato sababi to‘g‘risidagi tashxis axboroti chiqadi. Bularning hammasi dasturni zudlik bilan yaxshilashga, ya’ni undagi sintaksis xatolarni tuzatishga va uning to‘g‘ri ishlayotganiga ishonch hosil qildirishga imkon beradi. Agar xatolik dasturni o‘tkazish bosqichida paydo bo‘lgan bo‘lsa, xatolik topilgan joyning ko‘rsatilishi kerakli axborotni bermasligi mumkin, chunki xatolik berilgan dasturning oldingi operatorlarida noto‘g‘ri tayyorlanganligining natijasi bo‘lishi mumkin.

Masalan, xatolik manfiy sondan kvadrat ildiz chiqarilishi natijasida hosil bo‘lgan bo‘lsa, ildiz chiqarilayotgan operator ko‘rsatiladi, vaholanki, xatolikning dastlabki sababini qayerdadir oldindan, tegishli o‘zgaruvchiga manfiy qiymat berilgan joydan izlash kerakligi aniq. Bunday holatlarda, odatda, dasturni *F4, F7* va *F8* klavishlari bilan bog‘liq buyruqlar yordamida bajarishga o‘tiladi. Tahrirlashning yetarli ko‘nikmasini olgunga qadar bitta *F7* klavishidan foydalanish mumkin. Bu klavish bosilgach, muhit kompilatsiya, kompanovka (standart protsedura va funksiyalar kutubxonasi bilan bog‘lanish) va dasturni yuklash ishlarini amalga oshiradi, shundan keyin dasturni o‘tkazishni birinchi operator bajarilishi

oldidan to‘xtatadi. Bu operatorga ega dastur satri ekranda ajratilgan (rangi bilan) bo‘ladi. Endi *F7* ning har bir yangi bosilishi joriy satrda dasturlashtirilgan hamma operatorlarning bajarilishiga va ko‘rsatkich (kursor)ning keyingi satrga siljishiga olib keladi.

Dasturning shubhali joyida o‘zgaruvchi yoki ifodaning joriy qiymatini ko‘rish mumkin. Buning uchun kursorni foydalanuvchini qiziqtiradigan o‘zgaruvchi oldiga eltib qo‘yish va *Ctrl-F4* klavishlarini bosish kerak. Ekranda 3 ta maydondan iborat muloqot oynasi paydo bo‘ladi. Yuqori maydonda o‘zgaruvchi ismi bo‘ladi, 2 ta boshqa maydon bo‘sh bo‘ladi. O‘rta maydonda o‘zgaruvchining joriy qiymatini hosil qilish uchun *Enter* klavishi bosiladi. Agar *Ctrl-F4* ni bosishdan oldin kursor satrning bo‘sh joyida turgan yoki boshqa o‘zgaruvchi nomini ko‘rsatgan bo‘lsa, muloqot oynasining yuqori maydoni ham bo‘sh bo‘ladi yoki shu boshqa o‘zgaruvchining ismiga ega bo‘ladi. Bunday paytda klaviaturadan kerakli o‘zgaruvchi nomini kiritish va *Enter*ni bosish kerak. Xuddi shunday tarzda faqat kuzatilayotgan o‘zgaruvchilarning ismlarinigina emas, balki ifodalarni ham kiritish mumkin, muhit kiritilgan ifodaning qiymatini hisoblaydi va ko‘rsatadi.

7.4.3. Turbo Paskalning yordam xizmati. Turbo Paskal muhiti tarkibining ajralmas qismi uning yordam xizmatidir. Qiynalgan paytda unga kirish uchun *F1* klavishini bosish yetarli, ekranga zaruriy ma’lumotlar chiqadi. Bu ma’lumotlar muhitning joriy holatiga bog‘liq. Masalan, muhit xatoni topgan vaqtda *F1* bosilsa, xatolik sabablari to‘g‘risida qo‘shimcha ma’lumotlar va ulardan qutulish yo‘llari beriladi.

Yordam xizmatiga bevosita tahrir oynasidan murojaat qilishning to‘rt usuli bor:

F1 — kontekst-bog‘liq ma’lumotni olish;

Shift-F1 — ma’lum yordam axborotlari ro‘yxatidan ma’lumot tanlash;

Ctrl-F1 — kerakli standart protsedura, funksiya, standart o‘zgarmas yoki o‘zgaruvchi to‘g‘risida ma’lumot olish;

Alt-F1 — oldingi ma’lumotni olish;

Shift-F1 buyrug‘i bilan ekranda yordam ma’lumoti olish mumkin bo‘lgan standart protseduralar, funksiyalar, turlar, o‘zgarmaslarning imlo bo‘yicha tartibga solingan ro‘yxatiga ega oyna paydo bo‘ladi.

Bu ma'lumotni boshqacha ham olish mumkin. Ekranga protsedura (funksiya va hokazo)ning ismi yoziladi yoki kursor matnda mavjud standart ismga keltiriladi va *Ctrl-F2* bosiladi. Muhit kursorning yaqin atrofini tahlil qiladi, ismni ajratadi va kerakli ma'lumot beradi.

Ko'p hollarda ma'lumot *Turbo Paskal*ning tegishli imkoniyatlarini ko'rsatuvchi katta bo'lmagan misolga ega bo'ladi. Misolni ma'lumotnomadan «kesib olish» va tahrir oynasiga keltirish mumkin. Buning uchun ma'lumotnoma chaqirilgach, *Alt-E* bosiladi, paydo bo'lgan qo'shimcha menyudan *Copy examples*ning (misollarni nusxalang) davomi tanlanadi va *Enter* bosiladi, misol matni muharrirning ichki buferiga nusxalanadi. Misolni buferdan olish uchun *Esc* bosiladi. Yordam xizmatidan chiqish uchun kursor tahrir oynasidagi bo'sh satrga keltiriladi va *Shift-Insert* (bufer tarkibini dastur matniga nusxalash) va nusxalangan matnning ajratuvchi rangini olib tashlash uchun *Ctrl-K H* bosiladi.

7.4.4. Turbo Paskalda dasturlar bilan ishlash. Shaxsiy kompyuter ekraniga ma'lum bir axborotni chiqarishni amalga oshiruvchi murakkab bo'lmagan dasturni tuzishga harakat qilib ko'ramiz. Masalan, bu «Men *Turbo Paskal*da dastur tuzaman» jumlasi bo'lsin. Bunday dastur quyidagi ko'rinishga ega bo'lishi mumkin:

```
Program My_First_Program;
const
        Text = 'Men Paskalda dastur tuzaman',
begin
        WriteLn (Text);
end.
```

Eng avvalo, matnni ifodalash shaklini tahlil qilamiz. Dastur 6 qatordan iborat. Odatda, dastur satrlari matnning qandaydir ma'noli bo'laklarini ajratadi va dasturdagi ma'lum faoliyat bilan bog'liq bo'lmaydi, ya'ni dastur matnini satrlar bo'yicha joylashtirish, til sintaksisi talabi bilan emas, balki dasturchining xohishi bilan bo'ladi, masalan, shu dasturni quyidagicha ham yozish mumkin edi:

*Program My_First_Program; const Text = '*Men Paskalda dastur tuzaman'; *begin WriteLn (Text); end.*

Dasturlashning boshqa ba'zi bir tillaridan farq qilib, *Turbo Paskal*da bo'sh xona («probel») tilning alohida konstruksiyalarini

ajratuvchi sifatida ishlatiladi, shuning uchun quyidagi dastur xato bo‘lib hisoblanadi:

*ProgramMy_First_Program; constText= ′*Men Paskalda dastur tuzaman′; *beginwriteln(Text); end.*

*Turbo Paskal*da harflar balandliklari (bosh va kichik), agar ular matn o‘zgarmaslari bilan bog‘liq bo‘lmasa, e’tiborga olinmaydi. Dastur boshi, masalan, quyidagi ko‘rinishga ega bo‘lishi mumkin:

program my_first_program;

Endi har bir satrlar ma’nosiga to‘xtalamiz. Birinchi

Program My_First_Program;

satri *Program* so‘zi bilan boshlanadi va dastur ismini e’lon qiluvchi mazmunga ega bo‘ladi. *Program* so‘zi *Turbo Paskal*da rezervlangan, ya’ni u dastur ismini e’lon qilishdan boshqa maqsadlarda ishlatilishi mumkin emas. *Turbo Paskal*da ko‘plab rezervlangan so‘zlar to‘plami (II bob) bor. Ularning hech birini dasturchi biron-bir obyektning (o‘zgaruvchi, o‘zgarmas va h.k.) identifikatori (ismi) sifatida foydalanishi mumkin emas. *Turbo Paskal* muhiti muharriri, odatda, rezerv so‘zlarni dasturda rangi bilan ajratib ko‘rsatadi. Dastur ismi boshqa ishlatilmaganligi uchun uni e’lon qilishni talab qilish ortiqcha, shuning uchun bu satrning tushirib qoldirilishi dastur bajarilishiga hech qanday ta’sir etmaydi. Ko‘rilayotgan dasturning *My_First_Program* ismining inglizcha tarjimasi «Mening Birinchi Dasturim» degani. Ism bir butun so‘z (bo‘sh xonalarsiz) dan iborat bo‘lishi kerak. Shuning uchun bo‘sh xonalar o‘rnida — identifikatorlarda ostiga chizish belgisini ishlatish mumkin.

Birinchi satr nuqtali vergul (;) ajratuvchi bilan tugaydi. *Turbo Paskal* tilida bu ajratuvchi operator yoki bayon etishning oxirini anglatadi. Bu ajratuvchining ishlatilishi bitta satrda bir nechta operatorlarni joylashtirishga imkon beradi.

Ikkinchi satr yagona *const* rezerv so‘zidan iborat. U bundan keyin o‘z qiymatini o‘zgartirmaydigan bir yoki bir nechta o‘zgarmaslarning bayoni kelishini bildiradi. Dasturlash boshqa ko‘plab tillardan farq qilib, o‘zgarmas o‘z ismiga ega bo‘lishi mumkin. Masalan, π=3.14159265. Dastur ishlayotganda o‘zgarmasning *pi* ismi (π grekcha harf, shuning uchun u lotinchada pi deb yoziladi) kompilator tomonidan uning qiymatiga almashtiriladi.

*Turbo Paskal*da o‘zgarmasni e’lon qilish — uning ismi va qiymatini ko‘rsatishni bildiradi. Uchinchi satr bunday ko‘rsatmaga ega:

Text = ′Men Paskalda dastur tuzaman′.

Bu satr *Text* ismli o‘zgarmas «Men Paskalda dastur tuzaman» belgilar satridan iborat qiymatni o‘zlashtirishini bildiradi.

*Turbo Paskal*da har xil turdagi — butun yoki haqiqiy sonlar, ramzli, belgilar satri, massivlar va h.k. o‘zgarmaslar ishlatilishi mumkin. Satrni o‘z ichiga olgan ikki tuturiq *Text* — belgilar satri turidagi o‘zgarmas ekanligini bildiryapti. Tuturiqlar bu satrga tegishli bo‘lmaydi, kompilatorga ular o‘z ichidagi ramzlarni bir butun— matnli o‘zgarmas, deb qarash kerakligini ko‘rsatadi. Agar tuturiqni ham matnli o‘zgarmasga kiritish kerak bo‘lsa, uni ikki marta ketmaket yozish yetarli.

Dastlabki uch qator dastur ishlashida qandaydir aniq faoliyat bilan bog‘langan emas. Ular kompilatorga dasturning o‘zi va unda ishlatiladigan obyektlar to‘g‘risida axborot berishadi. Dasturning bu qismi bayon etish bo‘limi deyiladi. To‘rtinchi satrdagi *begin* rezerv so‘zi kompilatorga dasturning boshqa, ya’ni operatorlar bo‘limi boshlanganligi to‘g‘risidagi axborotni beradi. Bizning misolda bu bo‘lim:

WriteLn (Text)

U axborotni kompyuter ekraniga chiqaradi.

Dasturning hamma ishini *end* rezerv so‘zi yakunlaydi, *end* so‘zidan keyingi nuqta kompilatorga dastur matnining tugaganini bildiradi, *end* so‘zidan keyin ixtiyoriy matnni joylashtirish mumkin, u kompilator tomonidan ishlab chiqilmaydi.

Umuman, Paskalda va, xususan, *Turbo Paskal*da, kiritish-chiqarishning maxsus operatorlari yo‘q. *Turbo Paskal* tilida yozilgan dasturlarda tashqi dunyo bilan axborot almashtirish uchun, maxsus standart protseduralar ishlatiladi. Shunday qilib,

WriteLn (Text);

operatori o‘z mohiyatiga ko‘ra ma’lumotlarni chiqarish protsedurasiga murojaat operatori bo‘lib hisoblanadi (WRITE LINE — satrni yozish demak).

Protsedura tushunchasi *Turbo Paskal*da markaziy tushunchalardan (VIII bob) biridir va u protsedura operatorlarining ketmaketligi bo‘lib, unga ismi bilan murojaat qilinadi.

WriteLn protsedurasi *Turbo Paskal*ning standart yoki o‘rnatilgan protseduralariga kiradi. Standart protsedurani oldindan bayon etishga zaruriyat yo‘q, u protseduraga ega bo‘lgan ixtiyoriy dastur uchun ishlaydi. Chiqarish operatori va chiqarish protsedurasiga

murojaat o‘rtasidagi farq shundan iboratki, chiqarish protsedurasining ismi *Turbo Paskal*dagi boshqa protseduralar ismi kabi rezerv so‘z emas, demak, foydalanuvchi *WriteLn* ismli o‘zining xususiy protsedurasini yozishi mumkin.

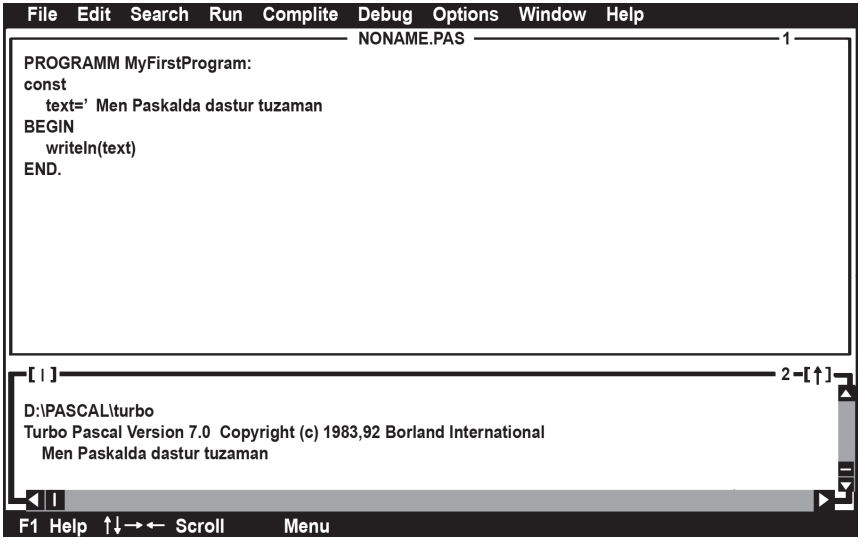
WriteLn protsedurasida ixtiyoriy sondagi parametrlarni ishlatish mumkin. Parametrlar protseduraga, darrov uning ismidan keyin, aylana qavslar ichida yozilgan ro‘yxat ko‘rinishida uzatiladi. Bizning misolda protseduraga yagona parametr — *text* o‘zgarmasi uzatiladi. *WriteLn* protsedurasiga murojaatda birinchi parametr sifatida axborotni qabul qiluvchining (chiqarish yo‘naltirilgan qurilma yoki diskli fayl) adresi ko‘rsatilishi mumkin. Shunday yo‘l bilan dasturchi ma’lumotlarni chiqarish adresini oson o‘zgartiradi. Agar, bizning misolimizdagi kabi, chiqarish adresi ko‘rsatilmagan bo‘lsa, chiqarish ekranga yo‘naltirilgan bo‘ladi.

Endi dasturni bajarishga urinib ko‘ramiz. Buning uchun matn terilgach, *Ctrl-F9* klavishlari bosiladi. Agar matn terilayotganda xatoga yo‘l qo‘yilmagan bo‘lsa, bir necha sekunddan keyin tasvir almashinuvi sodir bo‘ladi: dastur yuklanishi bilan *Turbo Paskal* ekranni, dasturchining ishlayotgan dasturi ixtiyoriga berish uchun, tozalaydi. Bunday ekran *dastur oynasi,* deb ataladi. Dastur ishini tugatib bo‘lgach, ekranda yana dastur matni yozilgan tahrir oynasi paydo bo‘ladi. Agar dastur oynasidagi tasvirlarni ko‘ra olmagan bo‘lsangiz, *Alt-F5* klavishlarini bosing. Ixtiyoriy klavish bosilgach, muhit ekranni tahrir oynasini qayta tiklash rejimiga o‘tkazadi.

Keyingi ishga o‘tishdan oldin *Turbo Paskal* muhitining ba’zi bir imkoniyatlari bilan batafsil tanishish foydali. Bosh menyudan tanlash rejimiga o‘tish uchun *F10* klavishini bosing, ko‘rsatkichni *Debug* (tuzatish) opsiyasiga keltiring va *Enter* klavishni bosing — ekranda shu opsiyaning ikkinchi darajali menyusi paydo bo‘ladi. Undan *Output* (dasturni chiqarish) opsiyasini topib, *Enter*ni bosing. Ekranda qaytadan dastur oynasi paydo bo‘ladi, lekin endi u ixtiyoriy klavishning bosilishi bilan yo‘qolmaydi — ekran bu oyna bilan doimiy bog‘liq bo‘ladi. Endi ekranda bir vaqtda ikki oynaning bo‘lishiga erishamiz: *F10* klavish bosiladi, *Window* opsiyasi tanlanib, *Enter* klavish bosiladi, ko‘rsatkich *Tile* opsiyasiga keltirilib, *Enter* yana bir marta bosiladi. Agar hamma ishlar to‘g‘ri qilingan bo‘lsa, ekran 7.2-rasmda ko‘rsatilgan ko‘rinishni oladi.

Dastur oynasini chizuvchi ikkilamchi ramka, aynan shu oyna, shu vaqtda faol ekanligini bildiradi. Tahrir oynasini faollashtiramiz:

Alt klavishini va uni qo‘yib yubormasdan, 1 raqamli (tahrir oynasi 1 raqamiga, dastur oynasi 2 raqamiga ega, ramkalarning yuqori o‘ng burchaklariga qarang) klavishni bosamiz. Endi dastur bilan keyingi ishlarni olib borish uchun hamma narsa tayyor.



7.2-rasm. Tahrir va dasturning oynali ekrani.

Ekranga chiqariladigan matnni o‘zgartirishga unnab ko‘ramiz. Masalan, uchinchi satr oxiridan nuqtali vergulni olib tashlaymiz, ya’ni:

Text = ′Men Paskalda dastur tuzaman′.

Dastur qayta ishga tushirilgandan keyin, *Ctrl-F9* klavishlari bosilgach, kompilator

Error 85: *″;″ecpected*

(85-xato: ″;″yo‘q),

deb axborot beradi, muharrir esa kursorni *begin* so‘zining birinchi belgisiga o‘rnatadi («;» belgi operator oxiridan xohlagancha bo‘sh xonalarni ajratishi mumkin, kompilator ajratuvchi belgini izlab, rezerv so‘z uchraguncha bu bo‘sh xonalarni o‘tkazib yuboradi, shuning uchun kursor *begin* so‘zi boshida turadi). Xato tuzatilib, dastur qayta ishga tushiriladi. Bu safar hamma ish talabdagiday ketadi, dastur oynasida matn o‘zgarmasdan berilgan belgilar to‘plamiga qat’iy mos keluvchi «′Men Paskalda dastur tuzaman′», matni paydo bo‘ladi.

7.4.5. EHMda Paskal tilida yozilgan dasturlar bilan ishlash ketma-ketligi. Yuqorida keltirilganlarni umumlashtirib, berilgan nazariy tushunchalarga tayangan holda, Paskal tilida yozilgan dasturlar bilan ishlashda eng ko‘p bajariladigan ishlarni tartibga solib, bajarilishi kerak bo‘lgan ishlarni yana bir bor quyida eslatib o‘tamiz:

Turbo Paskal bosh menyusi bilan ishlash.

F10 klavishi bosiladi.

Menyudan yoki muloqot oynadan, hech bir ish bajarmasdan, chiqish uchun *Esc* klavishi bosiladi.

Bosh menyu ixtiyoriy ichki menyusini tez ochish. *Alt* klavishi bosiladi va ushlab turiladi va qo‘yib yuborilmasdan menyu ismining birinchi harfi bosiladi. Masalan, *Alt* + *F* — ekranda *File* menyusini ochadi.

Menyu buyruqlarini tez chiqarish. Ochilgan menyu buyruqlarini zudlik bilan chiqarish uchun buyruq nomidagi rangi bilan ajralib turgan harf yozilgan klavishni bosish kerak. Masalan, ochilgan *File* menyusida «A» klavishi bosilsa, *Save AS...* buyrug‘i chiqariladi yoki bu tezkor klavishlar yordamida ham bajarilishi mumkin:

Alt + *F A*

Ishchi (joriy) katalogni o‘zgartirish

1. *File* menyusi ochiladi.
2. *Change dir...* buyrug‘i chiqariladi. Muloqot oyna ochiladi, uning *Directory name* maydonida joriy katalog nomi, *Directory tree* maydonida esa joriy diskning kataloglar shajarasi ko‘rsatilgan bo‘ladi.
3. *Directory name* maydoniga yangi joriy katalog nomi yoziladi, *TAB* klavishi yozilgan kataloglar shajarasida uning ismi ko‘rsatiladi.
4. Tanlangan katalog *Enter* klavishini bosish bilan mustahkamlanadi.
5. TAB klavishini bosish bilan kursor OK tugmaga o‘rnatiladi va *Enter* bosiladi.

Yangi dasturni kiritish uchun oyna ochish

1. *File* — menyu ochiladi.
2. *New* — chaqiriladi.

Diskka yangi dasturni yozish

1. *File* menyusini ochish.
2. *SaveAs*... buyrug'ini chiqarish. *Save FileA*s muloqot oynasi ochiladi, bu yerda *Save FileAs* maydoniga yangi dastur uchun fayl nomi biriktiriladi. Shuningdek, bu oynada yangi dasturni mavjud fayllardan birining nomi bilan atash uchun imkoniyat bo'ladi. Buning uchun TAB klavishini bosish (kursor *File* maydoniga o'tadi) va mavjud ro'yxatdan fayl ismini tanlash kerak.
3. *Enter* klavishi bosiladi. Mavjud ism bilan yozilayotganda tegishli ogohlantirish berilgan va faylni qayta yozish yoki uni tanlangan ism bilan yozishdan voz kechish imkoniyati bo'ladi.

Faylni tahrir qilish uchun ochish

1. *File* menyusini ochish.
2. *Open*... buyrug'ini chiqarish (*F3* klavish ham bu ishni bajaradi).
3. Ekranda *Open a File* oynasi paydo bo'ladi. *Name* maydoni *Files* maydoniga chiqariladigan fayllarni tanlash uchun shablonga ega bo'ladi. So'zsiz holda *.*Pas* shabloni tavsiya etiladi. Tegishli shablondagi fayllar ro'yxati *Files* maydoniga diskning joriy katalogdan mavjud bo'lgan fayllardan chiqariladi.
4. 4a. TAB klavishi bosiladi, kursor *File* maydoniga o'tadi, mavjud ro'yxatdan fayl nomi tanlanadi.
   4b. *Name* maydonidagi shablon o'zgartiriladi, shundan keyin 4a bosqichdagi ish bajariladi.
   4d. Ochiladigan faylning nomi bevosita *Name* maydoniga kiritiladi.
5. *Enter* klavishi bosiladi. 4a va 4b hollarda klavish ikki marta bosiladi.

Faol oynadagi dasturni eski nom bilan saqlash

I usul. *F2* klavishi bosiladi.
II usul. 1. *File* menyusi ochiladi.
2. *Save* buyrug'i chiqariladi.

Faol oynadagi dasturni bajarishga yuborish

Ctrl + *F9* klavishlari bosiladi.

Natijani chiqarish uchun ekranni tozalash

Ekranni tozalash uchun dasturda quyidagilarni bajarish zarur:
— *Uses* taklifida *Crt* standart moduli kiritiladi (*Uses Crt):*
— Dasturning operator bloki boshida *ClrScr* protsedurasi chiqariladi.
Misol:

```
Program PR;
Uses Crt;
var A: integer;
begin
      ClrScr;
      Readln (A);
      Writeln ('A=', A);
end.
```

Yangi dasturni yaratish va xatolarini tuzatishda bajariladigan ishlarning namunaviy rejasi

1. Yangi dastur kiritish uchun tahrirlashning yangi oynasini ochish.
2. Yangi dastur matnini terish.
3. Dastur matnini diskka yozish (yangi dastur matni ishga tushirilguniga F10/File/ SaveAs... buyruqlari orqali diskka yozilishi zarur).
4. Dasturni kompilatsiya (tarjima) qilish uchun yuborish (F10/Compile/Compile buyruqlari yoki Alt+F9 klavishlarini bosish bilan).
5. Agar dasturda imloviy xatoliklarga yo‘l qo‘yilgan bo‘lsa, ekranda tegishli axborot paydo bo‘ladi, kursor esa xatolik joyini ko‘rsatadi. Bunday paytda tahrir buyruqlari yordamida tuzatilishi, xotiraga yuborilishi (F10/File/Save buyruqlari yoki *F2* klavishini bosish bilan) va yana 4-va 5-bosqichlardagi ishlar dasturda xatoliklar qolmaguncha takrorlanishi kerak; Dasturni bajarishga yuborish (F10/Run/Run buyruqlari yoki *Ctrl+F9* klavishlarini bosish bilan).
6. Dastur natijalarini ko‘rish (F10/Debug/User Screen buyruqlari yoki *Alt + F5* klavishlarini bosish bilan).
7. Agar xato natijalar olingan bo‘lsa, yo‘l qo‘yilgan algoritmik xatolarni tuzatish va dasturni yana bajarishga yuborish kerak.
8. 4—7-bosqichlar to‘g‘ri yechimlar olinguncha takrorlanadi.
9. Tuzatilgan dastur diskda saqlanadi (*F2*).

Paskal tilidagi dastur bilan ishlashda tezkor klavishlar. Umumiy qo‘llanishdagi klavishlar

F10	— bosh menyuga kirish.
ESC	— dialog yoki menyu oynasini yopish.
Alt + x	— Paskal dasturi bilan ishlash oynasidan chiqish.
Ctrl + Break	— ishga tushirilgan dasturni to‘xtatib boshlang‘ich oynaga qaytish (ishga tushgan dastur qotib qolganda yoki yopiq siklga tushib qolinganda bu klavishlar bosiladi).
Print Screen	— ekran nusxasini printerda chop etish.
Pause	— ixtiyoriy klavish bosilguncha o‘zgarayotgan tasvirni ekranda to‘xtatib turish.

Yordam tizimi bilan ishlash klavishlari

F1 — yordam tizimining shu daqiqada faol bo‘lgan oyna yoki kursor ko‘rsatayotgan menyu buyrug‘i to‘g‘risidagi axborotini yoritadi.

F1 — (ikki marta) yordam tizimidan foydalanish to‘g‘risidagi ko‘rsatmani ekranga chiqaradi.

Ctrl + F1 — faol oynadagi kursor turgan ibora haqidagi axborotni ekranga chiqaradi.

Alt + F1 — *Help* oynasining oldingi oynasiga qaytish. Bu buyruqni ko‘p marta bajarilishi teskari tartibda *Help*ning 20 tagacha oxirgi oynalarini chiqaradi.

Shift + F1 — yordam tizimida mavjud bo‘lgan iboralarning imlo tartibdagi ro‘yxatiga ega bo‘lgan *Index* oynasini ekranga chiqaradi.

Fayllarni ochish, saqlash va tahrir oynalari bilan ishlash klavishlari

F2 — tahrirlashning faol oynasidagi dasturni diskdagi faylga eski nom bilan saqlash.

F3 — tahrirlash va ishga tushirish uchun zarur bo‘lgan faylni diskdan tanlash uchun *Open a File* muloqot oynasini chiqarish.

Alt + F3 — tahrirlashning faol oynasini yopish.

F6 — bir necha marta *F6* klavishning ketma-ket bosilishi ochilgan oynalar faolligi almashinuvining takrorlanishini keltirib chiqaradi.

Shift + F6 — *F6* klavishga o‘xshab ishlaydi, lekin oynalarni teskari ketma-ketlikda almashtirib boradi.

Alt + O — Window List muloqot oynasini ochadi. Unda Paskal ishga tushirilgan lahzadan ochilgan hamma oynalar ko‘rsatilgan bo‘ladi.

F5 — faol oynani to‘la ekran o‘lchamida ochadi, agar oyna shu o‘lchamda bo‘lsa, uning boshlang‘ich holatini tiklaydi.

Ctrl + F5 — faol oyna o‘lchamini va (yoki) uning ekrandagi o‘rnini o‘zgartirish. Oynani ekran bo‘yicha siljitish yo‘nalish klavishlari yordamida, o‘lchamni o‘zgartirish esa *Shift* va yo‘nalish klavishlarining bir vaqtda bosilishi bilan bajariladi.

Dastur matni bo‘lagi bilan ishlatish klavishlari

Shift + yo‘nalish klavishlari — dastur bo‘lagini ajratadi.

Shift + Del — dasturning ajratilgan bo‘lagini o‘chiradi va *Clipboard* buferiga joylashtiradi.

Ctrl + Ins — dasturning ajratilgan bo‘lagini *Clipboard* buferiga nusxalaydi.

Shift + Ins — *Clipboard*dagi ajratilgan bo‘lakni oynadagi kursor joylashgan xonaga qo‘yadi.

Ctrl + Del — ajratilgan bo‘lakni *Clipboard* buferiga uzatmasdan uchiradi.

Alt + BackSpace — tahrirlashdagi oxirgi ishni bekor qiladi.

Kompilatsiya va bajarishga yuborish klavishlari

Alt + F9 — tahrirlashning faol oynasidagi faylni kompilatsiya qilish.

F9 — ko‘p modulli dasturni *.exe* faylni yaratish bilan shartli kompilatsiya qiladi. Agar oxirgi kompilatsiya vaqtidan ba’zi bir modullarga o‘zgartirishlar kiritilgan bo‘lsa, faqat o‘zgartirilgan va ulardan bog‘liq bo‘lgan modullargina qayta kompilatsiya qilinadi. Oddiy dasturlar uchun xuddi shu ishni *Alt + F9* bajaradi.

Ctrl + F9 — tahrirlashning faol oynasidagi dasturni bajarishga yuborish.

Dasturlarni tuzatish klavishlari

Alt + F5 — dastur bajarilishi natijalarini ko‘rish.

F8 — dasturning qadam-baqadam bajarilishi. Protsedura va funksiyalarning chaqirilishi bitta operator (qadam) kabi bajariladi.

F7 — dasturning qadam-baqadam bajarilishi. Protsedura yoki funksiyalarni chaqirishda uning matniga kirish va operatorlarni qadam-baqadam bajarish.

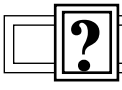
F4 — joriy satrdan kursor joylashgan satrgacha bo‘lgan dastur qismi qadam-baqadam bajariladi.

Ctrl + F2 — dasturni tahrir qilish ishini yakunlaydi va uni xotiradan bo‘shatadi.

Ctrl + F3 — *Call Stack* oynasini ochadi. Unda dasturning shu daqiqada bajarilayotgan protseduragacha chaqirilgan protseduralar ismlarining ketma-ketligi ko‘rsatiladi.

Ctrl + F4 — *Evaluate and modify* oynasini ochadi. Bu oynada qiymati aniqlanishi talab etilgan ifoda ko‘rsatiladi, dastur o‘zgaruvchilari va elementlarining qiymatlari qarab chiqiladi va ular o‘zgartiriladi.

Ctrl + F7 — *Add Watch* muloqot oynani ochadi. Bu oynaga tuzatishni bajarishda dasturchini qiymatlari qiziqtirayotgan ifodaning yoki o‘zgaruvchining ismini yozadi.



Nazorat savollari

1. *Turbo Paskal*da ish qanday boshlanadi?
2. *Turbo Paskal* ekrani ko‘rinishi qanday tashkil etuvchilardan iborat?
3. Funksional klavishlarning vazifasi nima va ular yordamida qanday buyruqlar ifodalanadi?
4. *Turbo Paskal* matn muharririning asosiy vazifasi nima va unda ishlash tartibi nimalardan iborat?
5. Matn muharririda ishlashning qanday rejimlari bor?
6. *Turbo Paskal*da fayllar bilan ishlashda qilinadigan asosiy ishlar nimalardan iborat?
7. Dasturni o‘tkazish va tahrir qilish qanday qilib amalga oshiriladi?
8. *Turbo Paskal*ning yordam xizmatidan qanday foydalaniladi?
9. *Turbo Paskal*da yoziladigan dasturga qo‘yiladigan shartlar nimalardan iborat?
10. *WriteLn* protsedurasi vazifasi va ishlatilish ko‘rinishlari.
11. Chiqarish operatori va chiqarish protsedurasiga murojaat o‘rtasida qanday farq bor?
12. *Debug, Output, Window, Tile* opsiyalarining mohiyati nimadan iborat?
13. *Turbo Paskal* bosh menyusi bilan qanday ishlar bajariladi?
14. Yangi dasturni yaratish va xatolarni tuzatishda ishlarning qanday ketma-ketligi bajariladi?

VIII bob. TIL OPERATORLARI

*Turbo Paskal*ning eng ko‘p ishlatiladigan operatorlaridan biri *o‘zlashtirish operatori* bilan tanishdik. Quyida tilning qolgan operatorlarini ko‘rib chiqamiz.

8.1. Murakkab va bo‘sh operator

Murakkab operator begin ... end operator qavslari (rezerv so‘zlar) ichida yozilgan dasturning ixtiyoriy operatorlar ketma-ketligidir. Murakkab operator *Turbo Paskal*ning muhim quroli bo‘lib, u strukturali dasturlashning zamonaviy texnologiyasi bo‘yicha (*GOTO* — o‘tish operatorisiz) dasturlar yozishga imkon beradi.

Turbo Paskal tili murakkab operator tarkibiga kiruvchi operatorlar xarakteriga hech qanday shartlar qo‘ymaydi. Ular ichida *Turbo Paskal*ning boshqa murakkab operatorlari ham bo‘lishi mumkin. Til operatorlari ularning ixtiyoriy chuqurlikda ichma-ich kelishiga yo‘l qo‘yadi:

```
begin;
        .....
            begin;
                .....
                    begin;
                        .....
                        .....
                    end;
                .....
            end;
        .....
end.
```

Aslida *begin ... end* so‘zlari bilan qamrab olingan operatorlarning

butun bo‘limi bitta murakkab operatorni ifodalaydi. Rezerv *end* so‘zi yopuvchi operator qavsi bo‘lgani uchun u bir vaqtda oldingi operatorning oxirini ham ko‘rsatadi, shuning uchun undan oldin «;» belgini qo‘yish shart emas (bundan keyingi misollarda shunday yo‘l tutamiz). Ba’zi misollarda *end*dan oldin nuqtali vergulning bo‘lishi, oxirgi operator va *end* operator qavsi o‘rtasida *bo‘sh operator* yotganligini bildiradi. *Bo‘sh operator* hech qanday ish bajarmaydi, faqat bunda dasturga ortiqcha nuqtali vergul kiritiladi. Bo‘sh operator, asosan, boshqarishni murakkab operator oxiriga uzatish uchun ishlatiladi.

8.2. Shartli operator

Shartli operator qandaydir shartni tekshirishga imkon beradi va tekshirish natijasiga ko‘ra u yoki bu ishni bajaradi. Shunday qilib, shartli operator — bu hisoblash jarayonining tarmoqlanish vositasidir.

Shartli operator quyidagi tasnifga ega:

IF <shart> THEN <operator1> ELSE <operator 2>,

bu yerda IF, THEN, ELSE — rezerv so‘zlar (agar, u holda, aksincha deb tarjima qilinadi);

<shart> — mantiqiy turdagi ixtiyoriy ifoda;

<operator 1>, <operator 2> — Turbo Paskalning ixtiyoriy operatorlari.

Shartli operator quyidagi algoritm bo‘yicha ishlaydi. Avval <shart> shartli ifoda hisoblanadi, agar natija TRUE (haqiqat) bo‘lsa, <operator 1> bajarilib, <operator 2> o‘tkazib yuboriladi; agar natija FALSE (yolg‘on) bo‘lsa, aksincha <operator 1> o‘tkazib yuborilib, <operator 2> bajariladi. Masalan,

```
var
        X,Y,max: Integer;
begin
        .....
        if X>max then
                Y: = max
else
                Y: = X;
```

Bu dastur bo‘lagi bajarilishi natijasida Y o‘zgaruvchi, agar u *max*dan oshmasa, X o‘zgaruvchi qiymatini, aks holda *max* qiymatni qabul qiladi.

Shartli operatorning ELSE <operator 2> qismi tushirib qoldirilishi mumkin. U paytda shartli ifodaning TRUE qiymatida <operator 1> bajariladi, aks holda bu operator o‘tkazib yuboriladi:

```
var
        X,Y,max: Integer
begin
        .....
        if X>max then
              max: = X;
        Y:=X.
```

Bu misolda *Y* o‘zgaruvchi hamma vaqt *X* o‘zgaruvchi qiymatini qabul qiladi, *max*da esa *X* ning maksimal qiymati eslab qolinadi.

<operator 1> va <operator 2> operatorlaridan ixtiyoriy biri, shu jumladan, shartli operator ham ixtiyoriy turda bo‘lishi mumkinligi uchun va bir vaqtda ichki shartlardan hammasi ham ELSE <operator 2> qismiga ega bo‘lmasligi uchun, shartlarni bir xilda izohlab bo‘lmasligi kelib chiqadi. Bu *Turbo Paskal*da quyidagicha hal qilinadi: ixtiyoriy uchragan ELSE qism unga eng yaqin «yuqorida»gi shartli operatorning THEN qismiga mos keladi. Masalan,

```
var
        a,b,c,d: integer;
begin
        a:=1; b:=2; c:=3; d:=4;
        if a>b then
              if c<d then
                    if c<0 then
                          c:=0
                    else
                          a:=b; {a teng 1 ga}
                    if a>b then
```

```
                    if c then
                        if c then
                            c:=0
                        else
                    else
                else
                    a:=b; {a teng 2 ga}.
```

Ixtiyoriy o‘nlik butun sonni 0...15 sohada kiritib, uni o‘n oltilik songa aylantiruvchi va natijani ekranga chiqaruvchi dasturni ko‘ramiz:

```
Program Hex
var
    n: Integer; {kiritiladigan son}
    ch:Char; {natija}
begin
    Write (`n=`);
    Readln (n); {son kiritiladi}
    {Sonning 0...15 sohaga tegishli ekanligini tekshirish}
    if (n>=0) and (n<=15) then
    begin                {ha, sohaga tegishli}
        if n<10 then
            ch:= chr (ord (`o`)+n)
        else
            ch:= chr (ord (`A`)+n-10);
        writeln (`n=`, ch);
    end
                        else   {sohaga tegishli emas}
    writeln (`xato`)
end.
```

O‘n oltilik sanoq tizimida har bir razryadda 16 ta raqam ishlatiladi: 0 ... 9 raqamlari razryadning dastlabki mumkin bo‘lgan 10 qiymatini, A ... F harflar esa qolganlarini bildiradi.

Dasturda 0 ... 9 raqamlari to‘plamining va A ... F harflar va ular kodlarining uzluksizligi va tartibga solinganligi hisobga olingan.

If operatori oddiy holatining umumlashgan ko‘rinishlari 8.1-jadvalda keltirilgan.

8.1-jadval

Tarmoqdagi operatorlar soni		if operatorning umumlashgan ko‘rinishi
then	else	
Bitta	Bitta	if ifoda then operator else operator
Bir nechta	Bitta	if ifoda then begin operator; . . . . . . operator; end else operator
Bitta	Bir nechta	if ifoda then operator else begin . . . . . . operator; end
Bir nechta	Bir nechta	if ifoda then begin operator; . . . . . . operator; end else begin operator; . . . . . . operator; end

8.3. Takrorlash operatorlari

Turbo Paskal tilida takrorlanuvchi bo‘laklarni dasturlashga yordam beruvchi 3 ta har xil operator bor.

8.3.1. Hisobchi FOR sikl operatori

U quyidagi strukturaga ega:

FOR <sikl_param.>: = <boshl._qiym.> TO <oxir_qiym.> DO <operator>.

Bu yerda, FOR, TO, DO — rezerv so‘zlar (uchun, bo‘lguncha, bajar deb tarjima qilinadi);

<sikl_param.> — Integer turidagi o‘zgaruvchi — sikl parametri;

<boshl._qiym.> — o‘sha turdagi ifodaning boshlang‘ich qiymati;

<oxir._qiym.> — o‘sha turdagi ifodaning oxirgi qiymati;

<operator> — *Turbo Paskal*ning ixtiyoriy operatori.

FOR operatorining bajarilishida avval <boshl._qiym.> ifoda hisoblanadi va <sikl._param.>: = <boshl._qiym.> o‘zlashtirish amalga oshiriladi. Shundan keyin siklik ravishda quyidagilar takrorlanadi:

- <sikl_param.> < = <oxir._qiym.>; sharti tekshiriladi, agar shart bajarilsa, FOR operatori o‘z ishini tugatadi;
- <operator> operatori bajariladi;
- <sikl_param.> parametr bir qiymatga oshiriladi.

FOR operatorining qo‘llanilishini quyidagi dasturda ko‘rsatamiz, unga ko‘ra klaviaturadan ixtiyoriy butun *N* soni kiritiladi va barcha 1 dan *N* gacha bo‘lgan butun sonlar yig‘indisi hisoblanadi.

```
Program Summ_of_Integer
var
        i,n,s: integer;
begin
        write ('N=`);
        Readln (n); {N kiritiladi}
        s: = 0;         {yig‘indining boshlang‘ich qiymati}
        for i:=1 to n do  {yig‘indini hisoblash sikli}
                S:=s +i;
        writeln (`yig‘indi=`,s) {natijani chiqarish}
end.
```

Ikki holatni aytib o‘tamiz. Birinchidan, FOR operator ishini boshqaruvchi shart <operator> operatorining bajarilishidan oldin tekshiriladi: agar shart FOR operatorining eng boshidayoq bajarilmasa, bajariluvchi operator biror marta ham bajarilmaydi. Ikkinchi holat, sikl parametrining o‘sish qadami qat’iy bir xil va (+1)ga teng. Operatorning boshqa ko‘rinishi mavjud:

FOR <sikl_param.>: = <boshl._qiym.> DOWNTO <oxir._qiym.> DO <operator>.

Rezerv TO so‘zning —1 DOWNTOga almashtirilishi, sikl

parametrining o‘sish qadami (-1)ga teng, boshqaruvchi shart esa <sikl._param.> = <oxir._qiym.> ko‘rinishini oladi.

Oxirgi dasturni ixtiyoriy yig‘indini (musbat va manfiy) hisoblashga yaroqli qilish uchun o‘zgartiramiz:

```
.....
s:=0;
if n>=0 then
        for i:=1 to n do
                s:=s+i
        else
        for i:=-1 downto n do
                s:=s+i;
.....
```

Takrorlashning qolgan ikki operatori faqat siklning bajarilishi yoki takrorlanishi shartini tekshiradi, lekin sikl hisobchisining o‘zgarishi bilan bog‘liq emas.

8.3.2. WHILE sikl operatori

Uning umumiy ko‘rinishi:

WHILE <shart> DO <operator>

Bu yerda, WHILE, DO—rezerv so‘zlar ([shart bajaril]guncha, bajar)

<shart> — mantiqiy turdagi shart;

<operator>— *Turbo Paskal*ning ixtiyoriy operatori.

Agar <shart> ifoda TRUE qiymatga ega bo‘lsa, u paytda <operator> bajariladi, shundan keyin <shart> ifoda hisoblanadi va uni tekshirish takrorlanadi. Agar <shart> FALSE qiymatga ega bo‘lsa, WHILE operatori o‘z faoliyatini to‘xtatadi.

Misol: FOR operatorini tushuntirishda olingan masalaning dasturini WHILE operatori yordamida yozamiz:

```
Program Summ_of_Integer;
const n=15
var
        i,s: integer;
begin
```

```
        s:=0;        {yig‘indining boshlang‘ich qiymati}
        i:=1;        {hisobchining boshlang‘ich qiymati}
        while i<=n do
            begin
                s:=s+i;      {yig‘indini hisoblash va nav-
                              batdagi
                i:=i+1;      qo‘shiluvchi son qiymatini aniq-
                              lash}
            end;
        writeln (`yiqindi=`,S); {natijani chiqarish}
end.
```

8.3.3. REPEAT ... UNTIL sikl operatori

Uning umumiy ko‘rinishi

REPEAT <sikl_tanasi> UNTIL <shart>

Bu yerda, REPEAT, UNTIL — rezerv so‘zlar (shart bajarilguncha takrorla);

<sikl_tanasi> — *Turbo Paskal* operatorlarining ixtiyoriy ketma-ketligi;

<shart> — mantiqiy turdagi ifoda;

<sikl tanasi> operatori hech bo‘lmaganda bir marta takrorlanadi, shundan keyin <shart> ifoda hisoblanadi; agar uning qiymati FALSE bo‘lsa, <sikl_tanasi> operatorlari takrorlanadi, aks holda REPEAT ... UNTIL operatori o‘z ishini yakunlaydi.

Klaviaturadan *N* (15) butun son kiritilib, 1 dan *N* (15)gacha bo‘lgan butun sonlar yig‘indisini hisoblash masalasi dasturini sikl operatorining 3-ko‘rinishi yordamida yozamiz:

```
Program Summ_of_Integer;
var
        i,n,s: integer;
begin
        write (`n=`);
        Readln (n);
        s:=0; {sikl shartida turgan o‘zgaruvchilarning}
        i:=1; {boshlang‘ich qiymatlari berilgan bo‘lishi kerak}
        repeat
```

```
        s:=s+i;      {sikl tanasining kamida bitta operatorida
                     shartning bajarilmaslik holi bo‘lishi uchun}
        i:=i+1       {sikl shartidagi qiymati o‘zgarishi kerak}
        until i>n
end.
```

E’tibor bering, *repeat ... until* juftliklar *begin ... end* operator qavslariga o‘xshaydi, shuning uchun *until* oldidan nuqtali vergul qo‘yish shart emas.

Sikl operatorlarini ustalik bilan boshqarish uchun *Turbo Paskal*da ikki protsedura kiritilgan:

break — sikldan zudlik bilan chiqishni tashkil qiladi; protsedura faoliyati boshqarishni sikl operatoridan keyin keluvchi operatorga uzatishga qaratilgan;

continue — siklning navbatdagi takrorlanishini vaqtidan oldin to‘xtatadi. Boshqarishni sikl operatorining eng oxiriga uzatadi.

Bu protseduralarning kiritilishi shartsiz o‘tish (GOTO) operatorini *Turbo Paskal*da ishlatish zaruriyatini deyarli yo‘q qiladi.

While, repeat va *for* operatorlari ishini taqqoslashga imkon beruvchi quyidagi jadvalni keltiramiz:

8.2-jadval

WHILE sikl (shart—hozircha haqiqat)	REPEAT operatori (shart—haqiqat bo‘lguncha)
1) sikl boshlanguncha, siklga ixcham kirish uchun, sikl shartlarini boshqaruvchi parametrlarning boshlang‘ich qiymatlari berilgan bo‘lishi kerak	
2) sikl tanasida bir necha takrorlashdan keyin siklni yakunlashga olib keluvchi, shart o‘zgaruvchilari qiymatlarini o‘zgartiruvchi operatorlar bo‘lishi kerak	
3) sikl shart True bo‘lguncha ishlaydi	3) sikl shart False bo‘lguncha ishlaydi
4) sikl shart False bo‘lganda tugaydi	4) sikl shart True bo‘lganda tugaydi
5) agar shartning siklga kirishdagi boshlang‘ich qiymati False bo‘lsa, sikl bir marta ham bajarilmaydi	5) sikl kamida bir marta, albatta, bajariladi
6) agar sikl tanasida bittadan ortiq operator ishlatiladigan bo‘lsa, uni murakkab operator qilib yozish kerak	6) sikl tanasidagi operatorlar soniga bog‘liqmas holda murakkab operatorni ishlatish talab qilinmaydi

FOR hisobchili sikl
1) sikl hisobchisi parametrining boshlang‘ich qiymatini sarlavhagacha berish talab qilinmaydi
2) sikl sarlavhasida turgan o‘zgaruvchilar qiymatini sikl tanasida o‘zgartirish mumkin emas
3) sikl takrorlanishlari soni sikl quyi va yuqori hamda sikl qadami qiymatlari bilan aniqlanadi
4) siklning normal ishlashi goto operatori yoki Break va Continue protseduralari bilan buzilishi mumkin
5) agar sikl qadami hisobchi qiymatini quyi chegaradan yuqori chegaraga qarama-qarshi bo‘lgan yo‘nalishda o‘zgartirib borsa, sikl bir marta ham bajarilmasligi mumkin

8.4. Tanlov operatori

Tanlov operatori dasturni davom ettirishning bir nechta tarmoqlaridan birini tanlashga imkon beradi. Tanlashni amalga oshiruvchi parametr bo‘lib, ixtiyoriy tartib turidagi (REAL va STRING bunga kirmaydi) ifoda — tanlov kaliti xizmat qiladi.

Tanlov operatori strukturasi quyidagicha:

CASE <tanlov_kaliti> OF <tanlov_ro‘yxati> [ELSE <operatorlar>]END.

Bu yerda, CASE, OF, ELSE, END — rezerv so‘zlar (hol, [un]dan, aks holda, tamom deb tarjima qilinadi);

<tanlov_kaliti> — tanlov kaliti;

<tanlov_ro‘yxati> — bir yoki bir nechta quyidagi ko‘rinishdagi konstruksiya;

<tanlov_o‘zgarmasi> : <operator>;

<tanlov_o‘zgarmas> — <tanlov kaliti> ifodasi turi bilan bir xil bo‘lgan o‘zgarmas;

<operatorlar> — *Turbo Paskal*ning ixtiyoriy operatorlari.

Tanlov operatori quyidagicha ishlaydi:

Avval <tanlov_kaliti> ifodasining qiymati hisoblanadi, keyin <tanlov_ro‘yxati> operatorlari ketma-ketligidan, oldida hisoblangan qiymatga teng bo‘lgan o‘zgarmas yozilgan operator izlanadi. Topilgan operator bajariladi va tanlov operatori o‘z ishini tugatadi. Agar tanlov ro‘yxatida hisoblangan qiymatga teng o‘zgarmas

topilmasa, boshqarish ELSE so‘zidan keyingi operatorga uzatiladi. ELSE <operator> qismini tashlab yuborish mumkin. Unda ro‘yxatdan kerakli o‘zgarmas topilmagach, hech narsa sodir bo‘lmaydi, tanlov operatori o‘z ishini tugatadi.

Quyida CASE operatori ishlatilishiga misol qilib dasturlar keltiramiz:

```
Program Tanlash;
var
        x,y:real; ch:char;
begin
        Readln (x,y);
        case ch of
                #72: y:=y—1;
                #80: y:=y+1;
                #75: x:=x—1;
                #77: x:=x+1
        end
end.
```

Tanlov ro‘yxatidagi operatorlardan ixtiyoriysiga bir emas, balki bir nechta, bir-biridan vergul bilan ajratilgan, tanlov o‘zgarmaslari to‘g‘ri kelishi mumkin. Masalan, quyidagi dastur belgilaridan biri kiritilganda (y yoki Y) «Ha» yoki (n yoki N) «Yo‘q» so‘zlarini ekranga chiqaradi:

```
var
        ch:char;
begin
        Readln (ch);
        Case ch of
                'n', 'N':writeln (`Yo‘q`);
                'y', 'Y':writeln (`Ha`)
        end
end.
```

Quyidagi dasturda esa ko‘rsatilgan belgiga qarab kiritilgan ma’lumotlarning raqami, harf (kichik va bosh) yoki belgi ekanligi ma’lum bo‘ladi:

```
var
        Symbol:char;
begin
        Readln (Symbol);
        case Symbol of
                `0` ... `9`: writeln (`Bu raqam`);
                `a` ... `z`: writeln (`Bu kichik harflar`);
                `A` ... `Z`:writeln (`Bu bosh harflar`);
                #10, #13, #26:writeln (`Bu boshqaruvchi belgi`)
        else writeln (`Bu boshqa belgi`)
        end
end.
```

8.5. Nishonalar (metka) va o‘tish operatorlari

Tasnifli dasturlashning hozirgi zamon texnologiyasi «GOTOsiz dasturlash» tamoyiliga asoslangan, chunki o‘tish operatorini ko‘p ishlatish dasturni tushunishni qiyinlashtiradi, uni chalkashtirib, xatolarini tuzatishni murakkablashtiradi.

Shunday bo‘lsa ham, ba’zi bir hollarda o‘tish operatorini ishlatish dasturni soddalashtirishi mumkin.

O‘tish operatorining umumiy ko‘rinishi

GOTO <metka>

Bu yerda, GOTO — rezerv so‘z ([nishonaga] o‘ting); <metka>— nishona.

Nishona (metka) *Turbo Paskal*da dasturning qandaydir bir operatorini nomlashga va shu bilan unga murojaat etishga imkon beradigan ixtiyoriy identifikatordir. Nishona sifatida sonlar ham ishlatilishi mumkin.

Nishona belgilanadigan operator oldiga qo‘yiladi va undan ikki nuqta bilan ajratiladi. Nishona dasturda paydo bo‘lishidan oldin, u *label* rezerv so‘zi bilan bayon etiladi, *label*dan keyin nishonalar ro‘yxati keltiriladi:

```
label
        1, lb1, lb2;
begin
```

```
                .....
                goto lb1;
                .....
          1:    .....
                .....
lb1:lb2:        .....
                .....
                goto lb2;
                .....
```

Nishonalardan foydalanishda quyidagi qoidalarga rioya qilish kerak:

- GOTO operator murojaat qilayotgan nishona bayonlar bo‘limida e’lon qilinishi kerak va u dastur tanasining biror yerida uchrashi shart emas.
- Protsedura (funksiya)da uchraydigan nishonalar uning o‘zida bayon etiladi, shuning uchun tashqaridan protsedura (funksiya) ichidagi nishonaga boshqarishni uzatish mumkin emas.

Nazorat savollari

1. *Turbo Paskal*ning o‘zlashtirish operatori qanday vazifani bajaradi?
2. Murakkab operator nima?
3. Shartli operatorning vazifasi va uning tasnifi.
4. Takrorlash operatorlariga qanday operatorlar kiradi?
5. *For* sikl operatorining mohiyati, uning tasnifi.
6. *While* sikl operatorining mohiyati, uning tasnifi.
7. *Repeat ... Until* sikl operatorining mohiyati, uning tasnifi.
8. Sikl operatorlarini taqqoslash mezonlari.
9. Tanlov operatori qanday operator, u qachon ishlatiladi?
10. Nishonalar (metkalar) nima uchun kerak?
11. O‘tish operatorining vazifasi nimalardan iborat?
12. Nega o‘tish operatoridan, mumkin qadar, kamroq foydalanish tavsiya etiladi?

IX bob. STATIK MA'LUMOTLARNING MURAKKAB TURLARI

9.1. Massivlar

9.1.1. Umumiy tushunchalar. Yuqorida ko'rib o'tilgan ma'lumotlarning oddiy turlari birlik obyektlardan (sonlar, belgilar va h.k.) foydalanishga imkon beradi. *Turbo Paskal*da bir butun deb qaraluvchi bir xil turdagi elementlar to'plamidan iborat obyektlar ham ishlatilishi mumkin. Bular massivlardir, ular bir qancha bir turdagi bir butun deb qaraluvchi obyektlar (sonlar, belgilar, satrlar va h.k.lar)ning rasmiy birlashmasidir.

Massiv ismi (identifikator) va massivning kerakli elementini ko'rsatish uchun zarur bo'lgan *o'lchamlari* (koordinatalar) bilan aniqlanadi. Massiv ismi uning hamma elementlari uchun yagonadir.

Har bir alohida elementga massiv o'lchamiga bog'liq ravishda bir yoki bir nechta indekslar yordamida murojaat qilish mumkin. Indekslar sifatida o'zgarmaslar va tartib turidagi o'zgaruvchilar ishlatiladi. Ixtiyoriy turdagi oddiy o'zgaruvchilar bilan bir qatorda, murakkab turdagi (massivlar, satrlar va h.k.) o'zgaruvchilar ham massiv elementlari bo'lishi mumkin.

Massiv turi bayoni quyidagi ko'rinishida beriladi:

<tur ismi> = ARRAY [<ind.tur.ro'yx.>] OF <tur>

Bu yerda, <tur ismi> — to'g'ri identifikator;
ARRAY, OF — rezerv so'zlar (massiv, ... dan);
<ind.tur.ro'yx.> — bir-biridan vergullar bilan ajratilgan, indeksli turlar ro'yxati; ro'yxatni o'rab olgan kvadrat qavslar — sintaksis talabi;
<tur> — *Turbo Paskal*ning ixtiyoriy turi.

Indeks turlari sifatida *Turbo Paskal*da LONGINT va LONGINT *baza-turli* soha turlaridan boshqa ixtiyoriy tartib turlarni ishlatish mumkin.

To'g'ri tuzilgan dasturda indeks soha-turi tomonidan aniqlangan chegaradan tashqariga chiqmasligi kerak. Masalan,

a:array [] *of Real* bo‘lsa, A [0] ni
c:array [] *of Boolean* bo‘lsa, C [38] ni ishlatish mumkin emas.

*of*dan keyin keluvchi <tur> *Turbo Paskal*ning ixtiyoriy turi, jumladan, boshqa massiv ham bo‘lishi mumkin, masalan,

```
var
    mat:array [0 .. 5] of array [-2 .. 2] of array [char] of Byte;
```

Bu yozuvni ixcham holda quyidagicha yozish mumkin:

```
type
    mat : array [0 .. 5, -2 .. 2, char] of Byte.
```

Umuman, strukturali turlarning, demak, massivning ichma-ichlik chuqurligi ixtiyoriy, shuning uchun indeks turining ro‘yxatdagi elementlar soniga ham chegara yo‘q, lekin ixtiyoriy massivning ichki uzunligi yig‘indisi 65520 baytdan oshmasligi kerak. Shaxsiy kompyuter xotirasida massiv elementlari biri ketidan ikkinchisi shunday keladiki, bunda kichik adresli xonadan kattasiga o‘tganda massivning eng chekka o‘ng indeksi tez almashinadi. Agar, masalan,

```
var
        a:array [1 .. 2, 1 .. 2] of Byte;
begin
        a[1,1]:=1;
        a[2,1]:=2;
        a[1,2]:=3;
        a[2,2]:=4;
end
```

bo‘lsa, xotirada 1,3,2,4 qiymatli baytlar ketma-ket joylashadi. Bu holat xotirani nusxalashning *MOVE* standart protsedurasini ishlatishda muhim bo‘lishi mumkin.

*Turbo Paskal*da bitta o‘zlashtirish operatori bilan bir massivning hamma elementlarini shu turdagi boshqa massivga uzatish mumkin, masalan,

```
var
        a,b:array [1 .. 5] of Single;
begin
        .....
        a:=b;
```

```
.....
end.
```

Shu o‘zlashtirishdan keyin *A* massivning hamma beshta elementi *B* massiv elementlari qiymatlarini qabul qiladi. Lekin massivlar ustida munosabat amallari aniqlanmagan. Masalan,

if a=b then ...

deb yozish mumkin emas.

Ikki massivni elementlari bo‘yicha taqqoslash mumkin, masalan,

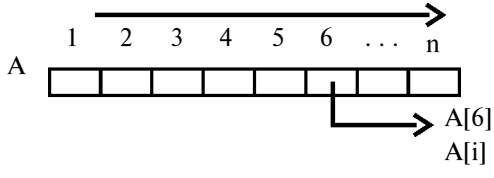
```
var
        a,b:array [1 .. 5] of Single;
        eg:Boolean;
        i:Byte;
begin
        .....
        eg:=True;
        for i:=1 to 5 do
                if a [i] <> b [i] then
                eg:=False;
        if eg then
        .....
end.
```

Masalalar yechishda bir, ikki va uch o‘lchovli massivlardan foydalaniladi. Katta o‘lchamdagi massivlar amalda oz uchraydi.

Sxematik ravishda bir, ikki va uch o‘lchovli massivlarni quyidagicha ifodalash mumkin:

Bir o‘lchovli massiv (vektor)

Indeks o‘zgarishining yo‘nalishi



Bayoni

```
const
    n=100;
var
    A: array [1 .. n] of Real;
```

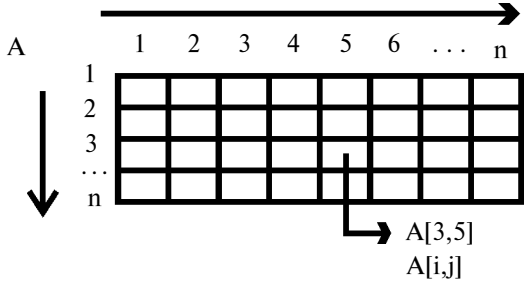
yoki

```
const
    n=100;
type
    Vector = array [1 .. n] of Real;
var
    A : Vector.
```

Ikki o'lchovli massiv (matritsa)

Ikkinchi indeks o'zgarishining yo'nalishi

Birinchi indeks o'zgarishining yo'nalishi



Bayoni

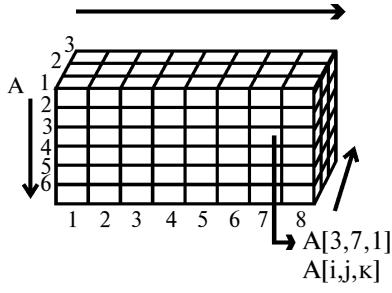
```
const
    m=30;  n=50;
var
    A: array [1 .. m, 1 .. n] of Real;
```

yoki

```
const
    m=30;  n=50;
type
    Matr = array [1 .. m, 1 .. n] of Real;
var
    A : Matr;
```

Uch o‘lchovli massiv

Ikkinchi indeks o‘zgarishining yo‘nalishi



Birinchi indeks o‘zgarishining yo‘nalishi

Uchinchi indeks o‘zgarishining yo‘nalishi

Bayoni

```
const
    m=30; n=50; p=20;
var
    A: array [1 .. m, 1 .. n, 1 .. p] of Real;
```

yoki

```
const
    m=30; n=50; p=20;
type
    T_Array = array [1 .. m, 1 .. n, 1 .. p] of Real;
var
    A : T_Array.
```

Massiv imkoniyatlarini namoyish etishning klassik misollariga saralash va izlash masalalari kiradi. Ular bilan tanishamiz.

9.1.2. Massivlarni saralash. Saralash masalasini yechishda, odatda, qo‘shimcha xotiradan minimal foydalanish talabi oldinga suriladi, bundan qo‘shimcha massivlardan foydalanish mumkin emasligi kelib chiqadi.

Saralashning har xil usullari algoritmlarining ishi tezligini baholash uchun, qoida bo‘yicha, ikki ko‘rsatkichdan foydalaniladi:

• o‘zlashtirishlar soni;
• taqqoslashlar soni.

Saralashning hamma usullarini ikki katta guruhga ajratish mumkin:

• saralashning to‘g‘ri usullari;
• saralashning yaxshilangan usullari.

Saralashning to‘g‘ri usullari, o‘z navbatida, uch guruhga bo‘linadi:

• kiritish yo‘li bilan saralash;
• tanlash yo‘li bilan saralash;
• almashtirish yo‘li bilan saralash («ko‘pik» usuli).

Saralashning yaxshilangan usullari ham to‘g‘ri usullar tayangan tamoyilga asoslanadi, lekin saralashni tezlashtiradigan ba’zi bir g‘oyalardan foydalanadi. Amalda to‘g‘ri usullar, past tezlikka ega bo‘lganligi uchun, kam ishlatiladi. Lekin to‘g‘ri usullar ularga asoslangan yaxshilangan usullarning mohiyatini yaxshi ko‘rsatadi. Bundan tashqari, ba’zi bir hollarda uzun bo‘lmagan massivda va (yoki) massiv elementlarining o‘ziga xos joylashgan vaqtida to‘g‘ri usullarning ba’zilari yaxshilangan usullardan ham o‘tishi mumkin.

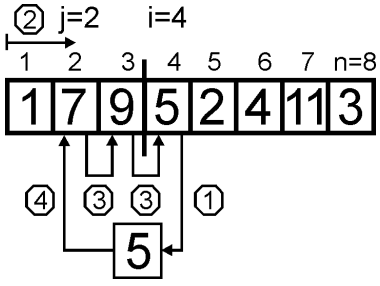
9.1.2.1. Kiritish yo‘li bilan saralash. **Usul tamoyili.** Massiv saralangan va saralanmagan ikki qismga bo‘linadi: saralanmagan qismdan elementlar navbat bilan tanlanadi va saralangan qismga, elementlar tartibini buzmasdan, kiritiladi. Algoritm ishi boshida massivning saralangan qismi sifatida faqat bitta birinchi element, saralanmagan qism sifatida esa qolgan elementlar olinadi.

Shunday qilib, algoritm har biri to‘rtta ishdan iborat $n-1$ kiritishdan (massiv n o‘lchamli) tashkil topadi:

• navbatdagi i—saralanmagan elementni olish va uni qo‘shimcha o‘zgaruvchida saqlash;
• massivning saralangan qismida olingan elementning mavjudligi elementlar tartibini buzmaydigan, j—o‘rinni izlash;
• topilgan o‘rinni bo‘shatish uchun massiv elementlarini $i-1$ dan $j-1$ gacha o‘ngga siljitish;
• olingan elementni topilgan j—o‘ringa joylashtirish.

Bu usulni amalga oshirish uchun bir-biridan kiritish o‘rnini izlash usullari bilan farq qiluvchi, bir nechta algoritmni taklif qilish mumkin. Mumkin bo‘lgan algoritmlardan birini amalga oshirish chizmasini ko‘ramiz.

Bitta kiritish ishini chizma sifatida quyidagicha bayon etish mumkin:



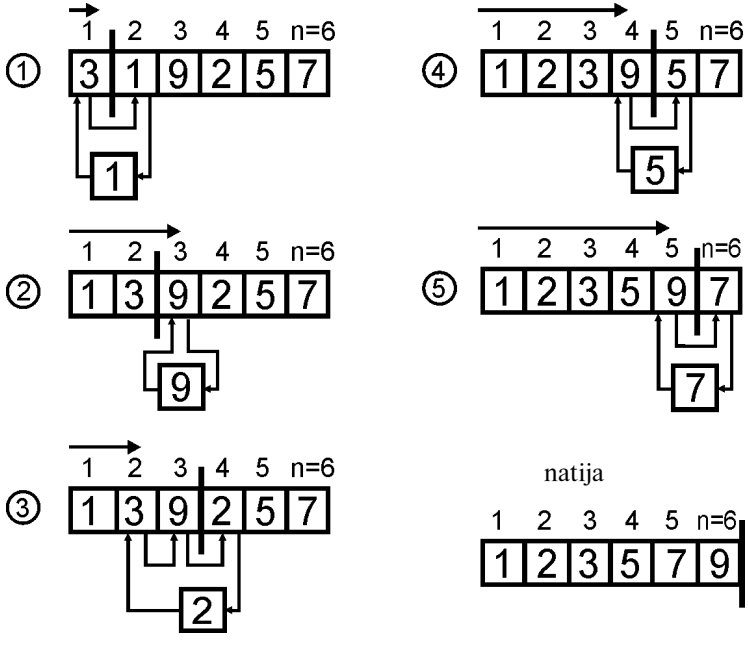
① — Saqlash

② — Qo‘yish joyini izlash

③ — Surish

④ — Qo‘yish

Kiritish yo‘li bilan saralash chizmasi. Chapda aylana ichida o‘tish raqami ko‘rsatilgan:



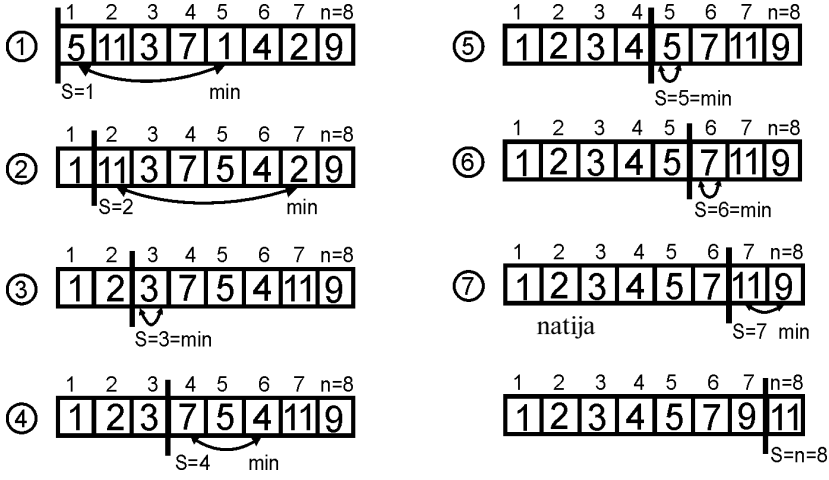
Ko‘rib chiqilgan algoritmni amalga oshirish dasturini keltiramiz:

```
program InsertionSort;
uses Crt;
const
        n = 20; {massiv uzunligi}
type
        TVector = array [1..n] of Real;
var
        Vector : TVector;
        B      : Real;
        i, j, k : Integer;
begin
        ClrScr;
        Writeln (‘Massiv elementlarini kiritish:’);
        for i : = 1 to n do Read (Vector [i]) ; Readln;

{-----------------------------------------------}
for i := 2 to n do
begin
        B := Vector[i];  {Saralanmagan elementni olish}
              {Qo‘yish o‘rnini topish sikli}
        j := 1;
        while (B > Vector[j]) do
        j := j + 1;  {j indeks sikl tugagandan keyin}
                     {qo‘yish o‘rnini belgilaydi}
{Qo‘yish o‘rnini bo‘shatish uchun elementlarni}
        {siljitish sikli}
        for k := i-1 downto j do
        Vector[k+1] := Vector[k];
{Topilgan o‘ringa olingan elementni kiritish}
        Vector[j] := B;
end;

{-------------------------------------------------}
        Writeln ('Saralangan massiv:');
        for i := 1 to n do Write (Vector[i]:8:2);
        Writeln;
end.
```

9.1.2.2. Tanlash yo‘li bilan saralash. **Usul tamoyili.** Massivda 1-elementdan *n*-elementgacha (oxirgi) bo‘lgan oraliqda minimal qiymatli elementni topamiz (tanlaymiz) va uni birinchi element o‘rni bilan almashtiramiz. Ikkinchi qadamda 2-dan *n*-gacha oraliqda minimal qiymatli elementni topamiz va uni ikkinchi element o‘rni bilan almashtiramiz.

Shunday tarzda *n-1* elementgacha almashtirishlarni bajaramiz.

Tanlash algoritmi chizmasini ko‘ramiz:



Tanlash usulini amalga oshiruvchi dastur matnini keltiramiz:

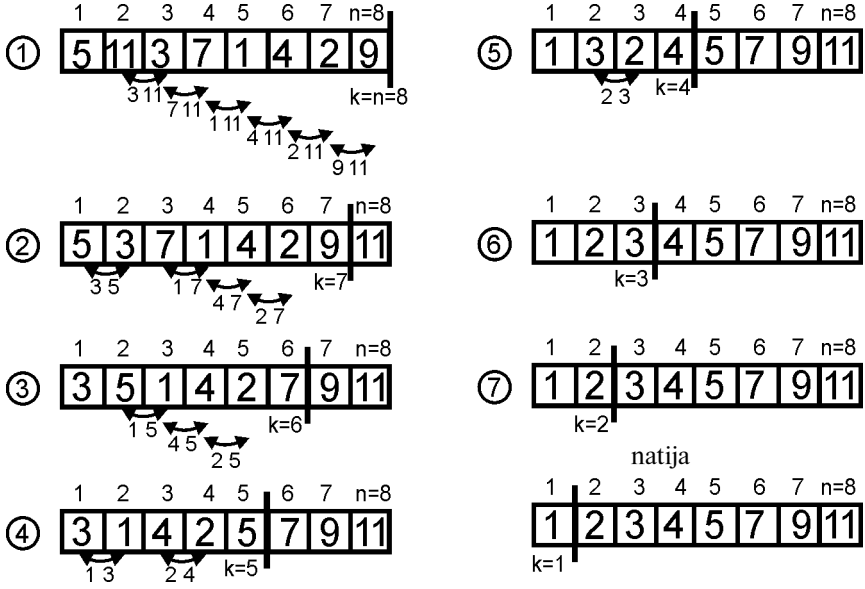
```
program SelectionSort;
uses Crt;
const
        n = 20; {massiv uzunligi}
type
        TVector = array [1..n] of Real;
var
        Vector : TVector;
        Min : Real;
        Imin, S : Integer;
        i       : Integer;
begin
```

```
            ClrScr;
            Writeln ('Massiv elementlarini kiritish:');
            for i := 1 to n do Read (Vector[i]); Readln;
{-------------------------------------------------------}
            for S := 1 to n-1 do
            begin
{S-elementdan n-gacha bo'lgan sohada minimal elementni izlash}
                  Min := Vector[S];
                  Imin : = S;
                  for i := S+1 to n do
                        if Vector[i] < Min then
                        begin
                              Min := Vector [i] ;
                                    Imin : = i
                        end;
    {Minimal va S-elementlarni o'rinlari bilan almashtirish}
                  Vector[Imin] :=Vector[S];
                  Vector[S] := Min;
    end;
{-------------------------------------------------------}
                        Writeln ('Saralangan massiv:');
                        for i := 1 to n do Write (Vector[i]:8:2);
                        Writeln;
    end.
```

9.1.2.3. Almashtirish yo'li («ko'pik» usuli) bilan saralash. **Usul tamoyili.** Chapdan o'ngga tomon navbat bilan ikki qo'shni element, tartibga solishning berilgan shartiga mos kelmasa, o'rinlari bilan almashtiriladi. Shundan keyin navbatdagi qo'shni elementlar olinadi, xuddi shunday tarzda, massiv oxirigacha, ish bajariladi.

Bir marta shunday o'tishdan keyin massivning oxirgi *n*-xonasida maksimal element joylashgan bo'ladi (birinchi «ko'pik» yuzaga chiqadi). Oxirgi element o'zining oxirgi xonasida turgani uchun, almashtirishning ikkinchi o'tishi endi *n*-1 elementgacha bo'ladi va h.k. Hammasi bo'lib, *n*-1 ta o'tish talab qilinadi.

Almashtirish usulining o'sib borish tartibida saralash algoritmi chizmasini ko'ramiz:



Dastur matnini keltiramiz.

```
program BubbleSort;
uses Crt;
const
        n = 20; {massiv uzunligi}
type
        TVector = array [1..n] of Real;
var
        Vector : TVector;
        B      : Real;
        i, k   : Integer;
begin
        ClrScr;
        Writeln ('Massiv elementlarini kiritish:');
        for i := 1 to n do Read (Vector[i]); Readln;
{----------------------------------------}
        for k := n downto 2 do
        {Navbatdagi maksimal elementning k-xonaga «qalqib
        chiqishi»}
                for i := 1 to k-1 do
                        if Vector[i] > Vector[i+1] then
                        begin
```

```
                    B := Vector[i] ;
                    Vector[i] := Vector[i+1];
                    Vector[i+1] := B
                end;
{-------------------------------------------------}
            Writeln ('Saralangan massiv:');
            for i := 1 to n do Write (Vector[i]:8:2);
            Writeln
end.
```

9.1.3. Saralash usullarini taqqoslash. Saralash usullari algoritmlarini nazariy va amaliy tadqiq qilish shuni ko‘rsatdiki, taqqoslash, shuningdek, o‘zlashtirish soniga ko‘ra ular *n* massiv uzunligiga bog‘liq. O‘zlashtirish tartibi $n \log(n)$ soniga teng bo‘lgan tanlash usuli bundan mustasno. Tanlash algoritmining bu xossasini bitta taqqoslashda katta sondagi o‘zlashtirishlar bajariladigan murakkab tasnifli ma’lumotlarni saralash masalalarida qo‘llash foydali. Bunday masalalarda tanlash usuli saralashning eng tez yaxshilangan usullari bilan bahslasha oladi.

Ko‘rib o‘tilgan to‘g‘ri usullar o‘zaro taqqoslansa, kiritish va tanlash usullari o‘rtacha taqriban teng kuchli va almashtirish («ko‘piklar») usuliga qaraganda bir necha marta (massiv uzunligiga bog‘liq ravishda) yaxshi.

9.1.4. Ikkilamchi izlash (binarli izlash, ikkiga bo‘lish bilan izlash). Ikkilamchi izlash usuli algoritmini berilgan elementni faqat tartibga solingan massivlarda qo‘llab bo‘ladi. Uni o‘sib borish tarzida tartibga solingan massiv misolida ko‘ramiz (*vector* [i] <=*Vector [i+1]).*

Ikkilamchi izlash usuli tamoyili

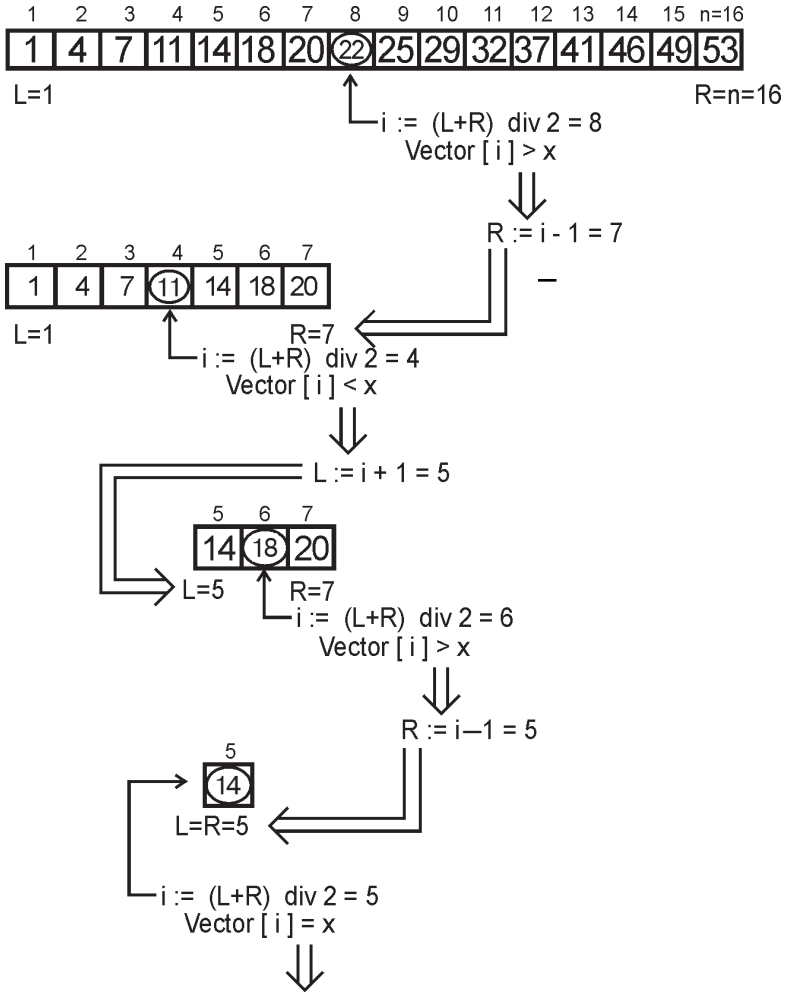
Boshlang‘ich massiv ikkiga bo‘linadi va taqqoslash uchun o‘rta element olinadi. Agar u izlanayotgan element bilan mos kelsa, izlash tugaydi. Agar o‘rtacha element izlanayotganidan kichik bo‘lsa, undan chapda yotgan elementlarning hammasi ham izlanayotganidan kichik bo‘ladi. Demak, ularni, massivning faqat o‘ng qismini qoldirib, keyingi izlash sohasidan o‘chirib tashlash mumkin. Xuddi

shunday, agar o‘rta element izlanayotganidan katta bo‘lsa, massivning chap tomoni saqlanib, o‘ng tomoni olib tashlanadi.

Ikkinchi bosqichda massivning qolgan yarmi bilan xuddi shunday ishlar bajariladi. Ikkinchi bosqich natijasida massivning $^1/_4$ qismi qoladi. Bu ish, element topilmaguncha yoki izlash sohasi nolga teng bo‘lmaguncha davom ettiriladi. Oxirgi holatda izlanayotgan element topilmaydi.

Ikkilamchi izlash algoritmi chizmasini ko‘ramiz:

Izlanayotgan element X 14



Ikkilamchi izlash usulini amalga oshiruvchi dasturning mumkin bo‘lgan variantlaridan birini keltiramiz:

```
program BinSearch
uses Crt;
const
        n=20; {massiv uzunligi}
type
        TVector = array [1 .. n] of Real;
var
        Vector : TVector;       {Boshlang‘ich massiv}
        X      : Real;                  {Izlanayotgan element}
        L, R : Integer;         {Izlash sohasining joriy chega-
                                 ralari}
        i      : Integer;
begin
        ClrScr;
        Writeln (Massiv elementlarini kiriting:);
        for i: = 1 to n do Read (Vector [i]); Readln;
        Write (Izlanayotgan elementni kiriting:);
        Readln (X);
{-------------------------------------------------}
        L : = 1; R : = n;
        while (L <= R) do {Chegaralar kesishmaguncha}
        begin
              i: = (L + R) div 2; {O‘rta element indeksi}
              if Vector[i] = X then
                    Break {Izlash siklidan chiqish,}
                    {chunki element topildi}
                                else
                    if Vector [i] < X then L : = i + 1
                                      else R : = i — 1;
        end; {while}
        if Vector [i] = X then
              Writeln ('Izlanayotgan element', i : 3, 'sohada
              topildi')
                    else
              Writeln ('Izlanayotgan element topilmadi');
{-----------------------------------------------------------}
end.
```

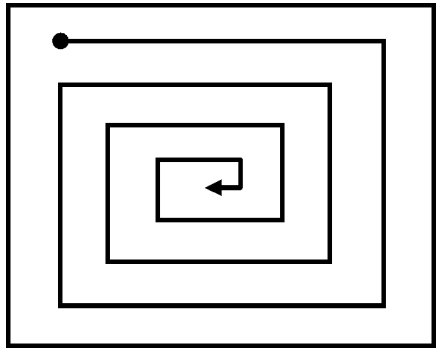
9.1.5. Ikki o‘lchovli massivlar (matritsalar) bilan ishlashga misollar. 1-masala. Elementlari butun sonlar bo‘lgan ikki o‘lchamli $m \times n$ massiv berilgan.

Simmetriyaning vertikal o‘qiga nisbatan matritsa elementlarining «ko‘zguli akslantirilishi»ni bajaring (birinchi ustun elementlarini oxirgisi bilan, ikkinchi ustun elementlarni bitta oldingisi bilan va h.k. o‘rinlari bilan almashtirish):

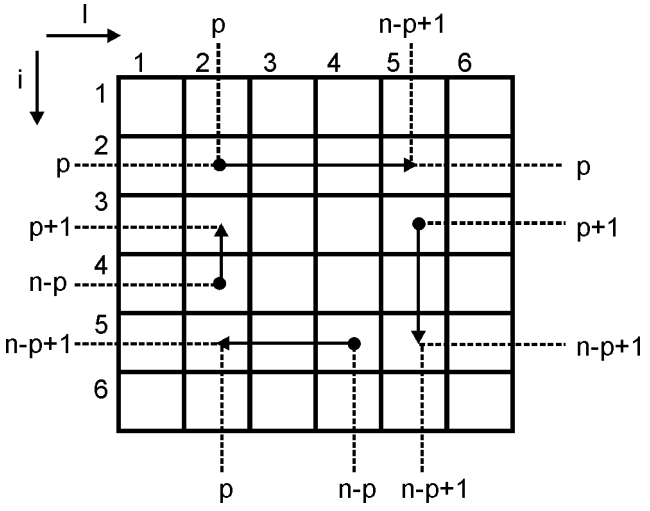
```
program VertMirrow;
uses Crt;
const m = 10; {satrlar soni}
      n = 15; {ustunlar soni}
type
      TMatr = array [l..m,l..n] of Integer;
var
      Matr : TMatr; {Boshlang‘ich matritsa}
      Finp : Text; {Boshlang‘ich ma’lumotlar fayli}
      B : Integer;
      i, j : Integer;
procedure PrintMatr;
var i, j : Integer;
begin
      for i := 1 to m do
      begin
            for j := 1 to n do Write ( Matr [i,j]:5 );
            Writeln;
      end;
      Writeln;
end;
begin ClrScr;
      {Matritsa boshlang‘ich qiymatlarini o‘qish}
      Assign (Fxnp, FINP.DAT); Reset (Finp);
      for i := 1 to m do
      begin
            for j := 1 to n do Read (Finp, Matr[i,j]);
            Readln(Finp);
```

```
        end;
{-------------------------------------------------}
        Writeln ('Boshlang‘ich matritsa:');
        PrintMatr;
{-------------------------------------------------}
{Matritsaning simmetriya vertikal o‘qiga nisbatan}
{"ko‘zguli akslanishi"}
        for j : = 1 to n div 2 do {1-dan o‘rtanchasigacha ustun-
                                   larni}
                                   {olamiz}
              for i := 1 to m do {simmetrik ustunlarning o‘rin-
                                   lari}
              begin {bilan almashtiramiz}
                    B := Matr[i,j];
                    Matr[i,j] := Matr[i,n-B+l];
                    Matr[i,n-j+l] := B
              end;
{-------------------------------------------------}
        Writeln ('O‘zgartirilgan matritsa:');
        PrintMatr;
end.
```

2-masala. Elementlari butun sonlar bo‘lgan *n*-tartibli kvadrat matritsa berilgan. Quyidagi rasmda ko‘rsatilgani kabi matritsaning «spiral» bo‘yicha aylanishini bajarib, elementlar qiymatlarini chop eting:



«Spiral» bo‘yicha aylanishni bajarish uchun indekslar o‘zgarishi qonuniyatini ko‘rsatuvchi chizmani ko‘ramiz:



Keltirilgan chizmaga mos dastur quyidagi ko‘rinishga ega:

```
program SpiralWrite;
uses Crt;
const   n = 10; {Kvadrat matritsa tartibi}
type
        TMatr = array [l..n, l..n] of Integer;
var
        Matr : TMatr; {Boshlang‘ich matritsa}
        Finp : Text; {Boshlang‘ich ma’lumotlar fayli}
        i, j, p : Integer;
procedure PrintMatr;
var i,j : Integer;
begin
        for i := 1 to n do
        begin
              for j := 1 to n do Write (Matr[i,j]:5);
               Writeln;
        end;
        Writeln;
end;
begin ClrScr;
{-----------------------------------------------}
```

```
    {Matritsa boshlang‘ich qiymatlarini o‘qish}
    Assign (Finp,'FINP.DAT');
    Reset (Finp);
    for i := 1 to n do
    begin
            for j := 1 to n do Read (Finp, Matr[i,j]);
             Readln(Finp);
    end;
{-------------------------------------------------------------}
    Writeln ('Boshlang‘ich matritsa:');
    PrintMatr;
{-------------------------------------------------------------}
{Matritsa elementlarini "spiral" bo‘yicha yozib olish}
    for p := 1 to (n+1) div 2 do
    begin
{P—«o‘rama» yuqori satr elementlarini yozib olish}
    for j := p to n-p+1 do Write (Matr[p,j]:4);
{P—«o‘rama» o‘ng ustun elementlarini yozib olish}
    for i := p+1 to n-p+1 do Write (Matr[i,n-p+1]:4);
{P—«o‘rama» quyi satr elementlarini yozib olish }
    for j := n-p downto p do Write (Matr[n-p+1,j]:4);
{P—«o‘rama» o‘ng ustun elementlarini yozib olish}
    for i := n-p downto p+1 do Write (Matr[i,p]:4);
    end;
{-------------------------------------------------------------}
end.
```

9.1.6. Satrlar. Satrlar belgilardan iborat bir o‘lchovli massivlarning alohida ko‘rinishini ifodalaydi. U ko‘p jihatlari bilan belgilarning bir o‘lchovli *ARRAY [0 .. N] of char* massiviga o‘xshab ketadi, lekin uning massivdan farqli tomoni shundaki, satr-o‘zgaruvchida belgilar soni 0 dan *N* gacha (*N*—satrdagi belgilarning maksimal soni) o‘zgaradi. *N* ning qiymati *String* [*N*] turni e’lon qilish bilan aniqlanadi va u 225 dan katta bo‘lmagan ixtiyoriy tartib turidagi o‘zgarmas bo‘lishi mumkin. *Turbo Paskal N* ni ko‘rsatmaslikka ruxsat beradi, bu holda satr uzunligi 255 ga teng, deb olinadi.

Satr *Turbo Paskal*da belgilar zanjiri deb tushuntiriladi. Satrning ixtiyoriy belgisiga bir o‘lchovli ARRAY [0 .. *N*] *of* CHAR massiv elementiga murojaat qilganday murojaat qilish mumkin, masalan:

```
var st:String;
begin
        .....
        if st[s] = 'A ' then ...
end.
```

Satrdagi eng birinchi bayt indeksi 0 va u satrning joriy uzunligiga ega. Satrning 1-aniqlik belgisi ikkinchi baytni egallaydi va 1 indeksga ega bo‘ladi. Satr uzunligi ustida zaruriy amallarni bajarish va shunday yo‘l bilan uzunlikni o‘zgartirish mumkin. Masalan, satrdagi hamma ma’lum bo‘shliqlarni quyidagicha yo‘qotish mumkin:

```
var
        st:String;
        i:Byte;
begin
        .....
        i:=ord(st[0]);      {i-}
        while (i<>0) and (st[i]=' ') do
        begin
                dec (i);
                st[0]:= chr(i)
        end;
        .....
end.
```

Satrning joriy uzunligini, ya’ni *ORD(st[0])* qiymatni *LENGTH (st)* funksiya yordamida ham hosil qilish mumkin, masalan,

```
while (Length (st) <>0) and (St[Length (st)] = ' ') do
        st[0]: = chr (Length (st)—1).
```

Satrlarga «+» — ulash amalini qo‘llash mumkin, masalan,

```
st:='a' +'b';
st:=st+'c';             {st     «abc» ga ega}
```

Agar ulangan satr uzunligi maksimal mumkin bo‘lgan uzunlikdan (*N* dan) oshsa, «ortiqcha» belgilar tashlab yuboriladi. Masalan, quyidagi dastur 1 belgini chop etadi:

```
var
        st:String [1];
begin
        st:='123';
        writeLn (st)
end.
```

Satrlar va belgilar ustidagi boshqa amallar quyida keltiriladigan standart protsedura va funksiyalar yordamida amalga oshiriladi.

CONCAT (S1 [,S2, ..., SN]) — S1, S2, ..., SN satr-parametrlar ulanishini ifodalovchi *string* turidagi funksiya satrni qaytaradi.

COPY (ST, INDEX, COUNT) — INDEX raqamdan boshlab, ST COUNT satrdan belgilarni nusxalovchi *string* turidagi funksiya.

DELETE (ST, INDEX, COUNT) — INDEX raqamli belgidan boshlab, COUNT belgilarni ST satrdan yo‘qotuvchi protsedura.

INSERT (SUBST, ST, INDEX) — INDEX raqamli belgidan boshlab SUBST satrni *ST* satr orasiga kirituvchi protsedura.

LENGTH (ST) — *Integer* turidagi funksiya, ST satr uzunligini qaytaradi.

POS (SUBST, ST) — *Integer* turidagi funksiya, ST satrdan SUBST ichki satrning birinchi kirishini izlaydi va o‘zi boshlangan xona raqamini qaytaradi, agar ichki satr topilmasa, nolni qaytaradi.

STR (x[:WIDTH[:DECIMALS]], ST) — chiqarish oldidan WRITELN protsedurasi bajargani kabi, ixtiyoriy haqiqiy yoki butun turdagi X sonini ST ramzlar satriga o‘zgartiruvchi protsedura; WIDTH va DECIMALS parametrlar, agar ular mavjud bo‘lsa, o‘zgarish formatini beradi: WIDTH haqiqiy yoki butun sonni ramzli ifodalash uchun ajratilgan tegishli maydonning umumiy enini, DECIMALS bo‘lsa, kasr qismidagi ramzlar sonini (bu parametr X faqat haqiqiy son bo‘lgandagina ma’noga ega) aniqlaydi.

VAL (ST, X, CODE) — ST ramzlar satrini butun yoki haqiqiy X o‘zgaruvchi turi bilan aniqlanadigan o‘zgaruvchining ichki ifodalanishiga o‘zgartiruvchi protsedura; CODE parametri, agar o‘zgartirish muvaffaqiyatli o‘tgan bo‘lsa, nolga ega bo‘ladi va shunda X ga o‘zgartirish natijasi joylashtiriladi, aks holda u xato ramz topilgan o‘rin raqamini ST satrda saqlaydi va bu holda X tarkibi

o‘zgarmaydi; ST satrda yetakchi bo‘shliqlar bo‘lishi mumkin, lekin ma’lum bo‘shliqlarga yo‘l qo‘yilmaydi; masalan, *val* (′123 ′, *k,i*) muvaffaqiyatli o‘tadi: K 123 qiymatni qabul qiladi va *i* ga 0 joylashtiradi, *val* (′ 123′ ,*k,i*) esa xato, deb topiladi: K ning qiymati o‘zgarmaydi, *i* esa 4 ga ega bo‘ladi.

UPSASE (CH) — CHAR turidagi funksiya, agar CH ramziy ifoda kichik lotin harfini ifodalasa, funksiya mos bosh harfni qaytaradi; agar CH qiymati ixtiyoriy boshqa ramziy bo‘lsa (shu jumladan, kirillcha kichik harf), funksiya uni o‘zgarishsiz qaytaradi.

Misollar:

```
var
        x:Real;
        y:Integer;
        st,st1:String
begin
        st:=concat ('12', '345');          {st satr 12345 ga ega}
        st1:=copy (st,3, Length (st)—2);   {st1 satr 345 ga ega }
        insert ('—', st1,2);               {st1 satr 3-45 ga ega }
        delete (st, pos ('2', st),3);      {st satr 15 ga ega}
        str (pi:6:2, st);                  {st satr 3,14 ga ega }
        val ('3,1415', x,y);               {y 2 ga ega, y o‘zga-
                                            rishsiz qoldi}
end.
```

Munosabat amallari (=, <>, >, <, >=, <=) ikkita satr ustidan belgilari bo‘yicha, chapdan o‘ngga tomon, belgilarning ichki kodirovkasini hisobga olgan holda bajariladi, agar bitta satr uzunligi bo‘yicha boshqasidan kichik bo‘lsa, qisqa satrning yetishmagan belgilari CHR(0) qiymat bilan almashtiriladi.

Quyidagi munosabat amallari TRUE qiymatni beradi:

```
 ″ ″ <′ . ′
′A′ > ′1′
′Turbo′ <′Turbo Pascal′
′Paskal′ > ″Turbo Pascal′
```

9.2. Yozuvlar

Yozuv — bu yozuv maydonlari deb ataluvchi belgilangan sondagi a'zolardan iborat ma'lumotlar tasnifidir. Massivdan farq qilib, yozuv a'zolari (maydonlari) har xil turda bo'lishi mumkin. Yozuvning u yoki bunisiga murojaat qila olish mumkin bo'lishi uchun maydonlar nomlanadi.

Yozuvlar turini e'lon qilish tasnifi quyidagicha:

<tur nomi> = RECORD <maydonlar ro'yxati> END,

bu yerda, <tur nomi> — to'g'ri identifikator;

RECORD, END — rezerv so'zlar (yozuv, oxiri);

<maydonlar ro'yx.> — oralarida nuqtali vergul qo'yilgan yozuv bo'limlari ketma-ketligini ifodalovchi maydonlar ro'yxati.

Yozuvning har bir bo'limi bir-biridan vergul bilan ajratilgan bir yoki bir nechta maydon identifikatoridan iborat. Identifikator (identifikatorlar)dan keyin ikki nuqta qo'yiladi va maydon (maydonlar) turi bayoni yoziladi, masalan:

```
type
        BirthDay=record
                day,month:byte;
                year:word;
        end;
var
        a,b:Birthday
        .....
```

Bu misolda BIRTHDAY (tug'ilgan kun) turi DAY, MONTH va YEAR (kun, oy va yil) maydonlarga ega yozuv; *A* va *B* o'zgaruvchilar BIRTHDAY turidagi yozuvlarga ega.

Massivdagi kabi, yozuv turi o'zgaruvchilarining qiymatlarini o'sha turdagi boshqa o'zgaruvchilarga o'zlashtirishi mumkin, masalan,

```
a: = b;
```

Agar murakkab ismdan foydalanilsa, ya'ni o'zgaruvchi ismi, keyin nuqta va maydon ismi ko'rsatilsa, yozuvning har bir a'zosi bilan ishlash mumkin:

```
a.day:=27
b.year:=1939
```

Ichma-ich maydonlar uchun aniqlik kiritish davom ettiriladi:

```
type
        Birthday=record
                day, month:Byte;
                year: word;
        end;
var
        c:=record
                name:string;
                bd: birthday
        end;
begin
        ..................
        if c.bd.year=1939 then .....
end.
```

Yozuv maydonlari bilan ishlashni yengillashtirish uchun qo‘shilish operatori — WITH ishlatiladi.

WITH <o‘zgaruvchi> DO <operator>,

bu yerda,

WITH, DO — kalit so‘zlar (dan, bajarish)

<o‘zgaruvchi> — yozuv turidagi o‘zgaruvchi ismi, undan keyin ichma-ich kelgan maydonlar ro‘yxati beriladi;

<operator> — *Turbo Paskal*ning ixtiyoriy operatori.

Misol:

```
with c.bd do month:=9;
```

Bunga quyidagilar teng kuchli:

```
with c do with bd do month:=9;
```

yoki

```
with c, bd do month:=9;
```

yoki

```
c. bd month:=9;
```

Turbo Paskal variant yozuvlar deb ataluvchi yozuvlar bilan ishlashga ruxsat etadi, masalan,

```
type
        Forma=record
        Name:String;
        Case Byte of
                0: (BirhtPlace : String[40]);
                1: (Country : String[20]);
                EntryPort : String[20]);
                EntryDate : 1 .. 31;
                ExitDate : 1 .. 31;
end.
```

Bu misolda FORMA turi NAME — bitta belgilangan maydon va CASE...OF ifoda bilan berilgan variant qismidan iborat yozuvni aniqlabdi. Variant qism bir nechta variantlar (misolda, ikkitasidan: 0 va 1)dan iborat bo‘ladi. Har bir variant tanlov o‘zgarmasi bilan aniqlanadi. Tanlov o‘zgarmasidan keyin ikki nuqta, aylana qavslar ichida yozilgan maydonlar ro‘yxati keladi. Har qanday yozuvda faqat bitta variantli qism bo‘ladi va agar u bo‘lsa, variantli qism hamma belgilangan maydonlardan keyin joylashishi kerak.

Variant qismining ajoyib xususiyati shunday holdan iboratki, unda berilgan variantlar bir-biri ustiga «qo‘yiladi», ya’ni ularning har biriga xotiraning bitta va o‘sha adresi ajratiladi. Bu turlarni o‘zgartirishning yangi qo‘shimcha imkoniyatlarini ochadi, masalan,

```
var
        mem 4 : record
        case Byte of
                0: (by:array [0 .. 3] of Byte);
                1: (wo:array [0 ..1] of word);
                2: (lo:longint);
end.
```

Bu misolda MEM4 yozuv uchta variantdan iborat, ularning har biri xotiradan 4 baytga teng bitta va o‘sha sohani egallaydi. Dasturda, yozuvning qaysi maydoniga murojaat qilishiga qarab, shu maydon 4 baytdan (BY maydon), ikkita WORD turidagi (WO maydon) yoki, nihoyat, LONGINT turidagi bitta butun sondan (LO maydon) iborat massiv, deb qaralishi mumkin. Masalan, bu yozuvga avval uzun butun sifatida qiymat berish, keyin esa natijani baytlar yoki so‘zlar bo‘yicha tahlil qilish mumkin:

```
var
        x:Word;
        xb:Byte;
        xl:Longint;
begin
        ..............
        with m do
                begin
                        lo:=trunc (2*pi*x);
                        if wo [1] = 0 then
                                if by [1] = 0 then
                                        xb:=x[0]
                                else
                                        x:=wo[0]
                                else
                                        xl:=lo
                        end;
                        ............
end.
```

Variant qismni ochuvchi CASE ... OF, ifoda tanlov operatoriga o‘xshaydi, lekin, aslida, variant qismning boshini ko‘rsatuvchi o‘ziga xos xizmat so‘zi rolini o‘ynaydi. Aynan shu uchun variant qism oxirida CASE ... OFning jufti sifatida END so‘zini qo‘yish kerak emas (chunki variant qismi — yozuvda hamma vaqt oxirgi, undan keyin RECORDning jufti sifatida END turadi). CASE ... OF iborada tanlov kaliti kompilator tomonidan e’tiborga olinmaydi: *Turbo Paskal* tomonidan unga qo‘yiladigan yagona talab shundan iboratki, kalit qandaydir standart yoki oldindan e’lon qilingan tartib turini aniqlaydi. Bu turning o‘zi esa quyida keladigan variantli maydonlarga ham, tanlov o‘zgarmasi tavsifiga ham hech ta’sir qilmaydi. Paskalning standart versiyasida tanlov kaliti sifatida tartib turidagi qandaydir o‘zgaruvchini ko‘rsatish zarur, bunda dasturning bajariladigan qismida bu o‘zgaruvchining qiymatini o‘zlashtirish va shunday tarzda maydonlar tanloviga ta’sir qilish mumkin. *Turbo Paskal*da, shuningdek, tanlov kaliti maydonida tartib turidagi o‘zgaruvchini ko‘rsatish va, hatto unga, dasturda, qiymat ham berish mumkin, bu maydon tanloviga ta’sir qilmaydi. *Turbo Paskal*da

tanlov o‘zgarmaslari ixtiyoriy, shu jumladan, takrorlanuvchi bo‘lishi mumkin, masalan,

```
type
        rec1=record
                a:Byte;
                b:Word;
        end;
        rec2=record
                c:Longint;
                case x:Byte of
                1:(d:Word);
                2:(e:record
                        case Boolean of
                                3: (f:rec1);
                                3: (g:single);
                                '3': (c:Word)
                        end)
                end;
var
        r:rec2
begin
        r.x:=255
        if r.e.g=0 then
                writeln ('o.k.')
        else
                writeln (r.e.g)
end.
```

Bu misolda yozuvdagi E maydonga aniqlanuvchi *Case Boolean of* ibora, faqat TRUE va FALSE — ikkita qiymatga ega bo‘luvchi, mantiqiy turni tanlov kaliti deb e’lon qiladi. Bundan keyin keladigan variantlarda tanlov o‘zgarmaslari bu turga umuman xos bo‘lmagan qiymatlarga ega bo‘libgina qolmasdan, balki ulardan yana ikkitasi takrorlanadi ham, variantlarning umumiy soni esa, kutilganidek ikkita emas, balki uchta bo‘ladi.

Maydon ismlari, yozuvning ular e’lon qilingan chegarasida, o‘ziga xos bo‘lishi kerak, lekin, agar yozuvlar maydon — yozuvlarga

ega bo‘lsa, ya’ni biri ikkinchisining ichiga kirgan bo‘lsa, ismlari ichma-ichlikning har xil pog‘onalaridan takrorlanishi mumkin (oxirgi misolda C maydonga qarang).

9.3. To‘plamlar

To‘plam — bir turli bir-biri bilan mantiqan bog‘liq obyektlar birlashmasidir. Bog‘liqliklar xususiyatlari faqat dasturchi tomonidan tushuniladi, *Turbo Paskal* uni nazorat qilmaydi. To‘plamga kiruvchi elementlar miqdori 0 dan 256 gacha bo‘lgan chegarada o‘zgarishi mumkin (tarkibida element bo‘lmagan to‘plam bo‘sh to‘plam deyiladi). Elementlar miqdorining barqarorligi bilan to‘plamlar massivlar va yozuvlardan farq qiladi.

Tarkibidagi hamma elementlari bir xil bo‘lganda (bunda elementlarning to‘plamda kelish tartibi ahamiyatsiz) va faqat shundagina ikki to‘plam teng kuchli deyiladi. Agar bir to‘plamning hamma elementlari, shuningdek, boshqasiga ham kirsa, birinchi to‘plam ikkinchisiga kirgan, deb aytiladi. Bo‘sh to‘plam ixtiyoriy boshqa to‘plamga kiradi.

To‘plamning aniqlanishi va berilishiga misol:

```
type
        digitChar=set of '0' .. '9';
        digit=set of 0 .. 9;
var
        s1,s2,s3:digitChar;
        s4,s5,s6:digit;
begin
        .....
        s1:=['1','2','3'];
        s2:=['3','2','1'];
        s3:=['2','3'];
        s4:=[0 .. 3,6];
        s5:=[4,5];
        s6:=[3 .. 9];
        .....
end.
```

Bu misolda *s*1 va *s*2 to‘plamlar teng kuchli, *s*3 to‘plam esa *s*2 ga kiradi, lekin unga teng kuchli emas.

To‘plam turi quyidagi ko‘rinishda bayon etiladi:

<tur ismi> = set of <baz.turi>

bu yerda,

<tur ismi> — to‘g‘ri identifikator;

set, of — rezerv so‘zlar (to‘plam, ... dan)j

<baz.tur> — elementlarning to‘plamdagi bazaviy turi, bu tur sifatida WORD, INTEGER, LONGINT turlaridan boshqa ixtiyoriy tartib turini ishlatish mumkin.

To‘plamni berish uchun to‘plam konstruktori ishlatiladi. To‘plam konstruktori, bu bir-biridan vergullar bilan ajratiluvchi to‘plam elementlari xususiyatlari ro‘yxatidir, ro‘yxat to‘rtburchakli qavslarga olinadi (yuqoridagi misolga qarang). O‘zgarmaslar, bazali turdagi ifodalar, shuningdek, shu bazali turdagi tur-soha elementlar xususiyati bo‘lishi mumkin.

To‘plamlar ustida quyidagi amallar aniqlangan:

* to‘plamlarning kesishishi; natija ikkala to‘plam uchun umumiy bo‘lgan elementlarga ega bo‘ladi; masalan, *s*4**s*6 [3] ga ega, *s*4**s*5 — bo‘sh to‘plam (yuqoridagi misol);

\+ to‘plamlarning birlashishi; natija birinchi to‘plamning hamma va ikkinchi to‘plamning yetishmagan elementlaridan iborat bo‘ladi:
*s*4+*s*5 [0,1,2,3,4,5,6] ga ega;
*s*5+*s*6 [3,4,5,6,7,8,9] ga ega;

— to‘plamlar farqi, natija birinchi to‘plamning ikkinchi to‘plamda bo‘lmagan elementlaridan iborat bo‘ladi:
*s*6-*s*5 [3,6,7,8,9] ga ega;
*s*4-*s*5 [0,1,2,3,6] ga ega;

= teng kuchliligini tekshirish, agar ikkala to‘plam teng kuchli bo‘lsa, TRUEni qaytaradi;

<> teng kuchlimaslikni tekshirish; agar ikkala to‘plam teng kuchli bo‘lmasa, TRUEni qaytaradi;

<= kirishni tekshirish; agar birinchi to‘plam ikkinchisiga kiritilgan bo‘lsa, TRUEni qaytaradi;

>= kirishni tekshirish; agar ikkinchi to‘plam birinchisiga kiritilgan bo‘lsa, TRUEni qaytaradi;

IN tegishlilikni tekshirish; bu binar amalda birinchi element ifoda, ikkinchisi bir xil o‘sha turdagi to‘plam, agar ifoda to‘plamga tegishli qiymatga ega bo‘lsa, TRUEni qaytaradi:

*3 in s6 TRUE*ni qaytaradi;

*2*2 in s1 FALSE*ni qaytaradi.

Bu amallarga qo‘shimcha qilib ikkita protsedurani ishlatish mumkin.

INCLUDE — to‘plamga yangi elementni kiritadi. Protseduraga murojaat: INCLUDE (S, I)

bu yerda:

S — *bazali TSetBase* turidagi elementlardan iborat to‘plam;

I — *TSetBase* turidagi to‘plamga kiritilishi zarur bo‘lgan element.

EXCLUDE — to‘plamdagi elementni o‘chiradi. Unga murojaat:

EXCLUDE (S,I)

Murojaat parametrlari xuddi INCLUDE protsedurasidagi kabi.

Ikki to‘plam ustida + (plus) va — (minus) amallariga o‘xshash ishlarni amalga oshiruvchi bu protseduralar + va — amallari to‘plamining yakka elementlari bilan ishlash uchun optimallashtirilganligi orqali farqlanadi va shuning uchun yuqori bajarish tezligi bilan ajralib turadi.

To‘plamlar bilan ishlash yo‘llarini namoyish etuvchi 9.8-dasturda, natural sonlarning birinchi yuztaligidan hamma oddiy sonlarni ajratib olish amalga oshiriladi. Uning asosiga «Eratosfen elagi» deb ataluvchi usul yotadi. Bu algoritmga ko‘ra, avval 2 dan *N* gacha bo‘lgan sohadagi hamma butun sonlardan iborat BEGINSET to‘plam tashkil topadi. Izlanayotgan oddiy sonlardan iborat bo‘lishi kerak bo‘lgan Misol SET to‘plamga 1 joylashtiriladi. Keyin quyida keltirilgan ishlar bir necha marta takrorlanadi:

- BEGINSETdan birinchi NEXT soni olinsin va Misol SET ga joylashtirilsin;
- BEGINSETdan NEXT soni va unga bo‘linuvchi boshqa sonlar, ya’ni 2*NEXT, 3*NEXT va hokazo yo‘qotilsin.

Ish BEGINSET to‘plam bo‘sh bo‘lgunicha takrorlanadi.

Bu dasturni ixtiyoriy *N* uchun ishlatish mumkin emas, chunki ixtiyoriy to‘plamda elementlar soni 256 dan ko‘p bo‘lmaydi.

```
Program Misol_numbers_detect;
{Birinchi N butunlardan hamma oddiy sonlarni ajratish}
const
        N=255;     {Boshlang‘ich to‘plamdagi elementlar soni}
type
        setOfNumber=set of 1 .. N;
var
        n1, next, i: Word;      {Yordamchi o‘zgaruvchilar}
        BeginSet;               {Boshlang‘ich to‘plam}
        MisolSet:SetOfNumber{Oddiy sonlar to‘plami}
begin
        BeginSet:=[2 .. N];     {Boshlang‘ich to‘plam yaratish}
        MisolSet:=[1];          {Birinchi oddiy son}
        next :=2;               {Navbatdagi oddiy son}
        while BeginSet <> [] do             {Asosiy sikl boshi}
        begin
              n1:= next; {n1-navbatdagi oddiy (next)ga bo‘li-
                          nadigan son}
{Boshlang‘ich to‘plamdan oddiymas sonlarni yo‘qotuvchi sikl}
              while n1<= N do
              begin
                    Exclude (BeginSet, n1);
                    n1:=n1+next     {navbatdagi bo‘linuvchi}
              end; {Yo‘qotish siklining oxiri}
Include (MisolSet, next);
{Boshlang‘ich to‘plamdan birinchi o‘chirilmagan navbatdagi
oddiy sonni hosil qilish}
        repeat
              inc (next)
        until (next in BeginSet) or (next>N)
  end;              {asosiy sikl oxiri}
  {natijalarni chiqarish}
  for i:=1 to N do
        if i in MisolSet then write (i:8);
              writeln
        END.
```

To‘plamni ko‘zdan kechirishni tugatishdan oldin katta bo‘lmagan tajribani o‘tkazish foydadan xoli emas. SETOFNUMBER turi bayonini quyidagicha o‘zgartiramiz:

```
type
        SetOfNumber = set of 1 .. 1;
```

dasturni yana bir marta ishga tushirib, ekranga quyidagilarni chiqaramiz:

```
1       3           5           7
```

BeginSet va *MisolSet* to‘plamlari endi bitta elementdan iborat bo‘ladi, dastur esa ularga yettitadan kam bo‘lmagan elementni joylashtira oldi.

Buning siri oddiy: to‘plamning ichki qurilmasi shundayki, uning har bir elementiga bitta ikkilamchi razryad (bir bit) mos qilib qo‘yiladi; agar element to‘plamga kiritilgan bo‘lsa, tegishli xona (razryad) 1 qiymatiga, aks holda 0 qiymatiga ega bo‘ladi. Xotiraning minimal birligi — 8 bitga teng bo‘lgan bir bayt. Kompilator to‘plamlarga bittadan bayt ajratadi, natijada ularning har birining quvvati 8 elementga ega bo‘ldi. To‘plamning maksimal quvvati — 256 element. Bunday to‘plamlar uchun kompilator 16 tadan aralash baytlar ajratadi.

Yana bir tajriba: bazali tur sohasini 1 .. 256 ga o‘zgartiramiz. Bu turning quvvati 256 element bo‘lsa ham, dasturni kompilatsiya qilishga urinish vaqtida kompilator quyidagi axborotni chiqaradi:

```
Error 23 : Set Base type out of range
        {xato23: To‘plamning bazali turi belgilangan chegaralardan tashqariga chiqyapti}
```

Kompilator bazali tur sifatida minimal chegarasi 0, maksimali 255 bo‘lgan butun sonli turni yoki 256 dan ko‘p bo‘lmagan (sanab o‘tuvchi turning quvvati 65536 element) ixtiyoriy sanab o‘tuvchi turlarni ishlatishga ruxsat beradi.

9.4. Turlarni o‘zgartirish va ularning birgalikda bo‘lishi

Bir necha marta aytganimizdek, *Turbo Paskal* — turdoshlashtirilgan til. U turlar konsepsiyalariga qat’iy rioya qilish asosida tuzilgan, unga ko‘ra tilda qo‘llaniladigan amallar faqat birgalikda bo‘lgan operandalar uchun aniqlangan. Haqiqiy sonlar ustida amallar bajarish muhokama qilinganida haqiqiy va butun turlarning birgalikda bo‘lish muammosiga to‘xtalib o‘tilgan edi (III bob, 3-bo‘lim, 2-qism). Xuddi shunga o‘xshagan muammolar har xil uzunlikdagi satrlar, belgilar va h.k.lar ustida amallar bajarilganda ham paydo bo‘ladi. Quyida turlarning birgalikda bo‘lishining eng to‘la ta’rifi keltiriladi.

Quyidagi shartlar bajarilsa, ikki tur birgalikda deb hisoblanadi:

- ikkalasi ham bir xil turda bo‘lsa;
- ikkalasi ham bir xil haqiqiy bo‘lsa;
- ikkalasi ham bir xil butun bo‘lsa;
- bir tur ikkinchi turning tur sohasi bo‘lsa;
- ikkala tur ham bitta bazali turning tur sohalari bo‘lsa;
- ikkalasi ham bitta bazali turning elementlaridan tuzilgan to‘plamlar bo‘lsa;
- ikkalasi ham bir xil maksimal uzunlikdagi joylashtirilgan satrlar (oldiga PACKED so‘zi yozib aniqlangan) bo‘lsa;
- bir tur tur-satr, boshqa tur-satr, joylashtirilgan satr yoki ramzli bo‘lsa;
- bir tur ixtiyoriy, ikkinchisi turdoshmas ko‘rsatkich bo‘lsa;
- bir tur obyektga, ikkinchisi unga qarindosh bo‘lgan obyektga ko‘rsatkich bo‘lsa;
- har ikkisi ham bir xil turdagi natijaga, parametrlar soniga, o‘zaro mos keluvchi parametrlar turiga ega protsedurali turlar bo‘lsa.

Turlarning birgalikda kelishi o‘zlashtirish operatorlarida alohida ma’no kasb etadi. T1 o‘zgaruvchi, T2 ifoda turi bo‘lsin, ya’ni T1:=T2 o‘zlashtirish bajariladi. Bu o‘zlashtirish quyidagi hollarda mumkin:

- T1 va T2 bir xil tur va bu tur-fayllarga yoki fayl massivlariga, yoki maydon-fayllariga ega yozuvlarga, yoki shunday yozuvlar massivlariga tegishli bo‘lmaydi;

- T1 va T2 birgalikdagi tartibli turlar va T2 qiymat T1ning mumkin bo‘lgan qiymatlar sohasida yotadi;
- T1 va T2 haqiqiy turlar va T2 qiymat T1ning mumkin bo‘lgan qiymatlar sohasida yotadi;
- T1 haqiqiy va T2 butun tur;
- T1 satr va T2 joylashtirilgan satr;
- T1 va T2 birgalikda joylashtirilgan satrlar;
- T1 va T2 birgalikdagi to‘plamlar va T2 ning hamma a’zolari T1 ning mumkin bo‘lgan qiymatlar to‘plamiga tegishli;
- T1 va T2 birgalikdagi ko‘rsatkichlar;
- T1 va T2 birgalikdagi ko‘rsatkichlarning protsedurali turlari;
- T1 obyekt va T2 uning avlodi.

Dasturda bir turdagi ma’lumotlar boshqa turdagi ma’lumotlarga o‘zgartirilishi mumkin. Bu o‘zgartirish ochiq va yashirincha bo‘lishi mumkin.

Turlarni ochiq o‘zgartirishda argumenti bitta turga, qiymati boshqa turga tegishli bo‘lgan maxsus o‘zgartirish funksiyalarining chaqirilishi ishlatiladi. ORD, TRUNC, ROUND, CHR funksiyalar shunday funksiyalardir.

*Turbo Paskal*da turlarni o‘zgartirishning yana ham umumiyroq mexanizmi ishlatilishi mumkin. Unga ko‘ra, o‘zgartirishga standart yoki turi foydalanuvchi tomonidan aniqlangan identifikatorni, turi o‘zgartiriladigan ifodaga, o‘zgartirish funksiyasining identifikatori kabi qo‘llanilganda (turlarni o‘zgartirishning avtoaniqlashi) erishiladi. Masalan, funksiyaning quyidagi chaqirishlariga yo‘l qo‘yiladi:

```
type
        MyType=(a, b, c, d);
.....
        MyType (2);
        Integer ('D');
        Pointer (longint (a) + $ FF);
        Char (127 mod c);
        Byte (k);
```

Ifoda turini o‘zgartirishning avtoaniqlashida uning ichki ifoda-

lanish uzunligining o‘zgarishi sodir bo‘lishi (uzunlik o‘sishi yoki kamayishi) mumkin.

*Turbo Paskal*da ma’lumotlarni o‘zgartirishning yana bir ochiq usuli mavjud: qandaydir turdagi o‘zgaruvchi egallagan xotira sohasiga, agar faqat yangi joylashtiriladigan qiymatning ichki ifodalanish uzunligi o‘zgaruvchini ichki ifodalash uzunligiga aniq teng bo‘lsa, boshqa turdagi ifoda qiymatini joylashtirish mumkin. Shu maqsadda turlarni o‘zgartirishning avtoaniqlangan funksiyasi qayta, lekin endi o‘zlashtirish operatorining chap qismida ishlatiladi:

```
type
        byt= array [1 .. 2] of Byte;
        int= array [1 .. 2] of Integer;
        rec=record
                        x,y:Integer
              end;
var
        vbyt:byt;
        vint:int;
        vrec:rec;
begin
        byt(vint[1])  [2]:=0;
        int(vrec)  [1]  :=256
end.
```

Turlarni yashirin holda o‘zgartirish faqat ikki holda mumkin:

- haqiqiy va butun sonli o‘zgaruvchilardan tuzilgan ifodalarda, butun sonli o‘zgaruvchilar haqiqiy o‘zgaruvchilarga avtomatik tarzda o‘zgartiriladi va hamma ifodalar butunicha haqiqiy tur ko‘rinishini oladi;
- xotiraning bitta sohasi navbat bilan u yoki bu turlardagi ma’lumotlarga ega (har xil turdagi ifodalarning xotiraga birlashtirilishi), deb tushuntiriladi.

Ma’lumotlarni xotiraga birlashuvi variantli maydon yozuvlaridan bir xil adresga ega turdoshlashtirilgan ko‘rsatkichlardan

foydalanilganda, shuningdek, har xil turdagi ma'lumotlarni bitta va o'sha mutlaq adres bo'yicha ochiq joylashtirilganda sodir bo'lishi mumkin. O'zgaruvchini kerakli mutlaq adres bo'yicha joylashtirish uchun avval u, ketidan ABSOLUTE standart direktiva, uning orqasidan yoki mutlaq adres, yoki avval aniqlangan o'zgaruvchi indeksini joylashtirish bilan bayon etiladi. Mutlaq adres WORD turidagi ikki nuqta bilan ajratilgan sonlar juftligi bilan ko'rsatiladi; birinchi son segment, ikkinchisi — adreslar siljishi sifatida (II bob) ko'rsatiladi. Masalan,

```
b:Byte absolute $0000 : $0055;
w:LongInt absolute 128:0.
```

Agar ABSOLUTE so'zidan keyin avval aniqlangan o'zgaruvchining identifikatori ko'rsatilgan bo'lsa, xotirada har xil turdagi o'zgaruvchilarning siljishi sodir bo'ladi, bunda bu ma'lumotlarni ichki ifodalash bir adres bo'yicha olib boriladi, masalan,

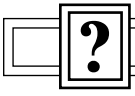
```
var
        x:REAL;
        y:array [1...3] of Integer absolute x.
```

Bu misolda X va Y o'zgaruvchilar bitta mutlaq adres bo'yicha boshlanib, joylashtiriladi. Shunday qilib, uzunligi 6 bayt bo'lgan xotiraning bitta sohasini va, demak, bu sohada joylashtirilgan ma'lumotlarni REAL turdagi ma'lumotlar, yoki INTEGER turidagi uchta ma'lumotli massiv, deb qarash mumkin. Masalan, quyidagi dastur ekranga π=3,1415 haqiqiy son ichki ifodalanishining birinchi ikki bayti tarkibini butun son ko'rinishda chiqaradi:

```
var
        x:Real;
        y:array [1 .. 3] of Integer absolute x;
begin
        x:=pi;
        writeln (y[1])
end.
```

Ekranga 8578 natija chiqariladi.

Turlarni yashirin o‘zgartirish dasturda topilishi qiyin bo‘lgan xatolarni topish manbayi bo‘lib xizmat qiladi, shuning uchun, qayerda mumkin bo‘lsa, ulardan qochish kerak.



Nazorat savollari

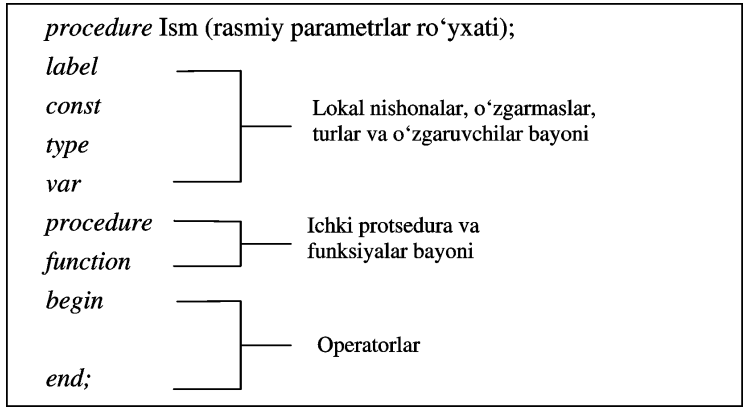
1. Ma’lumotlarning murakkab turlariga nimalar kiradi?
2. Massivlar nima?
3. Necha xil massivlar bo‘ladi, ular Paskal tilida qanday bayon etiladi?
4. Massivlarni saralashning qanday usullari bor, ularga qanday guruhlar kiradi?
5. Saralash masalasini yechishda algoritmlarning ish tezligi qanday ko‘rsatkichlar bilan baholanadi?
6. Kiritish yo‘li bilan saralash usulining mohiyati.
7. Tanlash yo‘li bilan saralash usulining mohiyati.
8. Almashtirish yo‘li bilan saralash usulining mohiyati.
9. Ikkilamchi izlash usuli mohiyati, uning tamoyillari.
10. Ikki o‘lchovli massivlarning o‘ziga xos xususiyatlari nima?
11. Matritsa «spiral» bo‘yicha aylanganda indekslar o‘zgarishining qonuniyati qanday bo‘ladi?
12. Satrlar *Turbo Paskal*da nimani ifodalaydi?
13. Satrlarning massivdan farqi nima?
14. Satrlar ustida qanday amallar bajarish mumkin?
15. Satrlarning qanday standart protsedura va funksiyalari mavjud?
16. Yozuv nima, yozuv turi qanday e’lon qilinadi?
17. Yozuvlar ustida qanday amallar bajarish mumkin?
18. To‘plam nima?
19. To‘plamlar massivlar va yozuvlardan nimasi bilan farq qiladi?
20. Qachon to‘plamlar teng kuchli va aksincha bo‘ladi, misollar keltiring?
21. To‘plamning qanday turlari bor?
22. To‘plamlar ustida qanday amallar bajarish mumkin?
23. Ikki turning birgalikda bo‘lishining qanday shartlari bor?
24. *Turbo Paskal*da turlar qanday o‘zgartiriladi?

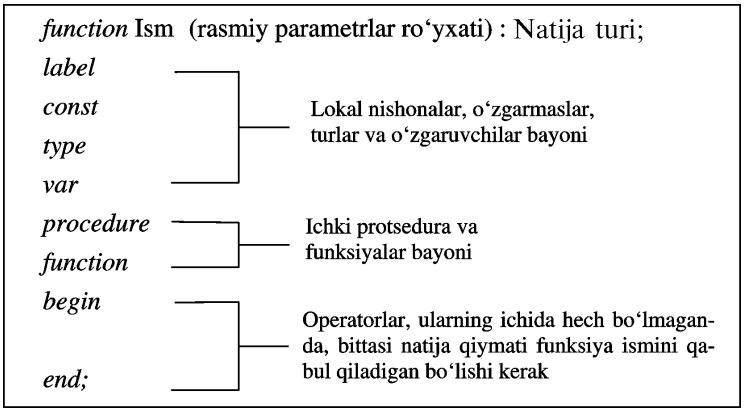
X bob. PROTSEDURA VA FUNKSIYALAR

10.1. Protsedura va funksiyalar tarkibi

Murakkab masalani yechishda uni ichki masalalarga bo‘lib, ularni esa, o‘z navbatida, kichikroqlarga, ular yana ham kichikroqlariga va shunday tarzda, dasturi oson tuziladigan masalalarga borib yetguncha bo‘lib yechiladi. *Turbo Paskal* dasturni qismlarga bo‘lishning har xil vositalari mavjud. Bo‘linishning eng yuqori pog‘onasida (katta masalalarda) modullar, quyi pog‘onada (eng oddiy ichki masalalarda) protsedura va funksiyalar (ko‘p hollarda) joylashgan bo‘ladi. *Obyekt—yo‘naltirilgan uslubiyat* dasturlar ishlab chiqishning yuqori va, shuningdek, quyi pog‘onalarini o‘z ichiga oladi.

Protsedura va funksiyalar ko‘pchilik dasturlash tillarida muhim vosita hisoblanadi. Ular yordamida qandaydir bir yagona harakatni bajarish uchun, operatorlar guruhini tashkil qilish mumkin. Protsedura (funksiya)ni dasturning har xil joylaridan chaqirish mumkin. Protsedura (funksiya) hisoblangan natijalarni qaytaradi va hisoblashlarni bajarishda ishlatiladigan axborotlarni qabul qiladi. Protsedura va funksiyalar operatorlar, kichik ma’lumotlar va ichki protsedura va funksiyalardan iborat. Protsedura va funksiyalarni bayon etish tasnifi quyidagi ko‘rinishga ega:





Keltirilgan tasnifdan ko'rinib turibdiki, protsedura va funksiyalarni bayon etishdagi farqlar faqat sarlavha va operatorlar bo'limiga tegishli. Bu farqlarni namoyish etish uchun bir masalaning ikki xil — biri protsedura, ikkinchisi funksiya yordamidagi yechimini keltiramiz.

Masala. Elementlari haqiqiy sonlar bo'lgan *n* o'lchamli massiv elementlari yig'indisini topish kerak bo'lsin.

Protseduradan foydalanib yechish:

```
program Psumma;
const
        n=20;                {massiv uzunligi}
type
        TVector=array [1..n] of Real;
var
        Vector : TVector;
        i      : Integer;
procedure Summa (Vec: TVector; Len: Integer; Name: String);
var
        i      : Integer;
        S      : Real;

begin
        S      : = 0;
        for    i : = 1 to Len do
               S : = S + Vec [i];
     Writeln ('Vektor elementlari yig'indisi ', Name, '=' , s:7:2)
end;
```

```
begin
        Writeln ('Massiv elementlarini kiriting:');
        for   i : = 1 to n do Read (Vector[i]); Readln;
{-----------------------------------------------------------}
{Protsedurani chaqirish alohida operator bilan bajariladi!}
                Summa (Vector, n, 'Vector');
{-----------------------------------------------------------}
end.
```

Masalani funksiyadan foydalanib yechish:

```
program Psumma;
const
        n=20;                {massiv uzunligi}
type
        TVector=array [1..n] of Real;
var
        Vector : TVector;
        Sum  : Real;
        i      : Integer;
function Summa (Vec: TVector; Len: Integer) : Real;
var
        i      : Integer;
        S      : Real;


begin
        S      : = 0;
        for    i : = 1 to Len do
               S : = S + Vec [i];
        Summa : = S {Natija qiymatiga funksiya ismini berish}
                    {funksiya tanasida zaruriy operator bo'lib
                    hisoblanadi}
end;
begin
        Writeln ('Massiv elementlarini kiriting:');
        for    i : = 1 to n do Read (Vector[i]); Readln;
{-----------------------------------------------------------}
{Ifodalarni yozish mumkin bo'lgan joyga xususiy holda o'zlashtirish}
{operatorining o'ng qismiga, funksiyani chaqirish mumkin}
        Sum : = Summ (Vector, n);
```

```
Writeln ('Vektor elementlari yig'indisi: Vector =', Sum:7:2);
{yoki bevosita Writeln chiqarish protsedurasi elementlari kabi}
        Writeln    ('Vektor elementlari yig'indisi Vector = ',
                                Summa (Vector, n))
end.
```

10.2. Protsedura va funksiyalardan foydalanishda identifikatorlarning faoliyat sohasi

Identifikator faoliyat sohasi deb, identifikator ishlatilishi mumkin bo'lgan dastur qismiga aytiladi.

Identifikator faoliyat sohasi ularni e'lon qilish o'rni bilan aniqlanadi. Agar identifikatorlar bir protsedura yoki funksiya doirasida ishlatilishi mumkin bo'lsa, ular lokal deyiladi. Agar identifikatorlar ta'siri bittadan kam bo'lmagan (bir nechta) ichma-ich kiritilgan protsedura va (yoki) funksiyalarga tarqalsa, bunday identifikatorlar *global* deyiladi.

«Lokal» va «global» tushunchalar, ma'lum protsedura (funksiya)ga ko'ra nisbatan, deb tushunilishi kerak.

Buni quyidagi misol bilan namoyish etamiz:

```
program Scope;
  ar A0, B0, C0 : Integer;
  procedure P1;
    var A1, B1, C1 : Integer;
    procedure P2;
      var A2, B2, C2 : Integer;
        begin
            {global A0, B0, C0, A1, B1, C1larni,
            shuningdek, lokal A2, B2, C2larni
            ishlatish mumkin}
      end;
    begin
      {global A0, B0, C0 larni, shuningdek, lokal
      A1, B1, C1 larni ishlatish mumkin}
  end.
begin
  {Faqat A0, B0, C0ni ishlatish mumkin}
end.
```

Bu misolda dasturda ishlatilayotgan hamma protsedura va funksiyalar uchun AO, BO, CO global bo‘ladi.

P1 protsedura ichida (bu misolda P2 protsedura uchun) bayon etilgan hamma protsedura va funksiyalar uchun A1, B1, C1 global, bir vaqtda P1 protseduraning o‘zi uchun lokal bo‘ladi.

P2 eng ichki, protsedurada e’lon qilingan A2, B2, C2 ma’lumotlar faqat lokal bo‘ladi.

Protsedura va funksiya identifikatorlari uchun ta’sir sohasini aniqlash qoidasini keltiramiz:

- protsedura (funksiya) ichida aniqlangan hamma identifikatorlar faoliyat ko‘rsatadi;
- protsedura (funksiya) ichida e’lon qilingan ismlar bilan farq qiluvchi hamma identifikatorlar faoliyat ko‘rsatadi;
- protsedura (funksiya) lokal identifikatorlari tashqi muhitda hech qachon faoliyat ko‘rsatmaydi;
- lokal va global identifikator ismlari mos kelgan paytda, faqat ichki lokal identifikator faoliyat ko‘rsatadi.

Birinchi uch qoida ko‘rilgan misol bilan tushuntirilgan bo‘lsa, to‘rtinchisini tushuntirish uchun yana bir misol keltiramiz:

```
program EqualNames;
var A, B, C : Integer;
    procedure P1;
    var A, B, C : Real;
        procedure P2;
        var A, B, C : Char;
        begin
            {A, B, C — ramzli o‘zgaruvchilar kabi
            ishlatiladi}
        end;
      begin
            {A, B, C — haqiqiy o‘zgaruvchilar kabi
            ishlatiladi}
      end.
begin
      {A, B, C — butun o‘zgaruvchilar kabi ishlatiladi}

end.
```

Ya'ni ismiga ko'ra tashqi protsedura ismlari bilan mos keluvchi identifikatorli ma'lumotlarni ichki protsedurada e'lon qilish tashqi identifikatorlar ta'sirini bekor qiladi va turi bo'yicha mos keladimi, yo'qmi o'zining lokal bayonlarini kiritadi.

Lokal ma'lumotlar protsedura (funksiya)ni chaqirishda yaratiladi va faqat uning bajarilish vaqtida mavjud bo'ladi. Lokal ma'lumotlar uchun xotirani ajratish, avtomatik ravishda, protsedura (funksiya) bajarilishining boshida, bu xotirani bo'shatish esa, protsedura (funksiya) bajarilishining tugashi bilan sodir bo'ladi.

Protsedura (funksiya) tanasida joylashgan operatorlar uning lokal ma'lumotlariga (o'zgarmas va o'zgaruvchilarga) murojaat qilishi va ularning qiymatini o'zgartirishi mumkin.

Lokal ma'lumotlar qiymati protsedura (funksiya) ishlagan vaqtgacha mavjud bo'ladi. U tugashi bilan protsedura (funksiya) operatorlari bilan qilingan lokal ma'lumotlar qiymatlarining barcha o'zgarishlari xotiraning bo'shashi bilan yo'qoladi.

10.3. Parametrlarni uzatish usullarining tasnifi

Protsedura (funksiya) faollashtirilganda unga parametrlarni uzatish mumkin. Uzatishda protsedura (funksiya) sarlavhasida bayon etilgan parametrlarning hammasi chaqirish operatorida ko'rsatilishini kuzatish zarur.

Protsedura (funksiya)ni bayon etishda uning sarlavhasida ko'rsatiladigan parametrlar *rasmiy parametrlar* deyiladi.

Protsedura (funksiya)ni chaqirishda ko'rsatiladigan parametrlar *haqiqiy parametrlar* deyiladi.

Ko'p protseduralar bir nechta parametrlarga ega. Dasturchining vazifasi — u chaqirishda ko'rsatayotgan parametrlarning ma'nosiga ko'ra, rasmiy parametrlarga mos kelishiga ishonch hosil qilishdir. Kompilator parametrlarning xato sonini va turlarning nomutanosibliginigina tekshira oladi.

Dasturlash tillarida amalga oshirilishi mumkin bo'lgan parametrlarni uzatish usullari quyidagi tasnifga ega:

Parametrlar quyidagi belgilariga ko'ra farqlanadi:

1. Uzatish mexanizmiga ko'ra:

a) qiymatiga ko‘ra uzatish (*value*);
b) adresi (murojaati)ga ko‘ra uzatish (*addr*).
2. Chaqiruvchi va chaqiriluvchi protsedura (funksiya)ning o‘zaro faoliyatiga ko‘ra:
a) faqat kiruvchi parametr sifatida (*in*);
b) faqat chiquvchi parametr sifatida (*out*);
d) kiruvchi, xuddi shunday chiquvchi sifatida (*inout*).

Bu farqlarga nazariy jihatdan parametrlarni uzatishning olti usuli mos kelishi mumkin:

1) *value in;*
2) *value out;*
3) *value inout;*
4) *addr in;*
5) *addr out;*
6) *addr inout.*

Mavjud dasturlash tillarida, parametrlarni uzatishning olti usuli mavjudligiga qaramasdan, odatda, ikki-uch usuligina ishlatiladi.

*Turbo Paskal*da parametrlarni uzatishning birinchi (*value in*), to‘rtinchi (*addr in*) va oltinchi (*addr inout*) usullarigina amalga oshiriladi. Mana shu uch usul bilan yaqindan tanishamiz:

value in parametrlari

1. Protsedura (funksiya) chaqirilganda:

a) rasmiy parametrlar va lokal ma’lumotlar uchun xotira ajratiladi (stekda yoki lokal ma’lumotlar uchun xotiraning maxsus joyida);

b) xotirada haqiqiy parametrlar qiymatlaridan rasmiy parametrlar uchun ajratilgan nusxa olish bajariladi.

2. Protsedura (funksiya) ishi vaqtida:

rasmiy parametrlar qiymatlarining o‘zgarishi haqiqiy parametrlarning xotira yacheykalaridagi tarkibiga hech qanday ta’sir etmaydi.

3. Protsedura (funksiya)ning tugashida:

a) rasmiy parametrlar va lokal ma’lumotlar uchun ajratilgan xotira tozalanadi;

b) protseduraning ish vaqtida olingan, rasmiy parametrlarning yangi qiymatlari xotira tozalanishi bilan birga tozalanadi.

addr in parametrlari

1. Protsedura (funksiya)ni chaqirishda:

a) faqat lokal ma'lumotlar va haqiqiy parametrlar adreslarini saqlash uchungina xotira ajratish (stekda yoki lokal ma'lumotlar uchun maxsus sohada) bajariladi;

b) haqiqiy parametrlar adreslarini (lekin qiymatlarini emas) ular uchun ajratilgan xotiraga nusxalash bajariladi.

2. Protsedura (funksiya)ning ishi vaqtida:

a) rasmiy parametrlar qiymatlarini o'zgartirish mumkin emas;

b) rasmiy parametrlar qiymatlarini faqat boshlang'ich ma'lumotlar sifatida ishlatish mumkin;

d) rasmiy parametr qiymatlari, nusxalangan adreslarni ishlatib, bevosita haqiqiy parametrlar xotirasidan tanlanadi.

3. Protsedura (funksiya)ning tugashida:

a) chaqirilgan protsedura (funksiya)ning chaqiruvchiga bunday parametrlar orqali ta'siri bo'lmaydi;

b) protsedura (funksiya) ishi uchun ajratilgan xotira tozalanadi.

addr inout parametrlari

1. Protsedura (funksiya) chaqirilganda:

a) faqat lokal ma'lumotlar va haqiqiy parametrlar adreslarini saqlash uchungina xotira ajratish (stekda yoki lokal ma'lumotlar uchun maxsus sohada) bajariladi;

b) haqiqiy parametrlar adreslarini (lekin qiymatlarini emas) ular uchun ajratilgan xotiraga nusxalash bajariladi;

d) haqiqiy parametrlar sifatida o'zgarmaslarni ishlatish man etiladi.

2. Protsedura (funksiya)ning ishi vaqtida:

a) bunday turdagi parametrlarning ishlatilishi uchun hech qanday chekliklar qo‘yilmaydi;

b) nusxalangan adreslardan foydalanib, rasmiy parametrlarni o‘zgartirish bevosita haqiqiy parametrlarga tegishli bo‘lgan xotira yacheykalarida bajariladi.

3. Protsedura (funksiya)ning tugashida:

a) natijani ataylab nusxalash talab qilinmaydi, chunki rasmiy parametrlar bilan hamma amallar haqiqiy parametrlarning xotira yacheykalari ustida bevosita bajariladi;

b) protsedura (funksiya) uchun ajratilgan xotira tozalanadi.

10.4. *Turbo Paskal*da parametrlarni uzatish

*Turbo Paskal*da parametrlarni uzatishning yuqorida aytilgan olti usulidan uchtasi amalga oshiriladi:

1. *value in* ko‘rinishidagi parametrlar.

*Turbo Paskal*da bunday parametrlar *parametr-qiymatlar* deyiladi. Protsedura (funksiya) sarlavhalari bayonida parametr-qiymatlar identifikatorlaridan oldin qo‘shimcha kalit so‘zlar qo‘yilmaydi.

2. *addr inout* ko‘rinishidagi parametrlar.

*Turbo Paskal*da bunday parametrlar *parametr-o‘zgaruvchilar* deyiladi. Protsedura (funksiya) sarlavhasini bayon etishda parametr-o‘zgaruvchilar identifikatorlaridan oldin *var* kalit so‘zi qo‘yiladi.

3. *addr in* ko‘rinishidagi parametrlar.

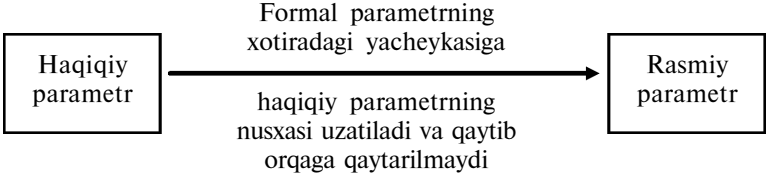
*Turbo Paskal*da bunday parametrlar *parametr-o‘zgarmaslar* deyiladi. Protsedura (funksiya) sarlavhasi bayonida parametr-o‘zgarmaslar identifikatoridan oldin *cons* kalit so‘zi yoziladi.

Parametrlarning dastlabki ikki turi Paskal tilining hamma versiyalariga xos. Uchinchi, parametr-o‘zgaruvchi turi esa *Turbo Paskal* 7-versiyasigagina yangi kiritilgan.

10.4.1. Parametr-qiymatlar. Parametr-qiymatlar bayon etilgan protsedura sarlavhasi quyidagi ko‘rinishga ega bo‘ladi:

Procedure MyProc (Par 1, *Par* 2: *Type* 1; *Par* 3, *Par* 4: *Type* 2).

Parametr-qiymatlar ish mexanizmini soddalashtirib quyidagi chizma bilan berish mumkin:



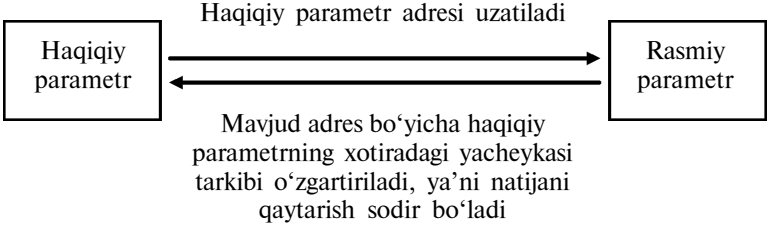
Haqiqiy parametr-qiymat sifatida o‘zgaruvchilar, shuningdek, har xil turdagi o‘zgarmaslar ishlatilishi mumkin. Faqat faylli turlar va ularga tayanadigan turlar ishlatilmaydi.

10.4.2. Parametr-o‘zgaruvchilar. Parametr-o‘zgaruvchilar bayon etilgan protsedura sarlavhasi quyidagi ko‘rinishga ega:

procedure MyProc (var Par 1, Par 2: Type 1; var Par 3, Par 4:

Type 2).

Parametr-o‘zgaruvchilarning ish mexanizmini soddalashtirib, quyidagi chizma bilan berish mumkin:

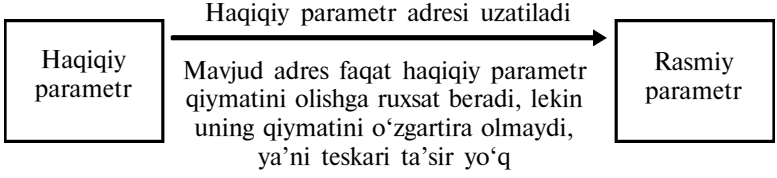


Haqiqiy parametr o‘zgaruvchi sifatida ixtiyoriy turdagi, shu jumladan, faylli va faylga tayanuvchi o‘zgaruvchilar ishlatilishi mumkin, lekin o‘zgarmaslardan foydalanishga yo‘l qo‘yilmaydi.

10.4.3. Parametr-o‘zgarmaslar. Parametr-o‘zgarmaslar bayon qilingan protsedura sarlavhasi quyidagi ko‘rinishga ega:

procedure MyProc (const Par 1, *Par* 2: *Type* 1; *const Par 3, Par* 4: *Type* 2).

Parametr-o‘zgarmaslar ishlash mexanizmini soddalashtirib, quyidagi chizma bilan berish mumkin:



Haqiqiy parametr-o‘zgarmaslar sifatida o‘zgaruvchilar ham, har xil turdagi o‘zgarmaslar ham ishlatilishi mumkin. Faqat faylli va ularga tayangan turlar ishlatilishiga yo‘l qo‘yilmaydi. Bundan tashqari, rasmiy parametr-o‘zgarmaslarga o‘zlashtirishni bajarish man etiladi va rasmiy parametr-o‘zgarmaslarni haqiqiy parametrlar sifatida boshqa protsedura (funksiya)larga uzatib bo‘lmaydi.

Xotiraning katta o‘lchamini egallaydigan, lekin parametrlarning boshlang‘ich qiymatlarini, algoritmik nuqtayi nazardan, o‘zgartirishi mumkin bo‘lmagan ma’lumotlar tarkibini uzatish talab qilingan hollarda, parametr-o‘zgarmaslardan foydalanish maqsadga muvofiq. Buning natijasida tezkor xotira tejamli foydalanilgan bo‘ladi va bir vaqtda boshlang‘ich ma’lumotlarning butun saqlanishiga kafolat beriladi.

10.4.4. Tursiz parametrlar. Tursiz parametrlar faqat adresiga ko‘ra, ya’ni parametr-o‘zgaruvchilar yoki parametr-o‘zgarmaslar kabi uzatiladi. Tursiz parametrlarning asosiy xususiyati protsedura sarlavhasida parametr turiga ko‘rsatmaning yo‘qligidir.

Tursiz parametrlar bayon etilgan protsedura sarlavhasida, uzatishning har xil usullarida quyidagi ko‘rinishga ega:

procedure MyProc (var Par 1, *Par* 2; *const Par* 3, *Par* 4).

Lekin tur yo‘qligi natijasida tursiz rasmiy parametrlarni, turli parametrlar kabi ishlatish mumkin emas. Ishlatishdan oldin tursiz rasmiy parametrni qandaydir bir turga keltirish kerak.

Tursiz parametrlar, turlilariga qaraganda, ishlatishda ancha qulay, uning to‘g‘ri qo‘llanilayotganligi to‘g‘risidagi mas’uliyat dasturchi zimmasida bo‘ladi, chunki kompilator turlarining mosligini tekshira olmaydi.

Misol sifatida, elementlari *Byte* turi sohasining ixtiyoriy turdagi, ixtiyoriy o‘lchamdagi bir o‘lchovli massivlarini kiritish, chiqarish va saralash universal protseduralari ishlatilgan dasturni ko‘ramiz. Bunda har bir massiv xotirada unga qancha kerak bo‘lsa, shuncha joy egallaydi.

Bundan tashqari, *SortVector* protsedurasi *Char* turidagi massivlarni saralashda ham muvaffaqiyat bilan ishlatilishi mumkin, chunki *ASCII* belgi kodlari *Byte* turidagi qiymatlar chegarasidan tashqariga chiqmaydi. *Inpvector* va *Printvector* protseduralar ramzli massivlar uchun to‘g‘ri kelmaydi, chunki ular sonli qiymatlarni kiritish-chiqarishga yo‘naltirilgan. Sonli qiymatlarni kiritish-chiqarish esa ramzli qiymatlarni kiritish-chiqarishdan farq qiladi. Buni quyidagi dasturda ko‘rsatamiz:

```
program Sort;
        uses Crt;
        const
                    1 = 7;
                    m = 10;
                    n = 15;
        type
              MaxVect = array [1..MaxInt] of Byte;
              TVector Byte = array [1.1l] of Byte;
              TVector 100 = array [1..m] of 1..100;
              TVector Char = array [1..1] of Char;
```

```
        var
                ByteVector : TVectorByte;
                Vector 100 : TVector100;
                CharVector : TVectorChar;
                i          : Word;
{-------------------------------------------------------}
        procedure SortVector (var Vector; Len : Word);
        var
                Min : Byte;
                Imin : Word;
                i, j : Word;
        begin
                for i : = 1 to Len-1 do
                begin
                        Min : = MaxVect (Vector) [i];
                        {Vector parametrini}
                        {MaxVect turiga keltirish}
                Imin : = i;
                for j : = i+1 to Len do
                        if MaxVect (Vector) [j] < Min then
                begin
                        Min : = MaxVect (Vector) [j];
                        Imin : = j
                end;
              MaxVect (Vector) [Imin] : = MaxVect (Vector) [i];
              MaxVect (Vector) [i] : = Min;
  end;
end;

{-------------------------------------------------------}
procedure InpVector (var Vector; Len : Word; T : String);
var i : Word;
begin
        ClrScr;
        Writeln ('Elementlarni quyidagi turda kiriting', Len :
                3, T, ' : ');
```

```
        for i : = 1 to Len do Read (MaxVect (Vector) [i] );
        Readln;
end;
{-------------------------------------------------------}
procedure PrintVector (var Vector; Len : Word);
var
        i : Word;
begin
                Writeln ('Saralangan massiv:');
                for i : = 1 to Len do Write (MaxVect (Vector) [i]: 8);
                Writeln;
                Writeln ('Enterni bosing ... ');
                Readln;
end;
begin
        InpVector (ByteVector, 1, 'Byte');
        SortVector (ByteVector, 1);
        PrintVector (ByteVector, 1);
{-------------------------------------------------------}
        InpVector (Vector100, m, '1 .. 100');
        SortVector (Vector100, m);
        PrintVector (Vector100, m);
{-------------------------------------------------------}
        ClrScr;
        Writeln ('Char turdagi elementlarni kiriting:'n:3');
        for i : = 1 to n do Read (CharVector [i]);
        Readln;
        SortVector (CharVector, n);
        Writeln ('Saralangan massiv:');
        for i : = 1 to n do Write (CharVector [i] : 8);
        Writeln;
        Writeln ('Enterni bosing ... ');
        Readln;
end.
```

10.4.5. Ochiq parametr-massivlar. Protsedura sarlavhasida *ochiq parametr-massivlar* quyidagicha bayon etiladi:

procedure OpenVector (vector:array of Type Vector).

Ochiq parametr-massiv parametr-qiymat, parametr-o‘zgarmas yoki parametr-o‘zgaruvchi bo‘lishi mumkin.

Uzatiluvchi haqiqiy parametrlar turiga ko‘ra bayon etilgan *Type Vector*ga mos kelishi kerak, lekin o‘lchamiga ko‘ra ular har xil, oddiy *Type Vector* o‘zgaruvchi yoki ixtiyoriy o‘lchamdagi, massiv bo‘lishi mumkin.

Har xil o‘lchamdagi massivlarni uzatishdagi egiluvchanlik bu massivlarni yagona rasmiy parametrlar sifatida yagona ko‘rinishida ifodalash hisobiga hosil qilinadi. Hamma rasmiy parametr-massivlar, protsedura doirasida, avtomatik ravishda, sarlavhada ko‘rsatilgan turdagi nol asosli (quyi nol chegarali) massivlar kabi bayon etiladi:

array [0 .. *N*-1] *of TypeVector;*

bu yerda, *N* — haqiqiy parametrdagi elementlar soni.

Boshqacha qilib aytganda, haqiqiy parametr-massiv indeksining haqiqiy o‘zgarish sohasi, indeksning 0 dan *N-1* gacha o‘zgarish sohasida ifodalanadi.

Uzatilgan haqiqiy parametr-massivning protsedura tanasidagi tavsifini aniqlash uchun, hamma vaqt 0 qaytaruvchi — *Low* standart funksiyasi, rasmiy parametr-massivda oxirgi element indeksini qaytaruvchi *High* standart funksiyasi va haqiqiy parametr-massiv o‘lchamini qaytaruvchi *SizeOf* funksiyasi ishlatiladi.

Ochiq parametr-massivlar yordamida oldingi misoldagiga o‘xshash muammolarni yechish mumkin. Lekin ochiq parametr-massivlar tursiz parametrlarga qaraganda oz egiluvchanlikka ega, chunki bu holda haqiqiy parametrlar sifatida faqat bir turdagi massivlar bo‘lishi mumkin.

Tursiz parametrlardan foydalanish bilan taqqoslash uchun, ochiq parametr-massivlar uchun oldingi misoldagi kabi, dasturni keltiramiz:

```
program Sort;
  uses Crt;
  const
        m=10;
        n=15;
  type
        TVector1 = array [1..m] of Byte;
        TVector2 = array [1..n] of Byte;
  var
        Vector1 : TVector1;
        Vector2 : TVector2;
  {---------------------------------}
procedure SortVector (var Vector : array of Byte);
  var   Min : Byte;
        Imin : Word;
        i, j : Word;
  begin
        for i: = 0 to High (Vector) — 1 do
        begin
              Min : = Vector [i];
              Imin : = i;
              for j : = i + 1 to High (Vector) do
                    if Vector [j] < Min then
                    begin
                          Min : = Vector [j];
                          Imin : = j
                              end;
                          Vector [Imin] : = Vector [i];
                          Vector [i] : = Min;
              end;
end;
{-----------------------------------------------------------}
procedure InpVector (var Vector : array of Byte);
var i : Word;
begin
        ClrScr;
```

```
        Writeln     ('Kiriting', (High (Vector)+1):3,
                    'Byte turdagi elementlarni:');
        for i: = 0 to High (Vector) do Read (Vector [i]);
        Readln;
end;
{-------------------------------------------------------}
procedure PrintVector (var Vector : array of Byte);
var i : Word;
begin
        Writeln ('Saralangan massiv:');
        for i: = 0 to High (Vector) do Write (Vector [i] : 8);
        Writeln;
        Writeln ('Enterni bosing ... ');
        Readln;
  end;
  begin
        InpVector (Vector1);
        SortVector (Vector1);
        PrintVector (Vector1);
{-------------------------------------------------------}
        InpVector (Vector2);
        SortVector (Vector2);
        PrintVector (Vector2);
  end.
```

10.5. Protsedurali direktivalar

10.5.1. near va far direktivalari. near va *far* direktivalar kompilatorga protseduralarni chaqirishning qaysi modeliga (*near* — yaqin yoki *far* — uzoq) ko‘ra chiqish kodini hosil qilish (generirlash) talab qilinayotganini ko‘rsatadi. Yaqin modelda chaqirish tezroq, uzog‘ida — sekinroq bajariladi. Lekin yaqin chaqirishlarning ta’sir sohasi chaqiriluvchi protsedura bayon etilgan modul chegarasidagina bo‘ladi.

near va *far* protseduralar protsedura (funksiya) bayonida uning bloki oldida ko‘rsatiladi:

```
function SH (x:Real): Real; far;
begin
        SH:=(Exp(x) — Exp(-x))/2;
end.
```

Chiqish kodi yaratilishini (generatsiyasini) ko‘rsatilgan chaqirish modellariga ko‘ra, kompilatorning {$F+} direktivasi bilan ham boshqarish mumkin. {$F+} holatida kompilirlanuvchi protsedura va funksiyalar hamma vaqt chaqirishning uzoq turiga (*far*) ega bo‘ladi, {$F-} holatida esa kompilator to‘g‘ri modelni tanlaydi. Sukut saqlanganda {$F-} direktiva ishlatiladi.

10.5.2. forward direktivasi. Protsedura va funksiyalarning ilgarilovchi, deb ataluvchi bayonlari uchun *forward* direktivasi qo‘llaniladi. U ikki protsedura o‘zaro rekursiyasini (bu tushuncha X bob, 6-bo‘limda beriladi) amalga oshirishda, shuningdek, matn tasnifini yaxshilashda va uning o‘qimishliligini oshirishda ishlatiladi. Buni quyidagi dasturda ko‘rsatamiz:

```
program Recurs;
  uses Crt;
        procedure Rec1 (i : Byte); forward;
        procedure Rec2 (i : Byte);
        begin
              Writeln ('rekursiya.');
              Rec1 (i)
        end;
        procedure Rec1;
        begin
              if i>0 then
              begin
```

```
                Write ('O‘zaro');
                Rec2 (i-1)
            end
        end;
begin   ClrScr;
            Rec1 (5)
end.
```

Natija:

```
O‘zaro rekursiya
O‘zaro rekursiya
O‘zaro rekursiya
O‘zaro rekursiya
```

10.5.3. interrupt direktivasi. To‘xtatish protseduralarini bayon etishga *interrupt* direktivasi xizmat qiladi. To‘xtatish protsedurasi sarlavhasida parametrlar sifatida registr ismlari bayon etilishi kerak:

procedure Intrpt (*Flags,* CS, IP, AX, BX, CX, DX, SI, DI, DS, ES, *BP:Word*); *interrupt*

Agar parametrlardan ba’zi birlari protsedurada ishlatilmasa, ular tushirib qoldirilishi mumkin. Lekin parametrlar ro‘yxatidagi registr ismlarining o‘zaro joylashuvini o‘zgartirish mumkin emas.

Turbo Paskal tilida mavjud boshqa protsedurali direktivalar (*export, external, assembler, inline*) faoliyatini qo‘shimcha adabiyotlardan o‘rganish mumkin.

10.6. Rekursiv munosabatlar, rekursiv funksiya va protseduralar

Rekursiya — hisoblash jarayonini tashkil qilishning shunday usuliki, bunda ichki dastur uni tashkil qiluvchi operatorlarni bajarish vaqtida o‘z-o‘ziga murojaat qiladi.

Funksiya yoki protseduraga faqat dasturning asosiy qismidangina murojaat qilmasdan, balki boshqa funksiyalar yoki protseduralardan ham murojaat qilish mumkin.

Rekursiv protseduraga oddiy misol ko‘ramiz:

```
procedure ab;
begin
        write ('A');          {1}
        ab;                   {2}
        write ('B');          {3}
end.
```

ab protseduraning birinchi operatori bajarilgandan keyin *A* harfi bosib chiqariladi. Protseduraning ikkinchi operatori shu protseduraning o‘ziga murojaat qilishdir. Demak, *ab* protsedura yana bajariladi, ya’ni keyingi *A* harfi bosib chiqariladi. Nazariy jihatdan, bu amallar cheksiz davom etadi va uchinchi operator hech qachon bajarilmaydi. Amalda esa bu dasturlarga ajratilgan resurslar (mashina vaqti yoki bosib chiqaradigan qurilma qog‘ozi) tamom bo‘lgandagina dasturning bajarilishi to‘xtaydi.

n ta *A* harf va *n* ta *B* harfdan iborat ketma-ketlikni bosib chiqaradigan yana bir protsedurani yozamiz:

```
procedure abn (n:integer);
begin
        write ('A');                {1}
        if n>1 then abn (n-1);      {2}
        write ('B');                {3}
end.
```

Dastur izohida bajarilish tartibi bo‘yicha operatorlar raqami ko‘rsatilgan (10.1-rasm), *n* o‘zgaruvchining har xil qiymatlarida qanday amallar bajarilishini ko‘rib chiqamiz.

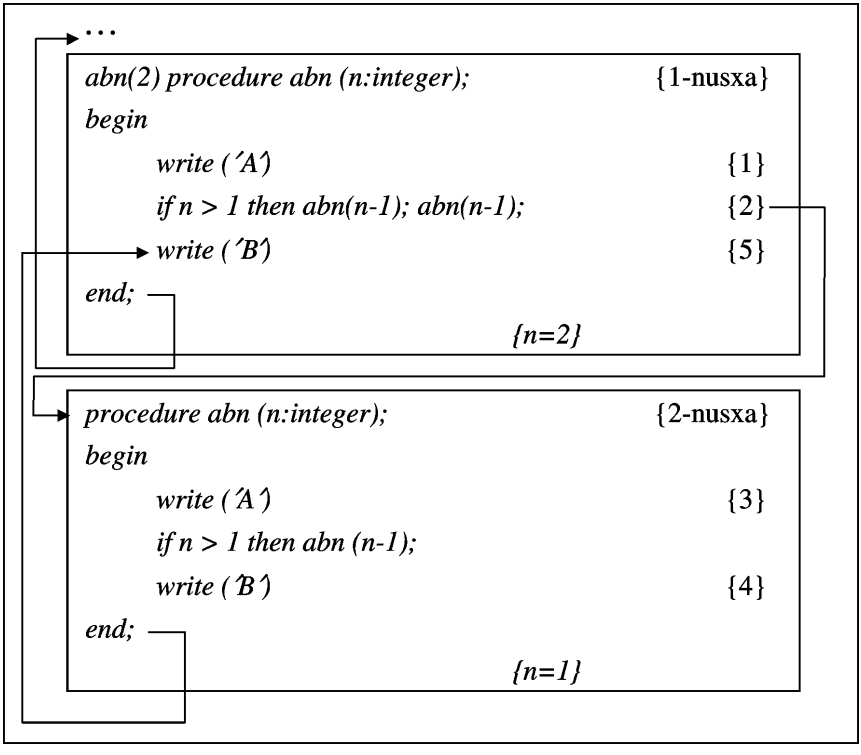
Agar *n=1* bo‘lsa, birinchi va uchinchi operatorlar bajariladi va EHM *AB* ni bosib chiqaradi.

abn (n-1) murojaat (2-operator) bajarilmaydi, chunki n>1 da shart o‘rinli bo‘lmaydi.

n=2 bo‘lganda avval birinchi operator bajariladi (*A* harf bosib chiqariladi), keyin esa *n>1* bo‘lgani uchun, protseduraga qayta murojaat qilish (*abn*) operatori bajariladi.

Bu vaqtda EHM xotirasida o‘zining *n* o‘zgaruvchisi bilan protseduraning ikkinchi nusxasi hosil qilinadi (10.1-rasm). Birinchi nusxaning ikkinchi operatori ketidan ikkinchi nusxaning birinchi operatori *(write (′A′))* bajariladi, *n=1* bo‘lgani uchun *n>1* shart o‘rinli bo‘lmaydi. Demak, keyin B harfni bosib chiqaradigan operator bajariladi. Shu bilan *abn* protseduraning ikkinchi nusxasi o‘z ishini tugatadi. Protsedura birinchi nusxasining bajarilishi ikkinchi nusxasiga murojaat qilingan joydan davom ettiriladi. *B* harfni bosib chiqargandan (5-operator) keyin birinchi nusxaning ishi tamom bo‘ladi. Shunday qilib, dasturning asosiy qismidan *abn* bilan murojaat qilinganda EHM:

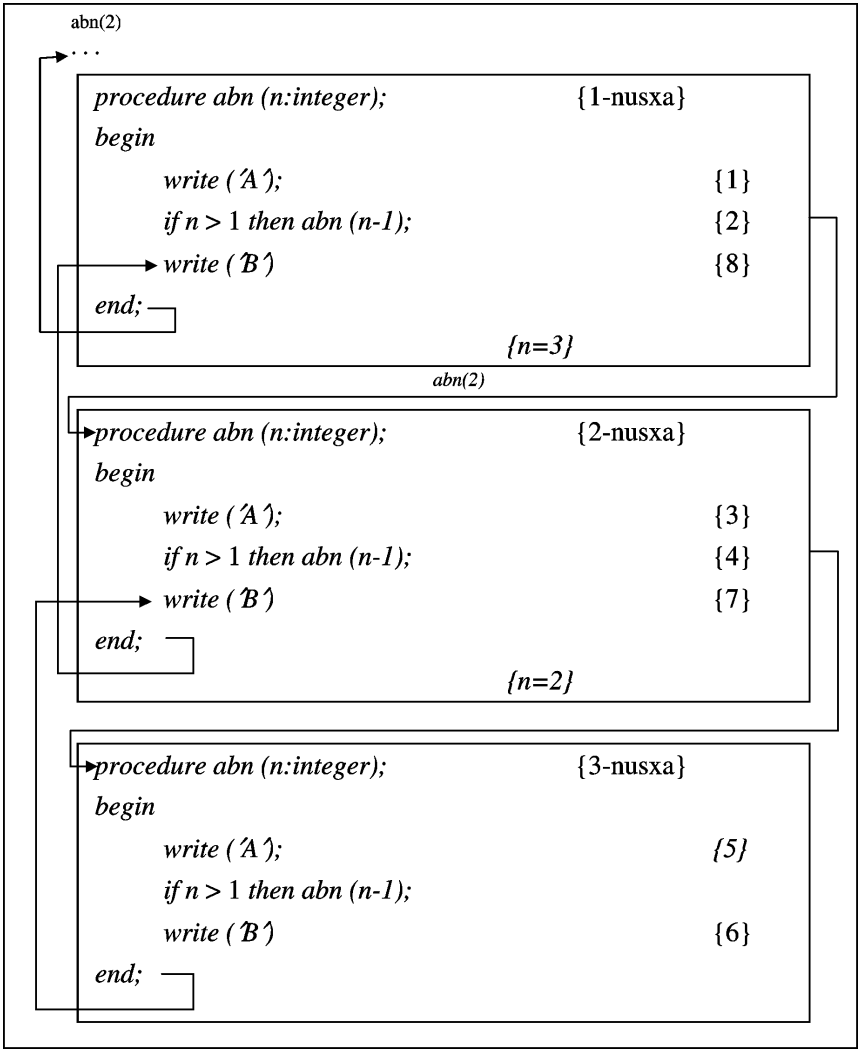
AABBni bosib chiqaradi.



10.1-rasm.

n-3 bo‘lganda mashina xotirasida *abn* protseduraning uchta nusxasi uchun joy ajratiladi. Operatorlarning bajarilish tartibi 10.2-rasmda ko‘rsatilgan, *abn* bilan murojaat qilinganda EHM:

AAABBBni bosib chiqaradi.



10.2-rasm.

Yuqorida ko‘rilgan masalani rekursiv protseduradan foydalanmasdan ham yechish mumkin ekanligini ta’kidlab o‘tamiz.

Yana bir misolni ko‘ramiz:

n sonning faktorialini hisoblovchi rekursiv funksiyani tuzamiz. Sonning faktoriali quyidagi formula bo‘yicha aniqlanadi:

$$n!=\begin{cases} 1 \quad , & agar \;\; n=1 \\ (n-1)!\cdot n; & agar \;\; n>1 \end{cases}$$

10 sonining faktorialini topish kerak bo‘lsin. Agar 9! qiymati ma’lum bo‘lsa, keltirilgan formulaning ikkinchi qismi bo‘yicha 10! qiymatni topish juda oson, 9! qiymat qanday topiladi? 8! qiymat ma’lum bo‘lsa, 9! qiymatni ham xuddi o‘sha formula bilan topish mumkin, 8! qiymat qanday topiladi? Hisoblash jarayoni oxirida 1! qiymat qanday topiladi, savoli tug‘iladi. Bu savolga formulaning birinchi qismi (*n!=1*) agar *n=1* bo‘lsa, javob beradi.

Mulohazamizni dasturlash tilida yozib,

```
function f(p:integer) : integer;
begin
if n=1
        then f:=1
        else f:=n*f(n-1)
end
```

funksiyani olamiz.

Faktorial va unga mos rekursiv funksiyaning o‘ziga xos xususiyati shundaki, *sonning faktorialini aniqlovchi formulada yana o‘sha faktorialni aniqlaydigan formula ishlatiladi, lekin parametrning qiymati bitta kam bo‘ladi*. Aynan bir qiymatning faktorialini hisoblash uchun hisoblash jarayonini, toki parametrning qiymati bir bo‘lgunga qadar qaytadan takrorlash kerak. Agar 1! alohida aniqlanmas ekan (formulaning birinchi qismida yozilgan), natijani olishning imkoni bo‘lmaydi: hisoblash hech qachon tugamaydi.

Navbatdagi misol. *Fibonachchi sonlari*. 1202-yili italyan matematigi Fibonachchi shunday masalani yechdi: «Bir juft quyonlar har oyda ikkitadan (urg‘ochi va erkak) nasl beradi, ular ham ikki oydan keyin nasl beradi. Agar yil boshida bir juft quyon bo‘lsa, yil oxirida quyonlar qancha bo‘ladi?»

Yil boshidan hisoblanganda *n* oydan keyingi quyonlar jufti sonini *F* (*n*) belgisi bilan belgilaymiz. Masala shartiga ko‘ra, yil boshida bir juft quyon bo‘lgan, bir oydan keyin ikki juft quyon bo‘lgan. Demak,

$$F(0)=1; \qquad F(1)=2.$$

Ikki oydan keyin bir juft nasl olamiz, ya’ni yil boshida qancha bo‘lsa, shuncha olamiz. Demak,

$$F(2)=F(0)+F(1).$$

Xuddi shunday mulohaza yuritib, quyidagi bog‘lanishni olamiz:

$$F(n)=F(n-2)+F(n-1).$$

Demak, *n* oydan keyin quyonlar sonini quyidagicha hisoblash mumkin:

$$F(0)=1; \qquad F(1)=2;$$
$$F(n)=F(n-2)+F(n-1).$$

Bu mulohazalarni *Turbo Paskal* tilida yozamiz:

```
function f(n:integer) : integer
begin
        if n=0
              then f:=1
              else
                    if n=1
                          then f:=2
                          else f:=f(n-2) + f(n-1)
end.
```

Bu funksiyani dasturga kiritamiz:

```
program fibon (output);
        var k:integer
        function f(n:integer) : integer;
        begin
              if n=0
                 then f:=1
                 else
                 if n=1
                    then f:=2
                    else f:=f(n-2) + f(n-1)
end;
```

```
begin
        for k:=10 to 12 do
                writeln (f(k))
end.
```

fibon dasturni bajarib, EHM quyidagilarni bosib chiqaradi:

178
288
466

10, 11 va 12 oydan keyin quyonlar soni xuddi shuncha bo‘ladi.

Ba’zi hollarda masalalarni rekursiyasiz yechib bo‘lmaydi. Shunday misol keltiramiz:

0 ga teng bo‘lmagan sonlar ketma-ketligi dastlabki ma’lumotlar bo‘lsin. Berilgan ketma-ketlikning oxirgi elementi 0 ekanligi ma’lum. Dastlabki ma’lumotlarni teskari tartibda bosib chiqarishni ta’minlaydigan dastur tuzish kerak. Masalan, agar dastlabki ma’lumotlar: 3 25 115 400 13 0 sonlar bo‘lsa, mashina 0 13 400 115 25 3 sonlarini bosib chiqarishi kerak.

Rekursiv *teskarilash* protsedurasi ishtirokidagi dasturni yozamiz:

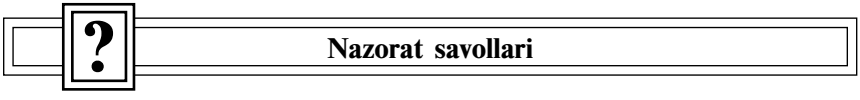
```
program teskari;
        procedure teskarilash;
var     x:integer;
begin
                read (x);
                if x<>0 then teskarilash;
                write (x);
end;
begin
                teskarilash;
end.
```

Bu dastur birgina — *teskarilash* rekursiv protseduraga murojaat qilish operatoridan iborat. *Teskarilash* protsedurasi parametrlarga ega emas. Bu protseduraga murojaat qilish soni berilgan ketma-ketlik uzunligiga bog‘liq. Bir necha hollarni ko‘rib chiqamiz.

Agar berilgan ketma-ketlik 0 ga teng bo‘lgan bitta sondan (bunday hol ham bo‘lishi mumkin) iborat bo‘lsa, protseduraning birinchi operatori *x* ning qiymatini o‘qiydi, ikkinchi operator (protseduraga murojaat qilish) bajarilmaydi, chunki *x=0*, uchinchi operator 0 qiymatni bosib chiqaradi. Shu bilan protseduraning hamda bir vaqtda dasturning ham bajarilishi to‘xtaydi.

Agar ketma-ketlik ikki sondan iborat bo‘lsa ($x_1 \neq 0$, $x_2 = 0$), avval *x* o‘zgaruvchiga nolmas qiymat o‘zlashtiriladi. Shart o‘rinli bo‘ladi, demak, teskarilash protsedurasi qayta bajariladi. Protseduraning ikkinchi nusxasida ikkinchi berilgan son o‘qilib, u *x* o‘zgaruvchiga o‘zlashtiriladi, *x=0* bo‘lgani uchun uchinchi operator 0 sonini bosib chiqaradi. Hisoblash jarayoni birinchi nusxaga qaytadi, u yerda *write (x)* operatori ikkinchi nusxaga murojaat qilish oldidan *x* o‘zgaruvchiga o‘zlashtirilgan birinchi ma’lumotning qiymatini bosib chiqaradi.

Ketma-ketlikning sonlari *n teskarilash* protsedurasining *n* ta nusxasi mos kelishini osongina tushunish mumkin. Bosib chiqarish operatori protsedurada shu protseduraga murojaat qilish operatoridan keyin joylashganligi uchun dastlabki ma’lumotlarni teskari tartibda bosib chiqarishni ta’minlaydi.



Nazorat savollari

1. Protsedura va funksiya bayoni tasnifi qanday ko‘rinishga ega?
2. Protsedura va funksiya bayonlari bir-biridan nimasi bilan farq qiladi?
3. Identifikatorlarning ta’sir doirasi nima?
4. Protsedura va funksiya identifikatorlari uchun ta’sir doirasini aniqlashning asosiy qoidalari nimalar?
5. Qanday parametrlar rasmiy va qaysilari haqiqiy deyiladi?
6. Parametrlar qaysi belgilariga ko‘ra farqlanadi?
7. Parametrlarni uzatishning qanday usullariga nazariy jihatdan yo‘l qo‘yish mumkin?
8. *Turbo Paskal*da parametrlarni uzatishning qanday usullari amalga oshiriladi?
9. Parametr-qiymatlarni uzatishning qoidalari qanday?
10. Parametr-o‘zgaruvchilarni uzatishning qoidalari qanday?
11. Parametr-o‘zgarmaslarni uzatishning qoidalari qanday?
12. Tursiz parametrlarning xususiyatlari nimada?
13. Ochiq parametr-massivlar va tursiz parametrlar o‘rtasidagi farq nimada?
14. *near* va *far* direktivalar nima uchun mo‘ljallangan?
15. *forward* direktivasini ishlatishning xususiyati nimada?
16. Uzish protsedurasini bayon etish uchun qaysi direktiva belgilangan?
17. Rekursiya hodisasi nima?
18. Rekursiya nega dasturlashda qo‘llaniladi?
19. Rekursiv protseduraga misol keltiring.
20. Rekursiv funksiyaga misol keltiring.

XI bob. **FAYLLAR**

11.1. Fizik va mantiqiy fayllar tushunchasi

Fayl tushunchasining ikki tomoni bor. Bir tomondan, fayl bu qandaydir axborotga ega bo‘lgan, tashqi xotiraning nomlangan sohasi. Bunday tushunishda, u qandaydir moddiy axborot tashuvchisida fizik mavjud bo‘lgani uchun *fizik fayl*, deb ataladi. Boshqa tomondan fayl, bu dasturlashda ishlatiladigan ma’lumotlarning ko‘plab tarkiblaridan biridir. Bunday tushunishda, u dastur yozishda faqat bizning mantiqiy tasavvurimizda mavjud bo‘lgani uchun *mantiqiy fayl* deyiladi. Dasturlarda mantiqiy fayllar ma’lum turdagi faylli o‘zgaruvchilar sifatida beriladi.

11.1.1. Fizik fayl tarkibi. Fizik fayl tarkibi axborot tashuvchi (qattiq yoki egiluvchan magnit diski)ning oddiy baytlar ketma-ketligini ifodalaydi:

bayt	bayt	. . .	bayt	bayt

11.1.2. Mantiqiy fayl tarkibi. Faylni dasturda qabul qilish usuli mantiqiy fayl tarkibini bildiradi. O‘xshatish ma’nosida, bu faylning fizik tarkibiga qaraladigan «oyna» («shablon»)dir. Dasturlash tillarida bunday «oyna»larga faylning tashkil etuvchilari sifatida ishlatiladigan ma’lumotlar turlari mos keladi. *Turbo Paskal*ning «shablon»laridan ba’zi birlarini quyidagicha tasavvurda ko‘rsatish mumkin:

file of Byte

bayt	bayt	. . .	bayt	Eof

file of Char

belgi kodi	belgi kodi	. . .	belgi kodi	Eof

File of Integer

Ishorali butun	Ishorali butun	. . .	Ishorali butun	Eof

File of T, bu yerda, T = *record*
a: Byte;
b: Char;
c: Integer;
end.

bayt	belgi kodi	ishorali butun	…	bayt	belgi kodi	belgi kodi	Eof

Faylning mantiqiy tarkibi massiv tarkibiga juda o‘xshaydi. Massiv va fayl o‘rtasidagi farq quyidagidan iborat.

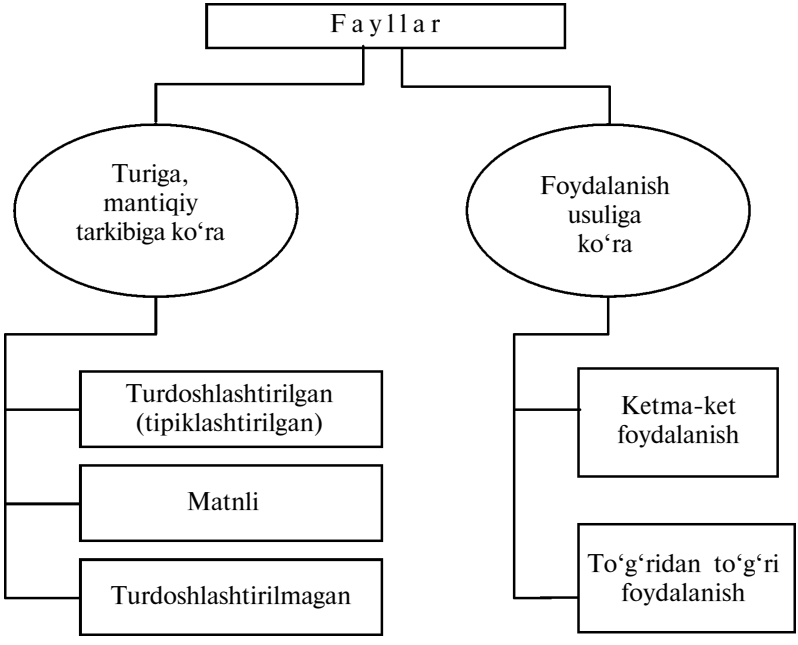
Massivda elementlar soni, xotirani taqsimlash vaqtida o‘rnatiladi va u to‘laligicha tezkor xotirada joylashadi. Massiv elementlarining raqamlanishi, uni e’lon qilishda ko‘rsatilgan quyi va yuqori chegaralarga mos ravishda bajariladi.

Faylda elemenlar soni dastur ishlash jarayonida o‘zgarishi mumkin va u axborotning tashqi tashuvchilarida joylashadi. Fayl elementlari chapdan o‘ngga tomon noldan boshlab (matn fayllaridan boshqa) raqamlanadi. Har bir vaqt daqiqasida fayl elementlari soni noma’lum bo‘ladi. Lekin fayl oxirida uning oxirini bildiruvchi 26 kodlik (CTRL+ Z) ASCII boshqaruvchi belgisi bilan yoziluvchi, maxsus *Eof* yozuvi bo‘ladi. Bundan tashqari, fayl uzunligini

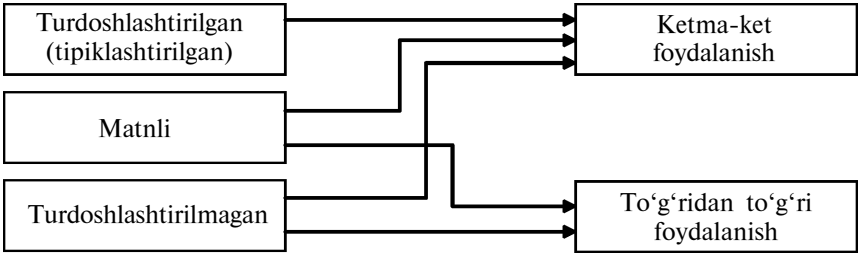
aniqlash va boshqa tez-tez talab qilinadigan ishlarni, fayllar uchun mo‘ljal-langan standart protsedura va funksiyalar yordamida bajarish mumkin.

11.2. *Turbo Paskal*da fayllarning sinflanishi

*Turbo Paskal*da fayllar ikki belgisiga ko‘ra sinflanadi:
— turiga (mantiqiy tarkibiga) ko‘ra;
— fayl elementlaridan foydalanish usuliga ko‘ra.



Har xil fayllarning biron-bir turiga nisbatan foydalanish usullari qo‘llanilish imkoniyatini quyidagi chizma bilan ko‘rsatamiz:



11.3. Fayllarning qo‘llanilishi, ochish va yopish

Qattiq yoki egiluvchan magnit disklarida yozilgan qandaydir fizik fayl bilan ishlash uchun, eng avval, uni faylli o‘zgaruvchi (mantiqiy fayl) bilan bog‘lash zarur. Bu o‘zgaruvchi yordamida mazkur fizik fayldan foydalanish mumkin bo‘ladi. Mantiqiy va fizik fayllarni bog‘lash, faqat yopiq fayl uchun ishlatiladigan, *Assign* protsedurasi yordamida bajariladi. Bu protseduraning birinchi parametri faylli o‘zgaruvchi, ikkinchi parametr esa, qiymati identifikatorlarni MS-DOSda yozish qoidalariga mosligini ko‘rsatuvchi fizik faylning ismi bo‘lgan, satrli o‘zgarmas yoki satrli o‘zgaruvchi identifikatoridir.

Assign(f, 'MyFile.Dat').

Keltirilgan faylda *f* mantiqiy faylni *MyFile.Dat* fizik fayl bilan, u MS-DOS joriy diskning katalogida mavjud degan shartni hisobga olib, bog‘lash bajariladi. Agar *Assign* protsedurasining harakatlari MS-DOS joriy qurilmalariga bog‘liq emasligi talab qilinsa, unda faylning disk, kataloglar yo‘li va fayl nomini ko‘rsatuvchi to‘la ismi yoziladi, masalan:

```
Name:='a:\ MyFiles\ MyFile.Dat';
Assign (f, Name).
```

Fayllardan o‘qish yoki yozish kabi qandaydir bir amallarni bajarish oldidan, bu fayllar ochilgan bo‘lishi kerak.

Fayllarni ochish *Reset* va *Rewrite* protseduralari bilan, yopilishi esa *Close* protsedurasi bilan bajariladi:

```
Reset(f);
Rewrite(f);
Close(f).
```

Reset protsedurasi *f* faylli o‘zgaruvchi bilan bog‘liq bo‘lgan mavjud fizik faylni ochadi. Agar *f*-matnli fayl bo‘lsa, uni faqat ketma-ket foydalanish usuli bilan o‘qib bo‘ladi, agar *f*-turdoshlashtirilgan fayl bo‘lsa, u ketma-ket va to‘g‘ridan to‘g‘ri foydalanish usullari bilan o‘qish uchun ham, yozish uchun ham ochiq bo‘ladi.

Ochilishda faylning joriy xonasi ko‘rsatkichi uning boshiga o‘rnatiladi.

Agar ko‘rsatilgan ismli fizik fayl bo‘lmasa, bajarish vaqti xatosi paydo bo‘ladi, uni kompilatorning *{$I-}* direktivasini o‘chirish bilan yo‘qotish mumkin. Direktivaning bunday ko‘rsatmasida faylni ochish ishini tugatish natijasini *IOResult* funksiyasi yordamida tahlil qilish mumkin. Funksiya, agar ish muvaffaqiyatli bajarilsa, 0 qiymatni va, aks holda, xatolikning nolmas qiymatini beradi.

Rewrite protsedurasi, ismi *f* faylli o‘zgaruvchi bilan bog‘liq bo‘lgan, yangi fizik fayl yaratadi. Agar shunday fayl mavjud bo‘lsa, u yo‘qotiladi va uning o‘rnida yangi bo‘sh fayl yaratiladi. Ochishda faylda joriy xonaning ko‘rsatkichi uning boshiga o‘rnatiladi.

Amalda hamma dasturlarda ishlatiluvchi yana bir funksiya—*Eof* funksiyasidir:

Eof(f).

Agar *f* faylda joriy xona ko‘rsatkichi faylning oxirgi elementidan keyin turgan bo‘lsa yoki fayl bo‘sh bo‘lsa, *Eof(f)* funksiya *True* qiymatini, aks holda, u *False* qiymatini qaytaradi.

Misol sifatida *MyFile.Dat* fayli elementlari yig‘indisini hisoblashni ko‘ramiz:

```
program Fsumma;
Uses Crt;
var
        f      : file of Integer;
        X      : Integer;
        Summa: Longint;
begin
        ClrScr;
        {$I-};
        Assign (f, `MyFile.Dat`);
        Reset (f);
        {I+}
        if IOResult<>0 then
              begin
                    writeln (`Fayl ochilishida xatolik`);
                    Halt (1);
```

```
        end;
                {Yig‘indini hisoblash}
    Summa:=0;
    while not Eof(f) do
    begin
        read (f, X);
        Summa:= Summa +X;
    end;
    writeln (`Fayl elementlari jami`, Summa,`ga teng`);
    close (f);
end.
```

11.4. Fayllar bilan ishlashning umumiy vositalari

11.4.1. System moduli protsedura va funksiyalari. System modulining protsedura va funksiyalarini hamma vaqt ham chaqirib bo‘ladi va bu ishni bajarishda qo‘shimcha modullar talab qilinmasligini eslatamiz. Yuqorida ko‘rib o‘tilgan protsedura va funksiyalardan tashqari, bu modul fayllar bilan ishlash uchun qayta nomlash, fayllarni yo‘qotish va kataloglar bilan ishlash protseduralariga ham ega.

Kataloglar bilan ishlash protseduralari.

System modulida kataloglar bilan ishlash uchun ChDir, MkDir, RmDir va GetDir protseduralar kiritilgan. Mazmuniga ko‘ra, ular MS-DOSning shunday buyruqlariga o‘xshash. Bu protseduralarning ishlatilishini quyidagi misolda ko‘ramiz:

```
program Direktories Demo;
Uses Crt;
var
    S: String;
begin
    ClsScr; {E: diskning o‘zak katalogini joriy qilib
            o‘rnatish:}
    ChDir (‘E:\’);
            {Joriy katalog va diskni ko‘rsatish}
```

```
        GetDir (0,S);
        Writeln (`Joriy disk va katalog:`,S);
                {MyDir ichki katalog yaratish}
        MkDir (`MyDir`);
                {MyDir ichki katalogiga o‘tish}
        ChDir (`MyDir`)
                {Joriy disk va katalogni ko‘rsatish}
        GetDir(0,S);
        Writeln(`Joriy disk va katalog:`,S);
              {E: diskning o‘zak katalogini joriy qilib o‘rnatish}
        ChDir (`\`);
                {MyDir ichki katalogini yo‘qotish}
        RmDir(`MyDir`);
                {Joriy disk va katalogni ko‘rsatish}
        GetDir(0,S);
        Writeln(`Joriy disk va katalog:`,S);
end.
```

Fayllarni qayta nomlash va o□chirish protseduralari

Rename fizik fayllarni qayta nomlashga, *Erase* protsedurasi esa ularni o‘chirishga xizmat qiladi. Bu protseduralar, qandaydir bir fizik fayl bilan bog‘langan, lekin ular uchun faylni ochish ishi hali bajarilmagan faylli o‘zgaruvchilar uchun qo‘llaniladi. Quyidagi dastur avval *MyFile.Dat* faylini *Result*ga qayta nomlash, keyin esa uni o‘chirishni namoyish etadi.

```
program  RenDelDemo;
var
        f:file;
begin
        Assign(f,`MyFile.Dat`);
        Rename (f,`Result.Dat`);
              {Qayta nomlangan fayllar bilan ishlash}
        Erase(f)
end.
```

11.4.2. DOS modulining o‘zgarmaslari, turlari, protsedura va funksiyalari. DOS modulining faylli vositalaridan foydalanuvchi dasturlarda DOS identifikatorini ko‘rsatuvchi *Uses* taklifi bo‘lishi kerak:

```
programm Example;
Uses DOS
                ...
begin
                ...
end.
```

Fayllar bilan ishlash uchun turlar

```
TfileRec= record
        Handle :Word;
        Mode : Word;
        RecSize : Word;
        Private :array [1..26] of Byte;
        UsesData:array[1..16] of Byte;
        Name :array [0..79] of Char;
End;
```

```
PTextBuf=^ TTextBuf;
TTextBuf=array[0..127] of Char;
TTextRec=record
        Handle:Word;
        Mode:Word;
        BufSize:Word;
        Private:Word;
        BufPos:Word;
        BufEnd:Word;
        BufPtr: PTextBuf;
        OpenFunc:Pointer;
        InOutFunc:Pointer;
        FlushFunc:Pointer;
```

```
        CloseFunc:Pointer;
        UserData:array[1..16] of Byte;
        Name:array[0..79] of Char;
        Buffer:TTextBuf;
End;
```

TFileRec turi turdoshlashtirilgan fayllar uchun ham, turdoshlashtirilmagan fayllar uchun ham ishlatiladi. *TtextRec* tur esa faqat matnlarda qo‘llaniladi.

```
SearchRec=record
        Fill:array[1..21] of Byte;
        Attr: Byte;
        Time: Longint;
        Size: Longint;
        Name: String[12];
end;
```

SearchRec turdagi o‘zgaruvchilar *Find First* va *Find Next* protseduralarda fayllar kataloglarini ko‘rib chiqish uchun ishlatiladi.

```
DateTime=record
        Year, Month, Day, Hoyr,
        Min, Sec : word;
end;
```

DateTime turi *UnpackTime* va *PackTime* protseduralarida sana va vaqtga ega bo‘lgan to‘rt baytli qiymatni tahlil qilish, joylashtirish va yaratish uchun ishlatiladi. Bu to‘rt baytli qiymat shundan keyin *GetFTime, SetTime, FindFirst* va *FindNext* protseduralarida ishlatiladi.

DOS modulining *ComStr, PathStr, DirStr, NameStr, ExtStr* satrli turlari *FSplit* satrli protsedurani chaqirishda fayllar ismlari va yo‘nalishlari (manzillari) bilan ishlashda foydalaniladi.

Fayllar bilan ishlash uchun o□zgarmaslar:

```
FmClosed=$D7BO
FmInput=$D7B1
FmOutput=$D7B2
FmInOut=$D7B3
```

FmClosed, fmInput, fmOuput, fmInOut o‘zgarmaslar *TTextRec* yoki *TFileRec* turidagi o‘zgaruvchilarda *Mode* maydonining qiymatini (yozish-o‘qish rejimi) o‘rnatish maqsadida fayllarni ochish va yechishda foydalaniladi.

```
ReadOnly = $01 {faqat o‘qish uchun}
Hidden = $02 {«yashiringan» fayl}
SysFile = $04 {tizim fayli}
VolumeID = $08 {tom ismi}
Directory = $10 {katalog}
Archive = $20 {arxiv fayli}
AnyFile = $3F {ixtiyoriy fayl}
```

ReadOnly, Hidden, SysFile,VolumeID, Directory, Archive, AnyFile o‘zgarmaslar fayllar atributlarini o‘rnatishda ishlatiladi.

Fayllar bilan ishlashning protsedura va funksiyalari

GetFTime — faylning oxirgi yozilish sanasi va vaqtini qaytaradi.
SetFTime — faylning oxirgi yozilish sanasi va vaqtini o‘rnatadi.
PackTime — *Date Time* yozuvini *SetFTime* protsedurasi ishlatadigan *Longint* turidagi sana va vaqtni ifodalashning ichki kodiga aylantiradi.
UnpackTime — sana va vaqtni *Longint* turida ifodalashning *GetF Time, FindFirst, FindNext* protseduralar bilan qaytariladigan ichki kodini *DateTime* joylashtirilgan yozuvga aylantiradi.
FExpand — fayl ismini qabul qiladi va to‘la aniqlangan ism (disk, katalog, kengaytma)ni qaytaradi.

FSearch — faylni kataloglar ro‘yxatidan izlaydi.
FindFirst — tarkibi faylning berilgan ismi va atributlari bilan mos keluvchi berilgan (yoki joriy) katalogda yozuvni izlashni amalga oshiradi.
FindNext — *Find First* protsedurasiga bundan oldin murojaat qilinganda berilganlar bilan fayl ismi va atributlari mos keluvchi navbatdagi yozuvni izlaydi.
GetFAttr — fayl atributlarini qaytaradi.
SetFAttr — fayl atributlarini o‘rnatadi.

Bundan tashqari, fayllar bilan ishlashda (ayniqsa, yangi fayllar yaratishda) DOS modulining yana ikki funksiyasi foydalidir.

DiskFree — berilgan diskovodda diskdagi bo‘sh baytlar sonini qaytaradi.
DiskSize — berilgan diskning baytlardagi to‘liq hajmini qaytaradi.

11.5. Turdoshlashtirilgan fayllar

Turdoshlashtirilgan faylning hamma elementlari bir xil turda bo‘lishi kerak. Turdoshlashtirilgan fayllar faylli va faylga tayanuvchi turdan boshqa ixtiyoriy turda bo‘lishi mumkin.

Mumkin bo‘lmagan e’lonlarga misol:

```
type   TF1 = file of file;
       TFR = record
             A: Integer;
             F: file of Real;
end;
TF2:file of TFR.
```

Turdoshlashtirilgan fayllarda ketma-ket foydalanish usuli ham, to‘g‘ridan to‘g‘ri foydalanish usuli ham yo‘l qo‘yiladi. To‘g‘ridan to‘g‘ri ishlash usulida turdoshlashtirilgan fayllarning elementlari hamma vaqt noldan boshlab raqamlanishini eslab qolish kerak.

11.5.1. Turdoshlashtirilgan fayllar bilan ishlash protsedura va funksiyalari. Turdoshlashtirilgan fayllardan o‘qish faqat *Read,* yozish esa faqat *Write* protsedurasi bilan amalga oshiriladi. Bunda o‘qish (yozish) birligi hamma vaqt fayl turi bilan bir xil turdagi o‘zgaruvchi bo‘ladi.

Read protsedurasi turdoshlashtirilgan fayllar uchun quyidagi formatga ega:

Read (faylli o‘zgaruvchi ismi, o‘zgaruvchilar ro‘yxati).

Read protsedurasi ro‘yxatidagi har bir o‘zgaruvchiga o‘qishda, fayldagi joriy xona ko‘rsatkichi navbatdagi elementga o‘tadi. Agar fayldagi joriy xona ko‘rsatkichi oxirgi elementdan keyin, ya’ni *(Eof(f)=True)* fayli oxirida turgan bo‘lsa, unda *Read* protsedurasining bajarilishi bajarilish vaqti xatoligiga olib keladi.

Write protsedurasi turdoshlashtirilgan fayllar uchun quyidagi formatga ega:

Write (faylli o‘zgaruvchi ismi, o‘zgaruvchilar ro‘yxati).

Har bir o‘zgaruvchini yozishda (o‘qishda ham) fayldagi joriy xona ko‘rsatkichi keyingi elementga o‘tadi. Agar fayl joriy xonasining ko‘rsatkichi oxirgi elementdan keyin, ya’ni *(Eof(f)=True)* fayli oxirida bo‘lsa, *Write* protsedurasini bajarishda fayl kengayadi.

To‘g‘ridan to‘g‘ri foydalanish uchun quyidagi protseduralar belgilangan:

FilePos — faylda joriy xona ko‘rsatkichi raqamini qaytaradi (xonalar noldan raqamlanadi).

FileSize — faylning joriy o‘lchamini qaytaradi (fayl elementlari soni birdan boshlab hisoblanadi).

Seek — fayldagi joriy xona ko‘rsatkichini berilgan raqamli (noldan boshlab hisoblanganda) elementga o‘tkazadi.

Truncate — fayl o‘lchamini joriy xonagacha qirqadi. Fayldagi joriy xonadan keyin joylashgan hamma elemenlar yo‘qotiladi va fayldagi joriy xona uning oxiri (*Eof(f)=True*) bo‘ladi.

Turdoshlashtirilgan fayl elementlariga to‘g‘ridan to‘g‘ri foydalanishga misol sifatida almashtirish («ko‘pik») usuli bilan faylni saralash dasturini ko‘ramiz (IX bob):

```
program  FileSort;
UsesCrt;
var
        f:file of integer;
        x,y: integer;
        i,j; Longint;
begin
        {$I-}
        Assign(f, `Sort.Dat`);
        Reset (f);
        {I+}
        if IOResult<>0 then
        begin
                writeln(`Faylni ochishda xatolik`);
                Halt (1);
            end;
            ClrScr;
            Writeln (`Boshlang‘ich fayl`);
            for i:=1 to FileSize (f) do
            begin
                Read (f,x);
                Write (x:8)
            end;
            Writeln;
            Close(f);
{-----------------------------------------------}
            Reset (f)
            for i:=FileSize (f)-1 downto 1 do
            {Navbatdagi maksimal elementning i-xonaga
            «qalqib» chiqishi}
            for j:=0 to i-1 do
            begin
                Seek (f,j);
                Read (f,x,y);
                if x>y then
                begin
```

```
                    Seek (f,j);
                    Write (f,y,x);
                end;
    end;
    Close(f);
    {-------------------------------------------}
      Reset (f);
      Writeln (`Saralangan fayl:`);
     for i:=1 to FileSize (f) do
      begin
            Read (f,x);
            Write (x:8)
      end;
      Close (f);
end.
```

Turdoshlashtirilgan faylning turi «oyna» («shablon») ekanligini eslatamiz, uni elementlaridan foydalanish mumkin bo‘lishi uchun fizik faylga «qo‘yamiz». Agar shunday mumkin bo‘lsa, bitta «oyna»dan foydalanib fayl yaratish, boshqa «oyna»ni ishlatib, uni o‘qish mumkinmi, savoli tug‘iladi. Haqiqatda bunday qilish mumkin. Masalan, quyidagi misolda *Char* turidagi faylga avval bir qator belgilar yoziladi, keyin esa shu fizik faylning o‘zi *Byte* turidagi fayl kabi ochiladi va natijada unga yozilgan belgilarning ASCII kodlari bosmaga chiqariladi:

```
program CharToByte;
Uses Crt;
var
        FC: file of Char;
        FB: file of Byte;
        B : Byte;
begin
        ClrScr;
        Assign (FC,`Test.Dat`);
```

```
        Rewrite(FC);
        for Ch:='0' to '9' do write (FC, Ch);
        for Ch:='A' to 'J' do write (FC, Ch);
        Close(FC);
        Assign(FB, 'Test.Dat');
        Reset(FB);
        While not Eof (FB) do
        begin
              Read (FB,B);
              Write (B:8);
        end;
        Close (FB)
end.
```

Natija

48	49	50	51	52	53	54	55	56	57
65	66	67	68	69	70	71	72	73	74

11.6. Matnli fayllar

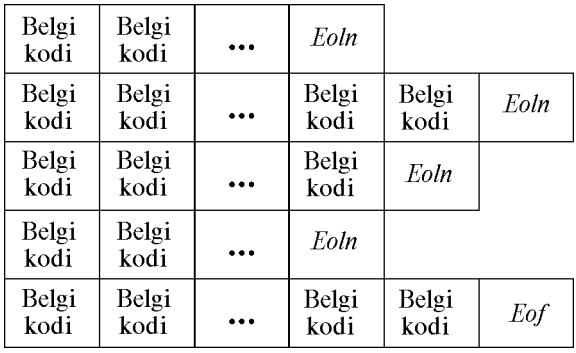
11.6.1. Matnli fayl tarkibi. Matnli fayllar *Turbo Paskal*da *file of Char* turidagi fayllar turlaridan biridir.

Matnli fayllarni bayon etish uchun *Text* turi ishlatiladi:

```
var
    TextFile: Text.
```

Matnli fayllarda fayl oxirini ko‘rsatuvchi *Eof* belgi bilan bir qatorda, satr oxirini ko‘rsatuvchi *Eoln* belgisi ham ishlatiladi. *Eoln* belgi ASCII kodining ikkita belgisi (13 kodi «karetkani qaytarish» va 10 kod—«satrni o‘tkazish») ketma-ketligini ifodalaydi.

Matnli faylni, har bir qatorining oxirida *Eoln* turgan, kitob betiga o‘xshatish mumkin:



Kiritish-chiqarish *Input* (klaviaturadan kiritish) va *Output* (displeyga chiqarish) standart fayllari matnli fayllardir. *Read, Readln, Write, Writeln* protseduralar matnli fayllarda standart kiritish-chiqarishda ham odatdagiday ishlatiladi. Farq faqat shundaki, bu protseduralarning birinchi parametri bilan faylli o‘zgaruvchi ko‘rsatilishi kerak.

```
Read (f, A, B);
Write (g,'A=', A, 'B=', B);
Readln (f, C, D);
Writeln (g, 'C=',C, 'D=', D);
```

11.6.2. Matnli fayllar bilan ishlash protsedura va funksiyalari. Umumiyga qo‘shimcha qilib matnli fayllar uchun quyidagi protsedura va funksiyalar ishlatiladi:

Append — mavjud faylni, fayl oxiriga elementlarni qo‘shish uchun, ochadi.

Flush — matnli fayl uchun chiqarish buferini tashlaydi.

Readln — xuddi *Read* kabi ishlaydi, yana qo‘shimcha tarzda joriy satrning hamma qolgan elementlarini o‘tkazib yuboradi va faylning joriy xonasi ko‘rsatkichini matnli faylning keyingi satriga o‘tkazadi.

SeekEof — matnli fayl uchun *Eof* (fayl oxiri) holatini qaytaradi.

SeekEoln — matnli fayl uchun *Eoln* (satr oxiri) holatini qaytaradi.

SetTextBuf — matnli fayl uchun kiritish-chiqarish buferini belgilaydi.

Writeln — xuddi *Write* kabi ishlaydi, lekin protsedurada ko‘rsatilgan qiymatlarni yozgandan keyin matnli faylga qo‘shimcha qilib satr oxiri—*Eoln* belgisini yozadi.

11.7. Turdoshlashtirilmagan fayllar

Turdoshlashtirilmagan faylni e’lon qilishda faqat *file* kalit so‘zi ko‘rsatiladi, masalan,

var F: file

Turdoshlashtirilmagan faylli o‘zgaruvchilar fayllar bilan quyi pog‘onadagi ishlarni bajarishga mo‘ljallangan. Ular yordamida ixtiyoriy turdagi faylga va mantiqiy tasnifga, *Byte* turidagi faylli o‘zgaruvchi vositasida ramzli faylga qanday murojaat qilinsa, xuddi shunday murojaat qilish mumkin. Farq faqat shundaki, turdoshlashtirilmagan fayl turdoshlashtirilgan fayl kabi qat’iy o‘rnatilgan o‘qish (yozish) birligiga ega emas.

Turdoshlashtirilmagan fayllarda bitta murojaatda taxminan kiritish (chiqarish) buferi kattaligiga teng keladigan bayt miqdori o‘qiladi (yoziladi), bu esa fayllar bilan ishlash tezligini oshiradi. Turdoshlashtirilmagan fayllarda kiritish (chiqarish) buferi sifatida ixtiyoriy o‘zgaruvchi ishlatilishi mumkin.

11.7.1. Turdoshlashtirilmagan fayllar bilan ishlash uchun protsedura va funksiyalar. Turdoshlashtirilgan fayllarda ishlatiladigan barcha protsedura va funksiyalar turdoshlashtirilmagan fayllarda ham ishlatiladi. Faqat bunda *Read* va *Write* protseduralari o‘rnida *BlockRead* va *BlockWrite* protseduralari ishlatiladi, *Reset* va *Rewrite* protseduralar ma’lumotlarni uzatishda ishlatiladigan yozuv o‘lchamini aniqlovchi ikkinchi *Word* turidagi parametrga ega bo‘ladi. Agar bu parametr tushirib qoldirilgan bo‘lsa, yozuv o‘lchami sukut holatida 128 baytga teng, deb qabul qilinadi.

Nazorat savollari

1. Fayl deb nimaga aytiladi?
2. Fizik va mantiqiy fayllar tasniflari o‘rtasida nima farq bor?
3. Massiv va fayl o‘rtasida qanday o‘xshashlik va farq bor?
4. *Turbo Paskal*da fayllar qanday belgilariga ko‘ra sinflanadi?
5. Faylni ochish uchun qanday ish bajarish kerak?
6. Faylni ochish uchun qanday protseduralar belgilangan va ular qanday ishlaydi?
7. Qaysi protseduralar kataloglar bilan ishlaydi?
8. *Close* protsedurasi nima uchun kerak?
9. *Rename* va *Erase* protseduralar qanday ishlarni bajaradi?
10. Turdoshlashtirilgan fayllarni qanday turlarda bayon etish mumkin?
11. Turdoshlashtirilgan fayllarning elementlari qanday raqamlanadi?
12. Turdoshlashtirilgan fayllardan o‘qish qaysi qoidalar bo‘yicha bajariladi?
13. *Write* protsedura turdoshlashtirilgan fayllar uchun qanday formatga ega?
14. Turdoshlashtirilgan fayl elementlaridan to‘g‘ridan to‘g‘ri foydalanish uchun qanday protsedura va funksiyalar bor?
15. Foydalanuvchi matn fayllarining *Input* va *Output* fayllaridan farqi nimada?
16. Matn fayllarining xususiyatlari nimada?
17. Matn faylining *File of Char*dan farqi nimada?
18. Turdoshlashtirilmagan fayllar qanday e’lon qilinadi?
19. Turdoshlashtirilmagan fayllarning turdoshlashtirilgan fayllardan farqi nimada?
20. Turdoshlashtirilmagan fayllarda qanday protsedura va funksiyalar ishlatiladi?

XII bob. ***TURBO PASKAL*NING GRAFIK REJIMI**

12.1. Grafik rejimdagi ekran

Kompyuter ekrani katta sondagi nuqtalardan iborat bo‘lgan to‘rtburchakli maydonni ifodalaydi. Ekran matn yoki grafik rejimda bo‘ladi. Matn rejimdan farq qilib, grafik rejimda har bir nuqta o‘z rangini o‘zgartirishi, har xil ranglarga bo‘yalgan nuqtalar chiziqlar, matnlar va boshqa tasvirlarni tashkil qilishi mumkin. Nuqtalar rangi hech bo‘lmaganda ikkita — oq va qora bo‘lishi mumkin.

Ekranning bir qismi grafik, ikkinchi qismi esa matn rejimida bo‘lgan holat yo‘q (balki kelajakda bo‘lishi mumkindir).

Gap shundaki, ekranda ko‘rinayotganlarning hammasi videoxotira tarkibining avtomatik tarzda tasvirlanishidir.

Joriy daqiqada matn yoki grafik rejim ishlatilayotganligiga bog‘liq ravishda videoxotiraning tarkibi butunlay farqli holda talqin qilinadi.

Videoxotira, nurli trubka kontrollyori, kiritish-chiqarish portlari va h.k.lar displey «adapteri» deb ataluvchi bitta pechat platasida joylashgan. Displey adapterlarining bir necha turlari bor. Ular o‘rtasidagi farqlar, xususan, quyidagi hollar bilan bog‘liq:

- Ekranning displey yuzasidagi nuqtalar turini quvvatlay olishi mumkin bo‘lgan qobiliyatiga;
- Ekranda bir vaqtda ko‘rsatilishi mumkin bo‘lgan ranglarning maksimal soniga.

Eng keng ishlatiladigan adapterlar quyidagilar:

- *CGA (Color Graphics Adapter);*
- *MCGA (Multi-Color Graphics Adapter);*
- *EGA (Enhanced Graphics Adapter);*
- *VGA (Video Graphics Array).*

Kompyuterga qanday adapter o‘rnatilishidan qat’i nazar, *Turbo Paskal*ning bir xil protsedura va funksiyalari to‘plamidan foydalanish mumkin, chunki ularning ma’lum adapterga sozlanishi avtomatik ravishda amalga oshiriladi. Bu sozlashni *grafik drayverlar* deb ataluvchi maxsus dasturlar bajaradi.

Drayverlar *BGI (Borland Graphics Interface)* kengaytmalariga ega bo‘lgan fayllarda joylashgan bo‘ladi. Masalan, *EGA* va *VGA* adapterlari bilan ishlovchi drayverlar *EGA, VGA, BGI* fayllarida, *CGA* va *MCGA* adapterlari bilan ishlovchilari esa *CGA* va *BGI* fayllarida bo‘ladi.

Har xil fayllarni ko‘rsatish uchun *Graph* modulida o‘zgarmaslar aniqlangan:

```
const

Detect = 0; {drayverni avtomatik ravishda aniqlash}
CGA = 1;
MCGA = 2;
EGA = 3;
EGA64 = 4;
EGAMono = 5;
IBM8514 = 6;
HercMono = 7;
ATT400 = 8;
VGA-9;
RS327=10;
```

Bunday turdagi adapter ko‘pincha bir necha grafik rejimlarda ishlay oladi. Bu rejimlarni *GRAPH* modulida ko‘rsatish uchun o‘zgarmaslar aniqlangan. Quyida *CGA*, *EGA* va *VGA* adapterlari uchun o‘zgarmaslar ro‘yxati keltiriladi:

```
const

CGACO = 0 {320 * 200, 1 − palitra, 4 xil rang}
CG AC1 = 1 {320 * 200, 1 − palitra, 4 xil rang}
CGAC2 = 2 {320 * 200, 1 − palitra, 4 xil  rang}
CGAC3 = 3 {320 * 200, 1 − palitra, 4 xil  rang}
CGAHi = 4 {640 * 200, 1 − palitra, 4 xil  rang}
EGALo = 0 {640 * 200, 4 sahifa, 16 xil  rang}
EGAHi = 1 {640 * 350, 2 sahifa, 16  xil  rang}
EGA64Lo = 0 {640 * 200, 1 sahifa, 16 xil  rang}
EGA64Hi = 1 {640 * 350, 4 sahifa, 4 xil rang}
```

VGALo = 0 {640 * 200, 4 sahifa, 16 xil rang}
VGAMed = 1 {640 * 350, 2 sahifa, 16 xil rang}
VGAHi = 2 {640 * 480, 1 sahifa, 16 xil rang}

Afsuski, *Turbo Paskal VGA* adapteri 320*200 nuqtali, 256 rangli rejimda ishlay olmaydi, ammo Borland firmasining alohida *«BGI TOOLKIT»* dastur mahsulotiga tegishli *VGA*256.*BGI* drayveri bor.

12.2. Ekranni grafik rejimga o‘tkazish

Matn rejimi ekran uchun odatdagi rejim bo‘lib hisoblanadi. Ekranni grafik rejimiga o‘tkazish uchun *Graph* modulining *InitGraph* protsedurasi ishlatiladi.

InitGraph (GD, GM, Path) — ekranni grafik rejimga o‘tkazadi. *GD* drayver raqami, *GM* — rejim raqami, *Path* — kerakli drayverga ega bo‘lgan fayl manzili. Agar *Path* o‘zgaruvchi bo‘sh *(Path=")* satrga ega bo‘lsa, unda drayver joriy katalogdan izlanadi.

GD va *GM* o‘zgaruvchi parametrlar bo‘lib hisoblanadi. Agar *InitGraph* ishga tushirilganda GD o‘zgaruvchi nolga teng bo‘lsa, unda kerakli drayver va bu drayver uchun optimal grafik rejim avtomatik ravishda aniqlanadi. Ko‘rgazmalilik uchun hatto *GRAPH* modulida 0 ga teng bo‘lgan *Detect* o‘zgarmasi kiritilgan.

InitGraph protsedurasiga simmetrik yondashgan *CloseGraph* protsedura bor, u drayverni xotiradan yuklaydi va boshlang‘ich videorejimni tiklaydi.

Quyidagi dastur grafik rejimni initsiallaydi va uni shu zahotiyoq yopadi.

```
uses    Graph;
var     GDriver, GMode: Integer;
begin
  Gdriver: = Detect; {drayverni avtomatik ravishda aniqlash,
                      chunki Detect = 0}
  lnitGraph (GDriver,GMode,'d:\tp'); {grafik rejimni initsiallash}
  ReadLn; {'Enter' klavishini bosishni kutish}
  CloseGraph; {CloseGraph protsedura xotiradan grafik drayverni
               yuklaydi va uning dastlabki videorejimini tiklaydi.}
end.
```

Umuman olganda, grafikli dasturning shabloni quyidagi ko‘rinishga ega:

```
uses Graph;
var GDriver, Gmode, ErrCode: Integer;
begin
Gdriver:=Detect; {drayverni avtomatik ravishda aniqlash, chunki
                  Detect=0}
InitGraph (GDriver,GMode, ' '); {grafik rejimni initsiallash, ' '
                  ichida kerakli drayverga ega bo‘lgan fayl manzili ko‘rsatiladi, agar ' ' ichi bo‘sh satrga ega bo‘lsa, unda drayver joriy katalogdan izlanadi.}

ErrCode:=GraphResult;
If ErrCode=grOk then
begin
                  {Bu yerga dasturning asosiy ko‘rsatmalari yoziladi};
end;
ReadLn; {'Enter' klavishini bosishni kutish}
CloseGraph; {CloseGraph protsedura xotiradan grafik drayverni
                  yuklaydi va uning dastlabki videorejimini tiklaydi.}
end.
```

Ba’zan shunday hollar bo‘lishi mumkinki, unda *InitGraph* protsedurasi o‘zining ishini normal bajara olmaydi, masalan, agar kerakli *BGI* fayl joylashgan manzil xato ko‘rsatilgan bo‘lsa yoki grafikni initsiallash uchun tezkor xotira sig‘imi yetmasa.

Bunday hollarda identifikatsiya uchun, grafikni initsiallashga urinib ko‘rishdan keyin *GraphResult* funksiyasidan foydalanish mumkin. Ammo *GraphResult*ning qo‘llanilishi keng, bu funksiya boshqa ko‘plab grafik amallarda ham xato kodini qaytaradi. Quyidagi o‘zgarmaslar *GraphResult* qaytarish kodlariga mos keladi.

```
Const {GraphResult funksiyaning qaytarish kodlari}
grOk = 0; {xatolar yo‘q}
grNoInitGraph=-1; {(BGI) grafika o‘rnatilmagan (GraphResult
                  protseduradan foydalanish kerak)}
grNotDetected = -2; {grafika uchun apparat ta’minoti topilmagan}
grFileNotFound =-3; {qurilma drayveri fayli topilmagan}
grInvalidDriver =-4; {qurilma drayveri fayli xato }
grNoLoadMem =-5; {drayverni yuklash uchun xotira yetmaydi}
grNoScanMem =-6; {sohani skanerlashda xotira chegaralaridan
                  chiqish}
```

grNoFloodMem =-7; {bo‘yaladigan sohani to‘ldirishda xotira chegarasidan chiqish}
grFontNotFound =-8; {shrift fayli topilmagan}
grNoFontMem =-9; {shriftni yuklash uchun xotira yetmaydi}
grInvalidMode=-10; {tanlangan drayver uchun yo‘l qo‘yib bo‘lmaydigan grafik rejim}

Quyidagi dastur xatolik mavjud-mavjudmasligini aniqlaydi:

```
uses  Graph;
var  GDriver, GMode, ErrCode: Integer;
begin
  Gdriver:= Detect; {drayverni avtomatik ravishda aniqlash}
  InitGraph(G Driver,GMode,''); {grafik rejimni initsiallash}
  ErrCode:=GraphResult;
  if ErrCode<>() then {xatolar bor}
  Writeln ('Grafik raqamini initsiallashda xatolik', Err Code)
  else Writeln ('xatolar yo‘q.');
  CloseGraph; {grafik rejimni o‘chirish}
end.
```

12.3. Grafik rejimda nuqtalar, ranglar, chiziqlar, shakllar

Graph modulida 80 ga yaqin protsedura va funksiyalar mavjud. Ular yordamida nuqtalar, kesmalar, ellipslar, to‘rtburchaklarni har xil ranglar bilan bo‘yash, 11-standart va boshqa xohlagancha standart bo‘lmagan usullar bilan shtrixlab chizish, matnlarni har xil ranglar bilan chiqarish, ekran sohalarini eslab qolib, surish mumkin.

12.3.1.Nuqtalar, ranglar. Ekranning har bir nuqtasi o‘zining koordinatalariga ega. Yuqori chap burchak koordinatasi (0,0). *x* koordinatasi chapdan o‘ngga tomon, *y* koordinatasi esa yuqoridan pastga tomon o‘zgaradi. Masalan, *VGAHi* (640*480) rejimda quyi o‘ng burchak koordinatalari (639*479). Ekran o‘rtasining koordinatalari esa (320*240). *PutPixel* protsedura ekran o‘rtasiga nuqtani qo‘yadi:

PutPixel (x, y, color) protsedura *x, y* koordinatali nuqtani *Color* parametri bilan aniqlangan rangga bo‘yaydi. Masalan, *PutPixel* (100, 120, *Red*) nuqta chaqirilganda (100, 120) koordinatali qizil nuqta paydo bo‘ladi.

PutPixel protsedura kerakli joyga kerakli rangdagi nuqtani qo‘yadi, unga simmetrik bo‘lgan *GetPixel* funksiya yordamida esa, aksincha, mazkur koordinatalarga ega bo‘lgan nuqta qanday rangga ega ekanligini aniqlash mumkin. *GetPixel (x, y)* funksiya *(x, y)* koordinatali nuqtaga rang qiymatini qaytaradi. Shunday qilib, agar *Col* o‘zgaruvchi bo‘lsa, unda:

Col:= *GetPixel* (50, 80);

Operator bajarilganidan keyin, *Col* rang qiymatiga (50, 80) nuqtada ega bo‘ladi.

Graph modulida oddiy shakllarni (kesmalar, aylanalar, ellipslar, to‘rtburchaklar va h.k.) chizish uchun bir necha protseduralar bor.

Line (x1, y1, x2, y2) protsedura *(x1, y1)* nuqtadan *(x2, y2)* nuqtaga kesma o‘tkazadi.

Circle (x, y, radius) protsedura markazi *(x, y)* nuqtada va radiusi Radiusga teng bo‘lgan aylana chizadi.

Rectangle (x1, y1, x2, y2) protsedura yuqori chap burchagi *(x1, y1)* da va quyi o‘ng burchagi *(x2, y2)* da bo‘lgan to‘rtburchak chizadi.

SetColor (Color) protsedura chizilayotgan rasmning joriy rangini o‘rnatadi.

Agar *SetColor* protsedura qandaydir boshqa bir rangni o‘rnatmagan bo‘lsa, joriy rang oq bo‘ladi.

Ranglarni belgilash uchun grafik rejimda ham matn rejimda ishlatiladigan o‘zgarmaslardan foydalanish mumkin:

```
const  Black=0; {qora}
       Blue = 1; {ko‘k}
       Green = 2;{yashil}
       Cyan = 3; {firuza rang}
       Red =4; {qizil}
       Magenta = 5; {ochiq qizil (malinarang}
       Brown = 6; {jigarrang}
       Light Grey =7; {ochiq kulrang}
       Dark Gray = 8; {to‘q kulrang}
       Light blue = 9; {ochiq havorang}
       Light Green = 10; {ochiq yashil}
       Light Cyan =11; {ochiq firuza rang}
       Light Red = 12; {yorug‘ qizil}
       Light Magenta = 13; {yorug‘ ochiq qizil}
       Yellow  = 14; {sariq}
       White =15; {oq}
```

12.3.2.Chiziqlar. Line, Circle, Rectangle protseduralar faqat chiziqlar chizadi. *Turbo Paskal* rasm chizish uchun ranglar bilan to‘ldirilgan shakllarni beradi. *SetColor* protsedura rasmning joriy rangini aniqlagani kabi *SetFillStyle* protsedura ham to‘ldirishning joriy rangini yoki shtrixlarini o‘rnatadi:

SetFillstyle (Style, Color) — to‘ldirishning joriy rangini — *Color* va joriy shtrixlarni (to‘ldirish stilini) o‘rnatadi. To‘ldirishning har xil stillarini ko‘rsatish uchun quyidagi o‘zgarmaslardan foydalanish mumkin:

```
const
EmptyFill - 0; {sohaga tag rang berish}
SolidFill = 1; {sohani berilgan rang bilan uzluksiz to‘ldirib
                borish};
LineFill = 2;  {yo‘g‘on gorizontal chiziqlar bilan ----to‘ldirish}
LtSlashFill = 3; {ingichka qiya chiziqlar./// bilan to‘ldirish}
SlashFill = 4; {yo‘g‘on qiya chiziqlar /// bilan to‘ldirish}
BkSlashFill = 5; {yo‘g‘on qiya chiziqlar \\\ bilan to‘ldirish}
LtbkSlashFill = 6; {qiya yo‘laklar \\\ bilan to‘ldirish}
HatchFill=7;  {katak bilan to‘ldirish}
XHatchFill = 8; {qiya katak bilan to‘ldirish}
InterLeaveFiIl=9; {juda zich qiya shtrixlar bilan to‘ldirish}
WideDotFill =10; {ahyon-ahyonda nuqtalar bilan to‘ldirish}
CloseDotFill =11; {zich nuqtalar bilan to‘ldirish}
UserFill=12; {foydalanuvchi belgilagan shtrixlar bilan
to‘ldirish}
```

12.3.3. Shakllar. Bar (x1, y1, x2, y2) protsedura joriy to‘ldirish rangi va shtrixovkadan foydalanib to‘rtburchak chizadi. To‘rtburchakning yuqori chap burchagi *(x1, y1)* nuqta, *(x2, y2)* esa quyi o‘ng nuqta.

Bar3D (x1, y1, x2, y2, Depth, Top) protsedura esa joriy to‘ldirish rangi va shtrixlardan foydalanib, parallelepiped chizadi. *Depth* o‘zgaruvchi parallelepiped «chuqurligiga» ega. *Tor* mantiqiy o‘zgaruvchi yuqori qirrani chizish kerak-kerakmasligini bildiradi. Agar *Tor* o‘zgaruvchi *True* qiymatiga ega bo‘lsa, yuqori qirra chiziladi, agar *Tor* o‘zgaruvchi *False* qiymatiga ega bo‘lsa, yuqori qirra chizilmaydi.

FillEllipse (X, Y, X Radius, Y radius) protsedura joriy to‘ldirish rangi va shtrixlardan foydalanib ellips chizadi. *Ellips* o‘qlari koor-

dinata o‘qlariga parallel. *x Radius* — ellips kengligi, *y Radius* esa uning balandligi.

```
    uses Graph;
    var Jd, gm: integer;
begin
    jd: = detect ;
    Initgraph (jd, gm, ‘ d: \tp’);
    Setillstyle (7, blue);
    Bar (0, 0, GetMaxX, GetMaxY);
    Setcolor (cyan);
    Setfillstyle (11, light red);
    FillEllipse (GetMaxX div 2, GetMaxY div2, 90, 100);
    Readln;
    CloseGraph;
end.
```

Quyidagi dastur aylanalar chiza borib, kungurali bezak hosil qiladi:

```
    uses Graph;
    var i, j : integer;
    jd, gm: integer;
begin
    d:=Detect; {drayverni avtomatik ravishda aniqlash}
    Initgraph (gd,gm,); {grafik rejimni initsiallash}
    For i:= 0 to 20 do
            For j:= 0 to 20 do
                    Circle (i*40, j*30,64);
    Readln;
    CloseGraph; {grafik rejimni o‘chirish}
end.
```

Ikkita, mos ravishda gorizontal va vertikal imkoniyatlarni qaytaruvchi qulay *GetMaxX* va *GetMaxY* funksiyalar bor.

Bu funksiyalardan ishlanadigan rejimga bog‘liq bo‘lmagan dasturlarni yozishda foydalanish qulay. Quyidagi dastur ixtiyoriy videorejimda ramkani butun ekran bo‘yicha chizadi:

```
uses Graph;
var i, j : integer;
gd, gm: integer;
begin
gd:=Detect; {drayverni avtomatik aniqlash}
Initgraph (gd,gm,); {grafik rejimni initsiallash}
Rectangle (0,0. GetMaxX, GetMaxY);
Readln;
CloseGraph; {grafik rejimni o‘chirish}
end.
```

12.4. Grafik rejimda matn. Ekran sohalari

Grafik rejimda rastr shriftidan va bir necha vektor shriftlaridan foydalanish mumkin. Rastr shrift nuqtalarning matritsasi bilan, vektor shrift esa simvolni tashkil etuvchi vektorlar qatori bilan beriladi.

Rastr simvol kattalashtirilganda uni tashkil qilgan nuqtalarni ko‘rish mumkin bo‘ladi, vektor simvol kattalashtirilganda esa tasvir sifati o‘zgarmaydi. U faqat ekran yuzasidagi nuqtalar soniga bog‘liq bo‘ladi.

Shriftlar fayllari *.CHR* kengaytmasiga ega. *Shrift* yuklanganda tegishli fayl *.BGI* (bu katalog *InitGraph* protsedurada beriladi) grafik drayver yotgan katalogda yoki joriy katalogda bo‘lishi kerak.

Shriftni tanlash va masshtablash *SetTextStyle* protsedurasi yordamida amalga oshiriladi:

SetTextStyle (Font, Direction, Size) — joriy shrift, matnning chiqish yo‘nalishi va simvollar o‘lchamini o‘rnatadi. *Font* shriftni, *Direction* — matnning chiqish yo‘nalishini (chapdan o‘ngga yoki quyidan yuqoriga), *Size* esa shrift o‘lchamini aniqlaydi.

Normal o‘lchamga, rastr shrifti uchun *Size*=l bo‘lganda, vektor shrifti uchun esa — *Size*=4 bo‘lganda erishiladi.

Matnning chiqish yo‘nalishida har xil shriftlarni ko‘rsatish uchun quyidagi o‘zgarmaslar aniqlangan:

```
const
{shriftlar}
DefaultFont= 0 {'8×8 standart rastr shrifti}
TriplexFond= 1 {vektor shrift}
```

SmallFond=2 {vektor shrift}
SansSerifFont= 3 {vektor shrift}
GothicFont = 4 {vektor shrift}
{matnni yo‘naltirish}
HorizDir= 0 {chapdan o‘ngga}
VertDir=1 {quyidan yuqoriga}

OutTextXY (X, Y, TextString) protsedura *(X, Y)* nuqtadan boshlab — *TextString* satrni joriy shrift, joriy yo‘nalish va simvollar o‘lchami bilan chiqaradi.

SetTextJustify (Horiz, Vert) protsedura, keyinchalik *OutTextXY* protseduralar bilan ishlatiluvchi, matnni avtomatik tekislashni o‘rnatadi. *Horiz* — matnni gorizontal, *Vert* esa vertikal tekislaydi.

Tekislashni ko‘rsatish uchun quyidagi o‘zgarmaslar aniqlangan:

```
const
    {gorizontal tekislash uchun:}
    LeftText = 0 {chapdan tekislash}
    CenterText = 1 {markazdan tekislash}
    RightText = 2 {o‘ngdan tekislash}
    {vertikal tekislash uchun:}
    BottomText = 0 {quyidan tekislash}
    TopText = 2 {yuqoridan tekislash}
program GameOver;
uses Graph;
var GD, GM:Integer;
begin
    GD:=Detect;
    InitGraph (GD, GM,'d:\tp\bgi'); {grafikni initsializatsiyalash}
    SetTextJustify (CenterText, CenterText); {tekislashni aniqlash}
    SetTextStyle (TriplexFonttHorizDir, 8); {shriftni, yo‘nalishini
                va o‘lchamini aniqlash}
    {katta bo‘lmagan siljish va har xil ranglar bilan bitta yozuv
    chiqariladi.}
    SetColor (White);
    OutTextXY (GetMaxX div 2, GetMaxY div 2, 'Game Over');
    SelColor (LightBlue);
    OutTextXY (GetMaxX div 2+2,GetMaxY div 2, 'Game Over');
    SetColor (LightRed);
    SetColor (LightRed);
    OutTextXY (GetMaxX div 2+3,GetMaxYdiv 2, 'Game Over');
```

```
    SetColor (LightRed);
    OutTextXY (GetMaxX div 2+4,GetMaxY div 2, 'Game Over');
    SetColor (White);
    Readln;
    CloseGraph;
end.
```

Triplex, SmallFont, SansSerif va *Gothic* vektor shriftlar ruscha simvollarga ega emas, ammo *Borland «BGI TOOLKIT»* firmasining dastur mahsulotida yangi vektor shriftlarni yaratishga yoki mavjud bo‘lgan alohida simvollarni almashtirishga imkon beruvchi shriftlar muharriri bor.

12.4.1. Ekran sohalari. GetImage, PutImage protseduralari va *ImageSize* funksiyadan foydalanib, ekranga tasvirlarning to‘rtburchakli sohalarni eslab qolish va chiqarish imkoniyatlariga ega bo‘lamiz.

ImageSize (x1, y1, x2, y2) funksiya berilgan to‘rtburchakli bo‘lakni yuqori chap burchagi *(xl, yl)*da, quyi o‘ng burchagi *(x2,y2)*da bo‘lgan sohada saqlash uchun zaruriy bo‘lgan xotira sig‘imini baytlarda qaytaradi.

GetImage (x1, y1, x2, y2, Area) protsedura tasvirning to‘g‘ri to‘rtburchakli bo‘lagini yuqori chap burchagi *(x1, y1)*da, quyi o‘ng burchagi *(x2, y2)*da bo‘lgan xotiraning *Area* sohasida saqlaydi.

PutImage (x, y, Area, Mode) protsedura ekranning berilgan joyiga tasvirning bo‘lagini chiqaradi. *(x, y)* — ekranning yuqori chap burchagi sohasi, bu yerga xotiraning *Area* sohasidan tasvir nusxalanadi.

Mode — tasvirni ekranga chiqarish rejimi. Tasvir ekranning berilgan sohasiga u yerdagi eski tasvirni yopgan holda yotishi yoki u bilan ma’lum tarzda o‘zaro holatda bo‘lishi mumkin.

Chiqarishning har xil rejimlarini bayon etish uchun har xil o‘zgarmaslar ishlatiladi:

```
  const {PutImage protsedurasi uchun konstantalar}
      NormalPut=0; {mavjud tasvirni almashtirish}
      XorPut=1; {YOKI (XOR inkor)}
    OrPut = 2 ; {mantiqiy YOKI (OR)}
    AndPut = 3; {mantiqiy VA (AND)}
    NotPut =4 ;{mantiqiy YO‘Q (NOT)}
```

NormalPut rejimida ekranga chiqarilayotgan har bir nuqta ustiga qo‘yilayotgan oldingi nuqtani yopadi. Qolgan rejimlarda qo‘yilayotgan nuqta rangini ifodalovchi har bir ikkilik bit ustiga mos mantiqiy qonunga asosan qo‘yilayotgan nuqta rangining juft biti bilan o‘zaro harakatda bo‘ladi. Ikkala boshlang‘ich nuqtalar ranglaridan farq qiluvchi natija rang hosil bo‘lishi mumkin. Hosil bo‘layotgan natija ranglarni oz bo‘lsa ham o‘zlashtirish uchun quyidagi dastur bilan mashq qilib ko‘rish mumkin.

```
uses Ctr,Graph;
var GD,GM:Integer;
P:Pointer;
i,size:integer;
begin
GD:=Detect;
InitGraph ( GD,GM,'d:\tp\bgi');
SetFillStyle (1,14);
Bar (0,0,GetMaxX,GetMaxY); {(0,0,40,40) to‘g‘ri to‘rtburchakda qandaydir rasm yaratiladi}
SetColor (1);
For I:=0 to10 do
Rectangle (20-2*I,20-2*I,20+2*I,20+2*I);
Out Text XY (80,20,'This is the image'); {(0,0,40,40) bo‘lakni eslab qolish uchun zaruriy xotira sig‘imi aniqlanadi}
Size:=ImageSize(0,0,40,40); {xotiraning talab qilinadigan sohasi ajratiladi va unga R ko‘rsatkich adreslanadi.}
GetMem (P,size); {R ko‘rsatayotgan sohada (0,0,40,40) bo‘lak eslab qolinadi.}
GetImage (0,0,40,40,P^); {NormalPut rejimida chiqarish}
for I:=1 to 10 do
PutImage (i*20,60,P^,NormalPut);
OutTextXY ( 250,80,'NormalPut'); {XorPut rejimida chiqarish}
for I:=1 to10 do
PutImage (i*20,100,P^,XorPut);
OutTextXY (250,120,'XorPut'); {OrPut rejimida chiqarish}
For I:=1 to 10 do
PutImage (I*20,140,P^,OrPut);
OutTextXY ( 250,160,' OrPut'); { AndPut rejimida chiqarish}
for I:=1 to 10 do
PutImage (i*20,180;P^, 'AndPut');
```

```
    OutTextXY ( 250,200,'AndPut');
    Readln;
    CloseGraph;
end.
```

12.5. Grafik rejimda palitra

Palitra — bu ranglar va ekranda real paydo bo‘layotgan ranglar raqamlari o‘rtasidagi moslikdir. Quyidagi uchta protsedura palitra bilan ishlashga imkon beradi:

SetPalette (Col1, Col2) protsedura *Col1* raqamli palitraga *Col2* da ko‘rsatilgan rangni o‘rnatadi.

```
    uses Graph;
    var  Driver, Mode:Integer;
begin
    Driver:=Detect;
    InitGraph(driver,mode,'d:\tp\bgi');
    SetPalette(Black, Blue);
    Readln;
    SetPalette(Black, Red);
    Readln ;
    CloseGraph;
end.
```

SetAllPalette (Palette) protsedura palitraning hamma ranglarini bir vaqtda o‘rnatadi. *Palette* adres bo‘yicha palitrani bayon etuvchi soha bo‘lishi kerak, bunda birinchi baytda palitraning uzunligi, undan keyin ranglar ko‘rsatiladi. *Palette* o‘zgaruvchini, ko‘pincha, oldindan aniqlangan *PaletteType* tipi sifatida bayon etish qulayroq.

```
    Type PaletteType = record
    Size: byte;
    Color: Array[0. . Max Colors] of ShortInt;
end;
```

Bu yerda *MaxColors* — 15 ga teng bo‘lgan, oldindan aniqlangan o‘zgarmas.

Quyidagi dasturda siklik harakatlar imitatsiyasida ishlatiladigan ranglar *SetAllPalette* yordamida siklik ravishda biri ikkinchisiga o‘tib turadi.

```
    uses Ctr , Graph;
    var Gd,Gm, I:integer;
    P:PaletteType;
begin Gd:=detect;
    InitGraph (Gd, Gm, 'd:\tp\bgi'); {ekranni oq rangga bo'yash}
    Bar (0, 0,GetMaxX, GetMaxY); {palitra uchun ranglarni
o'rnatish}
    P.Size:=MaxColors; {ketma-ket, oqdan tashqari, boshqa hamma
ranglar bilan chizish)}
    For I:=0 to MaxColors-1 do
begin
    SetFillStyle(1,I);
    Bar (50+1*10,0,50+(1+I)*10, GetMaxY);
end;
    {siklik ravishda ranglarni almashtirish}
    repeat
            for i=0 to MaxColors-1 do
                P.Colors[I]:=(P.Colors[I]+1) mod MaxColors;
            SetAllPalette(P);
    until keypressed;
    readln
    CloseGraph;
end.
```

Har bir rang qizil, yashil va ko'k tashkil etuvchilarning birlashmasi sifatida ifodalanishi mumkin. *SetRGBPalette* protsedura har bir tashkil etuvchining rangini o'zgartirishga imkon beradi.

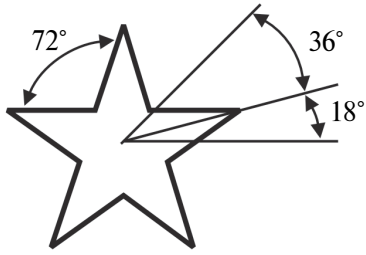
SetRGBPalette (Col, R, G, V) — *Col* raqamli qizil, yashil va ko'k tashkil etuvchi ranglarni mos ravishda *R, G* va *V* ga o'zgartiradi. *R, G* va *V* sonlarning faqat oltita m biti hisobga olinishi uchun ranglarni tashkil etuvchilarning qiymatlarini 0 dan 63 gacha chegarada ko'rsatish ma'noga ega.

SetRGBPalette protsedura *VGA* va *IBM* 8514 adapterlarida ishlaydi.

```
    uses Crt, Graph;
    var Gd, Gm:integer;
begin Gd:=Detect;
    InitGraph (Gd, Gm, 'd:\tp\bgi'); {qora ekran}
    Readln; {oq-pushti toblanish — qizil tashkil etuvchi salgina
    ustunlikka ega}
```

```
    SetRGBPalette (Black, 55, 45, 45);
    Readln; {binafsharang — ko‘k va qizil tashkil etuvchilar ustunlikka ega}
    SetRGBPalette (Black, 30, 5, 30);
    Readln;
    CloseGraph;
end.
```

Endi grafik yasashlarga doir bir necha masalalarning dasturlarini tuzib ko‘rsatamiz.



Masala 1. Ekranda rasmda ko‘rsatilgan burchak o‘lchamlarida besh burchakli yulduz chizuvchi dastur matni:

```
    uses Graph;
    Label bye;
    var
      R: integer; {yulduz radiusi}
      XO, YO: integer; {yulduz markazi koordinatalari}
      X, Y: integer; {OX o‘q va yulduz markazi hamda nur uchi
      bilan birlashtiruvchi to‘g‘ri chiziq o‘rtasidagi burchak}
      A: integer;  I: integer;
      GrDriver: integer; {drayver}
      GrMode: integer;  {grafik rejim}
      Err Code: integer;  {grafik rejimni initsializatsiya qilish natijasi}
      Res: integer;
begin
      GrDriver:=Detect;
      InitGraph (GrDriver, GrMode, 'c:/to/bji ');
      ErrCode:=GraphResult;
      if ErrCode<>gr Ok then
begin
      Writln ('Grafik rejim initsializatsiya xatosi');
      Goto bye;
end;
```

```
    XO:=100;  YO:=100;  R:=20;
    A:=18; {o‘ng gorizontal nurdan boshlab yasash}
    X:=XO+Round (r/2*cos(a+2*pi/360));
    Y:=YO- Round (r/2*sin(a+2*pi/360));
    move to (x, y);
    For i: =1 to 5 do
begin
    A:=a+36;
    X:=XO+Round (r/2*cos (a+2*pi/360));
    Y:=YO- Round(r/2*sin (a+2*pi/360));
    Line to (x, y); A:=a+36;
    If a >360 then a:=18;
    X: =xo+Round (r*cos (a+2*pi/360));
    Y:=yo-Round (r*sin (a+2*pi/360));
    Line to (x, y);
end;
    Readln; Bye:
end.
```

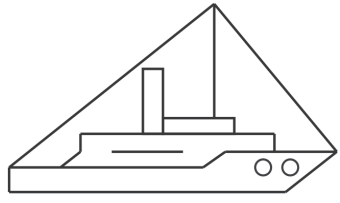
Masala 2. Ekranga Y=0.5x2+4x-3 funksiyaning nuqtali grafigini chizish dasturini tuzish. Argumentning o‘zgarish sohasi — 15 dan 5 gacha, o‘zgarish qadami — 0.1 ga teng. Grafik koordinatalar tekisligida chiqariladi, o‘qlarning kesishishi ekran markazida bo‘ladi.

Ekranga berilgan funksiya grafigini chiqaruvchi dastur matni:

```
uses Graph;
var
GrDriver: integer;   {drayver}
GrMode: integer;     {grafik rejim}
GrPath : string;     {drayver joylashish o‘rni}
ErrCode: integer;    {grafik rejimni initsializatsiya qilish natijasi}
mx, my: integer;     {x va y bo‘yicha masshtab, ekranning koordinata o‘qlari birligiga mos keluvchi nuqtalar soni}
px, py: integer;  X0, Y0: integer; {koordinata o‘qlari boshi}
px, py : integer;    {ekrandagi nuqtalar koordinatalari}
x, dx: real;         {argument va uning orttirmasi}
x1, x2: real;        {argumentning o‘zgarish sohasi}
y: real;             {funksiya qiymati}
begin
GrDriver:=VGA;       {(VGA rejimi})
```

```
GrMode := VGAHiI;   {(zichligi 640*480}
GRPath:='e:/tp/bgi';   {drayver, EGAVGA.BGI fayl e:/tp/bgi katalogda}
InitGraph (GrDriver, GrMode, 'c:/to/bgi ');
ErrCode:=GraphResult;
If ErrCode<>gr Ok then
begin
Writeln;   ('Grafik rejim initsializatsiya xatosi');
Writeln; ('Dastur natijasini yakunlash uchun 'enter'ni bosish kerak ');
Readln; Halt (1);
end;
xy:=320;  y0:=240;  mx:=20; my:=20;
Line (10, y0, 630, y0);
Line (x0, 10, x0, 470);
x1:=-15;  x2:=5;  dx:=0.1;  x:=x1;
While (x<x2) do
begin
y:=0.5*x*x+x*4-3;
px:=x0+round (x*mx);
py:=y0-round (y*my);
PutPixel (px, py, white);
x:=x+dx;
end.
```

Masala 3. Ekranga qayiqcha rasmini chizuvchi protsedura-dasturni yozish. Parametr sifatida protsedura chizishni olib borishi kerak bo‘ladigan asos nuqtaning koordinatalari va rangi qabul qilinadi.



Ekranga harakatlanuvchi murakkab tasvirni (qayiqchani) chiqaruvchi dastur matni:

```
uses Graph;
var
GrDriver: integer;  {drayver}
```

```
    GrMode: integer;    {grafik rejim}
    GrPath: string;     {drayver joylashish o‘rni}
    ErrCode: integer;   {grafik rejimni initsializatsiya qilish natijasi}
    x,y: integer;       {qayiqcha koordinatalari}
    Color: word;        {qayiqcha rangi}
    BkColor: word;      {ekran tag rangi}
    {qayiqcha}
    Procedure Titanik (x, y: integer; color, word); {asos nuqta
koordinatalari, qayiqcha rangi}
    Const
    dx=5;  dy=5;
    Var
    OldColor: word;
begin
    OldColor:= GetColor;    {joriy rangni saqlash}
    SetColor (color);       {Yangi rangni o‘rnatish}
    {konus}
    MoveTo (x,y);  LineTo (x, y-2*dx);
    LineTo (x+10*dx, y-2*dy);
    LineTo (x+11*dx, y-3*dy);
    LineTo (x+17*dx, y-3*dy);
    LineTo (x+14*dx, y);  LineTo (x, y);
    {ustqurma}
    MoveTo (x+3*dx, y-2*dy);
    LineTo (x+4*dx, y-2*dy);
    LineTo (x+4*dx, y-4*dy);
    LineTo (x+13*dx, y-4*dy);
    LineTo (x+13*dx, y-3*dy);
    LineTo (x+5*dx, y-3*dy, x+9*dx, y-3*dy);
    {kapitan ko‘prikchasi}
    Rectangel (x+8*dx, y-4*dy, x+22*dx,y-5*dy);   {truba}
    Rectangel (x+7*dx, y-4*dy, x+8*dx,y-7*dy);    {illuminatorlar}
    Circle (x+12*dx, y-2*dy, Trunc (dx/2));
    Circle (x+14*dx, y-2*dy, Trunc (dx/2));       {machta}
    Line (x+10*dx, y-5*dy, x+10*dx, y-10*dy);     {ustunlar}
    MoveTo (x+17*dx, y-3*dy);
    LineTo (x+10*dx, y-10*dy);
    LineTo (x, y-2*dy);
    SetColor (OldColor);                  {joriy rangni tiklash}
end;
```

```
begin
    GrDriver:= VGA;          {rejim VGA}
    GrMode := VGAHi;         {ekran nuqtalari zichligi 640*480}
    GrPath:='e:/tp/bgi';     {drayver EGAVGA.BGI, fayl e:/tp/bgi
katalogda}
    InitGraph (GrDriver, GrMode, 'c:/to/bgi ');
    ErrCode:= GraphResult;
    If ErrCode<> gr Ok then
begin
   x:=10;  y:=200;
end;
    Color := Light + Gray;
    SetBkColor (Blue);
    BkColor:=GetBkColor;
    Repeat
    Titanik (x, y, color);   {qayiqchani chizish}
    Delay (100);
    Titanik (x, y, bkcolor); {qayiqchani o'chirish}
    PutPixel (x, y, color);  {qayiqcha izi}
    x:=x+2;
    Until (x>500);
    OutText XY (10, 10, 'Reys tugadi');
    Readln;
    CloseGraph;
end.
```

Masala 4. Kitob do'konining tovar aylanishini aks ettiruvchi aylana diagrammani chiqarish dasturini tuzish. Boshlang'ich qiymatlar bo'lib so'mda toifalari bo'yicha sotilgan jurnallar, otkritkalar va kanselariya mollari xizmat qiladi.

Ekranga aylana diagrammani chiqaruvchi dastur matni:

```
Uses Graph;
Const
N=4;          {toifalar soni}
Name: array [1..n] of String [10]=('kitoblar', 'jurnallar',
'kansmollar', 'va boshqalar');
Var
son: array[1..n] of real;  {toifalar soni}
his: array [1..n] of real;  {toifalar uchun umumiy sonda}
sum: real;          {hamma toifalar uchun umumiy soni}
a1, a2: integer;    {sektorni boshlash va tugatish burchagi}
x, y: integer       {izohni chiqarish koordinatalari}
st: string;         {sonni tasvirlash}
i: integer;
GrDrive: integer;   {drayver}
GrMode: integer;    {grafik rejim}
GrPath: string;     {drayverning joylashgan o'rni}
ErrCode: integer;   {grafik rejimni initsiallash natijasi}
begin
GrDrive: = VGA;          {rejim VGA}
GrMode: = VGAHi;         {nuqtalar zichligi 640*480}
GrPath: = 'e:/tp/bgi';
InitGraph (GrDrive, GrMode, GrPath);
ErrCode:=GraphResult;
If ErrCode<> gr Ok then
begin
Writeln ('Grafik rejimni initsiallash xatosi');
Writeln ('Ishni tugatish uchun <ENTER>ni bosing');
Readln;
Halt (1);
end;
{boshlang'ich ma'lumotlarni kiritish}
Writeln ('Har bir toifa bo'yicha sonlarni kiriting');
sum:=0;
For i:=1 to n do
begin
Writeln (name[i], '—');
Readln (son [i]);
sum:=sum+son[i];
end;
```

```
    {har bir toifaning umumiy yig‘indidagi hissasini aniqlash}
    For I:=1 to n do his[i]:=son[i]/sum*100; {diagrammani yasash}
    a1:=0;          {OX o‘qidan}
    x:=350; y:=100; {izoh yoziladigan sohaning yuqori chap bur-
chagi}
    For i:=1 to n do
begin
    {sektor}
    a2:=a1+ round (3.6* dol [i]); {3, 6 gradusning 1% i}
    If a 2.3 to then a 2:=350;
    SetFileStyle (Solid fill, i);
    PieSlice (200, 200, a2, 100);
    a1:=a2; {joriy oxirning navbatdagi sektori}
    {izoh}
    Bar (x, y, x+30, y+10);
    Restangle (x, y, x+30, y+10);
    str (his [i]:6:1, st);
    OutText  XY (x+50, y, name [i] + '-' +st + '%');
end;
    Readln;
    CloseGraph;
end.
```

Nazorat savollari

1. Kompyuter ekranining qanday xususiyatlari bor?
2. Grafik drayverlar nima ish bajaradi va ular qanday fayllarda joylashgan?
3. Har xil fayllarni va rejimlarni ko‘rsatish uchun *Graph* modulida qanday o‘zgarmaslar aniqlangan?
4. Ekran grafik rejimga qanday o‘tkaziladi?
5. Qanday o‘zgarmaslar *GraphResult* qaytarish kodlariga mos keladi?
6. *PutPixel (x, y, color)* protsedura qanday ish bajaradi?
7. *GetPixel (x, y)* funksiya qanday ish bajaradi?
8. *Graph* modulida oddiy shakllarni chizish uchun qanday protseduralar bor?
9. *SetTextStyle (Font,Direction, Size)* protsedura nima ish bajaradi?
10. Palitra nima?
11. *SetPalette (Col1, Col2)* protsedura nima ish bajaradi?
12. *SetAllPalette (Palette)* protseduraning vazifasi nima?
13. *SetRGBPalette (Col, R, G, V)* protsedura nima ish bajaradi?

XIII bob. DASTURLASH USLUBIYATI

13.1. Masaladan dasturga

Endi shu paytgacha ko‘rib chiqilgan dasturlash vositalarini ko‘plab qiziqarli dasturlar yaratishga qo‘llaymiz. Agar dasturchi oldiga murakkab masala qo‘yilgan bo‘lsa, unga mos dasturni yaratish uchun uzoq vaqt va ko‘p harakat talab qilinadi. Shuning uchun dasturni yozishda ma’lum qoidalar to‘plamiga yoki dasturlash uslubiyatiga rioya qilish kerak bo‘ladi.

Masala dasturini yaratish jarayonini quyidagi bosqichlarga bo‘lish mumkin:

1. Masalani ta’riflash.
2. Yechish usuli (algoritmi)ni tanlash va tasvirlash.
3. Dasturni yaratish.
4. Dasturni tekshirish.
5. Dasturni takomillashtirish.
6. Dasturni tuzatish.

Quyida sanab o‘tilgan qismlarning har biri alohida ko‘rib chiqiladi.

13.2. Masalani ta’riflash

Dasturni yaratishda, eng avval, berilgan masalaning maqsadi va bo‘lajak dasturga qo‘yiladigan talablarni bilish kerak. Aks holda, dasturchi ayrim tafsilotlarni nazarga olmasligi va masalani yechish uchun zarur bo‘lgan, ammo uning tasvirida aniq ifodalanmagan amallar dasturda o‘z aksini topmasligi mumkin. Masalani ta’riflashda

dasturda qaysi dastlabki ma'lumotlardan foydalanish, qanday natijalar olish kerakligi aniq ko'rsatilishi kerak. Talab qilinayotgan natijalarni olish uchun zarur bo'lgan amallarga kelsak, ular keyinroq, dastur tuzishda aniqlanadi.

Misolni ko'rib chiqamiz. Yillarni ifodalovchi dastlabki ma'lumotlar ichidan kabisa yillarini aniqlaydigan dastur tuzish kerak. Shubhasiz, bunday topshiriqdan masalani aniq ta'riflash mumkin emas.

Birinchidan, dastlabki ma'lumotlarning o'zgarish chegaralari ko'rsatilmagan. Chunki yuz yillikni bildiradigan kabisa yillari alohida formula bilan aniqlanadi, shuning uchun dastlabki ma'lumotlar bizning eramizning yuz yilliklarinimi yoki eramizgacha bo'lgan yillarni bildiradimi shuni bilish zarur. *Ikkinchidan,* EHM natijalarni qaysi ko'rinishda bosib chiqarishi kerakligi aytilmagan.

Masalaning ta'rifini aniqlaymiz. Dastlabki ma'lumotlar — 1 dan 2000 gacha bo'lgan bizning eramizning yillarini ko'rsatuvchi sonlar. Shunday dastur yozish kerakki, unga asosan EHM, agar berilgan yil kabisa yili bo'lsa, KABISA va aks holda ODDIY so'zlarni bosib chiqarsin. Masalani bunday ta'riflash ancha aniq.

Boshqa misolni ko'rib chiqamiz. Berilgan sonlar yig'indisini hisoblash kerak. Bu yerda dastlabki ma'lumotlar sonini yoki sonlarning berilgan ketma-ketligi uzunligini aniqlash usulini bilish kerak. Bunday masalani ta'riflashning ayrim variantlarini keltiramiz:

1. *Dastlabki ma'lumotlar* — ikkita butun son. Ularni dasturga kiritish, yig'indisini hisoblash va bu yig'indini bosib chiqarish kerak.

2. *Dastlabki ma'lumotlar* — o'nta butun sonlar. Ularni dasturga kiritish, yig'indisini hisoblash va bu yig'indini bosib chiqarish kerak.

3. *Dastlabki ma'lumotlar* — nolga teng bo'lmagan sonlar ketma-ketligi. Ketma-ketlikning oxiri — nol. Dastlabki ma'lumotlarni o'qish, yig'indisini topish va uni bosib chiqarish kerak.

13.3. Yechish usuli (algoritmi)ni tanlash va tasvirlash

Masalani yecha borib, EHM dasturda aniqlanadigan katta hajmdagi hisoblash ishlarini bajaradi. Masalani yechuvchi dasturchi

esa yechish usulini, ya'ni qo'yilgan masalaning algoritmini bilishi zarur. Masalan, kabisa yillarini aniqlovchi dasturni tuzishdan avval dasturchi kabisa yilini aniqlash qoidasini bilishi kerak.

Agar masalani yechishning bir necha usullari ma'lum bo'lsa, uni tanlashda usulning universalligi, soddaligi va tejamliligini hisobga olish kerak. Tanlangan usul dasturning maksimal tezkorligini, xotiraning eng kam sarflanishini va dasturlash nuqtayi nazaridan eng yaxshi xususiyatlarni ta'minlashi kerak.

13.4. Dasturni yaratish

Tajriba shuni ko'rsatadiki, har bir dastur faqat o'ziga xos xususiyatlarga ega, shu bilan birga, yangi yaratilayotgan dasturni bir nechta, avval yozilgan, alohida dasturlardan yaratish ham mumkin. Dasturlash — bu ijodiy jarayon, shuning uchun dastur tuzishning umumiy usuli yo'q. Ammo ba'zi usullarni tavsiya qilish mumkin, ulardan foydalanib, dasturchi o'z ishini yengillashtiradi va uni ancha oson bajaradi. Bunday usullardan biri — masalani qismlarga ajratish. Katta masalalarni dasturlashda ma'lum qiyinchiliklar uchraydiki, amalda ularni ta'riflash mumkin emas.

Bunday hollarda masala, bir-biriga bog'liq bo'lmagan, dasturlanadigan qismlarga ajratiladi. Agar alohida qism hali ham ancha katta bo'lsa, uni qismlarga ajratishni, masalaning hosil qilingan qismlari ancha sodda va aniq bo'lgunga qadar, davom ettirish kerak. Bu usulni ta'riflangan masala misolida tushuntirib beramiz.

Dastlabki ma'lumotlar — bizning eramizning 1-dan 2000-gacha bo'lgan yillariga mos keluvchi sonlar. Shunday dastur yozish kerakki, u bajarilganda EHM kabisa yil uchun KABISA, aks holda ODDIY so'zlarni bosib chiqarsin. Bu masalani qismlarga ajratamiz:

1. Dastlabki ma'lumotlarni kiritish.
2. Berilgan yil kabisa yil ekanini tekshirish.
3. Natijalarni bosib chiqarish.

Ma'lumotlarni kiritish va natijalarni bosib chiqarish bu yerda ortiqcha qiyinchilik tug'dirmaydi, chunki ularga *Turbo Paskal* tilining ma'lum operatorlari mos keladi. Agar kabisa yillarini aniqlaydigan

funksiya ma'lum bo'lsa, dasturning ikkinchi qismini amalga oshirish osonlashadi. Bunday funksiyani keyinroq yozamiz, hozir esa yaratiladigan dasturning loyihasi (chizmasi)ni tuzamiz.

```
program kabisalar (input, output);
var yil: integer {(yil ≥ 1) and (yil ≤ 2000)};
function kabisa (y:integer) : boolean;
```

agar *y* yil kabisa bo'lsa, funksiyaning qiymati *true*, aks holda — *false* bo'ladi.

```
begin
        read (yil);
        if kabisa (yil) then write ('KABISA')
              else write ('ODDIY')
end.
```

Kabisalar dasturida *kabisa* funksiya tasviri to'g'ri burchakli to'rtburchak bilan almashtirilgan. Masalaning birinchi va uchinchi qismi dasturlashtirildi, qolgan qismiga esa yangi masala deb qarash mumkin. Uning ta'rifi to'g'ri burchakli to'rtburchak ichida yozilgan.

Kabisa funksiyani tasvirlash uchun yechish usuli — qoidasini aniqlab olish zarur. Bu usul (qoida)ga asosan kabisa yillari aniqlanadi:

1. Asrlarni (misol uchun, 1800, 1900, 2000) bildiradigan yillar soni to'rtga bo'linsa, ular kabisa yillari bo'ladi. Masalan, 2000-yil kabisa yil, chunki 18 va 19 sonlari 4 ga bo'linmaydi.

2. 4 ga bo'linadigan qolgan hamma yillar kabisa yillari bo'ladi. Masalan, 1988-yil kabisa, 1987-yil esa kabisa emas.

Bu qoidalar yordamida funksiyani tuzamiz:

```
function kabisa (y:integer) : boolean;
begin
        if y mod 100 = 0
```

```
        then  {asr}
                  if y mod 400 = 0
                  then kabisa:=true
                  else kabisa :=false
              else  {asr emas}
                        if y mod 4=0
                  then kabisa:=true
                  else kabisa:=false
    end.
```

Funksiyani dasturdagi to‘rtburchak o‘rniga qo‘yamiz:

```
program kabisalar (input, output);
    var yil:integer ((yil ≥ 1) and (yil ≤ 2000));
    function kabisa (y:integer) : boolean;
    begin
      if y mod 100 = 0
          then  {asr}
                  if y mod 400 = 0
                  then kabisa :=true
                  else kabisa :=false
              else  {asr emas}
                  if mod 4=0
                  then kabisa:=true
                  else kabisa:=false
          end;
    begin         {dastur boshi}
          read (yil);
          if kabisa yil
              then write ('KABISA')
              else write ('ODDIY')
    end.
```

Dasturning mustaqil qismlarini funksiyalar va protseduralar

ko‘rinishida yozish qulayligi ko‘rinib turibdi. Katta dasturlarni yaratishda, odatda, bu usullardan foydalaniladi.

Ko‘rib chiqilgan masala anchagina sodda. Uni qismlarga ajratmasdan ham yechish mumkin edi. Biroq, kelgusida bizga murakkab masalalarni yechishga to‘g‘ri keladi, shuning uchun yuqorida tasvirlangan usulni puxta o‘rganib olish muhim.

Yana bir bor ta’kidlaymiz, agar masalani qismlarga bo‘lishda hosil bo‘lgan qism hali ham juda murakkab bo‘lsa, tavsiya qilingan usulni qayta qo‘llash kerak. Masalaning bunday qismi so‘z bilan ta’riflanadi yoki to‘rtburchakda tasvirlanadi. Dasturda birorta ham bunday to‘rtburchaklar qolmagandan keyingina dastur tamom bo‘ldi, deb hisoblanadi.

Dasturning alohida qismlarini faqat funksiyalar va protseduralar ko‘rinishida dasturlash har doim ham foydali bo‘lavermasligini misolda ko‘rsatamiz. Dasturning yuqorida ko‘rgan loyihasini quyidagi ko‘rinishda yozish mumkin:

```
program kabisalar (input, output);
    var yil:integer        (( yil ≥ 1) and (yil ≤ 2000));
            b : boolean;
    begin
            read (yil);
```

Yilning kabisa ekanligini tekshirish, *ha* bo‘lsa, *b* o‘zgaruvchiga *true* qiymat, aks holda, *false* qiymat o‘zlashtiriladi.

```
    if b
        then write ('KABISA')
        else write ('ODDIY ')
    end.
```

To‘rtburchakda tasvirlangan dasturning qismini shartli operator yordamida bajaramiz:

```
if yil mod 100 = 0 then
        if yil mod 400 = 0
              then b:=true
              else b :=false
  else
        if yil mod 4=0
              then b:=true
              else b:=false
```

Shartli operatorni to‘rtburchak o‘rniga yozib, tugallangan dasturni olamiz.

Muhokama qilinayotgan masalani amalga oshirishning bu ikki ko‘rinishini solishtirib, funksiyani o‘z ichiga olgan dasturning o‘lchami birmuncha katta ekanligini ko‘ramiz. Ammo funksiya bu dasturning mustaqil qismi, demak, unda dasturning boshqa qismlarida o‘zgaruvchilar qanday nomlanganligini hisobga olmasdan, o‘zgaruvchilar uchun boshqa nomlarni ishlatish mumkin.

13.5. Dasturni tekshirish

Masalani dasturlashda osongina adashish mumkin.

Dasturning to‘g‘riligini har xil yo‘llar bilan tekshirish mumkin. Ulardan eng soddasini ko‘rib chiqamiz. U shundan iboratki, aniq dastlabki ma’lumotlarga ega bo‘lgan holda dasturchining o‘zi, EHM ishtirokisiz, dasturda yozilgan amallarni bajaradi. Bu holda barcha amallarni, xuddi mashina bajaradigandek, ularning mohiyatini o‘ylab ko‘rmasdan, beixtiyor bajarishi kerak. Olingan natijalarga qarab, dasturning to‘g‘riligini muhokama qilish mumkin.

Ikki sonini *n*-darajaga ko‘taradigan dastur yozamiz:

```
program daraja (input, output);
var k, n, p : integer;
begin
  read (n);
  p:=1
  for k:=1 to n do
        p:=p*2;
  write (p);
end.
```

Bu dasturni tekshirish uchun bir varaq qog‘oz olib, unga dastur matnida uchraydigan barcha o‘zgaruvchilarni yozib chiqamiz (jadvalning birinchi yo‘li):

O‘zgaruvchilar qiymatlari	*K*	*N*	*p*
Dasturning bajarilishidan oldin	?	?	?
read (n) operatori bajarilgandan keyin	?	3	?
p: = 1 operatori bajarilgandan keyin	?	3	1
1-sikl bajarilgandan keyin	1	3	2
2-sikl bajarilgandan keyin	2	3	4
3-sikl bajarilgandan keyin	3	3	8
Dastur bajarilgandan keyin	3	3	8

Dastur bajarilgunga qadar o‘zgaruvchilarning qiymatlari aniqlangan emas. Dasturning bajarilishida o‘zgaruvchilarga aniq qiymatlar o‘zlashtiriladi, ularni jadvalga ustun bo‘ylab pastga tomon yozib boramiz.

Daraja dasturda *read (n)* operator xotiradan *n* ning qiymatini o‘qiydi. *n=3* deb, faraz qilamiz, *p* — o‘zgaruvchiga 1 qiymat o‘zlashtiriladi. Sikl operatorlari bajarilganda *k* va *p* o‘zgaruvchilarning qiymatlari doim o‘zgarib boradi. Hisoblash oxirida *k=3, p*= 8 bo‘ladi. Demak, *daraja* dastur 2 sonini 3 = darajaga to‘g‘ri ko‘taradi, chunki $2^3 = 8$.

Dasturni tashkil qiluvchi funksiyalar va protseduralarni bir-biridan mustaqil ravishda tekshirish, olingan natijalarni esa

masalaning hammasini tekshirishda ishlatish maqsadga muvofiqdir. Rekursiv protsedura funksiyalarning to‘g‘riligini aniqlash ancha murakkab. O‘zgaruvchilarning qiymatlarini aniq ko‘rsatish uchun rekursiv protsedura yoki funksiyaning har bir nusxasini alohida jadvalga yozish kerak.

Dasturning to‘g‘riligini tekshirishda foydalaniladigan ma’lumotlar nazorat ma’lumotlari deyiladi. Qulay va shu bilan birga aynan shu masala uchungina xos nazorat ma’lumotlarini tanlash juda muhimdir. Masalan, agar dasturga sikl kiritilgan va uning takrorlanish soni dastlabki ma’lumotga bog‘liq bo‘lsa, bu holda nazorat qiymat sifatida shunday qiymatni tanlash maqsadga muvofiqki, unda sikl amallarini ko‘p marta bajarish talab qilinmaydigan bo‘lsin. Aks holda siklni tekshirish ko‘p vaqtni oladi.

Yuqorida yozilgan *daraja* dasturini tekshiramiz. $n = 3$ da dastur to‘g‘ri tuzilganligi ko‘rsatilgan edi. Yana $n=0$ va $n=1$ bo‘lgan ikki holni ko‘ramiz. Bu qiymatlar bilan dastur amallarini bajarib, mos ravishda $p=1$ va $p=2$ natijalarni olamiz. $n=0$ da dastur sikli bajarilmaydi va p o‘zgaruvchining qiymati 1 ga teng bo‘lib qolaveradi. Demak, daraja dastur, 2 sonini, $n \geq 0$ darajaga ko‘tarishga mo‘ljallangan. Shunday qilib, siklning to‘g‘riligini tekshirish uchun shunday nazorat qiymatlarini tanlash kerakki, unda sikl bir marta, bir necha marta bajariladigan yoki umuman bajarilmaydigan bo‘lsin. Shartli operatorni tekshirishda uning har bir qismi (tarmog‘i) hech bo‘lmaganda bir marta bajariladigan bo‘lishi kerak. Masalan, *kabisa* funksiyaning to‘g‘riligiga ishonch hosil qilish uchun ushbu 1600, 1900, 1988, 1987 nazorat ma’lumotlardan foydalanamiz. Funksiyada ko‘rsatilgan amallarni bajarib, *true, false, true, false* to‘g‘ri natijalarni olamiz.

Ba’zi nazorat qiymatlarda dastur natijalari to‘g‘ri bo‘lsa, dasturning o‘zi ham to‘g‘ri tuzilganligini tasdiqlash mumkinmi? Yo‘q, albatta!

Boshqa dastur misolida bu fikrning to‘g‘riligini tushuntirib boramiz. 2 sonini n darajaga ko‘tarish uchun quyidagi dastur tuzilgan bo‘lsin:

```
program daraja (input, output);
var k, n, p: integer;
begin
        read (n); p:=1;
        for k:=1 to n do
                p:=k * 2;
        write (p)
end.
```

daraja dasturning natijalari 0, 1, 2 dastlabki qiymatlarda, mos ravishda, 1, 2, 4 ga teng, ya'ni *daraja* dasturning natijalari bilan mos keladi. Biroq, $n \geq 2$ da noto'g'ri natijalar chiqishiga ishonch hosil qilish qiyin emas. Masalan, $n=3$ qiymatlar p 8 ga emas, 6 ga teng bo'ladi.

Dasturning to'g'riligini tekshiradigan boshqa ishonchli usul shundan iboratki, masalaning ta'rifi bo'yicha dastlabki ma'lumotlarning mumkin bo'lgan har qanday qiymatlarida dasturning to'g'riligini isbotlashdir. Dasturning natijalarini aniq qiymatlar bo'yicha tekshirishga qaraganda bu ancha murakkab. Har bir dastur tuzilishining to'g'riligini ko'rsatishga (asosan, misollarda) harakat qilib ko'ramiz.

O'zlashtirish operatori. Operatorning o'ng qismiga o'zgaruvchining qiymatini hisoblaydigan arifmetik ifoda yoziladi. Agar u to'g'ri yozilgan bo'lsa, o'zlashtirish operatori ham to'g'ri bo'ladi. U holda ifodaga tegishli hamma o'zgaruvchilarning qiymatlari aniqlangan yoki EHM xotirasiga kiritilgan yoki protsedura bajarilishida hisoblangan yoki bundan oldingi o'zlashtirish operatorlarga o'zlashtirilgan bo'lishi kerakligiga e'tiborni qaratish zarur. Ifodalarga o'zgaruvchilarning noma'lum qiymatlarini ishlatish dastur tuzishni o'rganayotgan dasturchilarning ko'p takrorlaydigan xatosidir. Barcha aytilganlar shartli operatorlardagi (*if*dan keyingi) va sikllardagi (*while*dan keyingi) mantiqiy ifodalarga ham taalluqlidir.

Shartli operator. Misol ko'rib chiqamiz. Dastlabki ma'lumot butun a soni bo'lsin, b natija a sonining mutlaq qiymatiga teng bo'lishi kerak. Bu masalani yechish uchun har xil dasturning to'rt parchasini ko'ramiz.

1. *if a < 0 then b : = - a*

a<0 bo‘lganda *b:= - a* operator bajariladi, ya’ni *a* o‘zgaruvchining manfiy qiymatlari uchun shartli operator to‘g‘ri yozilgan. Agar *a≥0* bo‘lsa, dastur natijasi aniqlanmagan. Demak, bu dastur noto‘g‘ri tuzilgan.

2. *b : = a;*
if a < 0
then b : = - a

a<0 bo‘lganda *b:= - a* operator bajariladi, ya’ni hosil qilingan natija to‘g‘ri bo‘ladi, *a≥0* da *b=a,* chunki shartli operator bajarilmaydi. Bu natija ham to‘g‘ri. Demak, ikkinchi holda dastur to‘g‘ri tuzilgan.

3. *if a > 0*
then b : = a
else
if a < 0
then b : = - a

a>0 bo‘lganda *b* ning qiymati *a* ga, *a<0* da esa *—a* ga teng. Ikkala natija ham to‘g‘ri, lekin *a=0* hol ko‘rib chiqilmadi. *a=0* da *b* aniqlangan bo‘ladi, chunki shartli operatorning hech qaysi tarmog‘i bajarilmaydi. Demak, dastur noto‘g‘ri tuzilgan.

4. *if a > 0*
then b : = a
else b : = - a

a>0 bo‘lganda *b:= - a* operator bajariladi. Qolgan hamma hollarda *b:=-a*. Olingan natijalar to‘g‘ri. Barcha mumkin bo‘lgan dastlabki ma’lumotlarning qiymatlarida to‘g‘ri natijalar hosil bo‘layotgani uchun dasturning to‘g‘riligi haqida xulosa chiqarish mumkin.

Sarlavhasida *while* bo‘lgan sikl. Siklni tekshirishda birinchi aniqlanadigan narsa shuki, har qanday mumkin bo‘lgan boshlang‘ich qiymatlarda bu sikl yakunlovchi bo‘ladimi?

Masalan,

```
while a > 0 do
write ('A')
```

sikl $a>0$ da cheksiz bajariladi, chunki bu o‘zgaruvchining qiymati siklning har bir takrorlanishida o‘zgarmaydi.

Sikl ishining tugallanishini ta’minlovchi zaruriy (ammo yetarli bo‘lmagan) shartni ko‘rsatamiz. Zaruriy shart — bu sikl sarlavhasidagi shartga kiruvchi biror o‘zgaruvchining qiymati sikl bajarilganda, albatta, o‘zgarishi kerakligidir. Misol ko‘ramiz:

```
while a > 0 do:
a : = a + 1.
```

Sikl ichida uning o‘zgaruvchisining qiymati o‘zgarib borishi ko‘rinib turibdi, ammo bu sikl tamom bo‘lmaydi, chunki *a* ning qiymati doimo oshib boradi.

Boshqa misol keltiramiz:

```
while a≥0 do:
a : = a — 1.
```

Bu yerda sikl chekli. Siklning har bir bajarilishida *a* o‘zgaruvchining qiymati manfiy bo‘lgunga qadar 1 ga kamayib boradi. Unda sikl sarlavhasida yozilgan shart buziladi va, demak, sikl o‘z ishini tamomlaydi. Har qanday boshlang‘ich qiymatlarda siklning to‘g‘riligini tekshirish ancha murakkab, chunki buning uchun siklda ko‘rsatilgan amallarni ko‘p marta bajarish kerak.

Keyinchalik aynan bir masalani tahlil qilishda *while* siklni tekshirishga yana qaytamiz.

Sarlavhasida *for* bo‘lgan sikl. Bunday turdagi sikllar quyidagi umumiy xususiyatga ega: agar sikl ichida uning o‘zgaruvchisining

qiymati o‘zgarmasa, bu sikl chekli va uning takrorlanish soni sikl sarlavhasida aniqlangan bo‘ladi. *for* siklning bu xususiyatini dasturchi dastur yaratayotganda yoddan chiqarmasligi kerak.

Protseduralar va funksiyalarni dasturning mustaqil qismi deb tekshirish tavsiya qilinadi. Rekursiv funksiyalar va protseduralar haqida alohida fikr yuritamiz. Ular xuddi siklga o‘xshash cheksiz bajarilishi mumkin. Masalan:

```
procedure p;
begin
        p; write ('A')
end.
```

Funksiyani yoki protsedura rekursiv bo‘lganda, dastlabki ma’lumotlar (parametrlar)ning har qanday qiymatlarida dastur chekli bo‘lishini tekshirish kerak. Rekursiv funksiyalar va protseduralar chekli bo‘lishi uchun ikki zaruriy (biroq, yetarli bo‘lmagan) shartga rioya qilish kerak:

1. Rekursiv funksiya (protsedura)ga murojaat qilish ma’lum shartda bajarilishi kerak (masalan, rekursiv murojaat qilish shartli operatorning qismi bo‘ladi).

2. Rekursiv funksiya (protsedura)ning tasvirida hech bo‘lmaganda bitta o‘zgaruvchining qiymati o‘zgarishi kerak. Rekursiv muroja-atning bajarilishi ana shu o‘zgaruvchiga bog‘liq bo‘ladi.

Sonning faktorialini hisoblovchi funksiyani ko‘ramiz. n— dastlabki ma’lumot ($n \geq 0$), $n!$ Faktorial — funksiya qiymatidir:

```
function f (n:integer) : integer;
begin
        if n > 0
                then f : = f (n - 1) * n
                else f : = 1
end.
```

Bu misolda har ikki zaruriy shart bajarilgan. Rekursiv funksiyaga murojaat qilish shartli operatorga kiradi. n —

o‘zgaruvchining qiymati har bir *f (n — 1)*ga murojaat qilishda o‘zgaradi, ya’ni funksiyaning yangi nusxasida parametr qiymatidan o‘zidan oldingi nusxadagiga qaraganda bitta kam bo‘ladi. Ma’lum vaqtdan keyin *n* parametrning qiymati nolga teng bo‘ladi va *f* funksiyaga keyingi murojaat qilish bajarilmaydi. Funksiya birinchi nusxaga qaytgandan keyin dastur o‘z ishini tamomlaydi.

Boshqa misol ko‘ramiz:

```
function ff (n:integer) : integer;
begin
  if n > 0
        then ff : = ff (n + 1) * n
        else ff : = 1
end.
```

ff(n) ga murojaat qilishda uchragan dastur *n>0* da chekli bo‘lmaydi.

Ba’zi hollarda rekursiv funksiya va protseduralarning to‘g‘riligiga ishonch hosil qilishga ularda yozilgan amallarning masala ta’rifiga mos kelishi zarurligi yordam beradi.

Shunday qilib, dasturni tekshirish mashaqqatli mehnat ekanligi ravshan bo‘ldi. Tajriba dasturning hammasi dastlabki ma’lumotlari mumkin bo‘lgan har qanday qiymatlarida to‘g‘riligini isbot qilishga qaraganda ba’zi nazorat qiymatlar bo‘yicha dastur natijalarining to‘g‘riligini tekshirish ancha soddaroq ekanligini ko‘rsatadi. Ammo dastlabki ma’lumotlarning cheklangan miqdordagi variantlarida ba’zi xato natijalarni oson o‘tkazib yuborish mumkin. Bundan tashqari, nazorat qiymatlarni tanlashda ham ma’lum qiyinchiliklar tug‘iladi.

Dasturni tekshirishning quyidagi tartibini tavsiya qilish mumkin:

1. Dastlabki ma’lumotlarning bitta yoki ikkita tanlab olingan variantlarida dasturning amallarini qo‘lda bajarib, natijaviy qiymatlarni olish.

2. Tekshiriladigan dasturda sikllar, rekursiv funksiya hamda protseduralarning chekligini ko‘rsatish.

3. Dastlabki ma’lumotlarning mumkin bo‘lgan har qanday

qiymatlarida dasturning to‘g‘riligini isbotlashga harakat qilish. Agar buni bajarish ancha murakkab bo‘lsa, unda bu dasturga xos bir nechta dastlabki ma’lumotlarni tanlab olish va shu qiymatlar bilan dasturdagi amallarni bajarish (ya’ni birinchi bosqichni nazorat qiymatlarning juda ko‘p variantlarida takrorlash) kerak.

13.6. Dasturni takomillashtirish

Har doim ham birdaniga ixcham va tejamli dastur yaratishga muvofiq bo‘linavermaydi. Ba’zida dastur tuzilgan, tekshirilgan bo‘lsa-da, dasturchida dasturni takomillashtirishning yangi fikrlari (qisqaroq, aniqroq, lo‘ndaroq yoki tejamliroq qilish) tug‘ilib qoladi. Sonning mutlaq qiymatini aniqlovchi shartli operatorni misol tariqasida ko‘rib chiqamiz:

```
if a≥0
        then b : = 0
        else
              if a < 0
                    then b : = -a
```

Bu shartli operatorning ikkinchi qismida ortiqcha shartli operator yozilganini sezish qiyin emas, chunki $a<0$ shartni ikki marta tekshirish sodir bo‘ladi.

Shartli operatorni qisqaroq va yaqqolroq yozamiz:

```
if a≥0
    then b : = 0
    else b : = -a
```

Boshqa misol ko‘ramiz:

```
for k: = 1 to 100 do
    write (a + b)
```

Siklning har bir bajarilishida *a* va *b* o‘zgaruvchilar qiymatlarining yig‘indisi hisoblanadi va bosib chiqariladi. Yig‘indini hisoblashni sikldan tashqariga chiqaramiz:

```
c: = a + b;
for k: = 1 to 100 do
        write (c)
```

Endi o‘zgaruvchilarning qiymatlarini qo‘shish faqat bir marta bajariladi. Demak, bu dasturni bajarishda mashina vaqti qisqaradi.

Kabisa yillarni aniqlovchi funksiyadagi shartli operatorni soddalashtirishga harakat qilamiz:

```
if yil mod 100 = 0
  then
        if y mod 400 = 0
              then kabisa:=true
              else kabisa:=false
        else
              if y mod 4=0
                    then kabisa:=true
                    else kabisa:=false
```

Agar berilgan son 400 ga bo‘linsa, bunday yil kabisa yili bo‘ladi, shuning uchun oldin berilgan sonni 400 ga bo‘linishini tekshirish, keyin esa qolgan hollarni ko‘rib chiqish kerak:

```
              if y mod 400 = 0
                    then kabisa:=true
                    else . . . . . .
```

Agar son 400 ga bo‘linmasdan 100 ga bo‘linsa, bu son asrning oddiy yilini belgilaydi. Shartli operatorni yozishni davom ettiramiz:

```
  if y mod 400 = 0
        then kabisa: = true
```

```
else
        if y mod 100=0
                then kabisa:=false
                else
                .....
```

Qolgan yillar sonining 4 ga bo‘linishini tekshiramiz va shartli operatorning oxirini yozamiz:

```
if y mod 400 = 0
        then kabisa:=true
        else
                if y kabisa 100 = 0
                        then kabisa:=false
                        else
                                if y mod 4 = 0
                                        then kabisa:=true
                                        else kabisa:=false
```

Shunday qilib, mulohazamiz natijasida shartli operator olindi.

13.7. Dasturni tuzatish

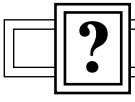
Agar dastur yozilgan, sinchiklab tekshirilgan bo‘lsa, uni mashinaga kiritish vaqti keldi. Endi dasturni EHM yordamida tuzatish zarur, chunki puxta tekshirilgan dasturda ham xatoliklar uchrashi mumkin. Xatolar mazmuniy (semantik), ko‘p uchraydigan imloviy (grammatik), shuningdek, sintaktik (zarur belgilar yoki so‘zlarni noto‘g‘ri ishlatish) xatolarga ajratiladi. Sintaktik xatolar dastur matnini kiritishda ham kelib chiqadi.

Sintaktik xatolarni topib, EHM uning dasturdagi joyini dasturchiga xabar qiladi. Mazmuniy xatoni aniqlash ancha murakkab. Buning uchun, dasturning to‘g‘riligini qo‘lda tekshirishdagiga o‘xshash, tanlab olingan nazorat qiymatlarda tajriba o‘tkaziladi. Xatolarni EHMning o‘zi qidirayotganligi uchun, endi

ancha «murakkab» ma'lumotlar bilan ish olib borish, sikllarning ko'p marta takrorlanishidan foydalanish mumkin.

Hamma mazmuniy xatolar ham tez va oson topilavermaydi. Murakkab dasturda chuqur yashiringan xatolar uchrashi mumkin. Ularni topish uchun dasturchi sezgiga, mantiqan fikrlashga, yaxshi bilimga ega bo'lishi kerak.

Shunday qilib, dasturni tayyorlash va tuzatish dasturchidan anchagina mehnat talab qilar ekan. Biroq, dastur yaratilgandan keyin u bilan ko'p foydalanuvchilar ishlashi mumkin. Zarur bo'lganda dastur matnini ko'chirish va turli xildagi EHMlar xotirasiga kiritish mumkin.



Nazorat savollari

1. Dasturni yaratish jarayoniga qanday bosqichlar kiradi?
2. Masalani ta'riflash bosqichining mohiyati?
3. Yechish usuli (algoritmni tanlash va tasvirlash bosqichi)ni bayon eting.
4. Dasturni yaxshilash bosqichida nima ishlar qilinadi?
5. Dasturni tekshirish bosqichi qanday bosqich?
6. Dasturni takomillashtirish bosqichining vazifasi nima?
7. Dasturda qanday xatoliklar bo'lishi mumkin, ular qanday aniqlanadi?
8. Sintaktik va imloviy xatoliklar qanday aniqlanadi?
9. Mazmuniy (semantik) xatoliklar qanday aniqlanadi?
10. Xatolarni aniqlash uchun dasturchi nimalarga ega bo'lishi kerak?

XIV bob. KATTA DASTURLAR MAJMUALARINI ISHLAB CHIQISH USLUBIYATLARI

14.1. Tasnifli (strukturali) dasturlash

Dasturlashning rivojlanishi 70-yillarning boshiga kelib, yetakchi dasturchilar oldiga dastur o‘lchami va miqdorining boshqarib bo‘lmas darajada o‘sayotganligi natijasida katta dasturlar majmualarini ishlab chiqishga yondashishni tubdan o‘zgartirish vazifasini qo‘ydi. Bu muammolardan kelib chiqib, 70-yillarning eng yaxshi dasturchilari tomonidan (Deykstr, Virt, Dal, Xoar, Yordan, Konstantin Mayer va boshqalar) loyihalarni yuritishning tasnifli uslubiyati, deb nomlangan qat’iy qoidalar ishlab chiqildi.

Bu uslubiyatga ko‘ra, dastur mavhum darajali pog‘onalar to‘plamidan iborat, deb qaraladi. Bu pog‘onalar dasturni tushunishni yengillashtirishi, dastur bajarilish ishonchliligini oshirishi, dastur ishlab chiqish muddatlarini qisqartirishga yordam berishi kerak.

Tasnifli dasturlashning asosiy maqsadlari quyidagilardan iborat:

1. Dastur o‘qilishini yaxshilash.

Buning uchun quyidagi qoidalarga rioya qilish kerak:

— til konstruksiyasi semantikasini murakkablashtirmaslik;

— boshqaruvchi konstruksiyalar va ma’lumotlar tasnifidan foydalanish ishlarini kichraytirishga intilish;

— dasturni boshidan oxirigacha boshqa betga boshqaruvchi o‘tishlarsiz o‘qilishga erishish darajasida ishlab chiqish.

2. Dastur samaradorligini oshirish.

Bu maqsadda dastur tarkibi modullarga shunday bo‘linishi kerakki, natijada xatolarni oson topish va tuzatish, shuningdek, ixtiyoriy modul matnini, samaradorlikni oshirish maqsadida, boshqalariga bog‘liq bo‘lmagan holda o‘zgartirish mumkin bo‘lsin.

3. Dastur ishonchliligini oshirish.

Dastur boshidan oxirigacha osongina testdan o‘tkazilsa, dasturni yaxshilash jarayonida muammolar tug‘ilmasa, dastur ishi yuqori ishonchli bo‘ladi. Yuqori ishonchlilik esa dasturni modullarga bo‘lishda yaxshi tasniflanishi bilan ta’minlanadi.

4. Dastur mahsulotini ishlab chiqish vaqti va qiymatini kamaytirish.

Agar dasturchilar guruhidagi har bir dasturchi oldingiga qaraganda dastur kodlarining katta sonini yozish va yaxshilash qobiliyatiga ega bo‘lsa, ya’ni dasturchi ishining mehnat unumdorligi oshsa, tasnifli dasturlash qoidalariga rioya qilsa, bu maqsadga erishish mumkin.

Tasnifli uslubiyatning asosiy tamoyillari

1. Mavhumlik tamoyili.

Mavhumlik dasturchiga talab qilinayotgan muammoni shu daqiqada ko‘p tomonlarini hisobga olmasdan turib fikrlashga imkon beradi. Bu tamoyildan foydalanib turib, dasturchi dasturni pog‘onalar bo‘yicha qarab chiqa oladi. Yuqori pog‘ona katta mavhumlikni ko‘rsatib, loyihaga qarashni soddalashtirsa, shu vaqtning o‘zida quyi pog‘ona dasturni amalga oshirishning kichik bo‘laklarini ko‘rsatadi.

2. Rasmiylik tamoyili.

«Rasmiylik» so‘zi qat’iy uslubiy yondashishni nazarda tutadi.

Rasmiylikni qandaydir jarayonga (texnik yoki notexnik) kiritish, ijodni bo‘g‘adi, degan qarashlarga olib keladi. Lekin tasnifli yondashish rasmiylikdan ijodiy jarayonga ma’lum intizom, qat’iylikni berish uchun foydalanadi. Bu esa yechim qabul qilishni tezlashtiradi va ko‘p xatolardan chetlab o‘tishga imkon beradi.

3. «Bo‘l va hukmronlik qil» tamoyili.

Bu tamoyil dasturni boshqarish uchun oddiy va mustaqil yaxshilash hamda testdan o‘tkazish imkoniga ega bo‘lgan alohida

bo‘laklarga (modullarga) bo‘lishni bildiradi. U dasturchiga, butun tizimni egallovchi ulkan miqdordagi bo‘laklar haqida o‘ylamasdan turib, tizimning alohida qismlari bilan ish olib borish imkonini beradi.

4. Pog‘onali boshqarish tamoyili.

Bu «bo‘l va hukmronlik qil» tamoyili bilan chambarchas bog‘liq. U dastur majmuasidagi modullar o‘rtasidagi o‘zaro aloqa bog‘liqliklarini, yuqorida ko‘rib o‘tilgan strukturali dasturlash maqsadlariga erishishni yengillashtiruvchi, pog‘onali tasniflash talabini ilgari suradi.

14.2. Modulli dasturlash

Modulli dasturlashning asosiy tamoyili «bo‘l va hukmronlik qil». Modulli dasturlash dasturni modul deb ataluvchi, tarkibi va ishlatilishi ma’lum qoidalarga bo‘ysunuvchi, o‘zaro bog‘liq bo‘lmagan, kattamas bo‘laklarga bo‘lib hosil qilishni bildiradi.

«Modul» so‘zi qaysi vaqtda dasturlash tillarining sintaksis konstruksiyasi (*Turbo Paskal*da *unit*) sifatida va qaysi vaqtda katta dasturni alohida bo‘laklarga bo‘lish birligi (protsedura yoki funksiyalar ko‘rinishida amalga oshirish nazarda tutilyapti) sifatida ishlatilayotganligini farqlay bilish kerak.

Modulli dasturlashdan foydalanish dasturni testdan o‘tkazish va xatolarni topishni yengillashtiradi.

14.3. Tasniflash va dasturlash standartlari

1. Dastur modullar deb ataluvchi, mustaqil qismlarga bo‘linishi kerak.
2. Modul — mustaqil bo‘lak, uning kodi fizik va mantiqiy nuqtayi nazardan boshqa modullar kodlaridan farq qiladi.
3. Modul faqat bitta mantiqiy ishni bajaradi.
4. Modul o‘lchami 100 ta operatordan oshmasligi kerak.
5. Modul bitta kiruvchi va bitta chiquvchi nuqtaga ega.
6. Modullar o‘rtasidagi o‘zaro bog‘liqlik pog‘onali tasnif orqali amalga oshiriladi.

7. Har bir modul, modulning ishlatilishi, undagi o‘zgaruvchilarning unga uzatiladigan va undan chiqadigan modullarga hamda uni chaqiruvchi modullarga tushuntirish beruvchi izohlar bilan boshlanishi kerak.
8. Keraksiz nishonlardan iloji boricha foydalanmaslik va GOTO operatoridan faqat modulning boshlang‘ich yoki oxirgi nuqtasiga o‘tishda foydalanish yoxud umuman foydalanmaslik kerak.
9. Hamma o‘zgaruvchilar va modullarning nomlari (identifikatorlari) ma’noli bo‘lishi kerak.
10. Identifikatorlarning qarindosh guruhlari bir xil prefiks bilan boshlanishi kerak.
11. Faqat standart boshqaruvchi konstruksiyalardan (tanlov, sikl, chiqish, blok) foydalanish kerak.
12. Bitta satrda bittadan ortiq operator yozmaslik kerak. Agar bitta satrni yozish uchun bittadan ortiq satr kerak bo‘lsa, keyingi satrlar joy tashlash bilan yoziladi.
13. *if* operatorlarining 3 dan ortiq ichma-ich ishlatilishiga yo‘l qo‘ymaslik kerak.
14. G‘ayritabiiy semantik til konstruksiyalaridan foydalanishdan qochish kerak.

14.4. Obyektga yo‘naltirilgan dasturlash uslubiyati

U yoki bu dasturlash tili asosida tegishli dasturlar xususiyatiga sezilarli ta’sir etuvchi qandaydir yetakchi g‘oya yotadi.

Dasturlarni protsedurali tasniflash g‘oyasi tarixan birinchi shunday g‘oya bo‘ldiki, unga ko‘ra dasturchi o‘z dasturiga qanday protseduralarni kiritishni hal qilishi va shundan keyin bu protseduralarni hal qilish uchun eng yaxshi algoritmlarni tanlab olishi kerak. Bu g‘oyaning paydo bo‘lishi hisoblash jarayonlarining algoritmik tomonlarini yetarli darajada o‘rganmaslik natijasi bo‘ldi. G‘oya kamchiliklari o‘tgan asrning 40—50-yillarida ishlab chiqilgan dastlabki dastur mahsulotlariga xos. Protsedurali-yo‘naltirilgan tilga fortran misol bo‘la oladi (bu til birinchi va hali ham eng keng tarqalgan dasturlash tillaridan biri). Dasturlarni protsedurali tasniflash g‘oyasining keyingi ishlatilishlari, dastur «binosi»ni qurishda g‘isht vazifasini o‘tovchi, nisbatan katta bo‘lmagan protseduralar to‘plamiga ega bo‘lgan keng dasturlash kutubxonasini yaratishga olib keldi.

Hisoblash matematikasi sohasidagi yutuqlar natijasida, urg‘u dasturlashda protseduralardan ma’lumotlarni tashkil qilishga berila boshlandi. Murakkab dasturlarni samarali ishlab chiqish ma’lumotlarni to‘g‘ri ishlatishning nazorat usullariga muhtoj ekanligi ma’lum bo‘ldi. Nazorat kompilatsiya bosqichida ham, dasturni o‘tkazish bosqichida ham o‘tkazilishi kerak, aks holda, amaliyotda isbot qilingani kabi, yirik dastur loyihalarini yaratishda qiyinchiliklar keskin oshadi. Bu muammoning oydinlashuvi ma’lumot turlarining ozmi-ko‘pmi rivojlangan tasnifiga ega bo‘lgan *Algol 60,* keyinchalik *Paskal, Modula, Si* va boshqa ko‘plab dasturlash tillarning yaratilishiga olib keldi. Ma’lumotlarni va protseduralarni modul ichiga «yashirish»ga intilish bilan strukturali dasturlar ishlab chiqishga modulli yondashish, bu yo‘nalish rivojlanishining mantiqiy natijasi bo‘ldi.

Simula-67 tilidan boshlab, dasturlashda obyekt—yo‘naltirilgan dasturlash, deb ataluvchi yangi yondashish sezila boshlandi. Undagi yetakchi g‘oya, ma’lumotlarni, ularni ishlab chiquvchi protseduralar bilan bir butunga — obyektga keltirishga intilishdir. Obyektlarning e’tiborli tomoni ma’lumotlarni va ularni ishlab chiquvchi algoritmlarni birlashtirish (inkapsulatsiya)dir, buning natijasida ma’lumotlar va protseduralar ko‘p tomondan o‘zining mustaqil ma’nosini yo‘qotadi. Aslida obyekt yo‘naltirilgan dasturlashga, protsedura va ma’lumotlarni tasodifiy va mexanik birlashtirish o‘rniga, ularning ma’noli bog‘lanishiga e’tibor berayotgan yangi pog‘onadagi modulli dasturlash, deb qarash mumkin.

Obyekt—yo‘naltirilgan dasturlashning qanday qudratli vositalarga ega ekanligini *Turbo Paskal* majmuasiga kiruvchi *Turbo Vision* kutubxonasi ko‘rgazmali namoyish etadi.

Obyekt—yo‘naltirilgan dasturlashning to‘liq ma’nodagi afzalliklari yetarli darajada murakkab bo‘lgan dasturlar ishlab chiqishda namoyon bo‘lishini aytib o‘tish kerak. Bundan tashqari, inkapsulatsiya obyektlarga o‘ziga xos «mustaqillik», dastur boshqa qismlariga maksimal darajada bog‘liq bo‘lmaslik xususiyatini beradi. To‘g‘ri tashkil etilgan obyekt, undan talab etilgan ishlarni muvaffaqiyatli amalga oshirish uchun, barcha zaruriy ma’lumotlar va ularni qayta ishlash protseduralariga ega bo‘ladi. Obyektga yo‘naltirilgan dasturlashni tayyor formulalar bo‘yicha hisob ishlari bilan bog‘liq bo‘lgan murakkab bo‘lmagan algoritmlarni dasturlashga qo‘llashga urinish, ko‘p hollarda, sun’iy ravishda keraksiz til konstruksiyalari

bilan dasturni to‘ldirishga olib keladi. Bunday dasturlar, odatda, algoritmni bir qator bir-biri bilan nisbatan bog‘liq bo‘lmagan qismlarga bo‘lib tashlashga muhtoj emas, ularni Paskalning an’anaviy usullar bilan ishlab chiqish osonroq.

Murakkab muloqot dasturlarini ishlab chiqishda dasturchi dasturni tasniflashi kerak. Chunki faqat shu holdagina u muvaffaqiyatga umid qilishi mumkin: tasniflanmagan dasturlarning «mezon salmog‘i» katta hajmli jarayonda, 1000—1200 satrli boshlang‘ich matn hajmidadir, odatda, beqiyos qiyinchiliklarga duch kelinadi. Tasnifli dasturlar, dasturlashning xususiy kutubxonasini ishlab chiqishga olib keladi. Mana shu vaqtda obyektga yo‘naltirilgan dasturning yangi vositalari yordamga keladi.

Obyektga yo‘naltirilgan dastur obyektlarga yaxshi xossa beruvchi uch muhim tamoyilga asoslanadi. Bu tamoyillarga inkapsulatsiya, meros olish va polimorfizm kiradi.

Inkapsulatsiya ma’lumotlar va ularni qayta ishlovchi algoritmlarni bir butunga birlashtirishdir. Obyektga yo‘naltirilgan dastur doirasida ma’lumotlar *obyekt maydonlari* deb, algoritmlar esa *obyekt usullari*, deb ataladi.

Inkapsulatsiya obyektni tashqi muhitdan maksimal darajada ajratishga imkon beradi. U ishlab chiqilayotgan dasturlar ishonchliligini sezilarli oshiradi, chunki obyektda kichraytirilgan algoritmlar dastur bilan ma’lumotlarning nisbatan katta bo‘lmagan hajmi bilan almashadi, bunda bu ma’lumotlarning soni va turi, odatda, yaxshilab nazorat qilinadi. Obyektga birlashtirilgan algoritmlar va ma’lumot-larning almashtirilishi yoki modifikatsiyalanishi bir butun dastur uchun, odatda, yomon kuzatiladigan oqibatlarga olib kelmaydi (dasturlar himoyasini ko‘tarish maqsadida obyektga yo‘naltirilgan dasturda global o‘zgaruvchilar deyarli ishlatilmaydi).

Meros olish obyektlarning o‘z avlodlarini tug‘dirish xususiyati-dir. Obyekt — avlod yaratuvchisidan hamma maydon va usullarni avtomatik tarzda meros oladi, u obyektlarni yangi maydonlar bilan boyitishi va yaratuvchi usullarini almashtirishi (qoplashi) yoki ularni to‘ldirishi mumkin.

Meros olish tamoyili obyekt xususiyatlari modifikatsiyasi muammosini hal qiladi. Obyektlar bilan ishlashda dasturchi, odatda,

o‘z xususiyatlariga ko‘ra aynan bir masalani yechishga juda yaqin bo‘lgan obyektni tanlaydi va undan, yaratuvchilarda amalga oshirilmagan ishlarni qila oladigan, bir yoki bir nechta avlodlarni yaratadi.

Polimorfizm tamoyili qarindosh (ya’ni bitta umumiy yaratuvchiga ega) obyektlarning ma’nosiga ko‘ra o‘xshash muammolarini har xil usullar bilan yechishni bildiradi. Obyektga yo‘naltirilgan dastur doirasida obyektning tabiiy xususiyatlari unga kiruvchi usullar to‘plami bilan aniqlanadi. Obyekt avlodida u yoki bu algoritmni o‘zgartira borib, dasturchi bu avlodlarga yaratuvchisida bo‘lmagan o‘ziga xos xususiyatlarni berishi mumkin. Usulni o‘zgartirish uchun uni avlod uchun yopish, ya’ni avlodda bir ismli usulni e’lon qilish va unda kerakli ishlarni amalga oshirish zarur. Natijada, yaratish-obyektida va avlod-obyektida har xil algoritmik asosga ega va, demak, obyektlarga har xil xususiyatlar beruvchi, bir xil ismli ikki usul ish ko‘rsatadi. Mana shu *obyektlarning polimorfizmi* deyiladi.

Nazorat savollari

1. Strukturali dasturlash qanday dasturlash?
2. Strukturali dasturlashning asosiy maqsadi nimalardan iborat?
3. Strukturali uslubiyatning asosiy tamoyillariga nimalar kiradi?
4. Modulli dasturlash nima?
5. Strukturali dasturlash standartlariga nimalar kiradi?
6. Obyekt-yo‘naltirilgan dasturlashning asosiy g‘oyasi nima?
7. Obyekt-yo‘naltirilgan dasturlash qanday tamoyillarga asoslanadi?
8. Inkapsulatsiya nima?
9. «Meros olish» tamoyilining mohiyati.
10. Polimorfizm deganda nimani tushunasiz?

FOYDALANILGAN ADABIYOTLAR

М. Зелковиц, А. Шоу, Дж. Гэннон. Принципы разработки программного обеспечения. Пер. с англ. М., «Мир», 1982.

Дж. Фокс. Программное обеспечение и его разработка. Пер. с англ. М., «Мир», 1985.

Практическое руководство по программированию. Пер. с англ. Под ред. Б.Мик, П.Хит, Н.Рашби. М., «Радио и связь», 1986.

А.Л. Брудно, Л.И. Каплан. Московские олимпиады по программированию. 2-е издание. М., «Наука», 1990.

Н. Вирт. Алгоритмы и структуры данных. Пер. с англ. М., «Мир», 1989.

G. Grigas. Dasturlash asoslari. T., «O'qituvchi», 1990.

В.В. Липаев. Проектирование программных средств. М., 1990.

В.Ф. Очков, Ю.Ю. Пухначев. 128 советов начинающему программисту. М., 1991.

А.В. Файсман. Профессиональное программирование на Турбо Паскале. Т., Информ Экс — Корпорейшн, 1992.

Н.Б. Культин. Программирование в Turbo Pascal 7.0. и Delphi/ Второе издание, переработанное и дополненное. СПб., БХВ — Санкт-Петербург, 1999.

Н.Б. Культин. Программирование на Object Pascal в Delphi 5. СПб., БХВ — Санкт-Петербург, 1999.

А.Н. Марченко. Программирование в среде Turbo Pascal 7.0. К., Век+, М., «ДЕСС», 1999.

В.В. Фаронов. Turbo Pascal 7.0. М., «Нолидж», 2000.

Н.Б. Культин. Turbo Pascal в задачах и примерах. СПб., БХВ — Санкт-Петербург, 2001.

MUNDARIJA

VI bob. ***TURBO PASKAL*** **MA'LUMOTLARINING ODDIY TURLARI**

VII bob. ***TURBO PASKAL*** **MUHITIDA ISHLASH**

VIII bob. **TIL OPERATORLARI**

IX bob. **STATIK MA'LUMOTLARNING MURAKKAB TURLARI**

X bob. **PROTSEDURA VA FUNKSIYALAR**

XI bob. **FAYLLAR**

XII bob. ***TURBO PASKAL*NING GRAFIK REJIMI**

XIII bob. **DASTURLASH USLUBIYATI**

XIV bob. **KATTA DASTURLAR MAJMUALARINI ISHLAB CHIQISH USLUBIYATLARI**

R18 **Razzoqov Sh. I., Yunusova M. J. Dasturlash.** Kasb-hunar kollejlari uchun o‘quv qo‘llanma (9-nashri). T.: «ILM ZIYO», 2017. — 232 b.

UO‘K: 004. 438 (075)
KBK 32.973-018

ISBN 978-9943-16-299-0

SHAVKAT INSANOVICH RAZZOQOV,
MUSLIMA JAMOLOVNA YUNUSOVA

DASTURLASH

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

9-nashri

Toshkent — «ILM ZIYO» — 2017

Muharrir *I. Usmonov*
Badiiy muharrir *M. Burxonov*
Texnik muharrir *F. Samadov*
Musahhih *F. Temirxo‘jayeva*

Noshirlik litsenziyasi AI № 275, 15.07.2015-y.

2017-yil 10-oktyabrda chop etishga ruxsat berildi. Bichimi $60\times90^{1}/_{16}$. «Tayms» harfida terilib, ofset usulida chop etildi. Bosma tabog‘i 14,5. Nashr tabog‘i 14,0. 300 nusxa. Buyurtma №727.

«ILM ZIYO» nashriyot uyi. 100129, Toshkent, Navoiy ko‘chasi, 30-uy.

«NISO POLIGRAF» MCHJ bosmaxonasida chop etildi.
Toshkent viloyati, O‘rta Chirchiq tumani, «Oq-Ota» QFY
Mash’al mahallasi Markaziy ko‘chasi, 1-uy.