O`ZBEKISTON RESPUBLIKASI

OLIY VA O`RTA MAXSUS TA'LIM VAZIRLIGI

QARSHI DAVLAT UNIVERSITETI

O.M. Shukurov, E.A.Eshboyev, B.H.Shovaliev

DELPHI VA C++ ALGORITMIK TILLARIDA DASTURLASH

(O’quv-uslubiy qo‘llanma**)**

Qarshi-2012

Bu metodik qo‘llanma dasturlashni o‘rganish uchun zarur bo‘lgan asosiy tushuncha va masalalarni o‘z ichiga oladi.

Mazkur metodik qo‘llanmadan oliy o‘quv yurtlarining matematika, amaliy matematika va informatika, matematika va informatika, informatika va informatsion texnologiyalar bakalavrlik yo‘nalishlarida tahsil olayotgan talabalar uchun mo‘ljallangan.

Taqrizchilar: dots. O. Turg’unov

dots. G’. Karimov

Tuzuvchilar: prof. O.Shukurov

katta o‘qit. E. Eshboyev

katta o‘qit. B. Shovaliyev

SO‘Z BOSHI

Biz yashayotgan asr shak-shubhasiz, «Informatsiya» bilan va uni avtomatik ravishda qayta ishlash imkonini beruvchi informatsion va kompyuter texnologiyalarining jadal sur’atlarda rivojlanishi bilan xarakterlanadi. Bu davrni bejiz, informatsion shov-shuvlar asri deb atashmayapti[1]. Bu fikrning tasdig‘i sifatida Internetni yodga olishning o‘zi kifoya bo‘lsa kerak. Adabiyotlarda «Modda», «Energiya» va «Informatsiya» moddiy olamning asosiy tashkil etuvchisi deb e‘tirof etilmoqda. Sivilizatsiyaning hozirgi kundagi rivoji, insoniyatning industrial jamiyatdan Informatsion jamiyat sari intilmoqda deyishga asos bo‘la oladi.

Elektron hisoblash mashinasining yaratilishi ham, insoniyatning rivojlanish tarixida erishilgan eng yuksak kashfiyotlar sirasiga kiradi. Hozirgi kunga kelib kompyuter va global tarmoq, butun dunyo bo‘yicha yig‘ilgan misli ko‘rilmagan katta hajmdagi informatsiyalarni insoniyat tomonidan foydalanilishiga imkon beradigan va uning intelektual imkoniyatlarini yuqori darajalarga ko‘taruvchi juda ham kuchli vositaga aylandi.

Dastur mahsulotlarining va texnik vositalarning jadal sur’atlar bilan rivojlanishi, kompyuterlarning apparat va dastur ta‘minotlarini tez ma‘naviy eskirishiga olib kelmoqda. Hali foydalanuvchi yangi dastur imkoniyatlarini to‘liq o‘zlashtirmasdan turib, sotuvga bu dasturning yanada mukammal variantlari taklif etilmoqda. Shunga qaramasdan kompyuterlar yaratishning va dasturlash texnologiyalarining asosiy tamoyil va g‘oyalari o‘z kuchida qolmoqda.

Hozirgi kunda dastur ta‘minotlari orasida Microsoft firmasi tomonidan yaratilgan operatsion sistemalar (Windows, Windows NT, Windows XP), matn muharrirlari (Word), Elektron jadvallar (Excel), berilganlar bazasi (Access) kabi ilovalar va dasturlash tillari hamda internetda ishlashning turli vositalari keng tarqalgan.

Windows muhitida foydalanuvchi interfeysini standartlashuvi foydalanuvchilarning kompyuter bilan muloqotini soddalashtirdi, ularni har bir yangi dastur paydo bo‘lganda yana qayta o‘rganishdek zerikarli ishdan ozod qildi.

Informatsion texnologiyalarning yana bir muhim jihatlaridan biri shundaki, bu fan jadal sur’atlarda o‘sib, yil sayin yangidan-yangi yo‘nalishlarga, mutaxassisliklarga tarmoqlanib ketmoqda: algoritmik, mantiqiy, ob‘ektga yo‘naltirilgan, vizual, parallel dasturlash texnologiyalari; animatsiya, multimediya, internet, berilganlar bazasi yo’nalishlari; ko‘p protsessorli, neyron arxitekturali kompyuterlar va hokazo. Ko‘rinib turibdiki, informatika meta fan darajasiga ko‘tarilib, uni bitta o‘quv kursi chegarasida to‘liq o‘zlashtirishning imkoni bo‘lmay qoldi.

Informatsion texnologiyalar sohasi bo‘yicha rus va ingliz tillarida qo‘llanmalar juda ko‘p chop etilmoqda. Oxirgi yillarda o‘zbek tilidagi qo‘llanmalar ham ko‘payib qoldi.

Ushbu taklif etilayotgan qo‘llanma asosan Delphi va C++ tillarini o‘rganmoqchi bo‘lganlar uchun mo‘ljallangan. Shu sababli kitobda Delphi va C++ tillariga bog’liq boshlang’ich ma’lumotlar yoritilgan. Bu qo’llanmadan Delphi va C++ dasturlash tillarini o‘rganuvchilar, dastur tuzishni o‘rganayotganlar hamda “Dasturlash asoslari”, “Informatika va dasturlash” fanlaridan olingan

nazariy bilimlarni mustahkamlash uchun foydalanishlari hisobga olingan. Ushbu qo‘llanmaga kiritilgan ma’lumotlar dasturlashning bazaviy kursidagi deyarli barcha bo‘limlarini, ya’ni skalyar turlar va boshqaruv operatorlaridan tortib, ma’lumotlarning murakkab turlari kabilarni o‘z ichiga oladi.

Ushbu kitob haqidagi tanqidiy fikr va mulohazalar eea1974@ Rambler.ru va eea1974@mail.ru elektron manzillarida mamnuniyat ila qabul qilinadi.

I BOB. DASTURLASH TILLARINING BOSHLANG'ICH TUSHUNCHALARI

1.1. Delphi va C++ dasturlash tillarining elementlari

Hozirgi kunda juda ko‘p algoritmik tillar mavjud. Bular ichida Delphi va C++ dasturlash tillari universal tillar hisoblanib, boshqa tillarga qaraganda imkoniyatlari kengroqdir. So‘nggi yillarda Delphi va C++ dasturlash tillari juda takomillashib, tobora ommalashib bormoqda. Mazkur tillardagi vositalar zamonaviy kompyuter texnologiyasining hamma talablarini o‘z ichiga olgan va unda dastur tuzuvchi uchun ko‘pgina qulayliklar yaratilgan.

Bu tillarga juda ko‘plab yangiliklar kiritilgan bo‘lib, ularning imkoniyatlari yanada kengaytirilgan. Delphi va C++ dasturlash tillari ham boshqa dasturlash tillari kabi o‘z alfavitiga va belgilariga ega.

Tillarda mavjud alfavit va leksemalarga quyidagilar kiradi:

- katta va kichik lotin alfaviti harflari;
- raqamlar - 0,1,2,3,4,5,6,7,8,9;
- maxsus belgilar: " {} | [] () + - / % \ ; ' : ? <=>_ ! & ~ # ^ . *

Alfavit belgilaridan tilning leksemalari shakllantiriladi:

- identifikatorlar;
- kalit (xizmatchi yoki zahiralangan) so‘zlar;
- o‘zgarmaslar;
- amallar belgilanishlari;
- ajratuvchilar.

Bu tillarda tuzilgan dasturlarda izohlar istalgan joyda berilishi mumkin. Ular satriy va blokli ko‘rinishlarda bo‘ladi. Satriy izohlar uchun “//”, blokli izohlar uchun “/*”, “*/” belgilari ishlatiladi.

1.2.Dastur tuzilmasi

Delphi tilida dastur matni bosh va asosiy bo‘limdan tashkil topadi. Bosh bo‘lim dastur nomi va o‘zgaruvchilar, o‘zgarmaslar, massivlar, belgilar (metkalar), protseduralar va funksiyalarni tavsiflashdan iborat bo‘ladi. Asosiy bo‘lim dastur tanasi deyilib, unda dasturda bajariladigan kerakligi operatorlar ketma-ketligi beriladi va u Begin (boshlamoq) so‘zi bilan boshlanib End (tugash) so‘zi bilan tugaydi. Umumiy holda dastur tuzilmasi quyidagi ko‘rinishga ega:

```
Program < dastur nomi>;
Uses   <Foydalanadigan kutubxonalar (modullar) ro‘yxati>;
Label <Ishlatiladigan belgilar (metkalar) ro‘yxati>;
Const <Ishlatiladigan o‘zgarmaslarni aniqlash>;
Type  <Yangi turlarni aniqlash>;
Var    <O‘zgaruvchilarni e'lon qilish>;
<Protsedura va funksiyalarni aniqlash >;
Begin
<Bajariladigan operatorlar ketma-ketligi>;
End.
```

C++ tilidagi dastur ko‘rinishini quyidagi misol yordamida keltirib o‘tamiz.

```
# include <iostream.h>   // sarlavha faylni qo‘shish
int main ()              // bosh funksiya tavsifi
{                        // blok boshlanishi
```

```
cout << "Salom Olam! "\n"; //  satrni chop etish
return 0;                  // funksiya qaytaradigan qiymat
}                          // blok tugashi
```

Dastur bajarilishi natijasida ekranga "Salom Olam!" satri chop etiladi.

Dasturning 1-satrida #inc1ude.. preprotsessor direktivasi bo‘lib, dastur kodiga oqimli o‘qish/yozish funksiyalari va uning o‘zgaruvchilari e’loni joylashgan «iostream.h » sarlavha faylini qo‘shadi. Keyingi qatorlarda dasturning yagona, asosiy funksiyasi -main() funksiyasi tavsifi keltirilgan. Shuni qayd etish kerakki, C++ dastursida albatta main() funksiyasi bo‘lishi shart va dastur shu funksiyani bajarish bilan o‘z ishini boshlaydi.

Dastur tanasida konsol rejimida belgilar ketma-ketligini oqimga chiqarish amali qo‘llanilgan. Ma’lumotlarni standart oqimga (ekranga) chiqarish uchun quyidagi format ishlatilgan:

cout « <ifoda>;

Bu yerda <ifoda> sifatida o‘zgaruvchi yoki sintsksis:i to‘g‘ri yozilgan va qandaydir qiymat qabul qiluvchi til ifodasi kelishi mumkin (keyinchalik, burchak qavs ichiga olingan o‘zbekcha satr ostini til tarkibiga kirmaydigan tushuncha deb qabul qilish kerak).

1.3.Identifikatorlar va kalit so‘zlar.

Dasturlash tillarida identifikator tushunchasi mavjud bo‘lib, dasturda obyektlarni nomlash uchun ishlatiladi. O‘zgarmaslarni, o‘zgaruvchilarni, belgi (metka), protsedura va funksiyalarni belgilashda ishlatiladigan nom **identifikatorlar** deyiladi. Identifikatorlar lotin alfaviti harflaridan boshlanib, qolgan belgilari harf yoki raqamlar ketma-ketligidan tashkil topgan bo'lishi mumkin. Masalan: axc, alfa.

1.3.1.O’zgarmaslar.

Dasturlash tillarida dastur bajarilishi vaqtida qiymati o‘zgarmaydigan identifikatorlar **o‘zgarmaslar** deyiladi. O‘zgarmaslar beshta guruhga bo‘linadi – butun, haqiqiy (suzuvchi nuqtali), sanab o‘tiluvchi, belgi (literli) va satr («string», literli satr).

Ular Delphida dasturning bosh qismida **Const** kalit so‘zi bilan e’lon qilinib, unga aniq qiymat tenglashtiriladi.

Misol. Const aa1=2.27;
Pi=3.14;
radius=14;

C++ tilida esa o‘zgarmas (literal) – bu fiksirlangan sonni, satrni va belgini ifodalovchi leksema hisoblanadi.

Kompilyator o‘zgarmasni leksema sifatida aniqlaydi, unga xotiradan joy ajratadi, ko‘rinishi va qiymatiga (turiga) qarab mos guruhlarga bo‘ladi.

Butun o‘zgarmaslar: ular quyidagi formatlarda bo‘ladi

- o‘nlik son;
- sakkizlik son;
- o‘n oltilik son.

O‘nlik o‘zgarmas 0 raqamidan farqli raqamdan boshlanuvchi raqamlar ketma-ketligi va 0 hisoblanadi: 0; 123; 7987; 11.

Manfiy o‘zgarmas – bu ishorasiz o‘zgarmas bo‘lib, unga faqat ishorani o‘zgartirish amali qo‘llanilgan deb hisoblanadi.

Sakkizlik o‘zgarmas 0 raqamidan boshlanuvchi sakkizlik sanoq sistemasi (0,1,..,7) raqamlaridan tashkil topgan raqamlar ketma-ketligi:

023; 0777; 0.

O‘n oltilik o‘zgarmas 0x yoki 0X belgilaridan boshlanadigan o‘n oltilik sanoq sistemasi raqamlaridan iborat ketma-ketlik hisoblanadi:

0x1A; 0X9F2D; 0x23.

Harf belgilar ixtiyoriy registrlarda berilishi mumkin.

Kompilyator sonning qiymatiga qarab unga mos turni belgilaydi. Agar tilda belgilangan turlar dastur tuzuvchini qanoatlantirmasa, u oshkor ravishda turni ko‘rsatishi mumkin. Buning uchun butun o‘zgarmas raqamlari oxiriga, probelsiz *l* yoki L (1ong), u yoki U (unsigned) yoziladi. Zarur hollarda bitta o‘zgarmas uchun bu belgilarning ikkitasini ham ishlatish mumkin:

451*u*, 012*Ul*, 0xA2*L*.

Haqiqiy o‘zgarmaslar: Haqiqiy o‘zgarmaslar – suzuvchi nuqtali son bo‘lib, u ikki xil formatda berilishi mumkin:

- o‘nlik fiksirlangan nuqtali formatda. Bu ko‘rinishda son nuqta orqali ajratilgan butun va kasr qismlar ko‘rinishida bo‘ladi. Sonning butun yoki kasr qismi bo‘lmasligi mumkin, lekin nuqta albatta bo‘lishi kerak. Fiksirlangan nuqtali o‘zgarmaslarga misollar: 24.56; 13.0; 66.; .87;
- eksponensial shaklda haqiqiy o‘zgarmas 6 qismdan iborat bo‘ladi:

1) butun qismi (o‘nli butun son);

2) o‘nli kasr nuqta belgisi;

3) kasr qismi (o‘nlik ishorasiz o‘zgarmas);

4) eksponenta belgisi ‘e’ yoki ‘E’;

5) o‘n darajasi ko‘rsatkichi (musbat yoki manfiy ishorali o‘nli butun son);

6) qo‘shimcha belgisi (‘F’ yoki f , ‘L’ yoki ‘*l*’).

Eksponensial shakldagi o‘zgarmas sonlarga misollar:

1e2; 5e+3; .25e4; 31.4e-1.

Belgi o‘zgarmaslar: Belgi o‘zgarmaslar qo‘shtirnoq (‘,’-apostroflar) ichiga olingan alohida belgilardan tashkil topadi va u char kalit so‘zi bilan aniqlanadi. Bitta belgi o‘zgarmas uchun xotirada bir bayt joy ajratiladi va unda butun son ko‘rinishidagi belgining ASCII kodi joylashadi. Quyidagilar belgi o‘zgarmaslarga misol bo‘ladi:

‘e’, ‘@’ , ‘7’, ‘z’, ‘w’, ‘+’, ‘sh’, ‘*’ , ‘a’, ‘s’.

1.1-jadval. C++ tilida escape –belgilar jadvali

escape belgilari	Ichki kodi (16 son)	Nomi	Belgining nomlanishi va unga mos amal
\\	0x5C	\	Teskari yon chiziqni chop etish
\’	0x27	‘	Apostrofni chop etish
\”	0x22	“	Qo‘shtirnoqni chop etish
\?	0x3 Ғ	?	So‘roq belgisi
\a	0x07	be1	Tovush signalini berish
\b	0x08	Bs	Kursorni 1 belgi o‘rniga orqaga qaytarish
\f	0x0C	ff	Sahifani o‘tkazish
\n	0x0A	lf	Qatorni o‘tkazish
\r	0x0D	cr	Kursorni ayni qatorning boshiga qaytarish
\t	0x09	ht	Kursorni navbatdagi tabulyatsiya joyiga o‘tkazish
\v	0x0D	vt	Vertikal tabulyatsiya (pastga)
\000	000		Sakkizlik kodi
\xNN	0xNN		Belgi o‘n oltilik kodi bilan berilgan

Ayrim belgi o‘zgarmaslar ‘\’ belgisidan boshlanadi, bu belgi birinchidan, grafik ko‘rinishga ega bo‘lmagan o‘zgarmaslarni belgilaydi, ikkinchidan, maxsus vazifalar yuklangan belgilar – apostrof belgisi, savol belgisini (?), teskari yon chiziq belgisini (\) va ikkita qo‘shtirnoq belgisini (“) chop qilish uchun ishlatiladi. Undan tashqari, bu belgi orqali belgini ko‘rinishini emas, balki oshkor ravishda uning ASCII kodini sakkizlik yoki o‘n oltilik shaklda yozish mumkin. Bunday belgidan boshlangan belgilar escape ketma-ketliklar deyiladi (1.1-jadval).

Ko’p belgili o’zgarmaslar: C++ tilida qo‘shimcha ravishda **wide** harfli o‘zgarmaslar va ko‘p belgili o‘zgarmaslar aniqlangan.

Wide harfli o‘zgarmaslar turi milliy kodlarni belgilash uchun kiritilgan bo‘lib, u **wchar _t** kalit so‘zi bilan beriladi hamda xotiradan 2 bayt joy egallaydi. Bu o‘zgarmas L belgisidan boshlanadi:

L’\013\022’, L'cc’

Ko‘p belgili o‘zgarmas turi **int** bo‘lib, u to‘rtta belgidan iborat bo‘lishi mumkin:

‘abs’ , ‘ \001\002\003\004’ .

Satr o‘zgarmaslar: Ikkita qo‘shtirnoq (“,”) ichiga olingan belgilar ketma-ketligi satr o‘zgarmas deyiladi:

“bu satr o‘zgarmas va uning nomi string\n”

Satr ichida escape ketma-ketligi ham ishlatilishi mumkin, faqat bu ketma-ketlik apostrofsiz yoziladi.

Probel bilan ajratib yozilgan satrlar kompilyator tomonidan yagona satrga ulanadi (konkatenatsiya):

“satr–bu belgilar massivi” /* bu satr keyingi satrga qo‘shiladi */

“, uning turi char[]”;

Bu yozuvga

“satr –bu belgilar massivi, uning turi char[]”;

yozuvi bilan ekvivalent hisoblanadi.

Uzun satrni bir nechta qatorga yozish mumkin va buning uchun qator oxirida ‘\’ belgisi qo‘yiladi:

“Kompilyator har bir satr uchun kompyuter xotirasida \ satr uzunligiga teng sondagi baytlardagi alohida \ xotira ajratadi va bitta -0 qiymatli bayt qo‘shadi”;

Yuqoridagi uchta qatorda yozilgan satr keltirilgan. Teskari yon chiziq (‘\’) belgisi keyingi qatorda yozilgan belgilar ketma-ketligini yuqoridagi satrga qo‘shish kerakligini bildiradi. Agar qo‘shiladigan satr boshlanishida probellar bo‘lsa, ular ham satr tarkibiga kiradi.

Satr xotirada joylashganda uning oxiriga ‘\0’ (0 kodli belgi) qo‘shiladi va bu belgi satr tugaganligini bildiradi. Shu sababli satr uzunligi, uning «haqiqiy» qiymatidan bittaga ko‘p bo‘ladi.

Turlangan o‘zgarmaslar: Turlangan o‘zgarmaslar xuddi o‘zgaruvchilardek ishlatiladi va initsializatsiya qilingandan (boshlang‘ich qiymat berilgandan) keyin ularning qiymatini o‘zgartirib bo‘lmaydi

Turlangan o‘zgarmaslar **const** kalit so‘zi bilan e’lon qilinadi, undan keyin o‘zgarmas turi va albatta initsializatsiya qismi bo‘lishi kerak.

Misol tariqasida turlangan va literli o‘zgarmaslardan foydalangan holda radius berilganda aylana yuzasini hisoblaydigan dasturni keltiramiz.

```
#include <iosream.h>
int main ()
{
const double pi=3.1415;
const int radius=3;
double square=0;
```

```
square=pi*radius*radius;
cout<<square<<'\n';
return 0;
}
```

Dastur bosh funksiyasining boshlanishida ikkita – pi va radius o‘zgarmaslari e’lon qilingan. Aylana yuzasini aniqlovchi square o‘zgarmas deb e’lon qilinmagan, chunki u dastur bajarilishida o‘zgaradi. Aylana radiusini dastur ishlashida o‘zgartirish mo‘ljallanmagan, shu sababli u o‘zgarmas sifatida e’lon qilingan.

Sanab o‘tiluvchi tur: Ko‘p miqdordagi, mantiqan bog‘langan o‘zgarmaslardan foydalanganda sanab o‘tiluvchi turdan foydalanish ma’qul. Sanab o‘tiluvchi o‘zgarmaslar **enum** kalit so‘zi bilan aniqlanadi. Mazmuni bo‘yicha bu o‘zgarmaslar oddiy butun sonlardir. Sanab o‘tiluvchi o‘zgarmaslar C++ standarti bo‘yicha butun turdagi o‘zgarmaslar hisoblanadi. Har bir o‘zgarmasga (songa) mazmunli nom beriladi va bu identifikatorni dasturning boshqa joylarida nomlash uchun ishlatilishi mumkin emas. Sanab o‘tiluvchi tur quyidagi ko‘rinishga ega:

enum <Sanab o‘tiladigan tur nomi> { $<nom^1>$ =$<qiymat^1>$, $<nom^2>$ = $<qiymat^2>$,…

$<nom^n>$ =$<qiymat^n>$ } ;

bu yerda, enum – kalit so‘z (inglizcha enumerate – sanamoq); <Sanab o‘tiladigan tur nomi>- o‘zgarmaslar ro‘yxatining nomi; <nom> - butun qiymatli konstantalarning nomlari; $<qiymat^i>$ - shart bo‘lmagan initsializatsiya qiymati (ifoda).

Misol uchun hafta kunlari bilan bog‘liq masala yechishda hafta kunlari dush (dushanba), sesh (seshanba), chor (chorshanba), paysh (payshanba), juma (juma), shanba (shanba), yaksh (yakshanba) o‘zgarmaslarini ishlatish mumkin va ular sanab o‘tiluvchi tur yordamida bitta satrda yoziladi:

enum Hafta {dush, sesh, chor, paysh, juma, shanba,yaksh} ;

Sanab o‘tiluvchi o‘zgarmaslar quyidagi xossaga ega: agar o‘zgarmas qiymati ko‘rsatilmagan bo‘lsa, u oldingi o‘zgarmas qiymatidan bittaga ortiq bo‘ladi. Kelishuv bo‘yicha birinchi o‘zgarmas qiymati 0 bo‘ladi.

Initsializatsiya yordamida o‘zgarmas qiymatini o‘zgartirish mumkin:

enum Hafta {dush=8 ,sesh,chor=12 ,paysh=13 , juma=16, shanba, yaksh=20};

Bu e’londa sesh qiymati 9, shanba esa 17 ga teng bo‘ladi.

Sanab o‘tiluvchi o‘zgarmaslarning nomlari har xil bo‘lishi kerak, lekin ularning qiymatlari bir xil bo‘lishi mumkin:

enum{no1=0,toza=0,bir,ikki,juft=2 ,uch} ;

O‘zgarmasning qiymati ifoda ko‘rinishda berilishi mumkin, faqat ifodadagi nomlarning qiymatlari shu qadamdagacha aniqlangan bo‘lishi kerak:

enum { ikki=2 ,turt=ikki*2 } ;

O‘zgarmasni qiymatlari manfiy son bo‘lishi ham mumkin: enum {minus2=-2 minus1,nol,bir} ;

1.3.2.O’zgaruvchilar.

Dastur ishlashi mobaynida qiymatlari o‘zgarishi mumkin bo‘lgan identifikatorga **o‘zgaruvchilar** deyiladi.

Dasturlash tillarida dastur bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o‘zgaruvchilar va o‘zgarmaslardan foydalaniladi. O‘zgaruvchi-dastur obyekti bo‘lib, xotiradagi bir nechta yacheykalarni egallaydi va berilganlarni saqlash uchun xizmat qiladi. O‘zgaruvchi nomga, o‘lchamga va boshqa atributlarga – ko‘rinish sohasi, amal qilish vaqti va boshqa xususiyatlarga ega bo‘ladi. O‘zgaruvchilarni ishlatish uchun ular albatta e’lon qilinishi kerak. E’lon natijasida

o‘zgaruvchi uchun xotiradan qandaydir soha zahiralanadi, soha o‘lchami esa o‘zgaruvchining aniq turiga bog‘liq bo‘ladi. Shuni qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin.

O’zgaruvchilar Delphi tilida dasturning bosh qismida **Var** kalit so‘zi bilan e’lon qilinadi. Ularning nomi keltirilib, turi belgilaniladi. O‘zgaruvchilarning eng ko‘p ishlatiladigan turlari **butun, haqiqiy, belgili, qator** va **mantiqiydir.** Ular mos ravishda butun — **Integer,** haqiqiy — **Real,** belgili — **Char,** qator (matn) — **String** va mantiqiy — **Boolean** deb yoziladi.

Masalan:
```
Var  a, dl, alfa : Integer;
     cl21, df: Real;
     Etx, xx : Char;
     St,Sw: String;
     fl : Boolean;
```

Mantiqiy turga tegishli o‘zgaruvchilar faqat ikkita qiymat qabul qiladi: «True» (rost) va «False» (yolg‘on).

C++ tilida esa o‘zgaruvchi e’loni uning turini aniqlovchi kalit so‘zi bilan boshlanadi va ‘=’ belgisi orqali boshlang‘ich qiymat beriladi (shart emas). Bitta kalit so‘z bilan bir nechta o‘zgaruvchilarni e’lon qilish mumkin. Buning uchun o‘zgaruvchilar bir-biridan ‘,’ belgisi bilan ajratiladi. E’lonlar ‘;’ belgisi bilan tugaydi. O‘zgaruvchi nomi 255 belgidan oshmasligi kerak.

O‘zgaruvchilarni e’lon qilish dastur matnining istalgan joyida amalga oshirilishi mumkin.

1.3.3.Kalit so‘zlar

Dasturlash tillarida kalit so‘zlar mavjud bo‘lib ulardan boshqa maqsadlarda foydalanilmaydi. Quyida Delphi va C++ tillarining kalit so‘zlarini alfavit tartibida keltiramiz.

Delphi tilida: and, array, as, asm, begin, case, class, const, constructor, destructor, dispinterface, div, do, downto, else, end, except, exports, file, finalization, finally, for, function, goto, if, implementation, in, inherited, initialization, inline, interface, is, label, library, mod, nil, not, object, of, or, out, packed, procedure, program, property, raise, record, repeat, resourcestring, set, shl, shr, string, then, threadvar, to, try, type, unit, until, uses, var, while, with, xor.

C++ tilida: asm, auto, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, explicit, extern, float, for, friend, goto, if, inline, int, long, mutable, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, swith, template, this, throw, try, typedef, typename, union, unsigned, virtual, void, volatile, while.

Protsessor registrlarini belgilash uchun quyidagi so‘zlar ishlatiladi:

_AH, _AL, _AX, _EAX, _BN, _BL, _BX, _EVX, _CL, _CN, _CX, _ESX, _DN, _DL, _DX, _EDX, _CS, _ESR, EBP, _FS, __GS, _DI, _EDI, _SI, _ESI, __BP, SP, DS, _ES, SS, _FLAGS.

Bulardan tashqari «_» (ikkita tag chiziq) belgilaridan boshlangan identifikatorlar kutubxonalar uchun zahiralangan. Shu sababli ‘_’ va «_» belgilarni identifikatorning birinchi belgisi sifatida ishlatmagan ma’qul. Identifikator belgilari orasida bo’sh joy belgisi (probel) ishlatish mumkin emas, zarur bo‘lganda uning o‘rniga ‘_’ ishlatish mumkin.

Misol uchun: silindr_radiusi, aylana_diametri.

1.4.Ma'lumotlar turlari

Ma’lumotlar turlarini Delphi tilida umumiy holda ikkiga ajratish mumkin:

♦ standart turlar. Bu turlar oldindan Delphi tili tomonidan aniqlangan bo‘ladi;

♦ dasturchi tomonidan kiritiladigan (aniqlanadigan) turlar. Standart turlar tarkibiga quyidagilar kiradi: butun, haqiqiy, belgili (simvol), qator, mantiqiy, ko‘rsatgichli, variant.

Dasturchi turlarni dasturning **Var** bo‘limida o‘zgaruvchilarni tavsiflashda aniqlaydi yoki maxsus turlarni aniqlash uchun mumkin bo‘lgan - **Type** turlarni tavsiflash bo‘limida aniqlanadi.

Bu bo‘lim umumiy holda quyidagicha bo‘ladi.

Type

<tur nomi>=< turning tavsifi>;

Misol:

```
Type
    TColor=(Red, Blue, Black);
Var Colorl, Color2, Color3: TColor;
```

Type bo‘limida dasturchi tomonidan yangi Tcolor nomli tur kiritilmoqda va u Red, Blue, Black mumkin bo‘lgan qiymatlarni qabul qiladi.

Var bo‘limida dasturchi tomonidan turi aniqlangan uchta Colorl, Color2, Color3 o‘zgaruvchilar tavsiflanmoqda.

Bu o‘zgaruvchilarni to‘g‘ridan-to‘g‘ri quyidagicha ham tavsiflash mumkin.

```
Var Colorl, Color2, Color3: (Red, Blue, Black);
```

Standart turlarni Type bo‘limida tavsiflash shart emas, ularni to‘g‘ridan-to‘g‘ri Var bo‘limida tavsiflash mumkin.

Delphida standart turlarni quyidagicha klassifikatsiya qilish mumkin:

- ♦ Oddiy
  - * Tartibli
    - • Butun
    - • Belgi
    - • Mantiqiy
    - • Sanoqli
    - • Chegaralangan
  - * Haqiqiy
- ♦ Qator
- ♦ Tuzilma
  - * To‘plam
  - * Massiv
  - * Yozuv
  - * Fayl
  - * Klass
  - * Interfeys
- ♦ Ko‘rsatgichli
- ♦ Protsedurali
- ♦ Variant

Oddiy turlarga tartiblashgan va haqiqiy turlar kiradi. Tartiblashgan turlar shu bilan xarakterlanadiki, uning har bir qiymati o‘zining tartiblangan nomeriga ega. Haqiqiy tur qiymatlari kasr qismi mavjud bo‘lgan sonlardan iboratdir.

Tartiblashgan turlarga butun, belgili, mantiqiy, sanoqli va chegaralangan turlar kiradi.

Butun turlar. Butun turlar butun sonlarni tasvirlash uchun ishlatiladi.

Tur	O‘zgarish diapazoni	O‘lcham (baytda)
Integer	-2147483648..2147483647	4
Cardinal	0..4294967295	4

Shortint	-128..127	1
Smallint	-32768.32767	2
Longint	-2147483648..2147483647	4
Int64	$-2^{63}..2^{63}$-l	8
Byte	0..255	1
Word	0..65535	2
Long Word	0..4294967295	4

Haqiqiy turlar. Haqiqiy turlar haqiqiy sonlarni tasvirlash uchun ishlatiladi.

Tur	**O‘zgarish diapazoni**	O‘lcham (baytda)
Real	$5.0*10^{324}..1.7*10^{308}$	8
Real48	$2.9*10^{39}..1.7*10^{38}$	6
	1.5*10-45 J 4*1 Q38	4
Double	$5.0*10^{-324}..1.7*10^{308}$	8
Extended	$3.6*10^{4951}..l.l*10^{4932}$	10
Comp	$-2^{63}+1..2^{63}$-l	8

Belgili turlar. Ma'lumotlaming belgili turlari faqat bitta belgini saqlash uchun xizmat qiladi.

Tur	O‘lcham (baytda)
Char	1
ANSChar	1
WideChar	2

Mantiqiy turlar. Mantiqiy turlar rost (True) yoki yolg‘on (False) qiymatning birini qabul qiladi.

Tur	O‘lcham (baytda)
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Dasturchi tomonidan kiritiluvchi turlar

Delphi tili dasturchiga o‘z turlarini kiritishga imkon beradi.

Bu turlar standart turlarga yoki avval kiritilgan turlarga asoslangan bo‘lib quyidagi turlarga tegishli bo‘lishi mumkin:

- sanovchi;
- interval;
- murakkab tur (yozuv).

Sanoqli turlar. Sanoqli turlar tartiblangan qiymatlar to‘plamini ishlatadi.

Tur =(1 Qiymat, 2 Qiymat, ... ,n Qiymat)

Masalan:

Type

Color=(black, green, yellow, blue, red, white);
Fam=(Petrov, Sidorov, Rahimov, Sobirov);
DayOfWeek=(mon, tue, wed, thu, fri, sat, sun);
Bu yerda
Color sanoq turi beshta ranglar ketma-ketligini aniqlaydi.
Fam sanoq turi to‘rtta familiyani aniqlaydi.
DayOfWeek sanoq turi hafta nomlarini aniqlaydi.
Odatda Delphi tilida turlar nomlari T harfidan boshlanadi (Type — tip so‘zidan).
Yangi tur ta’riflangandan so‘ng shu turga tegishli o‘zgaruvchini ta’riflash mumkin. Masalan:

```
Type
TDayOfWeek = (MON, TUE, WED, THU, FRI, SAT,SUN);
var
ThisDay, LastDay: TdayOfWeek;
```

Sanovchi tur ta’rifi qiymatlar o‘zaro munosabatini ko‘rsatadi. Eng chap element minimal, eng o‘ng element maksimal hisoblanadi. Yuqorida kiritiigan DayOfWeek turi eiementiari uchun quyidagi munosabat o‘rinli:

```
MON < TUE < WED < THU < FRI < SAT < SUN
```

Sanovchi tur eiementlari orasidagi munosabat o‘zgaruvchilarni boshqaruvchi instruksiyalarda qo‘llashga imkon beradi. Masalan:

```
if (Day – SAT) OR (Day = SUN) then
begin
{ agar kun shanba yoki yakshanba bo‘lsa bajarilsin }
end;
```

Bu instruksiyani quyidagicha yozish mumkin:

```
if Day > FRI then begin
{agar kun shanba yoki yakshanba bo‘lsa bajarilsin }
end;
```

Sanovchi tur ta’rifi nomlangan konstantalarni kiritishning qisqartirilgan shakli deb qarash mumkin. Misol uchun TdayOfWeek 13iapaso ta’rifi quyidagi ta’riflarga tengdir:

```
Const
MON=0; TUE=1; WED=2; THU=3; FRI=4; SAT=5; SUN=6;
```

Interval (13iapason) turi. Interval (13iapason) turi beriladigan qiymatga chegara qo‘yadi.

```
Type
<tur nomi>=<minimal>..<maksimal>;
```

Masalan:

```
Type Color=red..green; // rangga chegara Digit=0..9; //
butun sonlarga chegara Symb=’A’..’Z’; // harflarga chegara
```

Haqiqiy turlarga chegara qo‘yilmaydi.
Interval tur ta’rifida nomlangan konstantalardan foydalanish mumkin. Quyidagi misolda interval tur Tindex ta’rifida HBOUND nomlangan konstantadan foydalanilgan:

```
Const
HBOUND=100;
type
Tindex=l..HBOUND;
```

Interval turdan foydalanish massivlarni ta’riflashda qulaydir:

```
Type
Tindex =1..100;
```

var
tabl:array [Tindex] of Integer; i:Tindex;

Butun son turidan tashqari asos tur sifatida sanovchi turdan foydalanish mumkin. Quyidagi dastur qismida Tmonth sanovchi tur asosida interval tur Tsammer ta’riflangan:

Type
Tmonth = (Jan, Feb, Mar, Apr, May, Jun,
Jul, Aug, Sep, Oct, Nov, Dec);
Tsammer = Jun.. Aug;

Yozuv.`Dasturlash amaliyotida standart ma'lumotlardan tashkil topgan murakkab ma'lumotlar bilan ishlashga to‘g'ri keladi. Misol uchun talaba to‘g'risidagi ma'lumotda uning ismi sharifi, tug'ilgan yili, manzili, kursi, guruhi va hokazolardan iborat bo‘lishi mumkin. Bunday ma'lumotlarni ta'riflash uchun Delphi da yozuv (record) lardan foydalaniladi.

Yozuv - bu alohida nomlangan har xil turli komponentalardan iborat murakkab turdir.

Har qanday tur kabi «yozuv» type bo‘limida ta'riflanishi lozim. Bu ta'rifning umumiy ko‘rinishi: Nom = record

1_Maydon: l_Tip; 2_Maydon: 2__Tip;....; K_Maydon: K__Tip; end;

Ta'riflarga misoUar:

Type
TPerson = record
f_name: string[20];
l_name: string[20];
day: integer;
month: integer;
year: integer:
address: string[50]; end;
TDate = record
day: Integer; month: integer; year: integer;
end;

Yozuv turidagi o‘zgaruvchini quyidagicha ta'riflash mumkin:

Var
student: TPerson; birthday : TDate;

Yozuv elementiga (maydoniga) murojaat qilish uchun yozuv nomi va nuqtadan so‘ng maydon nomini ko‘rsatish kerak. Masalan:

Writeln («Imya:», student.fname + #13 + ᵛAdres:\ student.address);

Instruksiya ekranga student o‘zgaruvchi—yozuvning fname (nom) va address (adres) maydonlarini chiqaradi.

Ba'zida o‘zgaruvchi - yozuv turi o‘zgaruvchilar e'lon qilish bo‘limida e'lon qilinadi. Bu holda, yozuv turi o‘zgaruvchi nomidan so‘ng ko‘rsatiladi. Misol uchun student yozuvi var bo‘limida quyidagicha ta'riflanishi mumkin:

student: record
f_name:string[20];
l_name:string[20];
day: integer;
month:integer;
year: integer;
address:string[50];

end;

With instruksiyasi

With instruksiyasi dasturda maydonlar nomlarini o‘zgaruvchi — yozuv nomini ko‘rsatmasdan ishlatishga imkon beradi. Umumiy holda with instruksiyasi quyidagi ko‘rinishga ega:

with nom do
begin
(dastur instruksiyasi} end;
Misol uchun dasturda quyidagi yozuv ta'riflangan bo‘lsin: student: record
F_name: string[30];
l_name: string[20];
address: string[50];
end;

va studentlar to‘g'risidagi ma'lumotlar El, E2 va E3 o‘zgaruvchilarda joylashgan bo‘lsin. U holda:

student.f_name := El;
sludent.l_name := E2;
student.address := E3;

instruksiyalar o‘rniga quyidagi instruksiyani yozish mumkin:
with student do begin
f_name :— El; l_name := E2; address := E3;
end;

C++ tilida ma’lumotlar uchun turlar quyidagicha bo‘ladi.

C++ tilining tayanch turlari, ularning baytlardagi o‘lchamlari va qiymatlarining chegaralari 1.2-jadvalda keltirilgan.

1.2-jadval. C++ tilining tayanch turlari

Tur nomi	Baytlardagi o‘lchami	Qiymat chegarasi
bool	1	True yoki false
Unsigned short int	2	0..65535
Short int	2	-32768..32767
Unsigned long int	4	0..42949667295
Long int	4	-2147483648..2147483647
int(16 razryadli)	2	-32768.. 32767
Int (32 razryadli)	4	-2147483648..2147483647
Unsigned int (16 razryadli)	2	0..65535
Unsigned int (32 razryadli)	4	0..42949667295
Unsigned char	1	0..255
char	1	-128.. 127
float;	4	1.2E-38..3.4E38
double	8	2.2E-308..1.8E308
Long double(32 razryadli)	10	3.4e-4932..-3.4e4932
void	2 ёки 4	-

C++ tilida ham o’zgaruvchilarning turlari bir nech guruhlarga ajraladi. Ularni quyida qarab chiqamiz.

Butun son turlari. Butun son qiymatlarni qabul qiladigan o‘zgaruvchilar int (butun), short(qisqa) va 1ong(uzun) kalit so‘zlar bilan aniqlanadi. O‘zgaruvchi qiymatlari ishorali bo‘lishi yoki unsigned kalit so‘zi bilan ishorasiz son sifatida qaralishi mumkin.

Belgi turi. Belgi turidagi o‘zgaruvchilar char kalit so‘zi bilan beriladi va ular o‘zida belgining ASCII kodini saqlaydi. Belgi turidagi qiymatlar nisbatan murakkab bo‘lgan tuzilmalar – satrlar, belgilar massivlari va hokazolarni hosil qilishda ishlatiladi.

Haqiqiy son turi. Haqiqiy sonlar float kalit so‘zi bilan e’lon qilinadi. Bu turdagi o‘zgaruvchi uchun xotiradan 4 bayt joy ajratiladi va <ishora><tartib><mantissa> qolipida sonni saqlaydi. Agar kasrli son juda katta (kichik) qiymatlarni qabul qiladigan bo‘lsa, u xotirada 8 yoki 10 baytli ikkilangan aniqlik ko‘rinishida saqlanadi va mos Double va long double kalit so‘zlari bilan e’lon qilinadi. Oxirgi holat 32-razryadli platformalar uchun o‘rinli.

Mantiqiy tur. Bu turdagi o‘zgaruvchi bool kalit so‘zi bilan e’lon qilinib, xotiradan 1 bayt joy egallaydi va 0 (false, yolg‘on) yoki (true, rost) qiymat qabul qiladi. Mantiqiy tur o‘zgaruvchilar qiymatlar o‘rtasidagi munosabatlarni ifodalaydigan mulohazalarni rost (true) yoki yolg‘on (false) ekanligi tavsifida qo‘llaniladi va ular qabul qiladigan qiymatlar matematik mantiq qonuniyatlariga asoslanadi.

Mantiqiy mulohazalar ustida uchta amal aniqlangan:

1) inkor – A mulohazani inkori deganda A rost bo‘lganda yolg‘on yoki yolg‘on bo‘lganda rost qiymat qabul qiluvchi mulohazaga aytiladi. C++ tilida inkor – ‘!’ belgisi bilan beriladi. Masalan, A mulohaza inkori «!A» ko‘rinishida yoziladi;

2) konyuksiya- ikkita A va B mulohazalar konyuksiyasi yoki mantiqiy ko‘paytmasi «A && B» ko‘rinishga ega. Bu mulohaza faqat A va B mulohazalar rost bo‘lgandagina rost bo‘ladi, aks holda yolg‘on bo‘ladi (odatda «&&» amali «va» deb o‘qiladi). Masalan «bugun oyning 5- kuni va bugun chorshanba» mulohazasi oyning 5- kuni chorshanba bo‘lgan kunlar uchungina rost bo‘ladi;

3) dizyunksiya – ikkita A va B mulohazalar dizyunksiyasi yoki mantiqiy yig‘indisi «A || B» ko‘rinishda yoziladi. Bu mulohaza rost bo‘lishi uchun A yoki B mulohazalardan biri rost bo‘lishi yetarli. Odatda «||» amali «yoki» deb o‘qiladi.

Yuqorida keltirilgan fikrlar asosida mantiqiy amallar uchun rostlik jadvali aniqlangan (1.3-jadval).

1.3-jadval. Mantiqiy amallar uchun rostlik jadvali

Mulohazalar		Mulohazalar ustida amallar		
A	B	Not A	A and B	A or B
		!A	A&&B	A||B
False	false	True	false	False
False	true	True	false	true
True	false	False	false	true
True	true	False	true	true

Mantiqiy tur qiymatlari ustida mantiqiy ko‘paytirish, qo‘shish va inkor amallarini qo‘llash orqali murakkab mantiqiy ifodalarni qurish mumkin. Misol uchun, «x –musbat va uning qiymati [1..3] sonlar oralig‘iga tegishli emas» mulohazasini mantiqiy ifoda ko‘rinishi quyidagicha bo‘ladi:

(x>0)&&(y<1|| y>3).

Void turi. C++ tilida **void** turi aniqlangan bo’lib bu turdagi dastur obyekti hech qanday qiymatga ega bo‘lmaydi va bu turdan qurilmaning til sintsksis:iga mos kelishini ta’minlash uchun

ishlatiladi. Masalan, C++ tili sintsksis:i funksiya qiymat qaytarishini talab qiladi. Agar funksiya qiymat qaytarmaydigan bo‘lsa, u void kalit so‘zi bilan e’lon qilinadi.

Misollar.

```
Int a=0 A=1; float abc=17.5;
double Ildiz;
bool ok=true;
char LETTER=’z’;
Void mening_funksiyam();/*funksiya qaytaradigan  qiymat
inobatga  olinmaydi */
```

Turni boshqa turga keltirish: C++ tilida bir turni boshqa turga keltirishning oshkor va oshkormas yo‘llari mavjud. Umuman olganda, turni boshqa turga oshkormas keltirish ifodada har xil turdagi o‘zgaruvchilar qatnashgan hollarda amal qiladi (aralash turlar arifmetikasi). Ayrim hollarda, xususan tayanch turlar bilan bog‘liq turga keltirish amallarida xatoliklar yuzaga kelishi mumkin. Masalan, hisoblash natijasining xotiradan vaqtincha egallagan joyi uzunligi, uni o‘zlashtiradigan o‘zgaruvchi uchun ajratilgan joy uzunligidan katta bo‘lsa, qiymatga ega razryadlarni yo‘qotish holati yuz beradi.

Oshkor ravishda turga keltirishda, o‘zgaruvchi oldiga qavs ichida boshqa tur nomi yoziladi:

```
#include <iostream.h>
int main()
{
int integer_1=54;
int integer_2;
float floating=15.854;
integer_1=(int) floating; // oshkor keltirish;
integer_2=(int) floating // oshkormas keltirish;
cout<<’’yangi integer (oshkor): ‘’<<integer_1<<’’\n’’;
cout<<’’yangi integer (oshkormas): ‘’<<integer_2<<’’\n’’;
return 0;
}
```

Dastur natijasi quyidagi ko‘rinishda bo‘ladi:

```
Yangi integer (oshkor):15
Yangi integer (oshkormas):15
```

Masala. Berilgan belgining ASCII kodi chop etilsin. Masala belgi turidagi qiymatni oshkor ravishda butun son turiga keltirib chop qilish orqali yechiladi.

Dastur matni:

```
#include <iostream.h>
int main()
{
Unsigned char A;
Cout<<’’belgini kiriting:’’;
Cin>>A;
Cout<<’\’’<<A<<’’’-belgi ASCII kodi=’’(int)A<<’\n’;
Return 0;
}
```

Dasturning belgini kiriting so‘roviga

```
A <enter>
```

amali bajarilsa, ekranga

‘A’-belgi ASCII kodi=65
satri chop etiladi.

1.5.Arifmetik amallar. Qiymat berish operatori

Berilganlarni qayta ishlash uchun dasturlash tillarida amallarning juda keng majmuasi aniqlangan. Amal - bu qandaydir harakat bo‘lib, u bitta (unar) yoki ikkita (binar) operandlar ustida bajariladi, hisob natijasi uning qaytaruvchi qiymati hisoblanadi.

Tayanch arifmetik amallarga qo‘shish (+), ayirish (-), ko‘paytirish (*), bo‘lish (/) va bo‘lishdagi qoldiqni olish (%) amallarini keltirish mumkin.

Amallar qaytaradigan qiymatlarni o‘zlashtirish uchun Delphi tilida faqat “:=” C++ tilida esa “=” va uning turli modifikatsiyalari ishlatilib, quyidagilar hisoblanadi: qo‘shish, qiymat berish bilan (+=); ayirish, qiymat berish bilan (-=); ko‘paytirish, qiymat berish bilan (*=); bo‘lish, qiymat berish bilan (/=); bo‘lish qoldig‘ini olish, qiymat berish bilan (%=) va boshqalar. Bu holatlarning umumiy ko‘rinishi:

<o‘zgaruvchi><amal>=<ifoda>;

Quyidagi dastur matnida ayrim amallarga misollar keltirilgan.

Delphi tilida	C++ tilida

Var a,b,c;integer;
z:char;
begin
a:=0; b:=4; c:=90;
a:=b;
z:=’ ‘;
write(a, z);
a:=b+c+c+b; write(a,z);
a:=b-2;
write(a,z);
a:=b*3;
write(a,z);
a:=c div(b+6);
write(a,z);
write(a mod 2, z);
a:=a+b;
write(a,z);
a:=a*(c-50);
write(a,z);
a:=a-38;
write(a,z);
a:=a mod 8;
write(a,z);
end.
	#include <iostream.h>
	int main()
	{
	int a=0 , b=4, c=90;char z=’\t’;
	a=b; cout<<a<<z;
	//a=4
	a=b+c+c+b; cout<<a<<z;
	//a=4+90+90+4=188
	a=b-2; cout<<a<<z;
	//a=2
	a=b*3 cout<<a<<z;
	//a=4*3=12
	a=c/(b+6); cout<<a<<z;
	//a=90/(4+6)=9
	cout<<a%2<<z;
	//9%2=1
	a+=b; cout<<a<<z;
	//a=a+b=9+4=13
	a*=c-50; cout<<a<<z;
	//a=a*(c-50)=13*(90-50)=520
	a-=38; cout<<a<<z;
	//a=a-38=520-38=482
	a%=8; cout<<a<<z;
	//a=a%8=482%8=2
	return 0;
	}

Dastur bajarilishi natijasida ekranda quyidagi sonlar satri paydo bo‘ladi:

4 188 2 12 9 1 482 2

1.6. Ifoda tushunchasi

Ifoda - amallar, operandlar va punktatsiya belgilarining ketma-ketligi bo‘lib, kompilyator tomonidan berilganlar ustida ma’lum bir amallarni bajarishga ko‘rsatma hisoblanadi. Har qanday ';' belgi bilan tugaydigan ifodaga til ko‘rsatmasi deyiladi.

Til ko‘rsatmasiga misol:

Delphi tilida	C++ tilida
x:=3*(y-2.45); u:=summa(a, 9,c) ;	x=3*(y-2.45); u=summa(a, 9,c) ;

1.7.Inkrement va dekrement amallari

C++ tilida operand qiymatini birga oshirish va kamaytirishning samarali vositalari mavjud. Bular inkrement (++) va decrement(--) unar amallardir.

(Delphida ifodaning qiymatining birga oshirish va kamaytirish uchun mos ravishda *inc* va *dec* protseduralaridan foydalaniladi.)

Operandga nisbatan bu amallarning prefiks va postfiks ko‘rinishlari bo‘ladi. Prefiks ko‘rinishda amal til ko‘rsatmasi bo‘yicha ish bajarilishidan oldin operandga qo‘llaniladi. Postfiks holatda esa amal til ko‘rsatmasi bo‘yicha ish bajarilgandan keyin operandga qo‘llaniladi.

Prefiks yoki postfiks amal tushunchasi faqat qiymat berish bilan bog’liq ifodalarda o‘rinli: ;

```
x=y++; // postfiks
index =--i; // prefiks
count:++; // unar amal,  "++count;  " bilan ekvivalent
abc--  ; // unar amal,  "--abc;  " bilan ekvivalent
```

Bu yerda y o‘zgaruvchining qiymati x o‘zgaruvchisiga o‘zlashtiriladi va keyin bittaga oshiriladi, i o‘zgaruvchining qiymati bittaga kamaytirib, index o‘zgaruvchisiga o‘zlashtiriladi.

sizeof amali: Har xil turdagi o‘zgaruvchilar kompyuter xotirasida turli sondagi baytlarni egallaydi. Bunda, hattoki bir turdagi o‘zgaruvchilar ham qaysi kompyuterda yoki qaysi operatsion sistemada amal qilinishiga qarab turli o‘lchamdagi xotirani band qilishi mumkin.

Delphi tilida ixtiyoriy (tayanch va hosilaviy) turdagi o‘zgaruvchilarning o‘lchamini sizeof funksiyasi yordamida aniqlash mumkin. Bu funksiyani o’zgarmasga, turga va o‘zgaruvchiga qo‘llash mumkin. Uning umumiy ko‘rinishi quyidagicha:

Function Sizeof(X):Integer;

Bu ishni C++ tilida sizeof amali yordamida bajarish mumkin. Quyida keltirilgan dasturda kompyuterning platformasiga mos ravishda tayanch turlarining o‘lchamlari chop qilinadi.

Begin Writeln(‘integer turning o‘lchami’, sizeof(integer)); Write(‘real turning o‘lchami’, sizeof(real)); Write(‘double turning o‘lchami’, sizeof(double)); Write(‘char turning o‘lchami’, sizeof(char)); end.	Int main() {cout<<’’int turining o‘lchami: "«sizeof (int)«"\n" {cout<<’’float turining o‘lchami: "«sizeof (float)«"\n"; {cout<<’’double turining o‘lchami: "«sizeof (double)«"\n"; {cout<<’’char turining o‘lchami: "«sizeof (char)«"\n"; return 0; }

1.8.Razryadli mantiqiy amallar

Dastur tuzish tajribasi shuni ko‘rsatadiki, odatda qo‘yilgan masalani yechishda biror holat ro‘y bergan yoki yo‘qligini ifodalash uchun 0 va 1 qiymat qabul qiluvchi bayroqlardan foydalaniladi. Bu maqsadda bir yoki undan ortiq baytli o‘zgaruvchilardan foydalanish mumkin. Masalan, mantiqiy turdagi o‘zgaruvchini shu maqsadda ishlatsa bo‘ladi. Boshqa tomondan, bayroq sifatida baytning razryadlaridan foydalanish ham mumkin. Chunki razryadlar faqat ikkita qiymatni – 0 va 1 sonlarini qabul qiladi. Bir baytda 8 razryad bo‘lgani uchun unda 8 ta bayroqni kodlash imkoniyati mavjud.

Faraz qilaylik, qo‘riqlash tizimiga 5 ta xona ulangan va tizim taxtasidagi 5 ta chiroqcha (indikator) xonalar holatini bildiradi: xona qo‘riqlash tizimi nazoratida ekanligini mos indikatorning yonib turishi (razryadning 1 qiymati) va xonani tizimga ulanmaganligini indikator o‘chganligi (razryadning 0 qiymati) bildiradi. Tizim holatini ifodalash uchun bir bayt yetarli bo‘ladi va uning kichik razryadidan boshlab beshtasini shu maqsadda ishlatish mumkin:

7	6	5	4 ind5	3 ind4	2 ind3	1 ind2	0 ind1

Masalan, baytning quyidagi holati 1, 4 va 5 xonalar qo‘riqlash tizimiga ulanganligini bildiradi:

7	6	5	4	3	2	1	0
x	x	x	1	1	0	0	1

Quyidagi jadvalda Delphi va C++ tillarida bayt razryadlari ustida mantiqiy amallar majmuasi keltirilgan. .

3.1 –jadval. Bayt razryadlari ustida mantiqiy amallar

Amallar		Mazmuni
and	&	Mantiqiy VA (ko’paytirish)
or	|	Mantiqiy yoki (qo’shish))
xor	^	Istisno qiluvchi YOKI
not	~	Mantiqiy INKOR (inversiya)

Razryadli mantiqiy amallarning bajarish natijalarini jadval ko‘rinishida ko‘rsatish mumkin.

3.2-jadval. Razryadli mantiqiy amallarning bajarish natijalari

A	B	C:=A and B	C:=A or B	C:=A xor B	C:= not A
		C=A&B	C=A|B	C=A^B	C=~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Yuqoridagi keltirilgan misol uchun qo‘riqlash tizimini ifodalovchi bir baytli belgi turidagi o‘zgaruvchini e’lon qilish mumkin:

q_taxtasi: char; (char q_taxtasi=0;)

Bu yerda q_taxtasi o‘zgaruvchisiga 0 qiymat berish orqali barcha xonalar qo‘riqlash tizimiga ulanmaganligi ifodalanadi:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

Agar 3-xonani tizimga ulash zarur bo‘lsa

q_taxtasi =q_taxtasi | 0x04(16);

amalini bajarish kerak, chunki 0x04(16)=00000100 2 va mantiqiy YOKI amali natijasida q_taxtasi o‘zgaruvchisi bayti quyidagi ko‘rinishda bo’ladi:

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

Xuddi shunday yo‘l bilan boshqa xonalarni tizimga ulash mumkin, zarur bo‘lsa birdaniga ikkitasini (zarur bo‘lsa barchasini):

q_taxtasi=q_taxtasi | 0x1F16;

Mantiqiy ko‘paytirish orqali xonalarni qo‘riqlash tizimidan chiqarish mumkin:

q_taxtasi=q_taxtasi&0xFD; //0xFD16 =11111101 2

Xuddi shu natijani inkor amalidan foydalangan holda ham olish mumkin. Ikkinchi xona tizimga ulanganligi bildiruvchi bayt qiymati -00000010 2 , demak shu holatni inkor qilgan holda mantiqiy ko‘paytirishni bajarish kerak.

q_taxtasi=q_taxtasi&(~0x02);

Va nihoyat, agar 3-xona indikatorini, uni qanday qiymatda bo’lishidan qat’iy nazar qarama-qarshi holatga o‘tkazishni «inkor qiluvchi YOKI» amali yordamida bajarish mumkin:

q_taxtasi=q_taxtasi^0x04; // Ox04 16 =00000100 2

C++ tilida razryadli mantiqiy amallarni qiymat berish operatori birgalikda bajarilishining quyidagi ko‘rinishlari mavjud:

&= - razryadli VA qiymat berish bilan;

| = - razryadli YOKI qiymat berish bilan;

^= - razryadli istisno qiluvchi YOKI qiymat berish bilan.

Chapga va o‘ngga surish amallari: Baytdagi bitlar qiymatini chapga yoki o‘ngga surish uchun, C++ tilida “<<” va “>>” amallari qo‘llanilib, amaldan keyingi son bitlarni nechta o‘rin chapga yoki o‘nga surish kerakligini bildiradi.

Masalan:

unsigned char A=12; //A=00001100_2=$0x0C_{16}$

A=A<<2; // A=00110000_2=$0x30_{16}$=48_{10}

A=A>>3; //A=00000110_2=$0x06_{16}$=6_{10}

Razryadlarni n ta chapga (o‘nga) surish sonni 2^n soniga ko‘paytirish (bo‘lish) amali bilan ekvivalent bo‘lib va nisbatan tez bajariladi. Shuni e’tiborga olish kerakki, operand ishorali son bo‘lsa, u holda chapga surishda eng chapdagi ishora razryadi takrorlanadi (ishora saqlanib qoladi) va manfiy sonlar ustida bu amal bajarilganda matematika nuqtai-nazardan xato natijalar yuzaga keladi:

unsigned char B=-120; // B=10001000_2=$0x88_{16}$

B=B«2; // B=00100000_2=$0x20_{16}$=32_{10}

B=-120; // $B=10001000_2=0x88_{16}$

B=B»3; // $B=11110001_2=OxF1_{16}=-15_{10}$

Shu sababli, bu razryadli surish amallari ishorasiz (unsigned) turdagi qiymatlar ustida bajarilgani ma’qul.

1.9.Taqqoslash amallari

Dasturlash tillarida qiymatlarni solishtirish uchun taqqoslash amallari aniqlangan (3.3-jadval). Taqqoslash amali binar amal bo‘lib, quyidagi ko‘rinishga ega:

<operand1> <taqqoslash amali> < operand2>

Taqqoslash amallarining natijasi – taqqoslash o‘rinli bo‘lsa, true (rost), aks holda false (yolg‘on) qiymat bo‘ladi. Agar taqqoslashda arifmetik ifoda qatnashsa, uning qiymati 0 qiymatidan farqli holatlar uchun 1 deb hisoblanadi.

3.3-jadval. Taqqoslash amallari va ularning qo‘llanishi

Amallar		Qo‘llanishi		Mazmuni (o‘qilishi)
<	<	a<b	a<b	“a kichik b”
<=	<=	a<=b	a<=b	“a kichik yoki teng b”
>	>	a>b	a>b	“a katta b”
>=	>=	a>=b	a>=b	“a katta yoki teng ”
=	==	a=b	a==b	“a teng b”
< >	!=	a< >b	a!=b	“a teng emas b”

1.10. «Vergul» amali

Delphi tilida «vergul» amali asosan ajratuvchi sifatida qo‘llaniladi. Misol uchun bir necha o‘zgaruvchilarni bir tur ostida e’lon qilishda yoki bir necha o‘zgaruvchilarga qiymat kiritishni kiritish operatori yordamida tasvirlashda ishlatiladi.

C++ tilida bir nechta ifodalarni kompilyator tomonidan yaxlit bir ifoda deb qabul qilish uchun «vergul» amali qo‘llaniladi. Bu amalni qo‘llash orqali dastur yozishda ma’lum bir samaradorlikka erishish mumkin. Odatda «vergul» amali if va for operatorlarida keng qo‘llaniladi. Masalan, if operatori quyidagi ko’rinishda bo‘lishi mumkin:

if(i=Ca11func(),i<7) . . .

Bu yerda, oldin Ca11func() funksiyasi chaqiriladi va uning natijasi i o‘zgaruvchisiga o‘zlashtiriladi, keyin i qiymati 7 bilan solishtiriladi.

1.11.Amallarning ustunliklari va bajarilish yo‘nalishlari

An’anaviy arifmetikadagidek dasturlash tillarida ham amallar ma’lum bir tartib va yo‘nalishda bajariladi. Ma’lumki, matematik ifodalarda bir xil ustunlikdagi (prioritetdagi) amallar uchrasa (masalan, qo‘shish va ayirish), ular chapdan o‘ngga bajariladi. Bu tartib dasturlash tillarida ham o‘rinli, biroq ayrim hollarda amal o‘ngdan chapga bajarilishi mumkin (qiymat berish amalida).

Ifodalar qiymatini hisoblashda amallar ustunligi hisobga olinadi, birinchi navbatda eng yuqori ustunlikka ega bo‘lgan amal bajariladi.

Operator	Tavsifi	Ustunligi	Yo’nalishi
::	Ko‘rinish sohasiga ruxsat berish	16	=>

[]	Massiv indeksi	16	=>
()	Funksiyani chaqirish	16	=>
->	Tuzilma yoki sinf elementini tanlash	16	=>
++	Postfiks inkrement	15	<=
--	Postfiks dekrement	15	<=
++	Prefiks inkrement	14	<=
--	Prefiks dekrement	14	<=
Sizeof	O’lchamni olish	14	<=
(<typ>)	Turga akslantirish	14	
~	Bitli mantiqiy INKOR	14	<=
!	Mantiqiy inkor	14	<=
-	Unar minus	14	
+	Unar plyus	14	<=
&	Adresni olish	14	<=
*	Vositali murojaat	14	<=
new	Dinamik obyektni yaratish	14	<=
delete	Dinamik obyektni yo‘q qilish	14	<=
casting	Turga keltirish	14	
*	Ko‘paytirish	13	=>
/	Bo‘lish	13	=>
%	Bo‘lish qoldig‘i	13	=>
+	Qo‘shish	12	=>
-	Ayirish	12	=>
»	Razryad bo‘yicha o‘ngga surish	11	=>
«	Razryad bo‘yicha chapga surish	11	
<	Kichik	10	=>
<=	Kichik yoki teng	10	=>
>	Katta	10	=>
>=	Katta yoki teng	10	
==	Teng	9	=>
!=	Teng emas	9	=>
&	Razryadli VA	8	=>
^	Razryadli istisno qiluvchi YOKI	7	=>
|	Razryadli YOKI	6	=>
&&	Mantiqiy VA	5	=>
||	Mantiqiy yoki	4	=>
?:	Shart amali	3	<=
=	Qiymat berish	2	<=
*=	Ko‘paytirish qiymat berish amali bilan	2	<=
/=	Bo’lish qiymat berish amali bilan	2	<=
%=	Modulli bo’lish qiymat berish amali bilan	2	<=
+=	Qo’shish qiymat berish amali	2	

	bilan		
-=	Ayirish qiymat berish amali bilan	2	<=
<<=	Chapga surish qiymat berish amali bilan	2	<=
>>=	o‘ngga surish qiymat berish amali bilan	2	<=
&=	Razryadli va qiymat berish bilan	2	<=
^=	Razryadli istisno qiluvchi yoki qiymat berish bilan	2	<=
|=	Razryadli yoki qiymat berish bilan	2	<=
Throw	Istisno holatni yuzaga keltirish	2	<=
,	Vergul	1	<=

Dastur tuzuvchisi amallarni bajarilish tartibini o’zgartirishi ham mumkin. Xuddi matematikadagidek, amallarni qavslar yordamida guruhlarga jamlash mumkin. Qavs ishlatishga cheklov yo’q .

Quyidagi dasturda qavs yordamida amallarni bajarish tartibini o’zgartirish ko’rsatilgan.

```
var
  x, y,a,b,c:integer;
Begin
 x:=0; y;=0;
a:=3; b:=34; c:=82;
x:=a*b+c;
y:=(a*(b+c));
Writeln(‘x=’,x);
Writeln(‘y=’,y);
end.
```

```
#include <iostream.h>
Int main()
{
int x=0,y=0;
int a=3, b=34, c=82;
x=a*b+c;
y=(a*(b+c));
cout<<’’x=’’<<x<<’’n\’’
cout<<’’y=’’<<y<<’’n\’’
}
```

Dasturda amallar ustunligiga ko‘ra *x* qiymatini hisoblashda oldin *a* o’zgaruvchi *b* o’zgaruvchiga ko’paytiriladi va unga *c* o’zgaruvchining qiymati qo’shiladi .

Navbatdagi ko‘rsatmani bajarishda esa birinchi navbatda ichki qavs ichidagi ifoda *–(b+c)* ning qiymati hisoblanadi, keyin bu qiymat *a* ga ko‘paytirilib, *y* o‘zgaruvchisiga o‘zlashtiriladi.

Dastur bajarilishi natijasida ekranga

x=184

y=348

satrlari chop etiladi.

II BOB OPERATORLAR

Dasturlash tili operatorlari yechilayotgan masala algoritmini amalga oshirish uchun ishlatiladi. Operatorlar chiziqli va boshqaruv operatorlariga bo'linadi. Aksariyat holatlarda operatorlar nuqtali vergul (';') belgisi bilan tugallanadi va u kompilyator tomonidan alohida operator deb qabul qilinadi (C++ tilidagi for operatorining qavs ichida turgan ifodalari bundan mustasno). Bunday operator ifoda operatori deyiladi.

2.1. Qiymat berish, inkrement va dekrement operatorlari

Qiymat berish amallari guruhi, xususan, qiymat berish operatorlari ifoda operatorlari hisoblanadi, Delphi da qiymat berish operatori ':=' ko'rinishida bo'lib, C++ da esa u '=' ko'rinishiga ega:

k:=8; (k=8;).

O'zgaruvchilarga qiymat berish, ya'ni ularning oldingi qiymatini o'zgartirishning boshqa usullari ham mavjud. Bu kabi ishlarni inc(i); (i++;), dec(j); (--j;), k:=k+i; (k+=i;) operatorlari yordamida amalga oshirish mumkin.

Bo'sh va e'lon operatorlari: Dastur tuzish amaliyotida bo'sh operator – ';' ishlatiladi. Garchi bu operator hech qanday ish bajarmasa ham, hisoblash ifodalarining til qurilmalariga mos kelishini ta'minlaydi. Ayrim hollarda yuzaga kelgan «boshi berk» holatlardan chiqib ketish imkonini beradi.

C++ tilida o'zgaruvchilarni e'lon qilish ham operator hisoblanadi va ularga e'lon operatori deyiladi.

Kiritish va chiqarish operatorlari: Biror-bir masalani yechishning chiziqli bo'lgan algoritmiga dastur tuzishda algoritmdagi keltirilgan ketma-ketliklar asosida operatorlar yoziladi. Bunday dasturlarni tuzishda asosan o'zgaruvchilarga qiymatni kiritish, natijalarni chiqarish va shu bilan birga o'zlashtirish operatorlari ishlatiladi.

Dasturdagi o'zgaruvchilar qiymatlarini dastur ichida o'zlashtirish operatori yordamida ham berish mumkin. Lekin dasturga o'zgaruvchining qiymatni tashqaridan kiritish ehtiyoji ham tug'iladi.

Delphi tilida o'zgaruvchilarga qiymat kiritish uchun Read operatori qo'llaniladi. U quyidagi ko'rinishlarga ega:

Read(<o'zgaruvchi 1>,<o'zgaruvchi 2>,...,<o'zgaruvchi *n*>);

Readln(<o'zgaruvchi 1>,<o'zgaruvchi 2>,...,<o'zgaruvchi *n*>);

Readln;

C++ tilida o'zgaruvchilar qiymatini klaviaturadan kiritish uchun *cin>><o'zgaruvchi 1> >> <o'zgaruvchi 2> >>... >> <o'zgaruvchi n>* kiritish oqimidan foydalaniladi.

Bu yerda <o'zgaruvchi 1>, <o'zgaruvchi 2>, ..., <o'zgaruvchi *n*>lar qiymat qabul qiladigan o'zgaruvchilar nomi;

Quyida keltirilgan jadvalda Delphi va C++ tillarida o'zgaruvchilarga qiymat kiritish tasvirlangan.

Read(Sl, S2); Readln(xl, x2, x3); Readln.	cin>>S1>>S2; cin>>x1>>x2>>x3>>"\n"; cin>>"\n";

Bu yerda birinchi operator Sl va S2 o‘zgaruvchilar qiymatini klaviaturadan kiritadi. Ikkinchi operator esa xl, x2, x3 o‘zgaruvchilar qiymatini klaviauradan qabul qiladi va kiritishni keyingi qatorga o‘tkazadi. Oxirgi operator esa kiritishni kutadi va kursorni navbatdagi qatorga o‘tkazadi.

Chop etish operatori: Delphi da Write operatori oddiy ma’lumotlar, o‘zgaruvchilar va ifodalar qiymatini chop etish uchun ishlatiladi. U quyidagi ko‘rinishlarga ega:

Write(<o‘zgaruvchi 1>, <o‘zgaruvchi 2>, …<o‘zgaruvchi *n*>);

Writeln(<o‘zgaruvchi 1>, <o‘zgaruvchi 2>, …<o‘zgaruvchi *n*>);

Writeln;

C++ tilida esa oddiy ma’lumotlar, o‘zgaruvchilar va ifodalar qiymatini chop etish uchun *cout<< <o‘zgaruvchi 1> << <o‘zgaruvchi 2> <<...*

<< <o‘zgaruvchi n> kiritish oqimidan foydalaniladi.

Bu yerda <o‘zgaruvchi l>, <o‘zgaruvchi 2>, ..., <o‘zgaruvchi n> - oddiy matnlar yoki o‘zgaruvchilar nomi;

Quyida keltirilgan jadvalda Delphi va C++ tillarida oddiy ma’lumotlar, o‘zgaruvchilar va ifodalar qiymatini chop etish namunalari keltirilgan.

Writeln(Summa); Write('Natija yo‘q'); Write('Tenglama yechimi xl=', xl, 'x2=', x2);	cout<<Summa<<”\n”; cout<<”Natija yo‘q” cout<<”Tenglama yechimi x1=”<<x1<<”x2=” <<x2;

Oddiy ma'lumotlarni chiqarishda ularga matn deb qaraladi va u qo‘shtirnoq ichida yoziladi.

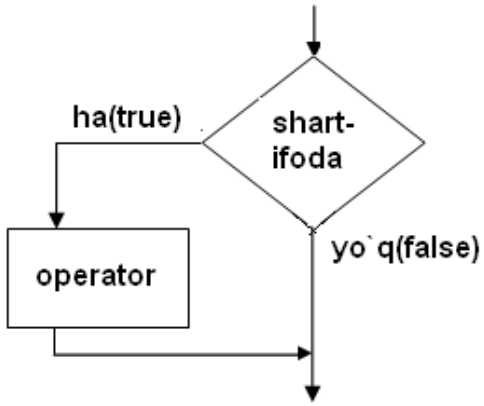
2.2.Shart operatorlari

Yuqorida mavzularda keltirilgan dasturlarda amallar yozilish tartibida ketma-ket va faqat bir marta bajariladigan holatlar, ya’ni chiziqli algoritmlar keltirilgan. Amalda esa kamdan-kam masalalar shu tariqa yechilishi mumkin. Aksariyat masalalar yuzaga keladigan turli holatlarga bog‘liq ravishda mos qaror qabul qilishni (yechimni) talab etadi. Delphi va C++ tillari dasturning alohida bo‘laklarining bajarilish tartibini boshqarishga imkon beruvchi qurilmalarning yetarlicha katta majmuasiga ega. Masalan, dastur bajarilishining birorta qadamida qandaydir shartni tekshirish natijasiga ko‘ra boshqaruvni dasturning u yoki bu bo‘lagiga uzatish mumkin (tarmoqlanuvchi algoritm). Tarmoqlanishni amalga oshirish uchun shartli operatordan foydalaniladi.

If operatori: If operatori qandaydir shartni rostlikka tekshirshi natijasiga ko‘ra dasturda tarmoqlanishni amalga oshiradi:

if <shart> then <operator>; (if (<shart>)<operator>;).

Bu yerda <shart> har qanday ifoda bo‘lishi mumkin. Odatda u taqqoslash amali bo‘ladi. Agar shart 0 qiymatidan farqli yoki rost (true) bo‘lsa, <operator> bajariladi, aks holda, ya’ni shart 0 yoki yolg‘on (false) bo‘lsa, hech qanday amal bajarilmaydi va boshqaruv if operatoridan keyingi operatorga o’tadi (i (agar u mavjud bo‘lsa). Ushbu holat 2.1 –rasmda ko‘rsatilgan.



2.1-rasm. if() shart operatorining blok sxemasi

Delphi va C++ tillarining qurilmalari operatorlarni blok ko‘rinishida tashkil qilishga imkon beradi.Blok Delphi tilida *begin* kalit so‘zi bilan boshlanib *end* kalit so‘zi bilan tugaydi. C++ tilida esa ‘{‘ va ‘}’ belgi oralig‘iga olingan operatorlar ketma-ketligi ko‘rinishida bo‘ladi. Blok kompilyator tomonidan yaxlit bir operator deb qabul qilinadi. C++ tilida blok ichida e’lon operatorlari ham bo’lishi mumkin va ularda e’lon qilingan o‘zgaruvchilar faqat shu blok ichida ko‘rinadi (amal qiladi), blokdan tashqarida ko‘rinmaydi. Blokdan keyin ‘;’ belgisi qo‘yilmasligi mumkin, lekin blok ichidagi har bir ifoda ‘;’ belgisi bilan yakunlanishi shart.

Quyida keltirilgan dasturda if operatoridan foydalanish ko’rsatilgan.

```
var
   b:integer;
begin
   read(b);
   if b>0 then
   begin
   //b>0 shart bajarilgan holat
   …
   Write(‘b-musbat son’);
   …
   end;
    if b<0 then
    Write(‘b-manfiy son’);
   //b<0 shart bajarilgan holat
 end.
```

```
#include <iostream.h>
int main()
{
int b;
cin>>b;
if (b>0)
{
 //b>0 shart bajarilgan holat
…
cout<<’’b- musbat son’’;
…
}
if (b<0)
cout<<’’b – manfiy son’’;
//b<0 shart bajarilgan holat
return 0;
}
```

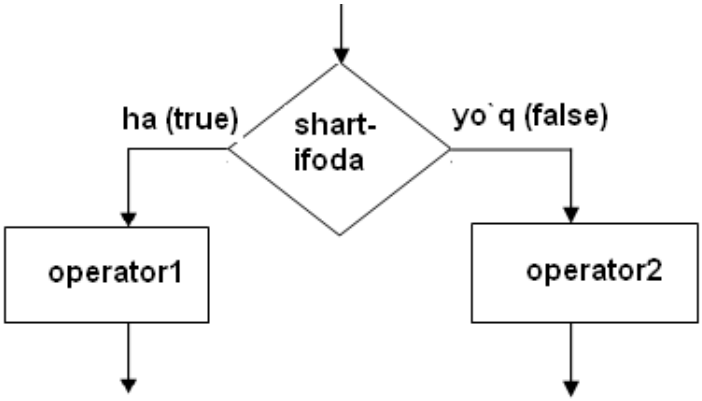
Dastur bajarilishi jarayonida butun turdagi b o‘zgaruvchi e’lon qilingan va uning qiymati klaviaturadan o‘qiladi. Keyin b qiymatini 0 sonidan kattaligi tekshiriladi, agar shart bajarilsa(true) , u holda ekranga “b – musbat son” xabari chiqadi. Agar shart bajarilmasa, bu operatorlar cheklab o‘tiladi. Navbatdagi shart operatori b o‘zgaruvchi qiymatini manfiylikka tekshiradi, agar shart bajarilsa, ekranga “b – manfiy son” xabari chiqadi.

If – else operatori: Shart operatorining if – else ko‘rinishi quyidagicha:

if <shart-ifoda > then <operator$_1$> else <operator$_2$>;

(if (<shart-ifoda>) <operator$_1$>; else <operator$_2$>;).

Bu yerda <shart-ifoda> 0 qiymatidan farqli yoki true bo‘lsa <operator$_1$>, aks holda <operator$_2$> bajariladi. If-else shart operator mazmuniga ko‘ra algoritmning tarmoqlanuvchi blokini ifodalaydi: <shart-ifoda> - shart bloki (romb) va <operator$_1$> blokning «ha» tarmog‘iga, <operator$_2$> esa blokning «yo‘q» tarmog‘iga mos keluvchi amallar bloklari deb qarash mumkin.



2.2-rasm. if(), else shart operatorining blok sxemasi

Misol tariqasida diskriminantni hisoblash usuli yordamida $ax^2+bx+c=0$ ko‘rinishidagi kvadrat tenglama ildizlarini topish masalasini ko‘raylik:

```
Var
    a,b,c, d,x1, x2:real;
 Begin
  Writeln(‘ax^2+bx+c=0  tenglama ildizini toppish);
Write(‘a- koiffitsiyentni kiriting:’);
read(a);
Write(‘b- koiffitsiyentni kiriting:’);
read(b);
Write(‘c- koiffitsiyentni kiriting:’);
read(c);
d=b*b-4*a*c;
if d<0 then
Write(‘tenglama haqiqiy ildizga ega emas!’);
if  d=0 then  begin
Writeln(‘tenglama yagona ildizga ega);
x1:=-b/(2*a);
Writeln(x1);
end
else begin
Writeln(‘tenglama ikkita ildizga ega’);
x1:=(-b+sqrt(D))/(2*a);
x2:=(-b-sqrt(D))/(2*a);
Writeln(x1, ‘  ’x2);
```

```
#include <iostream.h>
#include <math.h>
int main()
{
        float a,b,c;
float D,x1,x2;
cout<<’’ax^2+bx+c=0;  tenglama  ildizini
topish.’’;
cout<<’’\n a- koiffitsiyentni kiriting: ‘’;
cin>>a;
cout<<’’\n b- koeffitsientni kiriting: ‘’;
cin>>b;
cout<<’’\n c- koeffitsientni kiriting: ‘’;
cin>>c;
D=b*b-4*a*c;
If(D<0)
{cout<<’’tenglama  haqiqiy  ildizga  ega
emas!’’;
return 0;
}
if (D==0)
{cout<<’’tenglama yagona ildizga ega:’’;
x1=-b/(2*a);
cout<<’’\nx=’’<<x1;
return o;
```

end;
end.
	}
	else
	{cout<<’’tenglama ikkita ildizga ega:’’;
	x1=(-b+sqrt(D))/(2*a);
	x2=(-b-sqrt(D))/(2*a);
	cout<<’’\nx1=’’<<x1;
	cout<<’’\nx2=’’<<x2;
	}
	return 0;
	}

Dastur bajarilganda, birinchi navbatda tenglama koeffitsientlari – a, b, c o‘zgaruvchilar qiymatlari kiritiladi, keyin diskriminant – D o‘zgaruvchining qiymati hisoblanadi. Keyin D o‘zgaruvchining manfiy ekanligi tekshiriladi. Agar shart o‘rinli bo‘lsa, yaxlit operator bajariladi va ekranga “Tenglama haqiqiy ildizlarga ega emas” xabari chiqadi va dastur o‘z ishini tugatadi (“return 0;” operatorini bajarish orqali). Diskriminant noldan kichik bo‘lmasa, navbatdagi shart operatori uni nolga tengligini tekshiradi. Agar shart o‘rinli bo’lsa, keyingi qatorlardagi operatorlar bloki bajariladi – ekranga “ Tenglama yagona ildizga ega:” xabari, hamda x1 o‘zgaruvchi qiymati chop qilinadi va dastur shu yerda o‘z ishini tugatadi, aks holda, ya’ni D qiymatni noldan katta holati uchun else kalit so‘zidan keyingi operatorlar bloki bajariladi va ekranga “Tenglama ikkita ildizga ega:“ xabari, hamda x1 va x2 o‘zgaruvchilar qiymatlari chop etiladi. Shu bilan shart operatoridan chiqiladi va asosiy funksiyaning return ko’rsatmasini bajarish orqali dastur o‘z ishini tugatadi.

O‘z navbatida <operator$_1$> va <operator$_2$> ham shartli operator bo’lishi mumkin. Ifodadagi har bir else kalit so‘zi, oldindagi eng yaqin if kalit so‘ziga tegishli hisoblanadi (xuddi ochiluvchi va yopiluvchi qavslardek). Buni inobatga olmaslik mazmunan xatoliklarga olib kelishi mumkin.

Masalan:

if x=1 then
if y=1 then Write(‘x=1 va y=1’)
else Write(‘x< >1’);
	if (x==1)
	if (y==1) cout«”x=1 va y=1’’;
	else cout <<’’x< >1’’;

Bu misolda «*x< >1* » xabari *x* qiymatini 1 ga teng va *y* qiymatini 1 ga teng bo‘lmagan holda ham chop etiladi. Quyidagi variantda ushbu mazmunan xatolik bartaraf etilgan:

if x=1 then
begin
if y=1 then Write(‘x=1 va y=1’);
end
else Write(‘x< >1’);
	If (x==1)
	{
	If (y==1) cout<<”x=1 va y=1”;
	}
	else cout«”x< >1”;

Ikkinchi misol tariqasida uchta butun sonning maksimal qiymatini topadigan dastur bo‘lagini keltirishimiz mumkin:

var
x,y,z,max:integer;
…
Writeln(x,’ ‘,y, ’ ‘,z);
if x>y then
if z>x then max:=z
else max:=x
else
	…
	int x,y,z,max;
	cin»x»y»z;
	if (x>y)
	if (z>x) max=z;
	else max=x;
	else
	if (y<z) max=y;

if y>z then max:=y else max:=z; …	else max=z; …

C++ tilida shart operatorida umumiy bo‘lgan o‘zgaruvchilarni e’lon qilish man etiladi, lekin undagi bloklarda o‘zgaruvchilarni e’lon qilish mumkin va bu o‘zgaruvchilar faqat blok ichida amal qiladi. Quyidagi misolda bu holat bilan bog‘liq xatolik ko‘rsatilgan:

If (j>0) {int i; i=2*j;}

Else i=-j;//xato, chunki I blokdan tashqarida ko‘rinmaydi

Masala. Berilgan to‘rt xonali ishorasiz sonning boshidagi ikkita raqamining yig‘indisi qolgan raqamlar yig‘indisiga teng yoki yo‘qligi aniqlansin (raqamlar yig‘indisi deganda ularga mos son qiymatlarining yig‘indisi tushuniladi). Sonning raqamlarini ajratib olish uchun butun sonlar arifmetikasi amallaridan foydalaniladi:

```
Label 1;
var
  n, a3, a2, a1, a0:Word;
  begin
    Write(‘n ning qiymatini kiriting’);
     Read(n);
     If (n<1000) or (n>9999) then
       begin
        Writeln(‘kiritilgan son 4 xonali emas!’);
         Goto 1;
       end;
       a3:=n div 1000;
       a2:=(n mod 1000) div 100;
       a1:=(n mod 100) div 10;
       a0:=n mod 10;
       if  (a3+a2)=(a1+a0) then
       Write(‘a3+a2=a1+a0’)
       else Write(‘a3+a2< >a1+a0’);
       1:end.
```

```
#include  <iostream.h>
Int main()
{
Unsigned int n,a3,a2,a1,a0;//n=a3 a2 a1 a0 ko’rinishida
cout<<’’\nn-qiymatini kiriting:’’;
cin>>n;
If(n<1000| n>9999)
{
cout<<’’kiritilgan son 4 xonali emas!’’;
return1;
}
a3=n/1000;
a2=n%1000/100;
a1=n%100/10;
a0=n%10;
if(a3+a2==a1+a0) cout<<’’a3+a2=a1+a0’’;
else cout<<’’a3+a2<>a1+a0’’;
return 0;
}
```

Dastur ishorasiz butun son kiritishni taklif qiladi. Agar kiritilgan son 4 xonali bo‘lmasa (*n<1000* yoki *n>9999*), bu haqda xabar beriladi va dastur o‘z ishini tugatadi. Aks holda *n* sonining raqamlari ajratib olinadi, hamda boshidagi ikkita raqamning yig’indisi – (a3+a2) qolgan ikkita raqamlar yig‘indisi – (a1+a0) bilan solishtiriladi va ularning teng yoki yo‘qligiga qarab mos javob chop qilinadi.

?: shart amali: C++ tilida **“?”** amali ham aniqlangan bo‘lib tekshirilayotgan shart nisbatan sodda bo‘lsa, shart amalining <<?:>> ko‘rnishini ishlatish mumkin:

<shart ifoda> ? <ifoda1> : <ifoda2>;

Bu amal Delphi tilida mavjud emas.

Shart amali if shart operatoriga o‘xshash holda ishlaydi: agar <shart ifoda> 0 qiymatidan farqli yoki true bo‘lsa, <ifoda1>, aks holda <ifoda2>bajariladi. Odatda ifodalar qiymatlari birorta o‘zgaruvchiga o’zlashtiriladi.

Misol tariqasida ikkita butun son maksimumini topish ko‘raylik.

#include <iostream.h>

```
int main()
{
int a,b,c;
cout<<”a va b sonlar maksimumini topish dastursi.’’;
cout<<’’\n a- qiymatni kiriting: ‘’;
cin>>a;
cout<<’’\n b- qiymatni kiriting: ‘’;
cin>>b;
c=a>b?a:b;
cout<<’’\nSonlar maksimumi: ‘’<<c;
return 0;
}
```

Dasturdagi shart operatori qiymat berish operatorining tarkibiga kirgan bo‘lib, *a* o‘zgaruvchining qiymati *b* o‘zgaruvchining qiymatidan kattaligi tekshiriladi, agar shart rost bo‘lsa *c* o‘zgaruvchiga *a* o‘zgaruvchi qiymati, aks holda *b* o‘zgaruvchining qiymati o‘zlashtiriladi va *c* o‘zgaruvchining qiymati chop etiladi.

?: amalining qiymat qaytarish xossasidan foydalangan holda, uni bevosita cout ko‘rsatmasiga yozish orqali ham qo‘yilgan masalani yechish mumkin:

```
#include <iostream.h>
int main()
{
int a,b;
cout<<”a va b sonlar maksimumini topish dastursi.’’;
cout<<’’\n a- qiymatni kiriting: ‘’;
cin>>a;
cout<<’’\n b- qiymatni kiriting: ‘’;
cin>>b;
c=a>b?a:b;
cout<<’’\nSonlar maksimumi: ‘’<<(a>b) ?a:b;
return 0;
}
```

2.3. Tanlash operatori

Shart operatorining yana bir ko‘rinishi tanlash tarmoqlanish operatori bo‘lib, uning sintsksis:i quyidagacha:

case <ifoda> of
<ifoda$_1$>:<operatorlar guruhi$_1$>;
<ifoda$_2$>:<operatorlar guruhi$_2$>;
. . .
<ifoda$_n$>:<operatorlar guruhi$_n$>;
else <operatorlar guruhi$_{n+1}$>
end;
	switch (<ifoda>)
	{
	case <o‘zgarmas ifoda^1> : <operatorlar guruhi^1>; break;
	case <o‘zgarmas ifoda^2> : <operatorlar guruhi^2>; break;
	…
	case <o‘zgarmas ifoda^n> : <operatorlar guruhi^n>; break;
	default:: <operatorlar guruhi^{n+1}>;
	}

Delphi tilida tanlash operatori quyidagicha amal qiladi: birinchi navbatda <ifoda> qiymati hisoblanadi, keyin bu qiymat <o‘zgarmas ifoda$_i$> bilan solishtiriladi. Agar ular ustma-ust tushsa, shu qatordagi ‘:’ belgisidan boshlab, toki navbatdagi o‘zgarmas ifodagacha bo‘lgan <operatorlar guruhi^{i}> bajariladi va boshqaruv tarmoqlanuvchi operatordan keyingi joylashgan operatorga o‘tadi. Agar <ifoda> birorta ham <o‘zgarmas ifoda^{i}> bilan mos kelmasa, qurilmaning else qismidagi <operatorlar guruhi^{n+1}> bajariladi. Shuni qayd etish kerakki, qurilmada else kalit so‘zi faqat bir marta uchrashi mumkin.

Delphida bir operatorlar guruhiga bir necha o‘zgarmas ifodani mos qo‘yish ham mumkin. U holda o‘zgarmas ifodalar vergullar bilan ajratib yoziladi.

C++ tilida esa bu operator quyidagicha amal qiladi: birinchi navbatda <ifoda> qiymati hisoblanadi, keyin bu qiymat case kalit so‘zi bilan ajratilgan <o‘zgarmas ifoda$_i$> bilan solishtiriladi. Agar ular ustma-ust tushsa, shu qatordagi ‘:’ belgisidan boshlab, toki break kalit so‘zigacha bo‘lgan <operatorlar guruhi^{i}> bajariladi va boshqaruv tarmoqlanuvchi operatordan keyingi joylashgan operatorga o‘tadi. Agar <ifoda> birorta ham <o‘zgarmas ifoda^{i}> bilan mos kelmasa, qurilmaning default qismidagi <operatorlar guruhi^{n+1}> bajariladi. Shuni qayd etish kerakki, qurilmada default kalit so‘zi faqat bir marta uchrashi mumkin.

Tanlash operatorini qo‘llashga doir misolni qarab chiqaylik. Klaviaturadan kiritilgan “Jarayon davom etilsinmi?” so‘roviga foydalanuvchi tomonidan javob olinadi. Agar ijobiy javob olinsa, ekranga “Jarayon davom etadi!” xabari chop etiladi va dastur o‘z ishini tarmoqlanuvchi operatordan keyingi operatorlarni bajarish bilan davom ettiradi, aks holda “Jarayon tugadi!” javobi beriladi va dastur o‘z ishini tugatsin. Bu masala uchun tuziladigan dastur foydalanuvchining ‘*y*’ yoki ‘*Y*’ javoblari jarayonni davom ettirishni bildiradi, boshqa belgilar esa tugatishni anglatadi.

```
label 2;
var
 javob:char;
 begin
  Writeln(‘jarayon davom etsinmi? ‘y’,’Y’’);
   Read(javob);
    case javob of
      ‘y’, ‘Y’:
   Writeln(‘jarayon davom etadi’);
    else    begin
    Writeln(‘jarayon tugadi’);
     Goto 2;
     end;
     end;
     …//jarayon
     2:end.
```

```
#include <iostream.h>
int main()
{
char javob=’ ‘;
cout<<’’jarayon                    davom
etsinmi?(‘y’,’Y’):’’;
cin>>javob;
switch(javob)
{
case ‘y’:
case ‘Y’:
cout<<’’jarayon davom etadi!\n‘’;
break;
default:
cout<<’’jarayon tugadi!\n‘’;
return 0;
}
…//jarayon
return 0;
}
```

Tanlash operatorining C++ tilidagi ko‘rinishida break va default kalit so‘zlarini ishlatmasa ham bo‘ladi. Ammo bu holda operatorning mazmuni buzilishi mumkin. Masalan, default qismi

bo‘lmagan holda, agar <ifoda> birorta <o‘zgarmas ifoda$_i$> bilan ustma-ust tushmasa, operator hech qanday amal bajarmasdan boshqaruv tanlash operatordan keyingi operatorga o‘tadi. Agar break bo‘lmasa, <ifoda> birorta <o’zgarmas ifoda$_i$> bilan ustma-ust tushgan holda, unga mos keluvchi operatorlar guruhini bajaradi va «to‘xtamasdan» keyingi qatordagi operatorlar guruhini bajarishga o‘tib ketadi. Masalan, yuqoridagi misolda break operatori bo‘lmasa va jarayonni davom ettirishni tasdiqlovchi (‘Y’) javob bo‘lgan taqdirda ekranga

Jarayon davom etadi!
Jarayon tugadi!

xabarlari chiqadi va dastur o‘z ishini tugatadi (return operatorining bajarilishi natijasida).

Delphi tilida yuqoridagi kabi holat bilan bog‘liq xatolik ro‘y bermaydi, ammo har bir tanlovga tegishli bo‘lgan operatorlar guruhida bittadan ortiq operator ishtirok etsa ular begin va end kalit so‘zlari orasiga olib yozilishi shart. Else kalit so‘zining qo‘llanilishi esa xuddi default kalit so‘ziniki kabidir.

Tanlash operatori sanab o‘tiluvchi turdagi o‘zgarmaslar bilan birgalikda ishlatilganda samara beradi. Quyidagi dasturda ranglar gammasini toifalash masalasi yechilgan.

```
type
  ranglar=(qizil, tuq_sariq, sariq, yashil, kuk, zangori, binafsha);
  var
    rang:ranglar;
  begin
   ….
   case rang of
   qizil, tuq_sariq, sariq: Writeln (‘Issiq gamma tanlandi’);
   yashil, kuk, zangori, binafsha: Writeln (‘Sovuq gamma tanlandi’)
     else
     Writeln(‘’’Kamalak bunday rangga ega emas’);
      end;
  end.
```

```
#include <iostream.h>
int main()
{
enum Ranglar{qizil, tuq_sariq,sariq,yashil,kuk,zangori,binafsha};
Ranglar rang;
//...
switch(rang)
{
case qizil:
case tuq_sariq:
case sariq;
cout<<’’Issiq gamma tanlandi. \n’’;
break;
case yashil:
case kuk:
case zangori:
case binafsha:
cout<<’’Sovuq gamma tanlandi.\n’’;
break;
default:
cout<<’’Kamalak bunday rangga ega emas. \n’’;
}
return 0;
}
```

Dastur bajarilishida boshqaruv tanlash operatorga kelganda, rang qiymati qizil yoki tuq_sariq yoki sariq bo‘lsa, “Issiq gamma tanlandi’’ xabari, agar rang qiymati yashil yoki kuk yoki zangori yoki binafsha bo‘lsa, ekranga “Sovuq gamma tanlandi” xabari, agar rang qiymati sanab o‘tilgan qiymatlardan farqli bo‘lsa, ekranga “Kamalak bunday rangga ega emas “ xabari chop etiladi va dastur o‘z ishini tugatadi.

C++ tilidagi tanlash operatorida e’lon operatorlari ham uchrashi mumkin. Lekin switch operatori bajarilishida «sakrab o‘tish» holatlari bo‘lishi hisobiga blok ichidagi ayrim e’lonlar bajarilmasligi va buning oqibatida dastur ishida xatolik ro‘y berishi mumkin:

Misol.

```
//…
int k=0,n=0;
cin>>n;
switch (n)
{
int=10;   //xato,bu operator hech qachon bajarilmaydi
case 1:
int j=20;   //agar n=2 bo‘lsa,bu e’lon bajarilmaydi
case 2:
k+=i+j; //xato, chunki i,j o‘zgaruvchilar noma’lum
}
cout<<k;
//…
```

Tanlash operatorini qo‘llashga doir yana bir masalani qarab chiqamiz.

Masala. Quyida, sanab o‘tiluvchi turlar va shu turdagi o‘zgaruvchilar e’lon qilingan:

enum Birlik{detsimetr, kilometr, metr, millimetr, santimetr}

float y; Birlik r;

Birlikda berilgan x o‘zgaruvchisining qiymat metrlarda chop qilinsin.

```
Label 3;
Type birlik=(detsimetr, kilometr, metr, millimetr, santimetr);
  Var
   x,y:real; r:birlik;
  Begin
    Write(‘uzunlikni kiriting: x=’); read(x);
    Writeln(‘uzunlik birliklari’);
    Writeln(‘0-detsimetr’);
    Writeln(‘1-kilometr’);
    Writeln(‘2-metr’);
    Writeln(‘3-millimetr’);
    Writeln(‘0-santimetr’);
    Writeln(‘uzunlik birligini tanlang; r=’);
     read(r);
       case r of
         detsimetr: y=x/10;
         kilometr: y=x*1000;
         metr: y=x;
         millimetr: y=x/1000;
         santimetr: y=x/100;
        else begin
    Writeln(‘uzunlik birligi noto‘g‘ri kiritildi!);
```

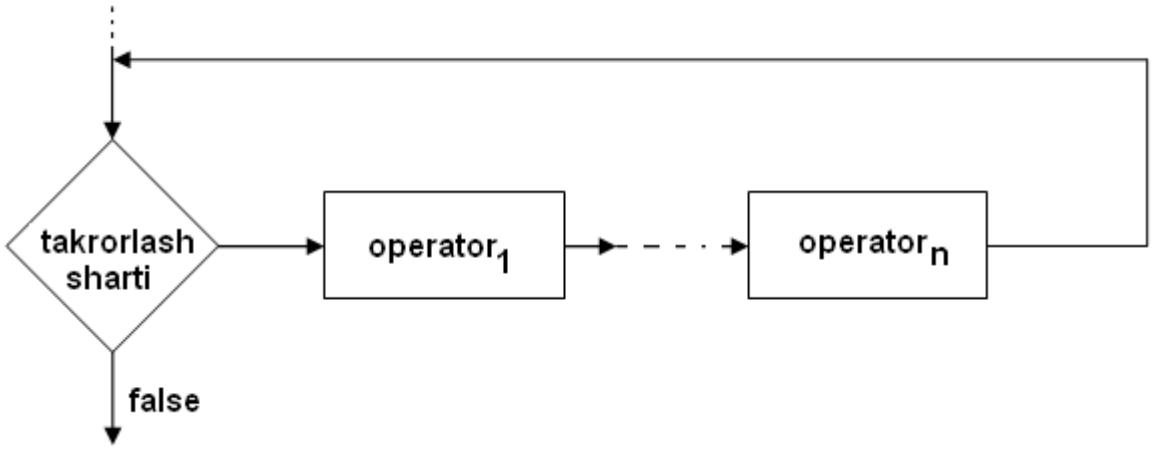
```
        #include <iostream.h>
        int main()
        {
        enum Birlik {detsimetr, kilometr, metr, millimetr, santimetr};
        float x,y;
        Birlik r;
        cout<<’’uzunlikni kiriting: x=’’;
        cin>>x;
        cout<<’’ uzunlik birliklari\n’’;
        cout<<’’ 0-detsimetr\n’’;
        cout<<’’1-kilometr\n’’;
        cout<<’’2-metr\n’’;
        cout<<’’3-millimetr\n’’;
        cout<<’’4-santimetr\n’’;
        cout<<’’uzunlikni birligini tanlang; r=’’;
        cin>>r;
        switch(r)
        {
        case detsimetr:y=x/10; break;
        case kilometr: y=x*1000; break;
        case metr: y=x; break;
```

Goto 3;
end;
Write(y,' metr',);
3:end.
	case millimetr: y=x/1000; break;
	case santimetr: y=x/100; break;
	default:
	cout<<''uzunlik birligi noto'g'ri kiritildi!'';
	return 0;
	}
	cout<<y<<''metr'';
	return 0;
	}

2.4. Takrorlash operatorlari

Dastur bajarilishini boshqarishning boshqa bir kuchli mexanizmlaridan biri – takrorlash operatorlari hisoblanadi.

Takrorlash operatori «takrorlash sharti» deb nomlanuvchi ifodaning rost qiymatida dasturning ma'lum bir qismidagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajaradi(itarativ jarayon)(2.3-rasm).



2.3-rasm. Takrorlash operatorining blok sxemasi

Takrorlash o‘zining kirish va chiqish nuqtalariga ega, lekin chiqish nuqtasining bo‘lmasligi mumkin. Bu holda takrorlashga cheksiz takrorlash deyiladi. Cheksiz takrorlash uchun takrorlashni davom ettirish sharti doimo rost bo‘ladi.

Takrorlash shartini tekshirish takrorlash tanasidagi operatorlarni bajarishdan oldin tekshirilishi mumkin (for, while takrorlashlari) yoki takrorlash tanasidagi operatorlari bir marta bajarilgandan keyin tekshirilishi mumkin (repeat-until, do-while).

Takrorlash operatorlari ichma-ich joylashgan bo‘lishi ham mumkin.

for* takrorlash operatori:** ***for takrorlash operatorining Delphi va C++ tillaridagi sintsksis:i farq qilganligi uchun ularni alohida-alohida qarab o‘tamiz.

Delphi tilida ***for*** operatori takrorianishlar soni aniq bo‘lgan sikiiik jarayoniar tashkil etishda ishlatiladi. Uning umumiy ko‘rinishi quyidagicha:

for <o‘zgaruvchi>:=<ifoda1> *to* <ifoda2> *do* <operatorlar guruhi>

Bu yerda <o‘zgaruvchi> - sikl parametri; <ifoda1>, <ifoda2> - <o‘zgaruvchi> parametrining boshlang‘ich va oxirgi qiymati bo‘lib, ular o‘zgarmas son, belgi yoki ifoda

bo‘lishi mumkin; <operatorlar guruhi> - sikl tanasi bo‘lib, bir necha operatorlardan tashkil topishi mumkin.

Agar sikl tanasi bittadan ortiq operatordan iborat bo‘lsa, ular **Begin** va **End** kalit so‘zlari ichiga olinadi.

Sikl operatoridagi <o‘zgaruvchi> ning dastlabki qiymati <ifoda1> ning qiymatiga teng bo‘lib, har bir takrorlanish bajarilganda uning qiymati bir birlik ortadi va bu jarayon o‘zgaruvchining qiymati <ifoda2> ning qiymatidan kata bo‘lgunga qadar davom etadi.

Agar sikl parametridagi to kalit so‘zi **downto** kalit so‘zi bilan almashtirilsa sikl parametri <o‘zgaruvchi> si har bir takrorlanishda o‘z qiymatini 1 birlik kamaytiradi va <o‘zgaruvchi> ning qiymati <ifoda2> dan kichik bo‘lgunga qadar davom etadi. Parametrli sikl quyidagi ko‘rinishni oladi:

for <o‘zgaruvchi>:=<ifoda1> downto <ifoda2> do <operatorlar guruhi>;

C++ tilida esa *for* takrorlash operatorining sintsksis:i quyidagi ko‘rinishga ega:

for (<ifoda1>; <ifoda2>;<ifoda3>) <operatorlar guruhi>;

Bu operator o‘z ishini <ifoda1> ifodasini bajarishdan boshlaydi. Keyin takrorlash qadamlari boshlanadi. Har bir qadamda <ifoda2> bajariladi, agar natija 0 qiymatidan farqli yoki true bo‘lsa, takrorlash tanasi - <operatorlar guruhi> bajariladi va oxirida <ifoda3> bajariladi. Agar <ifoda2> qiymati 0 (false) bo‘lsa, takrorlash jarayoni to‘xtaydi va boshqaruv takrorlash operatoridan keyingi operatorga o‘tadi. Shuni qayd qilish kerakki, <ifoda2> ifodasi vergul bilan ajratilgan bir nechta ifodalar birlashmasidan iborat bo‘lishi mumkin, bu holda oxirgi ifoda qiymati takrorlash sharti hisoblanadi. Takrorlash tanasi sifatida bitta operator, jumladan bo‘sh operator bo‘lishi yoki operatorlar guruhi kelishi mumkin.

Misol uchun 1 dan 20 gacha bo‘lgan butun sonlar yig‘indisini hisoblash masalasini ko‘raylik.

```
Var
   Summa, i:integer;
   Begin   Summa:=0;
  For i:=1 to 20 do
  Summa:=Summa+i;
  Writeln(‘yig`indi’=’,Summa);
  end.
```

```
#include <iostream.h>
int main()
{int Summa=0;
for (int i=1; i<=20; i++)
Summa+=i;
cout<<’’yig’indi=’’ <<Summa;
return 0;}
```

Dasturdagi takrorlash operatori o‘z ishini, i takrorlash parametriga (takrorlash sanagichiga) boshlang‘ich qiymat – 1 sonini berishdan boshlaydi va har bir takrorlash qadamidan (itaratsiyadan) keyin uning qiymati bittaga oshadi. Har bir takrorlash qadamida takrorlash tanasidagi operator bajariladi, ya’ni Summa o‘zgaruvchisiga *i* ning qiymati qo‘shiladi. Takrorlash sanagichi *i* ning qiymati 21 bo‘lganda “i<=20” takrorlash sharti (0-qiymati) bo‘ladi va takrorlash tugaydi. Natijada boshqaruv takrorlash operatoridan keyingi operatorga o‘tadi va ekranga yig’indi chop etiladi.

Yuqorida keltirilgan misolga qarab takrorlash operatorlarining qavs ichidagi ifodalariga izoh berish mumkin:

<ifoda1> - takrorlash sanagichi vazifasini bajaruvchi o‘zgaruvchisiga boshlang’ich qiymat berishga xizmat qiladi va u takrorlash jarayoni boshida faqat bir marta hisoblanadi. Ifodada o‘zgaruvchi e’loni uchrash mumkin va bu o‘zgaruvchi takrorlash operatori tanasida amal qiladi va takrorlash operatoridan tashqarida «ko‘rinmaydi» (C++ Builder kopiilyatori uchun);

<ifoda2> - takrorlashni bajarish yoki yo‘qligini aniqlab beruvchi mantiqiy ifoda, agar shart rost bo‘lsa, takrorlash davom etadi, aks holda yo‘q. Agar bu ifoda bo‘sh bo‘lsa, shart doimo rost deb hisoblanadi;

<ifoda3> - odatda takrorlash sanagichning qiymatini oshirish (kamaytirish) uchun xizmat qiladi yoki unda takrorlash shartiga ta’sir boshqa amallar bo‘lishi mumkin.

Takrorlash operatorida qavs ichidagi ifodalar bo‘lmasligi mumkin, lekin sintsksis: ‘;’ bo‘lmasligiga ruxsat bermaydi. Shu sababli sodda ko‘rinishdagi takrorlash operatori quyidagicha bo‘ladi:

for(;;) cout <<”Cheksiz takrorlash…’’;

Agar takrorlash jarayonida bir nechta o‘zgaruvchilarning qiymati sinxron ravishda o‘zgarishi kerak bo‘lsa, <ifoda1> va <ifoda3> ifodalarida zarur operatorlarni ‘,’ bilan yozish orqali bunga erishish mumkin:

```
for(int  i=10 , j=2 ; i<=20 ; i++ , j=i+10)
{
…
}
```

Takrorlash operatorining har bir qadamida j va i o‘zgaruvchi qiymatlari mos ravishda o‘zgarib boradi.

For operatorida takrorlash tanasi bo‘lmasligi ham mumkin. Masalan, dastur bajarilishini ma’lum bir muddatga «to‘xtab» turish zarur bo‘lsa, bunga takrorlashni hech qanday qo‘shimcha ishlarni bajarmasdan amal qilishi orqali erishish mumkin:

```
#include <iostream .h>
int main()
{
int delay;
…
for (delay=5000; delay>0; delay--);//  bo’sh operator
…
return 0;
}
```

Yuqorida keltirilgan 1 dan 20 gacha bo‘lgan sonlar yig‘indisini bo‘sh tanali (bo‘sh operatorli) takrorlash operatori orqali hisoblash mumkin:

```
…
for (int i=1;   i<=20;   summa+=i++) ;
…
```

Takrorlash operatori tanasi sifatida operatorlar guruhi ishlatishini faktorialni hisoblash misolida ko‘rsatish mumkin:

```
var
  a,i:integer;
  fact:longint;
begin
     fact:=1;
    Write(‘butun sonni kiriting’); read(a);
     if (a>=0) and (a<33) then begin
     for i:=1 to a do
     fact:= fact*i;
```

```
#include <iostream.h>
int main()
{
int a;
unsigned long fact=1;
cout<<’’butun sonni kiriting:_’’;
cin>>a;
if ((a>=0)&&(a<33))
{
```

```
    Write(a, ‘factorial ‘,fact,’ ga teng);
    end;
    end.
```

```
        for (int i=1; i<=a; i++) fact*=i;
        cout<<a<<’’factorial’’<<fact<<”      ga
teng \n”;
        }
        return 0;
        }
```

Dastur foydalanuvchi tomonidan 0 dan 33 gacha oraliqdagi son kiritilganda amal qiladi, chunki 34! Qiymati unsigned long uchun ajratilgan razryadlarga sig‘maydi.

Masala. Takrorlash operatorining ichma-ich joylashuviga misol sifatida raqamlari bir-biriga o‘zaro teng bo‘lmagan uch xonali natural sonlarni o‘sish tartibida chop qilish masalasini ko‘rishimiz mumkin:

```
var
 a2, a1, a0:char;
begin
   for a2:=’1’ to ‘9’ do
   for a1:=’0’ to ‘9’ do
   for a0:=’0’ to ‘9’ do
   if (a0< >a1) and (a1< >a2) and (a0< >a2)
then   Writeln(a2,a1,a0);
end.
```

```
#include <iostream.h>
int main()
{
unsigned char a2,a1,a0;
// uch xonali son raqamlari
for (a2=’ 1’ ;a2<=’ 9’ ;a2++)
//sonning 2-o‘rindagi raqami
for (a1=’ 0’ ;a1<=’ 9’ ;a1++)
//sonning 1-o‘rindagi raqami
for (a0=’ 0’ ;a0<=’ 9’ ;a0++)
 //sonning 0-o‘rindagi raqami
// raqamlarni o‘zaro teng emasligini tekshirish
if(a0!=a1 && a1!=a2 && a0!=a2)   //o‘zaro
teng emas
cout<<a2<<a1<<a0<<’\n’ ;
return 0;
}
```

Dasturda uch xonali sonning har bir raqami takrorlash operatorlari yordamida hosil qilinadi. Birinchi, tashqi takrorlash operatori bilan 2-xonadagi raqam (a2 takrorlash parametri) hosil qilinadi. Ikkinchi, ichki takrorlash operatorida (a1 takrorlash parametri) son ko‘rinishining 1-xonasidagi raqam va nihoyat, unga nisbatan ichki bo‘gan a0 parametrli takrorlash operatorida 0-xonadagi raqamlar hosil qilinadi. Har bir tashqi takrorlashning bir qadamiga ichki takrorlash operatorining to‘liq bajarilishi to‘g‘ri keladi.

While takrorlash operatori: ***While*** takrorlash operatori, operator yoki blokni takrorlash sharti yolg’on (false yoki 0) bo‘lguncha takror bajaradi.

While sikl operatori takrorlanishlar soni oldindan aniq bo‘lmagan hollarda takrorlanishni biror-bir shart asosida bajaradi. Berilgan shart oldin tekshiriladi va keyin shartning rost yoki yolg‘onligiga qarab kerakli operatorlar ketma-ketligi bajariladi. Bu operatorning Delphi va C++ tillaridagi sintsksis:ini keltiramiz:

Delphi da: **While <shart> do <operatorlar guruhi>;**

Bu yerda <shart>- mantiqiy ifoda; <operatorlar guruhi> - sikl tanasi bo‘lib, bir yoki bir necha operatorlar ketma-ketligidan iborat bo‘lishi mumkin. Mantiqiy ifoda ‘true’(rost) yoki ‘false’(yolg‘on) qiymatlardan birini qabul qiladi.

Agar mantiqiy ifoda ‘true’ qiymatni qabul qilsa <operatorlar guruhi> operatorlari bajariladi, aks holda bajarilmaydi, ya’ni sikl ishlashdan to‘xtaydi.

C++ tilida while operatori quyidagi sintsksis:ga ega:

while (<ifoda>) <operatorlar guruhi>;

Agar <ifoda> rost qiymatli o‘zgarmas ifoda bo‘lsa, takrorlash cheksiz bo‘ladi. Xuddi shunday, <ifoda> takrorlash boshlanishida rost bo’lib, uning qiymatiga takrorlash tanasidagi hisoblash ta’sir etmasa, ya’ni uning qiymati o‘zgarmasa, takrorlash cheksiz bo‘ladi.

While takrorlash shartini oldindan tekshiruvchi takrorlash operatori hisoblanadi. Agar takrorlash boshida <ifoda> yolg‘on bo‘lsa, while operatori tarkibidagi <operatorlar guruhi> qismi bajarilmasdan chetlab o’tiladi.

Ayrim hollarda <ifoda> qiymat berish operatori ko‘rinishida kelishi mumkin. Bunda qiymat berish amali bajariladi va natija 0 solishtiriladi. Natija noldan farqli bo‘lsa, takrorlash davom ettiriladi.

Agar rost ifodaning qiymati noldan farqli o‘zgarmas bo‘lsa, cheksiz takrorlash ro‘y beradi. Masalan:

```
While(1); // cheksiz takrorlash
```

Xuddi for operatoridek, ‘,’ yordamida <ifoda> da bir nechta amallar sinxron ravishda bajarilishi mumkin. Masalan, son va uning kvadratlarini chop qiladigan dasturda ushbu holat ko‘rsatilgan:

```
#include <iostream.h>
int main()
{
int n, n2;
cout<<’’sonni kiriting(1..10):_’’;
cin>>n;
n++;
while(n--,n2=n*n, n>0)
cout<<’’n=’’<<n<<’’ n^2=’’<<n2<<end1;
return 0;
}
```

Dasturdagi takrorlash operatori bajarilishida n soni 1 gacha kamayib boradi. Har bir qadamda n va uning kvadrati chop qilinadi. Shunga e’tibor berish kerakki, shart ifodasida operatorlarni yozilish ketma-ketligining ahamiyati bor, chunki, eng oxirgi operator takrorlash sharti sifatida qaraladi va n ning qiymati 0 bo‘lganda takrorlash tugaydi.

While takrorlash operatori yordamida samarali dastur kodi yozishga misol sifatida ikkita natural sonlarning eng katta umumiy bo’luvchisi (EKUB)ni Evklid algoritmi bilan topish masalasini ko’rishimiz mumkin:

```
var
   a,b:integer;
 begin
  Writeln(‘A va B natural sonlar EKUBini topish’);
  Write(‘A va B natural sonlarni kiriting:’);
   read(a, b);
   while a< >b do
   if a>b then a:=a-b else b:=b-a;
  Write(‘bu sonlar EKUBi=’,a);
 end.
```

```
int main()
{int a,b;
cout<<’’A va B natural sonlar EKUBini topish.\n’’;
cout<<’’A va B natural sonlarni kiriting:’’
cin>>a>>b;
while (a!=b) a>b? a-=b:b-=a;
cout<<’’bu sonlar
cin<<’’ bu sonlar EKUBi=’’<<a;
return 0;
}
```

Butun turdagi a va b qiymatlari oqimdan o‘qilgandan keyin ular qiymatlari toki o‘zaro teng bo‘lmaguncha takrorlash jarayoni ro‘y beradi. Takrorlashning har bir qadamida a va b sonlarining

kattasidan-kichigi ayriladi va ular tengligi tekshiriladi. Takrorlashdan keyingi ko‘rsatma vositasida a o‘zgaruvchisining qiymati natija sifatida chop etiladi.

Sharti keyin tekshiriladigan sikl operatori: Sharti keyin tekshiriladigan sikl operatori ham takrorlanishlar soni oldindan aniq bo‘lmagan hollarda takrorlanishni biror-bir shart asosida bajaradi. Oldin sikl tanasidagi operatorlar ketma-ketligi bajariladi. Berilgan shart keyin tekshiriladi.

Delphi da sharti oldin tekshiriladigan takrorlash operatorining berilgan sharti true (rost) bo‘lsa, boshqaruv sikldan keyingi operatorni bajarishga o‘tadi, aks holda, sikl takrorlanadi. Bu operatorning umumiy ko‘rinishi quyidagicha:

Repeat
<operatorlar guruhi>
Until <shart>

Bu yerda <shart>-mantiqiy ifoda, ‘true’ yoki ‘false’ qiymatni qabul qiladi; <operatorlar guruhi> - sikl tanasi bo‘lib, bir necha operatorlar ketma-ketligidan iborat bo‘lishi mumkin. Agar mantiqiy ifoda ‘false‘ qiymatni qabul qilsa siklda takrorlanish davom etadi, aks holda, to‘xtaydi.

C++ tilida sharti keyin tekshiriladigan operator sifatida *do-while* operatori ishlatiladi. Do-while takrorlash operatori quyidagi sintsksis:ga ega:

do <operatorlar guruhi>; while (<ifoda>);

```
var
     javob:char;
 begin
    Repeat
    …// sikl tanasi
     Writeln(‘jarayonni to‘xtatish(N):_’);
      Read(javob);
      until=N;
  end.
```

```
#include <iostream.h>
int main()
{
char javob;
do
{
…// sikl tanasi
cout<<’’ jarayonni to’xtashish(N):_’’;
cin>>javob;
}
while(javob !=”N”)
return 0;
}
```

Dastur toki “jarayonni to’xtashish (N):_’’so’roviga (N) javobi kiritilmaguncha davom etadi.

Shuni ta’kidlash kerakki *do-while* operatori *while* kalit so‘zidan keying ifodaning qiy mati rost bo‘lsa, takrorlanishni davom ettiradi, aks holda keying operatorlar bajarilishi davom etadi.

Masala. Har qanday 7 dan katta butun sondagi pul miqdoriga ega kupyuralarni 3 va 5 so‘mlik kupyuralarda berish mumkinligi isbotlansin. Qo‘yilgan masala p=3n+5m tenglamasini qanotlantiruvchi m, n sonlar juftliklarini topish masalasidir (p-pul miqdori). Bu shartning bajarilishini m va n o‘zgaruvchilarining mumkin bo‘lgan qiymatlarining barcha kombinatsiyalarida tekshirish zarur bo‘ladi.

```
var
  pul, n3, m5:Word;
  xato:boolean;
  begin
```

```
#include <iostream.h>
int main()
{
unsigned int pul;  //pu1-  kiritiladigan pul miqdori
```

```
    xato:=false;
    repeat
    if xato then
    Writeln('Kiritilgan pul qiymati 7 dan kichik !');
    xato:=true ;
   //keyingi takrorlash xato hisoblanadi.
   Write('Pul qiymatini kiriting(>7):');
    read(pul);
    until pul>7;
    n3:=0; //birorta ham 3 so'mlik yo'q
    repeat
    m5:=0//birorta ham 5 so'mlik yo'q
    repeat
    if 3*n3+5*m5=pul then
Writeln(n3,' ta 3 so`mlik va ', m5,' ta 5 so`mlik');
inc(m5);
until 3*n3+5*m5>pul;
inc(n3);
until 3*n3>pul;
end.
```

```
unsigned int n3,m5; //n-3 so'mliklar , m-5 so'mliklar soni
bool xato=fa1se;  //pu1 qiymatini kiritilgandagi xatolik
do
{
if (xato) cout<<''kiritilgan pul qiymati 7 dan kichik !'';
xato=true ; //keyingi takrorlash xato hisoblanadi
cout<<''\npul qiymatini kiriting (>7):'';
cin>>pul;
}
while (pul<=7); // toki 7 sonidan katta son kiritulguncha
n3=0 ; //birorta ham 3 so'mlik yo'q
do
{
m5=0;  // birorta ham 5 so'mlik yo'q
do
{
if (3*n3+5*m5==pul)
cout<<n3<<''ta 3 so'mlik + ''<<m5<<'' ta 5 so'mlik\n'';
m5++  //5so'mliklar bittaga oshiriladi
}
while(3*n3+5*m5<=pul);
n3++;  //3 so'mliklar bittaga oshiriladi
}
while(3*n3<=pul);
return 0;
}
```

Dastur pul qiymatini kiritishni so'raydi(pu1 o'zgaruvchisiga). Agar pul qiymati 7 sonidan kichik bo'lsa, bu haqda xabar beriladi va takror ravishda qiymat kiritish talab qilinadi. Pul qiymati 7 dan katta bo'lganda, 3 va 5 so'mliklarning mumkin bo'lgan to'la kombinatsiyasini amalga oshirish uchun ichma-ich takrorlashlar amalga oshiriladi. Tashqi takrorlash n3 (3 so'mliklar miqdori) bo'yicha, ichki takrorlash esa m5 (5 so'mliklar miqdori) bo'yicha, toki bu miqdordagi pullar qiymati pul qiymatidan oshib ketmaguncha davom etadi. Ichki takrorlashda m5 o'zgaruvchisining har bir qiymatida «3*n3+5*m5=pu1» sharti tekshiriladi, agar u o'rinli bo'lsa, yechim varianti sifatida n3 va m5 o'zgaruvchilar qiymatlari chop etiladi.

Pul qiymati 30 so'm kiritilganda, ekranga

0 ta 3 so'mlik +6 ta 5 so'mlik chop etiladi.

5 ta 3 so'mlik +6 ta 5 so'mlik

10 ta 3 so'mlik +0 ta 5 so'mlik

yechim variantlari chop etiladi.

2.5. Goto operatori va nishonlar.

Dasturda shunday holatlar bo‘ladiki, operatorlarning bajarilishiga qarab dasturning u yoki bu qismiga to‘g‘ridan-to‘g‘ri bajarishni uzatish ehtiyoji tug‘iladi. Bunday holatlarda shartsiz o‘tish operatoridan foydalanish mumkin.

Delphi va C++ tillarida shartsiz o‘tish operatorining sintsksis:i quyidagicha:

Goto <nishon>;

Bu yerda <nishon> - belgi(metka) bo‘lib identifikator (Delphi da butun son ham) bo‘lishi mumkin. Goto - o‘tish ma’nosini bildiradi.

<nishon> Delphida dasturning bosh qismida *label* kalit so‘zi yordamida e’lon qilingan bo‘lishi shart. <nishon> bajarilish uzatiladigan joyga <nishon>: shaklida qo‘yiladi.

Misol:

……
Goto L2;
……
L2: C:=x*y;
……

Delphi va C++ tillarida e’lon qilingan nishonlar qayerda e’lon qilinishiga qarab faqat e’lon qilingan (funksiya, qism dastur) sohada ko‘rinadi.

Goto operatorida qo‘llaniladigan identifikatorlar C++ tilida Delphidagi kabi e’lon qilinmaydi.

Shuni ta’kidlash lozimki C++ tilida dastur tuzish jarayonidagi ayrim hollarda goto operatoridan foydalanib «sakrab o‘tishi» hisobiga xatoliklar yuzaga kelishi mumkin. Masalan,

```
int i=0;
i++; if(i) goto m;

int j;
m: j+=I;
```

Bu misoldagi *goto* operatorining bajarilishi xatolikka olib keladi, chunki *j* e’lon qilinmay qoladi.

Shartsiz o‘tish operatori dastur tuzishdagi kuchli va shu bilan birga xavfli vositalardan biri hisoblanadi. Kuchliligi shundaki, u yordamida algoritmning «boshi berk» joylaridan chiqib ketish mumkin. Ikkinchi tomondan, bloklarning ichiga o‘tish, masalan takrorlash operatorlarining ichiga «sakrab» kirish kutilmagan holatlarni yuzaga keltirishi mumkin. Shu sababli, imkon qadar *goto* operatoridan foydalanmaslik kerak, ishlatilgan taqdirda ham qo’yidagi qoidalarga amal qilish zarur: blok ichiga, if…else va tanlash operatorlari ichiga hamda takrorlash operatorlari tanasiga tashqaridan kirish mumkin emas.

Garchi, nishon yordamida dasturning ixtiyoriy joyiga o‘tish mumkin bo’lsa ham, boshlang‘ich qiymat berish e’lonlaridan sakrab o‘tish man etiladi, lekin bloklardan sakrab o‘tish mumkin.

Xususan, nishon yordamida ichki blokdan tashqi blokka va tashqi blokdan ichki blokka o‘tishga C++ tili ruxsat beradi:

```
{…
goto ABC:
…
```

```
{int i=15;
…
ABC:
…
goto XYZ;
int y=10;
…
goto KLM;
…}
…
int k=0;
…
KLM:
…}
```

Lekin, yuqorida keltirilgan misoldagi barcha o‘tishlar mazmunan xato hisoblanadi.

Quyidagi dasturda ikkita natural sonlar EKUBini topish masalasidagi takrorlash jarayonini nishon va goto operatori vositasida amalga oshirish ko‘rsatilgan:

```
label  nishon;
var
   a,b:integer;
   Writeln(‘A va B natural sonlar EKUBini topish’);
   Write(‘A va B natural sonlarni kiriting:’);
    read(a,b);
    nishon: if a=b then
    Writeln(‘Bu sonlar EKUBi=’,a);
     else   begin
     if a>b then a:=a-b else b:=b-a;
     goto nishon;
     end.
```

```
int main()
{
int a,b;
cout<<”A va B natural sonlar EKUBini topish.\n”;
cout<<”A va B natural sonlarni kiriting: “
cin>>a>>b;
nishon:
if (a==b)
{
cout<<”Bu sonlar EKUBi=”<<a;
return 0;
}
a>b?a-=b:b-=a;
goto nishon;
}
```

Dasturdagi nishon bilan belgilangan operatorda *a* va *b* o‘zgaruvchilarining qiymati tengligi tekshiriladi. Agar ular teng bo‘lsa, ixtiyoriy bittasi, masalan *a* o‘zgaruvchisidagi son EKUB bo‘ladi va dastur ishini yakunlaydi. Aks holda, bu sonlarning kattasidan kichigi ayriladi va goto orqali ularning tengligi tekshiriluvchi shart operatoriga o‘tiladi. Takrorlash jarayoni *a* va *b* sonlar o‘zaro teng bo‘lguncha davom etadi.

Shuni qayd etish kerakki, bu masalani takrorlash operatorlari yordamida bajarish ancha samarali hisoblanadi.

2.6. break operatori

Takrorlash operatorlarining bajarilishida shunday holatlar yuzaga kelishi mumkinki, unda qaysidir qadamda, takrorlashni yakuniga yetkazmasdan takrorlashdan chiqish zarurati bo‘lishi mumkin. Boshqacha aytganda takrorlashni «uzish» kerak bo‘lishi mumkin. Bunda break

operatoridan foydalaniladi. Break operatorini takrorlash operatori tanasining ixtiyoriy (zarur) joylariga qo‘yish orqali shu joylardan takrorlashdan chiqishni amalga oshirish mumkin. E’tibor beradigan bo’lsak switch-case operatorining tub mohiyatiga ham break operatorini qo’llash orqali erishilgan.

Ichma – ich joylashgan takrorlash va switch operatorlarida break operatori faqat o‘zi joylashgan blokdan chiqish imkoniyatini beradi.

Quyidagi dasturda ikkita ichma-ich joylashgan takrorlash operatoridan foydalangan holda foydalanuvchi tomonidan kiritilgan qandaydir sonni 3 va 7 sonlariga nisbatan qanday oraliqqa tushishi aniqlanadi .Tashqi takrorlashda ‘’son kiriting (0-to’xtash):_’’ so‘rovi beriladi va javob javob_son o‘zgaruvchisiga o‘qiladi. Agar son noldan farqli bo‘lsa, ichki takrorlash operatorida bu sonning qandaydir tushishi aniqlanib, shu haqida xabar beriladi va ichki operatoridan chiqiladi. Tashqi takrorlashdagi so‘rovga javob tariqasida 0 kiritilsa, dastur o‘z ishini tugatadi.

```
#include <iostream.h>
int main()
{
int javob_son=0;
do
{
while (javob_son)
{
if (javob_son<3)
{cout<<’’3 kichik !’’; break;}
if(3<=javob_son&& javob_son<=7)
{cout<<’’3 va 7 oralig’da !’’; break;}
if (javob_son>7)
{cout<<”7 dan katta !”; break;}

}
cout<<”\nSon kiriting (0-to’xtash):_”;
cin>>javob_son;
}
while(javob_son !=0)
return 0
}
```

Amaliyotda break operatoridan cheksiz takrorlashdan chiqishda foydalaniladi.

```
For (;;)
{
//  1-shart
if (…)
{
…
break ;
}
//   2-  shart
if (…)
{
…
break;
}
```

```
…
}
```

Bu misolda cheksiz for takrorlashidan 1 yoki 2- shart bajarilganda chiqiladi,

Masala. Ishorasiz butun sonlar ketma-ketligi 0 qiymati bilan tugaydi. Bu yerda 0 ketma-ketlik hadi hisoblanmaydi. Ketma-ketlikni kamaymaydigan holda tartiblangan yoki yo‘qdigi aniqlansin.

```
#include <iostream.h>
int main()
{
unsigned int Ai_1=0,Ai;
cout<<’’ sonlar ketma-ketligini kiriting’’
cout<<(0-tugash alomati):\n’’;
cin>>Ai; //   ketma-ketlikning birinchi hadi
while(Ai)
{Ai_1>Ai;
cin>>Ai; //  navbatdagi had
if (Ai_1>Ai) break;
}
if (Ai_1)
{
cout<<’’ketma-ketlik kamaymaydigan holda tartiblangan’’;
if(!Ai)cout<<’’emas!’’;
else cout<<’’!’’;
}

}
else cout<<’’ketma-ketlik bo’sh!’’;
return 0;
}
```

Dastur ishga tushganda, boshida ketma-ketlikning birinchi hadi alohida o‘qib olinadi (Ai o‘zgaruvchisiga). Keyin Ai qiymati nolga teng bo’lmaguncha takrorlash operatori amal qiladi. Takrorlash tanasida Ai qiymati oldingi qiymat sifatida Ai_1 o‘zgaruvchisida eslab qolinadi va navbatdagi had Ai o‘zgaruvchisiga o‘qiladi. Agar oldingi had navbatdagi haddan katta bo‘lsa, break operatori yordamida takrorlash jarayoni uziladi va boshqaruv takrorlashdan keyingi shart operatoriga o‘tadi. Bu yerdagi shart operatorlari mazmuni quyidagicha agar Ai_1 noldan farqli bo‘lsa, ketma-ketlikning kamida bitta hadi kiritilgan bo’ladi (ketma-ketlik mavjud) va oxirgi kiritilgan had tekshiriladi. O’z navbatida agar Ai noldan farqli bo‘lsa, bu holat had1ar o’rtasida kamaymaslik sharti bajarilmaganligi sababli hadlarni
kiritish jarayoni uzilganligini bildiradi va bu haqda xabar chop etiladi. Aks holda ketma-ketlikni kamaymaydigan holda tartiblangan bo’ladi.

Continue operatori

Continue operatori xuddi break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin takrorlashdan chiqib ketmasdan keyingi qadamiga «sakrab» o‘tishini ta’minlaydi.

Continue operatorini qo‘llanishiga misol tariqasida 2 va 50 sonlar oralig‘idagi tub sonlarni topadigan dastur matnini keltiramiz.

```
#include <iostream.h>
int main()
{
```

```
bool bulinadi=false;
for (int i=2; i<50; i++)
{
for (int j=2; j<i/2; j++)
{if (i%j) continue;
else {bulinadi=true; break;}
}
// break bajarilganda boshqaruv o’tadigan joy
if (!bulinadi ) cout <<i<<” “;
bulinadi=false;
}
return 0;
}
```

Keltirilgan dasturda qo‘yilgan masala ichma-ich joylashgan ikkita takrorlash operatorlari yordamida yechilgan. Birinchi takrorlash operatori 2 dan 50 gacha sonlarni hosil qilishga xizmat qiladi. Ichki takrorlash esa har bir hosil qilinayotgan sonni 2 sonidan toki shu sonning yarmigacha bo‘lgan sonlarga bo‘lib, qoldig‘ini tekshiradi, agar qoldiq 0 sonidan farqli bo‘lsa, navbatdagi songa bo‘lish davom etadi, aks holda *bo‘linadi* o‘zgaruvchisiga *true* qiymat berib, ichki takrorlash uziladi (son o‘zining yarmigacha bo‘lgan qandaydir songa bo‘linar ekan, demak u tub emas va keyingi sonlarga bo‘lib tekshirishga hojat yo‘q). Ichki takrorlashdan chiqqandan keyin *bo‘linadi* qiymati false bo‘lsa (!bo`linadi), son tub bo‘ladi va u chop qilinadi.

Adabiyotlar.

1. Б. Страуструп. Язык программирования С++. Специальное издание.-М.:ООО «Бином-Пресс», 2006.-1104 с.
2. Павловская Т.А. С++. Программирование на языке высокого уровня – СПб.: Питер. 2005.-461 с.
3. Подбельский В.В. Язык С++.- М.; Финансы и статистика- 2003 562с.
4. Павловская Т.С. Щупак Ю.С. С/С++. Структурное программирование. Практикум.-СПб.: Питер,2002-240с
5. Павловская Т.С. Щупак Ю.С. С++. Объектно- ориентированное программ-мирование. Практикум.-СПб.: Питер,2005-265с
6. Глушаков С.В., Коваль А.В., Смирнов С.В. Язык программирования С++: Учебный курс.-Харьков: Фолио; М.: ООО «Издательство АСТ», 2001.-500с.
7. Ш.Ф. Мадрахимов, С. М. Гайназаров С++ тилида программалаш асослари. Т. 2009.
8. А.Файсман. Профессиональное программирование на Турбо Паскале. 1992.
9. М.В.Култин. Программирование в Турбо Паскале и Делпхи, Санкт-Петербурге, 2002.
10. Sh. A. Nazirov va boshqalar Delphi tilida dasturlash soslari. T. 2007.
11. WWW.Intuit.ru. Интернет-Университет информационных технологий. Москва.
12. Пильшиков В.Н. Упражнения по языку Паскаль-М.: МГУ, 1986.
13. Гофман В. Э., Хомоненко А.Д. Delphi 5. - СПб.: БХВ-Санкт-Петербург, 2000. -800с.
14. Немнюгин С.А. Turbo pascal, учебник. Изд. Питер., 2001, -496 с.
15. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо-паскаль. (версия 5.5).М.:МАИ,1992.-576с.
16. Абрамов С.А.,Гнезделова Капустина Е.Н.и др. Задачи по программ-мированию. - М.: Наука, 1988.
17. Вирт Н. Алгоритмы + структуры данных = программа.-М.:Мир,1985.-405с.
18. Информатика. Базовой курс. Учебник для Вузов., Санк-Петербург, 2001. под редакцией С.В.Симоновича.

19. Informatika va programmalsh.O’quv qo’llanma. Mualliflar: A.A.Xaldjigitov, Sh.F.Madraximov, U.E.Adamboev, O’zMU, 2005 yil, 145 bet.
20. O.Shukurov, F.Qorayev, E.Eshboyev, B.Shovaliyev “Programmalashdan masalalar to’plami”. Toshkent 2008,160 bet.
21. A.Xoldjigitov va boshqalar “Pascal tilida programmalash bo’yicha masalalar to’plami” Toshkent 2002.