Қарақалпақ мәмлекетлик университети

Әмелий математика кафедрасы

Паскал программалаштырыў тили

бойынша лекция текстлери

Аннотация

Программастырыў тийкарлары пәни студентлерге қойылған мәселени компьютер программасын дүзиўге үйретиў болып табылады. Усы мақсетте программаластырыў тиллери ҳәм орталықлары ҳаққында улыўма түсиниклер бериледи ҳәм бул тиллерден пайдаланыўға үйретиледи.

Бул лекцияда программаластырыўға кирисиўдиң теориялық тийкары болғанг алгоритмлерге өз алдына итибар қаратылған. Бул жерде алгоритмлерди классификациялаўда кейинги ўақытлардан компьютерде әмелге асырыў ушын зәрүр болған бир қатар математикалық түсиниклер – тәкирарлаў, жәрдемши алгоритм, рекурсия, яд, массив, индекс, параметр ҳ.т.б. киритилип ҳәр қыйлы класс мәселелериниң алгоритмлери дүзиледи.

МАЗМУНЫ

1-ЛЕКЦИЯ. ПРОГРАММАЛАСТЫРЫЎ ТИЙКАРЛАРЫ

Жоба:

1. Программаластырыў тили. Программаластырыў тили дәрежелери. Транслятор, интерпретатор ҳәм компилятор. Программаластырыў системалары.
2. Лекцияда баян етилетуғын тийкарғы(таяныш) мағлыўматлары: Программаластырыў тиллери ҳәм олардың жаратылыў тарийхы. Программаластырыў тили дәрежелери. Транслятор, интерпретатор ҳәм компилятор.

Алгоритмлердиң программа көринисинде жазылыўы ҳәм бул көринистеги басқа жазыўлар менен қандай парық қылады?

Әмелиятта алгоритм орынлаўшысы компьютер есапланады. Соның ушын компьютер ушын дүзилген алгоритм ол «түсинетуғын» тилде сүўретлениўи керек.

Орынлаўшы (компьютер) берилген буйрықларды анық ҳәм туўры орынлаўы ушын оған берилетуғын буйрықлар жазылыўы анық ҳәм орынлаўшы тәрепинен бир қыйлы түсинилиўи керек.

Сол себепли компьютер түсинетуғын тил программаластырыў тили, бул тилде жазылған алгоритм болса **компьютер программасы** делинеди.

Программаластырыў тили

Программаластырыў тиллери тийкарынан 2-жәҳән урысынан кейин жаратыла басланды. Бирақ оның тарийхы бир қанша алыс жылларға барып тақалады.

Археологиялық қазылмаларда табылған ылайдан исленген тахтайшада буннан 3800 жыл алдын (эрамыздан 1800-жыллар) Вавилонда процент (%) пенен байланыслы қурамалы әмеллер алгоритми келтирилген. Онда анық мәселе исленген болып, егер бийдай жылда 20% тен асып барса, оның муғдары еки мәрте өсиўи ушын неше жыл ҳәм ай керек болыўы алгоритми жазылған.

XIX әсир француз ойлап табыўшысы **Жозеф Мари Жаккард** 1804 жыл жуқа таўар ислеп шығыў процессинде станоклар ушын перфокартаны еслетиўши лента ислеткен ҳәм усының менен **перфокартаға** тийкар салған еди.

1836 жылы инглис **Шарлз Беббидж** ҳәзирги компьютерлердиң әўлады болған аналитикалық машина ислеп шығыўға кирисиўди ҳәм бул мәселени теориялық жақтан шешти. Бул машинаның тийкарғы қәсийети оның программа тийкарында ислеўи ҳәм есап-китап нәтийжелерин «еслеп» қалыўында еди.

1843 жылы инглис математиги **Огаста Ада Байрон (Лавлейс)** – шайыр лорд Байронниң қызы – аналитикалық машина буйрықлар тийкарында ислеўи кереклигин тастыйықлады. Ол берилген шәртлер орынланбағанша адымлар избе-излигин тәмийнлеўши буйрықларды жазды. (Бул ҳәзирги күнде тәкирарлаў (цикл) операторы делинеди). Буның менен ол программаластырыў ҳәм оның тилине тийкар салды.

Бул ҳәм басқа жаратылыўлар, компьютер толық жаратылғаннан кейин, оған зәрүр болған тил жаратыўды талап етип қойды.

Дүнья адамлары ортасындағы қарым-қатнаста жалғыз тил болмағанындай, ҳәр түрли компьютерлердиң жаратылыўы ҳәм олардың түрли тараўларда қолланылыўы, жалғыз программаластырыў тилин жаратыў мүмкин емеслигин көрсетти.

Перфокарта

Алан Тьюринг

1936 ж. Алан Тьюринг ҳәм оннан айрықша **Эмил Пост *абстракт есаплаў машинасының концепциясы идеясын алға қойды ҳәм жаратты.*** Олар ҳәр қандай мәселени шешиў алгоритмин дүзиў имканияты бар болса, онда автоматлар тәрепинен оларды шешиўдиң принципиал тийкарлары барлығын дәлилледи.

1941ж. Конрад Цузе электромеханик элементлер тийкарында ислеўши биринши универсал компьютерди жаратты. Бул компьютер екилик санақ системасына тийкарланған санлар менен ислеў имканиятын жаратты.

1949ж. дүньяда биринши болып Англияда **Морис Уилкс** басшылығында программаны өз ядында сақлай алатуғын ***ЭДСАК*** компьютерин жаратты.

1955-1959ж. Россиялы алымлар А.А. Ляпунов, С.С. Камынин, Е.З. Любимский, А.П. Ершов, Л.Н. Королев, В.М. Курошкин, М.Р. Шура-Бура ҳәм басқалар *«программалаўшы программалар»*, яғный трансляторға уқсас программа жаратты. В.В. Мартынюк болса программаларды жаратыў ҳәм отладка қылыўды тезлестириў мақсетинде *белгили кодлар системасын жаратты*.

1955-1959ж. Программаластырыў теориясы (А.А. Ляпунов, Ю.И. Янов, А.А. Марков, Л.А. Калужин) ҳәм санлы методларға (В.М. Глушков, А.А. Самарский, А.Н. Тихонов) тийкар салынды. Пикирлеў механизми схемасын ҳәм генетика процесслерин моделлестириў, кеселликлер диагностикасының алгоритмлери ислеп шығылды (А.А. Ляпунов, Б.В. Гнеденко, Н.М. Амосов, А.Г. Ивахненко, В.А. Ковалевский и др.).

Джон Бекус

1957ж. Джон Бекус *Фортран* тилиниң тийкаршысы.

1959ж. Узақ жыллар даўамында программаластырыў тараўында стандартқа айланған ***Алгол*** тили жаратылды.

1965ж. Джон Кемени ҳәм **Томас Курцлар** (Дортмунд колледжи, АҚШ) тәрепинен ***Basic*** программаластырыў тили жаратылды.

Никлаус Вирт

1970ж. Швейцар алымы **Никлаус Вирт *Pascal*** программаластырыў тилин жаратты.

1971ж. Француз алымы **Алан Колмари** логикалық программаластырыў тили ***Пролог*** (Programmin in logos)ти жаратты.

Деннис Ритчи

1972ж. Белл Лабораториес хызметкери **Деннис Ритчи** ***C*** программаластырыў тилин жаратады.

1973ж. **Кен Томпсон** ҳәм **Деннис Ритчи** ***Unix*** операциялық системасын жаратты.

1974ж. Intel фирмасы 4500 транзистор тийкарында биринши сегиз разрядлы 8080 универсал микропроцессорды жаратты.

Алтаир

1974ж. АҚШ әскерий офицери ҳәм инженери **Эдвард Роберц 8080** микропроцессор тийкарында ***Алтаир*** микрокомпьютерин жаратты.

Билл Гейц ҳәм Пол Аллен

1975ж. Жас программист **Пол Аллен** ҳәм Горвард университети студенти **Билл Гейцлер** Алтаир компьютери ушын ***Basic*** тилин ислеп шықты.

Кейин ала олар дүньяда программалар ислеп шығарыў бойынша ең үлкен ***Microsoft*** фирмасын дүзди.

1979 ж. SoftWare Arc фирмасы ***VisiCals*** (*Visible Calculator*) жеке компьютерлер ушын программалар пакетин жаратты.

Ҳақыйқаттанда программаластырыў тиллери жаратылыўы бойынша үш топардан ибарат:

Программаластырыў тили дәрежелери

Бүгинги күнде жүзлеп программаластырыў тиллеринен пайдаланылады. Ҳәр бир программаластырыў тили белгили бир тараў яки бағдарға байланыслы мәселелерди шешиў ушын арналған.

Бизге белгили, ҳәр қандай алгоритм – көрсетпелер избе-излиги ҳәм бул көрсетпелер орынланғаннан соң белгили бир нәтийжеге ерисиледи. Программаластырыў тили дәрежеси буйрық ҳәм көрсетпелер (детализация) менен белгиленеди - қаншелли (детализация) кемирек ҳәм әпиўайырақ болса, сонша тилдиң дәрежеси жоқары болады.

Төменги дәрежедеги программаластырыў тиллери компьютер қурылмалары менен туўры байланыста болып, буйрықлар олардың кодлары жәрдеминде жазылады. Бул сыяқлы буйрықлардан ибарат программалар үлкен көлемли болып, оларды редакторлаў бир қанша қыйын жумыс еди. Дәслепки компьютерлер (Эниак, МЭСМ ҳәм басқалар) усы тилде ислер еди.

Орта дәрежедеги программаластырыў тиллери буйрықларында тек цифрлар емес, бәлким базы инсан сөйлеўине жақын сөзлер қолланыла басланды ҳәм олар көбинесе ***ассемблер*** тиллери депте жүритиледи.

Бул тилдиң жақсы қәсийети сонда, олар жәрдеминде дүзилген программа басқа тиллерде дүзилгенинен 10-15 мәрте тезирек ислейди. Онда басқа дәрежедеги тиллерде болмаған әмеллер бар. Бундай тиллерге **АВТОКОД-БЕМШ, МАДЛЕН** ҳәм басқалар киреди. Ассемблер тиллеринде буйрықлар қысқартылған сөзлер ямаса сөзлер жыйындысынан ибарат болып, оларды ***мнемокодлар*** депте атайды.

Жоқары дәрежедеги программаластырыў тиллери бирқанша раўажланған дүзилиске ийе болып, онда ислетилетуғын хызметши сөзлер инсан сөйлеўине жүдә жақынластырылған. Бул тилде пайдаланыўшы әмеллерди избе-из жазып бара береди, компьютер оны керекли көриниске өткерип алады.

Бул дәрежедеги тиллер компьютер техникалық тәмийинлениўини раўажланыўы менен байланыслы ҳәм өз тарийхына ийе. Дәслепки бул дәрежедеги тил ***Планкалкюл*** деп аталып, ол ***1946 жылы*** немис илимпазы ***Конрад Цузе*** тәрепинен жаратылды.

Жаратылыў жылы	Тил аты
1946	Планкалкюл
1949	Қысқаша код
1950	“Едсак” ассемблер тили
1950	АО
1953	Автокод
1955	“Тез кодлаў”
1956	Ф-2, Флоу-метик
1957	ИПЛ-1, Мат-метик
1958	Фортран
1959	Алгол 58, АПТ, ЛИСП, Кобол
1960	Алгол 60
1964	ПЛ/1, Basic
1965	Алгол 65
1967	Лого
1968	Алгол 68
1969	АПЛ
1970	Pascal
1971	Фортран
1972	Пролог, Си
1979	Ада
1980	Смолток
1981	Модула-2

Программаластырыў тиллери дәрежелерин төмендеги критериялар бойынша ажыратыў мүмкин:

- машина тили;
- ассемблерлер;

- жоқары дәрежедеги программаластырыў тиллери;

Машина тили ҳәм ассемблерлер – төменги дәрежедеги программаластырыў тиллери есапланады. Бул тиллер жәрдеминде программа дүзилгенде мағлыўматларды қайта ислеў.

Жоқары дәрежедеги программаластырыў тиллери төмендеги түрлерге бөлинеди:

- **Алгоритмлик** (Basic, Pascal, C ҳ.т.б.);
- **Логикалық** (Prolog, Lisp ҳ.т.б.);
- **Объектке бағдарланған** (Object Pascal, C++, Java ҳ.т.б.);

Транслятор, компилятор ҳәм интерпретатор

Транслятор (инг*л. Translator* — аўдармашы) – қандайда бир жоқары программаластырыў тилинде дүзилген программаны машина тилине өткериўши программа-аўдармашы.

Трансляторлар компилятор ҳәм интерпретатор көринисинде болыўы мүмкин. Компилятор ҳәм интерпретаторлар программаны орынлаўда парық қылады.

Компилятор (инг*л. compiler* — жыйнаўшы) программаны *толық* оқыйды ҳәм оны аўдарады (машина тилине аўдарады).

Мәселен:

Turbo Basic ҳәм ***Turbo Pascal 7.0*** программаластырыў тиллери – *компилятор* есапланады.

Интерпретатор (инг*л. interpreter*— шолыўшы, аўызша аўдарма) программаны аўдарады ҳәм қатарма-қатар орынлайды.

Мәселен:

Quick Basic программаластырыў тиллери – **компилятор** есапланады.

Программа компиляция қылынғаннан соң программаластырыў тилинде жазылған программа ҳәм компиляторға зәрүрлик қалмайды. Мәселен, *info.bas* файлы компиляция етилгеннен соң *info.exe* файлына өзгереди ҳәм бул файл компиляторсыз да ислеўи мүмкин. Егер *info.bas* файлын интерпретатор қайта ислегенде ҳәр бир мәрте программа жумысын орынланыўы ушын интерпретатор иске түсиўи

шәрт. Усы себепли *компилятор* ҳәм *интерпретаторлардың* ислеў принципи бойынша бир қанша парық қылады.

Компилиляция етилген программалар тезирек ислейди, лекин *интерпретация* қылынып атырған программаларға өзгертиў киритиў аңсат.

Ҳәр бир программаластырыў тили қайсы мақсетте жаратилғанлығына қарап компиляция яки интерпретацияға арналған болыўы мүмкин. Мәселен, *Pascal* программаластырыў тили жүдә қурамалы мәселелер программасын дүзиўге арналған болып, бундай мәселелерди шешиў ушын программа орынланыўы тезлиги жоқары болыўы талап етиледи.

Basic программаластырыў тили программаластырыўды үйрениўши ҳәм бул тараўда биринши адым таслап атырғанларға арналған болып, бул тилде программаны орынлаў қатарма-қатар әмелге асырылғанлығы себепли баслаўшылар ушын жүда үлкен қолайлықлар жаратады.

Базыда бир программаластырыў тили ушын да ҳәм *компилятор*, ҳәм *интерпретатор* бар. Бундай жағдайда программаны жаратыў ҳәм сазлаўда интерпретатордан пайдаланылады, соң программа жумысын тезлестириў ушын бул файл компиляция етиледи.

Мәселен:

QuickBasic 4.5 ҳәм ***Visual Basic 6.0*** программаластырыў тиллери – *интерпретатор* + *компилятор* есапланады.

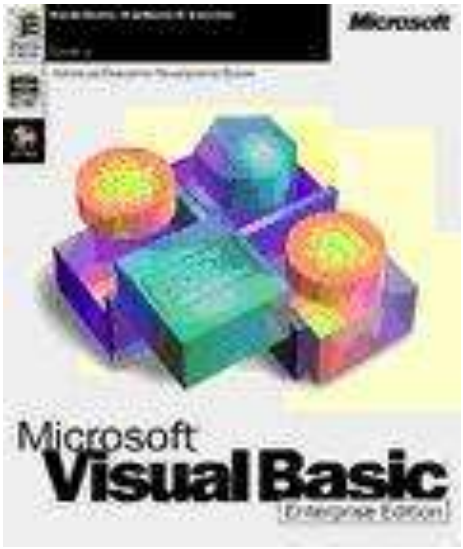
Программаластырыў системалары

Программаластырыў системалары — белгили бир программаластырыў тилинде жаңа программа жаратыў системалары.

Заманагөй программаластырыў системалары пайдаланыўшыға программа дүзиў ушын көплеп қолай имканиятлар жаратады. Бундай қурал ҳәм имканиятлар:

- ✓ *компилятор* ҳәм *интерпретатор*;
- ✓ программа текстин жаратыў ҳәм редакторлаў қурылмалары;
- ✓ стандарт программалар библиотекасы ҳәм функциялар;

- ✓ дүзилген программаға өзгертиў киритиў ҳәм сазлаў қурылмалары
- ✓ пайдаланыўшы ушын қолай интерфейс;
- ✓ көпайналы ислеў режими;
- ✓ график имканиятлар библиотекасы; библиотекалар менен ислеў утилиталар;
- ✓ ассемблер;
- ✓ жәрдем файллары;
- ✓ басқа специфик имканиятлар.



Кең тарқалған программаластырыў системалары–***Turbo Basic, Quick Basic, Turbo Pascal, Turbo C.***

Бүгинги күнде *Windows* операциялық системасы орталығында ислейтуғын программалар дүзиўге арналған программаластырыў системаларына зәрүрлик артпақта.

Microsoft Visual Basic пакети — визуал қурал (анимация, аудио-, видео-)лар жәрдеминде Windows-программалар дүзиў ушын қолай ҳәм кең тарқалған программаластырыў тили есапланады.

Төменде A=(a_1, a_2, ..., a_N) бир өлшемли массив элементлери S қосындысын есаплаўшы программа Basic ҳәм Pascal тиллеринде келтирилген.

Basic тили 1965 ж. Джон Кемени ҳәм Томас Курцлар (Дотмунд коллежи, АҚШ) тәрепинен жаратылған болып, программа дүзиўди үйрениўшилер ушын әпиўайы программалар дүзиўге арналған.

Basic тилиндеги программа

```
10 INPUT "N = "; N
20 DIM A(N)
30 FOR I = 1 TO N
40 PRINT "A(";I;") =";
50 INPUT A(I)
```

Basic тилиниң жүзлеп версиялары бар. Basic программаластырыў тили семясында ең әпиўайы әмеллерди орынлаўшы версияларынан баслап қурамалы ўазыйпаларды орынлаўшы версиялары киреди.

1975ж. Жас программист **Пол Аллен** ҳәм Гарвард университети студенти **Билл Гейцлер** Алтаир компьютери ушын Basic тилин жаратты. Кейин олар компьютер әлеминде программалық тәмийнат жаратыўшы ең үлкен компания – *Microsoft* фирмасына тийкар салады.

```
60 NEXT И
70 S = 0
80 FOR I = 1 TO N
90 S = S + A(I)
100 NEXT I
110 PRINT “QOSINDI=”; S
```

1970ж. Швейцариялы **Никлаус Вирт** ***Pascal*** программаластырыў тилин жаратты.

Pascal программаластырыў тили студентлерге программаластырыў тилин үйретиў мақсетинде жаратылған еди. Pascal программаластырыў тили өзиниң басланғыш көринисинде шекленген имканиятларға ийе еди, *Turbo Pascal* тили болса жүдә күшли программаластырыў тили есапланады.

1972ж Белл Лабораториес инженери **Деннис Ритсчи** тәрепинен ***Си*** программаластырыў тили жаратылады.

Pascal тилиндеги программа

```
program qosindi;
type mac = array [1..100] of real;
var a : mac;
  i, n: integer;
  s : real;
begin
  readln(n);
  for i : = 1 to 10 do
    readln(a[i]);
  s : = 0;
  for i : = 1 to 10 do
    s : = s + a[i];
  writeln(‘c = ‘, c:8:2);
end.
```

2-ЛЕКЦИЯ. PASCAL ПРОГРАММАЛАСТЫРЫЎ ТИЛИ ҲАҚҚЫНДА УЛЫЎМА ТҮСИНИК. ТИЛДИҢ ЭЛЕМЕНТЛЕРИ ҲӘМ СТРУКТУРАСЫ

Жоба:

1. Pascal тилиниң жаратылыўы
2. Pascal тили версиялары
3. Операторалар
4. Атамалар ҳәм идентификаторлар
5. Дағазалар
6. Pascal-программасының структурасы

Pascal программаластырыў тили Н.Вирт тәрепинен жаратылған. Программаластырыў тарийхында пайдаланыўшылар санының өсиў дәрежеси бойынша бул тилге тең келетуғын тил жоқ. Буны Pascal тилинде алгоритмлердиң фундаментал ҳәм ең әҳмийетли концепцияларының айдын ҳәм түсиниўге аңсат формада берилгенлиги менен түсиндириў мүмкин. Программа дүзиў барысында программист өзиниң қәлеўиндеги программаластырыў тили тәрепинен берилетуғын категориялар менен пикир жүритеди.

Pascal тилин жаратыўда еки мақсет нәзерде тутылған:

- ✓ түсинерли ҳәм табийғый рәўиште аңлатылған қатар фундаментал түсиниклерге тийкарланған программаластырыўды үйретиўге қолай болған системалы предмет сыпатында жаратыў;
- ✓ бар болған есаплаў машиналарында тилди пайдаланыўдиң исенимлилиги ҳәм нәтийжелилиги.

Pascal программаластырыў тили ең әҳмийетли еки принципке тийкарланған:

- ✓ структуралы программаластырыў принципи (оған классик структуралы программаластырыўдың тийкарғы басқарыўшы структураларын әмелге асырыў қурылмалары киреди);
- ✓ берилгенлерди аңлатыўдың структуралы принципи (ол К. Хоар тәрепинен усынылған структуралы түрлердиң толық топламын әпиўайы өзгериўшилер, массивлер, избе-из файллар, жазыўлар,

вариантлы жазыўлар, көрсеткишлер ҳәм оннан берилгенниң жаңа түрлерин жаратыўдың пайда болған қуралларын өз ишине алады);

Pascal программалаστырыў тили, заманагөй программалаστырыўда үлкен әҳмийетке ийе болған, программалардың туўрылығын дәлиллеўши аналитикалық усыллардың пайда болыўында үлкен рол ойнады. Ол ҳәзирги программалаστырыўдың туўрылығын дәлиллеўге имкан беретуғын программалаστырыў системаларына ийе болған бирден бир тил.

Pascal программалаστырыў тили, программаның бөлимлери яки толық программа туўрысында ҳәр қыйлы пикирлерди дәлиллеўди тәмийнлейтуғын, жақсы математикалық аппаратқа тийкарланған. К.Хоардың аксиоматик системалары жәрдеминде аңлатылады. Программаларды ислеўин «сазлаў»дың ески усыллары орнына, ҳәзирги пайтта пайда болып атырған программалардың туўрылығын тексеретуғын автоматик системаларды пайдаланыў мақсетке муўапық.

Әлбетте, барлық өзгериўшилерди сүўретлеўди талап етиўи артықша болып көриниўи мүмкин, лекин программа жазыў аңсатлығына қарағанда, оның исенимли болыўы әҳмийетлирек.

1982 жыл А. Эддиман басшылығындағы топар тәрепинен Pascal программалаστырыў тилиниң Британия стандарты жаратылды, ол бүгинги күнде халық-аралық стандартқа да айланды.

Ҳәзирде Turbo Pascalдың бир қанша версиясы жаратылып, жоқары дәрежедеги программалар жаратыў имканиятлары барған сайын кеңейип бармақта:

4.0 версиясынан баслап программа жазыўды, редактор етиўди ҳәм нәтийжелер алыўды аңсатластырыў ушын жаңа интегралласқан орталық пайда етилди;

5.5 версиясының пайда болыўы менен Turbo-Pascalда объектли программалаστырыў имканияты пайда болды;

6.0 версиясынан баслап болса Pascal программасы ишине төмен программалаστырыў тили болған Ассемблер тилинде жазылған программаларды қосыў жағдайы пайда етилди. Усының менен бир қатарда тилдиң интегралласқан орталығы да бир қатар өзгериске дуўшар болды.

Операторалар

Оператор түсиниги тилдиң ең тийкарғы түсиниклеринен бири болып, ҳәр бир оператор тилде тамамланған гәп(фраза) болып есапланады ҳәм мағлыўматлар анализиниң тамамланған басқышын аңлатады.

Операторларды еки топарға ажыратыў мүмкин. 1-топар операторларының қурамына басқа операторлар қатнаспайды ҳәм бул операторлар тийкарғы операторлар деп аталады. Тийкарғы операторларға төмендеги операторлар киреди: меншиклеў операторы, процедура операторы, өтиў операторы, бос оператор. 2-топар операторларының қурамына басқа операторлар да қатнасып, олар қурама(составной) оператор деп аталады.Оларға төмендеги операторлар киреди: шәрт оператор, таңлаў операторы, тәкирарлаў операторы.

Мәселени шешиў алгоритмине жоқарыдағы еки топар операторлардың избе-излиги шекленбеген муғдарда қатнасыўы мүмкин. Бул избе-изликтеги оператор ";" ажыратқыш белгиси менен ажыратылады. Солай етип, С арқалы қәлеген жазыў мүмкин болған операторды белгилесек, мәселе шешилиўиниң алгоритми төмендеги избе-излик бойынша аңлатылыўы мүмкин: **`C; C; C; ... ;C`**

Операторлар орынланғанда жазылыў тәртиби бойынша орынланады. Бул орынланыў тәртиби тек өтиў операторы арқалы бузылыўы мүмкин.

Атамалар ҳәм идентификаторлар

Мағлыўматлардың анализ процесин аңлатыўшы алгоритм ҳәр түрли объектлер үстинде жумыс алып барады. Бул объектлердиң ўазыйпасы ҳәм қабыл мәнислерине қарап арнаўлы атамалар бериледи. Усы атамаларды әдетте, идентификатор деп атайды. Идентификатор деп ҳәрип яки "_" белгисинен басланыўшы ҳәрип, цифр ҳәм "_" белгисиниң қәлеген избе-излигине айтылады:

<идентификатор>::=<ҳәрип>/<идентификатор><ҳәрип>/<идентификатор><рақам>

Егер төмендеги аралық түсиниклерин киритсек:

<ҳәрип яки цифр>::=<ҳәрип>/<цифр>

Жоқарыдағы анықлаўды төмендегише жазыў да мүмкин:

<идентификатор>::=<ҳәрип>{< ҳәрип яки цифр >}.

Хызметши сөзлерден идентификатор сыпатында пайдаланыў мүмкин емес. Идентификатор орнына қысқа атама бериў қолайлы. Turbo Pascalда атамада қатнасыўшы белгилер саны 63 белгиден артпаўы керек.

Атқа мысаллар:

_Summa, C, C1, _ALI

Дағазалар

Pascal тилиниң тийкарғы түсиниклеринен бири дағаза етиў есапланады. Программада қатнасыўшы барлық объектлердиң атамалары сәйкес түрде программаның бас бөлиминде, олардың қандай типтеги мәнислер қабыл етиўи мүмкинлигине қарап, дағаза етилип қойылыўы керек. Pascal тилинде дағаза етиўдиң 5 түри бар:

- Меткалар дағазасы
- Турақлылар дағазасы
- Тип анықлаў ушын дағаза
- Өзгериўшилер дағазасы
- Процедура ҳәм функциялар дағазасы

Өзгериўшилер

Өзгериўшилер программа объекти болып, ҳәр түрли мәнислерди ядта белгили бир атама менен сақлап турыў ушын қолланылады. Өзгериўши өз мәнисин программаның орынланыў даўамында меншиклеў операторы жәрдеминде қабыл етеди. Қабыл етилген мәнис өзгериўшиге басқа жаңа мәнис берилгенше сақланып турады ҳәм жаңа мәнис берилиўи менен ески мәнис өшип, жоқ болып кетеди. Ҳәр бир өзгериўшиге белгили бир типке тийисли мәнислерди қабыл етиў ҳуқықы бериледи.

Өзгериўши-бул идентификатор. Оның атамасы өзгериўшиниң мәнисине байланысыўда қолланылады яғный, программа текстиндеги атама усы өзгериўшиниң мәнисин аңлатады.

Pascal программасының структурасы

Pascal программа загаловкасы ҳәм денесинен (блок) ибарат. Олардан кейин ноқат қойылады. Заголовка денеден ноқатлы үтир

менен ажыратылады:

<программа>::=<программа заголовкасы>; <программа денеси>.

Заголовкасы program хызметши сөзинен басланады, соң программаның атамасы (идентификатор, ҳәриплер ҳәм цифрлардан ибарат, ҳәриптен басланады) ҳәм қаўсырма ишинде сыртқы орталығы менен өз-ара байланыс қылыўшы файллардың атлары жазылады:

<программа заголовкасы> ::= program <программа аты> (<файл аты>{, < файл аты >});

Көп программалы операциялық системаларға әмел қылыўшы Pascal тиллеринде **<файл аты>** орнында стандарт **INPUT (оқыў)** ҳәм **output (жазыў)** файллары келеди ҳәм ол программа тәрепинен оқыў-жазыў қурылмаларына буйыртпаны билдиреди. Жеке лерде <файл аты> болыўы шәрт емес. Программаның денеси блок болып, ол 6 бөлимнен ибарат :

<блок>::=<метка бөлими><турақлылар бөлими><түрлер бөлими>

<өзгериўшилер бөлими><процедура ҳәм функциялар бөлими>

<операторлар бөлими>.

Меткалар бөлиминде программада пайдаланылатуғын меткалар дағаза етиледи. Метка - бул қандайда бир операторға өтиў ушын қолланылған, **0...9999** оралығындағы пүтин сан яки әпиўайы идентификатор. Меткалар өтиў операторы пайдаланылғанда қолланылады. Программада ислетилетуғын барлық меткалар **label** хызметши сөзинен кейин басланыўшы меткалар бөлиминде дағаза етиледи, яғный метка бөлиминиң синтаксиси төмендегише жазылады:

<метка бөлими> ::= <бос>

яки **Label <метка> {,<метка>};**

Мәселен,

Label 6,2,56,888;

Турақлылар - программаның ислеўи даўамында өзгермей қалатуғын шама. Егер шама программада көп мәрте пайдаланылса, оны программада қайта-қайта жазбастан, бул шаманы турақлы деп алып, программадағы шаманың орнына турақлының атын жазыў қолай болады. Мәселен, ҳәммеге белгили (π=3,14159265...) саны. Бул

санды программада бир неше рет жазыў қолайсыз, соның ушын оны турақлы сыпатында алыў мақсетке муўапық.

Турақлылар бөлими программада пайдаланылатуғын турақлылар дағаза етиў ҳәм оларға мәнис бериў ушын қолланылады ҳәм төмендегише жазылады:

<турақлылар бөлими> ::= <бос> яки const <турақлының дағаза етиў> {,<турақлының дағаза етиў>}

<турақлының дағаза етиў> ::= <турақлының аты> = <турақлы>

<турақлы> ::= <скаляр мәнис> яки **<белги қатар >** яки **<турақлының аты>** яки **+ <турақлының аты>** яки **- <турақлының аты>**

<белги қатар> ::= <белги>{<белги>}

Мәселен,

```
Const n=4; m= 12; nn=n; t=2;
```

Pascal тилинде мәнислердиң төрт стандарт түринен басқа түрлерди де киритиў мүмкин. Буның ушын, проблемаға киритилип атырған жаңа түрге сәйкес ат бериў жеткиликли ҳәм өзгериўшилерди дағаза етиў бөлиминде бул түрден пайдаланыў мүмкин.

Түрлер бөлими программада киритилген түрлерди дағаза етиў ушын қолланылады ҳәм төмендегише жазылады:

<түрлер бөлими> ::= <бос>

яки **type <тур дағаза етиў>{;<тур дағаза етиў>};**

< тур дағаза етиў> ::= <тур аты> = <тур>

<тур> ::= <тур аты> яки <турдиң берилиўи>

Мәселен,

```
type log=Boolean; week1=(monday, fryday, sunday);
```

Ҳәр қандай программада өзгериўшилер деп аталыўшы программа объектлеринен пайдаланылады. Өзгериўши – бул мәнис қабыл қылыўшы объект. Оларға мәнис программа орынланыўы даўамында бериледи.

Өзгериўшилер бөлими программада пайдаланылатуғын өзгериўшилер қайсы түрлерге тийисли болғанларын дағаза етиўде қолланылады ҳәм төмендегише жазылады:

<өзгериўшилер бөлими > ::= <бос>

яки **Var <өзгериўшилердиң дағаза етиў> {; <өзгериўшилердиң дағаза етиў>};**

<өзгериўшилердиң дағаза етиў> ::= <өзгериўшиниң аты > {,< өзгериўшиниң аты >}:<тур>;

Мәселен,

```
Var x,d: real; d1: week1; asd: log;
```

Процедура ҳәм функция бөлими программа дүзиўши программада өз мақсети ушын кириткен процедуралар ҳәм функцияларды дағаза етиў ушын қолланылады.

Процедура ҳәм функция бөлими өзгериўшилер бөлиминиң даўамы болып, **Procedure** яки **Function** хызметши сөзлери менен басланып, қәлеген избе-изликте дағаза етиледи.

Операторлар бөлими программаның тийкарғы бөлими болып, бос болмайды. Оның ишинде программада керек болған әмеллер избе-излиги жазылады:

<операторлар бөлими> ::= begin <оператор> {;<оператор>} end;

3-ЛЕКЦИЯ. БЕРИЛГЕНЛЕРДИҢ ТИЙКАРҒЫ ТҮРЛЕРИ

Жоба:

1. Пүтин түрлер
2. Ҳақыйқый түрлер
3. Символлы түрлер
4. Логикалық түрлер

Берилгенлер еки түрли болыўы мүмкин: турақлылар (олар программаның ишинде фиксерленген ҳәм турақлы болыўы керек); өзгериўшилер (олар программаның ишинде өзгериўи мүмкин). Берилгенлер ҳәр қыйлы түрдеги компоненталарға ийе структураларға бирлесиўи мүмкин. Структураны жалғыз мәнис яки қандайда объекттиң мәниси сыпатында көриўимиз мүмкин. Pascal тилинде төмендегише келисилген:

- қәлеген берилген мәнис қандайдур берилгенлер структурасына киреди ҳәм өз алдына берилген мәнис әпиўайы берилгенлер структурасы деп аталады (әпиўайы структура);
- берилгенлер структурасының элементи әпиўайы яки қурамалы берилгенлер структурасы (қурамалы структура) болыўы мүмкин;
- бир қыйлы структуралар ҳәр қыйлы түрлерге ийе болған берилгенлерден ибарат болыўы мүмкин.

Pascal тилиндеги түрлерден пайдаланып, программа дүзиўши берилгенлерди ядта сақланыўға әҳмийет бермеген ҳалда, олар менен әмеллер орынлайды.

Түрлер ҳаққында айтқада, бул түрлердеги өзгериўши ҳәм турақлыларды қандай мәнис қабыл қылыўы ҳәм олар үстинде қандай әмеллер ҳәм стандарт функциялар ислетилиўин көрсетиўимиз жетерли.

Pascal тилиндеги түрлер тийкарғы ҳәм қосымша түрлерге бөлинеди.

Тийкарғы түрлер әпиўайы структуралардан ибарат болған мәнислерден, қосымша түрлер қурамалы структуралардан ибарат.

Тийкарғы түрлерден көрсеткиш түри өз алдына ажыратылған,

қалған түрлер скаляр түрлер деп аталады. Скаляр түрлер стандарт ҳәм программист жаратқан түрлерге бөлинеди. Стандарт түрлерге пүтин, ҳақыйқый, логикалық ҳәм белгили түрлер киреди.

Пүтин түр (integer)

Пүтин түрге пүтин санлар көплигиндеги мәнислер (әмелге асырыўға байланыслы) киреди. Олар үстинде 5 тийкарғы әмеллер орынланады: қосыў +, айырыў - , көбейтиў * , пүтинге бөлиў **div** ҳәм пүтин бөлимин алыў **mod**. Бул әмеллер пүтин нәтийже береди. Биринши төрт әмеллер әпиўайы, бесиншиси төмендеги қағыйда бойынша орынланады: тек n>0 болғанда қолланылады,

егер **m>=0** болса **m mod n = m - ((m div n) + n);**

егер **m<0** болса **m mod n = m-((m div n) + n) + n.**

Аңлатпада шептен оңға қарап **div**, **mod**, ***** әмеллери биринши болып, кейин **-** ҳәм **+** әмеллери орынланады. Пүтин мәнислерин салыстырыў мүмкин ҳәм мәнис бериў операторында пайдаланыўимиз мүмкин.

Төмендеги стандарт функциялар пүтин нәтийже береди:

abs(x) – **x** тың абсолют мәниси,

sqr(x) – **x** тың квадраты,

trunc(x) – **x** тың пүтин бөлими,

round(x) – **x** ты жақын пүтинге шекем дөңгелеклеў.

Пүтин түр тәртипленген болғаны ушын, төмендеги стандарт функцияларды қолланыўымыз мүмкин:

succ(x) - х-тан кейинги пүтин мәнисин,

pred(x) - х-тан алдынғы мәнисин,

ord(x) - тәртип номерин береди.

Ҳақыйқый түр (real)

Ҳақыйқый түрге ҳақыйқый санлар көплигиндеги мәнислер (әмелге асырыўға байланыслы) киреди. **Real** түри тәртипленбегенлиги ушын **succ**, **pred**, **ord** функцияларын қоллаў мүмкин емес. ЭЕМ де ҳәр бир ҳақыйқый сан қандайдыр қәтелик пенен сақланады, соның ушын олар үстинде теңликке салыстырыў әмеллери

орынланғанда, дурыс нәтийже бермеўи мүмкин. Олар үстинде 4 тийкарғы әмеллер орынланады: қосыў **+** , айырыў **–**, көбейтиў *****, бөлиў **/**. Бөлиў әмеллин орынлағанда еки операнд пүтин болғанда да, нәтийже ҳақыйқый болады.

Төмендеги функцияларды аргументи **real** яки **integer** болғанда да нәтийжеси ҳақыйқый болады: **sin(x)**- синус, **cos(x)** -косинус, **arctan(x)**-арктангенс, **ln(x)**-натурал логарифм, **exp(x)**-экспонента, **sqrt(x)**- х-тың квадрат корени. **abs** ҳәм **sqr** функциялары болса аргументи ҳақыйқый болғанда ғана ҳақыйқый нәтийже береди.

Ҳақыйқый турақлылар онлық санақ системасында бериледи. Олар фиксерленген ноқат ҳәм жүзиўши ноқат көринисинде жазылады. Биринши көринисте санды пүтин ҳәм бөлшек бөлимлери ноқат менен ажыратылады. Екиншисинде сан төмендегише жазылады:

<жүзиўши ноқат менен жазылған сан >::=< белгиси жоқ пүтин сан >E< пүтин сан

яки **<фиксерленген ноқат менен жазылған сан>E<пүтин сан>**

<фиксерленген ноқат менен жазылған сан>::=<пүтин сан> яки **<пүтин сан>.<пүтин сан>**

Мәселен : **0.002; 3.14; 22.0; 0.2E-5; 6E3; 29.; 839E-09; 24E+03 .**

Символлы түр (char)

Символлы түрге белгили бир көпликтеги символлар киреди. Мәселен, Turbo-Pascalда бул көплик - **256** элементтен ибарат болған **ASCII** таблицасы болып, ҳәр бир компьютер ядында турақлы рәўиште бул таблицаға ийе болады. Ҳәр бир белги таблицадағы тәртип орнын анықлаўшы сан (коды) менен анықланады. Усы себепли белги ЭЕМ ядында оның коды көринисинде сақланады ҳәм ол бир байт (8 бит - екилик разряд) орын алады.

Символлы түринде болған мәнислер төмендеги қағыйдаларды қанаатландырады:

1) усы көпликке 0 ден 9 ға шекем цифрлар киреди, олар өсип барыў бойынша тәртипленген ҳәм олар көпликте избе-из жайласқан;

2) ислеп атырған версияда көпликте кишкене латын ҳәриплери киритилген болса, олар ҳәм олардың көпликте тәртип номерлери өсип барыў тәртибинде болыўы шәрт;
3) екинши қағыйда үлкен латын ҳәриплери ушын да орынлы;
4) егер a ҳәм b - символлар болса ҳәм a>b болса, онда ord(a)>ord(b), егер **a<b** онда **ord(a)<ord(b)**, егер **a=b** онда **ord(a)=ord(b)** болады.

Символлы турақлыларға төмендеги мысал болады: **'p', '_', '1'**

Char түри тәртипленген түр болғаны ушын, төмендеги стандарт функцияларды қоллаўымыз мүмкин: **succ(x)** - **x** символдан кейинги символ, **pred(x)** - **x** символдан алдынғы символ, **ord(x)** - **x** символды тәртип номерин береди. **Sqr(n)**- **n**- тәртип номерли символды береди. Символлы түрдеги мәнислер ушын салыстырыў ҳәм мәнис бериў әмеллери пайдаланылыўы мүмкин.

Логикалық түр (boolean)

Бул түрге тек еки логикалық турақлы киреди: **false**(жалған) ҳәм **true**(рас), соның ушын логикалық түрдиң мәнислер көплиги тәртипленген ҳәм **false**-ниң тәртип номери **0**, **true** - **1**.

Логикалық мәнислер үстинде бийкарлаў **not**, конъюнкция **and**, дизъюнкция **or**, салыстырыў, мәнис бериў әмеллерин орынлаў мүмкин ҳәм олардың орынланыў тәртиби жоқарыдағыдай.

Төмендеги функциялар қолланылады:

Odd(x) - рас мәнис береди егер **x**-тақ сан, кери жағдайда жалған;

eoln(x) - рас мәнис береди егер **x**-атлы текстли файлдың көрсеткиши қатардың ақырында болса, кери жағдайда жалған;

eof(x) - рас мәнис береди егер **x**-атлы избе-из файлдың көрсеткиши ақырында турған болса, кери жағдайда жалған.

4-ЛЕКЦИЯ. МӘНИС БЕРИЎ ОПЕРАТОРЫ. ОҚЫЎ-ЖАЗЫЎ ОПЕРАТОРЛАРЫ

Жоба:

1. Мәнис бериў операторы
2. Оқыў операторы
3. Жазыў операторы

Pascal тилинде ҳәр бир оператор берилгенлерди қайта ислеў процесиниң қандайда логикалық тамамланған, ғәрезсиз басқышы. Яғный мәнислер алыў ушын қағыйдалар аңлатпалар жәрдеминде жазылады. Мәнис бериў операторы жаңа мәнис есаплаў ушын аңлатпаны пайдаланады ҳәм нәтийжени усы оператордың шеп тәрепинде турған өзгериўшиде сақлайды. Мәнис бериў операторы Pascalдың тийкарғы операторларына киреди ҳәм төмендегише жазылады:

<мәнис бериў операторы> ::= <өзгериўши>:=<аңлатпа>

Солай етип, мәнис бериў операторының орынланыў нәтийжесиде қайсыдыр өзгериўши жаңа мәнис қабыл етеди, алдынғы мәниси болса жоғалып кетеди.

Аңлатпа ҳәм өзгериўши бир қыйлы түрге тийисли болыўы керек. Аңлатпа операндлардан дүзиледи. Операндлар төмендегише болыўы мүмкин:

турақлылар – олардың мәнислери программа орынланыўынан алдын белгили ҳәм программа орынланыўы процесинде өзгермейди;

өзгериўшилер – олардың мәнислери программа орынланыўында анықланады ҳәм өзгериўи мүмкин;

аңлатпалар – олардың мәнислери есаплаўдан алдын белгили болмайды.

Егер мәнис бериў операторының шеп тәрепинде жазылған өзгериўши ҳәм оң тәрепинде жазылған аңлатпаның түрлери **real** яки **integer** болса, бундай оператор арифметикалық мәнис бериў операторы делинеди. Егер шеп тәрепинде турған өзгериўши **real** болса, оң тәрепинде жазылған арифметикалық аңлатпа **real** яки **integer** болыўы мүмкин. Егерде шеп тәрепинде жазылған

өзгериўши **integer** болса, онда аңлатпа тек **integer** болыўы мүмкин. Арифметикалық аңлатпаның барлық операндларының түрлери real яки integer болыўы шәрт. Аңлатпаның тийкарғы операндлары сыпатында турақлы, өзгериўши яки функция, әмеллер сыпатында болса мултипликатив топар әмеллери **=**, **div**, **mod**, **+** ҳәм аддитив топар әмеллери **+**,**-** пайдаланылыўы мүмкин.

Div–пүтин бөлиўди аңлатады, тийиндиниң пүтин бөлими қалдырылып, қалдық таслап жибериледи. Мысал:

```
7 div 2 = 3
5 div 3 = 1
-7 div 2 = -3
-7 div -2 = 3
0 div 2 = 0
```

mod- пүтин санлар тийиндини қалдығын анықлайды. Мысал:

```
7 mod 2 = 1
-14 mod 3 = 1
```

Мултипликатив топар әмеллериниң приоритети аддитив топарының әмеллеринен жоқары. Ҳәр бир топарда әмеллердиң приоритети бир қыйлы ҳәм олар аңлатпада шептен оңға қарап орынланады. Аңлатпаны керек болған есаплаў тартибин қаўсырма жәрдеминде жазыў мүмкин.

Pascal-программа ЭЕМ ниң операциялық системасының кеңейтирилген орталығында орынланады. Операциялық система символикалық рәўиште **input** ҳәм **output** деп аталыўшы стандарт текстли файллар арқалы программаны ЭЕМ ниң оқыў-жазыў қурылмалары менен байланыстырады.

Тийкарынан, берилгенлерди киритиў ушын терминал клавиатурасы яки дисклерден (ийилиўшең, қатты, лазерли) ҳәм басқалардан пайдаланылады. Мағлыўматларды шығарыў ушын дисплей экранынан, принтерден ҳәм басқа қурылмалардан пайдаланылады.

Мәселен, операциялық система **input** стандарт текстли файлды терминал клавиатурасы менен байланыстырды. Улыўма жағдайда киритиў стандарт процедураларына байланысыў төмендегише жазылады:

```
read(x1,x2,... ,xn);
readln(x1,x2,... ,xn);
```

бул жерде **xi** – киритип атырған дизимдеги элемент. Киритилип атырған дизимдеги элемент сыпатында тек өзгериўши пайдаланылыў мүмкин. Киритилип атырған мәнислер сыпатында **char**, **real**, **integer** түрдеги мәнислер пайдаланыў мүмкин. Pascalда берилгенлерди киритиўде төмендегише келисилген:

a) киритилип атырған мәнистиң түри **read** процедурасының фактикалық параметр сыпатында қолланылған өзгериўшиниң түри менен сәйкес келиўи керек;

b) егер бир **read** киритиў процедурада параметрлер сыпатында бир неше өзгериўши жазылса, онда бир қанша қыйыншылықлар пайда болады. Бул қыйыншылықлар киритилип атырған мәнислер бир-биринен қандайдур ажыратыўшылар жәрдеминде ажыратылыўы керек болғаны менен байланыслы. Бул ажыратыўшылар сыпатында ислетилип атырған белгилердиң көриниси операциялық система ҳәм алгоритмлик тилдиң конкрет әмелге асырыўына байланыслы. Тийкарынан, санлы мәнислерди киритиўде, оларды бир биринен ажыратыў ушын "пробел" қолланылады. Read киритиў процедурасындағы өзгериўшилердиң түри бири санлы (**real** яки **integer**), басқасы символ болса, онда түсинбеўшилик туўылады. Бул жағдайда ЭЕМ "пробел"-ди ажыратыўши яки белги сыпатында түсиниўи мүмкин. Көрсетилген қыйыншылықты ҳәр қыйлы әмелге асырыўларда ҳәр қыйлы шешилген;

c) **input** стандарт файлдан тек **char**, **integer**, **real** түрдеги мәнислерди киритиў мүмкин. Басқа түрдеги мәнислерди (**Boolean** ҳ.т.б) программа дүзиў жолы менен киритиледи.

Output стандарт текстли файлына шығарыў **write**, **writeln** шығарыў процедуралар жәрдеминде орынланады.

write(x1,x2,... ,xn) көринисиндеги оператор төмендеги операторлар избе-излигине еквивалент

```
write(x1),write(x2),... ,write(xn);
```

бул жерде ҳәр бир операторда тек бир элемент шығарылып атыр. Соның ушын, биз тек **write(x)** көринисиндеги операторды көремиз.

Шығарылып атырған элемент төмендеги үш көринистен бири

болыўы мүмкин:

```
e
e:m
e:m:n
```

бул жерде e - аңлатпа (**char**,**real**,**integer** еки **boolean** түрдаги), қатарлы турақлы яки қатарлы өзгериўши, m ҳәм n болса оң пүтин мәнис қабыл ететуғын аңлатпалар.

Мағлыўматларды қатарлар бойынша жайластырыў ушын **writeln** (жаңа қатарға өтиў) яки **writeln (x1,x2,...,xn)** (шығарыў ҳән жаңа қатарға өтиў) процедуралардан пайдаланамыз.

5-ЛЕКЦИЯ. ШӘРТ ҲӘМ ВАРИАНТ ОПЕРАТОРЛАРЫ. ӨТИЎ ОПЕРАТОРЫ, ҚУРАМА ОПЕРАТОР

Жоба:

1. Шәрт операторы
2. Вариант операторы
3. Өтиў операторы
4. Қурама оператор

Тармақланыўшы есаплаў процесслеринде айырым басқышлары (операторлары) бәрқулла бир қыйлы тәртипте орынланбайды, олар қандайдур тексерилип атырған шәртлерге байланыслы болады. Бундай процесслерди таңлаў операторлар жәрдеминде жазыў мүмкин. Таңлаў операторлар қурама операторлар классына киреди ҳәм шәртли ҳәм вариант операторларынан ибарат.

Pascal тилинде шәртли оператордың еки түри бар: толық шәртли оператор ҳәм қысқартылған шәртли оператор:

<толық шәртли оператор> ::= if <логикалық аңлатпа> then<оператор> elsee <оператор>;

Бунда **if,then,else** - хызметши сөзлер.

Солай етип, толық шәртли оператор дүзилиси төмендегише – **if B then C1 else C2;**, бул жерде **B** - логикалық аңлатпа, **C1** ҳәм **C2** - операторлар. Бундай шәртли оператор төмендегише орынланады: егер **B** логикалық аңлатпа **true** мәнис қабыл етсе, онда **C1** оператор орынланады, кери жағдайда **C2** оператор орынланады. Шәртли операторға мысал:

```
if x<0 then i:=i+1 else k:=k+1;
```

Алгоритмлер жазылыўында базыда сондай жағдайлар ушырайды, қандайдур операторларды тек логикалық шәрт рас болғанда орынлаў керек, кери жағдайда бул операторларды орынлаў керек емес. Бундай жағдайларда шәртли оператор қолланылады.

<қысқа шәртли оператор>::= if <логикалық аңлатпа> then <оператор>

Солай етип, қысқартылған шәртли оператор - **if B then C**; көринисинде болып төмендегише ислейди: егер **B** логикалық аңлатпа

true мәнис қабыл етсе, онда **C** оператор орынланады, кери жағдайда кейинги операторға өтиледи. Шәртли операторда **then** ҳәм **else** сөзлерин кейин тек бир оператор турыўы керек, егер алгоритмде усы орынларда бир неше операторлар жазылыўы керек болса, онда олардан қурамалы оператор жаратылады. Қысқартылған шәртли оператор ислетилиўинде дыққатлы болыў керек. Мәселен, төмендеги операторды

```
If B1 then if B1 then C1 else C2
```

еки қыйлы мәнисте түсиндириў мүмкин:

```
if B1 then begin if B1 then C1 end else C2
```

яки

```
if B1 then begin if B1 then C1 else C2 end
```

Pascal тилиниң қағыйдасы бойынша екинши оператордың мәниси дурыс, яғный ҳәр қайсы хызметши сөз **else** өзине ең жақын болған хызметши сөз **true** сәйкес келеди.

Вариант операторының синтаксис анықламасы төмендегише:

```
<вариант таңлаў операторы>::=case<оператор селекторы>
Of <вариант дизимниң ағзалары >end;
```

Вариант таңлаў операторы орынланыў пайытында алдын селектордың мәниси есапланады, соң селектордың мәнисине сәйкес метка менен берилген оператор орынланады ҳәм усының менен бирге вариант таңлаў операторы өз жумысын жуўмақлайды.

Мысал:

```
Case i mod 3 of
 0: m:=0;
 1: m:=-1;
 2: m:=1;
```

Өтиў операторы, қурама оператор

Өтиў операторы Pascal тилиниң тийкарғы операторлары классына киреди ҳәм программа операторларының нормал избе-излигин өзгертиў ушын қолланылады. Ол өзи қайсы операторға өтиўин анықлайды. Программаның қәлеген операторына тек бир метка қойыў мүмкин. Ол оператордан алдын жазылады ҳәм оннан ":"

жәрдеминде ажыратылады.

<меткаланған оператор> ::= <метка> : < меткаланбаған оператор>;

Программада барлық қолланылған меткалар ҳәр қыйлы болыўы шәрт. Өтиў операторда көрсетилген метка қайсыдур операторда болыўы шәрт. Өтиў операторы төмендегише жазылады:

<өтиў операторы > ::= goto < метка >

Мәселен, **goto 3;**

Программада қолланылған меткалар меткалар бөлиминде бир мәрте дағаза етилиўи шәрт, қайсы тәртипте дағаза етилиўиниң әҳмийети жоқ. Өтиў операторы жәрдеминде қурама оператордың ишине, яки танлаў операторының бир тармағынан екинши тармағына өтиў мүмкин емес.

Қурама оператор қурамалы операторлар классына тийисли болып, ол программада операторлар избе-излигин бир оператор сыпатында пайдаланыў зәрүр болған жағдайда қолланылады. Қурама оператор begin ҳәм end гилт сөзлери ишине алынған операторлар избе-излигинен ибарат болады:

< қурамалы оператор> ::= begin <оператор> {;<оператор>} end;

Мәселен,

```
begin i: =0; p:=p+1 end;
```

6-ЛЕКЦИЯ. ТӘКИРАРЛАЎ ОПЕРАТОРЛАРЫ

Жоба:

1. Тәкирарлаў операторы
2. Параметри кемейип баратуғын тәкирарлаў операторы
3. Шәрти кейин берилген тәкирарлаў операторы
4. Шәрти алдын берилген тәкирарлаў операторы

Бул оператор басқаша параметрли тәкирарлаў оператор делинеди. Оны пайдаланыў ушын тәкирарлаў процесси төмендеги қағыйдаларға ийе болыўы керек: процесстиң такирарланыў саны оның орынланыўынан алдын белгили болыўы керек; тәкирарлаў процессин скаляр түрдеги өзгериўши басқарады. Бул өзгериўши берилген басланғыш мәнистен берилген ақырғы мәниске шекем избе-из мәнислерди қабыл етеди. Көбинесе төмендеги тәкирарлаў операторы қолланылады:

```
for B:=E1 to E2 do C;
```

бул жерде **for** (ушын), **to** (өсип барады) ҳәм **do** (орынлаў) - хызметши сөзлер, **B** - скаляр түрдеги өзгериўши (ҳақыйқыйдан тысқары) - тәкирарлаў параметр, **E1** ҳәм **E2** - **B** түрдеги аңлатпалар, **C**-оператор - тәкирарлаў денеси.

Бул оператордың ислеў принципин түсиндиремиз: **B** тәкирарлаў параметрине **E1** басланғыш мәнисинен баслап, **E2** ақырғы мәнисине шекем избе-из мәнислер бериледи ҳәм ҳәр бири ушын **C** операторы орынланады. **E1** ҳәм **E2** мәнислери тәкирарлаў процесси басланыўынан алдын тек бир мәрте есапланады, **C** операторы **B** мәнисин өзгертиўи мүмкин емес. Егер **E2<E1** болса, **C** операторы улыўма орынланбайды, егер **E1=E2** болса, тәкирарлаў бир мәрте орынланады.

Солай етип, арифметикалық тәкирарлаў операторын төмендеги операторлар избе-излиги жәрдеминде жазыў мүмкин (**bn** ҳәм **bk** - **B** түрдеги жардемши өзгериўшилер):

```
bn:=E1;
bk: =E2;
if bn<=bk then
begin
  B:=bn; C; B:=succ(B); C; B:=succ(B); C; . . . ; B:=bk; C
```

```
end;
```

Pascalда тәкирарлаў тамамланғанынан кейин **B**-ның мәниси анық емес есапланады. Арифметикалық тәкирарлаў операторын пайдаланып, берилген **n** ушын **y=1+2+3+...+n** төмендегише шешиў мүмкин:

```
y:=0; for i:=1 to n do y:=y+i;
```

Базыда тәкирарлаў параметрин кемейип барыў тәртибинде өзгериўи керек болады. Бундай жағдайда Pascalда тәкирарлаў операторының төмендеги формасы бар:

```
for B:=E1 downto E2 do C;
```

бул жерде downto (кемейип барыў) - хызметши сөз. Бул көринистеги тәкирарлаў операторы төмендеги операторлар избе-излигине эквивалент:

```
bn:=E1; bk:=E2;
if bn>=bk then
begin
  B:=bn; C; B:=pred(B); C; B:=pred(B); C;. . . ;B:=bk; C
end;
```

Жоқарыда келтирилген мәселени бул оператор жәрдеминде төмендегише шешиў мүмкин:

```
y:=0; for i:=n downto 1 do y:=y+i
```

Мысал: $S = \sum_{k=1}^{n} \frac{1}{k}$ қосындыны есаплаў ушын программа дүзиң. $n \in N$

```
Program summ;
Var
  S:real;
  i,n:integer;
Begin
  read(n);
  S:=0;
  For i:=1 to n do
    S:=S+1/i;
  Writeln(S);
End.
```

Параметрли тәкирарлаў операторы алдыннан тәкирарлаў саны

белгили болғанда ғана пайдаланылыўы мүмкин. Мәселен, итерация процесслеринде талапқа жуўап бериўши шешимге неше адымда ерисиў алдыннан белгили емес болғанлығы себепли, оның программасында параметрли тәкирарлаў операторынан пайдалана алмаймыз. Буның ушын шәрти кейин берилген тәкирарлаў операторын пайдаланыў мүмкин. Ол төмендегише жазылады:

repeat C; C; . . . C until B;

бул жерде - **repeat** (тәкирарлаў) ҳәм **until** (ге шекем) - хызметши сөзлер, **C** – қәлеген оператор, **B** - логикалық аңлатпа. Бул оператор орынланғанда **repeat** ҳәм **until** арасындағы операторлар избе-излиги кеминде бир мәрте орынланады. Бул процесс **B** логикалық аңлатпа **true** мәнис қабыл қылғанда тамамланады.

Шәрти кейин берилген тәкирарлаў операторды қолланып, Нъютон усылы жәрдеминде $y=\sqrt{x}$ ты жуўық есаплаў программасын жазыў мүмкин (eps – берилген дәллик):

```
y:=1;
repeat
  b:=(x/y-y)/2;
  y:=y+b
until abs(b)<eps;
```

Көринип турғанындай, шәрти кейин берилген тәкирарлаў операторы арифметикалық тәкирарлаў операторынан бир қанша улыўмарақ. Бирақ, шәрти кейин берилген тәкирарлаў операторыда тәкирарлаў денеси бир мәрте орынланып атыр. Базы жағдайларда шәртке байланыслы рәўиште тәкирарлаў денесиндеги операторлар бир мәрте де орынланбаўы мүмкин. Бундай жағдайда шәрти алдын берилген тәкирарлаў операторын пайдаланыў керек. Мәселен, M ҳақыйқый сан берилген болсын. 3^k>M шәртти орынлаўшы ең кишкене терис болмаған k пүтин санды табыў керек. Егер шәрти кейин берилген тәкирарлаў операторынан пайдалансақ,

```
y:=1; k:=0;
repeat
  y:=y*3;
  k:=k+1
until y>M;
```

M<1 мәнисине нәтийже дурыс емес.

Соған уқсас мәселелер ушын шәрти алдын берилген тәкирарлаў операторынан пайдаланамыз:

```
while B do C;
```

бул жерде **while** (ҳәзирше), **do** (орынлансын) - хызметши сөзлер, **B** - логикалық аңлатпа, **C** - оператор. **C** операторы ҳәр бир рет орынланыўынан алдын **B**-ның мәниси есапланады, егер ол рас болса, **C** операторы орынланады, кери жағдайда тәкирарлаў тамам болады. Егер **B** - ның мәниси биринши мәртеден жалған болса, тәкирарлаў улыўма орынланбайды.

Шәрти алдын берилген тәкирарлаў операторы ең улыўма деп есапланады, себеби бул оператор жәрдеминде параметрли ҳәм шәрти кейин берилген тәкирарлаў операторларын жазыў мүмкин.

Мысал: $S = \sum_{k=1}^{n} \frac{1}{k}$ қосындыны есаплаў ушын программа дүзиң. $n \in N$

(Шәрти алдын ҳәм кейин берилген тәкирарлаў операторын пайдаланың)

1)

```
Program summ;
Var
    S:real;
    i ,n:integer;
Begin
  read(n);
  S:=0; i:=1;
  While i<=n do
  Begin
    S:=S+1/i;
    i:=i+1;
  End;
  Writeln(S);
End.
```

2)

```
Program summ;
Var
    S:real;
    i,n:integer;
```

```
Begin
  read(n);
  S:=0; i:=1;
  Repeat
    S:=S+1/i;
    i:=i+1;
  Until i>n
Writeln(S);
End.
```

7-ЛЕКЦИЯ. МАССИВЛЕР. БИР ӨЛШЕМЛИ ҲӘМ КӨП ӨЛШЕМЛИ МАССИВЛЕР

(Регуляр түрлер)

Жоба:

1. Массив түсиниги. Массивлердиң берилиўи
2. Массивтиң түри
3. Көп өлшемли массивлер түсиниги

Массив түсиниги

Pascal тилиндеги мағлыўматлар түри әпиўайы ҳәм қурамалы (структурированные типы) болып бөлинеди. Әпиўайы түрге - стандарт, саналыўшы ҳәм шекленген түрлер жатады, қурамалы түрге - массивлер, көпликлер, жазыўлар, файллар киреди.

Қурамалы түрлердиң элементлери – әпиўайы түрлер өз гезегинде қурамалы түрлер де болыўы мүмкин. Программалаштырыў тилине қурамалы түрлердиң киритилиўи оны имканиятын күшейтеди ҳәм мәселелердиң кең классларын шешиўши нәтийжели программаларды дүзиўге мүмкиншилик береди. Қурамалы түрлерди кенирек қараймыз. Математикада, экономикада, информатикада көп жағдайларда мағлыўматлардың берилген көплиги қолланылады. Мәселен: санлар избе-излиги, таблицалар, фамилиялар дизими ҳәм т.б. Мағлыўматлардың бир түрдеги көплигин қайта ислеў ушын массив түсиниги киритиледи. Массив деп бир түрдеги шекли сандағы мағлыўматлар көплиги түсиниледи. Массив бир ат пенен белгиленеди. Ҳақыйқый санлардың 1.6, 14.9, -5.0, 8.5, 0.46 көплигин массив деп атаў мүмкин ҳәм бир атама менен, мәселен: A деп белгилеў мүмкин массивтиң ҳәр бир элементи массивтиң индексли аты менен көрсетиледи. Математикада индекс орын қаўсырмаға алынады яки массив атамасының индексинде көрсетиледи.

Мәселен: **A1,A2,A3,A4** яки **A(1),A(2),A(3),A(4)** яки улыўма түрде **{Ai}** бунда **i=1,...,n**.

Pascal тилинде индекс квадрат қаўсырмаға алынады. Қаралып атырған мысал ушын, **A** массивиниң элементлери:

```
A[1]:=1.6; A[2]:=14.9; A[3]:=-5.0; A[4]:=8.5;
A[5]:=0.46;
```

Егер программада массив қолланылса, онда ол өзгериўшилер бөлиминде (**Var**) ямаса түрлер бөлиминде (**Type**) берилиўи керек.

Дәслеп **VAR** өзгериўшилер бөлиминде массивти бериўди қараймыз.

Массивтиң берилиўи төмендеги түрге ийе болады:

VAR <массив аты > : ARRAY [<T1>] of <T2>;

Бунда **ARRAY** (массив) **of** (дан) хызметши сөзлер; **T1**- **REAL** ҳәм **INTEGER** стандарт түрлеринен басқа қәлеген әпиўайы түрдеги индекс түри. **T2**-PASCAL тилинде рухсат етилген, массивтиң элементлериниң түри. Жоқарыдағы мысал ушын массивти бериў төмендеги түрге ийе болады:

VAR A : ARRAY[1..5] of REAL;

Бунда **A**-элементлери **REAL** таяныш түрдеги массивтиң аты; индекс түри **1** ден **5** ке шекем шекленген. Индекс түри стандарт яки ҳақыйқый түр болмайтуғын болғанлықтан төмендегидей етип массивти бериў мүмкин емес:

VAR A : ARRAY[5] of REAL;

Ямаса **VAR A : ARRAY[INTEGER] of REAL;**

Массивлерди дурыс бериў мысаллары:

```
VAR
  massiv: array[1..n]  of real;
  jil: array[yan..dek] of integer;
  l: array[1..n]  of boolean;
  M1: array[qatar]  of dongelek;
```

Егер бир қанша массив бир түрли индекслер түрге ҳәм бир түрли таяныш түрге ийе болса, онда бериўде массивлерди дизимнен бирлестириўге рухсат етиледи.

Мәселен: **VAR A,B,C : ARRAY[1..50] of REAL;**

Бул жерде ҳәр қайсысы **50** элементке (**1** ден **50** ге шекем) ийе болған ҳақыйқый санлардың үш **A**,**B**,**C** массивлер дизими дағазаланған:

A[1],A[2],...,A[50].

B[1],B[2],...,B[50].

C[1],C[2],...,C[50].

"**Индекс**" түсиниги менен "**индекстиң түри**" түсинигин алмастырмаў керек. Индекстиң түри тек массивти бериў бөлиминде қолланылады, ал индекс болса, массивтиң анық элементлерин белгилеў ушын операторлар бөлиминде көрсетиледи. Бул жағдайда индекстиң түри индекстиң түриниң берилиўинде сәйкес келиўи шәрт. Индекс болып аңлатпа, дара жағдайда турақлы яки өзгериўши болыўы мүмкин. Массивтиң элементи басқаша индексли өзгериўши деп аталады. Буннан ажыратыў ушын индекссиз өзгериўшини әпиўайы өзгериўши деп аталады. Массивтиң элементи мәнис бериў операторының шеп, оң тәрепиндеги аңлатпаларда-да турыўы мүмкин. Массивтиң элементлери үстинде, оның таяныш түрдеги мағлыўматлары үстинде ислеў мүмкин. Егер таяныш түр **INTEGER** болса, онда пүтин түрдеги мағлыўматлар үстинде ислеў мүмкин болған әмеллерди ҳәм стандарт функцияларды орынлаў мүмкин. Массивтиң элементлерин операторлар бөлиминде қолланыў мысаллары:

```
B[5] :=B[3]+1;
SUM :=SUM-C[K];
P1 :=A[2*I+1];
```

Массивтиң сан мәнислерин киритиў ҳәм шығарыў ушын тәкирарлаўлар қолланылады.

Мәселен: `for I:=1 to 9 do READ(A[I]);`

Тәкирарланыўшы **A** массивиниң элементлериниң **9** мәнисин киритиўди аңсатластырады яғный: **A[1],A[2],...,A[9],** ал

`FOR I:=1 to 9 do WRITE(A[I]);` тәкирарлаў сол массивтиң элементлериниң **9** мәнисин шығарады. Мәселен: **15** пүтин санның қосындысын есаплаң. Санларды белгилеў ушын элементлери **I** индекске ийе болған **X** массивиниң атын киритемиз, қосындысын **SUM** аты менен белгилеймиз. Онда **SUM = X[1]+X[2]+...+X[15]**.

Массивти бериў ҳәм оның элементлерин операторлар бөлиминде қолланыўшы программаның бөлеги:

```
VAR  x : array[1..15] of integer;
- - - - - - - - -
sum:=0;
for i:=1 to 15 до
begin
```

```
  read(x[i]);
  sum:= sum+x[i];
end;
```

Тәкирарлаўдың ишинде **X[I]** массивиниң элементиниң бир мәниси киритиледи ҳәм ол **SUM** қосындысының сол ўақыттағы мәнисине қосылады. Тәкирарлаў **15** мәрте қайталанады. Массивтиң элементлери пүтин түрге (**INTEGER**), ал индекс - **1..15** шекли түрге ийе. **I** өзгериўшиси бир тәрептен тәкирарлаў параметри болып, ал екинши тәрептен индекс болып қолланады. Бул жағдайда **I** өзгериўшисин еки түрли етип бериў мүмкин: а) **VAR** өзгериўшилер бөлиминде **I** өзгериўшисин шекли түрли етип көрсетиў мүмкин, мәселен:

```
VAR
     X: ARRAY[1..15] of INTEGER;
     I: 1..15;
```

б) Шекли түрдеги элементлер пүтин санлар болғанлықтан олардың түрин **INTEGER** деп киритиў мүмкин, мәселен:

```
VAR
     X: ARRAY[1..15] of INTEGER;
     I: INTEGER;
```

Pascal тилинде барлық өзгериўшилерди бериў керек болғанлықтан, бул мысалда **SUM** өзгериўшисин де бериў керек. Олда **INTEGER** түрин алады, себеби барлық қосылыўшылар санлар (**X[I]**) пүтин түрге ийе

Массивтиң түри

Pascal тилинде массивлерди өзгериўшилер бөлиминде анық бериўден басқа еки бөлимнен ибарат берилиў көриниси бар. Дәслеп түрлерди **TYPE** дағаза етиў бөлиминде массивтиң түри көрсетиледи. Кейин өзгериўшилерди **VAR** дағаза етиў бөлиминде көрсетилген түрге тийисли массивлер бериледи. Массивтиң түриниң киритилиўи дағаза етиў бөлимин көбейтеди, бирақ ол өз гезегинде программаны дүзетиўди аңсатластырады. Массивлерди бериў төмендеги түрге ийе болады:

```
TYPE <түрдиң аты > = ARRAY[<T1>] of <T2>;
VAR <массивтиң аты > : <түрдиң аты> ;
```

Бунда **T1** – индекстиң түри; **T2** - массив элементлериниң таяныш түри. Мейли, мәселен, программада ҳақыйқый түрдеги **10** элементтен ибарат **P** массиви қолланылатуғын болсын. Массивтиң түрин **MAC** атамасы менен белгилеймиз. Онда массивти бериўде төмендегидей етип орынлаў мүмкин:

```
TYPE MAC=ARRAY[1..10] of REAL;
VAR P : MAC;
```

Егер программада бир қанша, мәселен **P**,**A**,**B**,**C** массивлери **MAC** түринде болса, онда тек өзгериўшилерди бериў бөлими өзгереди:

```
VAR P,A,B,C : MAC;
```

P,**A**,**B**,**C** массивлери программаның операторлар бөлиминде қолланылады. **MAC** массивтиң түри тек дағаза етиў бөлимине формал түрде киритилген ҳәм ол программаның басқа жерлеринде көрсетилмейди ҳәм қайта исленбейди.

Жыйнақланған массивлер

Бир сан ямаса бир белги ЭЕМ де 1 сөз яки 2 байтта жайласады. Соның менен бир ўақытта, символларды сүўретлеп көрсетиў ушын ядтың бир байты жеткиликли. Pascal тилинде символлы мағлыўматларды қолланыўда, ЭЕМның ядын үнемлеў мақсетинде жыйнақланған массив түсиниги киритилген. Жыйнақланған массив элементлериниң екеўи **1** сөзде яки **2** байтта сақланады. Символлы мағлыўматлардың жыйнақланған массиви өзгериўшилер **VAR** бөлиминде төмендеги берилиўге ийе:

```
VAR массивтиң аты: PACKED ARRAY[<индекс түри>] of CHAR;
```

Мәселен, символлар қатары берилсин: **'SAPAEV KAMAL'**. Бул символлар қатарын бос орынды есаплағанда **12** символлы массив деп есаплаў мүмкин. Массивтиң аты **FAM** деп есаплаў мүмкин. Массивти бериў төмендеги түрге ийе болады:

```
VAR FAM : PACKED ARRAY[1..12] of CHAR;
```

Массивтиң бир элементи символдың бир мәнисин қабыл етеди, мәселен: **FAM[1]:='S'**, **FAM[2]:='A',...,** **FAM[12]:='L'**. Жыйнақланған массивлерди **TYPE** бөлимин қолланып бериў усыныс етиледи. Көрип өтилген мысал ушын дағаза етиў төмендегидей түрге ийе болады:

```
TYPE T=PACKED ARRAY[1..12] of CHAR;
VAR FAM : T;
```

Бунда дәслеп **T** атамасы массивтиң түри киритилген, кейин өзгериўшилер бөлиминде **FAM** массиви **T** түринде екенлиги көрсетилген. Демек, **FAM** символлардың жыйнақланған массиви болып есапланады. Жыйнақланған массивтиң элементлери программада жыйнақланбаған массивтиң элементлериндей етип қолланылады. Тек жыйнақланған массив ушын яд автомат түрде кем бөлинеди. Символлы турақлылар жыйнақланған массивлер түрин қабыл етеди деп есапланады:

```
PACKED ARRAY[1..N] of CHAR;
```

Бунда **N**-қатардың узынлығы. Символлы турақлылар тап солай тәртипке ийе массивлерге берилиўи мүмкин ҳәм олар менен салыстырыў мүмкин. Мәселен, массивтиң берилиўи төмендегидей түрге ийе болса:

```
TYPE C=PACKED ARRAY[1..19] of CHAR;
VAR katap : C;
```

Онда төмендегидей оператор дурыс болады:

```
katap: = 'РАШИД САПАЕВ - oquvchi';
```

Жыйнақланған массивлерди дүзиў ушын тәкирарлаўлар қолланылады. Мәселен, символлы мағлыўматлардың **B** массивин дүзиўди төмендегидей етип орынлаў мүмкин:

```
readln;
i:=1;
while not eoln do
begin
  read(b[i]);
  i:=i+1;
end;
```

B[I] массивтиң элементине киритилген символдың мәниси беріледи. **WHILE** тәкирарлаўы киргизилетуғын қатарда қатардың оқыры **[Enter]** символы ушырамағанша орынланады.

Мәселен: **PASCAL - АЛГОРИТМЛИК ТИЛ** [Enter] программаның бул бөлегиниң орынланыўы нәтийжесинде массивтиң элементлери төмендеги мәнислерди қабыл етеді: **B[1]='П', B[2]='А',**

B[3]='C',...,B[23]='Л'. Әлбетте, B массиви киргизилетуғын қатардың узынлығынан киши емес өлшемде берилиўи керек, мәселен:

```
VAR B : PACKED ARRAY[1..N] of CHAR;
```

Бунда **N>23**. Тәкирарлаў **WHILE** операторында киритилетуғын қатардың анық узынлығын көрсетиў мүмкин, мәселен **WHILE I<=23 do** ямаса **WHILE I<=K do**

Көп өлшемли массивлер түсиниги

Ҳәзирге шекем биз ҳәр бир элементи тек бир индексли массивлерди қараған едик. Бундай массивлерди, бир өлшемли массивлер деп аталады . Математикада көп ўақытларда көп өлшемли яғный массивлердиң массивин қолланылады. Еки өлшемли массивлер яғный матрицалар деп аталатуғын массивлер кең тарқалған. Мәселен, бир қанша қатардағы избе-из жайласқан пүтин санлар матрицалар болып есапланады:

```
5 4 3 6
2 8 1 7
4 3 9 5
```

Берилген матрица **3** ке **4** өлшемге ийе яғный ол **3** қатардан ҳәм **4** бағанадан ибарат. Егер матрицаны бир атама менен мәселен **A** менен белгилесек онда матрицаның ҳәр бир элементи еки индекс пенен белгиланеди, мәселен: **A[I,J]**. Бунда биринши индекс **I (I=1,2,3)** қатардың номерин белгилейди, екинши **J (J=1,2,3,4)** бағананың номерин белгилейди. Бундай матрицаны төмендеги түрде бериў мүмкин:

```
1) TYPE T=ARRAY[1..3,1..4] of INTEGER;
VAR A : T;
2) TYPE T=ARRAY[1..3] of ARRAY[1..4] of INTEGER;
VAR A : T;
```

Биринши жағдайда, индекстиң ҳәр бир түри бериледи, кейин массив элементлериниң әпиўайы таяныш түри **INTEGER** көрсетиледи. Екинши жағдайда, дәслеп **[1..3]** индексиниң мағлыўматлар түри бериледи, кейин қурамалы таяныш түр **ARRAY[1..4] of INTEGER;** көрсетиледи, ол өз гезегинде басқа индекстиң түрин дағаза етеди ҳәм әпиўайы таяныш түр **INTEGER** ди дағаза етеди. Егер программада

матрицаның айырым қатарларын ажыратып көрсетиў керек болса, онда төмендеги түрде дағаза етиў ықшам болып есапланады:

```
TYPE T1=ARRAY[1..4] of INTEGER;
T=ARRAY[1..3] of T1;
VAR  A:T;
     B:T1;
```

Бунда дәслеп матрицаның бир қатарының түри **T1** бериледи, оннан **T1** қатардың түри жәрдеминде матрицаның **T** түри бериледи, өзгериўшилер бөлиминде **A** яки өлшемли массив екенлиги яғный матрица екенлиги, ал **B** бир өлшемли массив екенлиги көрсетиледи.

Мысал. Ҳақыйқый санлардың қосындысын есаплаң. Программаны орынлаўдағы нәтийжени көрсетиў ушын **7** анық сан берилсин: **5.1, 6.4, 8.7, 1.9, 3.6, 2.0, 4.2**

Программа

```
program summa;
const n=7; (*санлардиң сани*)
var
  a: real;(*киритилетуғин сан*)
  sum:real;(*қосинди*)
  i:integer;(*тәкирарлаў параметри*)
begin
  sum:=0;
  writeln('санларди бос орин менен киритиң:');
  for i:=1 to n do
  begin
    read(a);
    sum:=sum+a
  end;
  writeln(sum:6:2)
end.
```

```
Санларды бос орын менен киритиң:
5.1 6.1 8.7 1.9 3.6 2.0 4.2
31.90
```

Бунда ҳәр-бир сан тәкирарлаўдың ишинде киритиледи ҳәм қосылатуғын қосынды **SUM** ға қосылады.

```
    Программа
program summa2;
const n=7; (*санлардиң сани*)
var
  a:array[1..n] of real;(*санлар массиви*)
  sum:real;(*қосынды*)
  i:integer;
begin
  (*санлар массивин киритиў*)
  writeln('санлар массивин пробел менен киритиң:');
  for i:=1 to н do read(a[i]);
    sum:=0;
  for i:=1 to n до
  sum:=sum+a[i];
  writeln(sum:6:2)
end.

санлар массивин пробел менен киритиң:
5.1 6.4 8.7 1.9 3.6 2.0 4.2
31.90
```

Бунда өзгериўшилер **VAR** бөлиминде **A** массиви анық дағаза қылынған. Массив ушын ядтың **7** ячейкасы ажыратылады яғный ҳәр бир сан ушын бир ячейка. Операторлар бөлиминде дәслеп тәкирарлаў жәрдеминде **A** массивиниң элементлериниң барлық мәнислери киргизиледи. Қосынды есаплаў ушын тәкирарлаў ықшамластырылады. Массивти қолланыў ЭЕМ ниң ядын пайдаланыўды жақсылайды.

8-ЛЕКЦИЯ. TURBO PASCALДА ҚУРАМАЛЫ ТҮРЛЕР

Жоба:

1. Саналыўшы түрлер
2. Шегараланған түрлер
3. Қатарлар
4. Көпликлер ҳәм олардың берилиўи
5. Көплик түри үстинде әмеллер.

Саналыўшы түрлер

Усы ўақытқа шекем биз мәнислердиң стандарт скаляр түрлери (варинт операторынан басқасы) үстинде айтып өттик ҳәм олардан программада пайдаландық. Бул түрлер Pascal тилиниң өзинде анықланған түрлер еди. Бирақ Pascal тили программа дүзиўши өзи ушын қолай болған жаңа түрлер киритиў имканиятын да береди. Бундай түрлер сыпатында шекленген ҳәм саналыўшы түрлерди көрсетиў мүмкин.

Pascal тилинде тек бир стандарт санап өтилиўши **boolean** түри бар. Оннан тысқары программа дүзиўши, оған зәрүр санап өтилиўши түрлерди дағаза етиўи мүмкин.

Қәлеген стандарт болмаған берилгенлер түри программада түр дағаза етиў жәрдеминде анықланған болыўы керек. Санап өтилиўши түрдиң берилиўи төмендегише жазылады:

```
<санап өтилиўши тур> ::= ( <at > {,<at>});
```

Қаўсырма ишиндеги үтир менен ажыратып жазылған <at> лар санап өтилиўши турақлылар. Олар санап өтилиўши түрдиң мәнислер көплигин пайда етеди. Санап өтилиўши түрдиң мәнислер көплиги тәртипленген ҳәм тәртип номери 0 ден басланады.

Санап өтилиўши түрге мысал:

```
Reng = (qizil, qara, aq, sari);
```

Hapte =(duyshembi, shiyshembi, sarshembi, piyshembi, juma, shembi, ekshembi);

```
Miyweler =(alma, juzim, shabdal);
```

Бул түрдиң мәнислер көплиги қаўсырмадағы төрт элементтен

ибарат. Бул элементлер жазылған түрдиң турақлылары болады ҳәм **qizil < qara < aq < sari** шәртин қанаатландырып, төмендеги тәртип номерлерге ийе болады: **qizil - 0, qara - 1, aq - 2, sari- 3**.

Программада дағаза етилип атырған түрлердиң көпшилигин (санап өтилиўши де шегараланғаны да) еки усыл жәрдеминде анықлаў мүмкин:

1) жаңа түрдиң анықланыўы түрлер бөлиминде бул түрди дағаза етиў ҳәм оған ат бериў жолы менен анықланады:

<тур дағаза етиў> ::= <тур аты >=<түрдиң берилиўи>

яки **<тур аты >=<тур аты>**

Мәселен, **type name =(mike,nike,jane,olga,ada);**

Түрлерди бериўде бир қыйлы атлар пайдаланыў мүмкин емес. Соң бул түрге тийисли өзгериўшилер **Var** бөлиминде бериледи: **Var child: name;**

2) бул түрдиң берилиўи **Var** бөлимиде өзгериўшниң берилиў жеринде **<тур>** компонентаның орнына жазылады.

Мәселен, **Var child:(mike,nike,jane,olga,ada);**

Санап өтилиўши түрдиң мәнислери үстиде тек салыстырыў ҳәм мәнис бериў әмеллерин ҳәм **succ(x)**, **pred(x)**, **ord(x)** стандарт функцияларын пайдаланыў мүмкин. Түрдиң берилиўинде биринши болып жазылған ат ушын **pred** ҳәм ақыры жазылған ат ушын **succ** функциялардың мәнислери анықланбаған. Санап өтилиўши түрдиң мәнислерин оқыў-жазыў ушын әпиўайы оқыў-жазыў процедураларын пайдаланып болмайды, вариант операторынан пайдаланыў керек.

Шегараланған түрлер

Ҳәр бир шегараланған түр берилген тәртипленген түр, соның ишинде, санап өтилиўши түр тийкарында анықланады. Бул берилген тәртипленген түр шегараланған түрге салыстырғанда тийкарғы делинеди ҳәм аралық жәрдеминде анықланады. Аралық өзиниң шегаралары, тийкарғы түрдеги мәнислери болып, төмендегише бериледи:

<шегараланған тур> ::= <турақлы1> .. <турақлы2>

бул жерде **<турақлы1>** ҳәм **<турақлы2>** - тийкарғы түрге тийисли

ҳәм биринщиси екинщисинен үлкен болмаўы керек. Санап өтилиўши түрге уқсас, шегараланған түр де жоқарыда айтылған еки жол менен берилиўи мүмкин:

Мәселен:

```
type boy=(mike..nike);
Var d: jan..ada;
Var c: boy;
```

Қатарлар

Символлы түрли өзгериўшилер **char** хызметши сөзи менен берилип, бул түртиң мәнислери ядтан 1 байт орын ийелейди. Pascal тилиниң барлық символлары бул түрдиң мәнислер областына тийисли. Символлы мәнисти ' (апостроф) белгиси ишине алып, яки **#** белгисинен кейин оның **ASCII** кодын жазып анықлаў мүмкин. Мысал: **'A'**, ямаса **#60**

Қатар – бул ' (апостроф) белгиси ишине алып жазылған символлардың әпиўайы избе-излиги: **' Аб21#9!сд', 'Программист А Якупов'**.

Қатар бос ямаса бир символлы болыўы да мүмкин. Қатарлы өзгериўши узынлығы 255 ке шекем болған символлы мәнислерди қабыл етиўи мүмкин. Улыўма алғанда, ҳәр бир қатарлы өзгериўшиге ядтан 256 байт орын ажыратылады. Ядты үнемлеў ушын қатардың түрин төмендегише көрсетиў мақсетке муўапық: **String[N]**, **N** – қатардағы символлар саны. Бул жағдайда катарлы өзгериўши ушын **N** байт орын ажыратылады.

Символлар ҳәм қатарлар үстинде бир қанша әмеллер орынлаў мүмкин, яғный қатардан керекли бөлекти кесип алыў, қатарларды бир бирине қосыў ҳәм нәтийжеде жаңа қатарлар пайда етиў. Қатарлар ҳаққында толық мағлыўматты керекли бөлимнен алыў мүмкин.

Символлар ҳәм қатарларға байланыслы төмендеги әпиўайы программаны келтиремиз:

```
Program String;
Var
  ch:char; {ch o’zgeriwshi simbolli ma’nisti qabil etedi }
```

```
  q1,q2: String; {q1 ha'm q2 o'zgeriwshiler 255 ten artpag'an  qatarlardi
   menshiklewi mu'mkin}
  N: String[5]; {N o'zgeriwshisi  5 simvoldan ibarat qatarlardi menshikleydi }
Begin
  ch:='A';{ ch o'zgeriwshisi  A simvoldi menshikleydi}
  N:='Asqar'; {N o'zgeriwshisi  5 ha'ripli Asqar so'zin menshikleydi }
  Q1:=ch+'li  ' +N; {q1 o'zgeriwshisi  na'tiuyjesi Ali Asqar so'zin menshikleydi}
  q2:=' '; {q2 o'zgeriwshisi  bos qatar, biraq yadtan 256 bayt orindi iyeleydi}
End.
```

Көпликлер ҳәм олардың берилиўи

Көплик түсиниги математика курсынан жақсы белгили болып, программаластырыў тиллеринде де қолланылады.

Pascal тилинде тек элементлери бир қыйлы түрге тийисли шекли көпликлер ғана қаралады. Көплик элементиниң түри бул көпликли түрдиң тийкарын қураўшы түр делинеди.

Pascalда көпликли түрдиң мәниси көплик болады. Көпликли түрдиң анық мәниси (өзгериўши ямаса турақлы) квадрат қаўсырмаға алынған көплик элементлериниң дизиминен ибарат болады:

<көпликли түри> ::= <көпликтиң дүзилиси>

<көпликтиң дүзилиси> ::= [ ] ямаса
[<элемент>{,<элемент>}]

<элемент> ::= < аңлатпа> яки <аңлатпа>..<аңлатпа>

Көплик элементлери тийкарғы түрдеги турақлы яки усы түрге тийисли аңлатпадан ибарат болыўы мүмкин. Көплик улыўма айтқанда бос болыўы мүмкин онда көплик **[ ]** көринисте жазылады.

Pascal тилинде жазылған көпликке мысаллар:

```
[ ], [ 2,4,5,3 ],
[ 'c','ғ', 'ə' ],
[ n,m,4 ],
[1,2. . 5],
[n..m],
[ 'ə'.. 'ф','т','х'. . 'з']
```

Көпликте элементлердиң жайласыў тәртибиниң әҳмийети жоқ ҳәм ҳәр бир элемент бир мәрте есапқа алынады.

Көпликли түрдиң берилиўи төмендеги көринисте болады:

<көпликли түр > ::= set of < тийкарғы түр>

<тийкарғы түр>::=< тийкарғы түрдиң берилиўи> ямаса < тийкарғы түрдиң аты>

Мәселен, **set of 1..3** көринистеги көпликли түр мәниси **1..3** диапазонындағы барлық пүтин санларды өз ишине алады. Сонлықтан, бул көпликли түдиң мәниси бос көплик ҳәм **1,2,3** санларынан ибарат көпликлер болады: **[ ], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3].** Усы көпликлер ғана жоқарыдағы көпликли түрдиң мәниси болады.

Көпликли түр үстинде әмеллер

Pascal программаластырыў тилинде көпликли түдиң мәнислери үстинде төмендеги әмеллер анықланған:

Мәнис бериў, яғный көпликли түрдеги өзгериўшиге көпликли аңлатпаны бериў, яғный аңлатпаның мәниси көплик болады (бунда оң ҳәм шеп тәрептеги мәнис бериў операторының түрлери бир бирине сәйкес болыўы керек);

бирлеспе +

кесиспе ***

айырма -

A=B қатнасы **A** ҳәм **B** көпликтин үстпе –үст түсиўи, **A<>B** - **A** ҳәм **B** көпликтиң үстпе –үст түспеўи, **A≤B** - **A** көпликтиң **B** көпликте жатыўы, **A≥B** - **A** көплик **B** көпликти өз ишине алады, **x in A** - **x** - элемент **A** көпликке тийисли.

Соны айтып өтиў керек, жоқарыда санап өтилген биринши үш әмелдиң нәтийжеси көплик болады ҳәм бул әмеллер еки орынлы есапланады. Олардың операндлары көпликли турақлылар, өзгериўшилер ҳәм көпликли аңлатпалар.

Қатнас әмели болса логокалық түрдеги мәнис береди ҳәм ол еки орынлы есапланады.

x in A әмелинде биринши операнд көпликтиң тийкарғы түриниң элементи, екинши операнд болса көпликли аңлатпа, көпликли турақлы яки көпликли өзгериўши болыўы мүмкин.

9-ЛЕКЦИЯ. ЖАЗЫЎЛАР

Жоба:

1. Аралас түрлер (жазыўлар) ҳаққында түсиник
2. Аралас түрлерди дағаза етиў
3. Иерархик жазыўлар
4. Байланыс операторы

Әмелиятта экономикалық ҳәм информацияны қайта ислеў мәселелерин шешиўде белгили түрдеги хүжжетлерди, каталоглар, дизимлер, ведомостлар қолланылады. Мәселен, студентлердиң анкета мағлыўматлары: фамилиясы, аты, атасының аты, туратуғын жери, туўылған жылы, қәнигелиги, топар номери ҳәм т.б. Бул жағдайларда түрли класстағы мағлыўматларды бир топарға бирлестириў зәрүрлиги туўылады. Көринип турғанындай, фамилия, ат– қатар, туўылған жылы, топар номери–сан. Pascal тилинде бундай мағлыўматларды аралас тип жәрдеминде аңлатыў мүмкин.

Аралас түр қурамалы түрлер классына тийисли болып, оның мәниси регуляр түрдиң мәнисине уқсас болады улыўма жағдайда тривиал болмаған берилгенлер структурасына уқсас. Аралас түрдиң мәниси бир неше компоненталардан ибарат болып, таблица шамадан парықлы бул компоненталар ҳәр қыйлы түрге тийисли болыўы мүмкин ҳәм бул компоненталарға байланысыў таблица шамадағыдай индекс бойынша емес, бәлким компонентаның аты бойынша әмелге асырылады.

Аралас түрдиң мәнисин әдетте **жазыў** деп аталады.

Аралас түрдиң мәниси қурамалы қурылмаға ийе болған қандайда бир объектти сүўретлеў ушын қолланылады, әдетте ҳәр қыйлы көринистеги информациялық системаларды жаратыўда қолланылады.

Аралас түрдиң мәниси шекли сандағы компонентадан ибарат берилгенлер структурасынан ибарат болып, бул компоненталар майдан деп аталады. Жазыўдың ҳәр бир майданына ат қойылып ҳәм бул майданның түри көрсетиледи. Соны еслетип өтиў керек, жазыўдағы майданның түрине ҳеш қандай шегара қойылмайды, соның ушын жазыўдың майданы өз гезегинде және жазыў болыўы

мүмкин ҳәм т. б. Бунда ҳәр бир майдандың тәсир етиў шегарасы өзи анықланған жазыўдың ишки бөлими болады. Бир жазыўдағы ҳәр бир майданның атлары ҳәр қыйлы болиўи керек. Егерде бир ат басқа бир аттың тәсир етиў шегарасында жатқан болса яки бул атлар бир жазыўдың ҳәр қыйлы майданларында жайласқан болса, онда бул атлар бир қыйлы деп қабыл етиледи. Ҳәр қыйлы жазыўлар болса бир қыйлы атлардан ибарат майданлардан ибарат болыўы мүмкин, себеби бул бир қыйлы атларға байланысқанда әлбетте сыртқы жазыўдың аты қолланылады.

Аралас түрлерди дағаза етиў

Аралас түрдиң берилиўи төмендеги көринисте болады:

<аралас түрдиң берилиўи> ::= record<майданлардың дизими >end

<майданлардың дизими >::=<жазыў секциясы>{;<жазыў секциясы>}

<жазыў секциясы>::=<майдан аты >{,<майдан аты >}:<түр>

Усы қағыйдадан пайдаланып берилген комплекс саннан ибарат аралас түрди төмендеги көринисте анықлаў мүмкин:

```
type
  complex=record
    re: real;
    im: real;
  end;
```

Бул нәрсе соны билдиреди, **complex** түрдеги ҳәр қандай мәнис жазыўдан ибарат берилгенлер структурасынан ибарат болып, ол еки компонентадан (майдан) **re** ҳәм **im** ибарат ҳәм бул компоненталарды ҳәр бири **real** түриндеги мәнис қабыл қылады.

Endи өзгериўшилер бөлиминде **complex** түриндеги өзгериўшини киритиў мүмкин, мәселен :

```
Var x, y: complex;
```

Бул дағаза етилген өзгериўшилер **x** ҳәм **y** өзгериўшиниң мәнислерин анықламайды, балким бул өзгериўшилердиң ҳәр бириниң мәниси **complex** деп аталған структурадан ибарат екенлигин билдиреди. Бул өзгериўшилердиң ҳәр бирине анық мәнис бериў ушын олардың ҳәр бирин компоненталарына мәнис бериў керек.

Өзгериўшилердиң компонентаҳларына байланысыў төмендеги көринистеги структура қолланылады:

< толық өзгериўшиниң аты >. <майдан аты >

Мысал ушын: **x.re := 5; x.im := 10; y.re := x.re +13;**

10-ЛЕКЦИЯ. ҮЛЕС ПРОГРАММАЛАР

Жоба:

1. Процедуралар. Процедуралардың берилиўи
2. Параметрсиз процедуралар
3. Параметрли процедуралар
4. Функциялар. Функциялардың берилиўи
5. Процедураның процедура-функциялардан парқы

Процедуралар. Процедуралардың берилиўи

Программаластырыўда сондай жағдайлар болады, онда программаның түрли орынларда мазмуны тәрептен бир қыйлы алгоритмди орынлаўда туўры келеди, ол ғәрезсиз мәниске ийе, яғный шешилип атырған тийкарғы мәселениң қандайда бир бөлим мәселесин шешиўге арналған, мәселен еки натурал санның ең үлкен улыўма бөлиўшисин табыў, вектордың компоненталарын өсиў яки кемиў көринисте тәртиплеў ҳәм басқа мәселелер.

Егер бул алгоритм көлеми үлкен ҳәм қурамалы болса, онда бул алгоритмди ҳәр сапар қайта жазыў программаны қурамаластырыў менен бир қатарда оны жазыў процессинде грамматикалық қәтелерге жол қойыў итималлығын асырады. Жоқарыдағы кемшиликлердиң алдын алыў мақсетинде, ҳәмде оның ықшамлығын ҳәм түсиникли көринисин тәмийнлеў ушын Pascal тили ҳәр қандай алгоритмди тийкарғы программаның текстинен ажыратып, оны ғәрезсиз программа объекти көринисте, тек бир мәрте жазыў имканын береди. Бул программа объекти процедура (үлес-программа) делинеди.

Процедура- процедура ҳәм функциялар бөлиминде процедураны дағаза етиў жәрдеминде иске түсириледи:

<процедураның берилиўи > ::= <процедура басламасы> : <блок>;

Блок программа процедураға байланыслы болып, ол программа дүзилисиндеги бөлимлер менен бир қыйлы синтаксиске ийе.

Процедураның басламасы **procedure** хызметши сөзинен, процедураның аты ҳәм үлес ишинде базы бир қосымша мағлыўматларды өз ишине алған болыўы мүмкин.

Берилген процедураны активлестириў, яғный иске түсириў ушын программаның зәрүр жеринде процедура операторды жазыў керек:

<процедура оператор> ::= <процедура аты > яки <процедура аты > (<фактик параметрлер дизими>);

Процедура параметри сыпатында (егер ол бар болса) сондай мәнислер қолланылады, оларсыз процедура ислей алмасын, яғный олар бул процедура ушын басланғыш берилгенлери анықланады.

Параметрсиз процедуралар

Процедураның еки қыйлы көринисин пайдаланыў мүмкин: параметрли процедура ҳәм параметрсиз. Параметрсиз процедураларда берилгенлер тийкарғы программаның блогынан алынады, яғный глобал мәнис есапланады. Бул болса рационал есапланбайды. Мысал сыпатында төмендеги программаны келтириў мүмкин:

```
program max1(input,output);
var
  x,y,u,v:    real;
  a,b,c:      real;
begin
  read(x,y);
  a:=x+y; b:=x*y;
  if a>b then c:=a else c:=b;
  u:=c;
  a:=0.5; b:=u;
  if a>b then c:=a else c:=b;
  v:=c;
  writeln(u,v)
end.

program max2(input,output);
var
    x,y,u,v: real;
    a,b,c:   real;
procedure max;
begin
  if a>b then c:=a else c:=b;
```

```
end;
begin
  read(x,y);
  a:=x+y; b:=x*y; max; u:=c;
  a:=0.5; b:=u; max; v:=c;
  writeln(u,v)
end.
```

Параметрли процедуралар

Процедуралар менен ислеўде программалаштырыўдың қолай усылы бул - параметрли процедуралар есапланады. Параметрлер еки қыйлы болыўы мүмкин: параметр мәнисли ҳәм параметр өзгериўшили.

Параметр мәнислер ҳаққындағы мәселени көрейик. Жоқарыдағы жазылған программада процедураның берилгенлери (**a** ҳәм **b**) алдыннан фиксерленбеген, бул шегаралаўдан ўаз кешейик. Буның ушын процедурада басқа мақсетлерге пайдаланбайтуғын мәнислерди ажыратып рәсмий рәўиште оларды **p1** ҳәм **p2** деп белгилеймиз. Усы белгилеўден пайдаланып процедураны төмендеги көринисте жазамыз:

```
program maxq (input,output);
var x,y,u,v: real; c:real;
procedure maxx(p1,p2:real);
begin
  if p1>p2 then c:=p1 else c:=p2
end;
Begin
  read(x,y);
  maxx(x+y,x*y); u:=c;
  maxx(0.5,u); v:=c;
  writeln(u,v)
End.
```

p1 ҳәм **p2** идентификаторлар процедураның **рәсмий (формал) параметрлери** делинеди ҳәм олар процедураның анық мәнислери емес, бәлким улыўма мәнислер сыпатында қаралады. Процедураға ҳәр бир байланысыўда, оның рәсмий параметрлери анықластырылады, соның ушын кейинги байланысыўлар аңсат болыўы ушын рәсмий

параметрлер процедураның басламасында анық көрсетиледи. Программаның исенимлилигин асырыў ушын рәсмий параметрлер менен бирге олардың түри де көрсетиледи.

Бундай процедураларға байланысыўда етиўде оған сәйкес процедура операторында оның атынан кейин қаўсырма ишинде фактик параметрлердиң дизимин көрсетиў керек. Процедураға байланысыў болғанда рәсмий параметрлердиң бул фактик параметрлер мәнислери менен алмастырылады. Рәсмий ҳәм фактик параметрлер түри ҳәм санап өтилиў тәртиби бойынша бир-бирине сәйкес келиўи керек.

Жоқарыда көрип шығылған процедураның рәсмий параметрлери оның параметр мәнислери делинеди, себеби бундай параметрлер процедураға байланысыўдағы оған сәйкес фактик параметрлердиң мәнислерин анлатады. Бундай жағдайда фактик параметрлер рәсмий параметрлердиң түрине сәйкес келген қәлеген аңлатпа болыўы мүмкин, атап айтқанда оның түрине сәйкес келиўши өзгериўши яки турақлы болыўы мүмкин.

Фактик параметрлер процедураның рәсмий параметрлерине сәйкес келыўши оның ишки өзгериўшилерине мәнислерди узатыў ушынғана хизмет етеди ҳәм басқа мақсетлерде пайдаланып болмайды.

Жоқарыда көрилген maxx процедура есапланған мәнисти барлық ўақытта с өзгериўшиге береди. Бул программаны пайдаланыўды қыйынластырады, себеби процедураны пайдаланыўден мақсет қандайда бир алгоритмди бир топар берилгенлер ушын есаплап, нәтийжени болса басқа топар өзгериўшиге узатады. Бул кемшиликти жоғалтыў ушын нәтийже узатылып атырған өзгериўшини де процедураның параметри сыпатында киритиў керек. Бирақ бул рәсмий параметрдиң қәсийети жоқарыда көрилген процедура параметр мәнисинен парық қылады: ол тек процедураға узатылған мәнис сыпатында емес, балким процедурадан сыртта бар болған өзгериўшини аңлатады. Бул өзгериўшиге процедура ишинде мәнис берилиўи, яғный процедураның ишинде бундай өзгериўшиге байланысыў мүмкин. Процедураның бундай рәсмий параметрине параметр -өзгериўши делинеди.

Параметр-өзгериўшини параметр-мәнисте парықлаў ушын процедураның басламасында рәсмий параметрлер дизиминде

параметр-өзгериўшиниң алдына Var хызметши сөз жазылады. Рәсмий параметр-өзгериўшиден кейин оның түри көрсетиледи. Параметр-өзгериўшиге сәйкес келиўши фактик параметр тек өзгериўши болыўы мүмкин.

Усы жағдайда келтирилген программаны төмендеги көринисте жазыў мүмкин:

```
program max4(input,output);
var x,y,u,v: real;
procedure maxxx(p1,p2:real; var pec:real);
begin
  if p1>p2 then pec:=p1 else pec:= p2;
end;
begin
  read(x,y);
  maxxx(x+y,x*y,u);
  maxxx(0.5,u,v);
  writeln(u,v)
end.
```

Егерде процедура параметрлери қурамалы түр болса, мәселен, таблица шамалар, бул түрлерди тийкарғы программаның түрлер дағазалаў бөлиминде бериў керек. Процедура дағазалаўда болса дағазаланған түрден пайдаланыў керек. Өлшеми үлкен болған өзгериўшилерди параметр-өзгериўши көринисте дағаза етиў машина ядын ҳәм машина ўақтын үнемлеў нәзеринен абзаллық береди, әсиресе таблица шамалар үлкен өлшемли болған жағдайда.

Функциялар. Функциялардың берилиўи

Функция дегенде функцияның мәниси деп аталыўшы мәнисти функцияның аргументи деп аталыўшы басқа бир мәнис пенен байланыслылығы түсиниледи.

Алгоритмлик тиллерде болса сондай функциялар қаралады, олар ушын функцияның мәнисин анықлаўшы алгоритм бериў мүмкин болыўы керек.

Pascal тилинде мәнислери тек әпиўайы түрге тийисли болған функциялар қаралады. Программаластырыўда ҳәр қандай функцияны да формула көринисте бериў мүмкин емес, Базы ўақытлары рекурсив көринистеги избе-излик арқалы функцияның мәниси есапланыўы

мүмкин. Егер бул избе-излик программаның бир неше жеринде келетуғын болса ҳәр сапар оны жазбастан тек бир мәрте функционал байланысты көрсетип оған ат берип, керек болса аргументлерин көрсетиў жеткиликли (процедурада көрилгениндей).

Функционал байланысты анықлаў ушын мөлшерленген процедураға **процедура-функция** делинеди.

Pascal тилинде функцияны анықлаў ушын <функцияны дағаза етиў> түсиниги хызмет етеди ҳәм булар функция ҳәм процедуралар бөлиминде жайластырылады.

Функцияны дағаза етиў синтаксиси процедураның синтаксисине уқсап анықланады:

<функцияны дағаза етиў> ::= <функцияның басламасы>;<блок>

<функцияның басламасы> ::= function <функция аты>; <түрдиң аты> (<рәсмий параметр аты>{,<рәсмий параметр аты>})

яки **function <функция аты>: <түрдиң аты>**

Бул қағыйдадан, процедура функцияның басламасы процедураның басламасына уқсап, бирақ басламаның оқырына функцияның мәнисине сәйкес келиўши түрди көрсетиў кереклиги көриип тур.

Соны айтып өтиў керек, процедура-функцияны параметрсиз, ҳәмде параметр-мәнисли ҳәм параметр-өзгериўшили көринисте пайдаланыў мүмкин. Процедура-функцияны шөлкемлестириўши блок әдеттегидей анықланады, бирақ функцияны есаплаў процессинде функцияның мәниси менен бирге аралық мәнислер де есапланады. Соның ушын функция мәнисин бул аралық мәнислерден парқлаў ушын функцияны есаплаўшы операторлар избе-излигинде кеминде бир төмендеги көринистеги мәнис бериў операторы қатнасыўы керек:

<функция аты>::=<аңлатпа>;

бул жазыў функцияның мәниси сыпатында оң тәрептеги аңлатпаның мәниси алыныўын билдиреди. Бул көринистеги операторлар тек функцияны дағаза етиўде ғана ислетиледи.

Процедураның процедура-функциялардан парқы

Солай етип процедура-функцияның процедурадан үш парқы бар:

1) процедура-функция **function** хызметши сөзинен басланады;

2) процедура - функцияның басламасында функция мәнисиниң түри көрсетиледи;

3) функцияны есаплаўшы программада кеминде бир шеп тәрепинде дағаза етилген функцияның аты болған мәнис бериў оператор көрсетилиўи керек ҳәм бул операторлардың кеминде биреўи орынланыўы керек.

Мәселен **n** факториалды есаплаўшы программаны Pascal тилинде төмендеги көринисте жазыў мүмкин:

```
function fact(n:integer):integer;
Var i,k: integer;
begin
  k:=1;
  for i:=1 to n do k:=k*i;
  fact:=k;
end;
```

Pascal тилиниң стандарт процедура ҳәм процедура-функциялар класы бар, оларды программада тиккелей пайдаланыў мүмкин.

Процедураны иске түсириў процедура-операторы жәрдеминде әмелге асырылар еди, процедура функцияны болса функцияны шақырыў менен әмелге асырылады. Функцияны шақырыўшы аңлатпа өз алдына оператор сыпатында келиўи келиўи мүмкин емес, ол мәнис бериў операторының оң тәрепинде еки басқа үлес-программаға байланысыўда параметр-мәнис сыпатында келиўи мүмкин.

```
y:= fact(7);... writeln('факториал=', fact(10));
```

11-ЛЕКЦИЯ. ФАЙЛЛАР

Жоба:

1. Pascal алгоритлик тилинде файл түсиниги
2. Файллар ушын арналған улыўма процедура ҳәм функциялар
3. Түрленген ҳәм түрленбеген файллар.
4. Түрленген файллар ҳәм олар ушын арналған улыўма процедура ҳәм функциялар.
5. Түрленбеген файллар ҳәм олар ушын арналған улыўма процедура ҳәм функциялар.

Pascal алгоритмлик тилинде файл түсиниги

Файл өзи не? **MS DOS** операциялық системасында бул тусиник киритилген еди ҳәм **файл** деп ядтың өз атына ийе областына айтылады. Файлда түрли мағлыўматлар сақланады.

Ҳәр бир файл менен файл көрсеткиши деген түсиник бириктирилген. Файл бир неше элементлерден ибарат болып, тек файлдың көрсетип атырған мағлыўматына пайдаланыўшы қабыл кылыўы мүмкин. Демек, физикалық тәрептен биз тек избе-из файлларға ийемиз. Яғный биз алдын биринши, кейин екинши, үшинши ҳәм т. б. мағлыўматларды оқыўымыз мумкин. Файл өз атына ийе. Мәселен: **d:\tp6\myfile.dat**.

Pascal тили программалық қураллары жәрдеминде, яғный программада файллар пайда кылыў ҳәм ондағы мағлыўматларды қайта ислеў мумкин.

Усы кунге шекем, биз Pascal программаластырыў тилинде бирнеше өзгериўшилердиң түрлери менен ислеп келдик. Булар скаляр, әпиуайы ҳәм қурамалы дузилген түрлер болып табылады. Бул түрдеги мағлыўматлар жәрдеминде мәселелерди шешиўде басланғыш мағлыўматлар клавиатурадан оператив ядға киритиледи ҳәм нәтиже экранға шығарылады. Олардан басқа программаларда пайдаланып болмайды, себеби олар системадан шығарылғаннан кейин ҳеш жерде сақланбайды. Бул мағлыўматларды ядда сақлаў ушын Pascal тилинде мағлыўматлардың файллы түрлери белгиленген.

Файл түрлери айрықша орын ийелейди. Файл түри менен ислесиўде белгили түсиниклерди излестириў талап етиледи.

Бириншиден, файллар түри неге ҳәм қашан қолланылады? Мақсет не? Мутәжлик неден келип шығып атыр? Екиншиси, басқа түрлерден неге үлкен парқы бар?

Бул сораўларға тек пайдаланыўшының көз-қарасынан келип шыққан жағдайда жуўап бере аламыз:

1. оғада көп өзгериўшилерден пайдаланғанда ҳәр дайым клавиатурадан киритиў белгили дәрежеде колайсызлықларға дус келемиз. Үлкен массивлерди еслең;
2. сондай мәселелер ушырасады, алдыннан шамалардың мәнислер саны белгисиз болады (мәселен нәтижелер), бул шамаларды файлға жазыў мақсетке муўапық;
3. ҳар қандай түрлер сыртқы қурылмаларға байланысып, олар менен ислесиўге имкан жаратады (программалық тил областында).

Ҳәм соңында басқа түрлерден файл түриниң парқы сонда, ол басқа түрлер структурасына кире алмайды. Мәселен,

```
Var
  ст: record
    n: integer;
    fio: string
  end;
  adr:array[1..15] of char:
  pr:1930..1975;
```

бул түрлер ишинде файл түрин ислетиў мүмкин емес.

Файллар ушын арналған улыўма процедура ҳәм функциялар

Файлда сақланып атырған мағлыўматлар түрине қарап Pascal алгоритмлик тилинде файллар төмендеги түрлерге бөлинеди:

1. **түрленген;**
2. **түрленбеген;**
3. **текстли.**

Түрленген файллар бир түрли элементлерден қуралған. Оларды тек белгили қурылмалардан жеткерип бериў мүмкин бирақ экранда

оқыў мүмкин емес. Файлдың элементлери машина кодларыда жазылады ҳәм сақланады.

Түрленбеген файлларда түрли түрдеги мағлыўматларды сақлаў мүмкин. Олар ҳәм машина кодлары менен жазылған болып байтлар топламынан қуралған.

Текстли (текст) файллар **ASCII** кодлардан қуралған ҳәм қатарларға ажрытылған болады. Текстли файлларда тек файлдың соңында файл ақыры белгиси, ҳәтте ҳәр қатардың соңында арнаўлы қатар соңы белгиси қойылады.

Файл түриндеги өзгериўши файл өзгериўшиси деп аталады, ол файлдың логикалық атын белгилейди ҳәм ол логикалық файлды сыртқы файл (физикалық) ортасында "**жеткерип бериўши**" ўазыйпасын атқарады.

Файл түри ушын арифметикалық әмеллер белгиленбеген. Ҳәтте файлларды салыстырыў ҳәм бир файлдың мәнисин екинши файлға меншиклеў әмеллери де анықланбаған.

Хәр бир түрдеги файллар үстинде улыўма алғанда төмендеги әмеллерди орынлаў мумкин ҳәм бул әмеллер ушын арнаўлы процедура ҳәм функциялар қолланылады.

1. Turbo Pascal файл менен ислесиўден алдын файлдың физикалық ҳәм логикалық атларын байланыстырыў лазым.

Бул өз алдына процедура жәрдеминде әмелге асырылады.

```
Assign (<файл өзгериўшиси>, '<name: string>');
```

Бул жерде name файлдың физикалық аты болып, бунда файлдың сыртқы (турақлы) ядда сақланған жолы көрсетиледи, мәселен,

```
Assign (F, 'd:\TP\myfile.dat');
```

Бул процедураның мәниси сонда, ол файл ушын жол ашып программадан сыртқы қурылмаға мүрәжәт қылыўы ҳәм информация алмастырыў имканын жаратып береди.

2. Файлға мағлыўмат жазыў ушын файлды ашыў. Буның ушын төмендеги процедура қолланылады.

```
Rewrite(<файл өзгериўшиси>);
```

Бул процедура орынланғанда ядда Assign процедурасында көрсетилген ат пенен таза файл оған мағлыўмат жазыў ушын

ашылады ҳәм файл көрсеткиши файлдың басына орнатылады. Бирақ бул процедураны абайлап пайдаланыў керек, себеби көрсетилген файл алдыннан ядда болса ондағы мағлыўматлар пүткиллей өширип тасланады..

3. Файлды оннан мағлыўматларды тез ядқа оқыў ушын ашыў:

Reset (<файл өзгериўшиси>); **Reset** процедурасы орынланғанда **Assign** процедурасында көрсетилген файл оннан мағлыўматларын оқыў ушын таярланады, яғный файл көрсеткиши файлдың биринши элементине келтирип қойылады.

4. Файл мағлыўматларын жазыў, киритиў. Буның ушын бизге таныс болған **Write (<файл өзгериўшиси>,<шама>);**

Бул жерде шама орнында өзгериўши ямаса көринис пайдаланылыўы мумкин. Процедура орынланғанда шаманың мәниси файл өзгериўшиси менен байланған файлда файл көрсеткиши орнатылған жерге жазылады. Соң файл көрсеткиши бир мәрте кейинги позицияға жылысады. **Write** процедурасын **Rewrite** процедурасы орынланғаннан кейин ғана пайдаланыў мумкин.

5. Файлдан мағлыўматларды оқыў. Бул мақсетте төмендеги процедурадан пайдаланылады:

Read (<файл өзгериўшиси>,<өзгериўши>);

Бул процедура орынланғанда **Reset** процедураси менен ашылған файлда файл көрсеткиши орнатылған жерде элементтиң мәниси процедурадағы өзгериўшиге меншиклейди. Кейин файл көрсеткиши тағы бир позицияға жылысады.

6. Түрли мақсетте ашылған барлық логикалық файллар әлбетте жабылыўы керек. Буның ушын төмендеги процедура арналған:

Close(<файл өзгериўшиси>);

Бул процедура орынланғанда информация жеткерип бериўдиң барлық каналлары жабылады.

7. Файл ақырын анықлаў функциясы:

EOF (<файл өзгериўшиси>);

Бул функцияның мәниси **Boolean** түринде болып, ол файл көрсеткиши файлдың ахырына орнатылғанда **True** мәнисине ийе болады, кери жағдайда оның мәниси **False** қа тең.

Файллардың қәлеген көриниси менен ислесиўде соны нәзерге алыў керек, бир ўақыттың өзинде бир файлдан оған мағлыўмат жазыў ушын ҳәм оннан мағлыўматларды оқыў ушын оннан пайдаланып болмайды. Оқыў ушын яки жазыў ушын ашылған файл әлбетте **Close** процедурасы жәрдеминде жабылған болыўы шәрт.

Түрленген ҳәм түрленбеген файллар

Түрленген файллар санлар, белгилер ҳәм олардан дүзилген структураның машиналық көринисинен ибарат. Олар берилгенлерди ЭЕМ ядында қандай сақласа тап сондай жағдайда сақланады. Соның ушын түрленген файллар жәрдеминде программаның жумысшы яды ҳәм дисктеги мағлыўматларды айырыў мүмкин, лекин туўрыдан - туўры берилгенлерди экранға шығарыў мүмкин емес.

Түрленбеген файлларда берилгенлердиң машиналық көринисинен ибарат. Бул файлдың түрленген файлдан парқы соннан ибарат түрленген файл алдыннан дағаза қылынған түрдеги берилгени менен ислейди, түрленбеген файллар болса берилгенлердиң дүзилисине ҳәм әҳмийетине байланыслы болмаған байтлар жыйнағы менен ислейди.

Қәлеген түрдеги файл түрлерде мағлыўматларды сақлаўдың ең киши бирлиги байт есапланады. Файллар менен ислеў принципи ҳәмме файллар ушын бир қыйлы болып ең дәслеп файл өзгериўшилерди физикалық файллар менен байланыстырыў керек.

Программада дағаза етилген файл өзгериўшилер мәнис бериў операторларында қатнаспаўы керек.

Процедура функция темасында рәсмий параметр сыпатында пайдаланылып атырған қәлеген түрдеги файл өзгериўшилер Var параметр сыпатында дағаза етилиўи керек.

Түрленген файл синтаксиси төмендегише анықланады:

<түрленген файлдың берилиўи> ::= file of < тур>

Түрленбеген файл синтаксиси болса төмендегише анықланады:

<түрленбеген файлдың берилиўи> ::= file

Turbo Pascal тилинде қәлеген түрдеги файллар ушын стандарт процедуралар анықланған болып, олардан ең көп пайдаланатуғынлары төмендегилерден ибарат:

assign(f,c) - **f** файллы өзгериўшини физикалық файл менен байланыстырады:

Reset(f) - логикалық файл **f** ти оқыў ушын ашады:

Rewrite(f) - логикалық файл **f** ти жазыў ушын ашады:

close(f) - логикалық файл **f** ушын оқыў-жазыў каналын жабады.

Түрленген файл ашылғаннан соң берилгенлерди киритиў - шығарыў төмендеги стандарт операторлар жәрдеминде әмелге асырылады:

read(f,<атлар дизими>), write(f,< атлар дизими >).

Биринши аргумент - логикалық файлдың аты, оннан кейин болса файл компонентасының түрине сәйкес келиўши өзгериўши (өзгериўшилер) болып, оған оқыў (**read**) пайытында файлдың гезектеги мәниси жазылады, яки кериси жазыў (**write**) пайтында өзгериўши мәниси файлға жазылады.

Түрленбеген файллар ушын төмендеги процедураларды пайдаланыў мүмкин, яғный **Reset(f,buf)** яки **Rewrite(f,buf)**, бунда **buf** - файлға бир мәрте байланысыўда оқыў яки оған жазыўдағы байтлар саны (блок өлшеми). Блоктың ең киши өлшеми **1** байт болыўы мүмкин.

Түрленбеген файллар да берилгенлерди оқыў яки оған жазыў ушын төмендеги процедураларды пайдаланыў керек:

blockread(f,buf,s,r) ва **blockwrite(f,buf,s,w)**

бул жерде **f** -файл аты, **r** – мағлыўматлар оқылатуғын өзгериўши, **w** - мағлыўматлар жазылатуғын өзгериўши, **s** - оқылыўы керек болған блоклар саны, **4** ҳәм **2** өзгериўшилер болса қатнасыўы шәрт болмаған өзгериўшилер болып, олар оқыў-жазыў процедуралары жәрдеминде оқылған (жазылған) блоклар санын билдиреди.

Түрленген файллар

Түрленген файллар бир түрдеги элементлерден қуралған. Олар программада төмендегише бериледи:

<файл түри>::=file of <элементлер түри>

<элементлер>::=<тур>

бул жерде элементлер түри файлды қурайтуғынлар яғный файлдағы мағлыўматлардың түри болып, бул түр сыпатында әпиуайы ҳәм қурамалы түрлерди (файлдан тысқары) пайдаланыў мумкин.

Түрленген файлларды **Type** ҳәм **Var** бөлимлеринде тәрийиплеў мумкин. Мәселен,

```
type fint=file of integer;
     tal=file of char;
     num=file of real;
Var p,q: file of integer; f:file of char;
 s:file of real;
 p,q:fint;
 ff:tal; s:num;
```

Элементлер түри орнында қурамалы түрлерди ҳәм пайдаланыў мумкин. Мәселен, жазыўларды,

```
Type Student=Record
     Fio: string[12];
     Gr:1975..1982;
     adress:string[15]
End;
Var St:File of Student;
```

Бул түрдеги файллар үстинде жоқарыда келтирилген улыўма процедура ҳәм функциялар қатарында тағы қосымша процедура ҳәм функцияларды пайдаланыў мумкин. Базы бир процедуралар файлды туўрыдан-туўры мүрәжет файлы сыпатында пайдаланыў имканын береди.

1. **FileSize(<файл өзгериўшиси>);** -функциясы файлдағы элементлер санын анықлайды, функцияның түри **Integer** (яки **LongInt**) болыўы керек.
2. **FilePos(<файл өзгериўшиси >):Integer;** -функцияси актив элементтиң файлдағы орнын анықлап береди, актив элемент деп файл көрсеткиши орнатылған элементге айтылады.

3. **Seek(<файл өзгериўшиси >,<элементтиң n-тәртип номери>);** -процедурасы файл көрсеткишини **n**-элементте орнатады.
4. **Truncate (<файл өзгериўшиси>);** -процедурасы оқылған файл элементиниң кейингисинен баслап қалған жазыўларды алып таслаў ушын қолланылады ҳәм файлдың жуўмақланған белгиси қойылады.

Түрленген файллар қатнасқан мәселелерди көрип шығамыз.

```
Program F1;
Var
  f: file of char;
  ch: char;
  i: integer;
begin
  assign(ф,’myfile.dat’);
  Rewrite(f);
  for i:=1 to 10 до
  begin
    readln(ch);
    write(f,ch);
  end;
  close (f);
  reset (f);
  while not EOF(f) do
  begin read(f,ch)
    write(ch,’,’)
  end;
  close (f)
end.
```

Бул программада **myfile.dat** сыртқы файлына **f** файл өзгериўшиси жәрдеминде **10** қәлеген белгилер (**Char** түриндеги) жазылады ҳәм бул белгилер үтир арқалы экранға избе-из файлдан оқып шығарылады.

```
program м;
VAR
  f:file of char;
  c: char; i:integer;
```

```
procedure cdf;
begin
  reset(f);
  for i:=1 to filesize(f) до
  begin
    read(f,ch); write (ch,’,’)
  end;
  close(f);
end;
BEGIN
  assign(f,’chfile.txt’);
  rewrite(f);
  for i:=1 to 10 do
  begin
    read(c); writeln(‘файл’)
  end;
  close(f);
END.
```

Түрленбеген файллар

Turbo Pascal программаластырыў тилинде өз алдына әҳмийетке ийе болған файллардан, яғный түрленбеген файллардан пайдаланыў мүмкин. Бул файлларды улыўмаластырылған түр деп атасақ алжаспаймыз. Файлды түрленбеген деп аталыўынан мақсет, файл түрли түрдеги мағлыўматлардан қуралған.

Түрленбеген файлларды тәрийиплеўде элементлер түри көрсетилмейди, тек **File** жәрдемши сөзинен пайдаланылады.

```
Var <файл өзгериўшиси >: File;
```

Түрленбеген файллар ушын мағлыўматларды киритиў яғный мағлыўматлар файлын жаратыў, мағлыўматларды файлдан оқыў, оператив ядда файл элементлерин қайта ислеў сыяқлы әмеллерди орынлаў мумкин.

Файлдағы элементлер түри алдыннан белгили болмағаны ушын ондағы мағлыўматлар бирдей узынлықтағы блокларға (жазыўларға) ажыратылып оқылады ҳәм усы көринисте файлға жазылады. Блок узынлығы байтларда алынады. Усы аҳыўалдан келип шыққан ҳалда

бул көринистеги файллар ушын төмендеги процедура ҳәм функцияларды пайдаланыў мумкин.

1. **Reset(F,C);** -процедураси файлды оқыў ушын ашады (жоқарыда берилген **Reset** процедурасы орныда қолланылады), бул жерде **F**-файл өзгериўшиси, **C**-ҳәр бир блок ушын белгиленген яд көлеми(байтларда алынады).
2. **Blockread(F,B,N);** -процедурасы, бул жерде **F**-файл өзгериўшиси, **N**-оқылыўы керек болған блоклар саны (**Integer**), **B**-оқылған блоклар жайластырылатуғын яддағы биринши адресс номери (**Integer**, **Word**). Бул процедура орынланғанда **F** ҳәм **B** да жайласқан **C** узынлықтағы **N** блоклар өзлестириледи.
3. **Rewrite(F,C);** -процедурасы **F** файлына **C** узынлықтағы жазыўларды жазыў ушын файлды ашады.
4. **BlockWrite(F,B,N);** -процедурасы **F** файлына оператив ядтың **B** адресли жайына **N** жазыўды жайластырады.
5. **FilePos(F)** – функциясы актив блоктың тәртип номерини анықлайды.
6. **FileSize(F)** – функциясы файлдағы блоклар узынлығын анықлап береди.

Мәселен, төмендеги программа көринисинде **F** файлын ашып оған үш блок мағлыўматларды жазыўға жәрдем береди:

```
Assign (F,'ABS.dat');
Rewrite (F,size);
BlockWrite(f,a,3);
Close(f);
```

Бул мағлыўматларды файлдан оқыў төмендеги көринис жәрдеминде орынланады:

```
Reset(F,size);
Blockread(F,A,3)
Close(f);
```

Және соны айтып өтиў лазым, түрленбеген файлларды қолланыў система айланасындағы яддан үнемли пайдаланыўға жәрдем береди.