ЛЕКЦИЯ № 3

Тақырыбы: Операциялар, операторлар, құрылымдар және бірлестіктер, функциялар. Функцияны және операцияны қайта анықтау түсінігі, динамикалық жадыны болу.

Лекция жоспары:

1. Функция
2. Динамикалық жадыны бөлу және босату
3. Getmem және Freemem процедуралары

Лекция мазмұны

1. Функция

Функция — бұл көмекші программа, яғни, бір ғана ортақ атау берілген инструкциялар қатары. Функцияның инструкцияларына өту процессі **функцияны шақыру** деп аталады. Функцияның инструкцияларынан программа дағы осы функцияны шақыру инструкциясына өту - **функциядан қайту** деп аталады.

Жалпы түрде функцияны шақыру инструкциясы мынадай көрініс те болады:

Айнымалы := Функция (Параметрлер) ; Мыналарға көңіл аудару қажет:

• әрбір функция белгілі бір типтегі мәнді қайтарады, сондықтан функцияның мәні меншіктелетін айнымалының типі функция типіне сәйкес болу керек;

• әрбір нақты функция үшін параметрлер сан мен типі қатал анықталған.

Жалпы түрде функцияның жариялануы мына көрініс те болады: function Атау (параметр1 : тип1, ..., параметрК : типК) : Тип; **var** // мұнда локал айнымалылардың сипатталуы **begin**

// мұнда функция инструкциялары
Атау := Өрнек; **end;** Мұнда:

• function — Delphi тілінің қызметші сөзі, ол өзінен кейін функция инструкциялары жазылатындығын білдіреді;

• атау — функция атауы. Программадан функция инструкцияларына өту үшін қолданылады;

• параметр — мәні функция мәнін есептеу үшін қолданылатын айнымалы. Кәдімгі айнымалыдан айырмашылығы, ол var сөзімен басталатын айнымалыларды сипаттау бөлімінде емес, функцияның атауында сипатталады. Нақты мәніне параметр программаның жұмысы барысында негізгі программадан функцияны шақыру нәтижесінде ие болады;

• тип — функцияның шақырған программаға қайтаратын мәнінің типі.

Төмендегі мысалда *isint* және *isFioat* функциялары келтірілген. *isint* функциясы редакторлау өрісіне бүтін санды енгізу барысында басылған перне символы рұқсат

етілген не етілмегендігін анықтайды. Сандар, <Enter> , <Backspace> пернелеріне рұқсат етілген деп есептеледі. *IsFloat* функциясының міндеті де осындай, бірақ, бөлшек сандар үшін. *IsFloat* функциясының екі параметрі бар: басылған перне коды және редакторлау өрісіне енгізіліп қойылған символдар қатары.

Функцияға мысал // бүтін санды енгізу кезінде енгізілген символдың

// рұқсат етілген не етілмегендігін анықтайды

function Islnt(ch : char) : Boolean;

begin if (ch >= '0'} **and** (ch <= '9') // сандар **or** (ch = 113) // <Enter> пернесі **or** (ch = #8) // <Backspace> пернесі **then** Islnt := True // символ рұқсат етілген **else** Islnt := False; // рұқсат етілмеген символ end;

// бөлшек санды енгізу кезінде енгізілген символдың // рұқсат етілген не етілмегендігін анықтайды **function** IsFloat(ch : char; st: **string)** : Boolean; **begin if** (ch >= '0') **and** (ch <= '9') // сандар

or (ch = #13) // <Enter> пернесі **or** (ch = #8) // <Backspace> пернесі **then begin**

IsFloat := True; // дұрыс символ Exit; // функциядан шығу

end;

case ch

of

'-': if Length(st) = 0 **then** IsFloat := True; ',': **if** (Pos(',',st) = 0)

and (st[Length(st)]'>= '0') and (st[Length(st)] <= '9') **then** // бұрын енгізілмеген болса үтір белгісін цифрдан соң енгізуге

болады

IsFloat := True; **else** // басқа символдарға рұқсат етілмейді

IsFloat := False; **end;**

end;

2. Динамикалық жадыны бөлу және босату

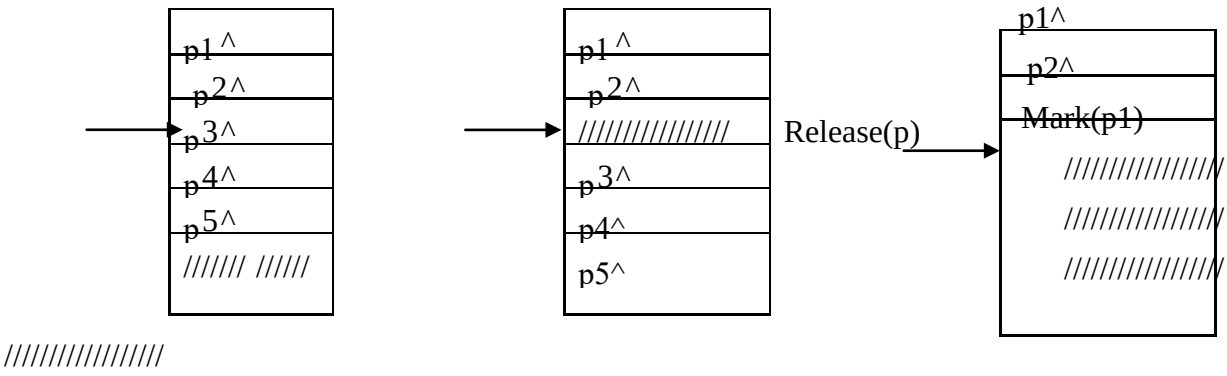
Келесі мүмкіндік үйменің бүтін бір фрагментін босатудан тұрады. Осы мақсатпен динамикалық жадыны бөлу алдынан HEAPPTR көрсеткішінің ағымдағы мәні MARK процедурасының көмегімен айнымалы-көрсеткішке сақталынып қойылады. Енді кез-келген уақытта MARK процедурасы сақтап қойған адрестен бастап динамикалық жадының соңына дейін босатуға болады. Бұл үшін RELEASE процедурасы қолданылады. Мысалы: var

p,p1,p2,p3,p4,p5 : integer;

begin

```
        new(p1);
        new(p2);
        mark(p);
        new(p3);
        new(p4);
        new(p5);
         …….
        release(p);
end.
```

Бұл мысалда MARK(P); процедурасы арқылы P көрсеткішіне HEAPPTR-ң ағымдағы мәні сақталынған, алайда айнымалы үшін жады бөлінбеді. RELEASE(P) процедурасын шақыру динамикалық жадының P көрсеткішіндегі адрестен бастап үйменің аяғына дейін босатады. Келесі суретте NEW-DISPOSE және NEW-MARK-RELEASE процедураларының механизмі, сонымен бірге RELEASE(P) операторының орнына мысалы DISPOSE(P3) қолданылған жағдай көрсетілген.



Сурет 3 – Динамикалық жадтың бөлінуі

3. Getmem және Freemem процедуралары

Біз білеміз, NEW процедурасының параметрі тек қана типтендірілген көрсеткіш бола алады. Типтендірілмеген көрсеткіштермен жұмыс істеу үшін мына процедуралар қолданылады:

GETMEM (P, SIZE) – жадыны бөлу;
FREEMEM(P, SIZE) – жадыны босату. Мұнда P – типтендірілмеген көрсеткіш;

SIZE – талап етілген не босатылатын үйме бөлігінің байттық өлшемі.

GETMEM процедурасын бір шақыруда 65521 байтқа дейін динамикалық жадыны иелеуге болады.

GETMEM-FREEMEM процедураларын қолдану жалпы динамикалық жадымен жұмыс істеу сияқты, қарапайым ережені қатал сақтауды талап етеді: қанша көлемде жады иеленсе, сонша көлемде жадыны босату керек және қай адрестен бастап иеленген болса, сол адрестен бастап босатылады. Типтендірілмеген көрсеткіштердің ендірілуі(стандартты Паскальда жоқ болатын) типтерді айқын емес түрлендіруге кең мүмкіндіктер ашады. Өкінішке орай, NEW және DISPOSE процедураларын корректсіз

қолдану типтерді күтілмеген түрлендіруге әкелуі мүмкін. Айталық, мынадай программа бар болсын:

```
var i,j :
Integer;
r : Real;
begin
new(i); {i := HeapOrg; HeapPtr:= HeapOrg + 2}
j := i; {j := HeapOrg}
j := 2;
dispose(i); {HeapPtr := HeapOrg}  new(r); {r :=
HeapOrg; HeapPtr:= HeapOrg + 6} r := pi;
WriteLn(j)  end.
```

Дисплей экранына не шығарылады? Бұл сұраққа жауап беру үшін HEAPPTR көрсеткішінің мәндерін бақылайық. Программа орындалу алдынан үйме басының адресі –HEAPORG-ке ие болған, бұл мән I көрсеткішіне ал, бұдан соң J көрсеткішіне беріледі. DISPOSE(I) орындалғаннан кейін үйме көрсеткіші қайтдан HEAPORG мәніне ие болды, осы адес NEW(R) процедурасындағы R көрсеткішіне беріледі. R адресі бойынша pi=3.14159 нақты саны орналастырылғаннан кейін, үйменің алғашқы 2 байтын осы санның ішкі көрінісінің бір бөлігі иелейді. Бұл кезде J-де әлі де HEAPORG адресі сақталынған, содықтан WRITELN(J^) операторы pi санының 2 байтын бүтін санның ішкі көрінісі деп түсінеді(өйткені J – INTEGER типті көрсеткіш) және 8578 мәнін көрсетеді.

Бақылау сұрақтары

1. Операциялар, операторлар, құрылымдар және бірлестіктер, функциялар?
2. Функцияны және операцияны қайта анықтау түсінігі?
3. Динамикалық жадыны болу?

Ұсынылатын әдебиеттер

1. Бадд Т. Объектно-ориентированное программирование в действии. Питер. 1997.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд./Пер. сс англ. –М.: «Издательство Бином», Спб.: «Невский диалект», 2001.
3. Бьярн Страуструп. Язык программирование C++. Киев: Диасофт, 1993. 1,2 часть.
4. Гамма Э. Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирование. Паттерны проектирования. – СПб: Питер, 2001.
5. Ишкова Э.А. C++ начала программирования. – М.:Бином, 2001.
6. Кетков Ю., Кетков А. Практика программирования: Visual Basic, C++ Builder, Delphi.