ЛЕКЦИЯ № 6

Тақырыбы: Жаратылысынан ие болу және композиция. Дараланған және көпшілік жаратылысынан ие болу – тілдерді қолдану принциптері, интерфестерді қолдану. Жаратылысынан ие болу түрлері.

Лекция жоспары:

1. Объекттерді топтық меншіктеу
2. Объекттердің модульде жариялану ерекшеліктері
3. SELF параметрі және тәсілді шақыру
4. Қолданушының интерфейсін құру. Қолданушы интерфейстерінің түрлері және оларды құру кезеңдері
5. Интерфейс бөлімі
6. Жүзеге асу бөлімі
7. Инициалдау бөлігі

Лекция мазмұны

1. Объекттерді топтық меншіктеу

Паскалда объекттер үшін меншіктеу операторы анықталған. Егер екі айнымалы бірдей тип-объектке ие айнымалылар ретінде жарияланса, онда былай жазуға болады:

Obj1:=Obj2;

Бұл жағдайда **Obj1**-ң барлық өріс мәндері **Obj2**-ң сәйкес өрістеріне көшіріліп өткізіледі.

Топтық меншіктеуді ретсіз пайдалануға болмайды. Кейбіреулер инкапсуляция Некоторые источники рассматривают его как посягательство на принцип инкапсуляции - ведь мы считаем (а компилятор проверяет), что эти два объекта устроены одинаково внутри, а мы, "находясь вне" обоих, не должны так считать. Алайда, Паскалда бұл бір экземплярдан екіншісіне өрістер мәнін көшірудің ең тиімді жолы.

Дельфи тілінде (объект экземплярларын көрсеткіш арқылы ғана пайдалануға болатыны бізге белгілі) топтық меншіктеулер қолданылмайды, мұның орнына параметрлі **Assign** тәсілі анықталған. Оның параметрі-объект, осы объекттің өріс мәндері қабылдап алынады. Топтық меншіктеуден айырмашылығы мынада: әрбір класста бұл тәсіл өзінше анықталған және өрістерді көшіру мүмкіндігін, көшіру процессін жұмсақ(гибко) басқаруға жағдай жасалынған.

2. Объекттердің модульде жариялану ерекшеліктері

Объектілі типтерді Паскаль тілінің модульдерінде(unit) жариялап, бұдан соң басқа модульдерде не негізгі программада пайдалануға болады. Әдетте үлкен мәселелерді шешуде осындай принципті қолданады. Модульдермен жұмыс істеу барысында класс атауы, егер ол экспортталатын болса, яғни модуль сыртында да экземпляр тудыру үшін қолданылуы тиіс болса, **interface** бөлімінде сипатталады, ал тәсілдер реализациясы - **implementation** бөлімінде.

Турбо Паскалдың 7-ші версиясында модульде жарияланған класс тәсілдері мен өрістерінің көрінуін сыртта (модуль сыртында, яғни басқа модульдерде) шектейтін құрал бар. Бұл **public** және **private** директивалары. Олар класс атауында бірнеше рет кездесуі мүмкін. Мұндай директивадан кейін өрістер мен тәсіл атаулары келтіріледі және оларға сәйкесінше **public** немесе **private** атрибуты беріледі. Үнсіз келісім бойынша (өрістер мен тәсілдер бұл директиваларды қолданбай сипатталса) **public** атрибуты беріледі.

public – класс өрістері мен тәсілдері программаның кез-келген жерінен көріне алады(олар объектте бар деп есептеледі). **private** – класс өрістері мен тәсілдері сол класс **қайсы модульде жарияланған болса сол жерде ғана көріне алады**.

Средств, чтобы сделать поля и методы видимыми только изнутри самого объекта (его методов) в Турбо Паскале нет. В Дельфи это директива **protected**. Мысалы:

```
unit A; interface
type TObj1=object
                procedure init;
        procedure done;
private
x,y:integer;
                  procedure Hidden;
         end;
var AObj:TObj1;

implementation
..... begin
  AObj.init;
  AObj.x:=10; {здесь - можно}
AObj.Hidden; end.
----------------------------
program My; uses A;
begin
  AObj.init;
 AObj.x:=10;              {а здесь так нельзя}
 AObj.Hidden; end.
```

Объекттерді жадыда орналастыру

Бір класстың барлық экземплярлары үшін жады өріске ғана бөлінеді.
Бір класстың барлық экземплярларының тәсілдері код сегментінде бір мәрте ғана жазылады, яғни, сол бір тәсіл берілген класстың барлық экземплярларына қызмет етеді. (Для всех экземпляров одного класса память выделяется только под поля.)
Объект экземпляры өз тәсілдеріне көрсеткішке ие емес (Бұған кейінірек тоқталамыз).

3. SELF параметрі және тәсілді шақыру

Тәсілді шақыру кезінде оған тағы бір параметр жіберіледі, бұл тәсілді шақырып отырған объект экземплярына алыс (дальний) 32-битті көрсеткіш.
Бұл көрсеткіш **Self** идентификаторы арқылы қолданылады.
Егер тәсіл жұмысының логикасы объекттің өзіне-өзі хабар жіберуінде болса(өз тәсілін **Self** шақырады), онда программаның оқылуын түсінкті ету үшін **Self** –ті айқын түрде көрсеткен дұрыс. Бұл параметрге шын мәнінде қажеттілік тәсіл ішінде ассемблерді пайдаланған уақытта туады. Паскалда тәсіл әрдайым алыс шақыруды қолданып шақырылады. Егер TMyObject классындағы тәсілдің атауы мынадай болса **procedure**

Init(x,y:integer); онда шақыру былай жазылған кездегідей орындалады **procedure**Init(x,y:integer;var SELF:TMyObject); және MyObject^.Init(10,20); деп шақырылған уақытта стекқа 10,20 сандары ендіріледі, бұдан соң **MyObject** көрсетіп тұрған жады орнының сегменті мен жылжуы (смещение) ендіріледі. Бұдан кейін компилятор алыс CALL командасын **Init** тәсіліне ену адресін көрсетумен бірге генерация жасайды (адрес бұл жағдайда қатаң түрде компилятор арқылы жазылады, және ешқандай таблицадан таңдап алынбайды).

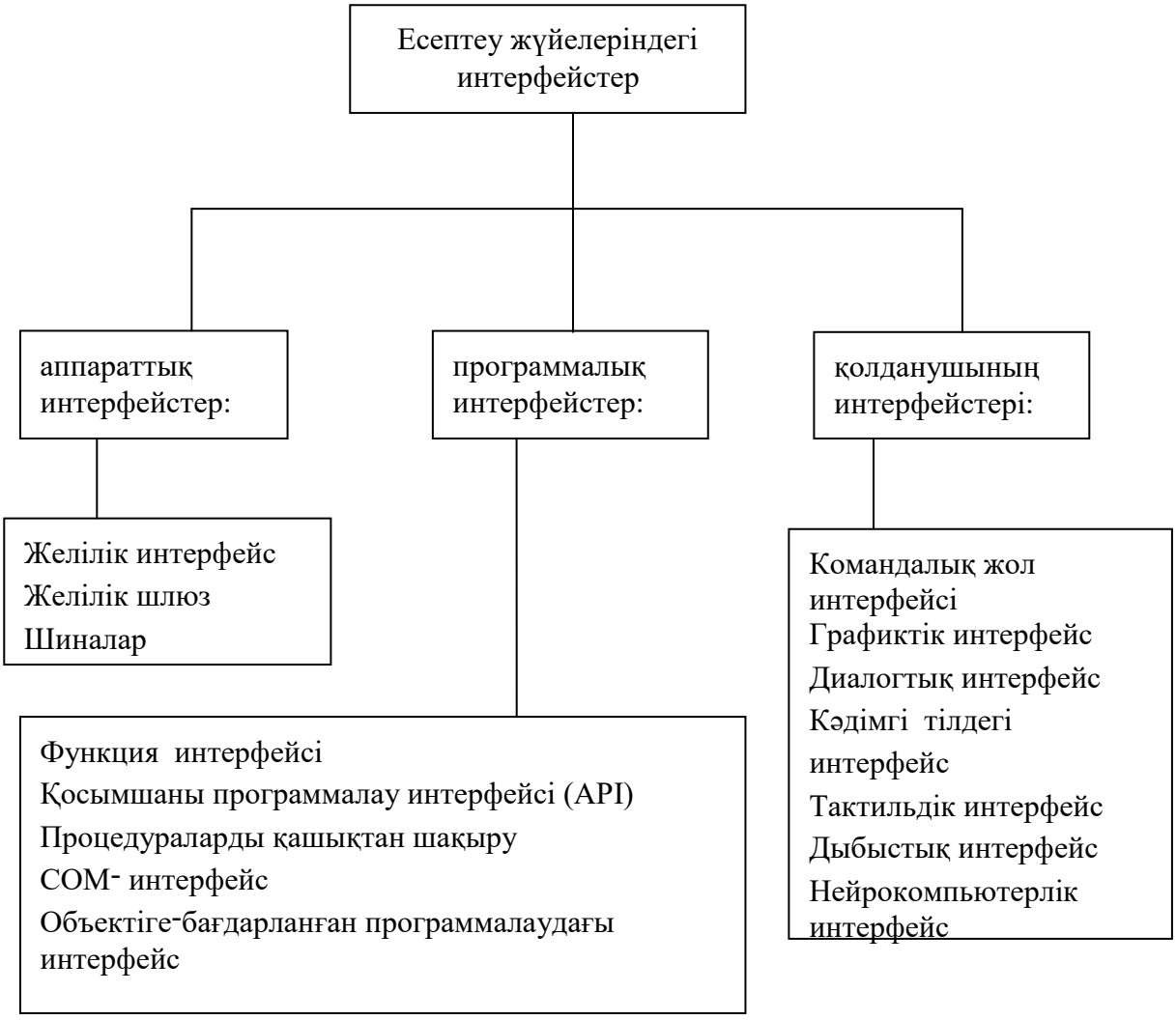
4. Қолданушының интерфейсін құру. Қолданушы интерфейстерінің түрлері және оларды құру кезеңдері

Программалау технологиясында, қолданушының компьютердің программалық және аппараттық жабдықтарымен жұмыс жасау үшін пайдаланатын әдістері мен құралдарының жиынтығын «қолданушының интерфейсі» деп атау келісілген.

Қазіргі заманауи ақпараттық жүйелердің жұмыс жасау мүмкіндіктері қолданушының интерфейстерімен тығыз байланысты. Мысалы, қандай да болмасын объектінің (компьютер, программалық жабдық немесе подпрограмма) интерфейсі өзгермейтіндей тұрақты болса, онда объектінің өзін өзгертуге көбірек мүмкіндік болар еді, яғни оның басқа объектілермен (мысалы, қолданушымен) қарым-қатынас қағидалары қарастырылмайды.

Есептеу жүйелеріндегі интерфейстерді компьютерлік техникаға қатынас тұрғысынан келесі топтарға бөледі (3.14- сурет):

- аппараттық интерфейс;
- программалық интерфейс;
- қолданушының интерфейсі.



Сурет 4 - Есептеу жүйелеріндегі интерфейстер

Қолданушының интерфейсі барынша қолданушыға жақын болуы қажет, себебі қолданушының назары программада емес, керісінше өзінің жасап жатқан құжатында болуы керек. Мысалы, жиі қолданылатын командалардың бірі 2-3 деңгейлерде яғни басқа бір ішкі мәзірлерде орналасса , оны іздеуге қолданушының біраз уақыты кетіп отыратын болса, онда интерфейсті қолданушыға жақын немесе ыңғайлы деп айтуға келмейді. Программалық жабдық үшін интерфейстің маңызы өте зор. Программа қанша жерден жақсы бола берсін, егер онымен жұмыс жасау ыңғайсыз болса, онда оны қолданушы қабылдай алмайды.

Қолданушы интерфейстерінің қазіргі көп тарағаны- графиктік интерфейс. Мысалы, Windows операциялық жүйесінің көп таралуының бір себебі, оның қолданушыға ыңғайлы графиктік интерфейсінің болуы деп айтылып жүр. Қолданушының графиктік интерфейсі (Graphical User Interface, GUI) 1970-ші жылдардың ортасында, Xerox Palo Alto Research Center-де (PARC), Smalltalk ортасында жұмыс істейтін Alto және Star машиналары үшін жасалған алғашқы жұмыстардан басталды. Кейде оны «визуальды интерфейс» немесе
«терезелі графикалық орта» деп атайды. Қазіргі заманауи жүйелердің көпшілігінде, мысалы, Microsoft Windows, Mac OS, Solaris, GNU/Linux, NeXTSTEP, OS/2, BeOS, Android, iOS, Bada, MeeGo жүйелерінде қолданушының графиктік интерфейстері пайдаланылған.

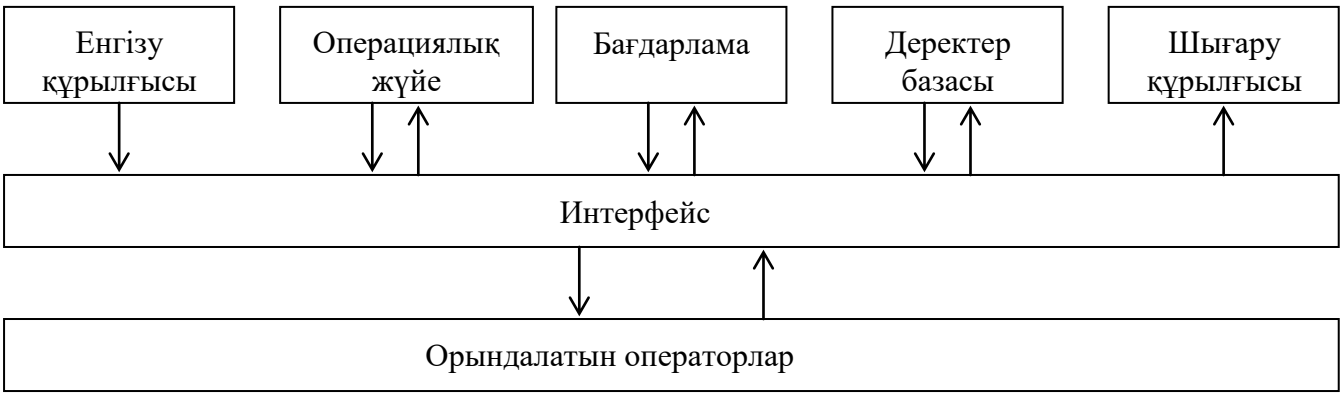
Қолданушының графиктік интерфейсі графиканы растрлық экран дисплейінде қолдануға мүмкіндік береді. Графика экранда элементтердің нақты орнын көрсетеді, ақпаратты берудің визуальды ортасы және көрнекі графика және форматталған мәтіндер барлығы бірігіп «не көрсең соны аласың» (WYSIWYG - What you see is what you get) мүмкіндігін береді.

Қазіргі қолданушылар компьютерді үйрену және жаңа программаларды меңгеру үшін көп уақыт шығындамайды. Windows үшін жазылған программалық жабдықтардың көбінде дерлік бір типтес қолданушы интерфейстері пайдаланылатындықтан, олар қолданушы үшін бір мағынада қабылданады. Windows жүйесі осыған көмектеседі. Windows үшін жазылған программалық жабдықтардың қолданушыға арналған интерфейсінің негізгі элементтері: терезелер, мәзірлер, сұхбат терезелері, суретті батырмалар және т.б. болып табылады.

Қолданушының интерфейсін құру кезеңдері программалық жабдықты құзу кезеңдерімен сәйкес келеді және есептің қойылуы, талаптар мен спецификацияларды анықтау, жобалау, жүзеге асыру, тестілеу мен жөндеу секілді кезеңдерді қамтиды.

Интерфейс бұл объектілердің жиынтығы, оның көмегімен ақпаратты белгілі бағытқа жіберу асырылады, - пішін, қарым – қатынас терезелері, басқару элементтері және т.б. Көптеген жағдайларда таңдап алынған интерфейс барлық бағдарламаның құрылымын анықтайды.

Объектілер бағдарламада жеткілікті түрде автономды болғандықтан, олардың арасында ақпаратты жіберу, сонымен бірге бағдарлама мен операциялық жүйе, бағдарлама мен сыртқы құрылғылар т.б. арасында ақпаратты беру үшін ***хабарлама жүйесі*** қолданылады.



Сурет 5 - Бағдарламаның сыртқы құрылғылармен әрекеттесуі

Мұндай бағдарламаның жұмысы да өз сипатында. Әдетте ол іске қосылғаннан кейін келесі хабарламаны күту режимінде тұрады. Хабарлама пайда болғанда бағдарлама оған талдау жасайды, қандай әрекет жасау керек екенін анықтайды, одан кейін сол әрекетті орындап, келесі хабарламаны күтеді. Әртүрлі хабарлама пайда болғаннан кейін бағдарлама шарт бойынша әртүрлі әрекет жасайды, барлық орындалатын операторлар бағыныңқы бағдарламалардың жиынтығын береді, олар әртүрлі объектінің әдістері ретінде бекітіледі (хабарламаны және оқиғаны өңдеу әдістерін қоса алады), немесе өздік бағыныңқы бағдарлама. Нақтылы түрде әртүрлі операциялық жүйелер және оларды баптау, олар белгілі командаларды күтеді, содан кейін оларды өңдеуге кіріседі.

5. Интерфейс бөлімі

Интерфейс бөлімі Interface қызметші сөзімен ашылады. Бұл бөлімде модульдің басқа модульдер мен негізгі программада пайдалануға болатын барлық глобал объектілері жарияланады. Глобаль көмекші программалардың интерфейс бөлімінде тек атаулары ғана жарияланады. Мысалы:

```
Unit Cmplx;
Interface Type
complex=record
re, im:real end;
Procedure AddC(x,y:complex; var z:complex);
Procedure Mulc(x,y:complex; var z:complex);
```

Бұл программада complex типі және AddC, Mulc процедуралары глобал болып табылады. Интерфейс бөлімінде көмекші программаларды жариялау авоматты түрде жадының ұзақ (дальней) моделін қолданумен компиляцияланады. Негізгі программадан және басқа модульдерден көмекші прогарммаларға қолжеткізу осылайша жүзеге асады. Ескере кету керек, модульдің интерфейс бөлімінде жарияланған барлық константалар мен айнымалылар негізгі программаның глобал константалары мен айнымалылары сияқты Турбо Паскальдің ортақ берілгендер сегментіне компилятор арқылы жайғастырылады (сегменттің максимал ұзындығы 65536 байт). Жариялаудың түрлі бөлімдерінің пайда болуы және олардың саны қалауымызша болуы мүмкін. Егер интерфейс бөлімінде сыртқы көмекші программалар жарияланса, модульдің **жүзеге асу** бөлімінде олардың атауларынан кейін **денелері** жазылуы керек (яғни EXTERNAL қызметші сөзі). Егер интерфейс бөлімінде машиналық кодтағы көмекші программа жарияланса модульдің **жүзеге асу** бөлімінде оның атауынан кейін INLINE сөзі мен машиналық коды келтіріледі. Модульдің интерфейс бөлімінде *алдын-ала сипатталуды* қолдануға болмайды.

6. Жүзеге асу бөлімі

Жүзеге асу бөлімі IMPLEMENTATION қызметші сөзімен басталады және интерфейс бөлімінде жарияланған көмекші программалардың тексті жазылады. Мұнда модульүшін локал болатын объекттер – қосымша типтер, константалар, айнымалылар және блоктар, егер жүзеге асу бөлімінде қолданылатын болса онда белгілер де жариялануы мүмкін

Модульдің интерфейс бөлімінде жарияланған көмекші программаның атауы орындалу бөлімінде келтірілуі тиіс. Мұнда формальді айнымалылардың тізімін және функция типін көрсетпей тастап кетуге болады. Өйткені олар интерфейс бөлімінде сипатталған. Егер атау толық түрде жазылса, онда ол интерфейс бөлімінде жарияланған атаумен түгел бірдей болуы керек. Unit Cmplx; Interface type complex = record re, im : real end;

```
Procedure AddC (x, y : complex; var z : complex);
Implementation
Procedure AddC;  begin
z.re := x.re +Y.re;
z.im := x.im +y.im
end;
end.
```

Локал айнымалылар және константалар, сонымен бірге модульді компиляциялау нәтижесінде туындаған барлық программалық код жадының жалпы сегментіне орналастырылады (в общий сегмент памяти) .

7. Инициалдау бөлігі

Инициалдау бөлігі модульді аяқтайды. Инициалдау бөлігі BEGIN қызметші сөзінен басталады. Инициалдау бөлігі бос болуы да мүмкін, бұл жағдайда BEGIN –нен кейін бірден END. сөзі жазылады, немесе бүткілдей жоқ болуы мүмкін.

Инициалданатын бөлімде программа фрагменті боп табылатын орындалатын операторлар жазылады. Бұл операторлар басқаруды негізгі программаға бергенше орындалады және әдетте оның жұмысына дайындық үшін қолданылады. Мысалы, мұнда айнымалылар инициалдануы, қажет файлдар ашылуы, өзара байланыстары (коммуникациалық каналдар т.б.
бойынша) орнатылуы мүмкін.

```
Unit FileText;
Interface
Procedure Print(s : string);
Implementation
var  f: text;
const
name = 'output.txt'; Procedure Print;
begin  WriteLn(f, s)   end;
{ Начало инициирующей части: }
begin  assign(f, name);  rewrite(f);
{ Конец инициирующей части }   end.
```

Инициалдау бөлімін бос етуге кеңес берілмейді, одан да жазбаған дұрыс (begin end.-ті). Бос бөлімде бос оператор бар болады, програмамны іске қосқан кезде басқару осы операторға беріледі. Бұл оверлейлік программаларды құру кезінде қиындықтар туғызады.

Бақылау сұрақтары

1. Жаратылысынан ие болу және композиция?
2. Дараланған және көпшілік жаратылысынан ие болу?
3. Тілдерді қолдану принциптері?

Ұсынылатын әдебиеттер

1. Дьюхарсг С., Старк К. Программирование на С++. Пер. с англ. Киев. Диасофт, 1993-272с.
2. Евангелос, Петрусос. Visual Basic и 6 VBA. Питер, 2000.
3. Йордон Э., Аргила Карл. Структурные модели в объектно-ориенттированном анализе и проектировании./Пер. с англ. П.Быстрова. М.:ЛОРИ, 1999.
4. Калверт Ч.Самоучитель по программированию на С++ Builder. Киев, 2000 г.

Гамма Э. Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирование. Паттерны проектирования. – СПб: Питер, 2001.