

Абстрактные классы

- Абстрактным классом называется класс, в котором есть хотя бы одна *чистая* (пустая) виртуальная функция, которой соответствует следующее определение

```
virtual тип имя_функции(список формальных параметров) = 0;
```

- В этой записи конструкция "**=0**" называется чистый спецификатор

```
virtual void fpure(void) = 0;
```

-
- ***Чистая виртуальная*** функция "ничего не делает" и недоступна для ВЫЗОВОВ
 - Ее назначение - служить основой для подменяющих ее функций в производных классах
 - Исходя из этого становится понятной невозможность создания самостоятельных объектов абстрактного класса
 - Абстрактный класс может использоваться только в качестве базового для производных классов
-

Абстрактный класс:

```
class B
{
    protected:
        virtual void func(char) =0;
        void sos(int);
};
```

На основе класса **V** можно по-разному построить производные классы:

```
class D: public B { void func(char); };  
class E: public B { void sos(int); };
```

- В классе **D** чистая виртуальная функция **func()** заменена конкретной виртуальной функцией того же типа. Функция **B::sos()** наследуется классом **D** и доступна в нем и в его методах. Класс **D** не абстрактный
 - В классе **E** переопределена функция **B::sos()**, а виртуальная функция **B::func()** унаследована. Тем самым класс **E** становится абстрактным и может использоваться только как базовый
-

- Механизм абстрактных классов разработан для представления общих понятий, которые в дальнейшем предполагается конкретизировать
- Эти общие понятия обычно невозможно использовать непосредственно, но на их основе можно, как на базе, построить частные производные классы, пригодные для описания конкретных объектов
- Например, из абстрактного класса "**фигура**" можно сформировать класс "**треугольник**", "**окружность**" и т.д. В качестве примера рассмотрим программу, в которой на основе базового класса **point** построен абстрактный класс **figure**

-
- В классе **figure** определены: конструктор, чистая виртуальная функция **show()** для вывода изображения фигуры, например, на экран дисплея
 - Кроме того, в класс входят методы **hide()** - убрать изображение фигуры с экрана дисплея и **move()** - переместить изображение фигуры в заданную точку экрана
 - Функции **hide()** и **move()** обращаются к чистой виртуальной функции **show()**
 - Однако реальное выполнение **show()** возможно только после создания производного класса, в котором чистая виртуальная функция **show()** будет подменена компонентной функцией для изображения конкретной фигуры
 - Введем класс **point**, определяющий понятие "точка на экране дисплея". Разместим описание класса в отдельном файле с именем **point.h**
-

```
// point.h - определение функций класса
```

```
class point
{ int x,y;
  public:
    point(int, int);
    int& point::givex(void);
    int& point::givey(void);
    void point::show(void);
    void point::hide(void);
    void point::move(int xn, int yn);
};
```

```
// POINT.CPP - определение функций класса
```

```
#ifndef POINTCPP
#define POINTCPP 1
#include <graphics.h>
#include "point.h"

point::point(int xi, int yi) { x = xi; y = yi; }
int& point::givex(void) { return x; }
int& point::givey(void) { return y; }
void point::show(void)
    { putpixel(x, y, getcolor()); }
void point::hide(void)
    { putpixel(x, y, getbkcolor()); } // Цвет фона
void point::move(int xn, int yn)
    { hide(); x = xn; y = yn; show(); }
#endif
```

Программа для иллюстрации работы
с классом **point**

```
//P9-04.CPP - работа с классом "точка на экране"
```

```
#include <graphics.h>
```

```
#include <conio.h> // Для getch();
```

```
#include "point.cpp" // Определение класса point
```

```
void main() {
```

```
    point A(200,50) ; // Создается невидимая точка A
```

```
    point B; // Невидимая точка B с нулевыми  
            // координатами
```

```
    point D(500,200) ; // Создается невидимая точка D
```

```
    int dr = DETECT, mod;
```

```
    initgraph(&dr, &mod, "c:\\borlandc\\bgi") ;
```

```
    A.show(); // Показать на экране точку A
```

```
    getch(); // Ждать нажатия клавиши
```

```
    B.show(); // Показать на экране точку B
```

```
    getch(); D.show(); // Показать на экране точку D
```

```
    getch(); A.move(); // Переместить точку A
```

```
    getch(); B.move(50, 60) ; // Переместить точку B
```

```
    getch(); closegraph(); // Закрыть графический режим
```

```
}
```

Определим абстрактный класс
figure (в файле **figure. cpp**):

```
//FIGURE.CPP - абстрактный класс
```

```
#include "point.cpp"
class figure: public point {
public:
        // Конструктор абстрактного класса figure
figure (point p): point(p.givex(), p.givey() ) { }
virtual void show()=0; // Чистая виртуальная функция
void hide()          // Функция для скрытия изображения
{ int bk, cc;
  bk = getbkcolor(); cc = getcolor (); setcolor(bk);
  show();          // Обращение к чистой виртуальной функции
  setcolor(cc);
}
void move(point p) // Перемещение фигуры в точку "p"
  { hide (); x = p.givex(); y = p.givey(); show(); }
};
```

На базе класса **figure** определим неабстрактные
классы:

```
//ELLIPS.FIG - конкретный класс "эллипс"
```

```
class ellips : public figure {  
    int rx,ry;  
    public:  
        // Конструктор:  
    ellips (point d, int radx, int rady): figure(d)  
        { rx = radx; ry = rady; }  
    void show()  
        { ellipse(x,y,0,360,rx,ry); return; }  
};
```

//CIRC.FIG - конкретный класс "окружность"

```
class circ: public figure {
    int radius;
public:
    circ(point e, int rad): figure(e)
        {radius = rad; } // Конструктор:
    void show() { circle(x,y,radius); }
};
```

В следующей программе используются все три
класса:

```
//P10-08.CPP - абстрактные классы и чистые виртуальные функции
```

```
#include <graphics.h>
```

```
#include "figure.cpp"
```

```
#include "circ.fig"
```

```
#include "ellips.fig"
```

```
#include <stdio.h>
```

```
void main(void) {
```

```
    point A(100,80), B(300,200);
```

```
    circ C(A, 60);
```

```
    ellips E(B,200,100);
```

```
    { int dr = DETECT, mod;
```

```
        initgraph(&dr, &mod, "c:\\borlandc\\bgi");
```

```
        A.show(); getch(); // Изобразить точку
```

```
        B.show(); getch(); // Изобразить точку
```

```
        C.show(); getch(); // Показать окружность
```

```
        E.show (); getch(); // Показать эллипс
```

```
        C.move(B); getch(); // Совместить фигуры
```

```
        E.hide(); getch(); // Убрать эллипс
```

```
        C.hide0 ; getch0 ; // Убрать окружность
```

```
    }
```

```
closegraph(); }
```

- В программе на базе класса **figure** определены два производных класса: **circ** (окружность) и **ellips** (эллипс)
- Для обоих классов унаследованный класс **point** определяет центры фигур
- В обоих классах определены конкретные методы **show()** и из абстрактного класса **figure** унаследованы функции **move()** и **hide()**
- Комментарии к операторам основной программы содержат полные (квалифицированные) имена исполняемых функций

- По сравнению с обычными классами абстрактные классы пользуются "ограниченными правами". Как говорилось, невозможно создать объект абстрактного класса. Абстрактный класс нельзя употреблять для задания типа параметра функции или в качестве типа возвращаемого функцией значения. Абстрактный класс нельзя использовать при явном приведении типов. В то же время можно определять указатели и ссылки на абстрактные классы
- Объект абстрактного класса не может быть формальным параметром функции, однако формальным параметром может быть указатель абстрактного класса. В этом случае появляется возможность передавать в вызываемую функцию в качестве фактического параметра значение указателя на производный объект, заменяя им указатель на абстрактный базовый класс.