# Лабораторная работа № 5 Тема. Наследование

**Теоретическое введение.** Важнейшим принципом ООП, реализованным в С++, является *наследование.* Класс, который наследуется, называется базовым классом, а наследующий класс – производным классом. Указателю на объект базового класса можно присвоить значение указателя на объект производного класса.

Все члены класса из разделов ***public*** и ***protected*** наследуются, а члены из раздела ***private –*** нет***.*** Члены раздела ***protected*** являются частными для базового и производного классов. При наследовании базового класса к его членам также может объявляться спецификатор ***доступ***.

**class имя\_класса: *доступ* имя\_базового\_класса**

**{/\*члены класса\*/}**

Cпецификатор ***доступ*** базового класса при наследовании может принимать одно из трех значений: ***public*** (по умолчанию), ***private, protected.*** Если спецификатор принимает значение ***public,*** то все члены разделов ***public*** и ***protected*** базового класса становятся членами разделов ***public*** и ***protected*** производного класса. Если доступ имеет значение ***private***, то все члены разделов ***public*** и ***protected*** становятся членами раздела ***private*** производного класса. Если доступ имеет значение ***protected***, то все члены разделов ***public, protected*** становятся членами раздела ***protected*** производного класса.

Конструкторы и деструкторы базовых классов не наследуются**,** однако при создании объектов производных классов конструкторы базовых классов выполняются в порядке наследования, а деструкторы в обратном порядке. При необходимости передачи аргументов конструктору базового класса из производного класса используется следующий синтаксис:

к**онструктор\_производного\_класса(арг):base(арг)**

**{/\*тело конструктора производного класса\*/}**

**Множественное наследование.** Один класс может наследовать атрибуты двух и более базовых классов, которые перечисляются после двоеточия через запятую. Если базовые классы содержат конструкторы, то они вызываются поочередно в порядке перечисления.

**Виртуальные функции и полиморфизм.** Механизм виртуальных функций в ООП используется для реализации полиморфизма: создания метода, предназначенного для работы с различными объектами за счет механизма позднего связывания *(late binding).* Виртуальные функции объявляются в базовом и производных классах с ключевым словом ***virtual****.* При этом каждый объект класса, управляемого из базового класса с виртуальными функциями, имеет указатель на *vmtbl (virtual method table)*, содержащую адреса виртуальных функций. Эти адреса устанавливаются в адреса нужных для данного объекта функций во время выполнения.

В отличие от перегружаемых функций виртуаль­ные объявляются в порожденных классах с тем же именем, возвращаемым значением и типом аргументов. Если различны типы аргументов, виртуальный механизм игнорируется. Тип возвращаемого значения переопределить нельзя.

Основная идея в использовании виртуальной функции состоит в следующем: она может быть объявлена в базовом классе, а затем переопределена в каждом производном классе. При этом доступ через указатель на объект базового класса осуществляется к этой функции из базового класса, а доступ через указатель на объект производного класса – из производного класса. То же происходит при передаче функции объекта производного класса, если аргумент объявлен как базовый класс.

*Абстрактные классы* – это классы, содержащие *чисто абстрактные виртуальные функции*. Чисто абстрактные виртуальные функции при объявлении в классе приравниваются к нулю. Абстрактные классы используются только для наследования, так как объекты таких классов не могут быть созданы.

**Пример.** Создаются три класса: MyPoint (базовый класс – «Точка»), MyEllipse (производный класс от класса MyPoint – «Окруж­ность») и MyCylinder (производный класс от класса MyEllipse – «Ци­линдр»). Классы объ­явлены в файле MyClass.h. Используются кнопки: Button1 – очистка области для рисования (компонент Image1), Button2 – создание объекта класса MyPoint и рисование точки, Button3 – соз­дание объекта класса MyEllipce и рисование окружности, Button4 – создание объекта класса MyCylinder и рисование цилиндра, компоненты СSpinEdit1 – СSpinEdit4 для задания координат точки и размеров фигур, а также перемещения объектов в области рисования.

**Заголовочный файл** Myclass.h

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class MyPoint \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MyPoint {

int x,y;

public:

int GetX(){return x;}

int GetY(){return y;}

void SetX(int x1){x=x1;}

void SetY(int y1){y=y1;}

MyPoint(int xx=0,int yy=0){x=xx;y=yy;}

virtual void Show();

virtual bool Checking();

// Проверка, не выходит ли точка за пределы области рисования ?

virtual void MoveTo(int x1,int y1);

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*classs MyEllipse \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MyEllipse:public MyPoint {

int xa,yb;

public:

int GetXA(){return xa;}

int GetYB(){return yb;}

void SetXA(int xa1){xa=xa1;}

void SetYB(int yb1){yb=yb1;}

MyEllipse(int a=0,int b=0,int c=0,int d=0):MyPoint(a,b)

{xa=c;yb=d;}

virtual void Show();

virtual bool Checking();

// Проверка, не выходит ли окружность за пределы области рисования ?

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class MyCylinder \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MyCylinder:public MyEllipse {

int h;

public:

int GetH(){return h;}

void SetH(int h1){h=h1;}

MyCylinder(int X=0,int Y=0,int R=0,int H=0):MyEllipse(X,Y,R,R/2)

{h=H;}

void Show();

bool Checking();

// Проверка, не выходит ли цилиндр за пределы области рисования ?

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class MyCylinder \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool ExistFigure=false;//Если ExistFigure==false, объект не изображался

MyPoint \*Figure;//- Указатель на базовый класс MyPoint.

**Файл** Unit1.Cpp

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Myclass.h"

#pragma package(smart\_init)

#pragma link "CSPIN"

#pragma resource "\*.dfm"

TForm1 \*Form1;

\_\_fastcall TForm1::TForm1(TComponent\* Owner): TForm(Owner)

{ }

//\*\*\*\*\*\*\*\*\*\* Реализация класса MyPoint \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MyPoint::Show()

( Form1->Image1->Canvas->Pixels[x][y]=clRed;}

bool MyPoint::Checking(){

//- Проверка, не выходит ли точка за пределы области рисования ?

if((x>=0)&&(x<=Form1->Image1->Width)&&(y>=0)&&

(y<=Form1->Image1->Height))

return true; else return false;

}

void MyPoint::MoveTo(int x1,int y1)

{ // если выходим за область рисования, то точку не сдвигаем.

int a=x,b=y;

x=x1;y=y1;

if(!Checking()){x=a;y=b;}

}

//\*\*\*\*\*\*\*\*\*\*\*Реализация класса MyEllipse \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MyEllipse::Show()

{

Form1->Image1->Canvas->Ellipse(GetX()-xa,GetY()-yb,GetX()+xa,GetY()+yb);

}

// Проверка, не выходит ли окружность за пределы области рисования ?

bool MyEllipse::Checking()

{

int rabx=GetX(),raby=GetY();

SetX(GetX()-xa);SetY(GetY()-yb);

if(MyPoint::Checking())

{

SetX(GetX()+2\*xa);SetY(GetY()+2\*yb);

if(MyPoint::Checking())

{

SetX(rabx);SetY(raby); // восcтанавливаем координаты x,y.

return true;

}

}

SetX(rabx);SetY(raby); // восcтанавливаем координаты x,y.

return false;

}

//\*\*\*\*\*\*\*\*\*\*Реализация класса MyCylinder \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MyCylinder::Show()

{

MyEllipse::Show();

Form1->Image1->Canvas->MoveTo(GetX()-GetXA(),GetY());

Form1->Image1->Canvas->LineTo(GetX()-GetXA(),GetY()-h);

Form1->Image1->Canvas->MoveTo(GetX()+GetXA(),GetY());

Form1->Image1->Canvas->LineTo(GetX()+GetXA(),GetY()-h);

SetY(GetY()-h);

MyEllipse::Show();

SetY(GetY()+h);

}

bool MyCylinder::Checking()

{

if(MyEllipse::Checking())

{

SetY(GetY()-h);

if(MyEllipse::Checking()){SetY(GetY()+h);return true;}

}

return false;

}

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{ // Чистим "графическое" окно

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

ExistFigure=false;

}

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

if(ExistFigure==true)delete Figure;

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

// Рисуем точку, проверив её попадание в область рисования

Figure=new MyPoint(CSpinEdit1->Value,CSpinEdit2->Value);

ExistFigure=true;

if(Figure->Checking()) Figure->Show();

else ShowMessage("Точка не попадает в область рисования!");

}

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

if(ExistFigure==true) delete Figure;

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

// Рисуем окружность, проверив её попадание в область рисования

Figure=new MyEllipse(CSpinEdit1->Value,CSpinEdit2->Value,

CSpinEdit3->Value,CSpinEdit3->Value);

ExistFigure=true;

if(Figure->Checking())Figure->Show();

else ShowMessage("Окружность не помещается в область рисования!");

}

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

if(ExistFigure==true) delete Figure;

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

// Рисуем цилиндр, проверив его попадание в область рисования

Figure=new MyCylinder(CSpinEdit1->Value,CSpinEdit2->Value,

CSpinEdit3->Value,CSpinEdit4->Value);

ExistFigure=true;

if(Figure->Checking())Figure->Show();

else ShowMessage("Цилиндр не помещается в область рисования!");

}

void \_\_fastcall TForm1::CSpinEdit1Change(TObject \*Sender)

{

if(ExistFigure==true){

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

Figure->MoveTo(CSpinEdit1->Value,CSpinEdit2->Value);

Figure->Show();

}

}

void \_\_fastcall TForm1::CSpinEdit2Change(TObject \*Sender)

{

if(ExistFigure==true){

Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));

Figure->MoveTo(CSpinEdit1->Value,CSpinEdit2->Value);

Figure->Show();

}

}

Задания для самостоятельного решения

При выполнении данной работы необходимо определить базовый класс и производные от него классы. Предусмотреть передачу аргументов конструкторам базового класса; использование виртуальных и перегруженных функций; обработку исключительных ситуаций.

Вариант А

В следующих заданиях требуется создать базовый класс (как вариант абстрактный базовый класс) и определить общие методы show ( ), get ( ), set ( ) и другие, специфические для данного класса. Создать производные классы, в которые добавить свойства и методы.

Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Объекты произ­водных классов идентифицировать конструктором по имени или идентификационному номеру.

Вызвать метод show ( ) базового класса и просмотреть массив объектов.

Использовать объекты для моделирования реальных ситуаций.

1. Создать базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость пере­возки пассажиров и грузов каждым транспортным средством.

2. Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.

3. Создать аналогичный базовый класс «Пассажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения.

4. Изменить задания 1–3, чтобы базовый класс стал абстрактным. Сделать некоторые методы абстрактными.

5. Создать базовый класс «Учащийся» и производные классы «Школьник» и «Студент». Создать массив объектов базового класса и заполнить этот массив объектами. Показать отдельно студентов и школьников.

6. Создать базовый класс «Музыкальный инструмент» и производные классы «Ударный», «Струнный», «Духовой». Создать массив объектов «Оркестр». Выдать состав оркестра, переопределив метод.

7. Определить базовый класс «Множество» и производный класс «Кольцо» (операции сложения и умножения обе коммутативные и ассоциативные, связанные законом дистрибутивности; сложение обладает обратной операцией – вычитанием). Ввести кольца целых чисел, многочленов, систему классов целых чисел, сравнимых по модулю. Кольцо является полем, если в нем определена операция деления, кроме деления на нуль. Рациональные числа, дробно рациональные функции.

8. Создать абстрактный класс «Работник фирмы» и производные классы «Менеджер», «Администратор», «Программист».

9. Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установить имя каждого животного и его характеристики.

10. Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша» и др. С помощью конструк­тора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Вариант Б

1. Создать класс Item (единица хранения в библиотеке), содержащий данные-члены: invNumber – инвентарный номер и taken – взято на руки или имеется в наличии, а также методы:

virtual void Show(); //показать информацию о единице хранения

bool isAvailable(); // есть ли единица хранения в наличии ?

int GetinvNumber(); //возвращает инвентарный номер

void Take(); // операция «взять»

void Return(); // операция «вернуть»

Построить производные классы Book и Magazin. Класс Book содержит данные-члены: author, title, publisher, year и методы: Author();

Title(); Publisher(); YearOf Publishing(); Show().

Класс Magazin включает данные-члены: volume; number; year; title и методы: Volume(); Title(); Number(); Year(); Show().

2. Создать базовый класс Polygon (многоугольник). Класс должен содержать методы для рисования многоугольника, вычисления периметра, нахождения площади и др. Построить производный класс Triangle (треугольник), содержащий также методы для нахождения точки пересечения медиан, длин медиан, длин биссектрис, координат точек пересечения биссектрис, высот треугольника.

3. Создать абстрактный класс Shape для рисования плоских фигур. Построить производные классы Square (квадрат, который характеризуется координатами левого верхнего угла и длиной стороны), Circle (окружность с заданными координатами центра и радиусом), Ellipse (эллипс с заданными координатами вершин описанного вокруг него прямоугольника), позволяющие рисовать указанные фигуры, а также передвигать их на плоскости.

4. Создать класс CPoint – точка и производные от него классы CcoloredPoint и CLine. На основе классов CcoloredPoint и CLine со-здать класс CcoloredLine. Все классы должны иметь методы для установки и получения значений всех координат, а также изменения цвета и получения текущего цвета.

5. Описать базовый класс Stroka. Обязательные данные-члены класса: указатель типа char – для хранения строки; значение типа int – длина строки.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметра C-строку (заканчивается нулевым байтом); конструктор копирования; получение длины строки; очистка строки (сделать строку пустой); деструктор.

Описать производный класс «БИТОВАЯ\_СТРОКА» (строки данного класса могут содержать только символы ‘0’ и ‘1’). Если в основе инициализирующей строки встретятся любые символы, отличные от допустимых, то БИТОВАЯ\_СТРОКА становится пустой. Содержимое строки рассматривается как двоичное представление целого числа со знаковым разрядом. Отрицательные числа хранятся в дополнительном коде.

Обязательные методы: конструктор без параметров; конструктор, принимающий в качестве параметра C-строку; конструктор копирования; деструктор; изменение знака числа (перевод числа в дополнительный код).

Переопределить следующие операции (длина строки результата в случае необходимости расширяется влево знаковым разрядом): присваивание; сложение (+); проверка на равенство (==).

6. Создать производный класс «СТРОКА10» (целое неотрицательное десятичное число) от класса «СТРОКА» (описание приведено выше).

Методы: конструктор без параметров; конструктор, принимающий в качестве параметра C-строку; конструктор копирования; деструктор; метод, определяющий, можно ли представить данное число в формате int; метод, определяющий, равно ли число нулю; метод, возвращающий представление числа в виде целого (int); метод, удаляющий незначащие нули.

Переопределить операции: сложение (+); проверка на больше (по значению) (>); проверка на меньше (<); присваивание (=).

7. Создать производный класс «БУЛЕВ ВЕКТОР» (**BoolVector**) от класса **Vector**. Компоненты принимают значения из множе­ства {0,1}.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров указатель на массив це­лого типа (если эле­менты массива содержат числа, отличные от 0 и 1, то создается пустой вектор) и размер вектора; конструктор копирования; деструктор; метод, возвращающий число единиц в векторе; метод, возвращающий позицию самой левой единицы в векторе.

Переопределить операции: поразрядная конъюнкция (&); поразрядная дизъюнкция (|); поразрядная инверсия (~); поразрядная операция ИСКЛЮЧАЮЩЕЕ ИЛИ (^); присваивание (=).

8. Создать производный класс «ТРОИЧНЫЙ ВЕКТОР» от класса **Vector**. Компоненты принимают значения из множества {0,1,2}.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров указатель на массив целого типа и размер вектора; конструктор копирования; деструктор; проверка двух векторов на ортогональность (два троичных вектора называются ортогональными, если в них существует пара одноименных компонент, имеющих в одном из векторов значение 0, а в другом – 1); метод, возвращающий число компонент в векторе, принимающих значение 2.

Переопределить операции: присваивание (=); поразрядная конъюнкция (пересечение) двух не ортогональных векторов (&): 0&0=0, 1&1=1, 2&2=2, 0&2=0, 2&0=0, 1&2=1, 2&1=1; индексиро­вание ([]).

9. Создать производный класс «БУЛЕВА МАТРИЦА» от класса «ЦЕЛОЧИСЛЕННАЯ МАТРИЦА».

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров целочисленный двумерный массив, содержащий матрицу, и ее размеры *n* и *m*; конструктор копирования; деструктор; метод возвращает число единиц в матрице; метод, возвращающий α-каноническую матрицу (в исходной матрице удалены повторяющиеся строки; строки составлены в порядке возрастания неотрицательных чисел, в качестве двоичных кодов которых рассматриваются данные строки).

Переопределить операции: поэлементная конъюнкция двух матриц (&); поэлементная дизъюнкция двух матриц (|); поэлементная операция ИСКЛЮЧАЮЩЕЕ ИЛИ двух матриц (^); произведение двух матриц A=[aij] и B=[ bjk], где i=, j=, k= (\*). При вычислениях операция целочисленного умноже­ния заменяется конъюнкцией, а сложение – дизъюнкцией; присваивание (=).

10. Создать производный класс «МИНОР» для базового класса «МАТРИЦА» размерности *n×m*. Переопределить для производного класса операции и методы (см. лаб. работу № 2, за­дание № 4).

11. Расширить возможности стандартного класса Time, чтобы можно было выводить время дня: утро, вечер и т. д.

12. Расширить возможности стандартного класса Date, чтобы можно было выводить время года: зима, лето и т. д.

13. Расширить возможности класса Annotation, чтобы можно было выводить время и дату изменения аннотации.

14. Расширить возможности класса Dictionary, чтобы можно было выводить дату последнего изменения в словаре.

15. Расширить возможности класса File, чтобы можно было выводить время и дату создания файла.

16. Расширить возможности класса Stack, чтобы можно было выводить время последнего сеанса работы со стеком.

17. Определить базовый класс для работы с прямоугольными матрицами, предусмотрев ввод-вывод матриц и выполнение следующих операций: сложение матриц; умножение матрицы на скаляр; перестановка строк матрицы по заданному вектору транспозиции; перестановка столбцов матрицы по заданному вектору транспозиции. В производном классе реализовать указанные операции для квадратных матриц, добавив выполнение следующих операций:транспонирование матрицы; умножение матриц.

Тесты

##### 1. Какая из записей является правильной записью абстрактного класса?

Варианты ответа:

1) abstract class A{virtual f()=0;};

\*2) class A{virtual f()=0;};

3) class A{virtual f();};

##### 2. Абстрактный класс – это класс, в котором:

1) есть хотя бы один виртуальный метод;

2) есть виртуальный конструктор;

3) есть виртуальный деструктор;

\*4) есть чисто виртуальный метод.

##### 3. Основная проблема множественного наследования состоит в:

1) замедлении выполнения программ;

2) необходимости явно указывать, к какому базовому классу принадлежит метод;

\*3) возможности существования двух экземпляров объекта базового класса;

4) неэкономном расходовании памяти.

4. Если записано

class A{public:virtual void f(){cout<<1;}};

class B:public A {public:virtual void f(){cout<<2;}};

то что будет напечатано, если

B b;A &a=b; a.f(); ?

Варианты ответа:

\*1) 2; 2) 21; 3) 12; 4) 1; 5) ошибка.

5. Если записано

class A {public:void f(){cout<<1;}};

class B:public A{public: void f(){cout<<2;}};

то что будет напечатано, если

B b; b.f(); ?

Варианты ответа:

\*1) 2; 2) 21; 3) 12; 4) 1.

6. Если записано

class A{public:virtual void f(){cout<<1;}};

class B:public A{public:virtual void f(){cout<<2;}};

то что будет напечатано, если

B b; b.f(); ?

Варианты ответа:

\*1) 2; 2) 21; 3) 12; 4)1; 5)ошибка.