# Лабораторная работа №4. Наследование и полиморфизм. Виртуальные функции

**1 Цель работы**

Цель работы – ознакомиться с основными механизмами простого и множественного наследования классов, изучить понятия виртуальных функций и абстрактных классов.

**2 Порядок выполнения работы**

* ознакомиться с описанием лабораторной работы;
* получить задание у преподавателя по вариантам;
* разработать и отладить программу;
* составить и защитить отчет по лабораторной работе у преподавателя.

**3 Содержание отчета**

* титульный лист;
* краткое теоретическое описание;
* задание на лабораторную работу, включающее математическую формулировку задачи;
* результаты выполнения работы, включающие схему алгоритма, тексты программ, результаты вычислений;

**4 Краткая теория**

**4.1 Простое наследование**

Объекты разных классов и сами классы могут находиться в отношении наследования, при котором формируется иерархия объектов, соответствующая заранее предусмотренной иерархии классов.

Иерархия классов позволяет определять новые классы на основе уже имеющихся. Имеющиеся классы обычно называют *базовыми* (иногда порождающими), а новые классы, формируемые на основе базовых, - *производными* (порожденными), иногда классами-потомками или наследниками. Производные классы «получают наследство» – данные и методы своих базовых классов – и, кроме того, могут пополняться собственными компонентами (данными и собственными методами). Наследуемые компоненты не перемещаются в производный класс, а остаются в базовых классах. Сообщение, обработку которого не могут выполнить производного класса, автоматически передается в базовый класс. Если для обработки сообщения нужны данные, отсутствующие в производном классе, то они ищутся автоматически и незаметно для программиста в базовом классе (см. рис. 4.1). Допускается множественное наследование – возможность для некоторого класса наследовать компоненты нескольких никак не связанных между собой базовых классов.

Выходное сообщение

Входное сообщение

Класс А

(базовый)

Класс В

(производный)

1

2

Рисунок 4.1 – Схема обработки сообщений в иерархии объектов:

1 – обработка сообщения методами производного класса;

2 – обработка сообщения методами базового класса.

При описании нового класса, производного от какого-то одного или нескольких базовых классов, можно добавлять новые функции-элементы и данные-элементы, сохраняя при этом все компоненты родителей, а можно родительские компоненты переопределить или перегрузить. В производном классе доступны все открытые и защищенные элементы базового класса (прямого или косвенного предшественника). Закрытые компоненты базового класса в производном классе недоступны.

Синтаксис объявления производного класса следующий:

**class <имя класса>:<спецификатор доступа>**

**<список классов-родителей>**

Производный класс может наследоваться от базового класса как **public**, **protected** или **private**. Защищенное и закрытое наследования встречаются редко и каждое из них нужно использовать с большой осторожностью.

При порождении класса как **public** открытые элементы базового класса становятся открытыми элементами производного класса, а защищенные элементы базового класса становятся защищенными элементами производного класса. Закрытые элементы базового класса никогда не бывают доступны для производного класса.

При защищенном наследовании открытые и защищенные компоненты базового класса становятся защищенными элементами производного класса. При закрытом наследовании открытые и защищенные компоненты базового класса становятся закрытыми элементами производного класса. При закрытом и защищенном наследовании не справедливо отношение, что объект производного класса является объектом базового класса.

В целом доступ к компонентам базового класса из производного класса можно представить следующей таблицей (таблица 4.1).

*Таблица 4.1 – Статусы доступа при наследовании*

|  |  |  |  |
| --- | --- | --- | --- |
| Доступ в  базовом классе | Спецификатор доступа перед базовым классом | Доступ в производном классе | |
| **struct** | **class** |
| **public** | отсутствует | **public** | **private** |
| **protected** | отсутствует | **public** | **private** |
| **private** | отсутствует | недоступны | недоступны |
| **public** | **public** | **public** | **public** |
| **protected** | **public** | **protected** | **protected** |
| **private** | **public** | недоступны | недоступны |
| **public** | **protected** | **protected** | **protected** |
| **protected** | **protected** | **protected** | **protected** |
| **private** | **protected** | недоступны | недоступны |
| **public** | **private** | **private** | **private** |
| **protected** | **private** | **private** | **private** |
| **private** | **private** | недоступны | недоступны |

Если в классе-наследнике переопределить компонентную функцию (ввести новую функцию с тем же именем), то для объектов этого класса новая функция отменит родительскую. Если обращаться к объекту этого класса, то вызываться будет новая функция. Если все-таки нужно вызвать именно функцию базового класса, надо использовать операцию разрешения области действия (**::**).

Пусть, например, был создан класс фигур **Shape**:

**class Shape**

**{**

**public:**

**void Draw();**

**...**

**};**

и наследующий ему класс кругов **Circl**:

**class Circl:public Shape**

**{**

**public:**

**void Draw();**

**};**

в каждом из классов объявили метод рисования **Draw**, а затем в программе выполнили операторы

**Shape \*PQ1 = new Shape;**

**PQ1->Draw(); //вызов Draw класса Shape**

**Circle \*PQ2 = new Circl;**

**PQ2->Draw(); //вызов Draw класса Circl**

**PQ2->Shape::Draw(); //вызов Draw класса Shape**

**((Shape \*)PQ2)->Draw(); //вызов Draw класса Shape**

В комментариях к коду указано, функции **Draw** каких классов вызывают эти операторы. Как видно, если мы обращаемся через указатель к самому объекту типа **Circl**, то вызывается переопределенная в нем функция. Но если мы обращаемся к нему как к объекту базового класса (последний из приведенных операторов) или соответствующим образом используем операцию разрешения области, то вызывается функция базового класса.

Если бы в классе-наследнике **Circl** отсутствовала функция **Draw**, то все приведенные операторы вызывали бы функцию базового класса.

Таким образом, механизм наследования позволяет использовать функции базового класса или переопределять их.

**4.2 Виртуальные функции и абстрактные классы**

Теперь рассмотрим другую задачу. Пусть мы имеем несколько классов, наследующих **Shape**: **Circl** (круг), **Rectang** (прямоугольник), **Square** (квадрат) и т.п. Каждый из этих классов имеет свою функцию **Draw**, которая умеет рисовать соответствующую фигуру. Программисту необходимо работать с объектами этих фигур как с объектами базового класса **Shape**, не разбираясь в истинной природе каждого объекта. И при этом нужно, чтобы программа сама понимала, что это за объект и как его рисовать. Например, требуется создать массив указателей на объекты различных фигур:

**Shape \*ShapeArray[10];**

загрузить его указателями на объекты разных фигур:

**ShapeArray[0] = new Circl;**

**ShapeArray[1] = new Rectang;**

**ShapeArray[2] = new Square;**

**...**

и затем в цикле выполнять рисование этих фигур:

**for (int i=0;i<3;i++)**

**ShapeArray[i]->Draw();**

Рассмотренный ранее механизм наследования такую задачу решить не может. Поскольку ко всем объектам мы обращаемся через тип их базового класса **Shape**, то только функция этого класса и будет вызываться.

Поставленную задачу *полиморфизма* позволяют решить *виртуальные функции*. Они не связаны с другими функциями с тем же именем в классах-наследниках. Если в классах-наследниках эти функции переопределены, то при обращении к такой функции во время выполнения будет вызываться та из виртуальных функций с одинаковыми именами, которая соответствует классу объекта, указанного при вызове. Поэтому, если в базовом классе **Shape** объявить функцию **Draw** как виртуальную, задача будет решена и каждая фигура будет рисоваться своей функцией.

Синтаксически это оформляется следующим образом. В базовом классе **Shape** функция объявляется следующим образом:

**virtual void Draw();**

Если функция была однажды объявлена виртуальной, она остается виртуальной и во всех классах-наследниках. Таким образом, для решения задачи полиморфизма хватило одного спецификатора **virtual**. Правда, обычно предпочитают для большей ясности программы в классах-наследниках тоже вводить спецификатор **virtual**, чтобы была ясна суть этих функций для тех, кто будет строить наследников данного класса. Но с точки зрения языка С++ это не обязательно.

Иногда в базовом классе определяют *чистую виртуальную функцию* (абстрактную функцию). Это функция, для которой не указана реализация. Для того чтобы определить такую функцию, достаточно указать, что ее тело равно нулю:

**virtual void Draw()=0;**

В нашем примере именно так целесообразно объявить функцию **Draw** в базовом классе **Shape**, поскольку непонятно, как можно нарисовать просто абстрактную фигуру. Реализация (тело) для чистой виртуальной функции не пишется.

Класс, в котором имеется хоть одна чистая виртуальная функция, называется *абстрактным*. Для абстрактного класса невозможно создать объект, но можно – указатель или ссылку на него. Такие классы предназначены только для построения на их основе классов-наследников.

**4.3 Множественное наследование**

Класс называют *непосредственным (прямым) базовым классом*, если он входит в список базовых при определении класса. В то же время для производного класса могут существовать косвенные или непрямые предшественники, которые служат базовыми для классов, входящих в список базовых. Наличие нескольких прямых базовых классов называют *множественным наследованием*. Например:

**class x1 { ... };**

**class x2 { ... };**

**class x3 { ... };**

**class y1:public x1,public x2,public x3 { ... };**

При множественном наследовании никакой класс не может больше одного раза использоваться в качестве непосредственного базового. Однако класс может больше одного раза быть непрямым базовым классом. При этом может происходить так называемое дублирование класса, соответствующее включению в производный объект нескольких объектов базового класса. Чтобы устранить дублирование объектов непрямого базового класса при множественном наследовании, этот базовый класс объявляют виртуальным. Для этого в списке базовых классов перед именем класса необходимо поместить ключевое слово **virtual**.

**5 Примеры программ**

1. На основе базового класса «книга» (поля: название, год издания) создать производный класс «руководство программисту», который, помимо перечисленных, содержит данные о требуемом уровне программиста и языке программирования. Описать виртуальную функцию вывода на экран информации о книге.

**#include <iostream>**

**using namespace std;**

**//базовый класс "книга"**

**class book**

**{**

**char\* title; //название книги**

**int year; //год издания**

**public:**

**book(char\*,int); //конструтор**

**virtual void display();//виртуальная функция печати**

**};**

**//производный класс "руководство программиста"**

**class programmers\_guide : public book**

**{**

**int level; //уровень требуемых знаний**

**char\* progr\_language; //язык программирования**

**public:**

**programmers\_guide(char\*,int,int,char\*); //конструктор**

**void display(); //вывод на экран**

**};**

**//конструктор базового класса**

**book::book(char\* book\_title, int book\_year)**

**:title(book\_title), year(book\_year)**

**{ }**

**//конструктор производного класса**

**programmers\_guide::programmers\_guide(char\* book\_title,**

**int book\_year, int book\_level, char\* book\_language)**

**//вызов конструктора базового класса**

**:book(book\_title, book\_year),**

**level(book\_level), progr\_language(book\_language)**

**{ }**

**//функция печати для базового класса**

**void book::display()**

**{**

**cout<<title<<", "<<year<<endl;**

**}**

**//функция печати для производного класса**

**void programmers\_guide::display()**

**{**

**book::display(); //вызов функции базового класса**

**cout<<"Level: "<<level<<"\nLanguage - "**

**<<progr\_language<<endl;**

**}**

**void main()**

**{**

**book b("Oxford Advanced Learner's Dictionary", 2006),**

**\*pointer;**

**programmers\_guide pg("C++ Programmer's Guide", 2007,**

**3,"C++");**

**pointer=&b; //указатель на объект базового класса**

**pointer->display(); //функция базового класса**

**pointer=&pg; //указатель на объект производного класса**

**pointer->display(); //функция производного класса**

**}**

Результат выполнения программы:

**Oxford Advanced Learner's Dictionary, 2006**

**C++ Programmer's Guide, 2007**

**Уровень: 3**

**Язык программирования – С++**

2. Создать абстрактный базовый класс «число» с виртуальной функцией вычисления модуля. Определить производные классы: действительное число, комплексное число со своими функциями модуля. Для проверки определить массив указателей на абстрактный класс, элементами которого являются указатели на объекты производных классов.

**#include <iostream>**

**#include <math.h>**

**using namespace std;**

**//базовый абстрактный класс "число"**

**class number**

**{**

**protected:**

**float real;**

**public:**

**number(float re) //конструктор**

**{**

**real=re;**

**}**

**//чистая виртуальная функция вычисления модуля**

**virtual float module() = 0;**

**};**

**//производный класс "действительное число"**

**class real\_numb : public number**

**{**

**public:**

**real\_numb(float re = 0.0) : number(re)**

**{ }**

**float module() //вычисление модуля**

**{**

**return real >= 0 ? real : -real;**

**}**

**};**

**//производный класс "комплексное число"**

**class complex\_numb : public number**

**{**

**float image; //мнимая часть**

**public:**

**complex\_numb(float re = 0.0, float im = 0.0) : number(re)**

**{**

**image=im;**

**}**

**float module() //вычисление модуля**

**{**

**return sqrt(pow(real,2) + pow(image,2));**

**}**

**};**

**void main()**

**{**

**//массив указателей на абстрактный базовый класс**

**number \*NumberArray[5];**

**//элементы массива - объекты производных классов**

**NumberArray[0] = new real\_numb(1.2);**

**NumberArray[1] = new complex\_numb(-3.0,4.0);**

**NumberArray[2] = new real\_numb;**

**NumberArray[3] = new complex\_numb;**

**NumberArray[4] = new complex\_numb(1.0);**

**//вызов функции модуля для каждого элемента массива**

**for (int i = 0; i < 5; i++)**

**cout<<"Module of number N"<<i+1<<" = "**

**<<NumberArray[i]->module()<<endl;**

**}**

Результат работы программы:

**Module of number N1 = 1.2**

**Module of number N2 = 5**

**Module of number N3 = 0**

**Module of number N4 = 0**

**Module of number N5 = 1**

**6 Контрольные вопросы**

1. В чем заключается механизм простого наследования? Как описать производный класс?
2. Как соблюдается принцип инкапсуляции (сокрытия данных) при простом наследовании?
3. Можно ли в классе-наследнике переопределять компонентные функции базового класса? Как это сделать?
4. Что такое полиморфизм? В каких случаях он применяется?
5. Нужно ли объявлять функцию виртуальной в производном классе, если она объявлена таковой в базовом классе?
6. Дайте определение чистой виртуальной функции.
7. В каких случаях целесообразно создание абстрактных классов?
8. Можно ли создать объект абстрактного класса? А указатель на него?
9. Опишите основные принципы множественного наследования.
10. В каких случаях базовый класс при множественном наследовании описывается как виртуальный?

**7 Варианты заданий для самостоятельного решения**

Заданы названия базовых и производных классов. Необходимо разработать поля и методы, наследуемые из базового класса, и собственные компоненты производных классов. Базовый класс может быть абстрактным. Наследование может быть множественным.

1-2. Первый базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Во втором базовом классе описать страны-производители. Производные классы – автомобиль (1 вариант), самолет (2 вариант); производные второго поколения – спортивный автомобиль, грузовой автомобиль (1 вариант), военный самолет, транспортный самолет (2 вариант).

3-4. Первый базовый класс – млекопитающие; поля – способ питания, вес, среда обитания. Во втором базовом классе описываются географические регионы. Производные классы – хищники (3 вариант) и травоядные (4 вариант).

5-6. Первый базовый класс – личность; поля – фамилия, пол, адрес. Во втором базовом классе задается структура университета – факультеты, кафедры, службы. Производные классы – студенты (5 вариант) и сотрудники университета (6 вариант).

7-8. Первый базовый класс – библиотека. Второй базовый класс – система УДК или ключевых слов. Производные классы – техническая (7 вариант) и художественная (8 вариант) литература.

9-10. Первый базовый класс – документ предприятия. Во втором базовом классе описываются корреспонденты предприятия. Производные классы – приказы (9 вариант), письма (10 вариант).

11-12. Первый базовый класс – точка. Производные классы – геометрические фигуры (11 вариант) и текст, который вписывается в фигуру (12 вариант).

13-14. Первый базовый класс – магазин; поля – номер, телефон, адрес, владелец, номер лицензии. Во втором базовом классе описываются поставщики товаров. Производные классы – отделы (13 вариант), продавцы (14 вариант).

15. Первый базовый класс – товар; поля – наименование, количество, производитель, цена за единицу изделия, общая цена. Во втором базовом классе описываются качественные характеристике товара. Производный класс – товар на складе. Создать массив указателей на класс. Вывести все данные о товарах, отсортированные по возрастанию цены за единицу изделия.

16. Создать абстрактный базовый класс с виртуальной функцией – площадь. Создать производные классы: прямоугольник, круг, прямоугольный треугольник, трапеция со своими функциями площади. Для проверки определить массив ссылок на абстрактный класс, которым присваиваются адреса различных объектов. Площадь трапеции: .

17. Создать абстрактный класс с виртуальной функцией – норма. Создать производные классы: комплексные числа, вектор из 10 элементов, матрица 2х2. Определить функцию нормы: для комплексных чисел – модуль в квадрате, для вектора – корень квадратный из суммы элементов по модулю, для матрицы – максимальное значение по модулю.

18. Создать абстрактный класс «кривые» вычисления координаты *y* для некоторого *х*. Создать производные классы: прямая, эллипс, гипербола со своими функциями вычисления *y* в зависимости от входного параметра *x*. Уравнение прямой: , эллипса: , гиперболы: 

19. Создать абстрактный базовый класс с виртуальной функцией – сумма прогрессии. Создать производные классы: арифметическая прогрессия и геометрическая прогрессия. Каждый класс имеет два поля типа **double**. Первое – первый член прогрессии, второе – постоянная разность (для арифметической) и постоянное отношение (для геометрической). Определить функцию вычисления суммы, где параметром является количество элементов прогрессии.

Арифметическая прогрессия 

Сумма арифметической прогрессии: 

Геометрическая прогрессия: 

Сумма геометрической прогрессии: 

20. Создать базовый класс «список» с виртуальными функциями вставки и извлечения. Реализовать на базе списка производные классы стека и очереди.

21. Создать базовый класс – фигура, и производные: круг, прямоугольник, трапеция. Определить виртуальные функции: площадь, периметр и вывод на печать.

22. Создать базовый класс – работник, и производные классы – служащий с почасовой оплатой, служащий в штате и служащий с процентной ставкой. Определить функцию начисления зарплаты.

23. Описать абстрактный базовый класс с виртуальной функцией – площадь поверхности. Создать производные классы: параллелепипед, тетраэдр, шар со своими функциями площади поверхности. Для проверки определить массив ссылок на абстрактный класс, которым присваиваются адреса различных объектов. Площадь поверхности параллелепипеда: . Площадь поверхности шара: . Площадь поверхности тетраэдра: 

24. Создать класс «человек», производные от которого мужчины и женщины. Определить виртуальную функцию реакции человека на вновь увиденного другого человека.

25. Создать абстрактный базовый класс с виртуальной функцией – объем. Создать производные классы: параллелепипед, пирамида, тетраэдр, шар со своими функциями объема. Для проверки определить массив указателей на абстрактный класс, которым присваиваются значения указателей на объекты производных классов. Объем параллелепипеда:  (*x, y, z* – стороны пирамиды). Объем пирамиды:  (*x, y* – стороны, *h* – высота). Объем тетраэдра: . Объем шара: 

26. Создать абстрактный класс – млекопитающие. Определить производные классы – животные и люди. У животных определить производные классы собак и коров. Определить виртуальные функции описания человека, собаки и коровы.

27. Создать базовый класс – предок, у которого есть фамилия. Определить виртуальную функцию печати. Создать производный класс – ребенок, у которого функция печати дополнительно выводит имя. Создать производный класс от последнего класса – внук, у которого есть отчество. Написать свою функцию печати.

28. Создать абстрактный базовый класс с виртуальной функцией – корни уравнения. Создать производные классы: класс линейных уравнений и класс квадратных уравнений. Определить функцию вычисления корней уравнений.