# Лабораторная работа №4

**Тема работы:** принцип полиморфизма, виртуальные функции, абстрактные классы.

**Цель работы:** изучить реализацию принципа полиморфизма через использование виртуальных функций при наследовании.

**Теоретические сведения:**

**Виртуальные функции.** В языке С++ полиморфизм реализуется посредством виртуальных функций. Виртуальная функция – это функция, объявленнаяс ключевым словом virtual в базовом классе и переопределенная в одном или нескольких производных этого класса. Объявление

virtual void print(void);

говорит о том, что функция print может быть различной для базового и разных производных классов. В производных классах функция может иметь список параметров, отличный от параметров виртуальной функции базового класса. В этом случае эта функция будет не виртуальной, а перегруженной. Механизм вызова виртуальных функций можно пояснить следующим образом. При создании нового объекта для него выделяется память. Для виртуальных функций (и только для них) создается указатель на таблицу функций, из которой выбирается требуемая функция в процессе выполнения. Если в некотором классе задана хотя бы одна виртуальная функция, то все объекты этого класса содержат указатель на связанную с их классом виртуальную таблицу. Эта таблица содержит адреса (указатели на первые инструкции) действительных функций, которые будут вызваны. Доступ к виртуальной функции осуществляется через этот указатель и соответствующую таблицу (т. е. осуществляется косвенный вызов функции). Если функция вызвана с использованием ее полного имени, то виртуальный механизм игнорируется. Свойство виртуальности проявляется только тогда, когда обращение к функции идет через указатель или ссылку на объект. Указатель или ссылка могут указывать как на объект базового, так и на объект производного класса.

Рассмотрим пример использования виртуальной функции

#include<iostream>

#include<iomanip>

#include<string.h>

using namespace std;

class base // базовый класс

{

public:

virtual char \* name(void)

{

return "noname";

}

virtual double area(void)

{ return 0; }

};

class rect: public base // производный класс «Прямоугольник»

{

int h,s; // размеры прямоугольника

public:

rect(int H, int S) // конструктор

{

h=H;

s=S;

}

char \* name(void); // вывод на экран названия фигуры

double area(void); // вывод на экран площади фигуры

};

char \* rect::name(void) // вывод на экран названия фигуры

{

return "прямоугольник";

}

double rect:: area(void) // вывод на экран площади фигуры

{

return h\*s;

}

class circl: public base // производный класс «Окружность»

{

int r; // радиус окружности

public:

circl(int R) // конструктор

{ r=R; }

char \* name(void); // вывод на экран названия фигуры

double area(void); // вывод на экран площади фигуры

};

char \* circl::name(void) // вывод на экран названия фигуры

{

return "круг";

}

double circl::area(void) // вывод на экран площади фигуры

{

return 3.14\*r\*r;

}

int main()

{

setlocale(LC\_ALL,"Russian");

base \*p[2]; // массив указателей на базовый класс

rect obj1(3,4);

circl obj2(5);

p[0]=&obj1;

p[1]=&obj2;

for(int i=0; i < 2; i++)

cout << "площадь " << p[i]->name() << setw(10) << p[i]->area() << endl;

return 0;

}

Массив указателей p хранит адреса объектов производных классов и необходим для вызова виртуальных функций этих классов. Если функции name() и area() в базовом классе объявлены как virtual и мы вызываем эти функции через указатель базового класса, указывающий на объекты производных классов, то программа будет динамически (т. е. во время выполнения программы) выбирать соответствующие функции name() и area() производного класса. Это называется динамическим связыванием (dynamic binding). Когда виртуальная функция вызывается путем обращения к заданному объекту по имени и при этом используется операция доступа к элементу «точка», тогда эта ссылка обрабатывается во время компиляции и это называется статическим связыванием.

Если функция была объявлена как виртуальная в некотором классе (базовом классе), то она остается виртуальной независимо от количества уровней в иерархии классов, через которые она прошла.

Приведем основные правила использования виртуальных функций:

– виртуальный механизм поддерживает полиморфизм на этапе выполнения программы. Это значит, что требуемая версия программы выбирается на этапе выполнения программы, а не компиляции;

– класс, содержащий хотя бы одну виртуальную функцию, называется полиморфным;

– виртуальные функции можно объявлять только в классах (class) и структурах (struct);

– виртуальными функциями могут быть только нестатические функции (без спецификатора static), т. к. характеристика virtual наследуется. Функция производного класса автоматически становится virtual;

– виртуальные функции можно объявлять со спецификатором friend для другого класса;

– виртуальными функциями могут быть только не глобальные функции (т. е. компоненты класса);

– если виртуальная функция объявлена в производном классе со спецификатором virtual, то можно рассматривать новые версии этой функции в классах, наследуемых из этого производного класса;

– для вызова виртуальной функции требуется больше времени, чем для не виртуальной. При этом также требуется дополнительная память для хранения виртуальной таблицы;

– при использовании полного имени при вызове виртуальной функции виртуальный механизм не поддерживается.

**Абстрактные классы.** Базовый класс обычно содержит ряд виртуальных функций, которые часто фиктивны и имеют пустое тело. Эти функции существуют как некоторая абстракция, конкретное значение им придается в производных классах. Такие функции называются чисто виртуальными (pure virtual function), т. е. такими, тело которых не определено. Общая форма записи чисто виртуальной функции имеет вид

**virtual прототип функции = 0;**

Если класс является производным класса с чисто виртуальной функцией и эта функция в нем не описана, тогда функция остается чисто виртуальной и в этом производном классе. Следовательно, такой производный класс является абстрактным. Хотя иерархия классов не требует обязательного включения в нее каких-либо абстрактных классов, однако программы, использующие объектно-ориентированное программирование, все же имеют иерархию, порожденную абстрактным базовым классом. Абстрактные классы могут составлять несколько уровней иерархии. В качестве примера можно привести иерархию форм (рис. 5).



Рис. 5. Иерархия наследования классов

Иерархия может порождаться абстрактным базовым классом *Shape.* На уровень ниже можно получить два абстрактных класса *TwoDShape* и *ThreeDShape*. При переходе еще на один уровень ниже можно определить конкретные классы для двухмерных (Circle, Rectangle) и трехмерных (Cube, Sphere) форм. Согласно языку UML имя абстрактного класса пишется курсивом.

**Контрольные вопросы**

1. Какая функция называется виртуальной?

2. Чем виртуальные функции отличаются от перегружаемых?

3. Какой класс называется абстрактным?

4. В чем состоит различие раннего и позднего связывания?

5. Опишите назначение виртуального деструктора.

**Порядок выполнения работы**

1. Изучить краткие теоретические сведения.
2. Ознакомиться с материалами литературных источников.
3. Ответить на контрольные вопросы.
4. Разработать алгоритм программы.
5. Написать, отладить и выполнить программу.

**Варианты заданий**

1. Реализовать абстрактный класс *Shape*, содержащий интерфейс иерархии. Создать производные абстрактные классы *TwoDShape* и *ThreeDShape*, от которых унаследовать всевозможные конкретные формы. Реализовать виртуальные функции print (для вывода типа и размера объектов каждого класса), area, draw и volume.
2. Разработать программу, для обработки информации о строительных материалах: поступление, учет и отгрузка. Для этого требуется разработать классы. Базовый класс должен быть абстрактным. Первый производный класс определяет количество стройматериалов каждого типа, например 50 перекрытий. Другой класс хранит данные о каждом виде стройматериалов, например площадь и стоимость за квадратный метр. Далее следует класс, хранящий описание каждого вида стройматериалов, которое включает название и сорт материала.
3. Разработать иерархию наследования, включающую следующие классы: город (наименование), магазин (наименование, тип), товар (наименование, сорт, количество, цена), банк (номер счета, сумма денег на счете) и покупатель (фамилия, сумма денег, сумма покупки). При этом класс «Город» – абстрактный базовый для классов «Магазин» и «Банк», класс «Товар» – для класса «Магазин», а классы «Магазин» и «Банк» – базовые для класса «Покупатель». Класс «Покупатель» содержит также методы выполнения различных операций с товаром.
4. Написать программу учета наличия транспортных средств (автобусы) в автопарке. Разработать следующие классы: абстрактный базовый класс «Автобус» содержащий поля с информацией о марке автобуса и госномере; производные классы «Мягкий автобус», «Жесткий автобус». Производные классы содержат следующие поля: количество мест, водители. По запросу выдавать информацию о свободных местах в автобусах, об автобусах в рейсе.
5. Создать базовый абстрактный класс «Книга», включающий название книги, фамилию автора. Реализовать производный класс «Отдел», включающий в себя название отдела. Написать программу, позволяющую добавлять и удалять книги из отдела.
6. Разработать иерархию наследования, включающую следующие классы: абстрактный базовый класс «Товар» (название товара, производитель, цена), производные классы: «Отдел» (название отдела), «Магазин» (наименование, тип), «Корзина» (сумма покупки). Класс «Банк» (номер счета, сумма денег на счете) не наследуется. Все классы должны содержать функции получения и изменения всех полей. Написать программу, позволяющую производить покупки.
7. Создать абстрактный базовый класс *Person*, описывающий обычного человека. Создайте производный класс *Student*, описывающий типичного студента. От класса *Student* наследуйте класс *GradStudent*, описывающий типичного аспиранта. Все классы должны содержать функции получения и изменения всех полей. Написать программу, позволяющую получать сведения о студентах и аспирантах.
8. Создать абстрактный базовый класс «Часы», а также производные классы «Механические часы» и «Электронные часы». Все классы должны содержать функции получения и изменения всех полей. Написать программу, позволяющую получать сведения о часах.
9. Создать абстрактный базовый класс «Студент», а также производные классы «Выпускник» и «Старшекурсник». Все классы должны содержать функции получения и изменения всех полей. Написать программу, позволяющую получать сведения о студентах.
10. Создать абстрактный базовый класс «Служащий», содержащий фамилию и имя служащего. Производный класс «Работники с почасовой оплатой» содержит следующие поля: оплата за час и часы, отработанные за неделю. Производный класс «Работники со сдельной оплатой» содержит следующие поля: оплата единицы продукции и число единиц продукции за неделю. Все классы должны содержать функции получения и изменения всех полей. Написать программу, позволяющую получать сведения о служащих.
11. Написать программу учета книг. Создать абстрактный базовый класс «Книга», содержащий следующие поля: название книги, фамилия автора, издательство. Получить производные классы:

* «Отдел технической книги», который содержит поля о названии отдела и названии отраслей техники;
* «Отдел художественной книги», который содержит поля о названии отдела и названии направлений в литературе;
* «Абонент», который наследуется от классов «Отдел технической книги» и «Отдел художественной книги». Класс «Абонент» включает следующие поля: ФИО студента, факультет, группа, названия книги, имеющейся у студента, и время ее возврата.

Все классы должны содержать функции получения и изменения всех полей. Программа должна выдавать сведения о студентах, просрочивших время возврата книг.