# Лабораторная работа №3

**Тема работы:** наследование, простое наследование.

**Цель работы:** изучить принципы наследования.

**Теоретические сведения:**

**Наследование** – это механизм получения нового класса из существующего. Существующий класс может быть дополнен или изменен для создания производного класса. При создании нового класса вместо написания полностью новых данных и функций программист может указать, что новый класс должен наследовать данные и функции ранее определенного базового класса. Этот новый класс называется производным классом. Каждый производный класс сам является кандидатом на роль базового класса для будущих производных классов. При простом наследовании класс порождается одним базовым классом. При множественном наследовании производный класс наследуется несколькими базовыми классами. Производный класс обычно добавляет свои данные и функции, так что производный класс в общем случае больше своего базового.

Механизм наследования помогает сделать разработку более экономной и читаемой. Совокупность классов можно описать в виде такой иерархической структуры, что, если класс B наследует структуру и поведение класса A, то класс A называется базовым, а класс B – производным (рис. 3).



Рис. 3. Диаграмма наследования классов

Различают простое наследование и множественное. В первом случае производный класс имеет один базовый класс (рис. 4).



Рис. 4. Простое наследование классов

Шаблон объявления производного класса можно представить следующим образом:

ключ\_класса имя\_производного\_класса :

необязательный\_модификатор\_доступа имя\_базового\_класса

{

Тело производного класса

};

Пример объявления производного класса.

class Location // базовый класс

{

int x,y;

public:

. . .

};

class Point:public Location // производный класс

{

Тело класса Point

};

Класс Location является базовым и наследуется с атрибутом public. Класс Point – производный класс. Двоеточие (:) отделяет производный класс от базового. Атрибут класса (модификатор прав доступа) может задаваться ключевыми словами public и private. Атрибут может опускаться – в этом случае принимается атрибут по умолчанию (для ключевого слова class – private, для struct – public). Объединение (union) не может быть базовым или производным классом.

Модификатор прав доступа используется для изменения прав доступа к наследуемым элементам класса в соответствии с правилами, представленными в табл. 1.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Ограничения на  доступ в базовом классе | Модификатор  наследования прав | Ограничения на доступ  в производном классе |
| private | private | Нет доступа |
| protected | private | private |
| public | private | private |
| private | public | Нет доступа |
| protected | public | protected |
| public | public | public |

Отметим, что в производных классах права на доступ к элементам базовых классов не могут быть расширены, а только более ограничены.

Пример использования прав доступа к элементам базового класса.

#include<iostream>

using namespace std;

#include<string.h>

class A // базовый класс

{

int a1;

public:

int a2;

void f1();

A(int i=0, int j=0)

{

a1=i;

a2=j;

}

};

class B:public A // производный класс

{

int b1;

public:

void f1()

{

//a1=1; // ошибка, a1 – private-переменная класса А и доступна

// только для методов и дружественных функций

// собственного класса

b1=0; // доступ к переменной типа private из метода класса

a2=1; // a2 унаследована из класса А с атрибутом доступа private

// и поэтому доступна в методе класса

}

B(int i=0, int j=0, int k=0):A(i,j)

{

b1=k;

}

};

int main()

{

A a\_ob1; // объявление объекта a\_ob1 класса А

B b\_ob1; // объявление объекта b\_ob1 класса В

// b\_ob1.a2+=1; // ошибка, т. к. a2 private

a\_ob1.a2+=1; // допустимая операция

return 0;

}

Рассмотрим еще пример, демонстрирующий наследование прав доступа к элементам базовых классов.

#include<iostream>

using namespace std;

#include<string.h>

#define N 10

class book // базовый класс

{

protected:

char naz[20]; // название книги

int kl; // количество страниц

public:

book(char \*, int); // конструктор класса book

~book(); // деструктор класса book

};

class avt: public book // производный класс

{

char fm[10]; // фамилия автора

public:

avt(char \*, int, char \*); // конструктор класса avt

~avt(); // деструктор класса avt

void see();

};

enum razd{teh, hyd,uch}; // перечисление

class rzd: public book

{

razd rz;

public:

rzd(char \*, int, razd); // конструктор класса rzd

~rzd(); // деструктор класса rzd

void see();

};

book::book(char \*s1, int i)

{

cout << "\n Конструктор класса book" << endl;

strcpy(naz,s1);

kl=i;

}

book::~book()

{

cout << "\nДеструктор класса book" << endl;

}

avt::avt(char \*s1, int i, char\* s2):book(s1,i)

{

cout << "\n Конструктор класса avt" << endl;

strcpy(fm,s2);

}

avt::~avt()

{

cout << "\nДеструктор класса avt" << endl;

}

void avt::see()

{

cout << "\nНазвание книги " << naz << endl << "\nКоличество страниц " <<

kl <<endl;

}

rzd::rzd(char \*s1, int i, razd tp):book(s1,i)

{

cout << "\n Конструктор класса rzd" << endl;

rz=tp;

}

rzd::~rzd()

{

cout << "Деструктор класса rzd" << endl;

}

void rzd::see()

{

switch(rz)

{

case teh: cout << "\n Раздел технической литературы" << endl; break;

case hyd: cout << "\nРаздел художественной литературы" << endl;

break;

case uch: cout << "\n Раздел учебной литературы" << endl; break;

}

}

int main()

{

setlocale(LC\_ALL,"Russian");

avt av("Книга 1", 123, "автор 1");

rzd rz("Книга 1", 123, teh);

av.see();

rz.see();

return 0;

}

Если базовый класс имеет конструктор с одним или более аргументами, то любой производный класс должен иметь конструктор. В нашем примере конструктор класса book задан в виде

book::book(char \*s1, int i)

{

cout << "\nКонструктор класса book";

strcpy(naz,s1);

kl=i;

}

Теперь объявление объекта в функции main (либо в другой функции) может осуществляться

book my\_ob("Дейтел", 1113);

В соответствии со сделанными выше замечаниями производный класс avt тоже должен иметь конструктор. В нашем примере он задан следующим образом:

avt::avt(char \*s1, int i, char s2):book(s1,i)

{

cout << "\n Конструктор класса avt";

strcpy(fm,s2);

}

**Конструкторы и деструкторы производных классов.** Если у базового и производного классов имеются конструкторы и деструкторы, то конструкторы выполняются в порядке наследования, а деструкторы − в обратном порядке. Общий синтаксис конструктора производного класса следующий:

конструктор\_производного\_класса(арг) : base(арг)

{

тело конструктора производного класса

}

**Контрольные вопросы**

1. Опишите модификаторы доступа и наследования. Как изменяются атрибуты доступа элементов класса при наследовании?
2. Как работают конструкторы при наследовании?
3. Как работают деструкторы при наследовании?
4. Какой класс называется производным?
5. Как объявляются производные классы?

**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. При наличии классов Circle (круг), Square (квадрат) и Triangle (треугольник), производных от Shape (фигура), определить функцию intersect() (пересечение), которая принимает две фигуры Shape\* в качестве аргументов и вызывает подходящие функции для определения того, пересекаются ли эти фигуры. Для решения этой задачи вам придется добавить подходящие (виртуальные) функции к классам. Не пишите код, проверяющий пересечение; просто убедитесь, что вызываются правильные функции. Такой подход часто называют двойной диспетчеризацией или мульти-методом.
2. Реализовать класс «Человек», включающий в себя фамилию, имя, отчество, год рождения и методы, позволяющие изменять/получать значения этих полей.

Реализовать следующие производные классы:

* «Преподаватель университета» с полями: должность, ученая степень, специальность, список научных трудов (массив строк);
* «Член комиссии» с полями: название комиссии, год назначения в комиссию, номер свидетельства, автобиография (массив строк);
* «Преподаватели – члены комиссии» (производный от первого и второго класса). Дополнительное поле – список работ, выполненных в комиссии.

Классы должны содержать методы доступа и изменения всех полей.

1. Реализовать класс «Человек», включающий в себя имя, фамилию, отчество, год рождения и методы, позволяющие изменять/получать значения этих полей.

Реализовать следующие производные классы:

* «Предприниматель» с полями: номер лицензии, адрес регистрации, УНН, данные о налоговых платежах (массив пар вида «дата, сумма»);
* «Турист» - содержит данные паспорта (строка), данные о пересечении границы в виде массива пар «дата, страна»;
* «Челнок» (производный от второго и третьего класса) – добавляется массив строк списка адресов, по которым покупается товар.

Классы должны содержать методы доступа и изменения всех полей.

1. Разработать базовый класс Fakultet, включающий в себя название факультета. Реализовать производный класс Student, включающий следующие компоненты данных: Ф.И.О. студента, год рождения, результаты сдачи последней сессии. Классы должны содержать методы доступа и изменения всех полей. Написать программу, которая выдает информацию об успеваемости студентов.
2. Разработать иерархию классов для нахождения корней квадратного уравнения ax2 + bx + c = 0. А – базовый для класса В, В – базовый для класса С. Компоненты классов А, В, С – коэффициенты a, b, c. Корни уравнения и методы решения находятся в классе С.

В следующих заданиях требуется создать базовый класс и определить общие методы show( ), get( ), set( ) и другие специфические методы для данного класса. Создать производные классы, в которых добавить специфические 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. Использовать объекты производных классов для моделирования реальных ситуаций и объектов.

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 базовый класс «Пасажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения пассажиров.
4. Создать базовый класс «Учащийся» и производные классы «Школьник» и «Студент». Создать массив объектов базового класса и заполнить этот массив объектами. Вывести информацию о студентах и школьниках.
5. Определить базовый класс для работы с матрицами, предусмотрев ввод, вывод матриц и выполнение следующих операций:

а) сложение матриц;

б) умножение матрицы на скаляр;

в) перестановка строк матрицы по заданному вектору транспозиции.

В производном классе реализовать указанные операции для квадратных матриц, добавив выполнение следующих операций:

а) транспонирование матрицы;

б) умножение матриц.

1. Расширить возможности стандартного класса Time, так чтобы была возможность выводить время дня: утро, вечер и т. д.
2. Расширить возможности стандартного класса Date, так чтобы была возможность выводить время года: зима, лето и т. д.
3. Написать программу для обслуживания клиентов магазина. Реализовать класс «Товар», «Отдел», «Корзина», «Банк». Каждый товар имеет характеристики (группа, тип, индивидуальные особенности, страна происхождения и т. д.) и штрих-код. При обслуживании клиента необходимо подготовить электронный чек, в котором должно быть указано название товара, его цена, количество, общая сумма покупки, дата и время покупки.
4. Написать программу, управляющую работой библиотеки. Создать классы: «Книга», «Отдел», «Библиотека». В классах реализовать следующие функции: добавление, удаление книг из отделов, выдача книг на абонемент. Классы должны содержать методы доступа и изменения всех полей.
5. Напишите иерархию наследования для классов Quadrilateral (четырехугольник), Trapezoid (трапеция), Parallelogram (параллелограмм), Rectangle (прямоугольник) и Square (квадрат). Используйте Quadrilateral как базовый класс иерархии. Сделайте иерархию настолько глубокой (т.е. настолько многоуровневой), насколько это возможно. Закрытыми данными класса Quadrilateral должны быть пары координат (x,y) четырех угловых точек Quadrilateral. Напишите программу драйвер, который создает объекты каждого из этих классов.