

Практическое занятие

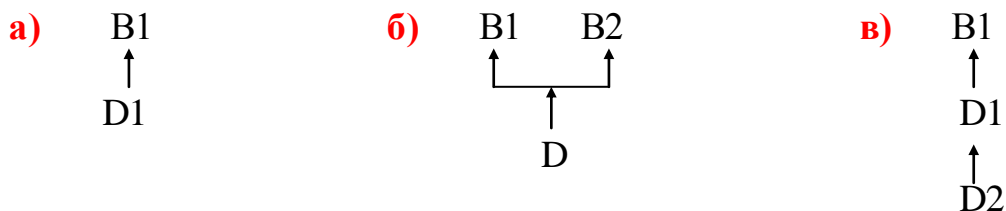
Тема: Проектирование классов с наследованием

Цель: получение практических навыков проектирования классов и исследование механизмов открытого и закрытого наследования в C++.

Наследование

Наследование – это процесс, посредством которого один объект может приобретать свойства другого. Так как объекты принадлежат к классам, то между классами в этом случае устанавливаются определенные взаимоотношения. Класс, который делегирует свойства, называется базовым (base), а который наследует свойства – производным (derived). У одного базового класса может быть несколько производных, и наоборот, один производный класс может наследовать характеристики нескольких базовых классов.

Схемные примеры наследования:



В примере (а) показана простейшая схема подчиненности с одним базовым (B1) и одним производным (D1) классом. В примере (б) показан один производный (D) класс от двух базовых (B1 и B2). На схеме (в) показано, что B1 есть базовый класс для D1, который в свою очередь является базовым для D2. В этом последнем случае B1 называется для D2 косвенно-базовым. Взаимосвязи классов в программе образуют **иерархию классов**.

При указании наследования соблюдаются следующие правила.

Базовый класс описывается обычным образом. Например:

```
class base
{
//содержание класса base
};
```

Для указания наследования после имени производного класса ставится двоеточие и указывается имя базового класса со **спецификатором доступа** (private, public). Для примера (а) описание будет выглядеть следующим образом:

```
class derived: public base
{
//содержание класса derived
};
```

Спецификатор выбирается в зависимости от того, как предполагается использовать члены класса base.

Для примера (б) описание производного класса будет выглядеть, например, так:

```
class derived: public B1, public B2 {  
    //содержание класса derived  
};
```

В примере (в) для производного класса D2 базовым будет класс D1, поэтому описание D2 соответствует схеме (а).

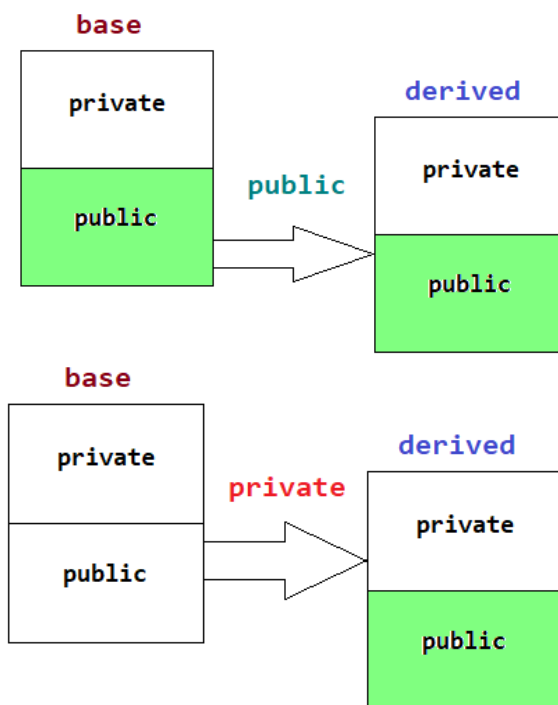
Взаимоотношения между членами классов и доступ к ним.

Базовый класс является первым в иерархии, он «не знает» сколько у него будет производных классов (или не будет вообще). Поэтому 1) он проектируется самодостаточным в плане доступа к закрытым членам и 2) члены производных классов в нем не упоминаются.

Как известно, **внутри** класса его члены открыты друг для друга: любая функция-член может обращаться к любой переменной-члену класса. Однако **извне** можно обратиться только к открытым членам класса.

Важно понимать, что при наследовании производный класс **полностью** включает в себя базовый класс. Однако не все члены классов открыты друг для друга в этом объединении. «Круг общения» членов производного класса расширяется только за счет **public**-членов базового класса. Это означает, что члены производного класса могут обращаться **напрямую** только к открытым членам класса **base**; **private**-члены базового класса можно использовать в описании функций производного класса только через открытые члены класса **base**. Это – общее правило, которое действует на этапе проектирования классов.

Когда создается объект производного класса, то **доступ** извне к его членам возможен только для открытой части конструкции, которая определяется спецификатором доступа. Если наследование производится со спецификатором доступа **public**, то открытые члены **base** становятся **public**-членами класса **derived**. Если наследование производится по типу **private**, то **public**-члены базового класса становятся **private**-членами класса **derived**. И в этом случае открытой частью конструкции будут только **public**-члены класса **derived**.



Пример.

```
#include <iostream>
using namespace std;
class base
{
    int x;
public:
    void set_x (int n) {x=n;}
    void show_x ( ) {cout<<x<<'\\n';}
};
class derived: public base
{
    int y;
public:
    void set_y (int k) {y=k;}
    void show_y ( ) {cout<<y<<'\\n';}
};
int main ( )
{
    derived obj;
    obj.set_x(10);
    obj.set_y(12);
    obj.show_x( );
    obj.show_y( );
    return 0;
}
```

В данном примере класс base наследуется как открытый. Поэтому в объекте obj, который является экземпляром производного класса, его public-функции доступны как обычные открытые члены. Но “добраться” напрямую до закрытого члена базового класса всё также невозможно; и инструкция типа

cout <<obj.x; // Неверно!

будет ошибкой.

Конструкторы и деструкторы в наследовании

Если у базового и производного классов имеются конструкторы и деструкторы, то конструктор базового класса выполняется раньше конструктора производного; для деструкторов соблюдается обратный порядок.

Если конструкторы имеют параметры, то:

а) все аргументы для конструктора базового и конструктора производного классов передаются конструктору производного класса;

б) затем, используя расширенную форму объявления конструктора производного класса, соответствующие аргументы передаются в базовый класс.

Пример.

```
#include<iostream>
using namespace std;
class base {
int i;
public:
    base(int n) {cout <<"Base constructor"; i=n;}
    ~base ( ) {cout<<"Base destructor";}
    void show_i( ) {cout <<i<<'\\t';}
};
class derived : public base {
int j;
public:
    derived(int n, int m): base(m)
    {cout<<"Derived constructor"; j=n;}
    ~derived( ) {cout<<"Derived destructor";}
    void show_j( ) {cout<<j<<'\\n';}
};
int main( )
{
    derived obj(10,20);
    obj.show_i( );
    obj.show_j( );
    return 0;
}
```

Оба класса в примере имеют конструкторы и деструкторы.

Вначале на экран выведется:

Base constructor,

затем - Derived constructor.

Конструктор производного класса объявлен так, что требует два аргумента; один он использует сам, другой передает в базовый класс. В результате инициализируются переменные **i** и **j**.

Затем на экран будет выведено:

20 10.

После этого выводится сообщение:

Derived destructor,

а потом- Base destructor

Дружественные функции в производных классах

Для производных классов также можно создавать дружественные функции с целью прямого доступа к закрытым членам классов. Рассмотрим особенности на примере.

```

#include <iostream>
using namespace std;

class two; //опережающее объявление производного класса
class base{
    int x;
    public:
    void setx(int a) {x=a;}
    int getx() { return x; }
};

class one: public base{
    int y;
    public:
    void sety(int a) {y=a;}
    int gety() { return y; }
    friend int minimum(one oob, two tob);
};

class two: public base{
    int z;
    public:
    void setz(int a) {z=a;}
    int getz() { return z; }
    friend int minimum(one oob, two tob);
};

//описание дружественной функции
int minimum(one oob, two tob){
    if(oob.y<tob.z)
        {cout<< "Min = "<<"one"<<endl;
         return oob.y;
        }
    else
        {cout<< "Min = "<<"two"<<endl;
         return tob.z;
        }
}

```

```

int main()
{
    one ob1;
    two ob2;
    ob1.setx(12); ob1.sety(15);
    ob2.setx(33); ob2.setz(37);
    cout<< minimum(ob1,ob2);
    return 0;
}

```

В данном примере дружественная функция **friend int minimum(one oob, two tob)** предназначена для работы с объектами двух производных классов, на что указывает список её параметров. Заявленная вначале в производном классе **one** дружественная функция использует имя второго производного класса **two**, который описан ниже в программе и поэтому не известен компилятору в данной точке программы. Для указания компилятору того, чем является имя **two**, в начале программы произведено опережающее объявление класса **two**: **class two;**

Внутри дружественной функции обращение к закрытым членам классов производится непосредственно, например **if(oob.y<tob.z).**

Вид экрана после выполнения программы:

```

Min = one
15
Process returned 0 (0x0)
Press any key to continue.

```

Генерация случайных чисел

Случайные числа в языке программирования C++ могут быть сгенерированы функцией **rand()** из стандартной библиотеки C++. Функция **rand()** генерирует числа в диапазоне от 0 до **RAND_MAX**. **RAND_MAX** - это константа, определённая в библиотеке **<cstdlib>**.

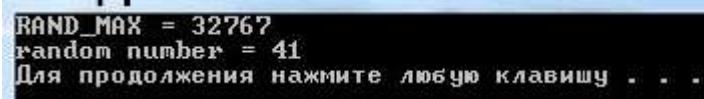
Реализация генератора случайных чисел:

```

#include <iostream>
using namespace std;
int main()
{
    // константа верхнего предела случайных чисел
    cout << "RAND_MAX = " << RAND_MAX << endl;
    // запуск генератора случайных чисел
    cout << "random number = " << rand() << endl;
    return 0;
}

```

Вид экрана:



```
RAND_MAX = 32767
random number = 41
Для продолжения нажмите любую клавишу . . .
```

Для генерации случайных чисел в ограниченном диапазоне используется инструкция:

```
random_number = firs_value + rand() % last_value;
```

где **firs_value** - минимальное число из желаемого диапазона, а **last_value** - ширина выборки.

Сборка проекта из нескольких файлов

Для удобства работы с программой её разбивают на несколько частей (файлов). Файлы различают:

- заголовочные, которые обозначаются с расширением «**.h**»;
- исходного кода, которые обозначаются с расширением «**.cpp**».

Главная функция помещается в файл с расширением «**.cpp**». К нему в тексте программы с помощью директивы **#include** подключаются остальные файлы, например,

```
#include "Classes.h".
```

Имя файла указывается в кавычках (а не в угловых скобках), чтобы не путать с библиотечными файлами.

Практикум

Вариант А:

Задание 1. Спроектировать структуру классов в соответствии с индивидуальным заданием (см. Приложение). Наследование осуществляется по типу **public**. Создать несколько объектов производных классов, задавая случайным образом их свойства. Определить *Вычисляемый показатель*.

Задание 2. Переработать проект, выполненный для *Задания 1*. Для каждого производного класса динамически выделить память для 5-элементного массива объектов. Для определения *Вычисляемого показателя* спроектировать дружественную функцию для работы с объектами производных классов. Проект разместить в нескольких файлах.

Задание 3. В проекте, разработанном для *Задания 2*, изменить спецификатор наследования одного из классов с **public** на **private**.

Вариант Б:

Задания 1 и 2 из Варианта А.

Отчет оформляется по общеустановленным правилам в *электронном виде* со следующим содержанием:

- 1) титульный лист,
- 2) тема и цель практического занятия,
- 3) задание на практическое занятие,
- 4) текст программы с комментариями,
- 5) результаты работы программы и
- 6) выводы по разработанным элементам программы.

Приложение

№ п/п	Базовый класс	Производные классы	Вычисляемый показатель
1.	Компьютер	Настольный компьютер, ноутбук	Экземпляр с наибольшей оперативной памятью
2.	Локальная сеть	Одноранговая сеть, сеть типа клиент-сервер	Минимальная стоимость монтажа
3.	Транспортное средство	Легковой автомобиль, грузовой автомобиль	Максимальный пробег на полном бензобаке
4.	Программный продукт	Операционная система, текстовый редактор	Последняя версия
5.	Документ	Паспорт, студенческий билет	Количество документов, выданных в прошлом году
6.	Диета	Для здоровья, для похудения	Максимальное дневное количество белков
7.	Периферийное устройство компьютера	Клавиатура, мышь	Минимальная цена устройства
8.	Хозтовары	Инструменты, посуда	Товар с максимальной ценой
9.	Представитель университета	Преподаватель, студент	Количество студентов, обучающихся у конкретного преподавателя
10.	Предприятие малого бизнеса	Магазин, парикмахерская	Название предприятия с максимальным числом сотрудников
11.	СУБД	Реляционная, иерархическая	Количество СУБД заданного производителя
12.	Страховой полис	Полис обязательного медицинского страхования, страхования жилища	Количество полисов на заданную фамилию
13.	Наушники	Проводные, беспроводные	Экземпляр с максимальным частотным диапазоном
14.	Недвижимость	Таунхаус, квартира в многоквартирном доме	Максимальная жилая площадь
15.	Часы	Электронные часы, механические часы	Самый дорогой экземпляр
16.	Телефон	Смартфон, кнопочный	Самая новая модель
17.	Бытовая техника	Телевизор, холодильник	Количество товаров заданной фирмы
18.	Магнитная карта для проезда на транспорте	Карта общего назначения для проезда в метро, льготная транспортная карта учащегося	Количество карт без поездок

		ся	
19.	Насекомое	Стрекоза, бабочка	Максимальный размер крыльев
20.	Канцелярские товары	Бумага, авторучка	Количество товаров заданной фирмы
21.	Товар	Посуда, продукты питания	Сумма покупки
22.	Учащийся	Школьник, студент	Учащийся с максимальным IQ (коэффициентом интеллекта)
23.	Путешествие (тур)	Морской круиз, отдых на море	Самый дешевый тур на 7 и более дней
24.	Мебель	Кровать, стол	Количество предметов из дерева
25.	Программное обеспечение (ПО)	Системное ПО, прикладное ПО	Представитель с максимальным объемом занимаемой памяти
26.	Представление	Театр, кино	Минимальное число зрителей в зале
27.	Запоминающее устройство	Флэш-карта, карта памяти	Экземпляры с максимальным и минимальным размером
28.	Принтер	Струйный, лазерный	Представитель с наибольшей производительностью
29.	Книга	Электронная, на бумаге	Количество книг одного автора
30.	Телефон	Смартфон, кнопочный	Самая новая модель
31.	Документ	Паспорт, водительские права	Количество документов на одну фамилию
32.	Товар	Кондитерские изделия, хлебные продукты	Минимальная по весу покупка
33.	Строительные материалы	Сыпучие материалы, инструмент	Сумма покупки