Практическое занятие №6

«Наследование»

(Продолжительность работы 2 часа)

Цели:

Изучить возможности наследования классов на языке C++.

1. Краткие теоретические сведения

Каждый объект одного и того же класса имеет собственную копию данных класса. Но существуют задачи, когда данные должны быть компонентами класса, и иметь их нужно только в единственном числе. Такие компоненты должны быть определены в классе как *статические* (**static**). Статические данные классов не дублируются при создании объектов, т.е. каждый статический компонент существует в единственном экземпляре.

Наследование является наиболее значимой возможностью ООП. Наследованием называется процесс создания новых классов, называемых наследниками, дочерними или производными классами из уже существующих – базовых или родительских классов. Производный класс получает все возможности базового класса, но имеет также и свои собственные.

Выигрыш от применения наследования состоит в том, что наследование позволяет использовать существующий код несколько раз. Имея написанный и отлаженный базовый класс его можно больше не модифицировать, а механизм наследования позволит приспособить его для новых задач путем порождения от него новых производных классов. Использование уже написанного и отлаженного кода увеличивает надёжность программ.

Наследование также является упрощением распространения библиотек классов. Программист может использовать классы, созданные кем-то другим, без модификации кода, просто создавая производные классы и добавляя к ним новые возможности.

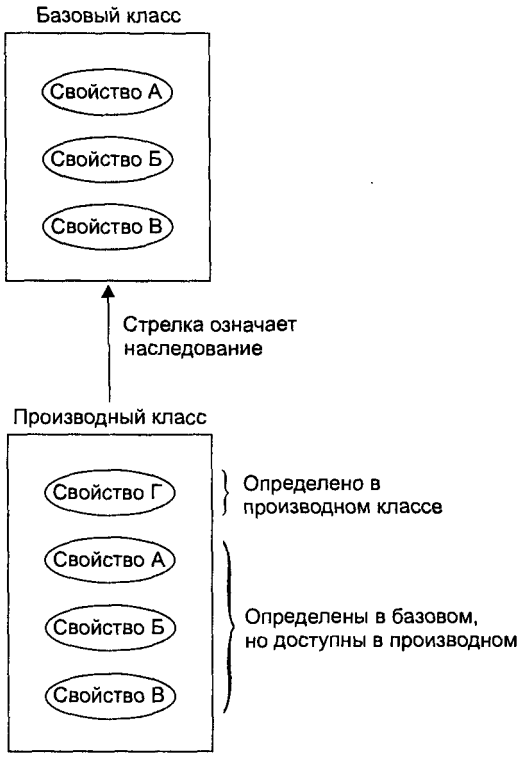

Рисунок 1 - Сущность наследования

БАЗОВЫЙ И ПРОИЗВОДНЫЙ КЛАССЫ

Класс в C++ может *наследовать* элементы-данные и элементы-функции от одного или нескольких б*азовых классов.* Сам класс называется в этом случае *производным* по отношению к базовым классам или *классом-потомком.* В свою очередь, производный класс может являться базовым по отношению к другим классам.

Принцип наследования, или *порождения* новых классов, позволяет абстрагировать (инкапсулировать) некоторые общие свойства и поведение в одном базовом классе, которые будут наследоваться всеми его потомками.

Наследование позволяет также модифицировать поведение базового класса. Производный класс может переопределять некоторые функции-элементы базового класса, оставляя основные свойства класса в неприкосновенности.

Синтаксис производного класса следующий:

```
class имя класса: ключ доступа имя_базового класса [, ...]
 {
тело_объявления_класса
} ;
```

Ключ_доступа — это одно из ключевых слов *private, protected* или *public*. Для производного класса доступны разделы *protected* и *public* базового класса; раздел *private* строго недоступен вне области действия базового класса, раздел *protected* недоступен для функций, не принадлежащих к базовому либо дочернему классу.

Таблица 1 - Наследование и доступ

Специфиикатор доступа	Доступ из самого класса	Доступ из производных классов	Доступ из внешних классов и функций
public	Есть	Есть	Есть
protected	Есть	Есть	Нет
private	Есть	Нет	Нет

Для доступа к элементам базового класса через производный можно сформулировать такое правило: права доступа, определяемые для них базовым классом, остаются неизменными, если они такие же или строже, чем специфицировано *ключом доступа*. В противном случае права доступа определяются ключом в определении производного класса. То есть ключ доступа, указанный при наследовании «сдвигает» права доступа в сторону «более строгих».

Например, при наследовании с ключом *public* права доступа к элементам базового класса остаются неизменными; при закрытом наследовании (ключ *private*) все элементы базового класса будут недоступны за пределами производного класса. При наследовании с ключом *protected,* элементы, которые были *public* становятся *protected*, остальные остаются без изменения.

При закрытом наследовании можно сделать некоторые открытые функции базового класса открытыми в производном, если переобъявить их имена в производном классе. Пример:

```
class First {
public:
void FFunc(void) ;
//...
 };

class Second: private First {
public:
```

```
First::FFunc; // First::FFunc() открыта в классе Second.
//.. .
};
```

Как правило, в практических задачах применяется почти исключительно открытое наследование.

Язык C++ допускает простое и сложное наследование. При простом наследовании у класса может быть только один родитель, при сложном – два и более.

ПРОСТОЕ НАСЛЕДОВАНИЕ

Пример простого наследования:

```
class Time       // Базовый класс - время.
{
int hr, min;
public:
Time(int h=12, int m=0): hr(h), min(m) {}            // Конструктор
void SetTime(int h, int m) {hr = h; min = m; }     //Метод установки времени
void Show( ) { printf("%02d:%02d", hr, min) };   // Метод, чтобы показать время
// на консоли
};

class Alarm: public Time      // Класс сообщений таймера – дочерний от Time.
{
char *msg;  // Указатель на строку-сообщение
public:
Alarm(char*);  // Конструктор нового класса
~Alarm() { delete [] msg; } // Деструктор
void SetMsg(char*);           // Метод установки сообщений
void Show(); // Переопределяет метод Show ( ).
};

Alarm::Alarm(char *str)   // Конструктор класса-потомка
{
msg = new char[strlen (str) + 1];
strcpy(msg, str);
}

void Alarm::SetMsg(char *str)   // Описание функции класса-потомка SetMsg( )
{
delete [] msg;
msg = new char[strlen (str) + 1];
strcpy(msg, str);
}

void Alarm::Show( ) // Переопределение функции Show( )
{
Time::Show(); // Вызов базовой Show()
printf(": %s\n", msg);
}
```

Теперь можно создавать переменные производного класса, например, таким образом:

```
int main ( ) {
Alarm a = "Test Alarm!!!"; // Время по умолчанию 12:00.
a.Show ( );
a.SetTime(7, 40); // Функция базового класса
a.Show ( ) ;
a.SetMsg("It's time! " ); // Функция производного класса.
a.Show( );
return 0;
}
```

Несколько слов о наследовании конструкторов. При создании объекта производного класса компилятор перед вызовом конструктора производного класса автоматически вызывает сначала конструктор родительского класса, но только тот, который может быть вызван без параметров (то есть либо конструктор по умолчанию, либо конструктор без параметров, либо конструктор с параметрами по умолчанию – как в примере выше у класса *Time*). Но конструктор родительского класса с параметрами не наследуется производным в том смысле, что он не будет вызываться автоматически при создании объекта производного класса с такими параметрами.

Вместе с тем, нужный конструктор базового класса можно вызвать явно через список инициализации. Например, дочерний класс из предыдущего примера можно было задать и так:

```
class Alarm: public Time  // Класс сообщений таймера.
{
char *msg;
public:
Alarm(char*);
Alarm(char*, int, int);       // Новый конструктор.
~Alarm( ) { delete[] msg; }
void SetMsg(char*) ;
void Show(); // Переопределяет Time:: Show ( ).
};

Alarm::Alarm(char *str, int h, int m): Time(h, m)  - // явный вызов конструктора
                                                     //родительского класса
{
msg = new char[strlen(str) + 1];
strcpy(msg, str);
}
```

Деструкторы базовых классов всегда вызываются компилятором автоматически.

СЛОЖНОЕ НАСЛЕДОВАНИЕ

Язык C++ допускает не только простое, но и *сложное наследование,* т. е. наследование от двух и более непосредственных базовых классов. Это позволяет создавать классы, комбинирующие в себе свойства нескольких независимых классов-предков. Синтаксис производного класса Alarm при сложном наследовании от классов Time и Message:

```
class Alarm: public Time, public Message
{
…….
};
```

В определенных ситуациях могут появиться проблемы, связанные со сложным наследованием. Например, в обоих базовых классах могут существовать методы с одинаковыми именами, а в производном классе такой метод не переопределяется. Как в этом случае объект производного класса определит, какой из методов базовых классов выбрать? Одного имени метода недостаточно, поскольку компилятор не сможет вычислить, какой из двух методов имеется в виду. Проблема решается путем использования оператора разрешения области действия, определяющего класс, в котором находится метод. Пример:

```
class A
{
  public:
    void show ( ) { cout << "Класс A\n"; }
};
class B
{
  public:
    void show ( ) { cout << "Класс B\n"; }
};
class C : public A, public B
{
};
////////////////////////////////////////////////////////
int main ( )
{
  C objC;           // объект класса C
  // objC.show ( ); // так делать нельзя - программа не скомпилируется
  objC.A::show ( ); // так можно
  objC.A::show ( ); // так можно
  return 0;
}
```

Другой вид неопределенности появляется, если создается производный класс от двух базовых классов, которые, в свою очередь, являются производными одного класса. Это создает дерево наследования в форме ромба. Компилятор не сможет решить, какой из методов использовать, и сообщит об ошибке.

Существует множество вариаций этой проблемы. Поэтому технология программирования предписывает избегать множественного наследования.

2. Практическое задание

Реализовать иерархию классов для простого (задание 1) и сложного (задание 2) наследования:

2.1. Индивидуальное задание. Простое наследование (50%)

1. Создать класс квадрат, члены класса – длина стороны. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – диагональ, периметр, площадь. Создать производный класс – правильная квадратная призма с высотой H, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с этими классами: дано N квадратов и M призм, найти квадрат с максимальной площадью и призму с максимальной диагональю.

2. Создать класс треугольник, члены класса – длины 3-х сторон. Предусмотреть в классе методы проверки существования треугольника, вычисления и вывода сведений о фигуре – длины сторон, углы, периметр, площадь. Создать производный класс – равносторонний треугольник, перегрузить в классе проверку, является ли треугольник равносторонним и метод вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано K треугольников и L равносторонних треугольников, найти среднюю площадь для K треугольников и наибольший равносторонний треугольник.

3. Создать класс окружность, член класса – радиус R. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – площади, длины окружности. Создать производный класс – круглый прямой цилиндр с высотой h, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N окружностей и M цилиндров, найти окружность максимальной площади и средний объем цилиндров.

4. Создать класс квадрат, члены класса – длина стороны. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – диагоналей, периметр, площадь. Создать производный класс – правильная пирамида с апофемой h, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N квадратов и M пирамид, найти квадрат с минимальной площадью и количество пирамид с высотой более числа a (a вводить).

5. Создать класс четырехугольник, члены класса – координаты 4-х точек. Предусмотреть в классе методы проверки существования четырехугольника вычисления и вывода сведений о фигуре – длины сторон, диагоналей, периметр, площадь. Создать производный класс – параллелограмм, предусмотреть в классе проверку, является ли фигура параллелограммом. Написать программу, демонстрирующую работу с классом: дано N четырехугольников и M параллелограммов, найти среднюю площадь N четырехугольников и параллелограммы наименьшей и наибольшей площади.

6. Создать класс треугольник, члены класса – координаты 3-х точек. Предусмотреть в классе методы проверки существования треугольника, вычисления и вывода сведений о фигуре – длины сторон, углы, периметр, площадь. Создать производный класс – равносторонний треугольник, предусмотреть в классе проверку, является ли треугольник равносторонним. Написать программу, демонстрирующую работу с классом: дано N треугольников и M равносторонних треугольников, вывести номера одинаковых треугольников и равносторонний треугольник с наименьшей медианой.

7. Создать класс прямоугольник, члены класса – длины сторон a и b. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – длины сторон, диагоналей, периметр, площадь. Создать производный класс – параллелепипед с высотой c, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N прямоугольников и M параллелепипедов, найти количество прямоугольников, у которых площадь больше средней площади прямоугольников и количество кубов (все ребра равны).

8. Создать класс окружность, член класса – радиус R. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – площади, длины окружности. Создать производный класс – конус с высотой h, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N окружностей и M конусов,

найти количество окружностей, у которых площадь меньше средней площади всех окружностей, и наибольший по объему конус.

9. Создать класс четырехугольник, члены класса – координаты 4-х точек. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – длины сторон, диагоналей, периметр, площадь. Создать производный класс – равнобочная трапеция, предусмотреть в классе проверку, является ли фигура равнобочной трапецией. Написать программу, демонстрирующую работу с классом: дано N четырехугольников и M трапеций, найти максимальную площадь четырехугольников и количество четырехугольников, имеющих максимальную площадь, и трапецию с наименьшей диагональю.

10. Создать класс равносторонний треугольник, член класса – длина стороны. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – периметр, площадь. Создать производный класс – правильная треугольная призма с высотой H, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N треугольников и M призм. Найти количество треугольников, у которых площадь меньше средней площади треугольников, и призму с наибольшим объемом.

11.Создать класс треугольник, члены класса – длины 3-х сторон. Предусмотреть в классе методы проверки существования треугольника, вычисления и вывода сведений о фигуре – длины сторон, углы, периметр, площадь. Создать производный класс – прямоугольный треугольник, предусмотреть в классе проверку, является ли треугольник прямоугольным. Написать программу, демонстрирующую работу с классом: дано N треугольников и M прямоугольных треугольников, найти треугольник с максимальной площадью и прямоугольный треугольник с наименьшей гипотенузой.

12.Создать класс четырехугольник, члены класса – координаты 4-х точек. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – длины сторон, диагоналей, периметр, площадь. Создать производный класс – квадрат, предусмотреть в классе проверку, является ли фигура квадратом. Написать программу, демонстрирующую работу с классом: дано N четырехугольников и M квадратов, найти четырехугольники с минимальной и максимальной площадью и номера одинаковых квадратов.

13.Создать класс треугольник, члены класса – длины 3-х сторон. Предусмотреть в классе методы проверки существования треугольника, вычисления и вывода сведений о фигуре – длины сторон, углы, периметр, площадь. Создать производный класс – равнобедренный треугольник, предусмотреть в классе проверку, является ли треугольник равнобедренным. Написать программу, демонстрирующую работу с классом: дано N треугольников и M равнобедренных треугольников, найти среднюю площадь для N треугольников и равнобедренный треугольник с наименьшей площадью.

14. Создать класс квадрат, член класса – длина стороны. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – периметр, площадь, диагональ. Создать производный класс – куб, добавить в класс метод определения объема фигуры, перегрузить методы расчета площади и вывода сведений о фигуре. Написать программу, демонстрирующую работу с классом: дано N1 квадратов и N2 кубов. Найти среднюю площадь квадратов и количество кубов с наибольшей площадью.

15.Создать класс четырехугольник, члены класса – координаты 4-х точек. Предусмотреть в классе методы вычисления и вывода сведений о фигуре – длины сторон, диагоналей, периметр, площадь. Создать производный класс – ромб, предусмотреть в классе проверку, является ли фигура ромбом. Написать программу, демонстрирующую

работу с этими классами: дано N четырехугольников и M ромбов, найти четырехугольник с минимальным периметром и среднюю площадь ромбов.

2.2 Индивидуальное задание. Множественное наследование (50%)

1. Используя родительский класс «ТРАНСПОРТ» породить производный класс «АВТОМОБИЛЬ». Используя классы «ВОДИТЕЛЬ» и «АВТОМОБИЛЬ», описать класс «ВОДИТЕЛЬ АВТОМОБИЛЯ». Расширить класс «ВОДИТЕЛЬ АВТОМОБИЛЯ» создав два производных класса «ВОДИТЕЛЬ СЛУЖЕБНОГО АВТОМОБИЛЯ» и «ВОДИТЕЛЬ ТАКСИ». Продумать для данной иерархии классов поля и методы (обязательно: вывод информации о водителе, автомобиле)

2. Используя родительский класс «СЛУЖАЩИЙ» породить производный класс «ДИРЕКТОР». Используя классы «ФИРМА» и «ДИРЕКТОР», описать класс «РУКОВОДИТЕЛЬ ФИРМЫ». Расширить класс «РУКОВОДИТЕЛЬ ФИРМЫ» создав два производных класса «РУКОВОДИТЕЛЬ ГОС.УЧ.» и «РУКОВОДИТЕЛЬ ООО». Продумать для данной иерархии классов поля и методы (обязательно: вывод информации о фирме и руководителе)

3. Используя родительский класс «НЕДВИЖИМОСТЬ» породить производный класс «ЗДАНИЕ». Используя классы «ЗДАНИЕ» и «ВЛАДЕЛЕЦ», описать класс «ВЛАДЕЛЕЦ ЗДАНИЯ». Расширить класс «ВЛАДЕЛЕЦ ЗДАНИЯ» создав два производных класса «ВЛАДЕЛЕЦ ЧАСТНОГО ДОМА» и «ВЛАДЕЛЕЦ ОТЕЛЯ ». Продумать для данной иерархии классов поля и методы (обязательно: вывод информации о владельце и здании)

4. Используя родительский класс «ТРАНСПОРТ» породить производный класс «САМОЛЕТ». Используя классы «ПИЛОТ» и «САМОЛЕТ», описать класс «ПИЛОТ САМОЛЕТА». Расширить класс «ПИЛОТ САМОЛЕТА» создав два производных класса «ПИЛОТ ГРАЖДАНСКОГО САМОЛЕТА» и «ПИЛОТ ВОЕННОГО САМОЛЕТА». Продумать для данной иерархии классов поля и методы (обязательно: вывод информации о пилоте, самолете)

5. Используя родительский класс «РАБОТНИК» породить производный класс «РЕЖИССЕР». Используя классы «ФИЛЬМ» и «РЕЖИССЕР», описать класс «РЕЖИССЕР ФИЛЬМА». Расширить класс «РЕЖИССЕР ФИЛЬМА» создав два производных класса «РЕЖИССЕР ХУДОЖЕСТВЕННОГО ФИЛЬМА» и «РЕЖИССЕР ДОКУМЕНТАЛЬНОГО ФИЛЬМА». Продумать для данной иерархии классов поля и методы (обязательно: вывод информации о фирме и руководителе)

3. Список рекомендуемой литературы

1. Павловская Т. А.C/C++. Программирование на языке высокого уровня : для магистров и бакалавров : учеб. для вузов / Т. А. Павловская. - Гриф МО. - Санкт-Петербург: Питер, 2013. - 460 с. : ил.

2. Professional C++, 3rd Edition. Marc Gregoire. ISBN: 978-1-118-85805-9. Paperback 984 pages. September 2014

4. Контрольные вопросы

1. В чем состоит назначение наследования?
2. Когда программисту-разработчику целесообразно прибегнуть к наследованию?
3. Напишите первую строку описания класса Child, который является public-производным от класса Parent.
4. Верно ли утверждение: создание производного класса требует коренных изменений в базовом классе?
5. Члены базового класса, для доступа к ним методов производного класса должны быть объявлены как public или _____ .

6. Пусть базовый класс содержит метод basefunc( ), а производный класс не имеет такого метода. Может ли объект производного класса иметь доступ к методу basefunc( ).
7. Истинно ли следующее утверждение: если конструктор производного класса не определен, то объекты этого класса будут использовать конструкторы базового класса?
8. Как используется оператор разрешения области действия для разрешения неопределенностей?
9. Истинно ли следующее утверждение: иногда полезно создать класс, объектов которого никогда не будет создано?
10. Пусть класс Derv является дочерним от класса Base. Пусть определена переменная типа Derv, расположенная в функции main( ). Через эту переменную можно получить доступ к
    а) членам класса Derv, объявленным как public;
    б) членам класса Derv, объявленным как protected;
    в) членам класса Derv, объявленным как private;
    г) членам класса Base, объявленным как public;
    д) членам класса Base, объявленным как protected;
    е) членам класса Base, объявленным как private;
11. Пусть существует класс Derv, производный от класса Base. Напишите объявление конструктора производного класса, принимающего один аргумент и передающего его в конструктор базового класса.
12. Истинно ли следующее утверждение: невозможно сделать объект одного класса членом другого класса?