Лабораторная работа №12

Наследование классов в языке С++

**Цели:**

Изучить возможности наследования классов на языке С++.

**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.



**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.



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 С++ 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.

а.Show ( );

a.SetTime(7, 40); // Функция базового класса

а.Show ( ) ;

а.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.

Варианты заданий:

|  |  |
| --- | --- |
| Номер варианта | Задание |
| 1, 15 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Человек**  **( Имя,**  **дата рождения)** | ◊ | **Школьник**  **(№ школы)** | ◊ | **Студент**  **(Название вуза,**  **специальность)** |   Выведите на экран характеристики объектов. |
| 2, 16 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Окружность**  **(Радиус, длина окружностиНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 3, 17 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Колесо**  **(Радиус)** | ◊ | **Садовая тележка**  **(Количество колес, грузоподъемность)** | ◊ | **Грузовик**  **(Марка, мощность)** |   Выведите на экран характеристики объектов. |
| 4, 18 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок – координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Равносторонний треугольник (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 5, 19 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Планер**  **(Модель,**  **конструктор,**  **год разработки)** | ◊ | **Самолет**  **(Максимальная скорость, максимальная дальность полета)** | ◊ | **Ракета**  **(Назначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 6, 20 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Круг – радиус равен длине отрезка (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 7, 21 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Рогатка**  **(Дальность боя)** | ◊ | **Автомат**  **(Конструктор)** | ◊ | **Баллистическая ракета**  **(Класс, поражающая мощность)** |   Выведите на экран характеристики объектов. |
| 8, 22 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Квадрат - сторона равна длине отрезка (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 9, 23 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Счеты**  **(Разрядность, выполняемые операции)** | ◊ | **Калькулятор**  **(Цена)** | ◊ | **Компьютер**  **(Количество процессоров, быстродействие)** |   Выведите на экран характеристики объектов. |
| 10, 24 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Ромб – большая диагональ равна длине отрезка, меньшая – половине длины отрезка (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 11, 25 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Река**  **(Название, период судоходства)** | ◊ | **Море**  **(Площадь)** | ◊ | **Океан**  **(Максимальная глубина)** |   Выведите на экран характеристики объектов. |
| 12, 26 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Прямоугольный треугольник – один из катетов равен длине отрезка, другой – половине длины отрезка (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |
| 13, 27 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Метла**  **(Крейсерская скорость, дальность полета)** | ◊ | **Ступа**  **(Материал, из которого она изготовлена)** | ◊ | **Ковер-самолет**  **(Грузоподемность)** |   Выведите на экран характеристики объектов. |
| 14, 28 | Разработать программу с использованием наследования классов, реализующую классы:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Точка**  **(Координаты)** | ◊ | **Отрезок - координаты начала наследуются из точки**  **(Координаты конца, длина)** | ◊ | **Равносторонний треугольник – сторона-основание и высота равны длине отрезка (ПлощадьНазначение, тип топлива, орбитальная или межпланетная)** |   Выведите на экран характеристики объектов. |