Лабораторная работа 4. Наследование в C++

1. ЦЕЛЬ РАБОТЫ

Изучение принципов создания и использования иерархии классов в программах на C++.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Наследование позволяет повторно использовать уже созданную часть программного кода в других проектах. Посредством наследования формируются связи между объектами, а для выражения процесса наследования используются термины «родители» и «потомки». В программировании наследование служит для сокращения избыточности кода.

Общий формат для обеспечения наследования имеет следующий вид:

```
class имя_производного класса: доступ имя_базового класса
{
     тело нового класса
 }
```

Элемент `доступ` может принимать одно из значений `public`, `private` или `protected`.

Базовый класс может наследоваться как общедоступный (`public`) и как приватный (`private`) базовый класс. В первом случае новый класс является расширением базового класса, в то время как во втором случае новый класс реализует специализацию базового класса. Спецификатор `protected` позволяет создать член класса, который будет доступен в рамках данной иерархии классов, но закрыт для остальных элементов программы.

Когда два или более класса порождаются от одного общего базового класса, можно предотвратить включение нескольких копий базового класса в объект-потомок этих классов путем объявления базового класса *виртуальным* при его наследовании.

Виртуальная функция — это функция, объявленная с ключевым словом `virtual` в базовом классе и переопределенная в одном или в нескольких производных классах. Таким образом, каждый производный класс может иметь собственную версию виртуальной функции. Возможна ситуация, когда виртуальная функция вызывается через указатель или ссылку на базовый класс. В этом случае C++ определяет, какая именно версия функции вызывается, по типу объекта, адресуемого этим указателем. Причем это решение принимается во время выполнения программы.

Полиморфизм времени исполнения достигается только при вызове виртуальной функции с использованием указателя или ссылки на базовый класс.

Если функция была объявлена как виртуальная, то она и остается таковой вне зависимости от количества уровней в иерархии классов, через которые она прошла. Атрибут `virtual` передается по наследству.

Когда класс не переопределяет виртуальную функцию, C++ использует первое из определений, которое он находит, идя от потомков к предкам.

Виртуальные функции в комбинации с производными типами позволяют языку C++ поддерживать полиморфизм времени исполнения. Этот полиморфизм важен для объектно-ориентированного программирования, поскольку он позволяет переопределять функции базового класса в классах-потомках с тем, чтобы иметь их версию применительно к данному конкретному классу. Таким образом, базовый класс определяет общий интерфейс, который имеют все производные от него классы, и вместе с тем полиморфизм позволяет производным классам иметь свои собственные реализации методов. Благодаря этому полиморфизм часто определяют фразой «один интерфейс — множество методов».

Успешное применение полиморфизма связано с пониманием того, что базовые и производные классы образуют иерархию, в которой переход от базового к производному классу отвечает переходу от большей к меньшей общности. Поэтому при корректном использовании базовый класс обеспечивает все элементы, которые производные классы могут непосредственно использовать, плюс набор функций, которые производные классы должны реализовать путем их переопределения.

Наличие общего интерфейса и его множественной реализации является важным постольку, поскольку помогает программистам разрабатывать сложные программы. Например, доступ ко всем объектам, производным некоторого базового класса, осуществляется одинаковым способом, даже если реальные действия этих объектов отличаются при переходе от одного производного класса к другому. Это означает, что необходимо запомнить только один интерфейс, а не несколько. Более того, отделение интерфейса от реализации позволяет создавать библиотеки классов, поставляемые независимыми разработчиками. Если эти библиотеки реализованы корректно, то они обеспечивают общий интерфейс, и их можно использовать для вывода своих собственных специфических классов.

Когда виртуальная функция не переопределена в производном классе, то при вызове ее в объекте производного класса вызывается версия из базового класса. Однако во многих случаях невозможно ввести содержательное определение виртуальной функции в базовом классе. Имеется два способа действий в подобной ситуации. Первый заключается в выводе какого-нибудь предупреждающего сообщения. Хотя такой подход полезен в некоторых ситуациях, он не является универсальным. Могут быть такие виртуальные функции, которые обязательно должны быть определены в производных классах, без чего эти производные классы не будут иметь никакого значения. В таких случаях необходим метод, гарантирующий, что производные классы действительно определят все необходимые функции. Язык C++ предлагает в качестве решения этой проблемы чисто виртуальные функции.

Чисто виртуальная функция (pure virtual function) является функцией, которая объявляется в базовом классе, но не имеет в нем определения. Поскольку она не имеет определения, то есть тела в этом базовом классе, то всякий производный класс обязан иметь свою собственную версию определения. Для объявления чисто виртуальной функции используется следующая общая форма:

```
virtual тип имя_функции(список параметров) = 0;
```

Здесь тип обозначает `тип` возвращаемого значения, а `имя_функции` является именем функции.

При введении чисто виртуальной функции в производном классе обязательно необходимо определить свою собственную реализацию этой функции. Если класс не будет содержать определения этой функции, то компилятор выдаст ошибку.

Если какой-либо класс имеет хотя бы одну чисто виртуальную функцию, то такой класс называется *абстрактным* (abstract). Важной особенностью абстрактных классов является то, что не существует ни одного объекта данного класса. Вместо этого абстрактный класс служит в качестве базового для других производных классов. Причина, по которой абстрактный класс не может быть использован для объявления объекта, заключается в том, что одна или несколько его функций-членов не имеют определения. Тем не менее, даже если базовый класс является абстрактным, все равно можно объявлять указатели или ссылки на него, с помощью которых затем поддерживается полиморфизм времени исполнения.

Абстрактные классы используются в качестве обобщенных концепций, на основе которых можно создавать более конкретные производные классы. Невозможно создать объект типа абстрактного класса; однако можно использовать указатели и ссылки на типы абстрактного класса.

Класс, содержащий хотя бы одну чисто виртуальную функцию, считается абстрактным.

Классы, производные от абстрактного класса, должны реализовывать чисто виртуальную функцию; в противном случае они также будут абстрактными.

Рассмотрим пример, представленный в разделе Виртуальные функции справочной системы MSDN. Класс `Account` создан для того, чтобы предоставлять общие функции, но объекты типа `Account` имеют слишком общий характер для практического применения. Таким образом, класс `Account` является хорошим кандидатом в абстрактный класс:

```
// deriv_AbstractClasses.cpp
// compile with: /LD
class Account {
public:
   Account( double d );   // Constructor.
   virtual double GetBalance();   // Obtain balance.
   virtual void PrintBalance() = 0;   // Pure virtual function.
private:
    double _balance;
};
```

Более подробные примеры были рассмотрены на лекциях по данной теме.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Ознакомьтесь с теоретическим материалом, приведенным в пункте «Краткие теоретические положения» данных методических указаний и конспектом лекций по данной теме.

Задание 2. Разработайте иерархию классов по своему варианту (или по собственному выбору). Кроме указанных в варианте задания свойств и методов, можно добавить свои, необходимые по смыслу предметной области, свойства и методы классов.

Минимальные требования:

- ✓ не менее двух виртуальных функций,
- ✓ не менее трех свойств у классов-потомков;
- ✓ не менее трех методов;
- ✓ наличие конструкторов у всех классов.

Составьте диаграмму классов и покажите ее для согласования преподавателю.

После этого реализуйте составленную иерархию классов на языке C++ (в виде подключаемых .h или .cpp файла или в виде DLL).

Задание 3. Разработайте основную программу (cpp-файл с функцией `main`), в которой используются созданные классы. В программе должны демонстрироваться возможности созданных классов.

4. ОФОРМЛЕНИЕ ОТЧЕТА

Отчет по работе должен содержать:

- название и цель работы;
- номер варианта;
- файлы *.h и *.cpp, содержащие описание и реализацию классов;
- текст основной программы с комментариями;
- описание разработанных классов, структур данных и функций, в том числе диаграмма классов;
- результаты работы программы для нескольких вариантов входных данных.

5. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шилдт Г. C++: базовый курс, 3-е издание. : Пер. с англ. – М.: «Издательский дом «Вильямс», 2005. – 624 с.

2. Пахомов Б.И. C/C++ и MS Visual C++ 2012 для начинающих. – СПб.: БХВ-Петербург, 2013. – 509 с.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы данных называются абстрактными в C++?
2. Каким образом производиться объявление класса?
3. Какие спецификаторы доступа используются при объявлении класса?
4. В чем заключается инкапсуляция? Как реализуется инкапсуляция при создании абстрактных типов данных?
5. Какие правила существуют для перегрузки операций для классов?

6. Что такое полиморфизм? Как реализуется полиморфизм при создании абстрактных типов данных?

ВАРИАНТЫ ЗАДАНИЙ

Вар.	**Задание**
1	Целое число, число в двоичной системе счисления (СС), число в троичной СС, число в СС с произвольным основанием.
2	Домашнее животное, собака, хомячок, служебная собака, рыбка
3	Домашнее животное, собака, кошка, канарейка, попугай
4	Домашнее животное, собака, кошка, служебная собака, овчарка
5	Уравнение, линейное уравнение, квадратное уравнение
6	Животное, млекопитающее, птица, зайцы, лисы, орлы, зоопарк
7	Уравнение, линейное уравнение, квадратное уравнение
8	Домашнее животное, собака, кошка, канарейка, попугай
9	Алгебраическая функция, гипербола, линейная функция, квадратичная функция
10	Товар, съедобный товар, несъедобный товар, игрушки, машинка, кукла, молоко, макароны
11	Уравнение, линейное уравнение, квадратное уравнение, уравнение третьей степени
12	Точка, окружность, сектор, дуга
13	Животное, млекопитающее, птица, бобры, ежи, воробьи
14	Человек, студент, преподаватель, лаборант, институт
15	Человек, работник унив-та, преподаватель, сотрудник бухгалтерии
16	Товар, съедобный товар, несъедобный товар, игрушки, машинка, зайчик, торт, конфеты
17	Товар, съедобный товар, молочный товар, молоко, творог, магазин
18	Машина, легковой автомобиль, грузовик, автобус, автомагазин
19	Человек, студент, студент-дипломник, выпускник, учебная группа
20	Животное, млекопитающее, птица, бобры, ежи, орлы, зоопарк
21	Печатное издание, журнал, книга, учебник, книжный магазин
22	Фигуры, четырехугольники, квадраты, ромбы, трапеции
23	Транспортное средство, корабль, яхта, пароход, автомобиль, легковой автомобиль
24	Товар, игрушки, машинка, кукла, торт, конфеты, магазин
25	Алгебраическая функция, гипербола, линейная функция, еще какая-нибудь функция
26	Уравнение, неравенство, линейное уравнение, квадратное уравнение, линейное неравенство, математическая модель
27	Алгебраическая функция, линейная функция, квадратичная функция, еще

	какая-нибудь функция
28	Животное, млекопитающее, птица, домашнее животное, собака
29	Домашнее животное, собака, хомячок, служебная собака, рыбка
30	