Задание № 11

Конструирование классов на основе принципа наследования

Цель работы: изучить механизм открытого (public) наследования в C++, познакомиться с понятием «виртуальная функция», освоить технологию конструирования и способы документирования программы, включающей в себя классы-наследники, изучить возможности инструментальных сред разработки по автоматической генерации кода.

Задание

1. В соответствии с вариантом задания разработать базовый класс. В базовый класс следует включить свойства и методы, общие для заданных классов-наследников. Базовый класс должен включать в себя не менее двух свойств и двух методов, один из которых – виртуальная функция.
2. Разработать классы, производные от базового класса (наследники). Классы-наследники должны наследовать от базового класса хотя бы одно свойство, а также должны иметь хотя бы одно собственное свойство. В классы-наследники должны быть включены следующие методы:
    a. Метод, наследуемый от базового класса без переопределения.
    b. Виртуальная функция базового класса, переопределённая в производном классе.
    c. Собственные методы производного класса. В состав производного класса должен быть включён хотя бы один метод, изменяющий какое-либо свойство класса.
3. Разработать программу, выполняющую следующие действия:
    a. Создание нескольких объектов на основе классов–наследников.
    b. Объединение объектов в массив (массив указателей на базовый класс).
    c. Отображение значений свойств объектов на экране в цикле.
    d. Изменение свойств объектов по номеру элемента массива.
    e. Вычисление заданного параметра.
    f. Выход из программы.
4. Объявление и реализацию каждого класса поместить в отдельный модуль.
5. Действия над объектами (просмотр, изменение, вычисление параметра) должны быть доступны через меню; последовательность выполнения действий – произвольная, в цикле.
6. Представить отчёт следующего содержания:
    a. Постановка задачи;
    b. Текст программы;
    c. Диаграмма классов с указанием свойств и методов класса;
    d. Диаграмма компонентов (см. Лабораторная работа № 3).

Варианты заданий

№ варианта	Базовый класс	Производные классы	Вычисляемый параметр
1	Товар	Телевизор, холодильник	Средняя цена
2	Магнитная карта для проезда на транспорте	Карта общего назначения для проезда в метро, льготная транспортная карта учащегося	Среднее количество поездок
3	Транспортное средство	Легковой автомобиль, грузовой автомобиль	Максимальная емкость бензобака
4	Компьютер	Настольный компьютер, ноутбук	Самый дешевый компьютер

5	Программный продукт	Операционная система, текстовый редактор	Последняя версия программы
6	Документ	Паспорт, студенческий билет	Количество документов на заданную фамилию
7	Периферийное устройство компьютера	Принтер, монитор	Минимальная цена устройства
8	Товар	Одежда, продукты питания	Сумма покупки
9	Страховой полис	Полис обязательного медицинского страхования, страхования жилища	Количество полисов на заданную фамилию
10	Периферийное устройство компьютера	Клавиатура, сканер	Средняя цена
11	Недвижимость	Коттедж, квартира в многоквартирном доме	Максимальная жилая площадь
12	Товар	Электронные часы, кондитерские изделия	Самый дешевый товар
13	Документ	Свидетельство ЕГЭ, зачетная книжка	Средний балл
14	Транспортное средство	Самолет, легковой автомобиль	Самое новое транспортное средство
15	Товар	Телевизор, холодильник	Количество товаров заданной фирмы
16	Магнитная карта для проезда на транспорте	Карта общего назначения для проезда в метро, льготная транспортная карта учащегося	Количество карт без поездок
17	Транспортное средство	Легковой автомобиль, грузовой автомобиль	Количество транспортных средств, выпущенных после заданного года
18	Канцелярские товары	Бумага, авторучка	Количество товаров заданной фирмы
19	Программный продукт	Операционная система, текстовый редактор	Количество программ, выпущенных заданной фирмой
20	Документ	Паспорт, студенческий билет	Документ, выданный раньше всех других
21	Периферийное устройство компьютера	Принтер, монитор	Количество устройств, выпущенных заданной фирмой
22	Товар	Одежда, продукты питания	Самый дорогой товар
23	Страховой полис	Полис обязательного медицинского страхования, страхования жилища	Полис с максимальным сроком действия
24	Периферийное устройство компьютера	Клавиатура, сканер	Средний срок эксплуатации
25	Недвижимость	Коттедж, квартира в многоквартирном доме	Общее количество проживающих
26	Документ	Свидетельство ЕГЭ,	Количество действительных

		зачетная книжка	документов
27	Транспортное средство	Самолет, легковой автомобиль	Самое новое транспортное средство
28	Компьютер	Настольный компьютер, ноутбук	Средняя цена компьютера
29	Канцелярские товары	Бумага, авторучка	Средняя цена товара
30	Телефон	Стационарный телефон, мобильный телефон	Минимальная цена товара

Справочный материал

Наследование

Наследование – это механизм, посредством которого на базе существующих классов создаются новые классы, получающие по наследству от базовых классов часть их свойств и методов. Наследование позволяет существенно экономить программный код.

Созданные с использованием механизма наследования новые классы называются классы-наследники или классы-потомки. Различают открытое, защищенное и закрытое наследование. Выбор варианта наследования Выбор варианта наследования влияет на доступность унаследованных членов класса в классе-наследнике. В данной работе рассматривается только открытое наследование.

При открытом наследовании новые классы наследуют свойства и методы, объявленные в разделе public и protected базовых классов (рисунок 1). При этом свойства и методы из раздела protected базового класса доступны только базовому классу и его наследникам, тогда как свойства и методы из раздела public доступны объектам любого класса. Исключением из этого правила являются конструкторы и деструкторы, которые не наследуются. В классе-наследнике наследуемые свойства получают уровень доступа в соответствии с доступностью в базовом классе: свойства и методы базового класса из раздела public наследуются с уровнем доступа public, а свойства и методы из раздела protected наследуются с уровнем доступа protected.

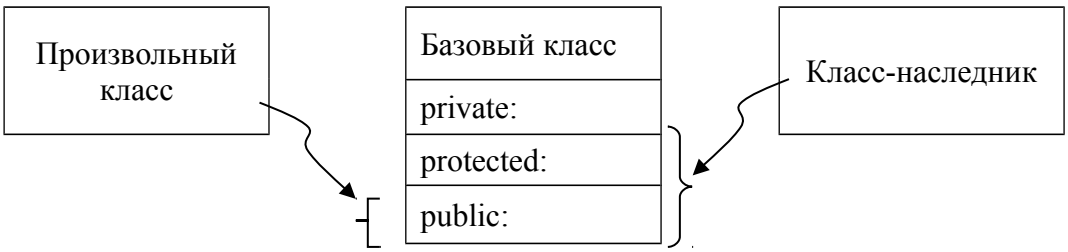

Рисунок 1 Уровни доступа к членам класса

```
//Объявление базового класса
class A
{
        ….      //Члены класса A
} ;
//Объявление класса-наследника
class B : public A   //  public — спецификатор доступа, означает публичный доступ
{
        ….      //Собственные члены класса B
} ;
```

В состав класса B входят собственные члены и члены класса A из разделов public и protected.

Рассмотрим пример наследования. В качестве базового класса возьмём класс Person, характеризующий любого человека, включим в раздел protected свойство класса name – имя человека.

```
class Person
{
protected:
        char name[20];
};
```

Создадим на основе класса Person класс-наследник Student:

```
class Student : public Person
{
        char department[20];
public:
        student(char* _name, char* _department);
        void message( );
};
//Конструктор
Student :: Student(char* _name, char* _department)
{
        strcpy(name, _name);
        strcpy(department, _department);
}
void Student::message( )    //message – метод, сообщающий сведения об объекте
{
        cout << "My name is " << name <<   ". I study at " << department << endl;
}
```

Как видно, методы класса Student используют как собственное свойство, так и свойство, полученное по наследству.

Создание объекта класса Student и вызов его метода ничем не отличаются от того, как это делается для класса без наследования:

```
int main()
{
        Student stud("Nick", "SS,SK,VT");
        stud.message( );
}
```

Во время выполнения такой программы на экране появляется сообщение:

My name is Nick. I study at SS,SK,VT

Теперь рассмотрим наследование методов базового класса. Включим в базовый класс конструктор, инициализирующий собственное свойство базового класса значением "Noname", и метод message, сообщающий информацию о классе.

```
class Person
{
protected:
        char name[20];
public:
        Person( );
        void message( );
};
Person :: Person( )
{
        strcpy(name, "Noname");
}
```

```
void Person::message( )
{
        cout << "My name is " << name << endl;
}
```

Таким образом, в класс-наследник Student теперь включены два метода message, имеющих одинаковый тип и одинаковый список параметров – один из базового класса, другой – собственный. Собственный метод доступен через имя объекта. Однако можно получить доступ к унаследованному методу, если использовать для доступа к методу указатель соответствующего типа - указатель на базовый класс. Рассмотрим это на примере:

```
int main()
{
        Person a;
        Student b("Ann", "SS,SK,VT");
        a.message( ):
        b.message( );
        Person* pperson = &b;
        pperson->message( );
        Student* pstudent = &b;
        pstudent->message( );
}
```

На экране увидим сообщения:

Noname	Сообщение об объекте A
Ann SS,SK,VT	Сообщение об объекте B, используется собственный метод message
Ann	Сообщение об объекте B, используется метод message, полученный по наследству
Ann SS,SK,VT	Сообщение об объекте B используется собственный метод message

В C++ можно избежать получения по наследству метода, если метод с таким же именем включён в класс-потомок, для этого используется механизм виртуальных функций.

Виртуальная функция и полиморфизм

Виртуальные функции - это функция, объявленная в базовом классе с помощью ключевого слова virtual, такая функция в классах-потомках замещается функцией, принадлежащей производному классу и имеющей то же имя. Реализация виртуальной функции базового класса в классе-наследнике означает переопределение (override) этой функции.

Замещение функции у класса-наследника проявляется в том, что при вызове функции с помощью указателя на базовый класс вместо функции базового класса вызывается функция наследника. Таким образом, результат вызова виртуальной функции через указатель на базовый класс зависит от класса объекта, для которого вызывается эта функция. Множество результатов одного и того же действия (вызов виртуальной функции, переопределенной в классах-наследниках) называется полиморфизм, а переопределенные функции — полиморфными.

```
class Person
{
protected:
        char name[20];
public:
        Person( );
        virtual void message( );
};
```

Теперь результатом выполнения программы будут следующие сообщения:

Noname	Сообщение об объекте A
Ann SS,SK,VT	Сообщение об объекте B, используется собственный метод message
Ann SS,SK,VT	Сообщение об объекте B используется собственный метод message
Ann SS,SK,VT	Сообщение об объекте B используется собственный метод message

С помощью виртуальных функций можно создать класс-наследник, имеющий тот же интерфейс, что и базовый класс, но обладающий своей собственной моделью поведения. Механизм виртуальных функций реализуется следующим образом: обычно обработка вызовов функций начинается на этапе компиляции и завершается на этапе редактирования связей, когда вызов метода жёстко связывается с соответствующей функцией (раннее связывание); если метод объявлен как виртуальный, выполняется так называемое позднее связывание – т.е. связывание вызова и функции во время выполнения программы.

Пример программы к 4-й лабораторной работе

В программе созданы классы:
Person — базовый класс,
Student, Prof - классы-потомки класса Person

Каждый класс помещен в отдельный модуль. Пути к заголовочным файлам в директивах #include указаны с учетом каталожной структуры проекта, формирующейся в среде Code::Blocks. Массив указателей на базовый класс ps получает адреса объектов-наследников и используется для обработки данных объектов с помощью цикла.

```
// Файл main.cpp — главный модуль
#include "include\Person.h"
#include "include\Student.h"
#include "include\Prof.h"

using namespace std;

int main()
{
   Student s1("Ivanov",1994, "ABC-122");
   Student s2("Wetrov",1993,"MRT-121");
   Prof p1("Rogov",1964,323232);
   Prof p2("Smirnov",1961,219564);
   Person* ps[4]; // Массив указателей на базовый класс
   int average=0, i;
   //Объединяем объекты разных типов в одном массиве
   ps[0]=&s1;
   ps[1]=&s2;
   ps[2]=&p1;
   ps[3]=&p2;

   //Вывод данных с помощью массива указателей
   for(i=0; i<4; i++)
      ps[i]->show(); //проявляется полиморфизм
```

```
    //Расчет среднего возраста
    for(i=0; i<4; i++)
    {
       average += 2012 - (*(ps+i))->getYear();
    }
    average /= 4;
    cout << "Sredniy vozrast:  " << average << endl;
    return 0;
}
```

// Файл person.h — базовый класс

```
#ifndef PERSON_H
#define PERSON_H

#include <iostream>
#include <string.h>
using namespace std;
class Person
{
   public:
      Person();
      virtual ~Person();
      int getYear();
      virtual void show();
      void setName(char*);
      void setYear(int);
   protected:
      char name[30];
      int year;
   private:
};
#endif // PERSON_H
```

// Файл person.cpp

```
#include "..\include\Person.h"

using namespace std;

Person::Person():year(0)
{
   strcpy(name, "Noname");
}
int Person::getYear()
{
   return year;
}
void Person::show()
{
   cout << "Name:  " << name << "   Year:  " << year << endl;
}
```

```
void Person::setName(char* n)
{
   strcpy(name, n);
}
void Person::setYear(int y)
{
   year = y;
}
```

// Файл student.h — класс-наследник

```
#ifndef STUDENT_H
#define STUDENT_H

#include "Person.h"

class Student : public Person
{
   public:
      Student(char*, int, char*);
      void setGroup(char*);
      void show();
   protected:
   private:
      char group[10];
};
#endif // STUDENT_H
```

// Файл student.cpp

```
#include "..\include\Student.h"

Student::Student( char* n, int y,char* g)
{
   strcpy(name, n);
   strcpy(group, g);
   year =y;
}
void Student::setGroup(char* g)
{
   strcpy(group, g);
}
void Student::show()
{
   cout << "Name:  " << name << "    Year:  " << year << "Group:  "<< group << endl;
}
```

// Файл prof.h - класс-наследник

```
#ifndef PROF_H
#define PROF_H

#include "Person.h"
```

```
class Prof : public Person
{
   public:
      Prof(char*, int, int);
      void setTabNumber(int);
      void show();
   protected:
   private:
      int tabNumber;
};

#endif // PROF_H
// Файл prof.cpp
#include "..\include\Prof.h"

Prof::Prof(char* n, int y, int t)
{
   strcpy(name, n);
   year=y;
   tabNumber=t;
}
void Prof::setTabNumber(int t)
{
   tabNumber = t;
}
void Prof::show()
{
   cout << "Name: " << name << "  Year:  " << year << "   TabNumber: "<< tabNumber;
   cout << endl;
}
```

Создание класса-наследника в среде Netbeans

1. Выполнить команду File → New File → Categories (C++) → File Types (C++ Class) → Next.
2. Ввести имя класса, нажать кнопку Finish. После этого в проект будет добавлен новый класс, причём объявление класса будет помещено в файл *.h, а реализация – в файл *.cpp.
3. Вручную добавить текст, определяющий базоый класс.
4. Вручную добавить в класс свойства и методы.

Создание класса-наследника в среде Code::Blocks

1. Выполнить команду File → New → Class.
2. В окне Create new class указать имя класса, установить флажок inherits anothe class и в окне Ancestor указать имя базового класса, в окне Scope оставить слово public.
3. В окне Member Variables добавить свойства класса, для каждого свойства оставить или сбросить флажки, предлагающие автоматически включить в класс методы типа Get и Set.
4. Нажать кнопку Create. После этого в проект будет добавлен новый класс, причём объявление класса будет помещено в файл *.h, а реализация – в файл *.cpp, а в каталоге проекта появятся каталоги include для хранения заголовочных файлов и src для хранения файлов *.cpp.

5. Добавить вручную недостающие свойства и методы, внести исправления в текст, полученный автоматически.

Создание класса-наследника в среде Visual C++ 6.0

1. Выполнить команду New Class…
2. В окне New Class указать в поле Name имя класса-наследника, в поле Base Class(es) в графе Derived From – имя базового класса, в графе As оставить значение public.
3. Нажать кнопку Ok. После этого в проект будет добавлен новый класс, причём объявление класса будет помещено в файл *.h, а реализация – в файл *.cpp.
4. Добавить в новый класс свойства (Add Member Variable) и методы (Add Member Function).

Создание класса-наследника в среде Visual C++.NET

1. Выполнить команду Add Class…
2. В окне Add Class выбрать категорию и шаблон класса – Generic.
3. В окне Class name ввести имя класса-наследника, в поле Base class – имя базового класса, в окне Access оставить значение public.
4. Нажать кнопку Finish. После этого в проект будет добавлен новый класс, причём объявление класса будет помещено в файл *.h, а реализация – в файл *.cpp.
5. Добавить в новый класс: добавить свойства (Add variable) и методы (Add function).

Диаграмма классов

Диаграмма классов — это один из видов диаграмм языка UML. Она предназначена для визуального представления связей, установленных между классами.

Обозначения класса, используемые в диаграмме классов, приведены на рисунке 2.

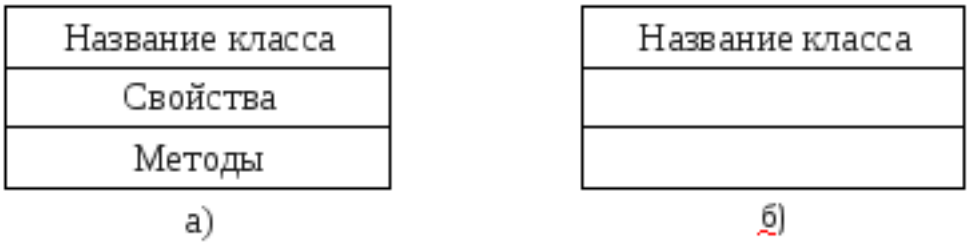

Рисунок 2. Обозначение классов в UML: а) подробное; б) упрощенное

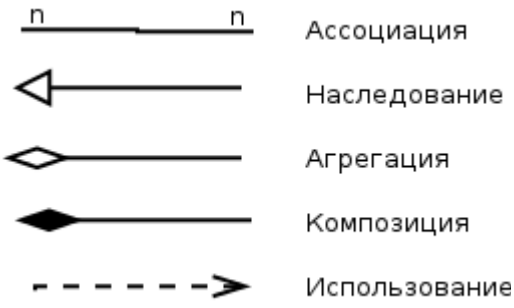

Рисунок 3. Обозначение связей в диаграмме классов

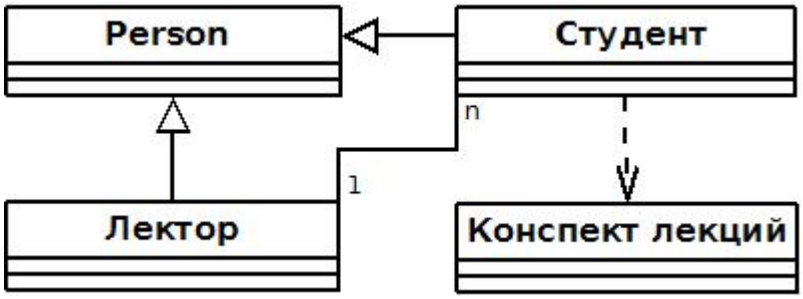

Рисунок 4. Пример диаграммы классов

На рисунке 3 показаны обозначения связей для этой диаграммы, пример диаграммы представлен на рисунке 4. Все связи, кроме ассоциативной, имеют направление. Связь направлена от зависимого класса к независимому, например, при наследовании независимым является базовый класс, в агрегации и композиции — внешний класс, в использовании — класс, предоставляющий свои ресурсы (сервер).

Вопросы к защите

1. Что такое наследование?
2. Перечислите преимущества использования наследования.
3. Какие члены базового класса будут доступны наследнику?
4. Что такое виртуальная функция?
5. Как проявляется полиморфизм?
6. Можно ли в один массив поместить объекты разных классов? Ответ обоснуйте.