Лабораторная работа №5

Наследование. Виртуальные функции. Полиморфизм.

1. Цель задания:

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2) Создание иерархии классов с использованием простого наследования и абстрактного класса.
3) Изучение полиморфизма и виртуальных методов.

2. Теоретические сведения

2.1. Абстрактные классы

Класс, содержащий хотя бы один чисто виртуальный метод, называется *абстрактным.*

Чисто виртуальный метод содержит признак = 0 вместо тела, например:

```
virtual void f(int) = 0;
```

Чисто виртуальный метод должен переопределяться в производном классе (возможно, опять как чисто виртуальный).

Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. Абстрактный класс может использоваться *только в качестве базового* для других классов — объекты абстрактного класса создавать нельзя, поскольку прямой или косвенный вызов чисто виртуального метода приводит к ошибке при выполнении.

При определении абстрактного класса необходимо иметь в виду следующее:

- абстрактный класс нельзя использовать при явном приведении типов, для описания типа параметра и типа возвращаемого функцией значения;
- допускается объявлять указатели и ссылки на абстрактный класс, если при инициализации не требуется создавать временный объект;
- если класс, производный от абстрактного, не определяет все чисто виртуальные функции, он также является абстрактным.

Таким образом, можно создать функцию, параметром которой является указатель на абстрактный класс. На место этого параметра при выполнении программы может передаваться указатель на объект любого производного класса. Это позволяет создавать *полиморфные функции,* работающие с объектом любого типа в пределах одной иерархии

3. Постановка задачи

1. Определить абстрактный класс.
2. Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).
3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.
5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

4. Ход работы

Задача

Базовый класс:
МАШИНА
торговая_марка - string
число_цилиндров - int
мощность - int
Создать производный класс ГРУЗОВИК, добавив в него характеристику грузоподъемности кузова типа int.

1. Создать пустой проект.
2. Добавить в проект класс Object.
3. В файл Object.h ввести следующее описание абстрактного класса Object:

```
#pragma once
class Object
{
public:
	Object(void);
public:
	~Object(void);
	virtual void Show()=0;//чисто виртуальная функция

};
```

4. Файл Object.cpp оставить без изменения.
5. Добавить в проект класс Car, унаследованный от класса Object. В классе переопределить метод Show(). Остальные методы остаются такими же, как и в лабораторной работе 4.
6. В файле Car.h должно быть следующее описание класса

```
#pragma once
#include "object.h"
#include <string>
#include <iostream>
using namespace std;
class Car:
	public Object
{
public:
	Car(void);
public:
	virtual ~Car(void);
	void Show();//функция для просмотра атрибутов класса с помощью указателя
	Car(string,int,int);
	Car(const Car&);
	string Get_mark(){return mark;}
	int Get_cyl(){return cyl;}
	int Get_power(){return power;}
	void Set_mark(string);
	void Set_cyl(int);
	void Set_power(int);
	Car& operator=(const Car&);
	friend istream& operator>>(istream&in,Car&c);
	friend ostream& operator<<(ostream&out,const Car&c);
protected:
	string mark;
	int cyl;
	int power;

};
```

7. В файле Car.cpp должно быть следующее определение методов класса

```
#include "Car.h"
```

```
Car::Car(void)
{
	mark="";
	cyl=0;
	power=0;
}
Car::~Car(void)
{
}
Car::Car(string M,int C,int P)
{
	mark=M;
	cyl=C;
	power=P;
}
Car::Car(const Car& car)
{
	mark=car.mark;
	cyl=car.cyl;
	power=car.power;
}
void Car::Set_cyl(int C)
{
	cyl=C;
}
void Car::Set_mark(string M)
{
	mark=M;
}
void Car::Set_power(int P)
{
	power=P;
}
Car& Car::operator=(const Car&c)
{
	if(&c==this)return *this;
	mark=c.mark;
	power=c.power;
	cyl=c.cyl;
	return *this;
}
 istream& operator>>(istream&in,Car&c)
{
	cout<<"\nMark:"; in>>c.mark;
	cout<<"\nPower:";in>>c.power;
	cout<<"\nCyl:";in>>c.cyl;
	return in;
}
	 ostream& operator<<(ostream&out,const Car&c)
{
	out<<"\nMARK : "<<c.mark;
	out<<"\nCYL : "<<c.cyl;
	out<<"\nPOWER : "<<c.power;
	out<<"\n";
	return out;
}
	 void Car::Show()
 {
	cout<<"\nMARK : "<<mark;
	cout<<"\nCYL : "<<cyl;
	cout<<"\nPOWER : "<<power;
	cout<<"\n";
 }
```

8. Добавить в проект файл Lab5_main.cpp и проверить работу класса Car. Для этого нужно создать переменную класса Car, ввести и вывести значения атрибутов этой переменной:

```
#include "Object.h"
#include "Car.h"
#include <string>
#include <iostream>
using namespace std;
void main()
{
	Car a;
	cin>>a;
	cout<<a<<endl;//вывод с помощью перегруженной операции
	Object *p=&a;
	p->Show();//вывод с помощью метода Show() и указателя

}
```

9. Откомпилировать и выполнить программу.
10. Добавить в проект класс Lorry, унаследованный от класса Car. В классе переопределить метод Show(). Остальные методы остаются такими же, как и в лабораторной работе 4.
11. В файле Lorry.h должно быть следующее описание класса

```
#pragma once
#include "car.h"
class Lorry :
	public Car
{
public:
	Lorry(void);
public:
	~Lorry(void);
	void Show();//функция для просмотра атрибутов класса с помощью указателя
	Lorry(string,int,int,int);
	Lorry(const Lorry & );
	int Get_gruz(){return gruz;}
	void Set_Gruz(int);
	Lorry& operator=(const Lorry&);
	friend istream& operator>>(istream&in,Lorry&l);
	friend ostream& operator<<(ostream&out,const Lorry&l);
protected:
	int gruz;

	};
```

12. В файле Lorry.cpp должно быть следующее описание класса

```
#include "Lorry.h"
Lorry::Lorry(void):Car()
{
	gruz=0;
}
Lorry::~Lorry(void)
{
}
Lorry::Lorry(string M, int C, int P, int G):Car(M,C,P)
{
	gruz=G;
}
Lorry::Lorry(const Lorry &L)
{
	mark=L.mark;
	cyl=L.cyl;
	power=L.power;
	gruz=L.gruz;
```

```
}
void Lorry::Set_Gruz(int G)
{
	gruz=G;
}
Lorry& Lorry::operator=(const Lorry&l)
{
	if(&l==this)return *this;
	mark=l.mark;
	power=l.power;
	cyl=l.cyl;
	return *this;
}
 istream& operator>>(istream&in,Lorry&l)
{
	cout<<"\nMark:"; in>>l.mark;
	cout<<"\nPower:";in>>l.power;
	cout<<"\nCyl:";in>>l.cyl;
	cout<<"\nGruz:";in>>l.gruz;
	return in;
}
 ostream& operator<<(ostream&out,const Lorry&l)
{
	out<<"\nMARK : "<<l.mark;
	out<<"\nCYL : "<<l.cyl;
	out<<"\nPOWER : "<<l.power;
	out<<"\nGRUZ : "<<l.gruz;
	out<<"\n";
	return out;
}
  void Lorry::Show()
 {
	cout<<"\nMARK : "<<mark;
	cout<<"\nCYL : "<<cyl;
	cout<<"\nPOWER : "<<power;
	cout<<"\nGRUZ : "<<gruz;
	cout<<"\n";
}
```

13. Проверить работу класса Lorry. Для этого нужно создать переменную класса Lorry, ввести и вывести значения атрибутов этой переменной:

```
#include "Object.h"
#include "Car.h"
#include <string>
#include <iostream>
using namespace std;
void main()
{
. . . . .
	Lorry b;
	cin>>b;
	cout<<b<<endl;//вывод с помощью перегруженной операции
	p=&b;
	p->Show();//вывод с помощью метода Show() и указателя

}
```

14. Откомпилировать и выполнить программу.
15. Добавить в проект класс Vector, который будет содержать указатели на объекты абстрактного класса Object.
16. В файл Vector.h ввести следующее описание класса Vector:

```
	#pragma once
#include "object.h"
#include <string>
#include <iostream>
```

```
using namespace std;
class Vector
{
public:
	Vector(void);//конструктор без параметров
	Vector(int);//конструктор копирования
public:
	~Vector(void);//деструктор
	void Add(Object *);//добавление элемента в вектор
	friend ostream& operator<<(ostream&out,const Vector&);//операция вывода
private:
	Object**beg;//указатель на первый элемент вектора
	int size;//размер
	int cur;//текущая позиция
};
```

17. В файл Vector.cpp ввести следующее определение методов класса Vector:

```
#include "Vector.h"
//конструктор без параметров
Vector::Vector(void)
{
	beg=0;
	size=0;
	cur=0;
}
//деструктор
Vector::~Vector(void)
{
	if(beg!=0)delete [] beg;
	beg=0;
}
//конструктор с параметрами
Vector::Vector(int n)
{
	beg=new Object*[n];
	cur=0;
	size=n;
}
//добавление объекта, на который указывает указатель p в вектор
void Vector::Add(Object *p)
{
	if(cur<size)
	{
		beg[cur]=p;
		cur++;
	}
}
//операция вывода
 ostream& operator<<(ostream&out,const Vector&v)
{
	if(v.size==0) out<<"Empty"<<endl;
	Object **p=v.beg;//указатель на указатель типа Object
	for(int i=0;i<v.cur;i++)
		{
			(*p)->Show();//вызов метода Show() (позднее связывание)
			p++;//передвигаем указатель на следующий объект
	}

	return out;
}
```

18. Проверить работу класса Vector. Для этого нужно создать переменную класса Vector, добавить в него объекты разных классов из построенной иерархии и вывести значения элементов вектора.

```
#include "Object.h"
#include "Car.h"
#include "Lorry.h"
#include "Vector.h"
#include <string>
#include <iostream>
using namespace std;
void main()
{
	Vector v(5);//вектор из 5 элементов
	Car a;//объект класса Car
	cin>>a;
	Lorry b;// объект класса Lorry
	cin>>b;
	Object*p=&a;//ставим указатель на объект класса Car
	v.Add(p);//добавляем объект в вектор
	p=&b;//ставим указатель на объект класса Lorry
	v.Add(p); //добавляем объект в вектор
	cout<<v;//вывод вектора
}
```

5. Варианты

№	Задание
1	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы изменения полей и сравнения пар (пара p1 больше пары p2, если (p1.first>p2.first) || (p1.first==p2.first &&p1.second>p2.second). Создать производный класс ДРОБЬ (FRACTION), с полями Целая_часть_числа и Дробная_часть_числа. Определить полный набор методов сравнения.
2	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы изменения полей и вычисления произведения чисел. Создать производный класс ПРЯМОУГОЛЬНИК (RECTANGLE), с полями-сторонами. Определить методы для вычисления площади и периметра прямоугольника.
3	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы изменения полей и вычисления произведения чисел. Создать производный класс ПРЯМОУГОЛЬНЫЙ_ТРЕУГОЛЬНИК (RIGHTANGLED), с полями-катетами. Определить метод вычисления гипотенузы.
4	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int

	Второе_число (second) – int Определить методы изменения полей и операцию сложения пар (a,b)+(c,d)=(a+b,c+d) Создать производный класс КОМПЛЕКСНОЕ_ЧИСЛО(COMPLEX), с полями Действительная_часть_числа и Мнимая_часть_числа. Определить операции умножения (a,b)*(c,d)= (a*c-b*d, a*d+b*c) и вычитания (a,b)-(c,d)= (a-b, c-d)
5	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы изменения полей и операцию сложения пар (a,b)+(c,d)=(a+b,c+d) Создать производный класс ДЕНЕЖНАЯ_СУММА(MONEY), с полями Рубли и Копейки. Переопределить операцию сложения и определить операции вычитания и деления денежных сумм.
6	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы изменения полей и операцию сложения пар (a,b)+(c,d)=(a+b,c+d) Создать производный класс ДЛИННОЕ_ЧИСЛО(LONG), с полями Старшая_часть_числа и Младшая_часть_числа. Переопределить операцию сложения и определить операции вычитания и умножения.
7	Базовый класс: ПАРА_ЧИСЕЛ (PAIR) Первое_число (first) - int Второе_число (second) – int Определить методы проверки на равенство и операцию перемножения полей. Реализовать операцию вычитания пар по формуле (a,b)-(c,d)=(a-b,c-d) Создать производный класс ПРОСТАЯ_ДРОБЬ(RATIONAL), с полями Числитель и Знаменатель. Переопределить операцию вычитания и определить операции сложения и умножения простых дробей.
8	Базовый класс: ТРОЙКА_ЧИСЕЛ (TRIAD) Первое_число (first) - int Второе_число (second) – int Третье_число (third) - int Определить методы изменения полей и сравнения триады. Создать производный класс DATE с полями год, месяц и число. Определить полный набор операций сравнения дат.
9	Базовый класс: ТРОЙКА_ЧИСЕЛ (TRIAD) Первое_число (first) - int Второе_число (second) – int Третье_число (third) - int Определить методы изменения полей и сравнения триады. Создать производный класс TIME с полями часы, минуты и секунды. Определить полный набор операций сравнения временных промежутков.
10	Базовый класс: ТРОЙКА_ЧИСЕЛ (TRIAD) Первое_число (first) - int Второе_число (second) – int

	Третье_число (third) - int Определить методы изменения полей и увеличения полей на 1. Создать производный класс DATE с полями год, месяц и число. Переопределить методы увеличения полей на 1 и определить метод увеличения даты на n дней.
11	Базовый класс: ТРОЙКА_ЧИСЕЛ (TRIAD) Первое_число (first) - int Второе_число (second) – int Третье_число (third) - int Определить методы изменения полей и увеличения полей на 1. Создать производный класс TIME с полями часы, минуты и секунды. Переопределить методы увеличения полей на 1 и определить методы увеличения на n секунд и минут.
12	Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int Определить методы изменения полей. Создать производный класс STUDENT, имеющий поле год обучения. Определить методы изменения и увеличения года обучения.
13	Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int Определить методы изменения полей. Создать производный класс EMPLOYEE, имеющий поля Должность – string и Оклад – double. Определить методы изменения полей и вычисления зарплаты сотрудника по формуле Оклад+Премия(% от оклада).
14	Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int Определить методы изменения полей. Создать производный класс TEACHER, имеющий поля Предмет – string и Количество часов – int. Определить методы изменения полей, а также увеличения и уменьшения часов.
15	Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int Определить методы изменения полей. Создать производный класс STUDENT, имеющий поля Предмет – string и Оценка – int. Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

6. Контрольные вопросы

1. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?
2. Какой класс называется абстрактным?
3. Для чего предназначены абстрактные классы?
4. Что такое полиморфные функции?
5. Чем полиморфизм отличается от принципа подстановки?
6. Привести примеры иерархий с использованием абстрактных классов.

7. Привести примеры полиморфных функций.
8. В каких случаях используется механизм позднего связывания?

7. Содержание отчета

1) Постановка задачи (общая и конкретного варианта).
2) Описание класса.
3) Определение компонентных функций.
4) Определение глобальных функций.
5) Функция main().
6) Объяснение результатов работы программы.
7) Ответы на контрольные вопросы.