Лабораторная работа №4 «Полиморфизм и виртуальные методы»

Цель

Научиться создавать абстрактные классы и определять виртуальные методы внутри этих классов, переопределять виртуальные методы внутри классов-наследников, создавать списки из объектов различных классов, понимать механизм позднего связывания.

Теоретическая информация

Виртуальные функции

Виртуальные функции предоставляют механизм **позднего (отложенного) или динамического связывания**. Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используется ключевое слово virtual:

```
virtual void print() {};
```

Понятие позднее связывание означает, что код вызова нужной функции формируется при выполнении программы. Иными словами, в исходном коде вызов функции только обозначается без точного указания того, какая именно функция должна быть вызвана.

Определение какую именно функцию необходимо вызвать происходит на этапе выполнении программы через специальную виртуальную таблицу автоматически.

Виртуальность наследуется. После того как функция определена как виртуальная, ее повторное определение в классе-наследнике (с тем же самым прототипом) создает в этом классе новую виртуальную функцию, причем спецификатор virtual может не использоваться.

Конструкторы не могут быть виртуальными, в отличие от деструкторов. Практически каждый класс, имеющий виртуальную функцию, должен иметь виртуальный деструктор.

Абстрактные классы

Абстрактным называется класс, в котором есть хотя бы одна чистая (пустая) виртуальная функция. Чистой виртуальной функцией называется функция, которая не имеет реализации, вместо этого у нее написана конструкция «= 0»:

```
virtual void print() = 0;
```

Чистая виртуальная функция ничего не делает и недоступна для вызовов. Ее назначение – служить основой для подменяющих ее функций в классах-наследниках. Если в классе-наследнике не переопределить чисто виртуальную функцию, то этот класс также будет считаться абстрактным классом.

Объекты абстрактных классов создавать нельзя. В коде программы можно использовать указатели или ссылками на объекты абстрактных классов, которые инициализировать указателями или ссылками на объекты классов-наследников. Таким образом можно создать массив указателей на базовый класс и инициализировать элементы массива указателями на объекты классов-наследников.

Полиморфизм

Механизм виртуальных функций реализует основополагающий **принцип полиморфизма**: «один интерфейс, несколько реализаций» или «один интерфейс, несколько методов».

Для реализации позднего связывания требуется следующее:

– классы обязаны образовывать иерархию с помощью механизма наследования;
– в иерархии классов были определены функции, имеющие одинаковое имя и список параметров;
– функции с одинаковым именем и параметрами должны быть отмечены как виртуальные (с ключевым словом virtual).

Для демонстрации принципа полиморфизма и позднего связывания можно воспользоваться примером: объявить 3 класса с именами `Article`, `Scientific`, `News`.

Класс `Article` является базовым для двух других классов. Классы содержат один виртуальный метод `print()`.

```
Scientific* sc1 = new Scientific();
Scientific* sc2 = new Scientific("Biology article", "Petrov", "ecology");

News* n1 = new News();
News* n2 = new News("Top news", "Sidorov", "Politics");

Article* arr1[4];           //массив указателей базового класса
arr1[0] = sc1;              //инициализация элементов массива указателями
                           //на объекты классов наследников
arr1[1] = sc2;
arr1[2] = n1;
arr1[3] = n2;

for (int i = 0; i < 4; i++) {
       arr1[i]->print();    //вызов виртуальной функции
}
```

Так как функция `print()` определена как виртуальная, то сработает механизм позднего связывания. Т.е. для элемента массива, который определен как указатель на объект базового класса, будет определено, что он на самом деле является указателем на объект класса-наследника. После этого через специальную виртуальную таблицу будет найдена версия функции, которую необходимо вызвать. В данном случае будут вызваны функции `print()` соответствующих классов-наследников. Если же при объявлении метода `print()` в классе `Article` убрать спецификатор virtual, то полиморфизм поддерживаться не будет, позднее связывание не произойдет и выполнится вызов функции из базового класса для всех элементов массива.

Задание

1. Дополнить и при необходимости модифицировать приложение, разработанное согласно варианта лабораторной работы №3.
2. Базовый класс сделать абстрактным.
3. Определить в базовом классе (он же дополнительный класс из лабораторной работы №1) чисто виртуальную функцию.
4. Определить в классах-наследниках (классы, созданные в лабораторной работе №3) перегрузить виртуальную функцию: реализовать вывод всех атрибутов как базового класса, так и класса-наследника.

5. Модифицировать основной класс таким образом чтоб массив содержащийся в нем мог хранить любые классы-наследники.
6. Дополнить меню программы пунктами для демонстрации и тестирования приложения.
7. Реализовать все необходимые проверки.
8. Сделать выводы.

Пример упрощенного варианта реализации

```
#include "pch.h"
#include <iostream>
#include <string>

using namespace std;

class Article {
	string name;
	string author;
public:
	string getName() {
		return name;
	}
	void setName(string name) {
		this->name = name;
	}
	string getAuthor() {
		return author;
	}
	void setAuthor(string author) {
		this->author = author;
	}
	Article() {
		name = "No name";
		author = "No author";
		cout << "Article: Default constructor" << endl;
	}
	Article(string _name, string _author) {
		name = _name;
		author = _author;
		cout << "Article: Constructor with param" << endl;
	}
	~Article() {
		cout << "Article: Destructor" << endl;
	}

	virtual void print() = 0;		//чисто виртуальный метод
};

class Scientific : public Article {			//класс-наследник 1
	string science;
public:
	string getScience() {
		return science;
	}
	void setScience(string science) {
		this->science = science;
	}
```

```
	Scientific() {
		science = "No science";
	}
	Scientific(string _name, string _author, string science): Article(_name, _author){
		this->science = science;
	}
	~Scientific() {
	}
	void print() {			//метод вывода
		cout << "Scientific -=-=- Name: " << getName() << " Author: "
			<< getAuthor() << " Science: " << getScience() << endl;
	}
};

class News : public Article {				//класс-наследник 2
	string area;
public:
	string getArea() {
		return area;
	}
	void setArea(string area) {
		this->area = area;
	}
	News() {
		area = "No area";
	}
	News(string _name, string _author, string area) {
		this->setName(_name);
		this->setAuthor(_author);
		this->area = area;
	}
	~News() {
	}
	void print() {			//метод вывода
		cout << "News -=-=- Name: " << getName() << " Author: " << getAuthor()
			<< " Area: " << getArea() << endl;
	}
};

class Newspaper {		//основной класс
	string name;
	Article* arr[4];
public:
	string getName() {
		return name;
	}
	void setName(string name) {
		this->name = name;
	}
	void setArticle(Article* art, int i) {
		arr[i] = art;
	}
	Article* getArticle(int i) {
		return arr[i];
	}
	void show() {		//метод вывода
		cout << "-=-=- Newspaper -=-=- Name: " << getName() << endl;
		for (int i = 0; i < 4; i++) {
			arr[i]->print();
		}
	}
};


int main()
```

```
{
	cout << "-=-=-=-=-=-= Step1" << endl;
	Scientific* sc1 = new Scientific();
	Scientific* sc2 = new Scientific("Biology article", "Petrov", "ecology");

	News* n1 = new News();
	News* n2 = new News("Top news", "Sidorov", "Politics");

	cout << "-=-=-=-=-=-= Step2" << endl;
	Article* arr1[4];		//массив указателей базового класса
	arr1[0] = sc1;
	arr1[1] = sc2;
	arr1[2] = n1;
	arr1[3] = n2;

	for (int i = 0; i < 4; i++) {
		arr1[i]->print();
	}

	cout << "-=-=-=-=-=-= Step3" << endl;
	Newspaper np;			//объект основного класса
	np.setName("Vestnik");
	for (int i = 0; i < 4; i++) {
		np.setArticle(arr1[i], i);
	}
	np.show();

	delete sc1;
	delete sc2;
	delete n1;
	delete n2;

}
```

Варианты

Варианты распределяются аналогично предыдущим лабораторным работам.

Итоговый отчет

Укажите в отчете:

1. Тему лабораторной работы, свою фамилию и имя, группу, вариант задания;
2. Листинг программы;
3. Скриншоты результатов выполнения программы;
4. Краткие пояснения к алгоритму работы;
5. Выводы.

Контрольные вопросы

1. Как описать абстрактный класс?
2. Для чего применяют виртуальные функции?
3. Что такое полиморфизм?
4. Что произойдет, если в классе-наследнике не переопределить чисто виртуальную функцию?
5. Можно ли создавать списки/массивы объектов абстрактного класса?