Лабораторная работа №3 «Наследование»

Цель

Научиться создавать иерархию классов, создавать объекты базовых классов и классов-наследников, вызывать методы базового класса и класса-наследника, понимать влияние модификаторов доступа при наследовании, понимать порядок вызова конструкторов и деструкторов при наследовании.

Теоретическая информация

Наследование

Понятие наследования является фундаментальным понятием объектно-ориентированного подхода. Фактически – это отношение, которое связывает два класса, один из которых выступает в роли обобщения для другого. При это отпадает необходимость дублировать в обоих классов атрибуты и методы, которые мы выносим в обобщенный класс. В более распространенной терминологии обобщенный класс называют суперклассом или родительским классом, а детализированный класс – классом-наследником, подклассом, производным или дочерним классом.

Синтаксис определения класса-наследника:

```
class Scientific: public Article {       //класс-наследник
}
```

Производный класс наследует описание родительского класса. Но он может быть изменен путем добавления новых атрибутов и методов, изменением кода существующих методов и изменением прав доступа.

В иерархии классов соглашение относительно доступности членов класса следующее:

– private – член класса может использоваться только функциями данного класса и функциями-“друзьями” своего класса. В классе-наследнике он недоступен.

– protected – то же, что и private, но дополнительно член класса с данным модификатором доступа может использоваться функциями и функциями-“друзьями” классов-наследников.
– public – член класса может использоваться любой функцией, которая является членом данного класса или класса-наследника, а также к public-членам возможен доступ извне через имя объекта.

Конструкторы и деструкторы классов-наследников

При создании объекта класса-наследника вызывается цепочка вызовов конструкторов, начиная с базового класса. Конструктор базового класса вызывается автоматически и выполняется до конструктора класса-наследника. Параметры конструктора базового класса указываются в определении конструктора класса-наследника. Таким образом происходит передача аргументов от конструктора класса-наследника конструктору базового класса.

```
Scientific(string _name, string _author, string science): Article(_name, _author){
            this->science = science;
            cout << "Scientific: Constructor with param" << endl;
      }
```

Уничтожаются объекты в обратном порядке: сначала наследник, потом его атрибуты-объекты, а потом базовый объект. Т.е. происходит цепочка вызовов деструкторов, начиная с класса-наследника.

Задание

1. Дополнить и при необходимости модифицировать приложение, разработанное согласно варианта лабораторной работы №2.
2. Создать несколько классов-наследников для дополнительного класса (т.е. дополнительный класс станет базовым классом для новых создаваемых классов). Каждый класс-наследник должен включать в себя атрибуты, отличающие его от других классов.
3. Определить в новых классах деструктор и конструкторы(по умолчанию, с параметрами, копирования).

4. В каждом конструкторе и деструкторе выдавать сообщение, показывающее, какой именно конструктор или деструктор и какого класса был вызван.
5. Определить в новых классах методы для ввода и вывода атрибутов как самого класса-наследника, так и базового класса. Метод вывода должен иметь то же имя, что и метод вывода в базовом классе.
6. Дополнить меню в основном теле программы тем же функционал что и в лабораторной работе №1, только теперь для классов наследников.
7. Модифицировать и, при необходимости, дополнить основной класс функционалом для добавления, удаления, просмотра и редактирования классов-наследников в массиве объектов (тип массива остается базовый класс).
8. Реализовать все необходимые проверки на вводимые данные.
9. 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(Article &art) {
		name = art.name;
		author = art.author;
		cout << "Article: Copy constructor" << endl;
	}
	~Article() {
		cout << "Article: Destructor" << endl;
	}
	void print() {			//метод вывода
		cout << "Article -=-=- Name: " << getName() << " Author: "
						<< getAuthor() << endl;
	}
};

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";
		cout << "Scientific: Default constructor" << endl;
	}
	Scientific(string _name, string _author, string science): Article(_name, _author){
		this->science = science;
		cout << "Scientific: Constructor with param" << endl;
	}
	~Scientific() {
		cout << "Scientific: Destructor" << endl;
	}
	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";
		cout << "News: Default constructor" << endl;
	}
	News(string _name, string _author, string area) {
		this->setName(_name);
		this->setAuthor(_author);
		this->area = area;
		cout << "News: Constructor with param" << endl;
	}
	~News() {
		cout << "News: Destructor" << endl;
	}
	void print() {			//метод вывода
		cout << "News -=-=- Name: " << getName() << " Author: " << getAuthor()
```

```
                        << " Area: " << getArea() << endl;
        }
};

int main()
{
        cout << "-=-=-=-=-=-= Step1" << endl;
        Article art1;
        Article art2("First article", "Ivanov");

        Scientific sc1;
        Scientific sc2("Biology article", "Petrov", "ecology");

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

        cout << "-=-=-=-=-=-= Step2" << endl;
        Article arr1[6];            //массив базового класса
        arr1[0] = art1;
        arr1[1] = art2;
        arr1[2] = sc1;
        arr1[3] = sc2;
        arr1[4] = n1;
        arr1[5] = n2;

        for (int i = 0; i < 6; i++) {
                arr1[i].print();
        }

        cout << "-=-=-=-=-=-= Step3" << endl;
        Scientific arr2[2];         //массив наследников
        arr2[0] = sc1;
        arr2[1] = sc2;

        for (int i = 0; i < 2; i++) {
                arr2[i].print();
        }
}
```

Варианты

Вариант 1

Тема проекта: приложение «Журнал регистрации корреспонденции».

Добавить классы: электронное письмо и почтовое письмо.

Вариант 2

Тема проекта: приложение «Учет денежных средств».

Добавить классы: Доходы, Расходы.

Вариант 3

Тема проекта: приложение «Энциклопедия «История в лицах».

Добавить классы: Живописцы, Писатели.

Вариант 4
Тема проекта: приложение «Касса кинотеатра».

Добавить классы: Бронирование, Оплаченные билеты.

Вариант 5
Тема проекта: приложение «Календарь планирования мероприятий».

Добавить классы: Разовое мероприятие, Повторяющееся мероприятие.

Вариант 6
Тема проекта: приложение «Каталог кинофильмов».

Добавить классы: Избранное, Заблокированное

Вариант 7
Тема проекта: приложение «Библиотека».

Добавить классы: Детская книга, Комикс.

Вариант 8
Тема проекта: приложение «Отдел кадров».

Добавить классы: Сотрудники, Уволенные.

Вариант 9
Тема проекта: приложение «Управление пользователями».

Добавить классы: Администратор, Гость.

Вариант 10
Тема проекта: приложение «Расписание занятий».

Добавить классы: Лабораторная работа, Лекция.

Вариант 11
Тема проекта: приложение «Журнал учителя».

Добавить классы: Зачет, Экзамен.

Вариант 12
Тема проекта: приложение «Учет клиентов».

Добавить классы: Физические лица, Юридические лица.

Вариант 13
Тема проекта: приложение «Тестирование знаний».

Добавить классы: Тест по теме, Контрольный тест.

Вариант 14
Тема проекта: приложение «Организация экскурсий».

Добавить классы: Местные экскурсии, Международные экскурсии.

Вариант 15
Тема проекта: приложение «Телефонный справочник».

Добавить классы: Юридическое лицо, Физическое лицо.

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 создать иерархию классов?
2. Как влияют модификаторы доступа на видимость атрибутов и методов при наследовании?
3. Можно ли обращаться к методам базового класса через объекты класса-наследника?
4. В каком порядке вызываются конструкторы и деструкторы при наследовании?
5. Можно ли поместить объекты базового класса и классов-наследников в один массив?