Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

**Лабораторная работа №2**

по дисциплине

«Конструирование ПО»

|  |  |
| --- | --- |
|  | Выполнил студент группы 181073  Наркевич Михаил Викторович |
|  | Проверил Сицко Владимир Александрович |

Минск 2022

**Лабораторная работа №2**

**Тема работы:** Виртуальные функции и абстрактные классы.

**Цель работы:** Создание консольной программы для реализации абстрактных классов и чисто виртуальных функций, создания и наследования виртуальных функций, изучения статического и динамического полиморфизма на основе виртуальных функций, включая виртуальные деструкторы.

**Общие требования к выполнению работы**

1.     Рассмотреть теоретические сведения по теме лабораторной работы.

2.     Создать проект консольной программы.

3.     По полученному базовому классу создать классы наследников по двум разным ветвям наследования (B←P1←P11 и B←P2):

-- во всех классах должны быть свои данные (характеристики объектов), расположенные в private или protected секциях;

-- во всех классах создать конструкторы инициализации объектов для всех классов (не забыть про передачу параметров в конструкции базовых классов);

-- во всех классах создать деструкторы;

-- остальные методы создавать по необходимости.

4.     Создать в базовом классе чисто виртуальные функции расчета (например, расчет площади фигуры и т.п.) и вывода объекта на экран (всех его параметров). Выполнить реализацию этих виртуальных функций в классах наследниках.

5.     Задать в базовом классе деструктор как виртуальный.

6.     В головной функции динамически создать массив указателей на базовый класс. Заполнить массив указателями на динамически создаваемые объекты производных классов (P1, P11, P2). Для каждого элемента массива проверить работу виртуальных функций. Удалить из памяти динамически выделенные объекты.

7.     Отладить и выполнить полученную программу. Проверить, что будет, если выше описанные методы не будут виртуальными.

8.     Создать отчет, включающий задание, листинг программы и результаты ее выполнения.

Ход работы:

Тема: «Книга».

Код программы:

#include <iostream>

#include <string>

using namespace std;

class Book {

protected:

string \_isbnNumber;

int \_numberOfPages;

public:

Book() {

\_isbnNumber = "325-35fwe5-23-543";

\_numberOfPages = 100;

}

Book(string isbnNumber, int numberOfPages) {

\_isbnNumber = isbnNumber;

\_numberOfPages = numberOfPages;

}

virtual double GetPrice() = 0;

virtual void GetInfo() = 0;

virtual ~Book() {

cout << "Cработал виртуальный деструктор класса Book" << endl;

}

};

class Treatise : public Book {

protected:

string \_theme;

public:

Treatise() {

\_theme = "someTheme";

}

Treatise(string isbnNumber, int numberOfPages, string theme) : \_theme(theme) {

\_isbnNumber = isbnNumber;

\_numberOfPages = numberOfPages;

}

double GetPrice() {

return \_numberOfPages \* 0.2;

}

void GetInfo() {

cout << "Научный труд с номером " << \_isbnNumber << " и количеством страниц " << \_numberOfPages << " рассматривающий тему " << \_theme << ".Стоимость:" << GetPrice() << endl;

}

~Treatise() {

cout << "Cработал деструктор класса Treatise" << endl;

}

};

class LightNovel : public Book {

protected:

int \_audienceAge;

public:

LightNovel() {

\_audienceAge = 300;

}

LightNovel(string isbnNumber, int numberOfPages, int audienceAge) : \_audienceAge(audienceAge) {

\_isbnNumber = isbnNumber;

\_numberOfPages = numberOfPages;

}

double GetPrice() {

return (\_numberOfPages \* 0.2) / 0.8;

}

void GetInfo() {

cout << "Ранобэ с номером " << \_isbnNumber << " и количеством страниц " << \_numberOfPages << " .Как и другие романы, но написанное для людей лет " << \_audienceAge << ".Стоимость:" << GetPrice() << endl;

}

~LightNovel() {

cout << "Cработал деструктор класса LightNovel" << endl;

}

};

class Manga : public LightNovel {

protected:

int \_numberOfImages;

public:

Manga() {

\_numberOfImages = 1200;

}

Manga(string isbnNumber, int numberOfPages, int audienceAge, int numberOfImages) : \_numberOfImages(numberOfImages) {

\_isbnNumber = isbnNumber;

\_numberOfPages = numberOfPages;

\_audienceAge = audienceAge;

}

double GetPrice() {

return ((\_numberOfPages \* 0.2) / 0.8) + \_numberOfImages \* 0.3;

}

void GetInfo() {

cout << "Манга с номером " << \_isbnNumber << " и количеством страниц " << \_numberOfPages << ". Для аудитории лет " << \_audienceAge

<< "." << endl << "Считай что ранобэ, но с кучей илюстраций, примерно " << \_numberOfImages << ".Стоимость:" << GetPrice() << endl;

}

~Manga() {

cout << "Cработал деструктор класса Manga" << endl;

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

Book\*\* arr = new Book \* [3];

arr[0] = new Treatise("hdaju-42klf-42dag", 600, "образование вселенной");

arr[1] = new LightNovel("4gdu-65l4f-21dadg", 200, 16);

arr[2] = new Manga("fgj9-02fsf-75fgd", 350, 16, 1200);

arr[0]->GetInfo();

arr[1]->GetInfo();

arr[2]->GetInfo();

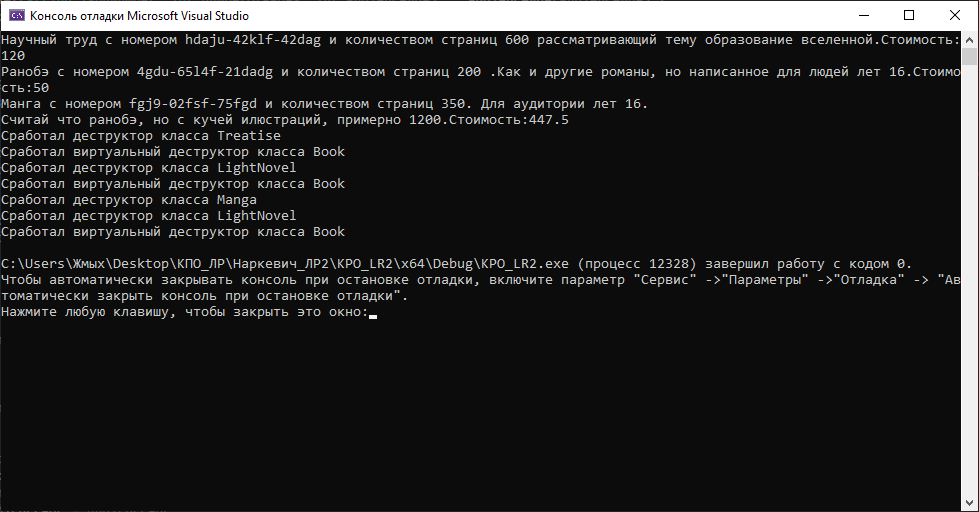
delete arr[0];

delete arr[1];

delete arr[2];

}

Скриншот:



Контрольные вопросы:

1.     Виртуальные функции.

Виртуальная функция в языке С++ — это особый тип функции, которая,

при её вызове, выполняет «наиболее» дочерний метод, который существует

между родительским и дочерними классами. Это свойство еще известно, как полиморфизм. Дочерний метод вызывается тогда, когда совпадает сигнатура (имя, типы параметров и является ли метод константным) и тип возврата дочернего метода с сигнатурой и типом возврата метода родительского класса. Такие методы называются переопределениями (или «переопределенными методами»). Чтобы сделать функцию виртуальной, нужно указать ключевое слово virtual перед объявлением функции.

2.     Абстрактные классы.

C++ позволяет создавать особый вид виртуальных функций, так

называемых чистых виртуальных функций (или «абстрактных функций»), которые вообще не имеют определения Любой класс с одной и более чистыми виртуальными функциями становится абстрактным классом, объекты которого создавать нельзя.

3.     Наследование виртуальных функций.

Пример:

class Person

{

public:

    Person(std::string n, int a)

    {

        name = n; age = a;

    }

    virtual void display()

    {

        std::cout << "Name: " << name << "\tAge: " << age << std::endl;

    }

private:

    std::string name;

    int age;

};

class Employee : public Person

{

public:

    Employee(std::string n, int a, std::string c):Person(n, a)

    {

        company = c;

    }

    void display() override

    {

        std::cout << "Name:" << name << "\tCompany: " << company << std::endl;

    }

private:

    std::string company;

};

Здесь видно, как переопределить в классе-наследнике виртуальную

функцию в наследнике.

4.     Виртуальный деструктор.

При работе с наследованием ваши деструкторы всегда должны быть

виртуальными, иначе деструкторы потомков могут быть не вызваны. Например, если в данной лабораторной работе не указать виртуальные деструкторы, то вызовется только деструктор класса Virus.

5.     Статический и динамический полиморфизм.

Полиморфизм может быть статическим или динамическим. В статическом полиморфизме выбор определяется во время компиляции. В динамическом полиморфизме выбор определяется во время выполнения.

Вывод: в ходе проделанной лабораторной работы были изучены и применены на практике особенности работы с переопределнием функций в классках потомка в C++, а также специальное ключевое слово virtual, деструкторы и абстрактные функции.