по лабораторной работе №2 на тему: конструирование простейшего классасредствами языка С++

Вариант 4

г. Санкт-Петербург

2019 г.

**Цель работы:**

Познакомиться с основными понятиями объектно-ориентированного программирования (класс, объект, свойство, метод, конструктор, деструктор), выполнить конструирование класса, предназначенного для хранения заданной структуры данных, изучить способы создания объектов.

**Задание:**

В работе требуется сконструировать класс с заданным набором свойств. Набор свойств следует взять из лабораторной работы № 1 в соответствии со своим вариантом. В класс также должно быть добавлено достаточное количество методов для просмотра и редактирования значений любого из свойств (можно однократное).

Требования к конструированию класса: доступ к свойствам — закрытый, к методам — открытый. В классе следует предусмотреть конструктор по умолчанию, конструктор с параметрами.

**Средаразработки:**

CodeBlocks

**Исходные данные:**

Количество строк: 3.

Столбцы:название товара, категория товара, цена, количество. Вычислить общую стоимость товаров.

**Описание программы:**

Данная программа предназначена для:

1. Создания объекта с помощью конструктора по умолчанию.
2. Редактирования свойств объекта с помощью конструктора с параметрами.
3. Просмотра свойств объекта.
4. Вычисления общей стоимости товаров.

Для реализации программы было использовано тип данных класс.

Класс — это пользовательский тип данных, объединяющий данные и алгоритмы для обработки этих данных. Класс моделирует группу каких-либо реальных объектов (студенты, машины), процессов (путешествия), явлений (погода).

Данные класса представлены в виде переменных и называются свойствами.

Алгоритмы представлены в виде функций и называются методами.

В классе существует разграничение доступа к его членам. Внутреннюю (закрытую) часть класса, доступную только этому классу, составляет раздел private, защищенная часть класса доступна классу и его наследникам — раздел protected, доступны для любых объектов без ограничения члены класса из раздела public (открытая часть класса).

Пример объявления класса:

class Book

{

public:// открытая часть класса

Book(); // Конструктор

~Book(); // Деструктор

void setAuthor(std::string);//Метод, устанавливающий

// новое значение свойству

std::string getAuthor(); // Метод, позволяющий

//прочитать значение свойства

. . .

private:// закрытая часть класса

string author;

string title;

int year; // годиздания

} ;

Данные для их защиты от воздействий извне помещаются в раздел private. Для доступа к таким данным используются методы, которые берут на себя контроль за корректностью использования данных сторонними объектами. Объединение в классе данных и методов с целью защиты данных называется инкапсуляция.

Для хранения текстовых данных в примере использовался стандартный тип string, объявленный в стандартной библиотеке языка С++. Он удобнее и надежнее для использования, чем массив символов в языке С, так как не надо заботиться о распределении памяти, кроме того, для типа string определена операция конкатенации строк +.

В класс можно включить несколько функций с одинаковыми именами, но различающихся списками параметров. Такая возможность основана на свойстве языка C++, называемом перегрузкой функций, а функции при этом называются перегруженными. Каждая перегруженная функция решает свою задачу и имеет свой алгоритм. Выбор функции, которая должна быть вызвана, определяется на этапе компиляции программы в точном соответствии со списком параметров. В данной работе перегруженными являются конструкторы. Использование перегруженных функций в классе представляет собой вариант полиморфизма, а именно, статический полиморфизм.

Объект создается по шаблону, который дает класс, при этом используется специальный метод — конструктор. Имя конструктора совпадает с именем класса, обычно он помещается в разделе public, для него не указывается тип возвращаемого значения.

Конструктор по умолчанию создает объект с не инициализированными свойствами:

Book::Book()

{

}

или всегда с одним и тем же набором значений:

Book::Book()

{

author = "Noname";

title = "Noname";

year = 0;

}

Конструктор с параметрами создает объект с заранее определенным набором свойств:

Book::Book(std::string auth, std:: string ttl, int y)

{

author = auth;

title = ttl;

year = y;

}

Деструктор выполняет разрушение объекта, он не имеет параметров, находится в разделе public, не имеет типа возвращаемого значения, а имя отличается от имени конструктора одним символом знаком ~ (тильда) в начале.

Book::~Book()

{

// Деструктор по умолчанию

}

Не требуется вставлять в программу явный вызов деструктора — он вызывается автоматически. Также деструктор можно не включать в класс — он будет добавлен автоматически компилятором. Деструктор должен быть обязательно включен в класс, если какие-либо данные объекта хранятся в динамической памяти.

В описании конструктора и любого другого метода класса (в реализации) используется оператор разрешения области видимости ::. Этот оператор позволяет включить идентификаторы в заданное пространство имен namespace. Для функций-членов класса пространством имен будет класс. Если не использовать оператор ::, получится глобальная функция, не связанная с классом.

Примеры создания объектов:

Book b1; // Создание объекта с помощью конструктора

// по умолчанию

Book b2("Pushkin","Evgeny Onegin",2003); // Создание

// объекта с помощью конструктора с параметрами

Book b3[3]; // Создание массива объектов с помощью

// конструктора по умолчанию

Для доступа к свойствам и методам в функциях-членах данного класса используется только имя свойства или метода

Book::Book(std::string auth, std::string ttl, int y)

{

author = auth;

. . .

}

Также возможно обратиться к члену того же класса с помощью указателя на текущий объект this

Book::Book(std::string auth, std:: ttl, int y)

{

this->author = auth;

. . .

}

Для доступа сторонних объектов к свойствам и методам объекта какого-либо класса используется имя объекта и операция точка (.)

b1.setAuthor("Gogol");

Сторонние объекты также могут обращаться к свойствам и методам какого-либо объекта с помощью указателя на объект и операции стрелка (>):

Book\* ptr = &b1;

ptr->getAuthor();

Пример простейшей реализации методов класса Book:

void Book::setAuthor(std::string auth)

{

author = auth;

}

std::string Book::getAuthor()

{

returnauthor;

}

**Пример работы программы:**

На приведённых ниже рисунках представлен пример работы программы.

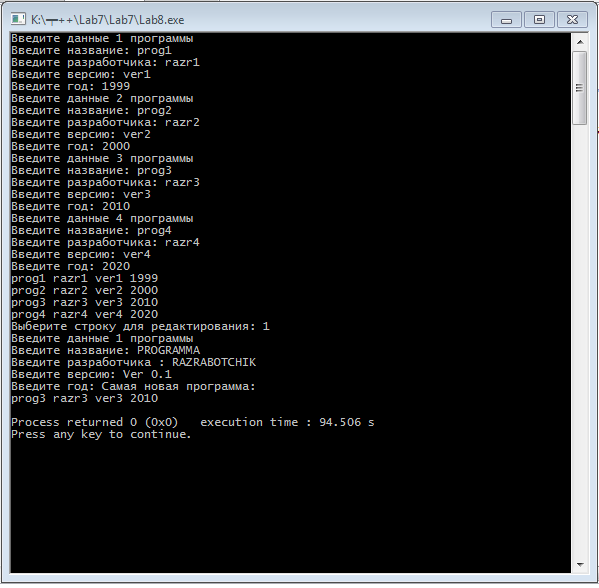


Рисунок 1 – Создание объектов классов с помощью конструктора по умолчанию. Редактирование свойств объекта с помощью конструктора с параметрами и расчёт общей стоимости товаров

**Вывод:**

В результате проделанной работы были получены следующие результаты:

1. Изучено создание и использование классов на основе MVS 2019 для Windows;
2. Разработана программа, выполняющая поиск самой новой программы и предоставляющая возможность просмотра и редактирования свойств объектов классов;
3. Инициализация объектов класса производится с помощью конструкторов по умолчанию, редактирование – с помощью конструкторов с параметрами.

**Код программы**

**#include <iostream>**

**#include <string>**

**#include <windows.h>**

**#include <string>**

**using namespace std;**

**class prog**

**{**

**public:**

**prog() {}//Конструктор по умолчанию создает объект с не инициализированными свойствами**

**prog (string valNazv, string valRazr, string valVer, int valYear) // Конструктор с параметрами создает объект с заранее определенным набором свойств**

**{**

**nazv = valNazv;**

**razr = valRazr;**

**ver = valVer;**

**year = valYear;**

**}**

**void set\_nazv(string valNazv)**

**{**

**nazv = valNazv;**

**}**

**void set\_razr(string valRazr)**

**{**

**razr = valRazr;**

**}**

**void set\_ver(string valVer)**

**{**

**ver = valVer;**

**}**

**void set\_year(int valYear)**

**{**

**year = valYear;**

**}**

**string get\_nazv()**

**{**

**return nazv;**

**}**

**string get\_razr()**

**{**

**return razr;**

**}**

**string get\_ver()**

**{**

**return ver;**

**}**

**int get\_year()**

**{**

**return year;**

**}**

**private :**

**string nazv;**

**string razr;**

**string ver;**

**int year;**

**};**

**int main()**

**{**

**setlocale(LC\_ALL, "Russian");**

**prog programm[4];**

**for (int i = 0; i <= 3; i++)**

**{**

**string nazv, razr, ver;**

**int year;**

**cout << "Введите данные " << i + 1 << " программы" << endl;**

**cout << "Введите название: ";**

**cin >> nazv;**

**programm[i].set\_nazv(nazv);**

**cout << "Введите разработчика: ";**

**cin >> razr;**

**programm[i].set\_razr(razr);**

**cout << "Введите версию: ";**

**cin >> ver;**

**programm[i].set\_ver(ver);**

**cout << "Введите год: ";**

**cin >> year;**

**programm[i].set\_year(year);**

**}**

**for (int i = 0; i <= 3; i++)**

**{**

**cout << programm[i].get\_nazv() + " ";**

**cout << programm[i].get\_razr() + " ";**

**cout << programm[i].get\_ver() + " ";**

**cout << programm[i].get\_year() << " " << endl;**

**}**

**cout << "Выберите строку для редактирования: ";**

**int red;**

**cin >> red;**

**for (int i = 0; i < 1; i++)**

**{**

**string nazv, razr, ver;**

**int year;**

**cout << "Введите данные " << red << " программы" << endl;**

**cout << "Введите название: ";**

**cin >> nazv;**

**programm [red - 1].set\_nazv(nazv);**

**cout << "Введите разработчика : ";**

**cin >> razr;**

**programm [red - 1].set\_razr(razr);**

**cout << "Введите версию: ";**

**cin >> ver;**

**programm [red - 1].set\_ver(ver);**

**cout << "Введите год: ";**

**cin >> year;**

**programm [red - 1].set\_year(year);**

**}**

**int last\_prog = programm[0].get\_year();**

**int string = 0;**

**for (int i = 0; i < 3; i++)**

**{**

**if (last\_prog < programm[i].get\_year())**

**{**

**last\_prog = programm[i].get\_year();**

**string = i;**

**}**

**}**

**cout << "Самая новая программа: " << endl;**

**cout << programm[string].get\_nazv() + " ";**

**cout << programm[string].get\_razr() + " ";**

**cout << programm[string].get\_ver() + " ";**

**cout << programm[string].get\_year() << " " << endl;**

**return 0;**

**}**