**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По лабораторной работе №18.1 на тему

«Конструкторы и деструкторы»

Вариант №11

Выполнил студент группы ИВТ-20-2б

Сабуров Павел Алексеевич

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –изучить таких элементы классов, как:

* Конструктор и его виды;
* Деструктор;

**Постановка задачи**

Задача – реализовать на языке программирования C++ программу, демонстрирующую работу конструкторов и деструкторов класса.

**Исходные данные для варианта №11:**

* Реализовать три вида конструкторов:
  + Конструктор без параметров;
  + Конструктор с параметрами;
  + Конструктор копирования;
* Реализовать деструктор;

Класс «автомобиль»

* Брэнд;
* Модель;
* Цена;

**Анализ задачи**

Для решения задачи были использованы следующие средства:

1. Язык программирования C++ (Microsoft Visual C++)
2. Текстовый редактор Microsoft Visual Studio Code

**Конструктор:**

Конструктор класса – особый метод, отвечающий за инициализацию экземпляра класса в программе, вызывается при объявлении нового экземпляра класса.

Класс «автомобиль» реализован следующим образом:

class Auto

{

private:

string \_brand;

string \_model;

int \_cost;

}

Get-теры и Set-теры определены следующим образом, реализована проверка на корректность присваиваемых данных:

void SetBrand(string brand)

{

if (brand.size() > 1)

\_brand = brand;

else

cout << "Error: invalid brand;\n";

}

string Model() { return \_model; }

void SetModel(string model)

{

if (model.size() > 0 && model != \_brand)

\_model = model;

else

cout << "Error: invalid model;\n";

}

int Cost() { return \_cost; }

void SetCost(int cost)

{

if (cost > 0)

\_cost = cost;

else

cout << "Error: invalid cost;\n";

}

Вывод данных осуществляется следующим методом:

void PrintData()

{

cout << "\nAutomobile:\n";

cout << " - Brand: " << \_brand << ";\n";

cout << " - Model: " << \_model << ";\n";

cout << " - Cost: " << \_cost << "$;\n";

}

**Реализация конструкторов:**

Конструктор без параметров – пустой конструктор:

Такой конструктор является стандартным, если программист не реализовал ни один конструктор, то компилятор добавит пустой конструктор в класс самостоятельно, так как каждый класс должен иметь хотя бы один конструктор.

Как правило, конструкторы без параметров представляют собой конструкторы, которые присваивают полям базовые, нулевые, значения, чтобы с ними можно было производить работу.

Реализация конструктора без параметров:

Auto()

{

\_brand = "unknown";

\_model = "not famous";

\_cost = 1;

}

Конструктор с параметрами, параметризованный конструктор:

Данный конструктор предназначен для инициализации данных прямо при объявлении экземпляра класса. Такой конструктор значительно уменьшает объём написанного кода;

Реализация конструктора с параметрами:

Auto(string brand, string model, int cost)

{

SetBrand(brand);

SetModel(model);

SetCost(cost);

}

Конструктор копирования:

Конструктор копирования предназначен для инициализации клона некоторого экземпляра класса в новом экземпляре класса. Как и конструктор копирования, позволяет значительно сократить объём вводимого кода.

Реализация конструктора копирования:

Auto(Auto& parentAuto)

{

\_brand = parentAuto.\_brand;

\_model = parentAuto.\_model;

\_cost = parentAuto.\_cost;

}

**Деструктор:**

Деструктор – это специальный метод, который отвечает за очистку памяти во время уничтожения экземпляра класса. Если в классе нет данных, объявленных в куче, деструктор остаётся пустым. Если деструктор не объявлен явно, то компилятор добавит его в класс самостоятельно.

В данном примере нет данных, объявленных в куче, соответственно деструктор остаётся пустым:

~Auto() {}

Главный метод – метод main, который демонстрирует работы конструкторов класса:

int main()

{

Auto firstAuto;

firstAuto.PrintData();

Auto secondAuto("Jeep", "Cherokee", 30000);

secondAuto.PrintData();

Auto thirdAuto(secondAuto);

thirdAuto.PrintData();

return 0;

}

**Полный исходный код программы на языке программирования C++:**

#include <iostream>

#include <string>

using namespace std;

class Auto

{

private:

string \_brand;

string \_model;

int \_cost;

public:

string Brand() { return \_brand; }

void SetBrand(string brand)

{

if (brand.size() > 1)

\_brand = brand;

else

cout << "Error: invalid brand;\n";

}

string Model() { return \_model; }

void SetModel(string model)

{

if (model.size() > 0 && model != \_brand)

\_model = model;

else

cout << "Error: invalid model;\n";

}

int Cost() { return \_cost; }

void SetCost(int cost)

{

if (cost > 0)

\_cost = cost;

else

cout << "Error: invalid cost;\n";

}

void PrintData()

{

cout << "\nAutomobile:\n";

cout << " - Brand: " << \_brand << ";\n";

cout << " - Model: " << \_model << ";\n";

cout << " - Cost: " << \_cost << "$;\n";

}

Auto()

{

\_brand = "unknown";

\_model = "not famous";

\_cost = 1;

}

Auto(string brand, string model, int cost)

{

SetBrand(brand);

SetModel(model);

SetCost(cost);

}

Auto(Auto& parentAuto)

{

\_brand = parentAuto.\_brand;

\_model = parentAuto.\_model;

\_cost = parentAuto.\_cost;

}

~Auto()

{

}

};

int main()

{

Auto firstAuto;

firstAuto.PrintData();

Auto secondAuto("Jeep", "Cherokee", 30000);

secondAuto.PrintData();

Auto thirdAuto(secondAuto);

thirdAuto.PrintData();

return 0;

}

**Ответы на вопросы:**

**Для чего нужен конструктор?**

Конструктор необходим для инициализации полей класса при его объявлении и инициализации самого класса;

**Сколько типов конструкторов существует в C++?**

C++ поддерживает 3 типа конструкторов:

1. Конструктор без параметров;
2. Конструктор с параметрами;
3. Конструктор копирования;

**Для чего используется деструктор? В каких случаях деструктор описывается явно?**

Деструктор используется для очистки динамической памяти при удалении класса, описывается явно, когда требуется удалять динамическую память.

**Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?**

Конструктор без параметров нужен для присвоения полям «нулевого значения», конструктор с параметрами необходим для начального заполнения полей пользовательскими значениями данных, конструктор копирования используется для порождения клонов некоторого экземпляра класса.

**В каких случаях вызывается конструктор копирования?**

В случаях, когда нужно получить клон объекта.

**Перечислить свойства конструкторов:**

1. Конструктор имеет то же имя, что и класс;
2. Конструктор не возвращает значение, даже void;

**Перечислить свойства деструкторов:**

1. Деструктор имеет то же имя, что и класс, со знаком «тильда» (~) в самом начале;
2. Деструктор не принимает аргументы;
3. Деструктор не возвращает значение, даже void.

**К каким атрибутам имеют доступ методы класса?**

К любым.

**Что представляет собой указатель this?**

this – это указатель на экземпляр класса, точнее на саму себя.

**Какая разница между методами, определёнными внутри класса и вне класса?**

Разница заключается в способе объявления:

void Open() { /\* \*/ }

void FileProcessor::Open() { /\* \*/ }

**Какое значение возвращает конструктор?**

НИКАКОЕ!!!!!!!!

**Какие методы создаются по умолчанию?**

1. Конструктор без параметров;
2. Деструктор;

**Какое значение возвращает деструктор?**

НИКАКОЕ!!!!!!!!

**Дано описание класса:**

class student {

int group;

string name;

student(string, int);

student(const student&);

~student();

};

**Какой метод отсутствует в описании класса?**

Конструктор без параметров.

**Какой метод будет вызван при выполнении следующих операторов?**

student\* s;

s = new student;

Вызовется конструктор без параметров.

**Какой метод будет вызван при выполнении следующих операторов?**

Student s(“Ivatov”, 20);

Вызовется конструктор с параметрами.

**Какой конструктор будет вызван при выполнении следующих операторов?**

void print(student a) { a.show() };

Вызовется конструктор копирования.

Класс описан следующим образом:

class student {

string name;

int age;

public:

void set\_name(string);

void set\_age(int);

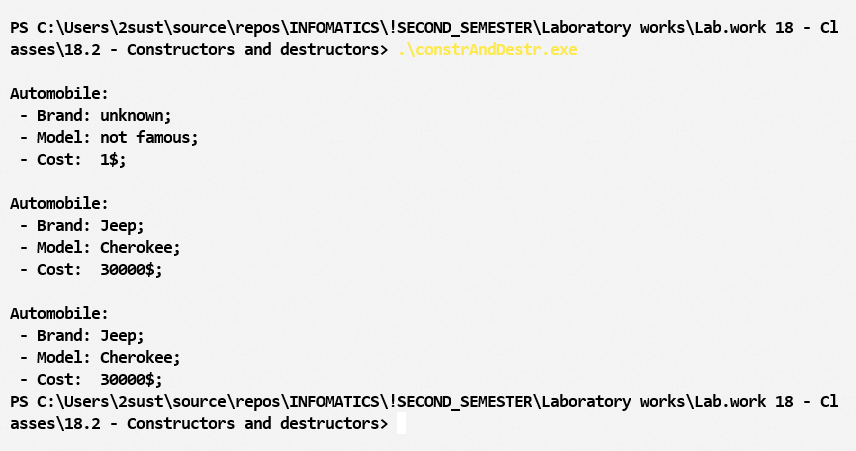
};

Student p;

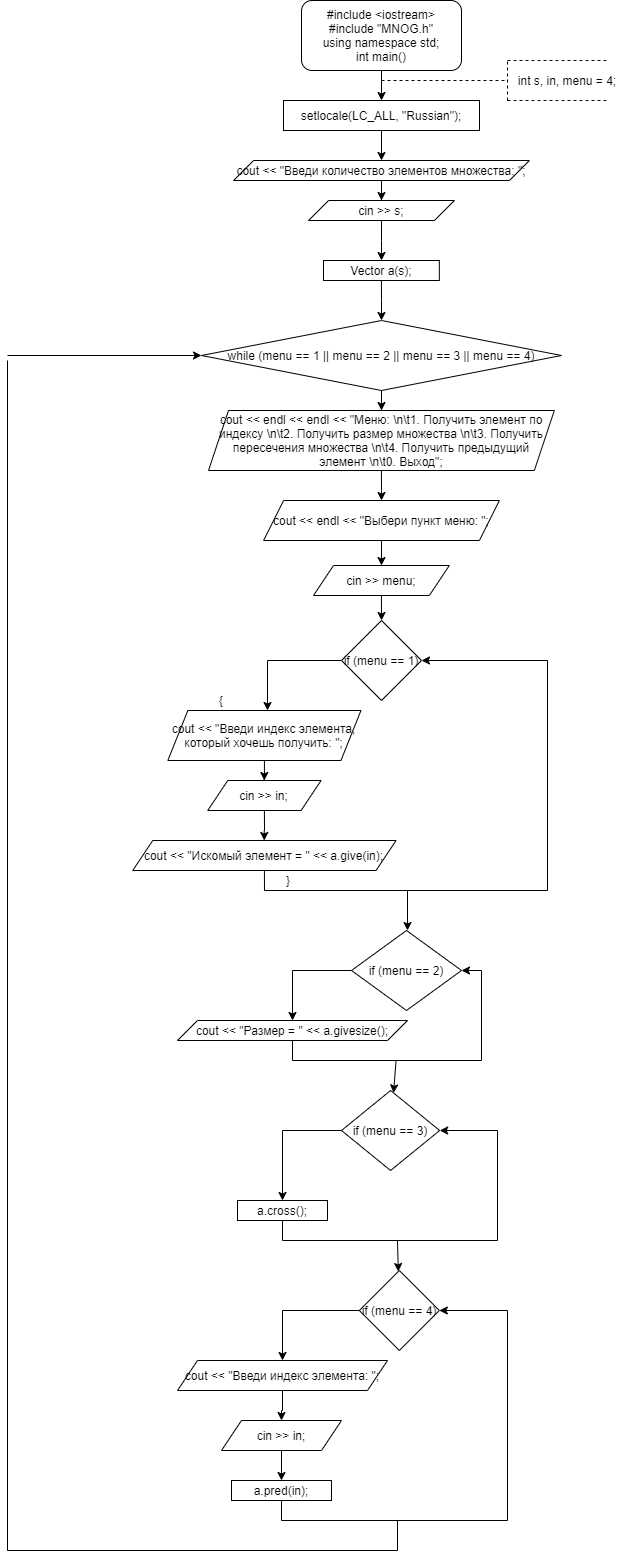
**Каким образом можно присвоить новое значение атрибуту name объекта p?**

Через метод set\_name, потому что он имеет модификатор доступа public.

Скриншот успешно выполненной программы:



Блок-схема



Vector.cpp:

