Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 - «Программная инженерия»

**«Лабораторная работа №2**

**"Классы и объекты. Использование конструкторов"»**

Выполнил студент гр. РИС-24-3б Носков Егор Михайлович

Проверил: Доцент кафедры ИТАС Ольга Андреевна Полякова

|  |  |
| --- | --- |
|  | |
| (оценка) (подпись) | |
|  |
| (дата) |

Г. Пермь, 2024

**Постановка задачи**

**Вариант 1:**



**Программа на языке C++**

#include <iostream>

#include <string>

#include <clocale>

#include <limits> // Для игнорирования остатка строки после ввода числа

using namespace std;

class Student {

    string fio;

    string gruppa;

    float srball;

public:

    Student() {

        fio = "-";

        gruppa = "-";

        srball = 0;

        cout << "Конструктор без параметров для объекта " << this << endl;

    }

    Student(string F, string G, float S) {

        fio = F;

        gruppa = G;

        srball = S;

        cout << "Конструктор с параметрами для объекта " << this << endl;

    }

    Student(const Student& student) {

        fio = student.fio;

        gruppa = student.gruppa;

        srball = student.srball;

        cout << "Конструктор копирования для объекта " << this << endl;

    }

    ~Student() {

        cout << "Деструктор для объекта " << this << endl;

    }

    string get\_fio() {

        return fio;

    }

    void set\_fio(string F) {

        fio = F;

    }

    string get\_gruppa() {

        return gruppa;

    }

    void set\_gruppa(string G) {

        gruppa = G;

    }

    float get\_srball() {

        return srball;

    }

    void set\_srball(float S) {

        srball = S;

    }

    void show() {

        cout << "ФИО: " << fio << endl;

        cout << "Группа: " << gruppa << endl;

        cout << "Средний балл: " << srball << endl;

    }

};

Student Make\_Student() {

    string fio, gruppa;

    float srball;

    cout << "Введите ФИО" << endl;

    getline(cin, fio);

    cout << "Введите группу" << endl;

    getline(cin, gruppa); // Исправлено: использована правильная переменная

    cout << "Введите средний балл" << endl;

    cin >> srball;

    cin.ignore(numeric\_limits<streamsize>::max(), '\n');

    Student student(fio, gruppa, srball);

    return student;

}

void Print\_Student(Student student) {

    student.show();

}

int main() {

    setlocale(LC\_ALL, "RU");

    system("chcp 1251");

    system("cls");

    Student student1;

    student1.show();

    Student student2("Носков Егор Михайлович", "РИС-24-3Б", 2.33);

    student2.show();

    Student student3 = student1;

    student3.set\_fio("Александров Александр Александрович");

    student3.set\_gruppa("ИВТ-24-1");

    student3.set\_srball(3.44);

    Print\_Student(student3);

    student1 = Make\_Student();

    student1.show();

    return 0;

}

**Блок-схема**

**Контрольные вопросы**  
  
1. Конструктор нужен для инициализации объекта класса при его создании. Он обеспечивает, что объект находится в согласованном и валидном состоянии с самого начала.  
  
2. Основные типы:  
• Конструктор по умолчанию (без параметров)  
• Конструктор с параметрами  
• Конструктор копирования  
• Конструктор перемещения (C++11 и новее)  
  
3. Деструктор используется для освобождения ресурсов, захваченных объектом в течение его жизненного цикла (например, освобождение памяти, закрытие файлов). Он вызывается автоматически при уничтожении объекта. Явно описывается, когда классу требуются специальные действия при уничтожении объекта, например, если класс управляет динамической памятью.  
  
4. • Конструктор без параметров (по умолчанию): Создаёт объект, используя значения по умолчанию для его членов.  
• Конструктор копирования: Создаёт новый объект как копию существующего объекта того же класса.  
• Конструктор с параметрами: Создаёт объект, инициализируя его члены значениями, переданными в качестве аргументов.  
  
5. Конструктор копирования вызывается в следующих случаях:  
• Когда объект передается в функцию *по значению*.  
• Когда объект возвращается из функции *по значению*.  
• Когда объект инициализируется другим объектом того же класса (например, Student s2 = s1;).  
• Когда объект создается с использованием конструктора копирования явно (например, Student s2(s1);).  
  
6. • Имя конструктора совпадает с именем класса.  
• Конструктор не возвращает значения (даже void).  
• Может быть перегружен (иметь несколько конструкторов с разными параметрами).  
• Если не определён явно, компилятор генерирует конструктор по умолчанию (если нет других конструкторов) и конструктор копирования.  
• Может иметь список инициализации членов (рекомендуется).  
  
7. Имя деструктора совпадает с именем класса, предваряемым символом тильды (~).  
• Деструктор не принимает аргументов.  
• Деструктор не возвращает значения (даже void).  
• В классе может быть только один деструктор.  
• Если не определён явно, компилятор генерирует деструктор по умолчанию.  
• Вызывается автоматически при уничтожении объекта.  
  
8. Методы класса имеют доступ ко *всем* атрибутам класса, независимо от спецификатора доступа (private, protected, public). (Однако, доступ может быть ограничен, если используется наследование.)  
  
9. this - это неявный указатель, который передается каждому нестатическому методу класса. Он указывает на объект, для которого был вызван метод. Он позволяет методу обращаться к членам данного объекта.  
  
10. *Методы, определенные* ***внутри****класса, обычно компилируются как* встроенные (inline)\*, если компилятор считает это уместным (но это не гарантируется).  
\* Методы, определенные **вне** класса, должны указывать принадлежность к классу с использованием оператора разрешения области видимости (::).  
\* С точки зрения функциональности, разницы нет; оба имеют доступ ко всем членам класса.  
  
11. Конструктор *не возвращает* никакого значения (даже void). Его задача - инициализация объекта.  
  
12. Если явно не определены, компилятор сгенерирует:  
• Конструктор по умолчанию (если нет других конструкторов)  
• Конструктор копирования  
• Оператор присваивания копированием  
• Деструктор  
• Конструктор перемещения (C++11 и новее)  
• Оператор присваивания перемещением (C++11 и новее)  
  
13. Деструктор *не возвращает* никакого значения (даже void). Его задача - освобождение ресурсов.  
  
14. Судя по остальным методам, явно отсутствует конструктор *по умолчанию*, а также методы set\_name() и set\_group(). Хотя конструктор по умолчанию будет создан автоматически компилятором, явно он в коде не указан.  
  
15. Будет вызван конструктор *по умолчанию* (без параметров).  
  
16. Будет вызван конструктор с *двумя параметрами* (string, int).  
  
17. \* student s1("Ivanov",20); - вызывается конструктор с двумя параметрами (string, int).  
\* student s2=s1; - вызывается конструктор копирования.  
\* student s1("Ivanov",20); - вызывается конструктор с двумя параметрами (string, int).  
18.• student s1("Ivanov",20); - конструктор с параметрами (string, int).  
• student s2; - конструктор по умолчанию.  
• s2=s1; - Оператор присваивания копированием (если не перегружен, используется сгенерированный компилятором).  
  
19. Конструктор копирования будет использоваться, так как параметр student a передается *по значению*.  
  
20: ```c++  
Student p;  
p.set\_name("Новое имя");  
  
```

**Ссылка на репозиторий с отчетом Github**

****