Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ

"Лабораторная работа ООП № 2. Классы и объекты.

Использование Конструкторов."

Выполнил

Студент группы РИС-22-2Б

Хохряков А.С

Проверил

Доцент кафедры ИТАС

Полякова О.А

Пермь 2023

# Содержание отчета

1. Постановка задачи.
2. Анализ задачи.
3. UML-Диаграмма
4. Ответы на вопросы
5. Текст программы

Постановка задачи

1. Определить пользовательский класс.

2. Определить в классе следующие конструкторы: без параметров, с параметрами,

копирования.

3. Определить в классе деструктор.

4. Определить в классе компоненты-функции для просмотра и установки полей

данных (селекторы и модификаторы).

5. Написать демонстрационную программу, в которой продемонстрировать все три

случая вызова конструктора-копирования, вызов конструктора с параметрами и

конструктора без параметров.

Вариант 3:

Пользовательский класс МАРШРУТ

Пункт отправления- string

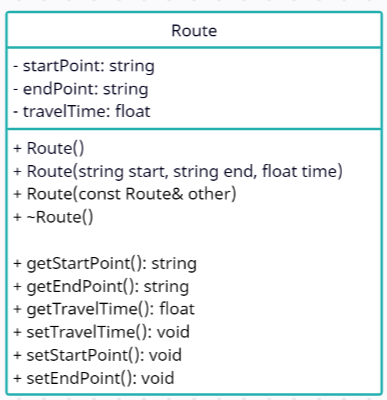
Пункт назначения - string

Время в пути - float

Анализ задачи

1. Определить класс Route, который содержит закрытые поля startPoint, endPoint и travelTime.
2. Определить три конструктора: конструктор по умолчанию, конструктор с параметрами и конструктор копирования.
3. Определить деструктор класса.
4. Определить методы getStartPoint, getEndPoint, getTravelTime, setStartPoint, setEndPoint и setTravelTime для чтения и записи закрытых полей класса.
5. Создать объект route1 класса Route с помощью конструктора по умолчанию, затем устанавливаются его поля методами setStartPoint, setEndPoint и setTravelTime.
6. Вывести значения.

UML – Диаграмма



Ответы на вопросы

1. Конструктор используется для инициализации объектов при их создании.
2. В С++ существует три типа конструкторов: конструктор по умолчанию (без параметров), конструктор с параметрами и конструктор копирования.
3. Деструктор используется для освобождения ресурсов, занятых объектом, перед тем, как объект будет удален из памяти. Деструктор описывается явно, если объект использует динамически выделенную память или другие ресурсы, которые нужно освободить вручную.
4. Конструктор без параметров используется для создания объекта с начальными значениями по умолчанию. Конструктор с параметрами используется для создания объекта с заданными значениями свойств. Конструктор копирования используется для создания нового объекта, который полностью копирует свойства существующего объекта.
5. Конструктор копирования вызывается при создании нового объекта, который полностью копирует свойства существующего объекта, например, при передаче объекта в функцию по значению или при возвращении объекта из функции по значению.
6. Свойства конструкторов: они имеют тот же самый имя, что и класс; они не возвращают значения; они вызываются автоматически при создании объекта; они могут иметь различные параметры.
7. Свойства деструкторов: они имеют тот же самый имя, что и класс, но с символом ~ в начале; они не возвращают значения; они вызываются автоматически при удалении объекта; они не имеют параметров.
8. Методы класса имеют доступ ко всем свойствам и методам класса, а также к приватным свойствам и методам того же класса, в котором они определены.
9. Указатель this представляет собой указатель на текущий объект, который вызывает метод.
10. Методы, определенные внутри класса, имеют доступ к приватным свойствам и методам этого класса, а методы, определенные вне класса, не имеют такого доступа.
11. Конструктор не возвращает значения. Его задача - инициализировать объект.
12. По умолчанию создаются конструктор по умолчанию (без параметров), конструктор копирования, оператор присваивания, деструктор и набор методов доступа (геттеров и сеттеров) для свойств.
13. Деструктор не возвращает значения. Его задача - освободить ресурсы, занятые объектом.
14. В описании класса отсутствует метод для вывода информации о студенте, например, метод show(), который может быть определен как void show(), и который выводит на экран имя и номер группы студента.
15. При выполнении операторов student \*s; s = new student; будет вызван конструктор без параметров.
16. При выполнении операторов student s("Ivanov",20); будет вызван конструктор с параметрами.
17. При выполнении операторов student s1("Ivanov",20); student s2=s1; будет вызван конструктор копирования.
18. При выполнении операторов student s1("Ivanov",20); student s2; s2=s1; будет вызван конструктор с параметрами для s1 и оператор присваивания для s2.
19. При передаче параметра student a в функцию print() будет вызван конструктор копирования для создания копии объекта a. При выходе из функции будет вызван деструктор для удаления копии объекта a.
20. Для присвоения нового значения атрибуту name объекта p можно использовать метод set\_name() следующим образом:

p.set\_name("Новое значение");

В данном случае метод set\_name() устанавливает новое значение для атрибута name объекта p.

Текст программы

#include <iostream>

#include <string>

using namespace std;

class Route {

private:

string startPoint;

string endPoint;

float travelTime;

public:

Route() {

startPoint = "";

endPoint = "";

travelTime = 0.0;

}

Route(string start, string end, float time) {

startPoint = start;

endPoint = end;

travelTime = time;

}

Route(const Route& other) {

startPoint = other.startPoint;

endPoint = other.endPoint;

travelTime = other.travelTime;

}

~Route() {

cout << "Route object destroyed" << endl;

}

string getStartPoint() const {

return startPoint;

}

string getEndPoint() const {

return endPoint;

}

float getTravelTime() const {

return travelTime;

}

void setStartPoint(string start) {

startPoint = start;

}

void setEndPoint(string end) {

endPoint = end;

}

void setTravelTime(float time) {

travelTime = time;

}

};

int main() {

setlocale(0, "en\_US.UTF-8");

Route route1;

route1.setStartPoint("Moscow");

route1.setEndPoint("Saint Petersburg");

route1.setTravelTime(5.5);

cout << "Route 1: " << route1.getStartPoint() << " - " << route1.getEndPoint() << ", travel time: " << route1.getTravelTime() << endl;

Route route2("Kazan", "Yekaterinburg", 3.5);

cout << "Route 2: " << route2.getStartPoint() << " - " << route2.getEndPoint() << ", travel time: " << route2.getTravelTime() << endl;

Route route3 = route1;

cout << "Route 3 (copy of Route 1): " << route3.getStartPoint() << " - " << route3.getEndPoint() << ", travel time: " << route3.getTravelTime() << endl;

return 0;

}