Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №2
з дисципліни «Об’єктно-орієнтоване програмування»

Тема:
«Конструктори і деструктори»

Виконав: студент групи КБ-24мб
Воробйов Д.С.
Перевірив:
Козірова Н.Л.

Кропивницький 2024

Мета: ознайомитись з основними поняттями конструктор і деструктор в ООП та навчитись їх програмно реалізовувати мовою С++.

Завдання:

Завдання 1:

1. Реалізуйте конструктор за замовчуванням, конструктор з параметрами та копіюючий конструктор для вашого класу з лабораторної роботи 1:

– конструктор за замовчуванням має встановлювати значення полів за замовчуванням;

– конструктор з параметрами має приймати значення для кожного поля;

– копіюючий конструктор має копіювати значення полів з іншого об'єкта класу.

2. Реалізуйте деструктор для класу. Деструктор має повідомлення про знищення об'єкта;

3. У функції main створіть об'єкт за допомогою конструктора за замовчуванням та виведіть значення його полів;

4. Створіть новий об'єкт за допомогою конструктора з параметрами та встановіть значення для полів. Виведіть значення полів цього об'єкта;

5. Створіть ще один об'єкт і скопіюйте значення полів з першого об'єкта за допомогою копіюючого конструктора. Виведіть значення полів цього об'єкта;

6. Завершіть функцію main, що призведе до виходу з області видимості створених об'єктів і виклику їх деструкторів. Переконайтесь, що повідомлення про знищення об'єктів виводяться.

Завдання 2

Розробіть клас «Граф» - Graph, який представляє неорієнтований граф із заданою кількістю вершин п. Клас повинен містити конструктори. Реалізуйте методи для додавання та видалення ребер, перевірки наявності ребра між двома вершинами, знаходження всіх сусідів даної вершини. Створіть масив об'єктів класу Graph. Передайте пари об'єктів у функцію, яка перевіряє, чи є графи ізоморфними, та повертає результат у головну програму.

Результати (обробка даних, графіки, діаграми, таблиці)

Завдання 1:

Лістинг Country.h

```
#ifndef COUNTRY_H
#define COUNTRY_H
#include <string>

class Country {
public:
    Country();

        Country(std::string name, std::string capital, int
population);

    Country(const Country& other);

    ~Country();

    void setName(std::string name);
    std::string getName();

    void setCapital(std::string capital);
    std::string getCapital();

    void setPopulation(int population);
    int getPopulation();
private:
    std::string name;
    std::string capital;
    int population;
};

#endif
```

Лістинг Country.cpp

```
#include "Country.h"
```

```
#include <iostream>


Country::Country() {
    name = "Not initialized";
    capital = "Not initialized";
    population = 0;
    std::cout << "Default constructor called." << std::endl;
}

Country::Country(std::string name, std::string capital, int
population) {
    this->name = name;
    this->capital = capital;
    this->population = population;
    std::cout << "Parameterized constructor called." << std::endl;
}

Country::Country(const Country& other) {
    name = other.name;
    capital = other.capital;
    population = other.population;
    std::cout << "Copy constructor called." << std::endl;
}

Country::~Country() {
    std::cout << "Destructor called." << std::endl;
}

void Country::setName(std::string name) {
    this->name = name;
}
std::string Country::getName() {
    return name;
}
void Country::setCapital(std::string capital) {
    this->capital = capital;
```

```
}
std::string Country::getCapital() {
    return capital;
}
void Country::setPopulation(int population) {
    this->population = population;
}
int Country::getPopulation() {
    return population;
}
```

Лістинг main.cpp

```
#include <iostream>
#include "Country.h"

using namespace std;

int main(){
    Country defaultConstructor;
    cout << "Default Country: " << defaultConstructor.getName() <<
endl;
    cout << "Default Capital: " << defaultConstructor.getCapital()
<< endl;
            cout << "Default Population: " <<
defaultConstructor.getPopulation() << endl << endl;

    Country UKR("Ukraine", "Kyiv", 41258478);
    cout << "Parameterized Country: " << UKR.getName() << endl;
    cout << "Parameterized Capital: " << UKR.getCapital() << endl;
    cout << "Parameterized Population: " << UKR.getPopulation() <<
endl << endl;

    Country copiedUKR(UKR);
    cout << "Copied Country: " << copiedUKR.getName() << endl;
    cout << "Copied Capital: " << copiedUKR.getCapital() << endl;
    cout << "Copied Population: " << copiedUKR.getPopulation() <<
endl << endl;
    return 0;}
```

```
Default constructor called.
Default Country: Not initialized
Default Capital: Not initialized
Default Population: 0

Parameterized constructor called.
Parameterized Country: Ukraine
Parameterized Capital: Kyiv
Parameterized Population: 41258478

Copy constructor called.
Copied Country: Ukraine
Copied Capital: Kyiv
Copied Population: 41258478

Destructor called.
Destructor called.
Destructor called.
```

Рисунок 1 – Результат роботи програми

Завдання 2

Лістинг Graph.h

```
#ifndef GRAPH_H
#define GRAPH_H

#include <vector>

class Graph {
public:
    Graph(int n);

    void addEdge(int u, int v);
    void removeEdge(int u, int v);
    bool hasEdge(int u, int v);
    std::vector<int> getNeighbors(int u);
    bool isIsomorphic(Graph& other);
    void printGraph();
private:
    int numVertices;
    std::vector<std::vector<int>> adjMatrix;
};
#endif
```

Лістинг Graph.cpp

```
#include "Graph.h"
#include <algorithm>
#include <iostream>

Graph::Graph(int n) {
    numVertices = n;
    adjMatrix.resize(n, std::vector<int>(n, 0));
}

void Graph::addEdge(int u, int v) {
    adjMatrix[u][v] = 1;
    adjMatrix[v][u] = 1;
}

void Graph::removeEdge(int u, int v) {
    adjMatrix[u][v] = 0;
    adjMatrix[v][u] = 0;
}

bool Graph::hasEdge(int u, int v) {
    return adjMatrix[u][v] == 1;
}

std::vector<int> Graph::getNeighbors(int u) {
    std::vector<int> neighbors;
    for (int v = 0; v < numVertices; v++) {
        if (adjMatrix[u][v] == 1) {
            neighbors.push_back(v);
        }
    }
    return neighbors;
}

bool Graph::isIsomorphic(Graph& other) {
    if (numVertices != other.numVertices) {
        return false;
```

```
    }

    for (int i = 0; i < numVertices; i++) {
        std::vector<int> neighbors1 = this->getNeighbors(i);
        std::vector<int> neighbors2 = other.getNeighbors(i);

        sort(neighbors1.begin(), neighbors1.end());
        sort(neighbors2.begin(), neighbors2.end());

        if (neighbors1 != neighbors2) {
            return false;
        }
    }

    return true;
}

void Graph::printGraph() {
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            std::cout << adjMatrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
}
```

Лістинг main.cpp

```
#include "Graph.h"
#include <iostream>

using namespace std;


int main() {
    Graph graph1(4);
    Graph graph2(4);

    graph1.addEdge(0, 1);
```

```
    graph1.addEdge(0, 2);
    graph1.addEdge(1, 2);
    graph1.addEdge(1, 3);

    graph2.addEdge(0, 1);
    graph2.addEdge(0, 2);
    graph2.addEdge(1, 2);
    graph2.addEdge(1, 3);

    cout << "Graph 1:" << endl;
    graph1.printGraph();

    cout << "Graph 2:" << endl;
    graph2.printGraph();

    if (graph1.isIsomorphic(graph2)) {
        cout << "Graphs are isomorphic." << endl;
    } else {
        cout << "Graphs are not isomorphic." << endl;
    }

    return 0;
}
```

```
Graph 1:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
Graph 2:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
Graphs are isomorphic.
```

```
Graph 1:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
Graph 2:
0 1 1 0
1 0 0 1
1 0 0 0
0 1 0 0
Graphs are not isomorphic.
```

Рисунок 2 – Результат роботи програми

Висновки: в лабораторній роботі було вивчено основні концепції конструкторів і деструкторів в ООП, а також їх реалізацію на C++.