Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

з комп’ютерного практикуму № 1

на тему :

„ Класи с++ ”

Варіант №19

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав студент** |  | Цицилюк Анна Валеріївна |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  |  |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

ЗМІСТ

[1 Мета роботи 3](#_Toc497849112)

[2 Постановка задачі 4](#_Toc497849113)

[3 Тексти програмного коду 6](#_Toc497849114)

[4 Результат роботи програми 9](#_Toc497849115)

[Висновки 10](#_Toc497849116)

# Мета роботи

*Мета роботи* – дослідити основні концепції об’єктно-орієнтованого програмування. Дослідити особливості використання классів і об’єктів, а також особливості застосування констукторів і деструкторів.

# Постановка задачі

Визначити класс «Оріїнтований граф »,який має такі елементи: розмірність матриці суміжності n, покажчик на тип даних елементів матриці суміжності. Визначити такі конструктори:

- конструктор ініціалізації, що дозволяє користувачеві задати вагову матрицюсуміжності (відстані) певного розміру.

Визначити метод класу, знаходить всі шляху між двома заданими користувачем вершинами в графі, при цьому вони не повинні мати загальних вершин крімпочаткової і кінцевої.

# Тексти програмного коду

**OrGraf.cpp**

#include "OrGraf.hpp"

//#include "File.hpp"

#include <iostream>

#include <vector>

using namespace std;

int i ,j;

OrientedGraph::OrientedGraph(int n):n(n),arr(new int\*[n]){      //матрица смежности (конструктор)

    //  int \*\*arr=new int \*[n];

    for( i = 0;i<n; i++)

        arr[i]=new int [n];                         //выделение памяти под массив

    for(i=0;i<n;i++)

        for(j = 0;j<n;j++)

        {if(i != j)

        {std::cout<<"Введіть відстань між "<<i+1<<" та "<<j+1<<" вершинами(Якщо відповідного ребера не існує введіть від'ємне значення )"<<std::endl;   //заполнение массива

            std:: cin>>arr[i][j];}

            // arr[j][i]=arr[i][j];}

        else arr[i][j]=0;

        }

}

void OrientedGraph::Print()

{

    for(i = 0;i<n;i++)

    {for(j = 0;j<n;j++)

        if(arr[i][j] >=0)

            std::cout<<arr[i][j]<<"   ";

        else std::cout <<"\_"<<"   ";            //если ребра нет ---ставит \_

        std:: cout<<std::endl;

    }

}

OrientedGraph::~OrientedGraph(){    //просто деструктор

    for(int i = 0; i<n;i++)

    {

        delete [](arr[i]);

    }

    delete [](arr);

};

void OrientedGraph::FindVay(int start,int end,std::vector<int> Stek,int size,int \*\*arr) //рекурсивная функция поиска путей параметры--- начало,конец,стек с текущим путем,размер матрицы,сама матрица

{

    //{cout<<"NEW"<<endl;

    if(Stek.size()==0)                                  //если начинаем сначала-удалять просто пока не чего

    {

        if(arr[start][end]>0)                               //если с начала можна попасть в конец сразу выводим и удаляем этот путь с матрицы

        {std::cout<<start+1<<" \_ "<<end+1<<std::endl;

            arr[start][end]=-99;                            //так званое "удаление"

            FindVay(start, end, Stek, size, arr);           //рекурсивный запуск функции с пустым стеком

        }

        else {for(i=0;i<size;i++)

            if(arr[start][i]>0)                         //поиск путей со старта

            {Stek.push\_back(i);                         //помещение найденой вершины в стек

                FindVay(start, end, Stek, size, arr);   //рекурсивный запуск функции

                arr[start][i]=0;}}                      //если при рекурсии путь не был найден то нужно удалить эту стартовую вершину

    }

    else {                      //ЕСЛИ СТЕК НЕ ПУСТ

        int element=Stek.back();                                //просто переменная для удобной работы(последний из стека)

        int S=Stek.size();                                      //текущий размер стека

        if(arr[element][end]>0)                                 //если с этой вершины можно перейти к конечной

        {

            std::cout<<start+1;                                 //выведение маршрута

            for(i=0;i<S;i++)

                std::cout<<"\_"<<Stek[i]+1;

            std::cout<<"\_"<<end+1<<endl;

            for(i=0;i<S;i++)                                    //удаление путей с матрицы

                for(j=0;j<size;j++)

                {

                    arr[Stek[i]][j]=-99;

                    arr[j][Stek[i]]=-99;

                }

            Stek.clear();               //очищение стека

        }

        else {{for(i=0;i<size;i++)          //поиск следуйщей вершины графа

            if(arr[element][i]>0)

            {Stek.push\_back(i);             //пополнение стека

                FindVay(start, end, Stek, size, arr);//рекурсивный запуск

                arr[element][i]=0;}}        //удаление даного элемента

            for(i=0;i<size;i++)             //если после перебора всех вариантов с вершины нельзя дойти до конца

            {arr[element][i]=-99;           //удаление всех путей к ней

                arr[i][element]=-99;}

            if(S!=0){//for(i=0;i<S;i++)cout<<Stek[i];

                Stek.pop\_back();}}          //удаление ее из стека

        FindVay(start, end, Stek, size, arr);//рекурсивный перезапуск

    }

}

void OrientedGraph::FindAll(){      //функция самого запуску поиска

    int start,end;

    int \*\*a=new int\*[n];            //копирование массива

    for( i = 0;i<n; i++)

        a[i]=new int [n];

    for(i=0;i<n;i++)

        for(j=0;j<n;j++)

            a[i][j]=arr[i][j];

    std::cout<<"початок шляху"<<endl;   //ввод даных пользователем

    std::cin>>start;

    std::cout<<"кінець"<<endl;

    std::cin>>end;

    std:: vector <int> Stek;            //создание стека

    //Stek.push\_back(1);

    if(start<=n&&end<=n)

        FindVay(start-1, end-1, Stek, n, a);    //запуск функции прохода

    else cout<<"Таких вершин немає!!!!!!!!"<<endl;

}

**OrGraf.hpp**

#include <vector>

#include <stdio.h>

//#ifndef File\_hpp

//#define File\_hpp

class OrientedGraph     //класс ориентированый граф

{

private:

    int n;          //размерность матрицы смежности

    int \*\*arr;      //сама матрица смежности

    public :

    //  OrientedGraph();

    OrientedGraph(int n);   //конструктор для матрицы смежности

    void Print();           //выведение матрицы расстояния

    ~OrientedGraph();

    void FindVay(int start,int end,std::vector<int> Stek,int size,int \*\*arr);

    // void Find(int n,int m,int \*\*arr);

    void FindAll();

};

**main.cpp**

#include <iostream>

#include "OrGraf.hpp"

using namespace std;

int main(int argc, const char \* argv[]) {

    int n;

    cout<<"Введіть кількість вершин графу";

    cin>>n;

    OrientedGraph MyGraf(n);        //заполнение матрицы

    MyGraf.Print();                 //выведение матрицы

    int m;

    do{

        cout<<"Знайти шлях?(1-так/0-ні)";

        cin>>m;

        if(m)

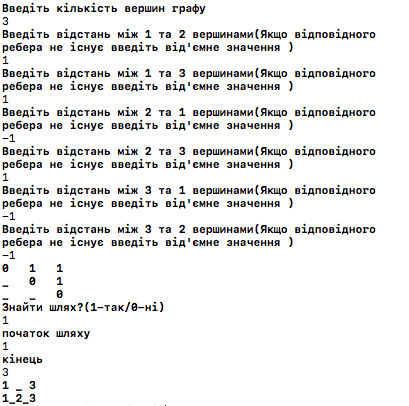
            MyGraf.FindAll();               //поиск путей

    }while (m);

    return 0;

}

# Результат роботи програми



Висновки

Я дослідила основні концепції об’єктно-орієнтованого програмування. Вивчила особливості використання классів і об’єктів, а також особливості застосування констукторів і деструкторів.