Государственное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Донецкий национальный технический университет»

Д09.03.04-ПОИС.20-20/6251.ЛР

***Кафедра*** искусственного интеллекта  
 и системного анализа

Лабораторная работа №2

по дисциплине "Объектно-ориентированное программирования"

на тему: "Одиночное и множественное наследование. Полиморфизм."

Проверили:

\_\_\_\_\_\_\_\_\_\_\_\_ асс. А.П. Семёнова

(дата, подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ асс. И.В. Савицкая

(дата, подпись)

Выполнил:

\_\_\_\_\_\_\_\_\_\_\_\_\_ ст.гр.ПИ-20г В.А. Пустовой

(дата, подпись)

Донецк – 2021

Тема: Одиночное и множественное наследование. Полиморфизм.

Цель работы: Приобретение практических навыков реализации наследования и иерархии классов на языке С++.

Задание. Создать и протестировать следующие классы:

1. Абстрактный класс

Создать абстрактный класс Container, в котором определить чисто виртуальные функции для добавления, удаления и определения количества элементов в контейнере.

1. Класс одиночного наследования.

2.1 Модифицировать класс динамической структуры данных, содержащей строки (л.р. №1), чтобы он был производным от класса Container.

2.2 Создать 2 различных класса, производных от класса динамической структуры, позволяющие сортировать и фильтровать выходные данные. Для этого вводятся виртуальные функции, соответственно, для сортировки – функция для сравнения 2 строк, для фильтрации – предикат, определяющий, будет ли строка присутствовать в выборке фильтра.

3. Класс множественного наследования.

Создать класс, осуществляющий одновременно сортировку и фильтрацию.

Вариант:

Сортировка – 4) По убыванию вещественного значения, содержащегося в строке.

Фильтрация – 7) Строки, содержащие подстроку “ООП”.

Ход работы:

Листинг:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>//4 вариант сортировка 7 фильтр

#include <windows.h>

#include <string>

using namespace std;

struct vertex{

string weight;

int edges[100];};

class Container{

public:

virtual void add\_top(int q, int w, vertex z) = 0;

virtual void del\_top(int q) = 0;

virtual void get\_top\_count() = 0;};

class Graph: public Container{

public:

int g\_top\_count;

int g\_edges\_count;

vertex g\_vert[100];

Graph() {}

Graph(int a) { g\_top\_count = a; }

Graph(const Graph& A)

{

g\_top\_count = A.g\_top\_count;

g\_edges\_count = A.g\_edges\_count;

for (int i = 1; i < A.g\_top\_count; i++)

{

for (int k = 1; k < 99; k++)

g\_vert[i].edges[k] = A.g\_vert[i].edges[k];

g\_vert[i].weight = A.g\_vert[i].weight;

}

}

void print(){

cout << "Число вершин: " << g\_top\_count << endl;

for (int i = 1; i <= g\_top\_count; i++)

{

cout << "Вес вершины " << i << ": " << g\_vert[i].weight << ';' << endl;

}

cout << "Число рёбер: " << g\_edges\_count << endl;

for (int k = 1; k <= g\_edges\_count; k++)

{

for (int i = 1; i <= g\_edges\_count; i++)

{

cout << g\_vert[k].edges[i] << ' ';

}

cout << endl;

}

}

void set\_all(int top, int edge, vertex vert[100]){

g\_top\_count = top;

g\_edges\_count = edge;

for (int i = 1; i <= 99; i++)

{

g\_vert[i].weight = vert[i].weight;

for (int k = 1; k <= 99; k++)

g\_vert[i].edges[k] = vert[i].edges[k];

}

}

virtual void Container::get\_top\_count() { cout << g\_top\_count; }

virtual void Container::del\_top(int q){

for (int i = q; i <= g\_top\_count; i++)

g\_vert[i].weight = g\_vert[i + 1].weight;

for (int i = 1; i <= g\_top\_count; i++)

{

for (int j = q; j <= g\_top\_count; j++)

{

g\_vert[i].edges[j] = g\_vert[i].edges[j + 1];

}

}

for (int i = 1; i <= g\_top\_count; i++)

{

for (int j = q; j <= g\_top\_count; j++)

{

g\_vert[j].edges[i] = g\_vert[j + 1].edges[i];

}

}

g\_edges\_count--;

g\_top\_count--;

}

void del\_edge(int q, int w)

{

g\_vert[q].edges[w] = 0;

g\_vert[w].edges[q] = 0;

}

virtual void Container::add\_top(int q, int w, vertex z)

{

g\_top\_count++;

g\_edges\_count++;

g\_vert[g\_top\_count].weight = z.weight;

for (int i = 0; i <= g\_top\_count; i++)

{

if (z.edges[i] == 1)

{

g\_vert[q].edges[i] = 1; //баг, вершины записываются друг на друга

g\_vert[i].edges[q] = 1;

}

}

}

void add\_edge(int q, int w)

{

g\_vert[q].edges[w] = 1;

g\_vert[w].edges[q] = 1;

}

};

class Sorting : public Graph //4 вариант

{

public:

virtual void sorting(int s\_top\_count, double d\_buffer[])

{

double temp;

for (int i = 1; i <= s\_top\_count; i++) {

for (int j = 1; j <= s\_top\_count - i; j++) {

if (d\_buffer[j] > d\_buffer[j + 1]) {

temp = d\_buffer[j];

d\_buffer[j] = d\_buffer[j + 1];

d\_buffer[j + 1] = temp;

}

}

}

cout << "\nОтсортированный граф:\n";

for (int i = 1; i <= s\_top\_count; i++)

{

cout << d\_buffer[i] << ' ';

}

}

};

class Filter: public Graph //7 вариант

{

public:

virtual void fiter(int f\_top\_count, vertex buffer[])

{

cout << "\nОтфильтрованные веса: " << endl;

for (int i = 1; i <= f\_top\_count; i++)

{

if (buffer[i].weight.find("ООП") != -1)

cout << buffer[i].weight << ' ';

}

}

};

class Mult : virtual public Filter, virtual public Sorting{};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int top\_count, edges\_count;

vertex array[100];

int q, w, var\_case;

cout << "Введите число вершин: ";

cin >> top\_count;

if (top\_count > 99) {

cout << endl << "Слишком много, надо меньше 100";

exit;

}

cout << "Введите веса вершин: ";

for (int i = 1; i <= top\_count; i++)

cin >> array[i].weight;

cout << "Введите число рёбер: ";

cin >> edges\_count;

if (edges\_count > 99) {

cout << endl << "Слишком много, надо меньше 100";

exit;

}

for (int k = 1; k <= 99; k++)

{

for (int i = 1; i <= 99; i++)

{

array[k].edges[i] = 0;

}

}

cout << "Рёбра: " << endl;

for (int i = 0; i < edges\_count; i++) {

scanf("%d - %d", &q, &w);

array[q].edges[w] = 1;

array[w].edges[q] = 1;

}

Graph F;

F.set\_all(top\_count, edges\_count, array);

F.print();

double d\_buffer[101];

for (int i = 1; i <= top\_count; i++)

d\_buffer[i] = stod(array[i].weight);

Sorting S;

S.sorting(top\_count, d\_buffer);

Filter Filt;

Filt.fiter(top\_count, array);

Mult M;

M.fiter(top\_count, array);

M.sorting(top\_count, d\_buffer);

int top\_new\_number, top\_del, edge\_new\_number;

vertex top\_new;

do

{

cout << "\n0 - для выхода, 1 - принт, 2 - добавить вершину, 3 - удалить вершину, 4 - добавить ребро, 5 - удалить ребро, 6 - определить количество элементов" << endl;

cin >> var\_case;

switch (var\_case)

{

case 0:

return 0;

case 1:

F.print();

break;

case 2:

cout << "Введите номер вершины: ";

cin >> top\_new\_number;

cout << endl << "Введите вес вершины: ";

cin >> top\_new.weight;

cout << endl << "Введите кол-во рёбер: ";

cin >> edge\_new\_number;

for (int i = 0; i < edge\_new\_number; i++)

{

cin >> q;

top\_new.edges[q] = 1;

}

F.add\_top(top\_new\_number, edge\_new\_number, top\_new);

break;

case 3:

cout << "Введите номер удаляемой вершины: ";

cin >> top\_del;

F.del\_top(top\_del);

break;

case 4:

cout << "Введите вершины, соединённые этим ребром: ";

cin >> q >> w;

F.add\_edge(q, w);

break;

case 5:

cout << "Введите вершины, соединённые этим ребром: ";

cin >> q >> w;

F.del\_edge(q, w);

break;

case 6:

cout << "Количество элементов: ";

F.get\_top\_count();

cout << endl;

default:

break;

}

} while (var\_case != 0);

}

Экранные формы

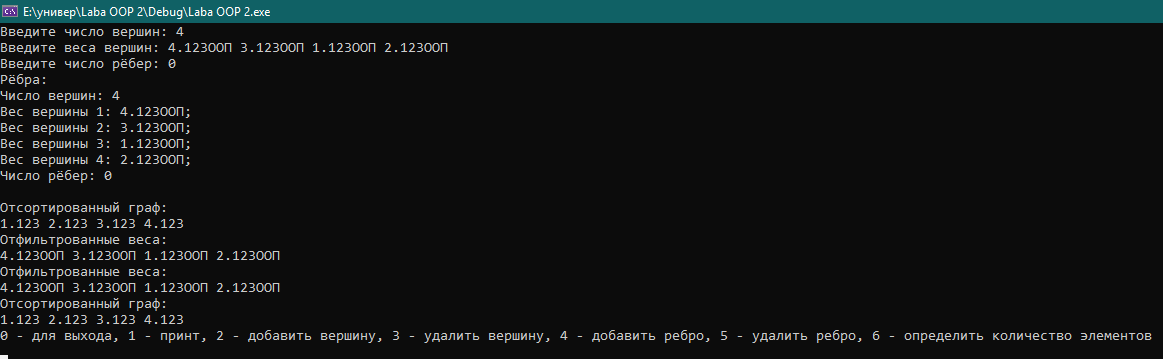


Рисунок 1 – Фильтрация и сортировка весов

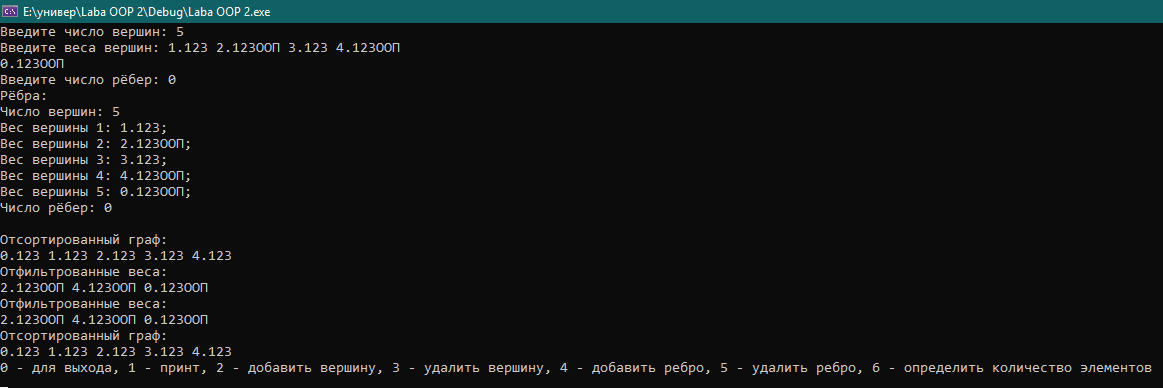


Рисунок 2 – Фильтрация и сортировка

Вывод: приобрели практические навыки реализации наследования и иерархии классов на языке С++.