Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой _____

Преподаватель _____
(подпись)
«___» _____ 2022 г.

Отчет
по лабораторной работе № 1

"Упорядочение исходного списка работ сетевого графика"

по дисциплине «Разработка и реализация проектов»

Студент группы ПИ-92 Шинтяпин И. И.

Преподаватель доцент, к.э.н. Астахова А.В.

Барнаул 2022

Задание

1. Изучить параграфы 2.1 – 2.4, обратить внимание на правила построения сетевых графико
2. Ознакомиться с тестовым примером в Приложении к данному заданию.
3. Для разработки алгоритма и отладки программы разработать свои тестовые примеры.
4. Разработать алгоритм частичного упорядочения работ сетевого графика по заданному списку не упорядоченных взаимосвязанных работ некоторого проекта с учетом требований, изложенных в Приложении, и с учетом минимизации времени работы алгоритма на больших массивах. Выявить начальную и конечную вершины графа. Выявить все полные пути от начальной вершины до конечной.
   При разработке алгоритма и программы не должны использоваться стандартные библиотеки работы с графами и программы расчета сетевых графиков.
5. Привести описание алгоритма.
6. Составить и отладить программу на выбранном языке программирования, которая позволяет упорядочить заданный список работ и выводит на экран монитора:
   - исходный список работ;
   - частично упорядоченный (с шифрами событий и с индексами) список работ;
   - список всех полных путей СГ – от начальной вершины до конечной.
7. Оформить отчет. Защитить работу.

Решение

1. **Описание алгоритма:**
   В рамках реализации данного задания был создан класс Graf (файл Graf.h), содержащий динамический массив vector таблицы путей (работ) начального сетевого графика в виде структуры Table и таблицу работ упорядоченного СГ, которая будет заполнена в процессе работы алгоритмов упорядочивания. Полями данных структур являются целочисленные переменные: шифр предшествующего события, шифр последующего события и продолжительность соответствующей работы (время) и булевая переменная отвечающая на вопрос: посещена ли данная работа или нет. Также класс Graf содержит шифр первой вершины и последней вершины сетевого графика.
   Методы данного класса:

   - void search_first_top(); // поиск первой вершины графа
   - void rekurs_search(vector<int> *way);// рекурсивная функция обхода вершин графа
   - Graf(ifstream *fin);// конструктор
   - void print();// метод печати таблицы путей (работ)
   - void optimized_graf();// функция оптимизации (удаления петель, одинаковых работ, поиска конечной вершины СГ)
   - void search_all_way();// функция поиска и вывода всех полных путей в СГ
   - void graf_struct();// упорядочивание СГ

   Реализация методов в файле Graf.cpp.
   Чтение начальной таблицы происходит из файла input.dat, вывод в консоль.
   Изначально происходит поиск начальной (первой) вершины сетевого графика с помощью внешнего и вложенного цикла по строкам таблицы путей графа. Во вложенном цикле каждый раз происходит проверка, что текущая вершина не является стартовой (в нее входит дуга). Если найдена стартовая вершина, проверяется первая ли она, если первая, то происходит запоминание ее шифра и переход к следующей вершине. Если стартовая вершина уже существует, то происходит добавление фиктивной вершины и добавляется дополнительная запись в таблицу путей СГ.
   Далее происходит оптимизация сетевого графика (удаления петель, одинаковых работ) и поиска конечной вершины. Данная задача также решается с помощью внешнего и вложенного цикла по строкам таблицы сетевого графика. Изначально происходит сравнение на совпадение стартовой и конечной вершины каждой дуги (работы), если найдена петля, она удаляется. Во вложенном цикле происходит сравнение двух строк таблице на совпадение, если они одинаковые и их веса равны – удаляется одна из строк, если веса равные, пользователю предлагается выбрать, какую из строк удалить. После чего происходит поиск конечной вершины сетевого графика по немного измененному алгоритму поиска стартовой вершины. Последний этап – поиск всех полных путей в графе. Реализуется с помощью рекурсивного обхода вершин СГ. Сначала в массив пути происходит запись стартовой вершины графа и следующей за ней вершины из таблицы, затем вызывается рекурсивная функция, в которой первым делом происходит проверка, что достигнута последняя вершина графа, если так, то вывод полного пути из массива и удаление оттуда

последней вершины, возврат из функции вверх по стеку. Если последняя записанная в массив вершина не конечная, то с помощью цикла находится последующая за ней вершина и записывается в массив пути и происходит рекурсивный вызов данной функции. Если все следующие вершины после текущей просмотрены, то эта вершины удаляется из массива и возврат из функции вверх по стеку.

Частичное упорядочивание СГ происходит в функции graf_struct(). Здесь создается очередь top, в которую будут добавляться вершины графа по мере достижения. Сначала в очередь заносится первая (стартовая) вершина СГ и начинается цикл с предусловием, который будет продолжаться пока размер новой таблицы работ не совпадет с изначальной. В данном цикле есть второй цикл по всем строкам таблицы работ, в котором происходит проверка, что начальная вершина текущей работы равна последней в очереди, запомненной вершине и, что текущая работа еще не посещена. Если это так, то происходит запись данной работы в новую таблицу struct_graf и добавление конечной вершины данной работы в очередь работ, также отмечается что текущая работа посещена. После вложенного цикла – удаление из очереди последней последней вершины.

2. **Код программы:**

Файл Graf.h

```
#pragma once
#include <iostream>
#include <vector>
#include <Conio.h>
#include <fstream>
using namespace std;


struct Table
{
	int start_top;// начальная вершина дуги графа
	int end_top;// конечная вершина дуги графа
	int weight;// вес дуги
	bool metka;// метка вершины, посещена или нет
};


class Graf
{
private:
	vector<Table> graf; // вектор, содержащий дуги графа и веса в виде таблицы
	vector<Table> struct_graf;
	int first_top;// первая вершина графа
	int last_top;// последняя вершина графа

	void search_first_top(); // поиск первой вершины графа
	void rekurs_search(vector<int> *way);// рекурсивная функция обхода вершин графа
	void graf_struct();// упорядочивание СГ
public:
	Graf(ifstream *fin);// конструктор
	void print(const bool flag);// метод печати таблицы путей (работ)
	void optimized_graf();// функция оптимизации (удаления петель, одинаковых работ, поиска конечной вершины СГ)
	void search_all_way();// функция поиска и вывода всех полных путей в СГ
};
```

Файл Graf.cpp

```
#include "stdafx.h"
#include "Graf.h"

// конструктор
Graf::Graf(ifstream *fin)
{
	Table *index = new Table;

// заполение таблицы работ СГ
	while (*fin >> index->start_top)
	{
		*fin >> index->end_top;
		*fin >> index->weight;
		index->metka = false;
		graf.push_back(*index);
	}
	delete index;
	// начальные значения первой и послед вершин СГ
	first_top = -500;
	last_top = -500;
}

// метод вывода СГ в консоль
void Graf::print(const bool flag)
{
	if (!flag)
	{
		cout << "Таблица П1:" << endl;
		printf("*********************************\n");
		printf("*     A    *     B    *     T    *\n");
		printf("*********************************\n");
		for (auto i = graf.begin(); i != graf.end(); i++)
		{
			printf("*%10d*%10d*%10d*\n", i->start_top, i->end_top, i->weight);
			printf("*********************************\n");
		}
	}
	else
	{
		cout << "Таблица П2:" << endl;
		printf("*********************************\n");
		printf("*     A    *     B    *     T    *\n");
		printf("*********************************\n");
		for (auto i = struct_graf.begin(); i != struct_graf.end(); i++)
		{
			printf("*%10d*%10d*%10d*\n", i->start_top, i->end_top, i->weight);
			printf("*********************************\n");
		}

		cout << "\nНачальная вершина: " << first_top << endl;
		cout << "Конечная вершина: " << last_top << endl << endl;
	}
}

// функция оптимизации (удаления петель, одинаковых работ, поиска конечной вершины СГ)
void Graf::optimized_graf()
{
	search_first_top(); // поиск первой вершины графа
	// итерация по таблице путей СГ (внешний цикл)
	one:
	for (auto i = graf.begin(); i !=graf.end(); i++)
```

```
	{
		if (i->start_top == i->end_top)// условие нахождения петли
		{
			cout << "Найдена петля, автоматическое удаление работы" << endl;
			graf.erase(i);// удаление записи в таблице
			goto one;// возврат в начало цикла
		}
		two:
		for (auto j = graf.begin(); j != graf.end(); j++)// вложенный цикл для сравнения
всех вершин СГ
		{
			if ((i->start_top == j->start_top) && (i->end_top == j->end_top) && (i !=
j))// условие дублирования работы
			{
				cout << "ошибка, работа " << i->start_top << "->" << i->end_top<<"
дублируется" << endl;
				if (i->weight != j->weight)// если веса разные
				{
					cout << "но имеет 2 веса:" << endl;
					cout << "1) " << i->weight << endl;
					cout << "2) " << j->weight << endl;
					cout<<"выбрать нужное ( по умолчанию удалиться 2) ): ";
					switch (_getche())
					{
					case '1': // удаление 1 работы
						graf.erase(i);
						cout << endl;
						goto one;
						break;
					default:// по умолчанию удаляется 2 работа
						graf.erase(j);
						cout << endl;
						goto two;
						break;
					}
				}
				else// если вес совпадает, удаление одной из работ
				{
					cout << "с одним весом: " << i->weight <<endl;
					cout << "автоматическое удаление одной работы" << endl;
					graf.erase(j);
					goto two;
				}
			}
		}
	}
	// поиск последней вершины СГ
	bool flag_last_top;// флаг нахождения послед вершины
	three:
	int size = graf.size();// запоминание кол-во строк таблицы до поиска
	for (int i = 0; i < size; i++)// внешний цикл по таблице работ СГ
	{
		flag_last_top = true;// начальное значение флага
		for (int j = 0; j < size; j++)// вложенный цикл для сравнения вершин СГ
		{
			// условие, что вершина не последняя
			if                          ((graf[i].end_top                                  ==
graf[j].start_top)&&(graf[i].end_top!=graf[j].end_top))
			{
				flag_last_top = false;// обнуление флага
				break;// завершение цикла
```

```
                        }
                }
                if (flag_last_top && !graf[i].metka )// если найдена последняя вершина
                {
                        if (last_top == -500)// проверка, первая эта вершина или нет
                        {
                                // если вершина первая, запоминание ее шифра и отметка всех работ с
этим шифром как пройденные
                                last_top = graf[i].end_top;
                                for (int temp = 0; temp < graf.size(); temp++)
                                {
                                        if (graf[temp].end_top == graf[i].end_top)
                                        {
                                                graf[temp].metka = true;
                                        }
                                }
                        }
                        else// если уже есть послед вершина
                        {
                                cout << "Вершины графа (события) с шифрами " << last_top << " и " <<
graf[i].end_top << " конечные." << endl;
                                cout << "Ввести фиктивную конечную вершину или, если вершина уже
фиктивна, добавить работу?(1-да, 2-нет): ";
                                char c = _getche();
                                cout << endl;
                                if (c == '1')
                                {
                                        // создание фиктивной вершины
                                        Table *index = new Table;
                                        index->start_top = last_top;
                                        index->end_top = 100;
                                        index->weight = 0;
                                        graf.push_back(*index);

                                        index->start_top = graf[i].end_top;
                                        graf.push_back(*index);

                                        delete index;
                                        if (last_top != 100)
                                        {
                                                last_top = 100;
                                        }
                                        for (auto temp = 0; temp < graf.size(); temp++)
                                        {
                                                if (graf[temp].end_top == graf[i].end_top)
                                                {
                                                        graf[temp].metka = true;
                                                }
                                        }
                                }
                                else// удаление одной из вершин
                                {
                                        graf.erase(graf.begin() + i);
                                        last_top = -500;
                                        for (int i = 0; i < graf.size(); i++)// итерация по таблице
вершин графа
                                        {
                                                graf[i].metka = false;
                                        }
                                        goto three;
                                }
```

```
                }
            }
        }
        for (int i = 0; i < graf.size(); i++)// итерация по таблице вершин графа
        {
            graf[i].metka = false;
        }
        graf_struct();
}

// метод поиска первой вершины
void Graf::search_first_top()
{
        bool flag_search;// флаг, отвечающий за определение начальной вершины
        int size = graf.size();
        for (auto i = 0; i < size; i++)// итерация по таблице вершин графа
        {
            flag_search = true;
            for (auto j = 0; j < size; j++)// вторая итерация для поиска первой вершины
            {
                // условие, что в вершину входит дуга, значит это не стартовая вершина,
обнуление флага
                if ((graf[i].start_top == graf[j].end_top) && (graf[i].start_top !=
graf[j].start_top))
                {
                    flag_search = false;
                    break;
                }
            }
            if (flag_search && !graf[i].metka)
            {
                // если это первая начальная вершина
                if (first_top == -500)
                {
                    first_top = graf[i].start_top;
                }
                else
                {
                    // создание фиктивной вершины
                    Table *index = new Table;
                    index->start_top = -100;
                    index->end_top = first_top;
                    index->weight = 0;
                    graf.push_back(*index);

                    index->end_top = graf[i].start_top;
                    graf.push_back(*index);

                    delete index;
                    if (first_top != -100)
                    {
                        first_top = -100;
                    }
                }
                for (auto temp = 0; temp < graf.size(); temp++)
                {
                    if (graf[temp].start_top == graf[i].start_top)
                    {
                        graf[temp].metka = true;
                    }
                }
```

```
			}
		}
		for (auto i = 0; i < graf.size(); i++)// итерация по таблице вершин графа
		{
			graf[i].metka = false;// обнуление меток
		}
}

// метод поиска всех полных путей
void Graf::search_all_way()
{
		vector<int> way;// массив, содержащий текущий путь
		way.push_back(first_top);
		cout << "Полные пути СГ:" << endl;
		for (int i = 0; i < graf.size(); i++)
		{
			if (graf[i].start_top == first_top)
			{
				way.push_back(graf[i].end_top);// запоминание след вершины в массиве путей
				rekurs_search(&way);// рекурсивный вызов функции поиска путей по стеку
			}
		}

}

// рекурсивный метод поиска маршрута
void Graf::rekurs_search(vector<int> *way)
{
		// условия нахождения полного пути СГ
		if (way->back() == last_top)
		{
			for (auto i = way->begin(); i != way->end(); i++)
			{
				cout << *i << " ";// вывод пути
			}
			cout << endl;
			way->pop_back();// удаление послед вершины и возврат вверх по стеку
			return;
		}

		for (int i = 0; i < graf.size(); i++)
		{
			if (graf[i].start_top == way->back())// условие нахождения след вершины
			{
				way->push_back(graf[i].end_top);// запоминание вершины
				rekurs_search(way);// очередной вызов функции
			}
		}
		way->pop_back();
		return;
}

void Graf::graf_struct()// метод упорядочивания СГ
{
		vector<int> top;// очередь для хранения вершин СГ
		top.push_back(first_top);// положили в очередь первую вершину
		int size = graf.size();// запоминание размера таблицы для оптимизации программы
		while (struct_graf.size() != size)// пока новый СГ не сформирован
		{
			for (int i = 0; i < size; i++)// цикл по элементам таблицы СГ
```

```
		{
			if (top.at(0) == graf.at(i).start_top&&!graf.at(i).metka)// если найдена следующая вершина и данная работа не посещена
			{
				struct_graf.push_back(graf.at(i));//  добавление  текущей  строки таблицы в новую таблицу СГ
				top.push_back(graf.at(i).end_top);// добавление в очередь конечной вершины текущей строки
				graf.at(i).metka = true;// работа посещена
			}

		}
		top.erase(top.begin());// после цикла удаление послед вершины
	}
}
```

Файл Main.cpp

```
#include "stdafx.h"
#include <iostream>
#include <Conio.h>
#include "Graf.h"
#include <Windows.h>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
	SetConsoleCP(1251);// подключение русскоязычного ввода/вывода
	SetConsoleOutputCP(1251);
	system("color F0");
	ifstream fin("input.dat");// чтение данных из файла input.dat
	if (fin.is_open() == false)
	{
		cout << "Ошибка открытия файла\nзавершение работы" << endl;
		_getch();
		return 1;
	}
	Graf graf(&fin);// создание и инициализация СГ
	fin.close();

	graf.print(0);// вывод таблицы работ
	graf.optimized_graf();// оптимизация СГ, построение частично структурированной таблицы
	graf.print(1);// вывод таблицы работ
	graf.search_all_way();// поиск всех полных путей СГ
	_getch();
	return 0;
}
```

3. Тесты программы

1)

```
Таблица П1:
**********************************
*     A    *     B    *     T    *
**********************************
*        10*         9*         1*
**********************************
*        10*         8*         1*
**********************************
*         5*         4*         1*
**********************************
*         5*         3*         1*
**********************************
*         4*         8*         1*
**********************************
*         3*         9*         1*
**********************************
Вершины графа (события) с шифрами 9 и 8 конечные.
Ввести фиктивную конечную вершину или, если вершина уже фиктивна, добавить работу?(1-да, 2-нет): 1
Таблица П2:
**********************************
*      A    *      B    *      T    *
**********************************
*       -100*         10*          0*
**********************************
*       -100*          5*          0*
**********************************
*         10*          9*          1*
**********************************
*         10*          8*          1*
**********************************
*          5*          4*          1*
**********************************
*          5*          3*          1*
**********************************
*          9*        100*          0*
**********************************
*          8*        100*          0*
**********************************
*          4*          8*          1*
**********************************
*          3*          9*          1*
**********************************

Начальная вершина: -100
Конечная вершина: 100

Полные пути СГ:
-100 10 9 100
-100 10 8 100
-100 5 4 8 100
-100 5 3 9 100
_
```

2)

```
Таблица П1:
**********************************
*     A    *     B    *     T    *
**********************************
*          1*          2*          5*
**********************************
*          2*          4*          7*
**********************************
*          1*          3*          6*
**********************************
*          3*          5*          8*
**********************************
Вершины графа (события) с шифрами 4 и 5 конечные.
Ввести фиктивную конечную вершину или, если вершина уже фиктивна, добавить работу?(1-да, 2-нет): 2
Вершины графа (события) с шифрами 4 и 3 конечные.
Ввести фиктивную конечную вершину или, если вершина уже фиктивна, добавить работу?(1-да, 2-нет): 2
Таблица П2:
**********************************
*     A    *     B    *     T    *
**********************************
*          1*          2*          5*
**********************************
*          2*          4*          7*
**********************************

Начальная вершина: 1
Конечная вершина: 4

Полные пути СГ:
1 2 4
```

3)

```
Таблица П1:
**********************************
*     A    *     B    *     T    *
**********************************
*         1*         2*        10*
**********************************
*         2*         3*         5*
**********************************
*         2*         4*         1*
**********************************
*         1*         6*         4*
**********************************
*         6*         7*         1*
**********************************
*         4*         6*         1*
**********************************
*         4*         5*         4*
**********************************
*         3*         9*         4*
**********************************
*         5*         9*         1*
**********************************
*         7*         5*         1*
**********************************
*         7*         8*         1*
**********************************
Вершины графа (события) с шифрами 9 и 8 конечные.
Ввести фиктивную конечную вершину или, если вершина уже фиктивна, добавить работу?(1-да, 2-нет): 1
```

```
Таблица П2:
**************************************
*     A     *     B     *     T     *
**************************************
*          1*          2*         10*
**************************************
*          1*          6*          4*
**************************************
*          2*          3*          5*
**************************************
*          2*          4*          1*
**************************************
*          6*          7*          1*
**************************************
*          3*          9*          4*
**************************************
*          4*          6*          1*
**************************************
*          4*          5*          4*
**************************************
*          7*          5*          1*
**************************************
*          7*          8*          1*
**************************************
*          9*        100*          0*
**************************************
*          5*          9*          1*
**************************************
*          8*        100*          0*
**************************************


Начальная вершина: 1
Конечная вершина: 100

Полные пути СГ:
1 2 3 9 100
1 2 4 6 7 5 9 100
1 2 4 6 7 8 100
1 2 4 5 9 100
1 6 7 5 9 100
1 6 7 8 100
```

4)

```
Таблица П1:
**********************************
*     A    *     B    *     T    *
**********************************
*         1*         2*         2*
**********************************
*         1*         2*         1*
**********************************
*         1*         6*         1*
**********************************
*         2*         3*         1*
**********************************
*         3*         6*         1*
**********************************
*         6*         5*         1*
**********************************
*         3*         4*         1*
**********************************
*         3*         4*         1*
**********************************
*         4*         4*         1*
**********************************
ошибка, работа 1->2 дублируется
но имеет 2 веса:
1) 2
2) 1
выбрать нужное ( по умолчанию удалиться 2) ): 1
ошибка, работа 3->4 дублируется
с одним весом: 1
автоматическое удаление одной работы
Найдена петля, автоматическое удаление работы
Вершины графа (события) с шифрами 5 и 4 конечные.
Ввести фиктивную конечную вершину или, если вершина уже фиктивна, добавить работу?(1-да, 2-нет): 1
```

```
Таблица П2:
**************************************
*      A     *      B     *      T     *
**************************************
*           1*           2*           1*
**************************************
*           1*           6*           1*
**************************************
*           2*           3*           1*
**************************************
*           6*           5*           1*
**************************************
*           3*           6*           1*
**************************************
*           3*           4*           1*
**************************************
*           5*         100*           0*
**************************************
*           4*         100*           0*
**************************************

Начальная вершина: 1
Конечная вершина: 100

Полные пути СГ:
1 2 3 6 5 100
1 2 3 4 100
1 6 5 100
```

5)

```
Таблица П1:
************************************
*     A    *     B    *     T    *
************************************
*          1*          2*         15*
************************************
*          2*          3*         10*
************************************
*          2*          3*         20*
************************************
*          1*          4*          9*
************************************
*          4*          2*          5*
************************************
*          4*          3*          4*
************************************
*          3*          5*          1*
************************************
ошибка, работа 2->3 дублируется
но имеет 2 веса:
1) 10
2) 20
выбрать нужное ( по умолчанию удалиться 2) ): 2
Таблица П2:
************************************
*     A    *     B    *     T    *
************************************
*          1*          2*         15*
************************************
*          1*          4*          9*
************************************
*          2*          3*         10*
************************************
*          4*          2*          5*
************************************
*          4*          3*          4*
************************************
*          3*          5*          1*
************************************

Начальная вершина: 1
Конечная вершина: 5

Полные пути СГ:
1 2 3 5
1 4 2 3 5
1 4 3 5
```