Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)
Факультет «Робототехника и комплексная автоматизация» (РК)
Кафедра «Системы автоматизированного проектирования» (РК6)

Отчет по практике по НИРС на тему «Метод критического пути»

Студент: А.Р.Василян

Группа: РК6-43Б

Научный руководитель: Берчун Ю.В.

Дата:

Москва 2021

Цель:
Разработать программную реализацию поиска критического пути проекта(в управленческой деятельности).

Входные данные:
На вход программа получает файл с информацией по каждой работе в виде: наименование работы, длительность её выполнения и наименования предшественников.

Алгоритм:
Вначале программа открывает файл input.txt с входными данными. Если такого файла нет, то выводит ошибку. Далее в цикле из этого файла берётся по одному слову и символу после этого слова. Если этот символ не переход на следующую строку, то слова до этого символа и следующее за ним слово будут считаться одной строкой из нескольких слов, которые относятся к одной и той же задаче. Если же находится переход на следующую строку, то все записанные слова даются на вход конструктору класса Task, а полученный объект класса Task даётся на вход методу new_task класса Project, который добавит этот объект в вектор, находящийся в приватном поле объекта project типа Project, а также этот метод приватные поля prev_index и next_index индексами предшествующих задач и следующих задач соответственно.

По итогу цикла у нас есть объект класса Project project, заполненный данными каждой задачи проекта из файла. Затем вызываются функции find_est_eft(), find_lst_lft() и find_slk(), которые вычислят для каждой задачи ранние сроки(est, eft), поздние сроки (lst, lft), резерв (slk). Все вычисленные данные выводятся.

Далее с помощью метода find_critical_path выводятся все задачи, которые входят в какой-либо критический путь.

И в самом конце файл закрывается, и программа заканчивает работу.

Выходные данные:
Программа по окончанию работы выводит в поток стандартного вывода est, eft, lst, lft, slk каждой задачи, а также все задачи, которые входят в какой-либо критический путь.

Текст программы:
task,hpp

```
#ifndef TASK
#define TASK

#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Task
{
private:
    string name;    // наименование задачи
    int comp_time;  // время выполнения
```

```
    std::vector<std::string> prev; // предшественники
    int est;
    int eft;
    int lst;
    int lft;
    int slk;
    std::vector<int> prev_index;
    std::vector<int> next_index;
    int output;
public:
    Task(vector <string>&);
    void new_prev_i(int);
    void new_next_i(int);

    int& get_est() {return est;};
    int& get_eft() {return eft;};
    int& get_lst() {return lst;};
    int& get_lft() {return lft;};
    int& get_slk() {return slk;};
    int& get_output() {return output;};

    string get_name1() {return name;};
    int get_comp_time1() {return comp_time;};
    int get_q1() {return prev.size();};
    string get_prev1(int i) {return prev[i];};
    int get_prev_index1(int pi) {return prev_index[pi];};
    int get_next_index1(int ni) {return next_index[ni];};
    int get_next_q1() {return next_index.size();};
    int get_est1() {return est;};
    int get_eft1() {return eft;};
    int get_lst1() {return lst;};
    int get_lft1() {return lft;};
    int get_slk1() {return slk;};
    int get_output1() {return output;};
};

#endif
```

project.hpp

```
#ifndef PROJECT
#define PROJECT

#include <iostream>
#include <string>
#include "task.hpp"
#include <vector>
using namespace std;

class Project
{
private:
    std::vector<Task> project;
```

```
public:
    int get_size() {return project.size();};
    Task get_task(int i) {return project[i];};
    void new_task(Task task_n);
    void find_est_eft();
    void find_lst_lft();
    void find_slk();
    void find_critical_path();
};
#endif
```

task.cpp

```
#include "task.hpp"
#include <stdlib.h>
#include <vector>
#include <string>

Task::Task(vector <string>& f_input)
{
    name = f_input[0];
    comp_time = atoi(f_input[1].c_str());
    for (int i = 2; i < f_input.size(); i++)
    {
            prev.push_back(f_input[i]);
    }
    est = 0;
    eft = 0;
    lst = 0;
    lft = 0;
    slk = 0;
    output=0;
}

void Task::new_prev_i(int pi)
{
     prev_index.push_back(pi);
}

void Task::new_next_i(int ni)
{
     next_index.push_back(ni);
}
```

project.cpp

```
#include "task.hpp"
#include "project.hpp"
#include <vector>

void Project::new_task(Task task_n)
{
    project.push_back(task_n);
    for (int i = 0; i < project.size()-1; i++)
```

```
		for(int k=0; k < project[project.size()-1].get_q1(); k++)
		{
			if(project[project.size()-1].get_prev1(k) == project[i].get_name1())
			{
				project[project.size()-1].new_prev_i(i);
				project[i].new_next_i(project.size()-1);
			}
		}
}

void Project::find_est_eft ()
{
    project[0].get_eft() = project[0].get_comp_time1();
    int max=0;

    for (int i = 1; i < project.size(); i++)
    {
		for(int k=0; k < project[i].get_q1(); k++)
		{
			if(project[project[i].get_prev_index1(k)].get_eft1() >= max)
			{
				max=project[project[i].get_prev_index1(k)].get_eft1();
			}
		}
		project[i].get_est() = max;
		project[i].get_eft() = max + project[i].get_comp_time1();
		max=0;
    }
}


void Project::find_lst_lft ()
{
    project[project.size()-1].get_lft()=project[project.size()-1].get_eft1();
    project[project.size()-1].get_lst()=project[project.size()-1].get_lft1() - project[project.size()-
1].get_comp_time1();
    int min = project[project.size()-1].get_lft1();
    project[project.size()-1].get_output()=1;

    for (int i = project.size()-2; i >=0; i--)
    {
		for(int k=0; k < project[i].get_next_q1(); k++)
		{
			if(project[project[i].get_next_index1(k)].get_lst1()<=min)
			{
				min = project[project[i].get_next_index1(k)].get_lst1();
				if (project[project[i].get_next_index1(k)].get_output1() == 1)
					project[i].get_output() = 1;
			}
		}
		if (min == project[project.size()-1].get_lft1())
			min = project[project.size()-1].get_eft1();;
```

```
            project[i].get_lft() = min;
            project[i].get_lst() = min - project[i].get_comp_time1();
            min = project[project.size()-1].get_lft1();
    }
}

void Project::find_slk()
{
    for (int i = 0; i < project.size(); i++)
    {
            project[i].get_slk() = project[i].get_lst1() - project[i].get_est1();
    }
}

void Project::find_critical_path()
{
    for (int i = 0; i < project.size(); i++)
            if(project[i].get_output1() == 1)
                    cout<<project[i].get_name1()<<endl;
}
```

crit_path.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "task.hpp"
#include "project.hpp"

using namespace std;

int main()
{

    setlocale(LC_ALL, "ru");

    string path = "input.txt";
    ifstream fin;
    fin.open(path);
    Project project;

    if (!fin.is_open())
    {
            cout<<"Ошибка открытия файла."<<endl;
    }
    else
    {
            cout<<"Файл открыт."<<endl;
            std::string str;
            char str1;
            std::vector<std::string> f_input;

            while (!fin.eof())
```

```
		{
			str = "";
			fin >> str;
			fin.get(str1);
			cout<<str;
			cout<<str1;
			if (str !="")
			{
				if (str1 != '\n')
					f_input.push_back(str);
				else
				{
					f_input.push_back(str);
					project.new_task(Task (f_input));
					f_input.clear();
				}
			}
		}

	}

	project.find_est_eft();
	project.find_lst_lft();
	project.find_slk();

	for (int i = 0; i < project.get_size();i++)
	{
		cout<<project.get_task(i).get_name1()<<endl;
		cout<<"est="<<project.get_task(i).get_est1()<<endl;
		cout<<"eft="<<project.get_task(i).get_eft1()<<endl;
		cout<<"lst="<<project.get_task(i).get_lst1()<<endl;
		cout<<"lft="<<project.get_task(i).get_lft1()<<endl;
		cout<<"slk="<<project.get_task(i).get_slk1()<<endl;
		cout<<endl;

	}

	cout<<"Задачи, принадлежашие критическим путям :"<<endl;
	project.find_critical_path();

	fin.close();
	return 0;
}
```

Вывод программы:

В файле:

```
task1 1
task2 3 task1
task3 4 task2
task4 2 task1
taskb 2 task4
task5 2 task4
task6 1 task3 task5
```

Вывод:

Файл открыт.
task1 1
task2 3 task1
task3 4 task2
task4 2 task1
taskb 2 task4
task5 2 task4
task6 1 task3 task5

task1
est=0
eft=1
lst=0
lft=1
slk=0

task2
est=1
eft=4
lst=1
lft=4
slk=0

task3
est=4
eft=8
lst=4
lft=8
slk=0

task4
est=1
eft=3
lst=4
lft=6
slk=3

taskb
est=3
eft=5
lst=7
lft=9
slk=4

task5
est=3
eft=5
lst=6
lft=8
slk=3

task6
est=8
eft=9
lst=8
lft=9
slk=0

Задачи, принадлежашие критическим путям :
task1
task2
task3
task4
task5
task6

Используемая литература:

1. bigor.bmstu.ru – база и генератор образовательных ресурсов
2. Лекции и семинары по курсу объектно-ориентированное программирование
3. Прикладное программирование на языке C++: учебное пособие / Т.М. Волосатова, С.В. Родионов, Д.Т. Шварц. – Москва: Издательство МГТУ им. Н. Э. Баумана, 2015. – 146, [2] с.: ил.
4. Управление проектами. Фундаментальный курс Серия «Учебники Высшей школы экономики»