Министерство образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

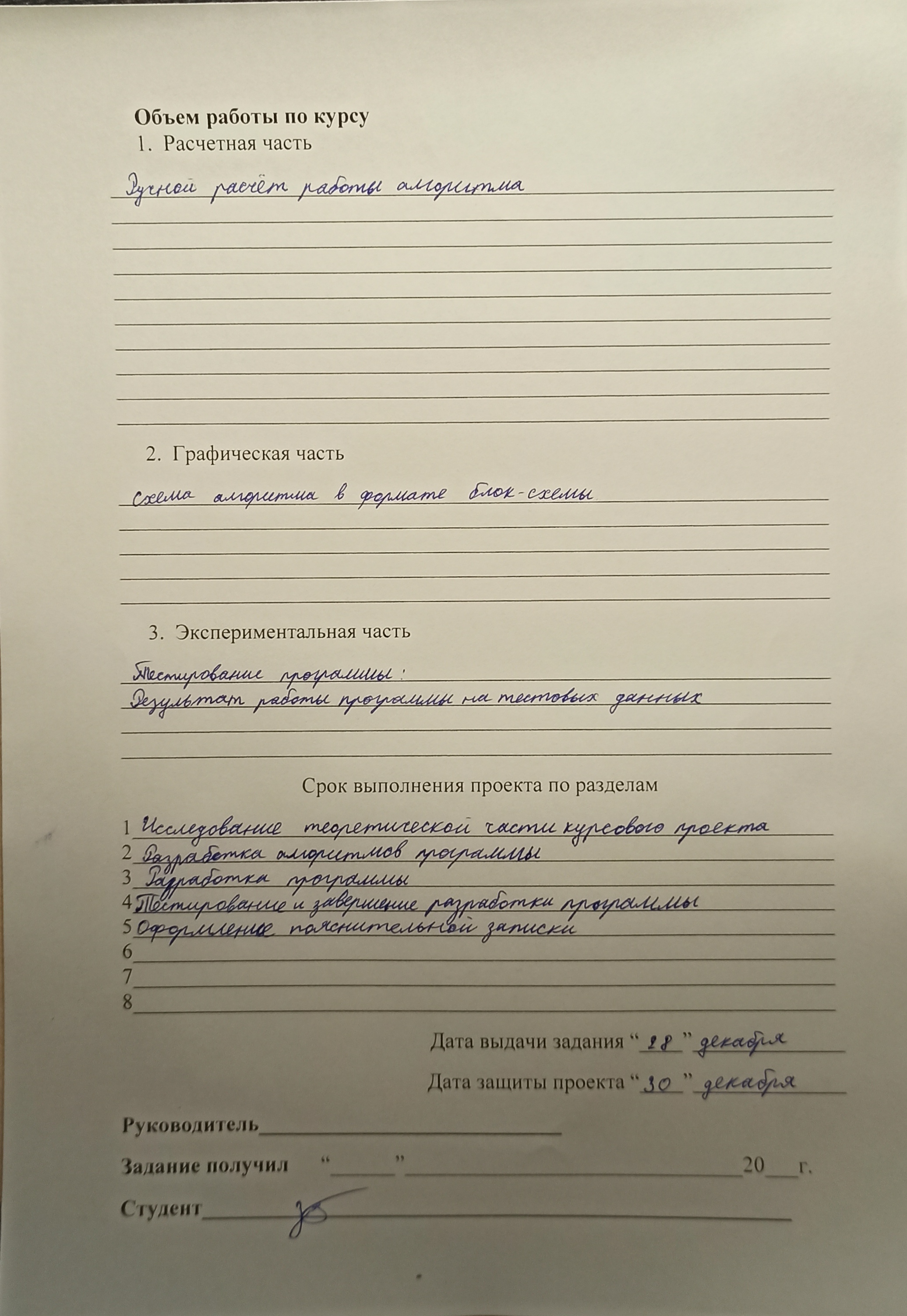
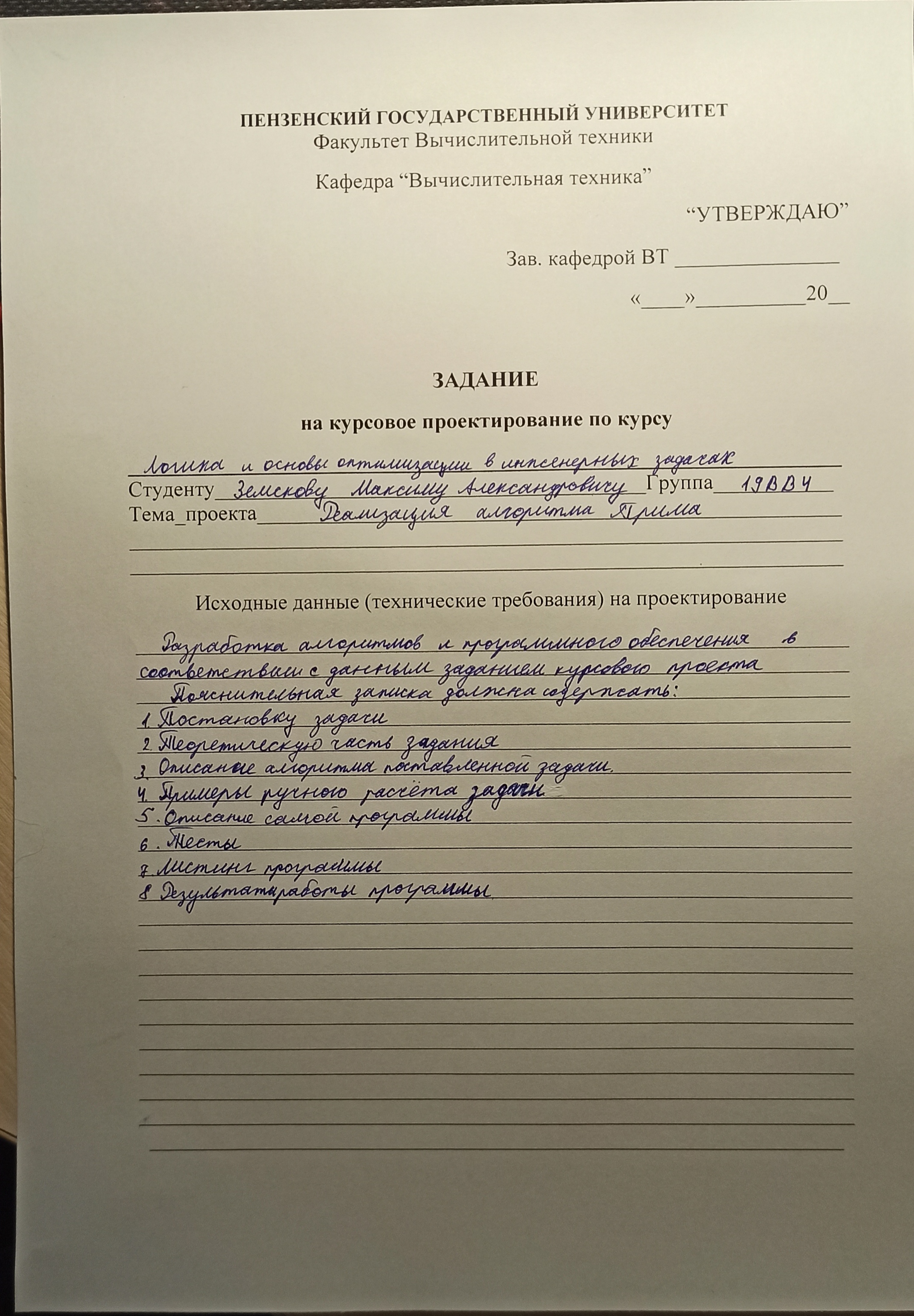
**Пояснительная записка**

к курсовой работе  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Реализация алгоритма Прима»

**Выполнил:**Студент группы 19ВВ4  
Земсков Максим

**Приняли:**Митрохин М.А.  
Юрова О.В.

Пенза 2020



**Содержание**

[Реферат 4](#_Toc59630631)

[Введение 5](#_Toc59630632)

[1. Постановка задачи 6](#_Toc59630633)

[2. Теоретическая часть задания 7](#_Toc59630634)

[3. Описание алгоритма программы 9](#_Toc59630635)

[4. Описание программы 10](#_Toc59630636)

[5. Результат работы программы 15](#_Toc59630637)

[6. Тестирование 16](#_Toc59630638)

[7. Ручной расчёт задачи 19](#_Toc59630639)

[Заключение 20](#_Toc59630640)

[Список литературы 21](#_Toc59630641)

[Приложение А. 22](#_Toc59630642)

Реферат

Отчёт 26 стр, 9 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, НЕОРИЕНТИРОВАННЫЙ ГРАФ, ПОИСК ОСТОВНОГО ДЕРЕВА.

Цель исследования – разработка программы, способная находить остовное дерево, используя алгоритм Прима.

В работе рассмотрены принципы работы алгоритма Примы, нахождение расстояний между вершинами графов разных размеров и типов.

Введение

Алгоритм Прима - это алгоритм минимального остовного дерева, что принимает граф в качестве входных данных и находит подмножество ребер этого графа, который формирует дерево, включающее в себя каждую вершину, а также имеет минимальную сумму весов среди всех деревьев, которые могут быть сформированы из графа.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2019, язык программирования – Си++.

Целью данной курсовой работы является разработка программы на языке Си++, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм Прима, интерфейс и графическое отображение результатов работы программы.

1. Постановка задачи

Требуется разработать программу, которая определит минимальное остовное дерево графа, используя алгоритм Прима.

Исходный взвешенный граф в программе должен задаваться матрицей смежности, причём программа должна генерировать данные в зависимости от выбранных параметров графа. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности и выбирал произвольную генерацию матрицы или ввод с клавиатуры. После обработки этих данных на экран должна выводиться матрица смежности графа и своеобразная таблица, в которой отображаются кратчайшие расстояния между вершинами. Результат работы программы должен дополнительно выводиться в файл. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

1. Теоретическая часть задания

Граф называется *взвешенным* (*нагруженным*), если каждому его ребру поставлено в соответствие некое значение – вес или длина ребра.

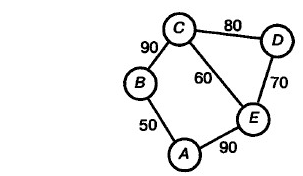
****

Рисунок 1 — Пример взвешенного графа

Длина пути во взвешенном связном графе — это сумма длин (весов) тех рёбер, из которых состоит путь. Тогда можно поставить задачу нахождения кратчайшего пути в графе. Такая задача возникает при оптимизации транспортных путей.

Неориентированный граф – граф, рёбра которого не имеют определённого направления.

Ориентированный граф – граф, рёбра которого имеют определённое направление.

При представлении графа матрицей смежности информация о рёбрах

графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается единицей, иначе нулём.

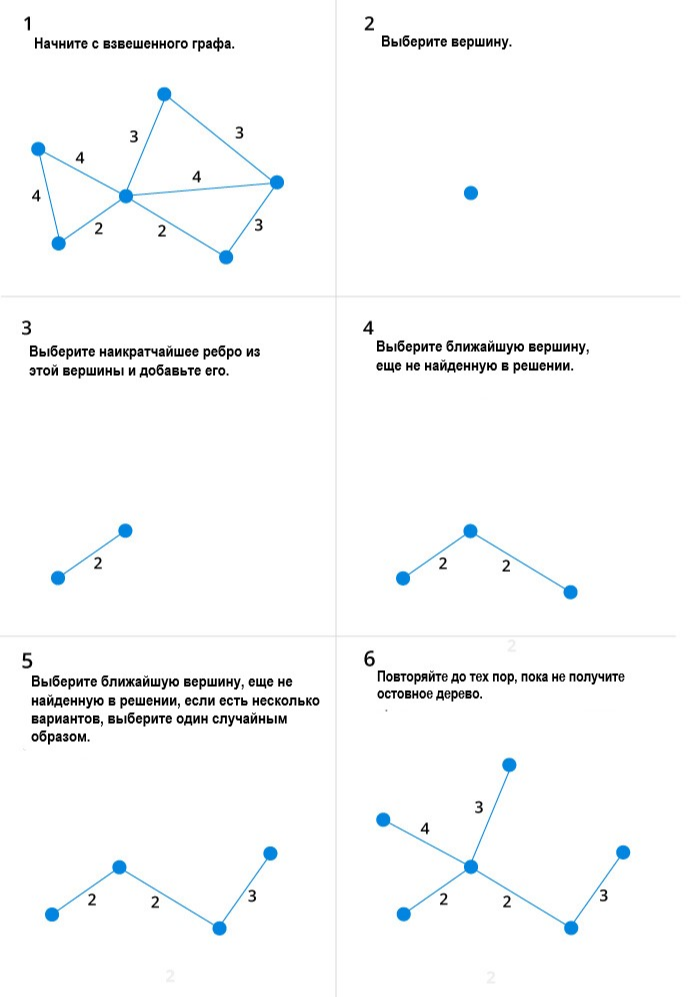
Пусть задан граф G= (V, A) и выбрана его вершина X1. Нужно найти кратчайшее расстояние от неё до остальных вершин. Изначально присваиваем вершине X1 метку равную 0, а всем остальным — равную бесконечности. Далее выбираем вершину X с наименьшей меткой (сейчас это X1). Перебираем все вершины, которые напрямую связаны с X и присваиваем им метки равные сумме метки X и расстоянию из X в выбранную вершину. Если предыдущее значение метки меньше этой суммы, то оно не изменяется. После рассмотрения всех этих вершин вершина X помечается как посещённая. И из не посещённых снова выбирается вершина с наименьшей меткой. Этот алгоритм повторяется, пока не останется не посещённых вершин. В конце метка каждой вершины будут равны расстоянию между ней и X1. Если до вершины нет пути из X1, метка останется равной бесконечности.

1. Описание алгоритма программы

Псевдокод для алгоритма Прима показывает, как мы создаем два набора вершин U и V-U. U содержит список вершин, которые были посещены, а V-U – список вершин, которые не были посещены. Один за другим мы перемещаем вершины из набора V-U в набор U, соединяя ребро с наименьшим весом.

Шаги для реализации алгоритма Прима следующие:

1. Инициализируйте минимальное остовное дерево с произвольно выбранной вершиной.
2. Найдите все ребра, которые соединяют дерево с новыми вершинами, найдите минимум и добавьте его в дерево.
3. Продолжайте повторять шаг 2, пока не получите минимальное остовное дерево.



1. Описание программы

Для написания данной программы использован язык программирования Си++. Язык программирования Си++ — универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

Для программной реализации алгоритма понадобилось 6 функций: memory(), matrix\_random(), matrix\_vvod(), matrix\_vivod(), function(), main().

Работа программы начинается с главного меню. Можно выбрать один из пунктов: Заполнение массива рандомными числами, заполнение массива с клавиатуры, результаты работы программы и выход (рис. 2).

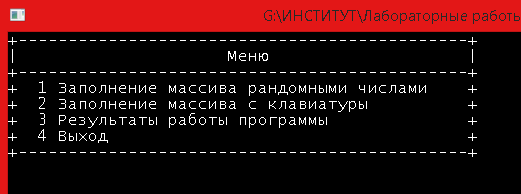


Рисунок 2 — Ввод параметров графа

Для реализации меню используется главная функция программы main.

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

setlocale(LC\_ALL, "Rus");

system("cls");

printf("+---------------------------------------------+\n");

printf("| Меню |\n");

printf("+---------------------------------------------+\n");

printf("+ 1 Заполнение массива рандомными числами +\n");

printf("+ 2 Заполнение массива с клавиатуры +\n");

printf("+ 3 Результаты работы программы +\n");

printf("+ 4 Выход +\n");

printf("+---------------------------------------------+\n");

scanf\_s("%d", &term);

while (term != 0)

{

if (term == 1)

{

system("cls");

memory();

matrix\_random();

function();

\_getch();

}

else if (term == 2)

{

system("cls");

memory();

matrix\_vvod();

function();

\_getch();

}

else if (term == 3)

{

system("cls");

char arr[N];

rez = fopen("Rezultat.txt", "r");

while (fgets(arr, N, rez) != NULL)

printf("%s", arr);

printf("\n");

fclose(rez);

\_getch();

}

else if (term == 4)

{

exit(0);

}

else {

printf("Введите значение от 1-4");

}

printf("+---------------------------------------------+\n");

printf("| Меню |\n");

printf("+---------------------------------------------+\n");

printf("+ 1 Заполнение массива рандомными числами +\n");

printf("+ 2 Заполнение массива с клавиатуры +\n");

printf("+ 3 Результаты работы программы +\n");

printf("+ 4 Выход +\n");

printf("+---------------------------------------------+\n");

scanf\_s("%d", &term);

}

return 0;

Для выделения памяти под динамичный двумерный массивисползуется функция memory.

printf("Размер матрицы:");

scanf\_s("%d", &V);

G = (int\*\*)malloc(V \* sizeof(int\*));

for (int i = 0; i < V; i++)

{

G[i] = (int\*)malloc(V \* sizeof(int)

}

Для рандомной генерации динамичного двумерного массива заданного размера используется функция matrix\_random.

srand(time(NULL));

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

G[i][j] = rand() % 20;

}

}

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

Для генерации динамичного двумерного массива заданного размера путем ввода значений с клавиатуры используется функция matrix\_vvod.

for (int i = 0; i < V; i++)

{

for (int j = i+1; j < V; j++)

{

printf("G[%d][%d] = ", i, j);

scanf\_s("%d", &G[i][j]);

}

}

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

Для вывода на экран сгенерированной матрицы используется функция matrix\_vivod.

rez = fopen("Rezultat.txt", "a");

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

printf("%2d ", G[i][j]);

fprintf(rez, "%2d ", G[i][j]);

}

printf("\n");

fprintf(rez, "\n");

}

fprintf(rez, "\n");

fclose(rez);

printf("\n");

Далее запускается основной цикл, который перебирает все вершины, определяя минимальные расстояния между вершинами с помощью функции function.

matrix\_vivod();

selected = (int\*)malloc(V \* sizeof(int));

for (int i = 0; i < V; i++) {

selected[i] = false;

}

no\_edge = 0;

selected[0] = true;

rez = fopen("Rezultat.txt", "a");

printf("Ребро : Вес ребра");

fprintf(rez, "Ребро : Вес ребра");

printf("\n");

fprintf(rez, "\n");

fclose(rez);

rez = fopen("Rezultat.txt", "a");

while (no\_edge < V - 1) {

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < V; i++) {

if (selected[i]) {

for (int j = 0; j < V; j++) {

if (!selected[j] && G[i][j]) {

if (min > G[i][j]) {

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d", x, y, G[x][y]);

fprintf(rez, "%d - %d : %d\n", x, y, G[x][y]);

printf("\n");

selected[y] = true;

no\_edge++;

}

fclose(rez);

rez = fopen("Rezultat.txt", "a");

fprintf(rez, "------------------------------------------------------------------------------------");

fprintf(rez, "\n");

fclose(rez);

free(G);

Также присутствует алгоритм вывода на экран результатов предыдущих опытов работы программы, путем считывания данных из файла. Реализован в функции main.

char arr[N];

rez = fopen("Rezultat.txt", "r");

while (fgets(arr, N, rez) != NULL)

printf("%s", arr);

printf("\n");

fclose(rez);

1. Результат работы программы

В консольное окно выводится текстовое отображение результатов работы программы: матрица смежности и таблица с кратчайшими расстояниями между вершинами (рис. 3).

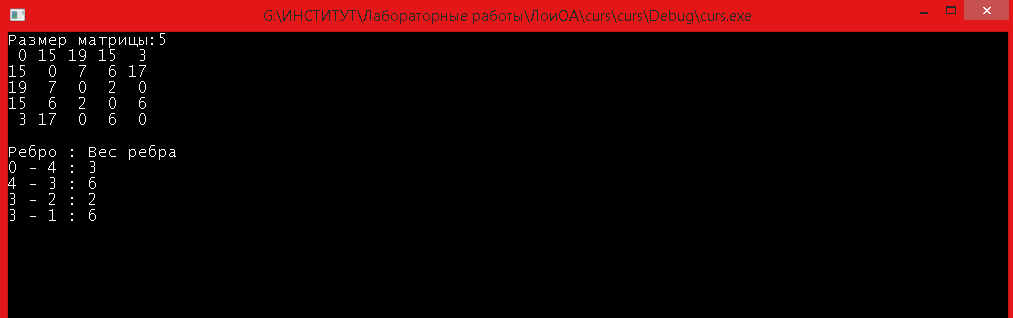


Рисунок 3 — Вывод результата работы программы

Результат всегда записывается в файл Rezultat.txt. Формат записи аналогичен выводу в консольное окно (рис. 4).

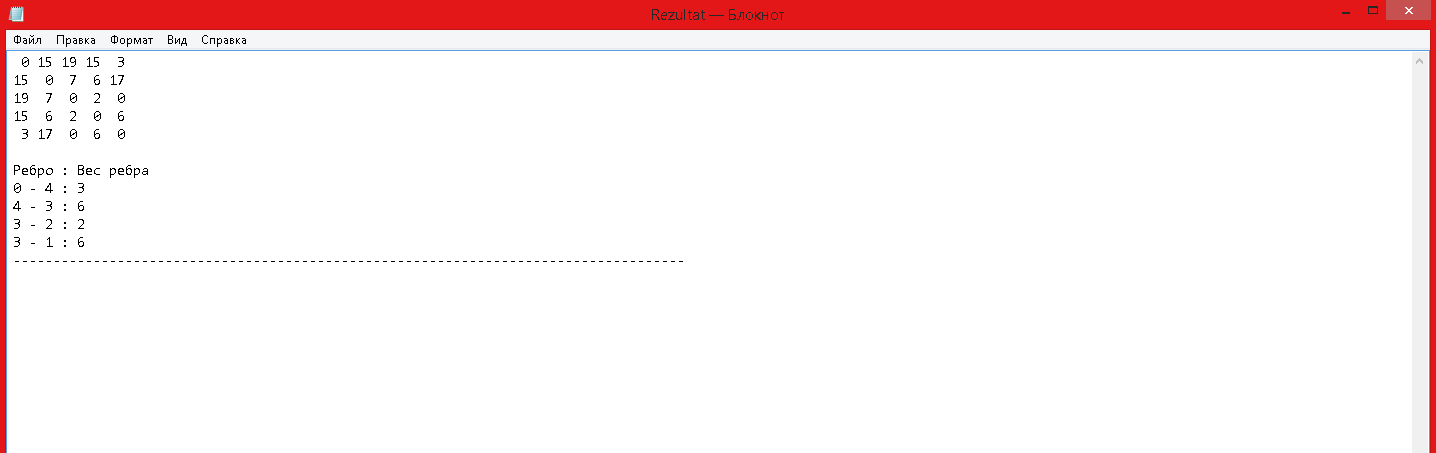


Рисунок 4 — Файл с результатом работы программы

1. Тестирование

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, выводом графики, алгоритмом программы, взаимодействием функций. Ниже продемонстрирован результат тестирования программы при вводе пользователем различных количеств вершин, с выбором способа заполнения графа.

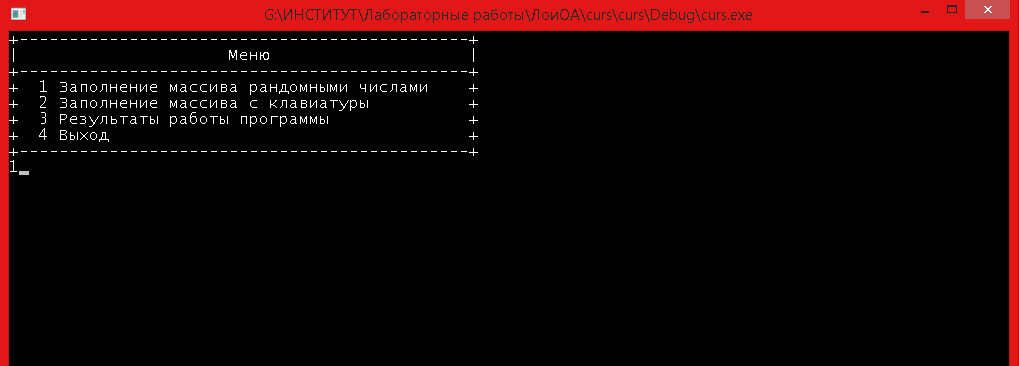


Рисунок 5 — Тестирование: запуск программы, навигация по меню, выбор способа заполнения матрицы смежности

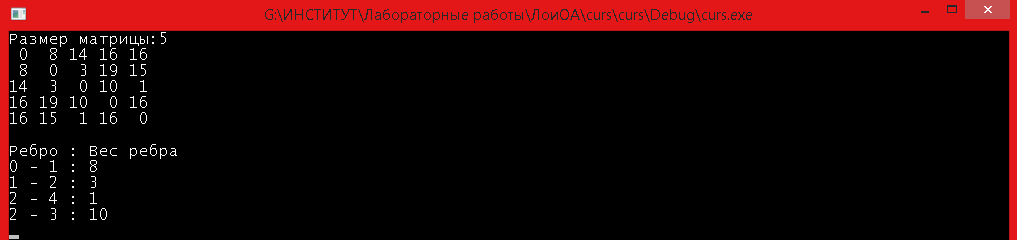


Рисунок 6 — Тестирование: заполнение матрицы рандомными значениями

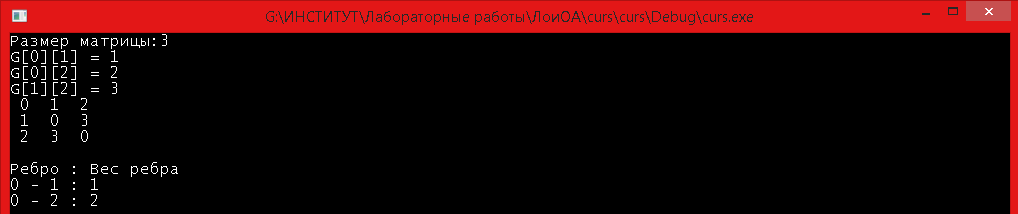


Рисунок 7 — Тестирование: заполнение матрицы вводимыми значениями



Рисунок 8 — Тестирование: вывод данных предыдущих опытов

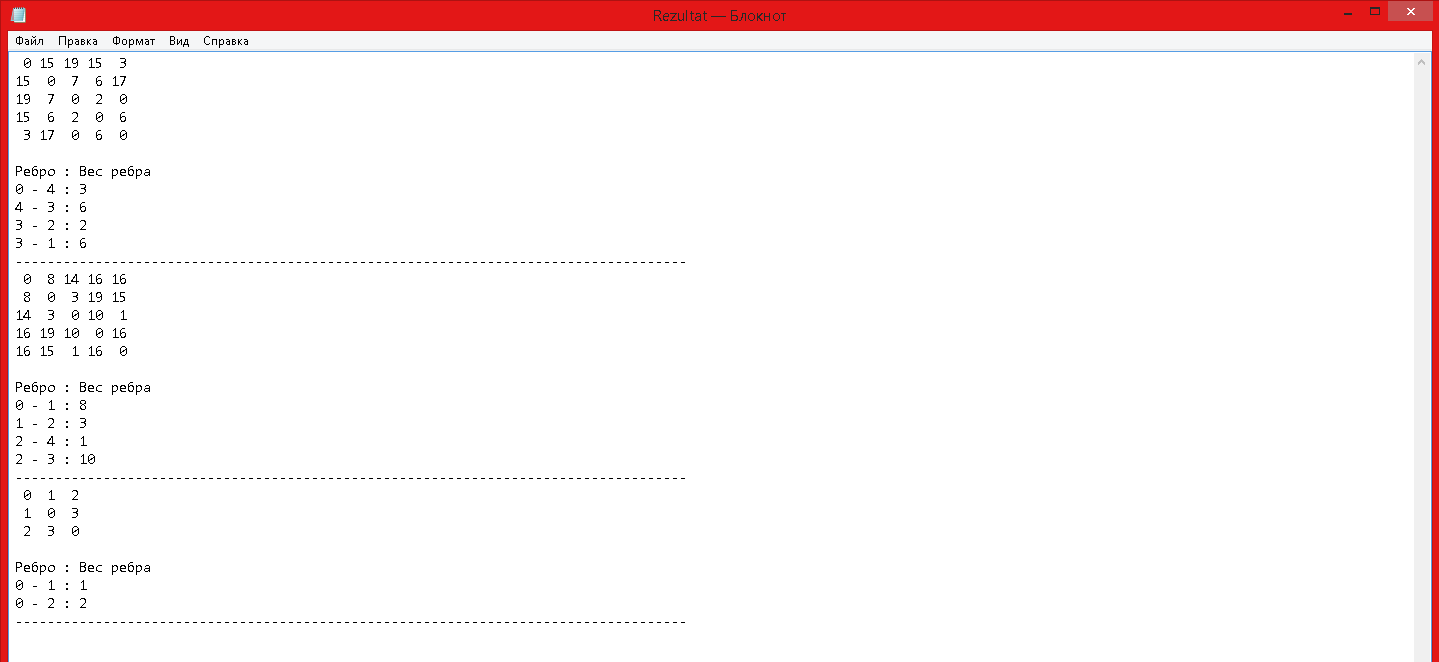


Рисунок 9 — Тестирование: вывод данных предыдущих опытов (файл)

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод сообщения о выборе количества вершин | Верно |
| Расчёт кратчайших расстояний | На выходе Получены корректные значения | Верно |
| Вывод результатов работы программы в файл | В файле корректно отображается результат работы программы | Верно |

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

1. Ручной расчёт задачи

Проведём проверку программы посредством ручных вычислений на примере графа с 5 вершинами (рис. 6).

Начнём рассчитывать расстояния от вершины 0. Ближайшей из соседних является вершина 1, расстояние до неё равно 8.

К вершине 1 ближайшей является вершина 2, расстояние до неё равно 3.

Переходим к вершине 2. Из не посещённый самой близкой является вершина 4, расстояние до неё равно 1.

Также от этой вершины проходит путь до не посещенной вершины 3, расстояние до неё равно 10.

Результат совпал с результатом работы программы. Таким образом можно сделать вывод, что программа работает верно.

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Примы для поиска остовного дерева графа MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приёмы создания матриц смежностей, основанных на теории графов. Приобретены навыки по осуществлению алгоритма Примы. Углублены знания языка программирования Cи++.

Недостатком разработанной программы является примитивный пользовательский интерфейс и не самый удобный формат вывода результата работы программы.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ - М.: МЦНМО, 2001. - 960 с.

2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978

3. Герберт Шилдт «Полный справочник по C++» - Вильямс, 2006

4. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977. 208 с.

5. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. 2009 г.

6. 3. Оре О. Графы и их применение: Пер. с англ. 1965. 176 с.

Приложение А.

Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>

#include <time.h>

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <windows.h>

#define INF 9999999

#define N 1000

//#define V 5

using namespace std;

FILE\* rez;

int V = 0;

int\*\* G;

int\* selected;

int no\_edge;

int x;

int y;

int term;

void memory() {

printf("Размер матрицы:");

scanf\_s("%d", &V);

G = (int\*\*)malloc(V \* sizeof(int\*));

for (int i = 0; i < V; i++)

{

G[i] = (int\*)malloc(V \* sizeof(int));

}

}

void matrix\_random() {

srand(time(NULL));

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

G[i][j] = rand() % 20;

}

}

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

}

void matrix\_vvod() {

for (int i = 0; i < V; i++)

{

for (int j = i+1; j < V; j++)

{

printf("G[%d][%d] = ", i, j);

scanf\_s("%d", &G[i][j]);

}

}

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

}

void matrix\_vivod() {

rez = fopen("Rezultat.txt", "a");

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

printf("%2d ", G[i][j]);

fprintf(rez, "%2d ", G[i][j]);

}

printf("\n");

fprintf(rez, "\n");

}

fprintf(rez, "\n");

fclose(rez);

printf("\n");

}

void function() {

matrix\_vivod();

selected = (int\*)malloc(V \* sizeof(int));

for (int i = 0; i < V; i++) {

selected[i] = false;

}

no\_edge = 0;

selected[0] = true;

rez = fopen("Rezultat.txt", "a");

printf("Ребро : Вес ребра");

fprintf(rez, "Ребро : Вес ребра");

printf("\n");

fprintf(rez, "\n");

fclose(rez);

rez = fopen("Rezultat.txt", "a");

while (no\_edge < V - 1) {

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < V; i++) {

if (selected[i]) {

for (int j = 0; j < V; j++) {

if (!selected[j] && G[i][j]) { // not in selected and there is an edge

if (min > G[i][j]) {

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d", x, y, G[x][y]);

fprintf(rez, "%d - %d : %d\n", x, y, G[x][y]);

printf("\n");

selected[y] = true;

no\_edge++;

}

fclose(rez);

rez = fopen("Rezultat.txt", "a");

fprintf(rez, "------------------------------------------------------------------------------------");

fprintf(rez, "\n");

fclose(rez);

free(G);

}

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

setlocale(LC\_ALL, "Rus");

system("cls");

printf("+---------------------------------------------+\n");

printf("| Меню |\n");

printf("+---------------------------------------------+\n");

printf("+ 1 Заполнение массива рандомными числами +\n");

printf("+ 2 Заполнение массива с клавиатуры +\n");

printf("+ 3 Результаты работы программы +\n");

printf("+ 4 Выход +\n");

printf("+---------------------------------------------+\n");

scanf\_s("%d", &term);

while (term != 0)

{

if (term == 1)

{

system("cls");

memory();

matrix\_random();

function();

\_getch();

}

else if (term == 2)

{

system("cls");

memory();

matrix\_vvod();

function();

\_getch();

}

else if (term == 3)

{

system("cls");

char arr[N];

rez = fopen("Rezultat.txt", "r");

while (fgets(arr, N, rez) != NULL)

printf("%s", arr);

printf("\n");

fclose(rez);

\_getch();

}

else if (term == 4)

{

exit(0);

}

else {

printf("Введите значение от 1-4");

}

printf("+---------------------------------------------+\n");

printf("| Меню |\n");

printf("+---------------------------------------------+\n");

printf("+ 1 Заполнение массива рандомными числами +\n");

printf("+ 2 Заполнение массива с клавиатуры +\n");

printf("+ 3 Результаты работы программы +\n");

printf("+ 4 Выход +\n");

printf("+---------------------------------------------+\n");

scanf\_s("%d", &term);

}

return 0;

}