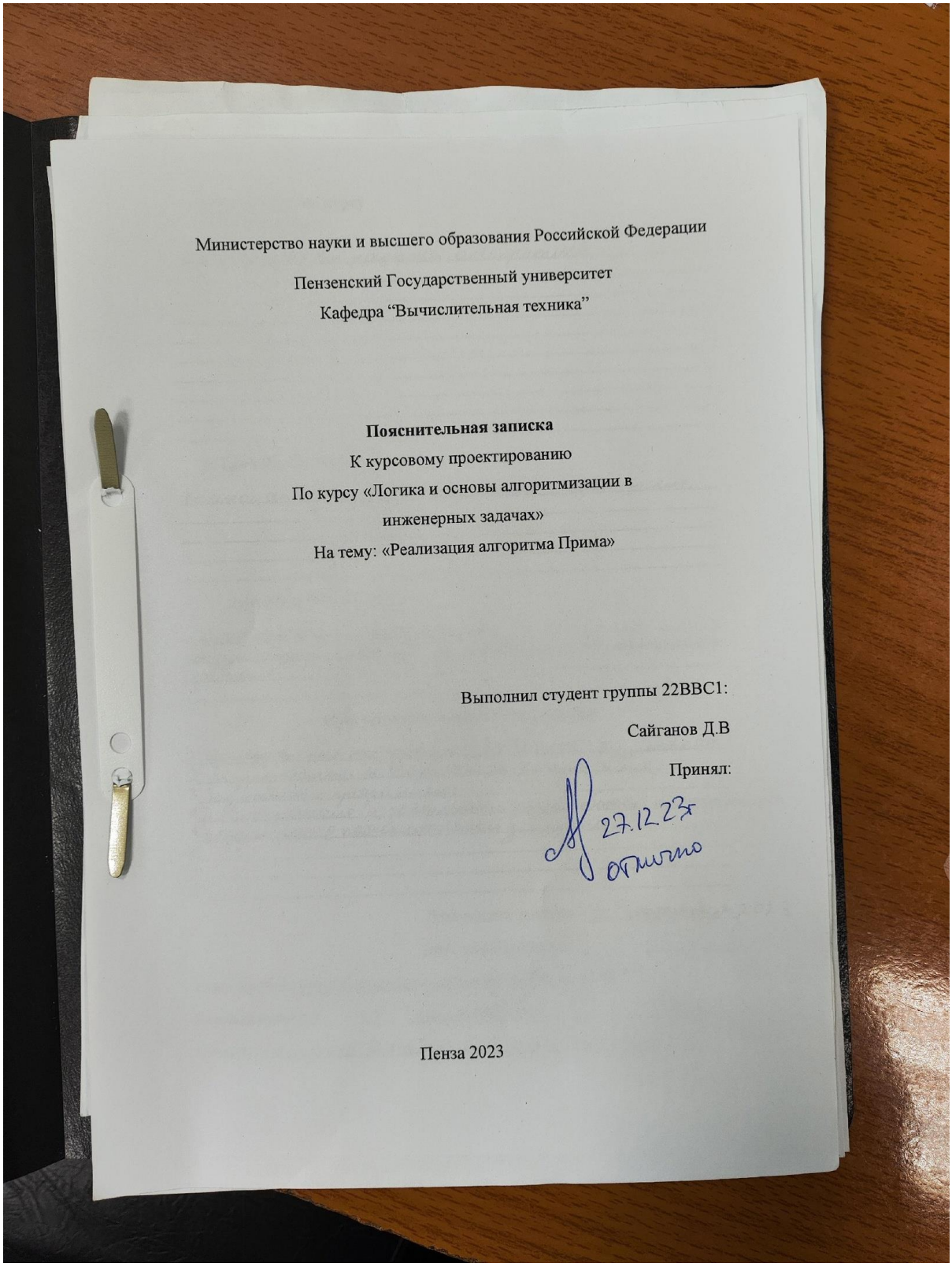

Министерство науки и высшего образования Российской Федерации

Пензенский Государственный университет

Кафедра “Вычислительная техника”

Пояснительная записка

К курсовому проектированию

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему: «Реализация алгоритма Прима»

Выполнил студент группы 22ВВС1:

Сайганов Д.В

Принял:

27.12.23

отлично

Пенза 2023

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ _____

«___»_____20__

ЗАДАНИЕ

на курсовое проектирование по курсу

"Логики и основы алгоритмизации в инженерных задачах"
Студенту Сайганову Дмитрию Владимировичу Группа 22ВВС1
Тема_проекта Реализация алгоритмов приема

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с заданием курсового проекта.
Пояснительная записка должна содержать:
1. Постановку задачи
2. Теоретическую часть
3. Описание алгоритма поставленной задачи
4. Пример ручного расчёта задачи и вычислений (на небольшом участке работы алгоритма)
5. Описание самой программы
6. Тесты
7. Список литературы
8. Листинг программы
9. Результаты работы программы

Объем работы по курсу

1. Расчетная часть

ручной расчёт работы алгоритма

2. Графическая часть

схема алгоритма в формате блок-схем.

3. Экспериментальная часть

Тестирование программы;
безультатная работа программы на тестовых данных.

Срок выполнения проекта по разделам

1 Исследование теоретической части курсового
2 Разработка алгоритмов программы.
3 Разработка программы
4 Тестирование и завершение разработки программы
5 оформление пояснительной записки
6
7
8

Дата выдачи задания "6" сентября 2023

Дата защиты проекта "____" _____

Руководитель Юрова Ольга Викторовна

Задание получил "26" сентября 2023г.

Студент Сайганов Дмитрий Владимирович

Министерство науки и высшего образования Российской Федерации

Пензенский Государственный университет

Кафедра “Вычислительная техника”

Пояснительная записка

К курсовому проектированию

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему: «Реализация алгоритма Прима»

Выполнил студент группы 22ВВС1:

Сайганов Д.В

Принял:

Пенза 2023

Содержание

Реферат

Отчет 34 стр,9 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, АЛГОРИТМ ПРИМА.

Цель исследования – реализация алгоритма Прима для нахождения минимального остовного дерева.

Введение

Алгоритм Прима — алгоритм построения минимального остовного дерева взвешенного связного неориентированного графа. Алгоритм впервые был открыт в 1930 году чешским математиком Войцехом Ярником, позже переоткрытРобертом Примом в 1957 году, и, независимо от них, Э.Дейкстрой в 1959 году. На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Результатом работы алгоритма является остовное дерево минимальной стоимости.

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2019, 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.

1. Постановка задачи

Требуется разработать программу которая осуществит поиск минимального остовного дерева взвешенного неориентированного графа.

Исходный граф в программе должен задаваться матрицей смежности. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

2. Теоретическая часть задания

Граф Graph (рисунок 1) задается множеством вершин X1, X2, …, Xn и множеством ребер, соединяющих между собой определенные вершины. Ребра из множества A неориентированы, что обозначается обычной линией на графе, которая показывает достижимость данной вершины.

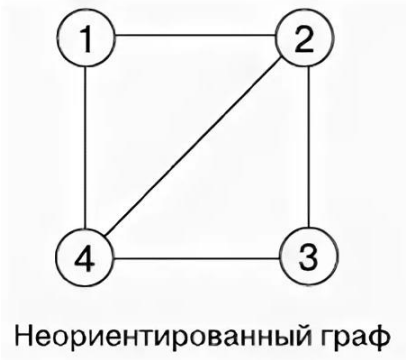

Рисунок 1 - Неориентированный граф

Алгоритм имеет следующий вид: искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины. Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой - наоборот, во всех остальных, кроме этих двух. Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины. В итоге будет построен остов, являющийся минимальным. Если граф был изначально не связен, то остов найден не будет.

3. Описание алгоритма программы

Для программной реализации алгоритма понадобятся 2 массива:

poceshen (int) – массив для отслеживания выбранной вершины.

G (int) - для возможности динамического ввода, хранения данных.

Имеется граф G. Каждая из вершин, входящая в множество poceshen, изначально отмечена false, т.е. не посещенная.

В качестве исходного пункта выбирается вершина [0] и ей присваивается, что она посещена: poceshen [0] = true; затем находятся все соседние вершины в множестве, вычисляется расстояние от вершины, выбранной в начале (нулевой). Если вершина уже находится в множестве, то она отбрасывается, иначе выбираем другую вершину, ближайшую к выбранной вершине на шаге 1. Алгоритм выполняется до тех пор, пока не получится минимальное остовное дерево.

Ниже приведен псевдокод функций Graph_Prima, printMatrix, main.

Graph_Prima()

1.дляi=0 покаi<n делатьi=i+1

2.poceshen[i] = false

3.конеццикла

4.edge = 0;

5.poceshen[0] = true;

6.Пока edge< V-1 делать

7.int min = 10000000;

8.x = 0; y = 0;

9. дляi=0 покаi<V делатьi=i+1

10. еслиpoceshen[i]

11. дляj=0 покаj<Vделатьj=j+1

12. еслиpoceshen[j] == false && G[i][j]

13.если min > G[i][j]

14.min = Gi][j];x = i;y = j;

15.конец условия

16.конец условия

17.конец цикла

18.конец условия

19.конец цикла

20.вывод «x – y : G[x][y]»

21.poceshen[y] = true;

22. edge++

23.конеццикла

Main()

1.ввод «ввести количетсво вершин графа,z»

2.вывод «сгенерировать матрицу автоматически»

3.вывод «ввести матрицу с клавиатуры»

4.вывод «выход»

5.s=scanf_s();

6.если s==’1’

7. Дляi=0 покаi< z делатьi=i+1

8. Для j=i+1 пока j < z делать j=j+1

9. если рандом 1 или 2

10. если 1 вызов функции addEdge

11. Конец цикла

12. Конец цикла

13. вывод «матрицы»

14. Вызов функции Prima

15. Конец условия

16. если s==’2’

17. Ввод ch

18. Если ch == 0: выход из цикла

19. Ввод <<Первой вершины,q1>>

20. Ввод <<Второй вершины,q2>>

21. Ввод <<Вес ребра между вершинами>>

22. Вызов функции addEdge

23. Вызов функции printMatrix

24. Конец условия

25 .если s==’3’

26.Выход из программы

27. конец условия

printMatrix()

1.дляi=0 покаi<Vделатьi=i+1

2.для j=0 пока j<V делать j=j+1

3. вывод «G[i][j]»

4.конец цикла

5. Вывод «\n»

6.конец цикла

4. Описание программы

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

Данная программа многомодульная, поскольку состоит из нескольких функций: main, Prima, addEdge, printMatrix.

Работа программы начинается с выбора количества вершин. Далее выводится меню. Если пользователь выберет «случайное заполнение матрицы», то выводится матрица смежности, связь ребер и их вес. Если же пользователь выбрал пункт «ввод ручками» ,то на экран выводится запрос ввода двух вершин и вес ребра, по такому принципу заполняется вся матрица . Также предусмотрен выход из программы.

```
intmain()
{
setlocale(LC_ALL, "rus");
srand(time(NULL));
intz;
printf("Введите количество вершин:");
scanf_s("%d", &z);
Graph c(z);
int s;
while (1) {
system("cls");
printf("|-----------------------|\n");
```

```
printf("| Количество вершин:%d    |\n", z);
printf("|-----------------------|\n");
printf("| Чтоделаем?            |\n");
printf("|-----------------------|\n");
printf("| 1)Случайноезаполнение |\n");
printf("|-----------------------|\n");
printf("| 2)Ввод ручками        |\n");
printf("|-----------------------|\n");
printf("| 3)ехит                |\n");
printf("|-----------------------|\n");
printf(" Введите: ");
scanf_s("%d", &s);
int q1 = 0, q2 = 0, q3 = 0;
int ch;
switch (s)
{
case 1:
system("cls");
for (int i = 0; i< z; i++)
{
for (int j = i + 1; j < z; j++)
{
if (bool(rand() % 2)) {
c.addEdge(i, j,rand() % 100);
```

```
}
}
}
printf("-----------------\nМатрица смежности:\n");
for (int i = 0; i< z; i++)
{
if (i == 0)
{
printf("%7d", i);
}
else
{
printf("%5d", i);
}

}
printf("%\n");
printf("---------------------------------\n");
c.printMatrix();

printf("---------------------------------\n");
c.Prima();
system("pause");
break;
```

```
case 2:


system("cls");


while (1) {
system("cls");
printf("Если закончили введите '0': ");
scanf_s("%d", &ch);


if (ch == 0) break;
h2:
printf("\nВведите 1-вершину(от 0 до %d): ",z-1);
scanf_s("%d", &q1);


if (q1 > z)
{
printf("Такой вершины нет\n");
goto h2;
}


h1:
printf("Введите 2-вершину(от 0 до %d): ",z-1);
scanf_s("%d", &q2);
```

```
if (q2 > z)
{
printf("Такой вершины нет\n");
goto h1;
}
printf("\n");
printf("Введите Вес ребра: ");
scanf_s("%d", &q3);

c.addEdge(q1, q2, q3);
printf("\n");
c.printMatrix();
system("pause");
}

system("cls");
printf("-----------------\nМатрицасмежности: \n");
c.printMatrix();
printf("-----------------\n");
c.Prima();
system("pause");

break;
```

```
case 3:

exit(0);



}

system("cls");

}

return 0;

}
```

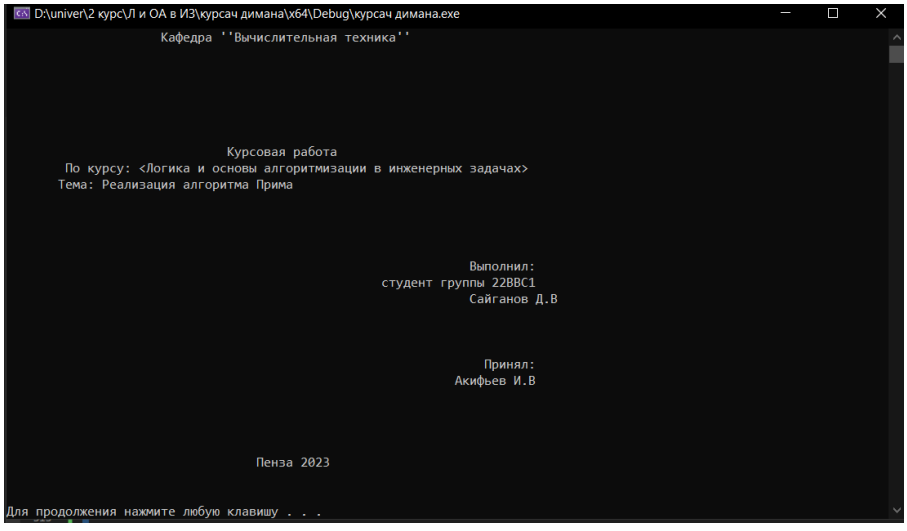

Рисунок 2 –Титульный лист и ввод вершин.

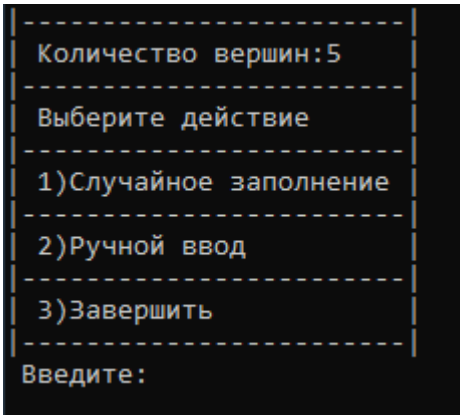

Рисунок 3 - Меню.

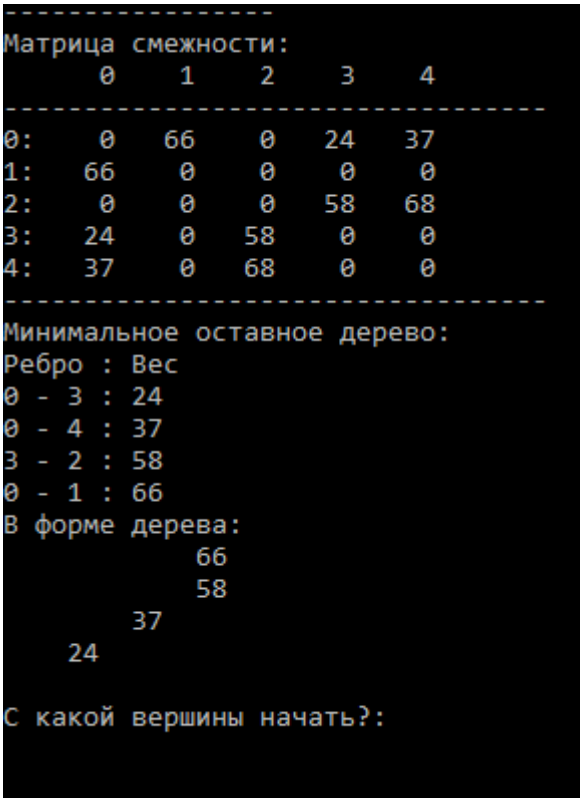

Рисунок 4 - Автоматическое заполнение матрицы.

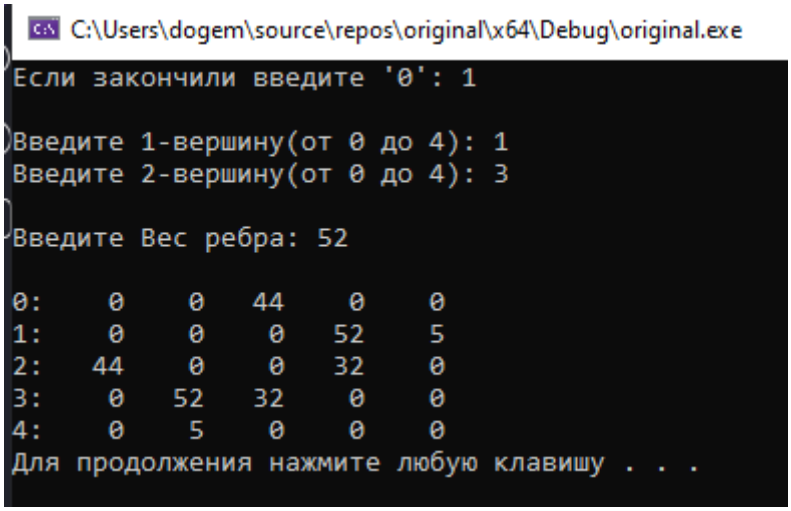

Рисунок 5 - Заполнение матрицы вручную.

5. Тестирование

Среда разработки Microsoft Visual Studio 2019 представляет все средства, необходимые при разработке и отладки многомодульных программ. Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество ошибок и проблем, связанных с вводом данных, изменением дизайна вводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных количеств вершин.

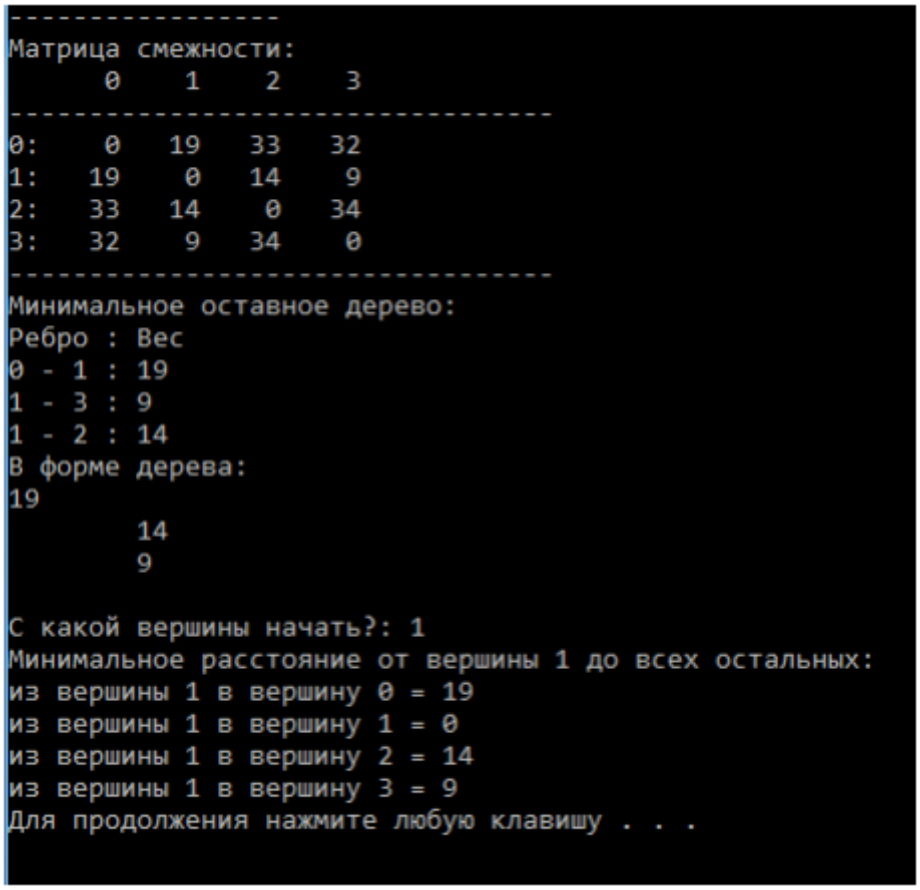

Рисунок 6 - Работа программы при автоматической генерации матрицы 4x4.

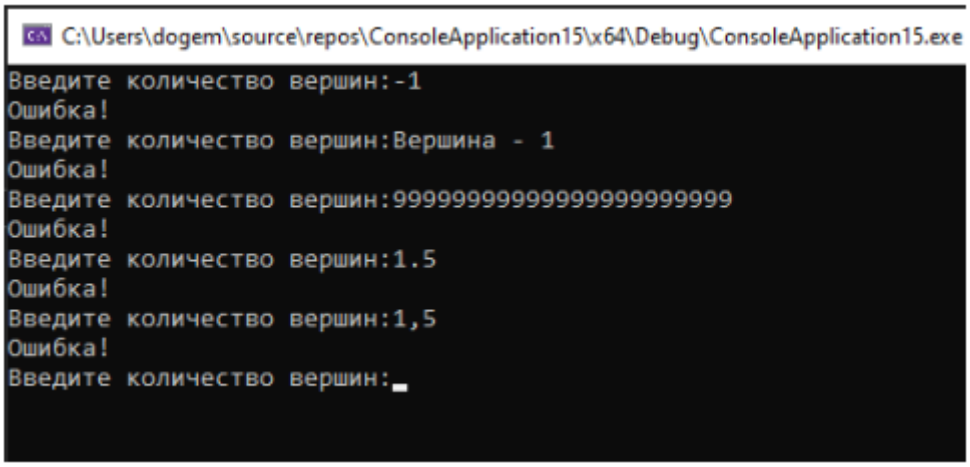

Рисунок 7 - Обработка ввода количества вершин, если ввести отрицательное число, написать символами, ввести ввести большое число, дробное число.

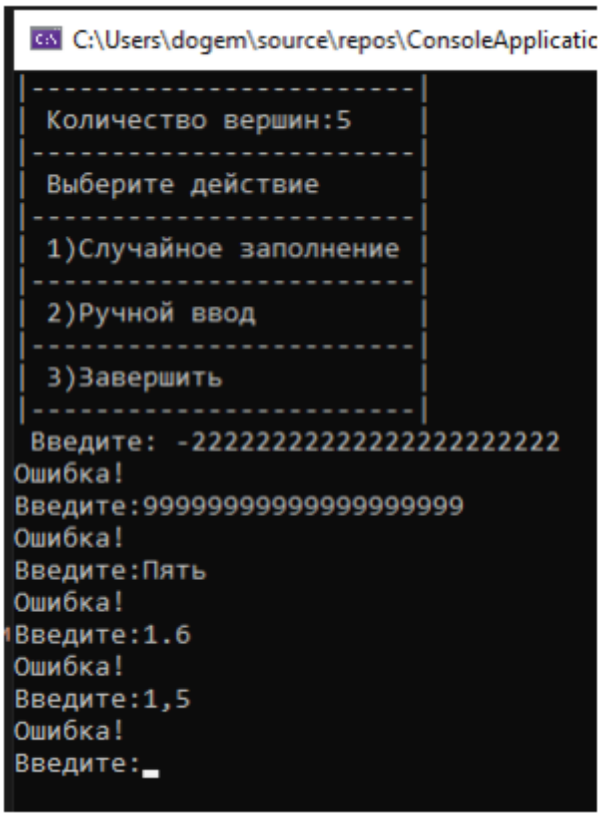

Рисунок 8 - Обработка выбора действия, если ввести отрицательное число, большое число, написать символами, дробное число.

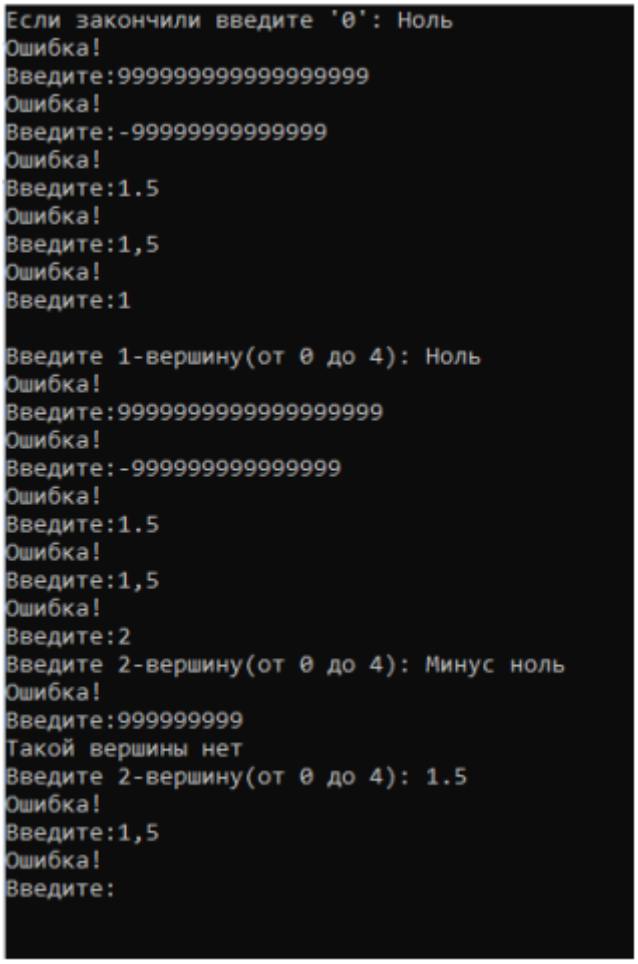

Рисунок 9 - Обработка при ручном вводе, если ввести отрицательное число, большое число, написать символами, дробное число.

Таблица

Описание теста	Ожидаемый результат	Полученный результат
Запуск программы	Вывод меню, вывод сообщения «сгенерировать матрицу автоматически» или «ввести матрицу с клавиатуры»	**Верно**
Выбор генерации матрицы	Вывод сообщения о количестве вершин в графе	**Верно**
Ввод матрицы с клавиатуры	Вывод сообщения о количестве вершин, ввод элементов, вывод элементов	**Верно**
Вывод результата	Вывод правильного результата на разно-размерных графах, идентичность с ручным расчетом	**Верно**
Правильность работы алгоритма	Совпадение ручных расчетов с результатом работы алгоритма	**Верно**
Проверка на наличие изолированных вершин	Должна выполняться проверка на наличие изолированных вершин	**Верно**

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

6. Ручной расчёт программы

Проведем проверку программы посредством ручных вычислений на примере графа с четырьмя вершинами (рисунок 6). Начинаем обход из нулевой вершины, проверяем есть ли путь в другие вершины, если да, то идем. В нашем случаем можем направится в вершины с номерами: 1,2,3. Выгоднее всего будет направится в вершину 1, т.к. её вес меньше всего. Из вершины 1 мы можем пойти в вершины 2 и 3, идём сначала в 3 т.к наименьший вес ребра, из тройки возвращаемся в 1 т.к там нет минимальных рёбер, и последним шагом идём из вершины 1 в вершину 2, т.к в вершину 2 из вершины 1 минимальный вес ребра. Итого: 0 – 1 : 19 1 – 3 : 9 1 – 2: 14 Результат ручных расчетов совпадает с результатом работы алгоритма, таким образом можно сделать вывод, что программа работает верно.

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Прима в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежности, а также работы с новыми алгоритмами. Углублены навыки знания языка программирования Си.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

1. Уилсон Р. Введение в теорию графов.Пер.с анг.1977.208с.

2. Оре О. Графы и их применение: Пер.с анг.1965.176

3. Язык программирования Си Брайан Керниган, Деннис Ритчи

Листинг программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <conio.h>
#include <stdlib.h>
#include <locale.h>
#include <string>
#include <stdio.h>
#include <time.h>
#include <iostream>

using namespace std;
FILE* file;

struct Node
{
	int x;
	Node* l, * r;
};

void PrintTree(Node*& Tree, int r)//функция вывода дерева где r-уровень
{
	if (Tree->r != NULL)
		PrintTree(Tree->r, ++r);
	for (int i = 0; i < (4 * r); i++)
		printf(" ");
	printf("%d\n", Tree->x);
	if (Tree->l != NULL)
		PrintTree(Tree->l, ++r);
	--r;
}


void add_node(int x, Node*& MyTree)
{
	if (NULL == MyTree)
	{
		MyTree = new Node;
		MyTree->x = x;
		MyTree->l = MyTree->r = NULL;
	}

	if (x < MyTree->x)
	{
		if (MyTree->l != NULL) add_node(x, MyTree->l);
		else
		{
			MyTree->l = new Node;
			MyTree->l->l = MyTree->l->r = NULL;
			MyTree->l->x = x;
		}
	}

	if (x > MyTree->x)
	{
		if (MyTree->r != NULL) add_node(x, MyTree->r);
		else
		{
			MyTree->r = new Node;
			MyTree->r->l = MyTree->r->r = NULL;
			MyTree->r->x = x;
		}
	}
}

class Graph {
private:
	int** G;
	int V;

public:
	Graph(int V)
	{
```

```
		Graph::V = V;
		G = new int* [V];
		for (int i = 0; i < V; i++) {
			G[i] = new int[V];
			for (int j = 0; j < V; j++)
				G[i][j] = 0;
		}
	}

	void addEdge(int i, int j, int w) {
		G[i][j] = G[j][i] = w;
	}

	void printMatrix() {
		fprintf(file, "\n");
		for (int i = 0; i < V; i++) {
			printf("%d:", i);
			fprintf(file, "%d", i);
			for (int j = 0; j < V; j++) {
				printf("%5d", G[i][j]);
				fprintf(file, "%5d", G[i][j]);
			}
			printf("\n");
			fprintf(file, "\n");

		}
	}
	void Prima();
	void Dijkstra(int start);
};

void Graph::Prima() {
	Graph Gr(V);
	int edge;
	edge = 0;
	int* poceshen = new int[V];

	memset(poceshen, 0, (V * 2) * 2);
	Node* Tree = NULL;

	poceshen[0] = true;
	int x, y, shet = 0;
	for (int i = 0; i < V; i++)
	{
		for (int j = i; j < V; j++)
		{
			if (G[i][j] != 0) shet++;
		}
	}

	printf("Минимальное оставное дерево:\n");
	printf("Ребро : Вес\n");
	while (edge < V) {
		int min = 1000000; x = 0; y = 0;
		for (int i = 0; i < V; i++) {
			if (poceshen[i]) {
				for (int j = 0; j < V; j++) {
					if (poceshen[j] == false && G[i][j]) {
						if (min > G[i][j]) {
							min = G[i][j];
							x = i;
							y = j;
						}
					}
				}
			}
		}
		Gr.addEdge(x, y, G[x][y]);
		if (G[x][y] != 0)
		{
			printf("%d - %d : %d\n", x, y, G[x][y]);
			add_node(G[x][y], Tree);
		}
```

```
                poceshen[y] = true;
                edge++;
        }
        printf("В форме дерева:\n");
        PrintTree(Tree, 0);
        int start;
        printf("\nС какой вершины начать?: ");
        scanf_s("%d", &start);
        printf("Минимальное расстояние от вершины %d до всех остальных:", start);
        printf("\n");
        Gr.Dijkstra(start);
}

void Graph::Dijkstra(int start)
{
        int* distance = new int[V];
        bool* visited = new bool[V];
        int count, index;
        for (int i = 0; i < V; i++)
                distance[i] = INT_MAX, visited[i] = false;
        distance[start] = 0;
        for (count = 0; count < V - 1; count++)
        {
                int min = INT_MAX;
                for (int i = 0; i < V; i++)

                        if (!visited[i] && distance[i] <= min)
                        {
                                min = distance[i]; index = i;
                        }
                int u = index;
                visited[u] = true;
                for (int i = 0; i < V; i++)
                        if (!visited[i] && G[u][i] && distance[u] != INT_MAX && distance[u] + G[u][i]
< distance[i])
                                distance[i] = distance[u] + G[u][i];
        }
        for (int i = 0; i < V; i++)
                if (distance[i] != INT_MAX) {
                        printf("из вершины %d в вершину %d = %d\n", start, i, distance[i]);
                }
                else {
                        printf("из вершины %d в вершину %d = не получилось дойти(\n", start, i);
                }

}


int main()
{
        setlocale(LC_ALL, "rus");
        srand(time(NULL));
        int z;
        int res;
        file = fopen("Prima.txt", "w");
        printf("\t Министерство науки и высшего образования Российской Федерации\n");
        printf("\t\t   Пензенский государственный университет\n");
        printf("\t\t     Кафедра ''Вычислительная техника''\n\n\n\n\n\n\n");
        printf("\t\t\t      Курсовая работа\n");
        printf("\tПо курсу: «Логика и основы алгоритмизации в инженерных задачах»\n");
        printf("       Тема: Реализация алгоритма Прима\n\n\n\n\n");
        printf("\t\t\t\t\t\t\t       Выполнил:\n");
        printf("\t\t\t\t\t\t   студент группы 22ВВС1\n");
        printf("\t\t\t\t\t\t\t       Сайганов Д.В\n\n\n\n");
        printf("\t\t\t\t\t\t\t         Принял:\n");
        printf("\t\t\t\t\t\t\t     Акифьев И.В\n\n\n\n\n");
        printf("\t\t\t\t  Пенза 2023\n\n\n");
        system("pause");
        system("cls");
        printf("Введите количество вершин:");

        while (!(cin >> z) || (cin.peek() != '\n') || z < 0 || z > 20)
        {
                cin.clear();
```

```
        while (cin.get() != '\n');
        cout << "Ошибка! \nВведите количество вершин:";
}

Graph c(z);
int s;

while (1) {
        system("cls");
        printf("|------------------------|\n");
        printf("| Количество вершин:%d    |\n", z);
        printf("|------------------------|\n");
        printf("| Выберите действие      |\n");
        printf("|------------------------|\n");
        printf("| 1)Случайное заполнение |\n");
        printf("|------------------------|\n");
        printf("| 2)Ручной ввод          |\n");
        printf("|------------------------|\n");
        printf("| 3)Завершить            |\n");
        printf("|------------------------|\n");
        printf(" Введите: ");
        while (!(cin >> s) || (cin.peek() != '\n'))
        {
                cin.clear();
                while (cin.get() != '\n');
                cout << "Ошибка! \nВведите:";
        }

        int q1 = 0, q2 = 0, q3 = 0;
        int ch;
        switch (s)
        {
        case 1:
                system("cls");
                for (int i = 0; i < z; i++)
                {
                        for (int j = i + 1; j < z; j++)
                        {
                                if (bool(rand() % 2)) {
                                        c.addEdge(i, j, rand() % 100);
                                }
                        }
                }
                printf("-----------------\nМатрица смежности:\n");
                for (int i = 0; i < z; i++)
                {
                        if (i == 0)
                        {
                                printf("%7d", i);
                                fprintf(file, "%7d", i);
                        }
                        else
                        {
                                printf("%5d", i);
                                fprintf(file, "%5d", i);
                        }

                }
                printf("\n");
                fprintf(file, "\n");
                printf("---------------------------------\n");
                fprintf(file, "---------------------------------");
                c.printMatrix();

                printf("---------------------------------\n");
                fprintf(file, "---------------------------------");
                c.Prima();
                system("pause");
                break;
        case 2:

                system("cls");

                while (1) {
```

```
                        system("cls");
                        printf("Если закончили введите '0': ");

                        while (!(cin >> ch) || (cin.peek() != '\n'))
                        {
                                cin.clear();
                                while (cin.get() != '\n');
                                cout << "Ошибка! \nВведите:";
                        }

                        if (ch == 0) break;
                h2:
                        printf("\nВведите 1-вершину(от 0 до %d): ", z - 1);

                        while (!(cin >> q1) || (cin.peek() != '\n') || q1 >= z)
                        {
                                cin.clear();
                                while (cin.get() != '\n');
                                cout << "Ошибка! \nВведите:";
                        }

                        if (q1 > z)
                        {
                                printf("Такой вершины нет\n");
                                goto h2;
                        }

                h1:
                        printf("Введите 2-вершину(от 0 до %d): ", z - 1);

                        while (!(cin >> q2) || (cin.peek() != '\n'))
                        {
                                cin.clear();
                                while (cin.get() != '\n');
                                cout << "Ошибка! \nВведите:";
                        }

                        if (q2 > z)
                        {
                                printf("Такой вершины нет\n");
                                goto h1;
                        }
                        printf("\n");
                        printf("Введите Вес ребра: ");
                        scanf_s("%d", &q3);

                        c.addEdge(q1, q2, q3);
                        printf("\n");
                        c.printMatrix();
                        system("pause");
                }

                system("cls");
                printf("-----------------\nМатрица смежности: \n");
                c.printMatrix();
                printf("-----------------\n");
                c.Prima();
                system("pause");

                break;

        case 3:
                exit(0);

        }
        system("cls");
    }
    fclose(file);
    return 0;
}
```