

Оглавление

[Реферат 5](#_Toc91271104)

[Введение 6](#_Toc91271105)

[1. Постановка задачи 7](#_Toc91271106)

[2. Теоретическая часть 8](#_Toc91271107)

[3. Описание алгоритма программы 9](#_Toc91271108)

[4. Описание программы 14](#_Toc91271109)

[5. Тестирование 17](#_Toc91271110)

[6. Ручной расчёт программы 22](#_Toc91271111)

[Заключение 23](#_Toc91271112)

[Список литературы 24](#_Toc91271113)

[Приложение А 25](#_Toc91271114)

[Листинг программы 25](#_Toc91271115)

# 

# Реферат

Отчет 27 страниц, 9 рисунков

АЛГОРИТМ ПРИМА

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

В работе представлен алгоритм Прима.

# 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.

Алгоритм Прима строит минимальное остовное дерево, добавляя на каждом шаге к строящемуся остову безопасное ребро минимальной длины.

Ребро называется безопасным, если при добавлении его к строящемуся остову не нарушается свойство ацикличности.

В качестве среды разработки мною была выбрана *MicrosoftVisualStudio2010*, 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, осуществляющий нахождение минимального остовного дерева.

# Постановка задачи

Требуется реализовать программу, которая осуществит поиск минимального остовного дерева.

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 должны выводиться матрица смежности, вершина, ближайшая вершина и вес ребра между этими вершинами. Необходимо предусмотреть различные варианты поиска, чтобы программа не выдавала ошибок и работа правильно. Устройство ввода - клавиатура и мышь.

# Теоретическая часть

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

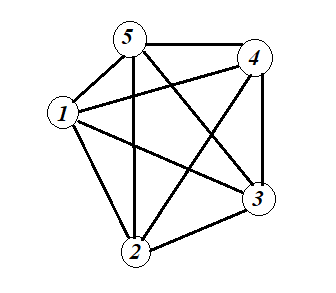


Рисунок 1 – Неориентированный граф

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где пути из одной вершины в другую и обратно обозначаются единицей, иначе нулем.

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.

# Описание алгоритма программы

Для программной реализации алгоритма понадобятся два массива: selected (int) – массив для отслеживания выбранной вершины (trueили false), matr (int) –для возможности динамического ввода, хранения данных.

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

В качестве исходного пункта выбирается вершина [0] и ей присваивается, что она посещена: selected [0] = true; затем находятся все соседние вершины в множестве matr, 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.

НижеприведенпсевдокодфункцииAlg\_Prima, Output, main.

**Alg\_Prima()**

1.дляi=0 покаi<nделатьi=i+1

2.selected[i] = false

3.конеццикла

4.no\_edge = 0;

5.selected[0] = true;

6.Пока no\_edge< n-1 делать

7.int min = INF;

8.x = 0; y = 0;

9. дляi=0 покаi<nделатьi=i+1

10.если selected[i]

11. дляj=0 покаj<nделатьj=j+1

12. если !selected[j] &&matr[i][j]

13.если min >matr[i][j]

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

15.конец условия

16.конец условия

17.конец цикла

18.конец условия

19.конец цикла

20.вывод «x – y :matr[x][y]»

21.selected[y] = true;

22. no\_edge++

23.конеццикла

24. вывод стоимости дерева

**Output()**

1.дляi=0 покаi<nделатьi=i+1

2.дляj=0 покаj<nделатьj=j+1

3. вывод «matr[i][j]»

4.конец цикла

5. Вывод «\n»

6.конец цикла

**Main()**

1.вывод «Добро пожаловать! Выберите один из пунктов меню, что

начать:»

2.вывод «автоматическая генерация матрицы»

3.вывод «ручной ввод»

4.вывод «завершить сеанс»

5.d=\_getch();

6.если d==’1’

7.вывод «введите размер матрицы»

8.ввод «n»

9.selected = (int\*)malloc(n \* sizeof(int))

10.matr = (int\*\*)malloc(n \* sizeof(int\*))

11. дляi=0 покаi<nделатьi=i+1

12.matr[i] = (int\*)malloc(n \* sizeof(int))

13.конец цикла

14. дляi=0 покаi<nделатьi=i+1

15.дляj=0 покаj<nделатьj=j+1

16.еслиi==j

17.matr[i][j]=0

18.конец условия

19 еслиi<j

20.matr[i][j] = rand()%80

21.matr[j][i] = matr[i][j]

22.конец условия

23. еслиmatr[i][j]>0

24.stp++

25.конец условия

26.конец цикла

27.конец цикла

28. вызов функции Output(matr,n)

29.дляi=0 покаi<nделатьi=i+1

30. если stp == 0

31. Выход из программы

32.Конец условия

33.Конец цикла

34.вызов функции Alg\_Prima

35.конец условия

36.если d==’2’

37.вывод «введите размер матрицы»

38.ввод «n»

39.selected = (int\*)malloc(n \* sizeof(int));stp = (int\*)malloc(n \* sizeof(int));

40.matr = (int\*\*)malloc(n \* sizeof(int\*))

41. дляi=0 покаi<nделатьi=i+1

42.matr[i] = (int\*)malloc(n \* sizeof(int))

43.конец цикла

44. дляi=0 покаi<nделатьi=i+1

45.дляj=0 покаj<nделатьj=j+1

46.еслиi==j

47.matr[i][j]=0

48.вывод «matr[i][j]»

49.конецусловия

50.еслиi<j

51.ввод «matr[]i[j]»

52.matr[j][i] = matr[i][j]

53.конец условия

54. еслиmatr[i][j]>0

55.stp++

56.конец условия

57.конец цикла

58.конец цикла

59.вызов функции Output(matr,n)

60.дляi=0 покаi<nделатьi=i+1

61. если stp == 0

62. Выход из программы

63.Конец условия

64.Конец цикла

65.вызов функции Alg\_Prima

66.если d==’3’

67.Выход из программы

68. конец условия

# 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(VisualC++).*

Данная программа многомодульная, поскольку состоит из нескольких функций: main, Alg\_Prima, Output.

Работа программы начинается с запроса генерации матрицы. Если пользователь выбрал сгенерировать матрицу, то на экран выводится запрос на количество вершин в графе. Если же пользователь выбрал пункт «ввести матрицу с клавиатуры», то на экран выводится запрос на количество вершин в графе, а затем элементы массива вводятся с клавиатуры. Такжепредусмотрен выход из программы.

printf(" Добро пожаловать! Выберите один из пунктов меню, что начать: \n\n");

printf(" 1. Автоматическая генерация матрицы. (нажмите 1) \n");

printf(" 2. Ручной ввод (нажмите 2) \n");

printf(" 3. Завершить сеанс (нажмите 3)\n");

printf("=========================================================================");

d = \_getch();

if (d == '1')

{

printf("\n");

printf("Введите размер матрицы: ");

scanf("%d", &n);

printf("=========================================================================");

stp = (int\*)malloc(n \* sizeof(int));

selected = (int\*)malloc(n \* sizeof(int)); //выделение памяти под массив отслеживания вершин

matr = (int\*\*)malloc(n \* sizeof(int\*)); //выделение памяти под массив указателей

for (int i = 0; i < n; i++) //выделение памяти для массива значений

{

matr[i] = (int\*)malloc(n \* sizeof(int));

}

for (int i = 0; i < n; i++) // генерация массива случайных чисел

{

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

{

if (i == j)

{

matr[i][j] = 0;

}

if (i< j)

{

matr[i][j] = rand() % 80;

matr[j][i] = matr[i][j];

}

if (matr[i][j] > 0)

{

stp[i]++;

}

}

}

printf("\nРезультат: \n");

Output(matr, n); //вызов функции печати сгенерированного массива

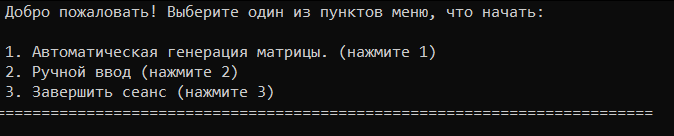


Рисунок 2 – Приветствие.

После выбора пользователе варианта генерации матрицы происходит ее вывод посредством функцииOutput:

for(inti = 0; i< n; i++) // выводсгенерированнойматрицы

{

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

{

printf("%3d ",matr[i][j]);

}

printf("\n");

}

printf("\n ");

Ниже можно увидеть оформление начального запроса и дальнейшего действия с ним:

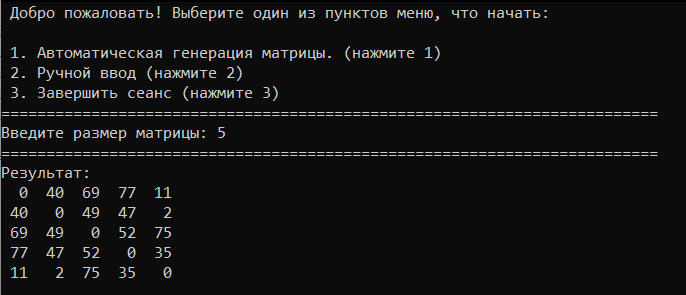


Рисунок 3 – Автоматическая генерация матрицы.

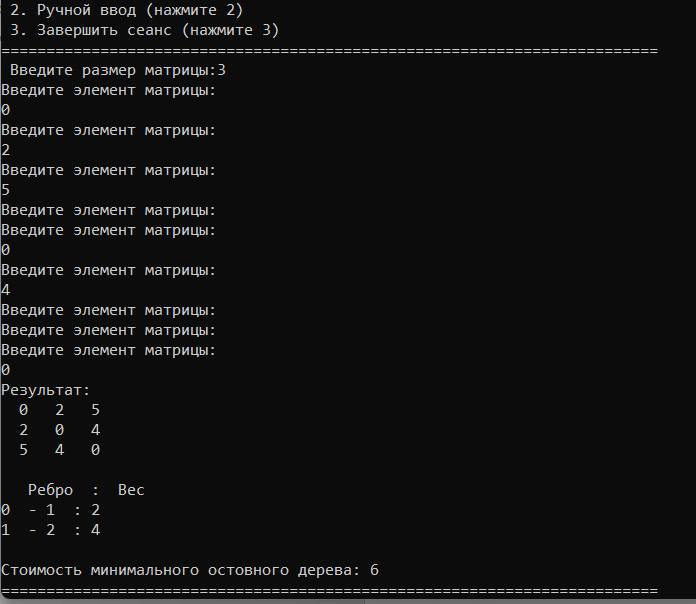


Рисунок 4 – Ручной ввод.

# Тестирование

Среда разработки MicrosoftVisualStudio 2010 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.

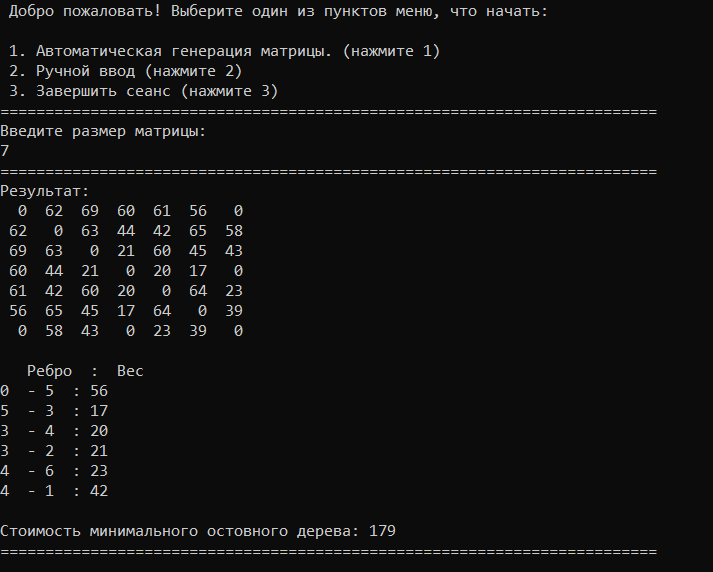


Рисунок 5 – Работа программы при автоматической генерации матрицы 7х7

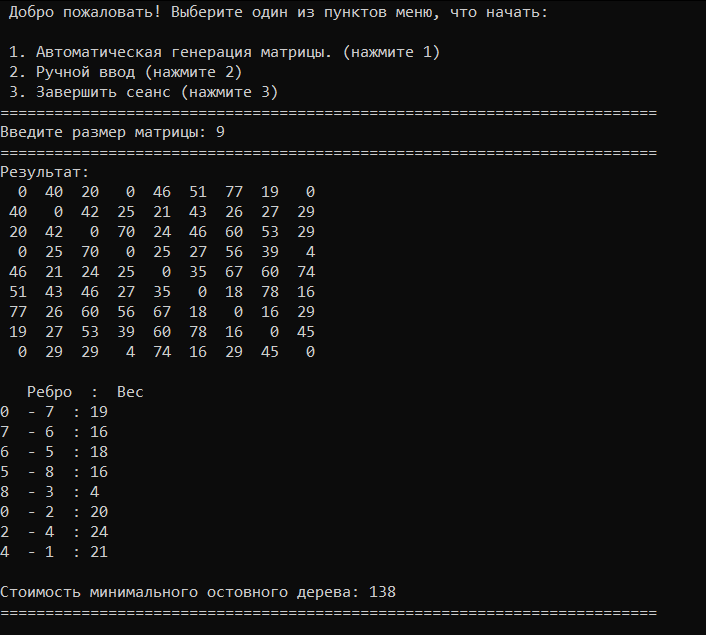


Рисунок 6 - Работа программы при автоматической генерации матрицы 9х9

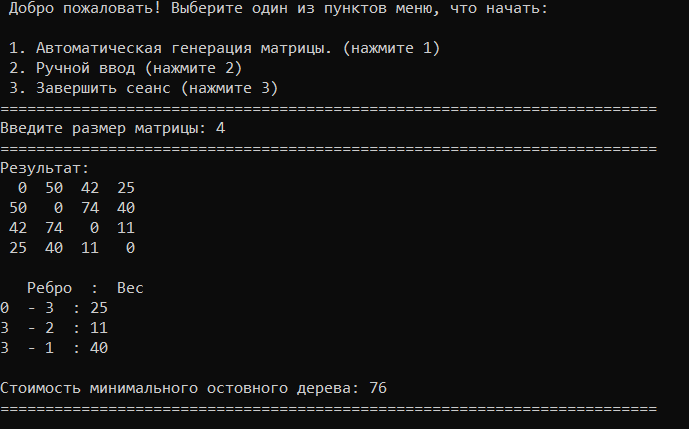


Рисунок 7 - Работа программы при автоматической генерации матрицы 4х4

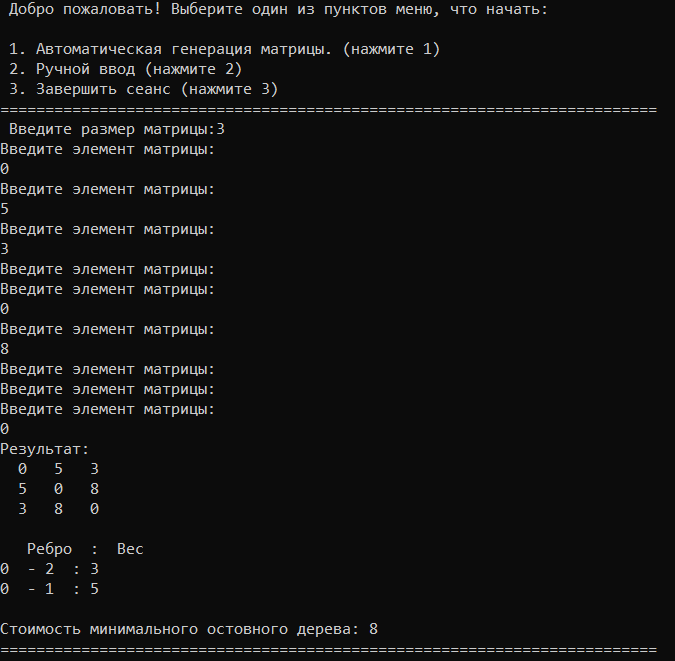


Рисунок 8 - Работа программы при ручном вводе матрицы 3х3

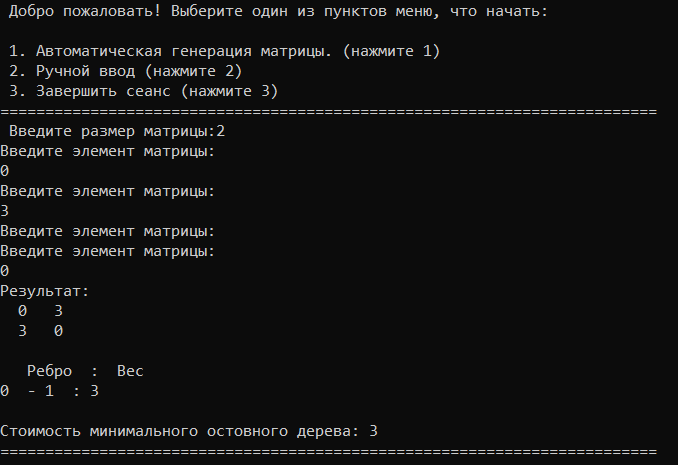


Рисунок 9 - Работа программы при ручном вводе матрицы 2х2

|  |  |  |
| --- | --- | --- |
| 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 |

**Таблица**

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

# Ручной расчёт программы

Проведем проверку программы посредством ручных вычислений на примере графа с четырьмя вершинами (рисунок 7).

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 пункт три, т.к. его вес меньше всего. Из вершины 3 мы можем направится в пункты 1,2. Выберем пункт 2, так путь в него наименьший. Из пункта 2 мы можем пойти в пункт 1 или вернуться обратно в пункт 2 и затем пойти в пункт 1, но уже по другому пути. Программа выберет 2-й вариант, потому что именно таким образом мы получим кратчайшее ребро.

Итого:

0-3: 25

3-2: 11

3-1: 40

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 2010*.

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. Герберт Шилдт «полный справочник по С++» - вильямс,2006
3. З. Оре О. Графы и их применение: Пер.сангл.1965.176с.

# Приложение А

# Листинг программы

#include<conio.h>

#include<iostream>

#include<locale.h>

#include<windows.h>

#include<cstring>

#include<fstream>

usingnamespacestd;

#defineINF 9999999

#defineV 5

FILE\* file = fopen("text.txt", "a");

voidAlg\_Prima(intn, int\*\* matr, int\* selected)

{

intno\_edge;

// Изначально установим false

for (inti = 0; i<n; i++)

{

selected[i] = false;

}

no\_edge = 0;

// число ребер в минимальном остовном дереве будет

// всегда меньше (V -1), где V-число вершин в

//графе

selected[0] = true;

intx; // номерстроки

inty; // номерстолбца

int sum = 0;

printf("Ребро : Вес\n");

while (no\_edge<n - 1)

{

int min = INF;

x = 0;

y = 0;

for (inti = 0; i<n; i++)

{

if (selected[i])

{

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

{

if(!selected[j] &&matr[i][j]) // не выбрана и есть край

{

if (min >matr[i][j])

{

min = matr[i][j];

x = i;

y = j;

}

}

}

}

}

sum = sum + matr[x][y];

printf("%d ", x); printf(" - "); printf("%d ", y);

printf(" : "); printf("%d\n", matr[x][y]);

fprintf(file, "%d ", x); fprintf(file, " - "); fprintf(file, "%d ", y);

fprintf(file, " : "); fprintf(file, "%d\n", matr[x][y]);

selected[y] = true;

no\_edge++;

}

printf("\nСтоимость минимального остовного дерева: %d", sum);

fprintf(file, "\nСтоимость минимального остовного дерева: %d\n", sum);

fprintf(file, "\n=========================================================================");

printf("\n=========================================================================\n");

}

voidOutput(int\*\* matr, intn) // выводсгенерированнойматрицы

{

for (inti = 0; i<n; i++) // выводсгенерированнойматрицы

{

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

{

printf("%3d ", matr[i][j]);

}

printf("\n");

}

printf("\n ");

}

intmain()

{

srand(time(0));

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int\*\* matr; //указательдлямассивауказателей

int\* selected; // Массив для отслеживания вершины

int\* stp;

int n, d;

printf(" Добро пожаловать! Выберите один из пунктов меню, что начать: \n\n");

printf(" 1. Автоматическая генерация матрицы. (нажмите 1) \n");

printf(" 2. Ручной ввод (нажмите 2) \n");

printf(" 3. Завершить сеанс (нажмите 3)\n");

printf("=========================================================================");

d = \_getch();

if (d == '1')

{

printf("\n");

printf("Введите размер матрицы: ");

scanf("%d", &n);

printf("=========================================================================");

stp = (int\*)malloc(n \* sizeof(int));

selected = (int\*)malloc(n \* sizeof(int)); //выделение памяти под массив отслеживания вершин

matr = (int\*\*)malloc(n \* sizeof(int\*)); //выделение памяти под массив указателей

for (int i = 0; i < n; i++) //выделение памяти для массива значений

{

matr[i] = (int\*)malloc(n \* sizeof(int));

}

for (int i = 0; i < n; i++) // генерация массива случайных чисел

{

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

{

if (i == j)

{

matr[i][j] = 0;

}

if (i< j)

{

matr[i][j] = rand() % 80;

matr[j][i] = matr[i][j];

}

if (matr[i][j] > 0)

{

stp[i]++;

}

}

}

printf("\nРезультат: \n");

Output(matr, n); //вызов функции печати сгенерированного массива

//printf("\nIzolir versh:\n");

for (inti = 0; i<n;i++)

{

if (stp[i] == 0)

{

printf("Алгоритм не работает при наличии изолированных вершин");

exit(0);

}

}

Alg\_Prima(n, matr, selected); //вызов функции реализации алгоритма Прима

}

if (d == '2')

{

printf("\n");

printf(" Введите размер матрицы:");

scanf("%d", &n);

stp = (int\*)malloc(n \* sizeof(int));

selected = (int\*)malloc(n \* sizeof(int)); //выделение памяти под массив отслеживания вершин

matr = (int\*\*)malloc(n \* sizeof(int\*)); //выделение памяти под массив указателей

for (int i = 0; i < n; i++) //выделение памяти для массива значений

{

matr[i] = (int\*)malloc(n \* sizeof(int));

}

for (inti = 0; i< n; i++) // генерациямассивачисел

{

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

{

printf("Введите элемент матрицы:\n");

if (i == j)

{

matr[i][j] = 0;

printf("%d\n", matr[i][j]);

}

if (i< j)

{

scanf("%d", &matr[i][j]);

matr[j][i] = matr[i][j];

}

if (matr[i][j] > 0)

{

stp[i]++;

}

}

}

printf("Результат: \n");

Output(matr, n); // вызов функции вывода результата генерации

for (inti = 0; i<n;i++)

{

if (stp[i] == 0)

{

printf("Алгоритм не работает при наличии изолированных вершин");

exit(0);

}

}

Alg\_Prima(n, matr, selected); // алгоритмприма

}

if (d == '3')

{

exit(0);

}

return 0;

}