

Министерство науки и высшего 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 нахождения изолированных вершин в
графе»

Выполнил:

Студент группы 21ВВ4

Савкин В.В.

Приняли:

Юрова О.В.

Акифьев И.В.

Хорошо
С 29.12.22

Пенза 2022

Содержание

Реферат	3
Введение.....	4
1) Постановка задачи и обязательные требования.	7
2) Описание структуры программы.	8
4) Тестирование программы.....	16
Заключение	20
Список литературы.....	21
Приложение А. Листинг программы	22

Реферат

Реферат

Отчёт стр. 24, рисунков 10, таблиц 1.

ГРАФ, ТЕОРИЯ ГРАФОВ,

НАХОЖДЕНИЕ ИЗОЛИРОВАННЫХ ВЕРШИН ГРАФА

Цель исследования – разработка программы, которая находит и выводит номер изолированной вершины в графе.

Введение

Целью данной курсовой работы является создание программы, которая будет искать изолированные вершины в графе.

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2019, язык программирования – C.

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

«__»__ 20__

ЗАДАНИЕ

на курсовое проектирование по курсу

"Логика и основы алгоритмизации вычислительных задач"
Студенту Савкина Вадиму Владимировичу Группа 21ВВ4
Тема проекта Реализация алгоритма нахождения
изопрошенных вершин в графе

Исходные данные: (технические требования) на проектирование

- 1) Текстовое или графическое меню;
- 2) Возможность задания параметров размера графа;
- 3) Возможность задания и выбора алгоритмического (случайного) или ручного (с клавиатуры или из файла) задания графа (матрицы смежности);
- 4) Возможность сохранения результатов работы программы;
- 5) Описание программы;
- 6) Тесты;
- 7) Ссылка на литературу;
- 8) Визуализация работы программы.

Объем работы по курсу

1. Расчетная часть

Ручной расчет работы алгоритма

2. Графическая часть

Схема алгоритма в формате блок-схема

3. Экспериментальная часть

Тестирование программы
Визуализация работы программы на тестовых данных

Сроки выполнения проекта по разделам

- 1 Изучение методической части курсовой
- 2 Разработка алгоритма
- 3 Разработка программы
- 4 Тестирование и завершение разработки
- 5 Оформление расчетной записки
- 6
- 7
- 8

Дата выдачи задания "10" сентября

Дата защиты проекта " " "

Руководитель

А. А. Акимов И.В.

Задание получил

"10" сентября

2022 г.

Студент

Савкин В.В.

В.В.С.

1) Постановка задачи и обязательные требования.

Каждый проект должен иметь:

- 1 Текстовое или графическое меню;
- 2 Возможность задания пользователем размера графа (множества);
- 3 Возможность выбора автоматического (случайного) или ручного (с клавиатуры или из файла) задания графа (элементов множества);
- 4 Возможность сохранения результатов работы программы.

2) Описание структуры программы.

1. Вызов функции setcolor (Функция отвечает за графическую составляющую курсовой работы, меняя цвет выводимого текста)

Int main()

2. Создание и запись файла, в который будет сохраняться результат программы
3. if (file == NULL) если создать файл нельзя, тогда
4. printf("Error");
5. Ввод переменной chainse
6. Если значение chainse != 1 && chainse != 2 && chainse != 3
7. Вывод сообщения «ошибка» и возврат обратно в меню
8. Если значение chainse == 3 – выход из программы
9. Если значение chainse == 1 – автоматическое генерирование матрицы смежности графа
10. Если значение chainse == 2 – ручное генерирование матрицы смежности графа
11. Ввод n - количества вершин графа
12. Задание размера матрицы M[n] – case 1 (автоматическое заполнение)
Либо задание размера матрицы R[n] – case 2 (ручное заполнение)
13. Запуск цикла генерации матрицы смежности for(int i = 0; i < n; i++) – по строкам
14. for(int j = 0; j < n; j++) – по столбцам
15. if(i == j) – элементы главной диагонали равны 0 – для case 1
{
 M[i][j] = 0
}
Else if(i != j)
{
 M[i][j] = rand() %3 – случайное заполнение матрицы числами от 0 до 2 включительно
 M[j][i] = M[i][j] – отзеркаливание элементов поверх главной диагонали вниз

```

    }

printf("R[%d][%d] = ", i, j);
    scanf_s("%d", &R[i][j]); - ввод элементов матрицы вручную для
    case 2
    вывод матрицы R[i][j]
16. for(int i = 0; i < n; i++) – по строкам
17. for(int j = 0; j < n; j++) – по столбцам
18. if( M[i][j] != 0 ) – если элемент матрицы не равен 0 case 1

    if( R[i][j] != 0 ) – если элемент матрицы не равен 0 case 2

19. k увеличивается на 1
20. if(M[i][j] != 0 && i > j ) – case 1

    if(R[i][j] != 0 && i > j ) – case 2

21. c увеличивается на 1
22. Вывод матрицы M[i][j] – case 1
    Вывод матрицы R[i][j] – case 2
23. G[i] = k – массив, который хранит в себе элементы матрицы, не равные
    0 – case 1
    L[i] = k массив, который хранит в себе элементы матрицы, не равные 0
    – case 2

24. k = 0
25. Запись в файл числа элементов массива
26. Запись самого массива M
27. for(int i = 0; i < n; i++)
28. if(G[i] == 0) если в графе есть изолированная вершина – case 1
    if(L[i] == 0) если в графе есть изолированная вершина – case 2

29. вывод номера изолированной вершины
30. Вывод сообщения с выбором: перейти в начальное меню или выйти из
    программы
31. Ввод v
32. Если v == 1, то перейти в меню, если v == 2 – выйти из программы,
    иначе ошибка и возврат к вводу v

```

3) Описание программы

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

Программа начинается с вывода в консоль текстового меню. Пользователю предлагается выбрать реализацию создания матрицы смежности графа. Пользователь может заполнить матрицу случайным образом с помощью функции `rand()`, нажав клавишу 1, либо заполнить матрицу вручную с клавиатуры, нажав клавишу 2. Пользователь также имеет возможность выйти из программы нажатием клавиши «3». Если пользователь не нажал одну из этих клавиш, программа выведет в консоль сообщение «Ошибка» и вернёт пользователя в меню, чтобы выбрать реализацию заполнения матрицы смежности повторно.

Выбор пользователем режима заполнения матрицы был реализован с помощью функции `switch()`.

```
printf("\t\t\tПРОГРАММА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ\n\n\n");
    printf("Выберите реализацию заполнения графа\n");
    printf("Нажмите 1 если хотите заполнить граф случайными числами\n");
    printf("Нажмите 2 если хотите заполнить граф вручную\n");
    printf("Нажмите 3 если хотите выйти из программы\n");
    scanf_s("%d", &choise);

switch (choise)
```

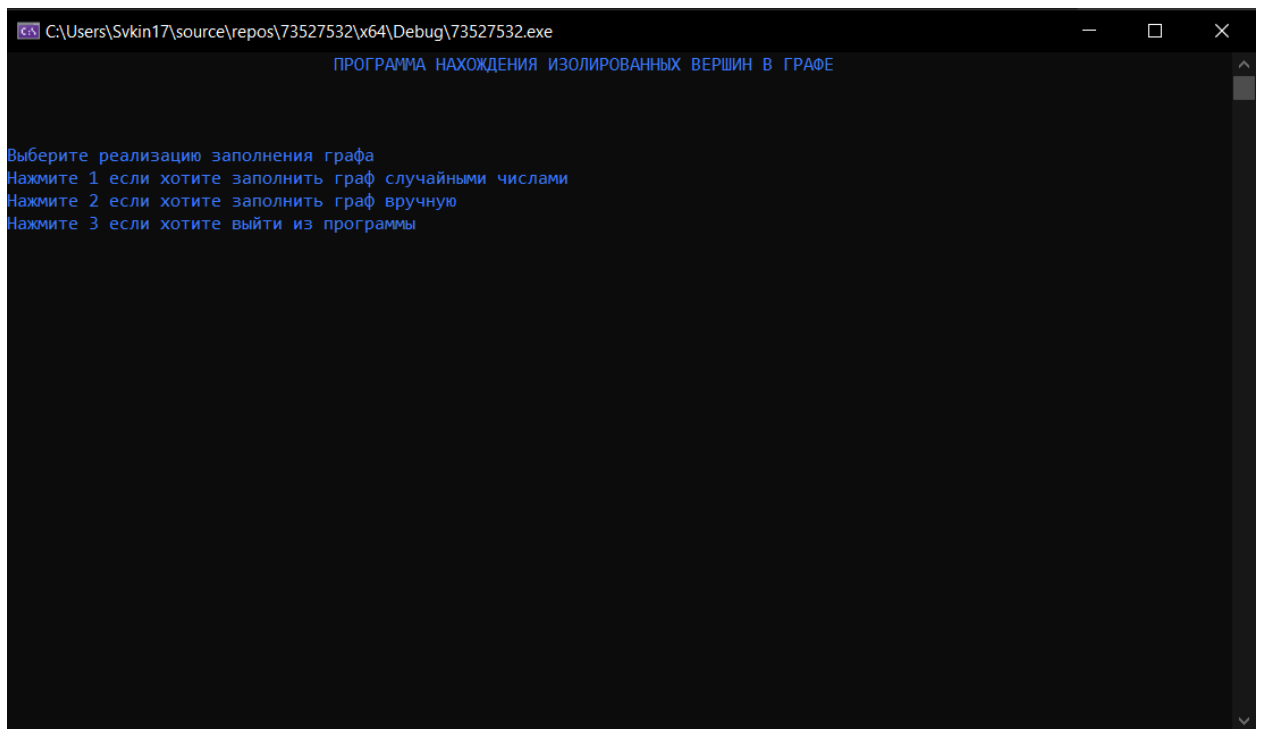


Рисунок 1

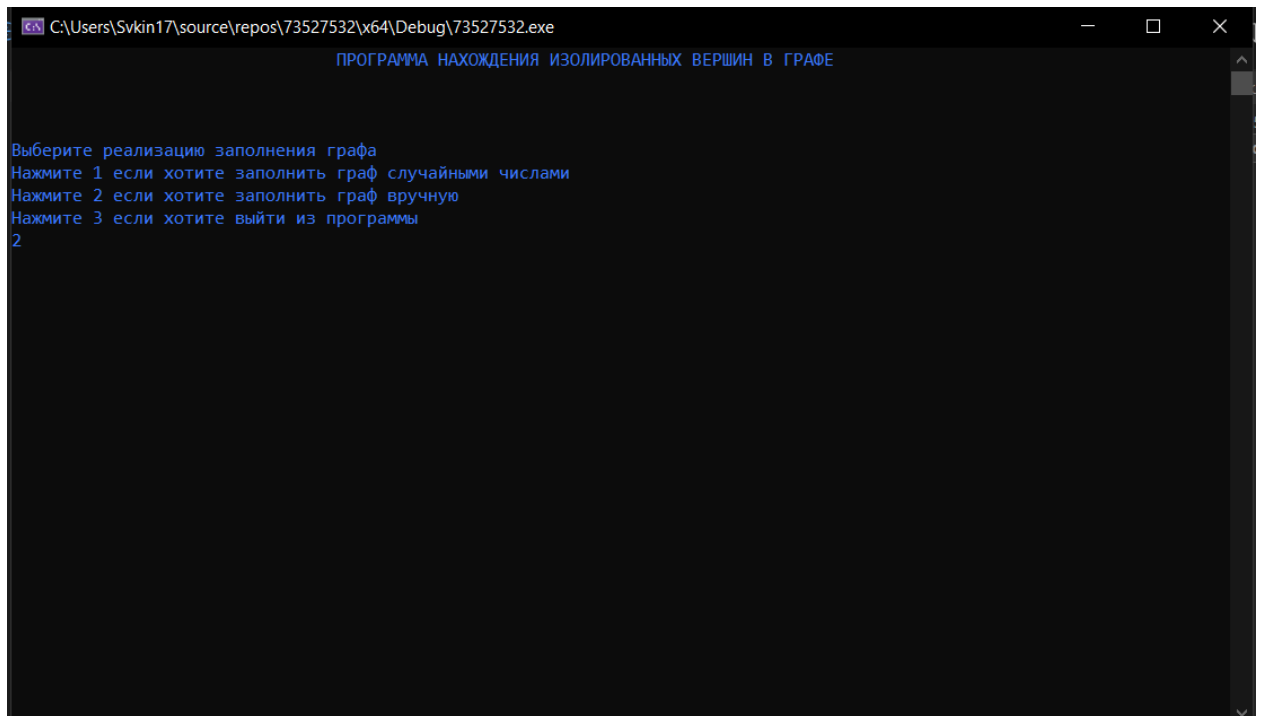


Рисунок 2 – выбор ручного режима заполнения матрицы

После выбора режима заполнения матрицы, программа «прыгает» в один из операторов case(), где пользователь вводит количество вершин графа. Далее происходит заполнение матрицы либо автоматически, либо вручную как на рисунках 3,4.

case 2:

```
{
    printf("Введите количество вершин в графе: \n");
    scanf_s("%d", &n);
    file = fopen("output.bin", "wb"); //wb - флаг на запись бинарного файла

    int v;
    int* L = new int[n];
    int** R = new int* [n];
    m = n;
    for (int i = 0; i < n; i++)
    {
        R[i] = new int[m];
    }
    SetColor(LightBlue, Black);
    printf("Матрица смежности:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            SetColor(LightBlue, Black);
            printf("R[%d][%d] = ", i, j);
            scanf_s("%d", &R[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d ", R[i][j]);
        }
        printf("\n");
    }
}
```

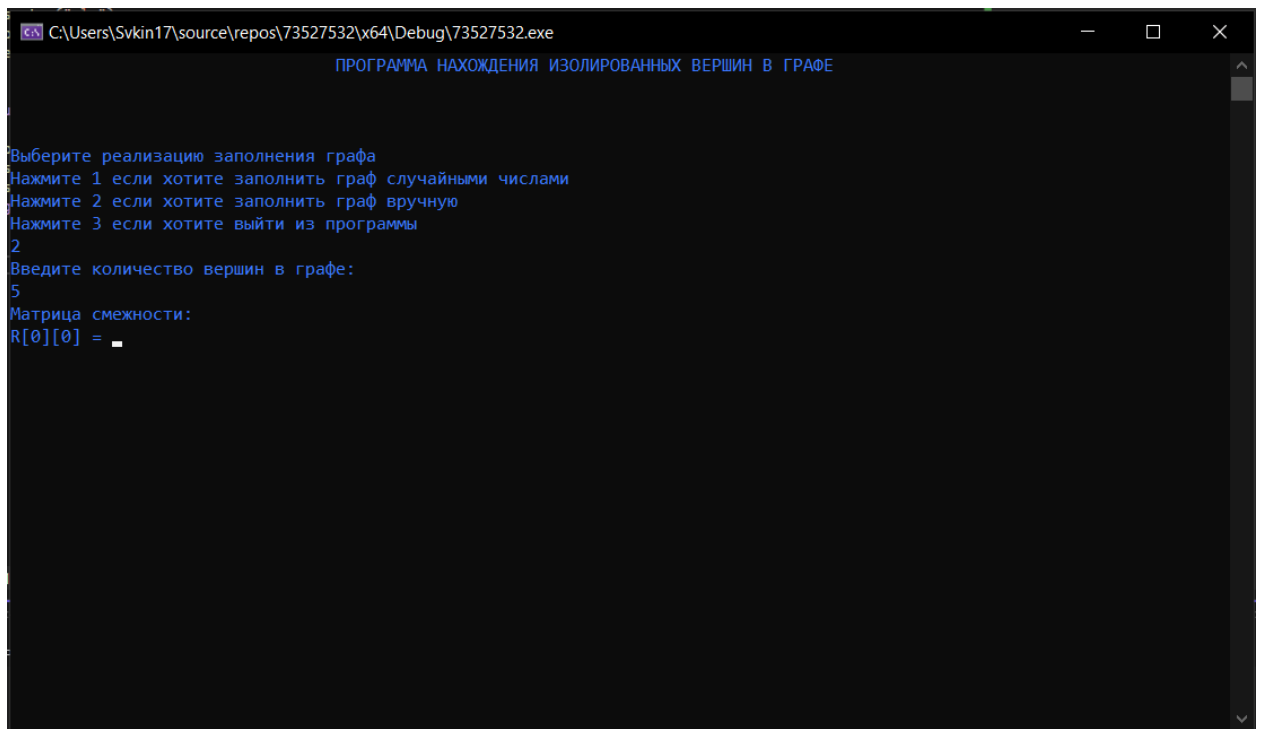


Рисунок 3 – показание ручного режима заполнения матрицы

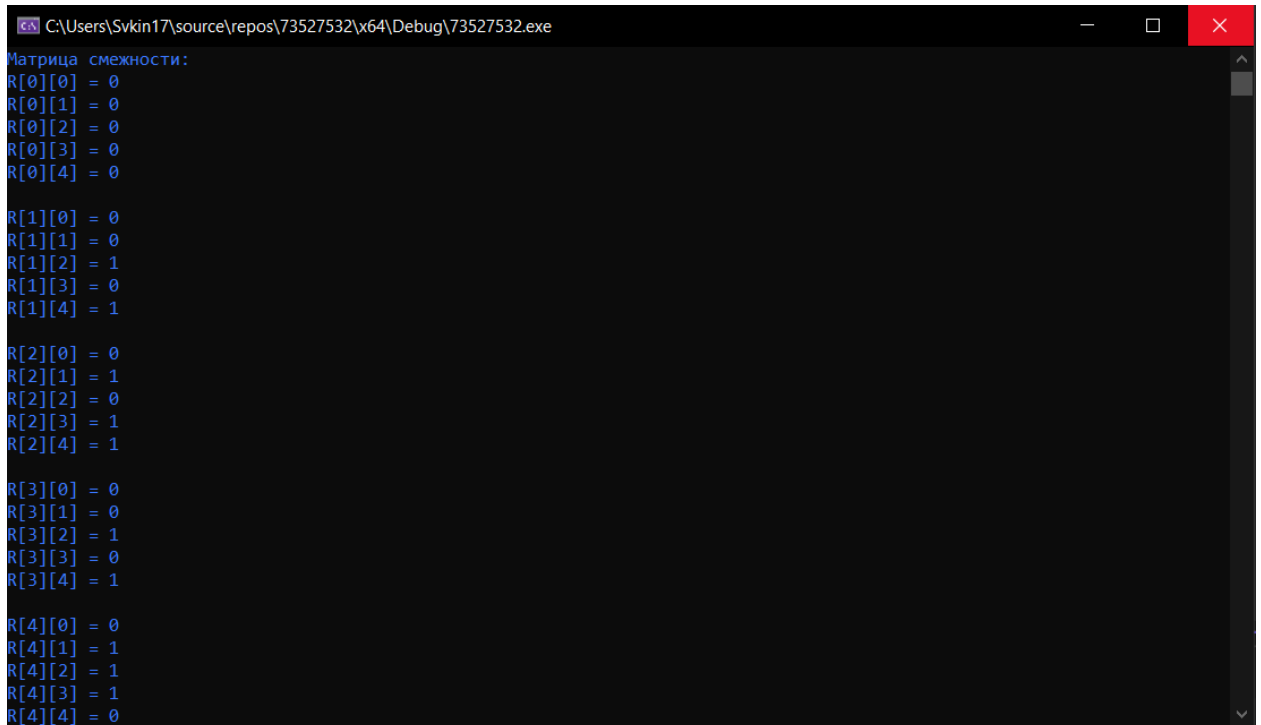


Рисунок 4 – заполненная матрица

Далее программа находит размер графа на основе данных в матрице, после чего выводит в консоль номер изолированной вершины, если таковая имеется.

Пользователю предлагают вернуться в первоначальное меню, нажав клавишу «1», либо закрыть программу, нажав клавишу «2».

```
for (int i = 0; i < n; i++) // цикл по строкам
{
    for (int j = 0; j < m; j++) // цикл по столбцам
    {
        if (R[i][j] != 0)
        {
            k++;
        }
        if (R[i][j] != 0 and i > j)
        {
            c++;
        }
    }
    L[i] = k;
    k = 0;
}
fwrite(&n, sizeof(int), 1, file); //записываем число элементов в массиве (для его чтения
далее)
fwrite(R, sizeof(int), n, file); //запись самого массива
fclose(file); //закрываем файл

for (int i = 0; i < n; i++)
{
    if (L[i] == 0) {
        SetColor(LightBlue, Black);
        printf("Изолированная вершина %d\n", i + 1);
    }
}
start3:
printf("Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из
программы \n");
scanf_s("%d", &v);
if (v == 1){
    system("PAUSE");
    system("cls");
    goto start;
}
else if (v == 2) {
    system("PAUSE");
    system("cls");
    printf("Выход");
    exit(0);
}
else(v != 1 && v != 2); {
    printf("Ошибка, попробуйте снова\n");
    system("PAUSE");
    system("cls");
    goto start3;
}
}
```

```
C:\Users\Svkin17\source\repos\73527532\x64\Debug\73527532.exe

R[1][2] = 1
R[1][3] = 0
R[1][4] = 1

R[2][0] = 0
R[2][1] = 1
R[2][2] = 0
R[2][3] = 1
R[2][4] = 1

R[3][0] = 0
R[3][1] = 0
R[3][2] = 1
R[3][3] = 0
R[3][4] = 1

R[4][0] = 0
R[4][1] = 1
R[4][2] = 1
R[4][3] = 1
R[4][4] = 0

0 0 0 0 0
0 0 1 0 1
0 1 0 1 1
0 0 1 0 1
0 1 1 1 0

Изолированная вершина 1
Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из программы
```

Рисунок 5 – нахождение изолированной вершины

Далее алгоритм повторяется ещё раз.

```
C:\Users\Svkin17\source\repos\73527532\x64\Debug\73527532.exe

ПРОГРАММА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ

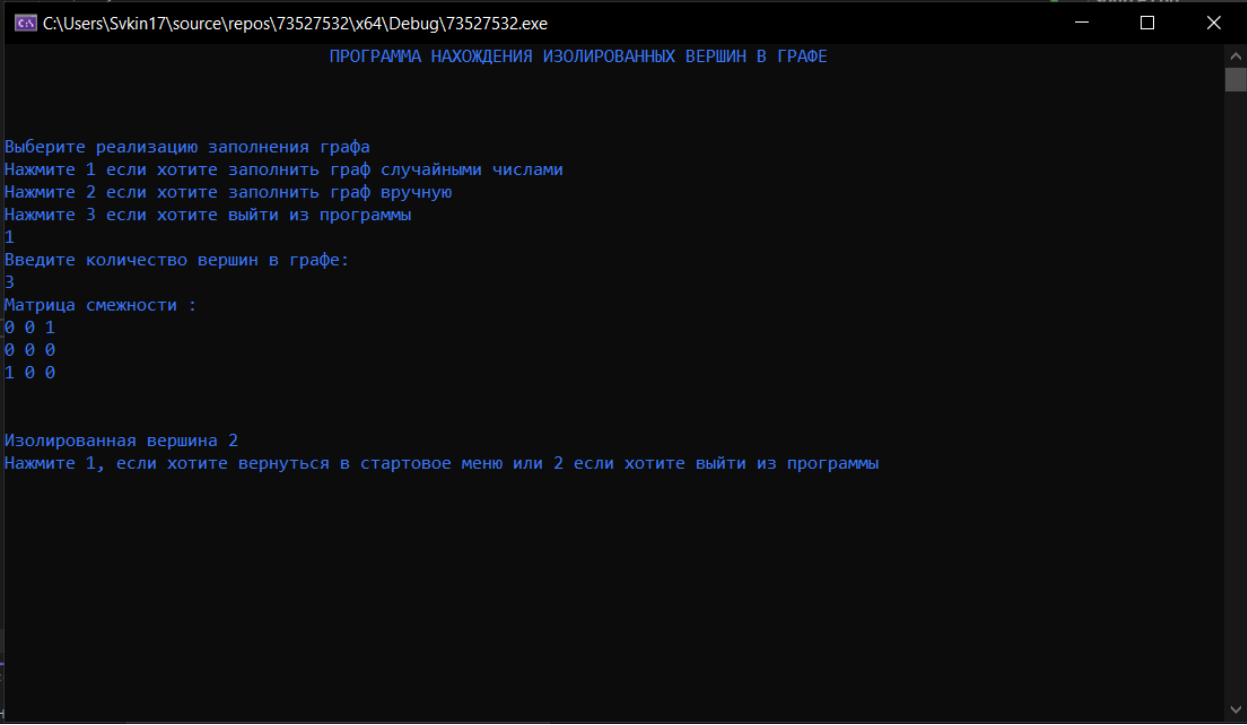
Выберите реализацию заполнения графа
Нажмите 1 если хотите заполнить граф случайными числами
Нажмите 2 если хотите заполнить граф вручную
Нажмите 3 если хотите выйти из программы
```

Рисунок 6 – возвращение в стартовое меню

4) Тестирование программы

Среда разработки Microsoft Visual Studio 2022 предоставляет все средства, необходимые при разработке и отладке многомодульной программы. 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, изменением дизайна выводимых данных, алгоритмом программы.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различным количестве вершин и вывод матрицы смежности графа с изолированной вершиной.



```
C:\Users\Sykin17\source\repos\73527532\x64\Debug\73527532.exe
ПРОГРАММА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ

Выберите реализацию заполнения графа
Нажмите 1 если хотите заполнить граф случайными числами
Нажмите 2 если хотите заполнить граф вручную
Нажмите 3 если хотите выйти из программы
1
Введите количество вершин в графе:
3
Матрица смежности :
0 0 1
0 0 0
1 0 0

Изолированная вершина 2
Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из программы
```

Рисунок 7 – тестирование при выборе автоматического заполнения матрицы и вводе 3 вершин, изолированная вершина под номером 2

```
C:\Users\Sykin17\source\repos\73527532\73527532.exe
ПРОГРАММА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ

Выберите реализацию заполнения графа
Нажмите 1 если хотите заполнить граф случайными числами
Нажмите 2 если хотите заполнить граф вручную
Нажмите 3 если хотите выйти из программы
1
Введите количество вершин в графе:
7
Матрица смежности :
0 1 1 1 2 2 0
1 0 2 0 0 0 1
1 2 0 2 1 0 1
1 0 2 0 0 2 0
2 0 1 0 0 1 2
2 0 0 2 1 0 1
0 1 1 0 2 1 0

Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из программы
```

Рисунок 8 – тестирование при выборе автоматического заполнения матрицы и вводе 7 вершин, изолированной вершины нет

```
C:\Users\Sykin17\source\repos\73527532\73527532.exe
R[1][4] = 0
R[2][0] = 0
R[2][1] = 0
R[2][2] = 0
R[2][3] = 0
R[2][4] = 0
R[3][0] = 1
R[3][1] = 1
R[3][2] = 1
R[3][3] = 0
R[3][4] = 1
R[4][0] = 1
R[4][1] = 0
R[4][2] = 0
R[4][3] = 1
R[4][4] = 0
1 1 1 0 0
0 0 1 1 0
0 0 0 0 0
1 1 1 0 1
1 0 0 1 0
Изолированная вершина 3
Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из программы
```

Рисунок 9 - тестирование при выборе ручного заполнения матрицы и вводе 5 вершин, изолированная вершина под номером 3

```

C:\Users\Sykin17\source\repos\73527532\Debug\73527532.exe
R[5][7] = 0

R[6][0] = 0
R[6][1] = 0
R[6][2] = 1
R[6][3] = 0
R[6][4] = 1
R[6][5] = 1
R[6][6] = 0
R[6][7] = 1

R[7][0] = 0
R[7][1] = 1
R[7][2] = 0
R[7][3] = 1
R[7][4] = 0
R[7][5] = 0
R[7][6] = 1
R[7][7] = 0

0 1 0 1 1 1 0 0
1 0 0 1 1 1 0 1
0 0 0 1 1 1 1 0
0 1 1 0 0 0 0 1
1 1 1 0 0 1 1 0
1 1 1 0 1 0 1 0
0 0 1 0 1 1 0 1
0 1 0 1 0 0 1 0

Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из программы

```

Рисунок 10 - тестирование при выборе ручного заполнения матрицы и вводе 8 вершин, изолированной вершины нет

Описание теста	Ожидаемый результат	Полученный результат
Запуск программы	Вывод текстового меню с выбором режима генерации матрицы смежности графа	верно
Выбор генерации матрицы	Вывод сообщения о выборе количества вершин в графе	верно
Выбор количества вершин в графе	Вывод матрицы, если был выбран автоматический метод генерации либо заполнение и вывод матрицы, если был	верно

	выбран ручной метод генерации	
Нахождение изолированной вершины	Нахождение изолированной вершины в матрице смежности при любом выборе реализации генерации матрицы	верно
Возвращение в начало программы	Возвращение к текстовому меню с выбором реализации генерации матрицы при окончании программы	верно

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

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения изолированных вершин в графе в Microsoft Visual Studio 2022.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приёмы создания матриц смежностей. Приобретены навыки по осуществлению алгоритма нахождения изолированных вершин. Углублены знания языка программирования СИ.

Недостатком разработанной программы является примитивный пользовательский интерфейс, так как 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. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ – М.: МЦНМО, 2001. – 960с.
2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978
3. З. Оре. Графы и их применение: Пер. с англ. 1965. 176с.

Приложение А. Листинг программы

```
#define _CRT_SECURE_NO_WARNINGS
#define _CRT_NONSTDC_NO_WARNINGS
#pragma once
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <locale.h>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>

using namespace std;

HANDLE hConsole;
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);

enum ConsoleColor
{
    Black = 0,
    Blue = 1,
    Green = 2,
    Cyan = 3,
    Red = 4,
    Magenta = 5,
    Brown = 6,
    LightGray = 7,
    DarkGray = 8,
    LightBlue = 9,
    LightGreen = 10,
    LightCyan = 11,
    LightRed = 12,
    LightMagenta = 13,
    Yellow = 14,
    White = 15
};

void SetColor(ConsoleColor text, ConsoleColor background)
{
    SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));
}

int main()
{
    setlocale(LC_ALL, "RUS");
    srand(time(NULL));

    int n, m;
    int c = 0;
    int k = 0;
    int choise = 0;

    FILE* file = NULL; //указатель

    file = fopen("output.bin", "wb"); //wb - флаг на запись бинарного файла

    if (file == NULL) { //если нельзя создать или открыть
        printf("Error");
        getchar();
        return 1;
    }
}
```

start:

```
SetColor(LightBlue, Black);
printf("\t\t\t\tПРОГРАММА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ\n\n\n");
printf("Выберите реализацию заполнения графа\n");
printf("Нажмите 1 если хотите заполнить граф случайными числами\n");
printf("Нажмите 2 если хотите заполнить граф вручную\n");
printf("Нажмите 3 если хотите выйти из программы\n");
scanf_s("%d", &choise);
```

```
switch (choise)
{
case 1:
{
    printf("Введите количество вершин в графе: \n");
    scanf_s("%d", &n);

    m = n;
    int v;
    int** M = new int* [n];
    int* G = new int[n];

    for (int i = 0; i < n; i++)
    {
        M[i] = new int[m];
    }
    SetColor(LightBlue, Black);
    printf("Матрица смежности : \n");

    for (int i = 0; i < n; i++) // цикл по строкам
    {
        for (int j = 0; j < m; j++) // цикл по столбцам
        {
            if (i == j)
            {
                M[i][j] = 0;
            }
            else if (i != j)
            {
                M[i][j] = rand() % 3;

                M[j][i] = M[i][j];
            }
        }
    }
    for (int i = 0; i < n; i++) // цикл по строкам
    {
        for (int j = 0; j < m; j++) // цикл по столбцам
        {
            if (M[i][j] != 0)
            {
                k++;
            }
            if (M[i][j] != 0 and i > j)
            {
                c++;
            }
            SetColor(LightBlue, Black);
            printf("%d ", M[i][j]);
        }
        G[i] = k;
        k = 0;
        printf("\n");
    }
}
```

```

    }
    fwrite(&n, sizeof(int), 1, file); //записываем число элементов в массиве (для его чтения далее)
    fwrite(M, sizeof(int), n, file); //запись самого массива
    fclose(file); //закрываем файл
    printf("\n\n");

    for (int i = 0; i < n; i++)
    {
        if (G[i] != 0)
        {
            printf("Изолированных вершин нет\n");
            break;
        }
        if (G[i] == 0) {
            printf("Изолированная вершина %d\n", i + 1);
        }
    }

start2:
    printf("Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из
программы \n");
    scanf_s("%d", &v);
    if (v == 1) {
        system("PAUSE");
        system("cls");
        goto start;
    }
    else if (v == 2) {
        system("PAUSE");
        system("cls");
        printf("Выход");
        exit(0);
    }
    else(v != 1 && v != 2); {
        printf("Ошибка, попробуйте снова\n");
        system("PAUSE");
        system("cls");
        goto start2;
    }

}

case 2:
{
    printf("Введите количество вершин в графе: \n");
    scanf_s("%d", &n);
    file = fopen("output.bin", "wb"); //wb - флаг на запись бинарного файла

    int v;
    int* L = new int[n];
    int** R = new int* [n];
    m = n;
    for (int i = 0; i < n; i++)
    {
        R[i] = new int[m];
    }
    SetColor(LightBlue, Black);
    printf("Матрица смежности:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            SetColor(LightBlue, Black);

```

```

        printf("R[%d][%d] = ", i, j);
        scanf_s("%d", &R[i][j]);
    }
    printf("\n");
}
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        printf("%d ", R[i][j]);
    }
    printf("\n");
}
for (int i = 0; i < n; i++) // цикл по строкам
{
    for (int j = 0; j < m; j++) // цикл по столбцам
    {
        if (R[i][j] != 0)
        {
            k++;
        }
        if (R[i][j] != 0 and i > j)
        {
            c++;
        }
    }

    L[i] = k;
    k = 0;
}

fwrite(&n, sizeof(int), 1, file); //записываем число элементов в массиве (для его чтения далее)
fwrite(R, sizeof(int), n, file); //запись самого массива
fclose(file); //закрываем файл

for (int i = 0; i < n; i++)
{
    if (L[i] != 0)
    {
        printf("Изолированных вершин нет\n");
        break;
    }
    if (L[i] == 0) {
        SetColor(LightBlue, Black);
        printf("Изолированная вершина %d\n", i + 1);
    }
}

start3:
printf("Нажмите 1, если хотите вернуться в стартовое меню или 2 если хотите выйти из
программы \n");
scanf_s("%d", &v);
if (v == 1) {
    system("PAUSE");
    system("cls");
    goto start;
}
else if (v == 2) {
    system("PAUSE");
    system("cls");
    printf("Выход");
    exit(0);
}
else(v != 1 && v != 2); {
    printf("Ошибка, попробуйте снова\n");
    system("PAUSE");
}

```

```

        system("cls");
        goto start3;
    }
}

case 3:
{
    system("PAUSE");
    system("cls");
    printf("Выход");
    exit(0);
}

default:
{
    printf("Ошибка\n");
    system("PAUSE");
    system("cls");
    goto start;
}
}
}

```