

**Содержание**

Реферат 4

Введение 5

1. Постановка задачи 7
2. Теоретическая часть задания 8
3. Описание алгоритма программы 10
4. Описание программы11
5. Тестирование 14
6. Ручной расчет задачи 18

Заключение 20

Список литературы21

Приложение A. Листинг программы22

**Реферат**

Отчет 23 стр, 18 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, ИЗОЛИРОВАННЫЕ ВЕРШИНЫ

РЕАЛИЗАЦИЯ АЛГОРИТМА НАХОЖДЕНИЯ ИЗОЛИРОВАННЫХ ВЕРШИН В ГРАФЕ

Цель исследования – разработка программы для нахождения изолированных вершин в графе.

В работе создан алгоритм поиска изолированных вершин, в результате которого находится изолированные вершины. Так же программа имеет графический интерфейс, выборность заполнения граф случайным или ручным способом и возможность установки связности вершин в графе. В данном алгоритме считается степень вершины.

**Введение**

Язык программирования Cи [4] был создан Деннисом Ритчи из AT&T Bell Laboratories как язык программирования высокого уровня для разработки операционной системы UNIX. В настоящее время язык широко используется в различных областях. Cи особенно популярен у системных программистов, потому что позволяет писать программы просто и кратко.

Язык Си считается языком системного программирования, хотя он удобен и для написания прикладных программ. Среди преимуществ языка Си следует отметить переносимость программ на компьютеры различной архитектуры и из одной операционной системы в другую, лаконичность записи алгоритмов, логическую стройность программ, а также возможность получить программный код, сравнимый по скорости выполнения с программами, написанными на языке ассемблера.

**Теория графов** - один из обширнейших разделов дискретной математики, широко применяется в решении экономических и управленческих задач, в программировании, химии, конструировании и изучении электрических цепей, коммуникации, психологии, психологии, социологии, лингвистике, других областях знаний. **Теория графов** систематически и последовательно изучает свойства графов, о которых можно сказать, что они состоят из множеств точек и множеств линий, отображающих связи между этими точками.

Основателем теории графов считается Леонард Эйлер (1707-1882), решивший в 1736 году известную в то время задачу о кёнигсбергских мостах.

**Графом называется** система объектов произвольной природы (вершин) и связок (рёбер), соединяющих некоторые пары этих объектов.

В теории графов вершиной называется фундаментальная единица, образующая графы.

Степенью вершины графа называется число рёбер, инцидентных ей. Вершина называется изолированной, если её степень равна нулю. То есть это вершина, не являющаяся конечной ни для какого ребра.

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

1. **Постановка задачи**

Цель исследования – разработка программы для нахождения изолированных вершин в графе.

Пользователь должен иметь возможность сам вводить размер графа. В реализованной программе должна присутствовать функция выбора автоматического или ручного введения графа. Программа должна иметь либо графическое, либо текстовое меню.

Устройство ввода – клавиатура и мышь.

Задания выполняются в соответствии с вариантом №8 (Реализация алгоритма нахождения изолированных вершин в графе).

1. **Теоретическая часть задания**

**Графом, согласно определениям [1-2],** называется геометрическая фигура, состоящая из точек и соединяющих их линий. Точки называются **вершинами** графа, а линии — **ребрами**. Каждому ребру сопоставлены две вершины — концы ребра. Пример графа представлен на рис.1.

Если некоторое ребро u соединяет две вершины A и B графа, то говорят, что ребро u **инцидентно** вершинам A и B, а вершины в свою очередь **инцидентны** ребру u. Вершины, соединенные ребром, также называют **смежными**.

**Степенью вершины** называют количество ребер, для которых она является концевой (при этом петли считают дважды).

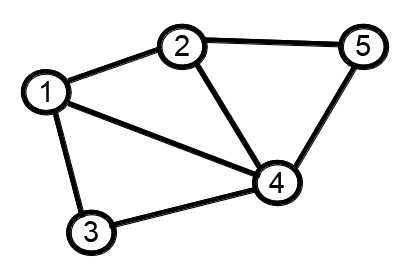


Рисунок 1 – Пример графа

Изолированные вершины—это такие вершины, которые не имеют инцидентных ребер, т.е. их степень нулевая. Из всего этого следует, что изолированные вершины недостижимы их любых других вершин. Висячие вершины—это такие вершины, которые имеют только одно инцидентное ребро.

Изолированная вершина графа имеет степень 0, а висячая —степень 1.

Граф, состоящий только из изолированных вершин, называется нуль - графом.

На рис.2 представлен пример графа. У данного графа вершина №5 является изолированной, т.к. ее степень равна нулю, а вершина №3 – висячей, т.к. ее степень равна 1.

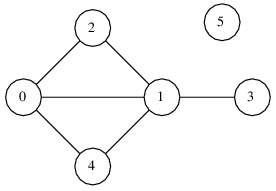


Рисунок 2 – пример графа с изолированной и висячей вершинами.

1. **Описание алгоритма программы**

Для реализации программы был применен данный алгоритм:

Задается ненаправленный граф матрицей смежности. В цикле смотрим элементы построчно. Первый цикл от нуля до n (n- количество вершин в графе). В нем переменную проверки изолированной вершины обнуляем. Далее еще один цикл до n. В нем при каждой единице в строке матрицы переменная прибавляет единицу. После цикла переменная проверяется, если она равна нулю, то выводится, т.к. эта вершина изолированная.

В данном алгоритме считается степень вершины.

Ниже представлен псевдокод алгоритма поиска изолированных вершин в графе.

Алгоритм нахождения изолированных вершин в графе.

Вход: size - размер матрицы, m[i, j] – матрица смежности

Выход: iso – количество изолированных вершин.

1. для i = 0; пока i < num делать i = i + 1
   1. step= 0;
   2. для j = 0; пока j < num; делать j=j+1
      1. если m[i][j] == 1 делать
         1. step ++
      2. если step == 0 делать
         1. вывести i
         2. iso = iso + 1

Полный код программы можно увидеть в Приложении А.

1. **Описание программы**

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

Работа программы начинается с открытия консоли (рис.3).

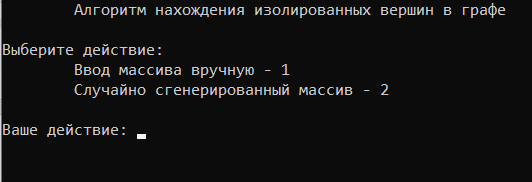


Рисунок 3 – начало работы программы

Пользователю предлагается выбрать способ задания графа: случайно сгенерированный (нажать 1) или вручную (нажать 2).

При выборе ручного ввода графа нужно указать размерность матрицы (рис.4).

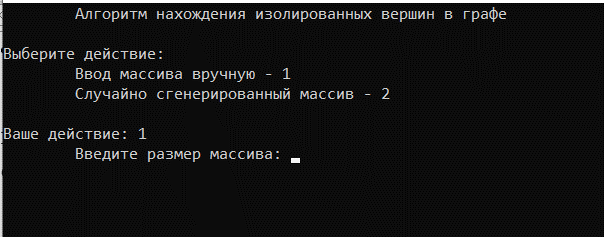


Рисунок 4 – Ввод размерности массива

После ввода матрицы появится информация о том, какие вершины изолированы и сколько их (рис.5).

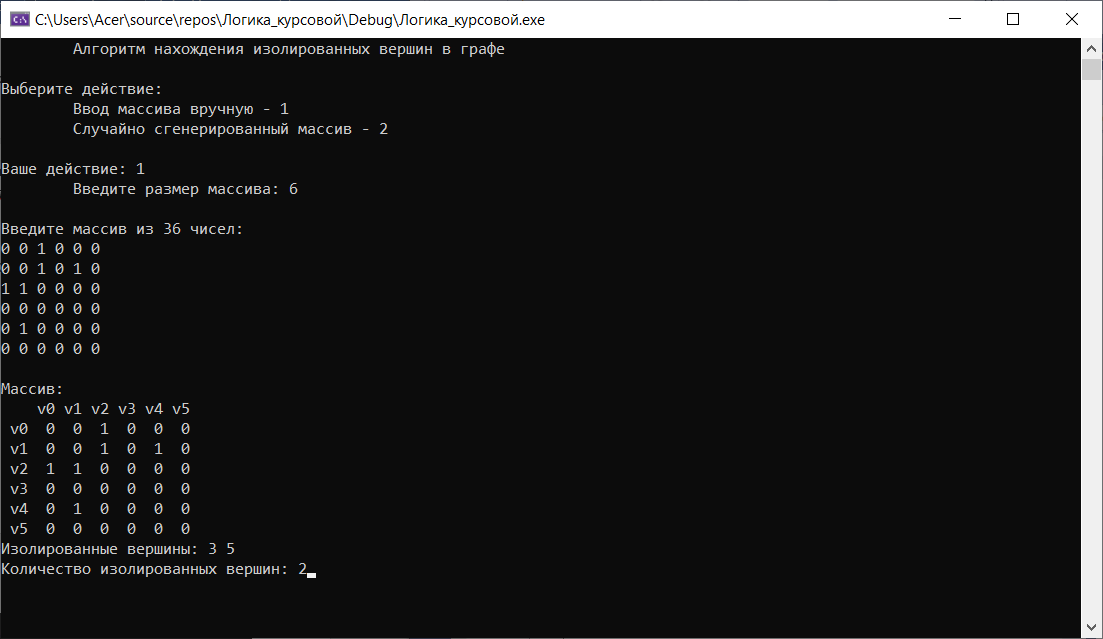


Рисунок 5 – Ручной ввод матрицы

При выборе случайно генерации так же нужно указать размерность матрицы (рис.6).

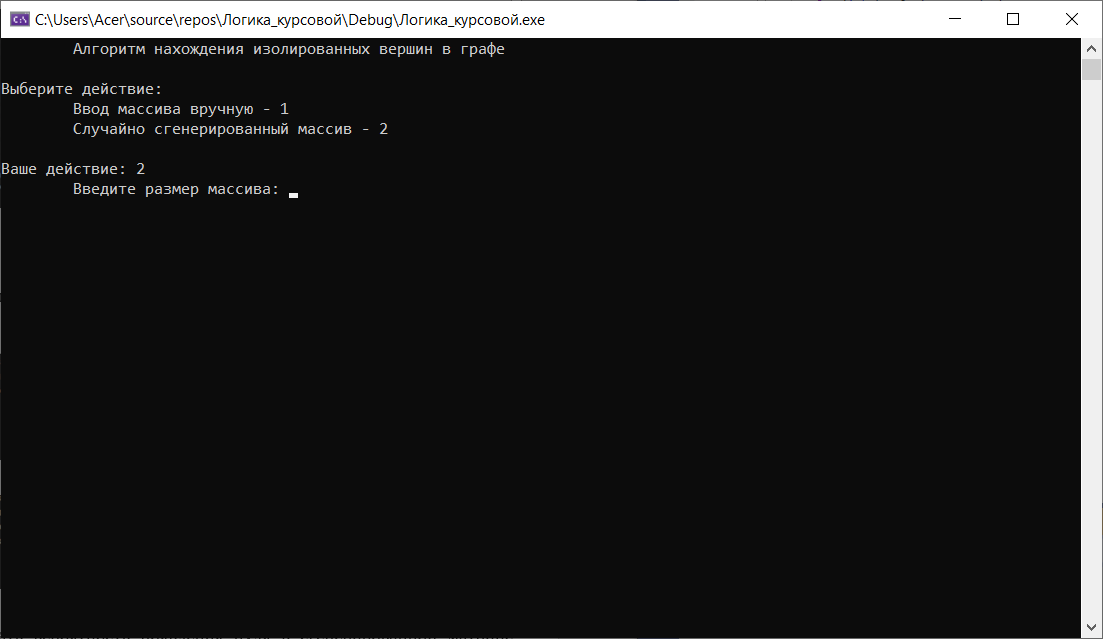


Рисунок 6 – Случайный ввод матрицы

Далее нужно указать вероятность появления нуля в сгенерированной матрице. Вводить стоит дробное число от 0 до 1. 0 – низкая вероятность появления нуля; 0,9 – высокая вероятность появления нуля (рис.7).

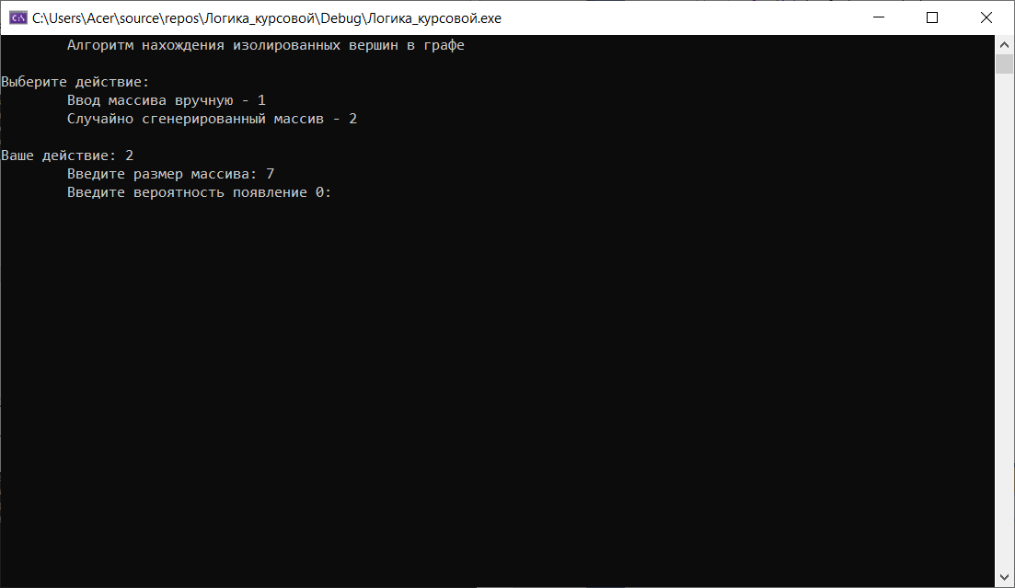


Рисунок 7 – Ввод вероятности появления 0

После генерации матрицы выходит информация о изолированных вершинах и их количестве (рис. 8).

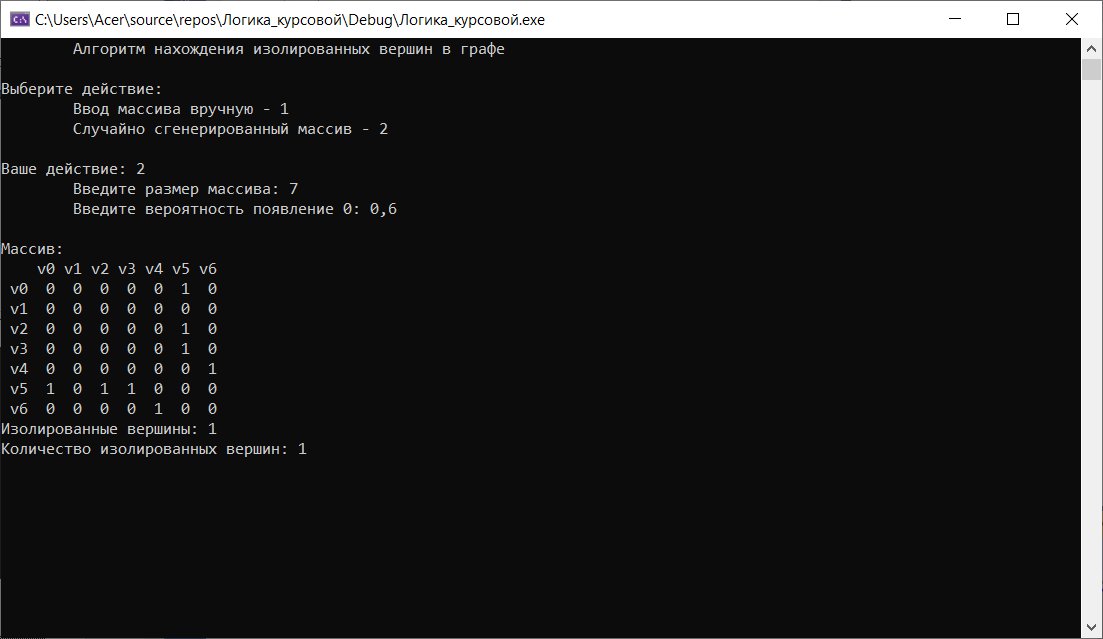


Рисунок 8 – Работа программы при случайном вводе матрицы

1. **Тестирование**

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Ожидаемый результат** | **Полученный результат** |
| Запуск программы | Открытие консольного приложения | Верно |
| Выбор режима ввода матрицы | Результат в зависимости от выбора пользователя | Верно |
| Ручной ввод матрицы | Ручной ввод матрицы и действия над ней | Верно |
| Вывод изолированных вершин | Вывод в консоль номер изолированных вершин | Верно |
| Вывод количество изолированных вершин | Вывод в консоль количество изолированных вершин | Верно |
| Рандомный ввод матрицы | Матрица генерируется рандомно с заданной размерностью и вероятностью | Верно |

Тестирование проводилось на различных этапах создания программы. В ходе тестирования было выявлено и исправлено несколько ошибок.

Ниже представлены результаты тестирования при ручном и случайном вводе матрицы.

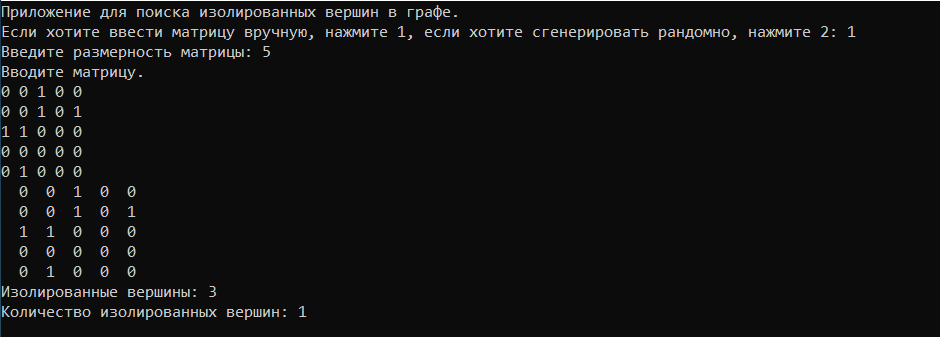


Рисунок 9 – Тестирование программы при ручном вводе матрицы

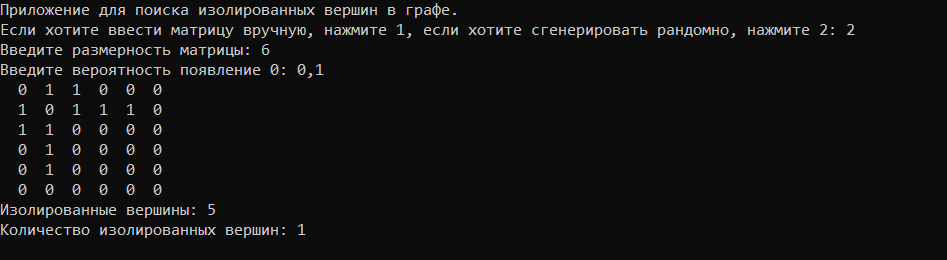


Рисунок 10 – Тестирование программы при случайном вводе матрицы размерностью 6 и вероятностью 0,1

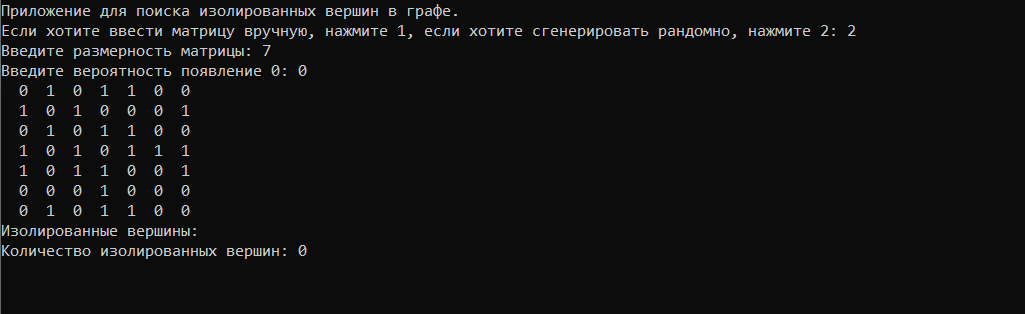


Рисунок 11 – Тестирование программы при случайном вводе матрицы размерностью 7 и вероятностью 0

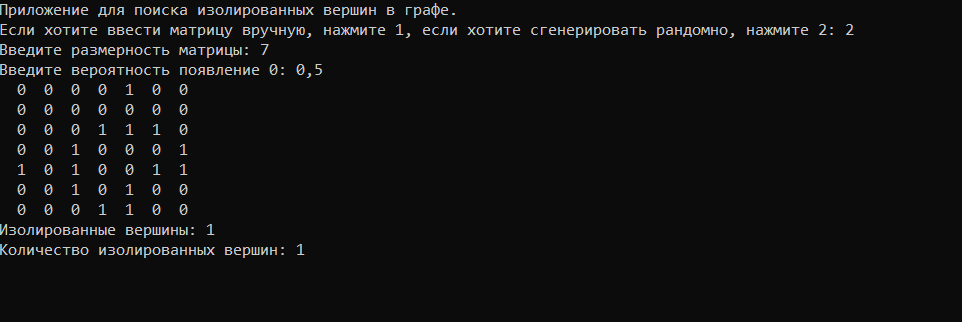


Рисунок 12 – Тестирование программы при случайном вводе матрицы размерностью 7 и вероятностью 0,5.

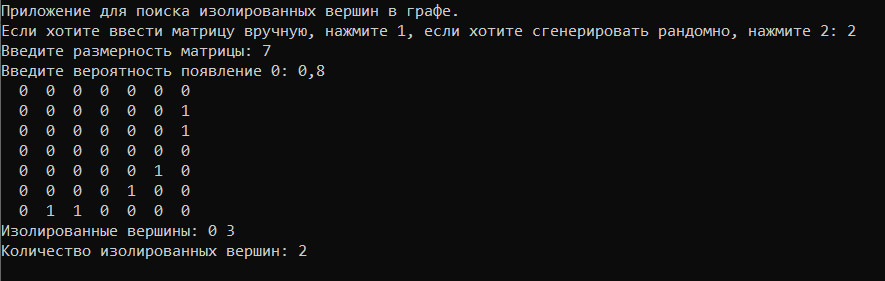


Рисунок 13 – Тестирование программы при случайном вводе матрицы размерностью 7 и вероятностью 0,8.

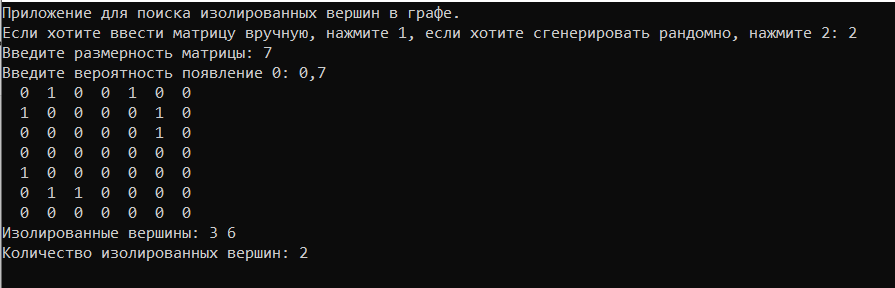


Рисунок 14 – Тестирование программы при случайном вводе матрицы размерностью 7 и вероятностью 0,7.

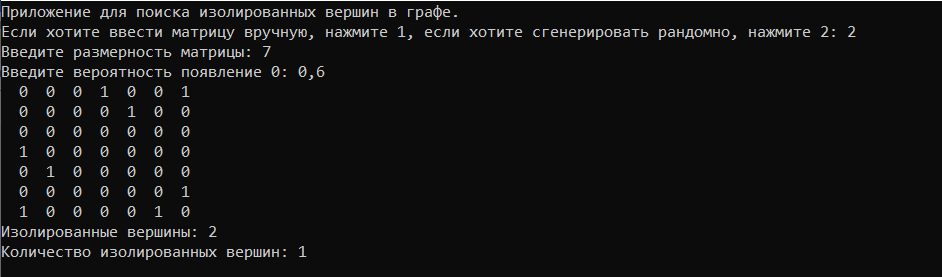


Рисунок 15 – Тестирование программы при случайном вводе матрицы размерностью 7 и вероятностью 0,6.

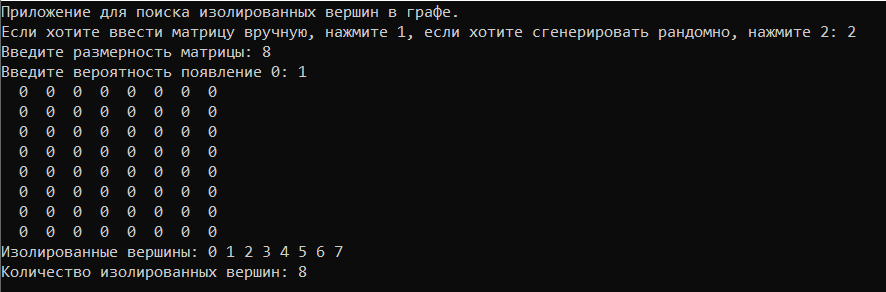


Рисунок 16 – Тестирование программы при случайном вводе матрицы размерностью 8 и вероятностью 1

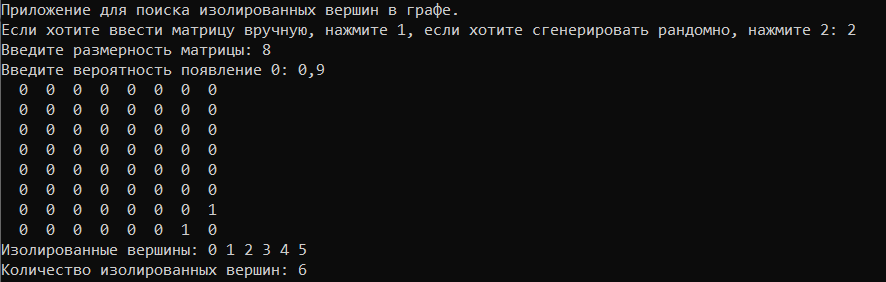


Рисунок 16 – Тестирование программы при случайном вводе матрицы размерностью 8 и вероятностью 0,9

В результате тестирования было доказано, что алгоритм работает верно

1. **Ручной расчет задачи**

Проведем ручную проверку задачи на примере данного графа размерность 6 и вероятностью появления нуля 0,7 (рис.18):

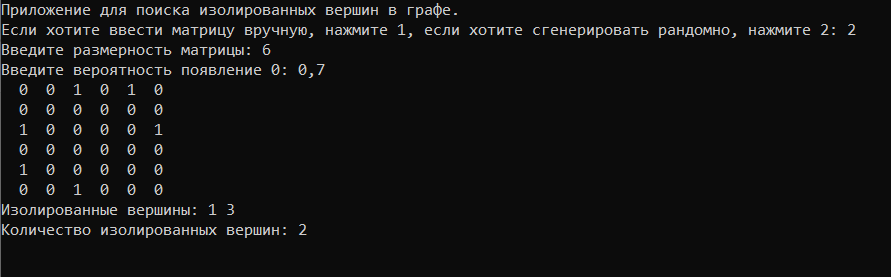


Рисунок 18 – Пример для ручного расчета

При случайной генерации мы получаем данную матрицу смежности:

0 0 1 0 1 0

0 0 0 0 0 0

1 0 0 0 0 1

0 0 0 0 0 0

1 0 0 0 0 0

0 0 1 0 0 0

Далее обнуляется счетчик изолированных вершин. Программа проверяет матрицу построчно. Перед проверкой очередной строки обнуляется переменная для проверки, step. В цикле просматривается каждый элемент первой строчки. Если программа встречает 1, то к переменной step прибавляется 1. К концу первой строки step равна 2. После прохождения первой строки проверка, является ли данная вершина изолированной. Т.к. step не равно 0, то вершина №0 не изолирована. Переходим к проверке следующей строки. После прохода через все элементы step равняется нулю, следовательно вершина изолированная. Выводится номер этой вершины и к счетчику изолированных вершин прибавляется 1. После проверки следующей строки step равняется двум, следовательно вершина не изолированная и переходим к проверке следующей строки. После проверки 4 строки переменная step равняется 0, следовательно вершина изолированная, выводится номер вершины и к счетчику изолированных вершин прибавляется единица. После проверки последних двух строк переменная step равняется единице, следовательно они не являются изолированными. В итоге получается только две изолированные вершины, это первая и третья.

Сравнив результат работы программы с ручным расчетом, осуществленным с помощью [2-3], можно сказать, что программа работает верно.

**Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения изолированных вершин в графе в среде разработки Microsoft Visual Studio 2019.

При выполнении данной работы были получены навыки работы с графом, представленным матрицей смежности, и был разработан и реализован алгоритм нахождения изолированных вершин в графе. Так же были улучшены навыки программирования на языке Си, были отработаны некоторые задачи, схожие с [4-5].

Программа имеет небольшой, вполне, но достаточный для использования функционал возможностей и интерфейс.

**Список литературы**

1. Теория графов. Алексеев В.Е., Захарова Д.В. Электронное учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2012. – 57 с.
2. Зыков А.А. Основы теории графов. - М.:Наука, 1987, 384 с.
3. Мельников О. И. Незнайка в стране графов: Пособие для учащихся. Изд. 3-е, стереотипное. М.: КомКнига, 2007. — 160 с.
4. Брайан Керниган, Деннис Ритчи, Алан Фьюэр. Язык программирования Си. Задачи по языку Си. — Москва: Финансы и статистика, 1985. — 279 с.
5. Пахомов Б.И. C/C++ и MS Visual C++ 2008 для начинающих. Учебник. 2009 год. 609 стр.
6. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. Из серии "300 лучших учебников".2003 год. 461 стр.

**Приложение A. Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <locale.h>

#include <queue>

#include <stdio.h>

#include <time.h>

#include <locale>

#include <conio.h>

#include <malloc.h>

using namespace std;

int\* step;

int num, num2;

int\*\* m;

int menu;

int main() {

setlocale(LC\_ALL, "Rus");

printf("\tАлгоритм нахождения изолированных вершин в графе\n\n");

printf("Выберите действие:\n");

printf(" Ввод массива вручную - 1\n Случайно сгенерированный массив - 2\n");

printf("\nВаше действие: ");

scanf("%d", &menu);

printf(" Введите размер массива: ");

scanf("%d", &num);

num2 = num \* num;

m = (int\*\*)malloc(num \* sizeof(int\*));

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

m[i] = (int\*)malloc(num \* sizeof(int));

}

float n;

float prob;

if (menu == 2) {

printf(" Введите вероятность появление 0: ");

scanf("%f", &prob);

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

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

m[i][j] = rand() % 2;

n = ((float)rand()) / RAND\_MAX;

if (n < prob) {

m[i][j] = 0;

}

m[j][i] = m[i][j];

m[i][i] = 0;

}

}

}

else {

printf("\nВведите массив из %d чисел:\n", num2);

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

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

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

}

}

}

printf("\n");

printf("Массив:\n");

printf(" ");

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

printf(" v%d", i);

printf("\n");

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

printf(" v%d", i);

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

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

}

printf("\n");

}

int iso = 0;

printf("Изолированные вершины: ");

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

int step = 0;

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

if (m[i][j] == 1) step++;

}

if (step == 0) {

printf("%d ", i);

iso++;

}

}

printf("\n");

printf("Количество изолированных вершин: %d", iso);

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

free(m[i]);

free(m);

getchar();

getchar();

}