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»

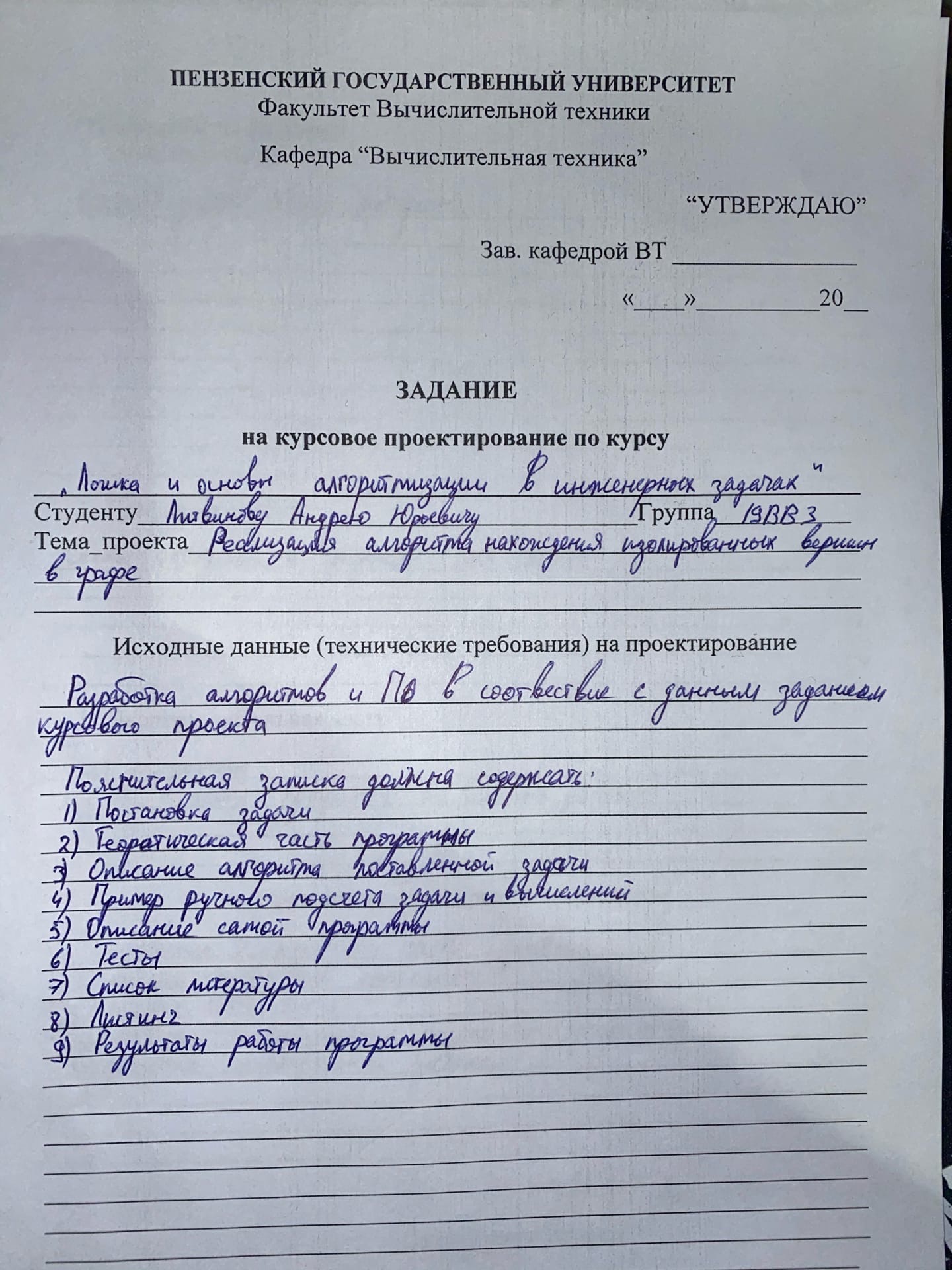
***Выполнил работу студент группы 19ВВ3:***

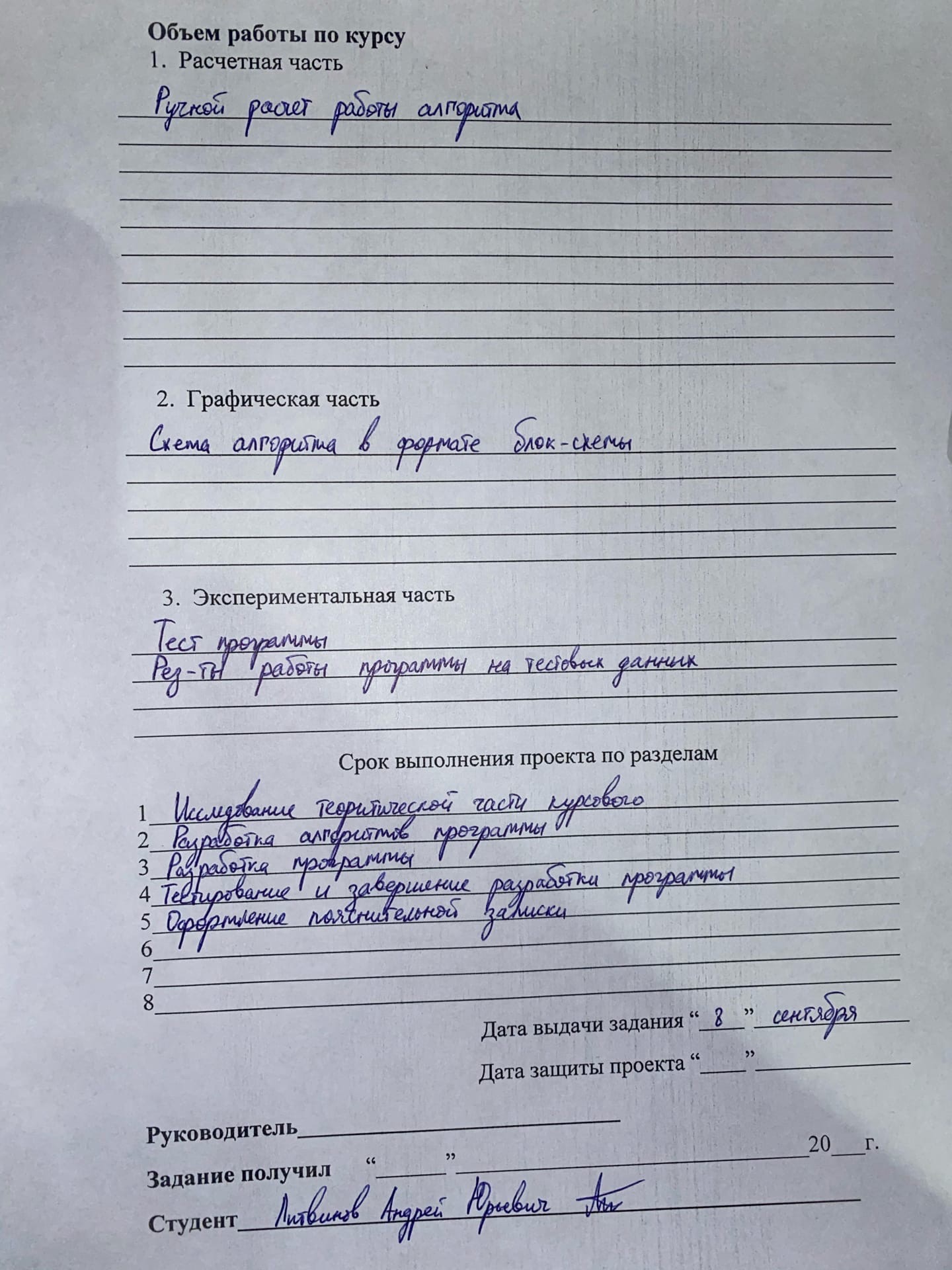
Литвинов А.Ю.

**Приняли:**

д.т.н. профессор Митрохин М. А.

Пенза 2020





**Содержание**

[Реферат 5](#_Реферат.)

[Введение 6](#_Введение)

1. [Постановка задачи 7](#_TOC_250007)
2. [Теоретическая часть задания 8](#_TOC_250006)
3. [Описание алгоритма программы 9](#_TOC_250005)
4. [Описание программы 1](#_Описание_программы)1
5. [Тестирование](#_TOC_250003) 18

[Заключение 26](#_TOC_250001)

[Список литературы 27](#_TOC_250000)

[Приложение A. Листинг программы 28](#_Приложение_А.)

# Реферат.

Отчет по курсовой работе: 31 страниц, 17 рисунков, 1 таблица, 1 приложение и 6 источников.

Цель работы — создание программы для реализации алгоритма нахождения изолированных вершин в графе.

Ключевые слова: алгоритм, граф, матрица, массив, консоль, цикл, размер матрицы, вершина, файл, функция, программа.

В работе рассмотрен возможный способ нахождения изолированных вершин в данном графе. Реализован способ с помощью алгоритма основанного на условии предполагающем нахождение отсутствия связи между вершинами.

# 

# Введение

Изолированная вершина — вершина, не соединённая с другими вершинами ни дугами, ни рёбрами. однако такая вершина может иметь петли. Так же можно сказать, что это вершина без инцидентных ей дуг или ребер. А в случае графа без петель это вершина степени 0.

В данной работе предполагается создание или наличие матрицы, не имеющей петли. Главная диагональ матрицы смежности заполнена нулями.

В качестве среды разработки была выбрана программа VisualStudio2019, язык программирования Си.

Целью данной курсовой работы является создание программы для реализации алгоритма нахождения изолированных вершин в неориентированном и невзвешенном графе.

# 

# Постановка задачи

Необходимо осуществить программную реализацию алгоритма поиска изолированных вершин графа.

Пользователь должен вводить степень порядка графа. В соответствии с этим после обработки данных на экран должна выводиться матрица смежности. Необходимо предусмотреть возможность возникновения ошибок.

Устройствами ввода являются клавиатура и мышь. Необходимо научиться анализировать и различать события, возникающие при работе с устройствами ввода. Необходимо однозначно идентифицировать и выполнять те или иные действия в зависимости от действий пользователя. Это необходимо для легкого использования программы.

Дан невзвешенный и неориентированный граф, предполагается, что граф не содержит циклов отрицательного веса. Программа должна быть построена так, чтобы пользователь мог вводить порядок массива и вероятность отсутствия связей между вершинами графа. После обработки этих данных на экран должна выводиться сгенерированная матрица и номера изолированных вершин (если таковые имеются) или их отсутствие. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

# 

# Теоретическая часть задания

Представим неориентированный, невзвешенный граф. Представленный граф имеет четыре вершины, из которых две вершины V2 и V3 имеют связь между собой. Оставшиеся две вершины V1 и V4 находятся отдельно от других. Данные вершины по определению являются изолированными. Они не имеют связи между собой и другими вершинами.

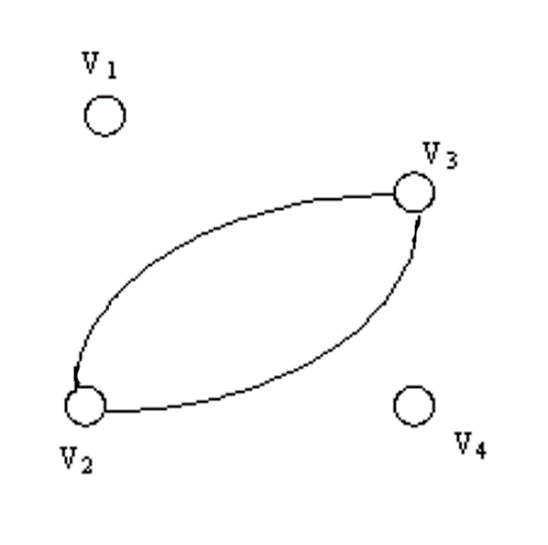


Рисунок 1 – Пример изолированных вершин

Изолированные вершины делятся на два типа. На обычные изолированные вершины, которые не имеют связи с другими вершинами графа, но могут иметь замкнутые цепи, называемые петлями, а также на голые вершины графа, не имеющие не связей, не петель.

В данной программе рассматриваются случаи, когда изолированная вершина не имеет петель, а, следовательно, в матрице смежности она будет иметь нулевую степень связности. Именно на этом основана программа для реализации алгоритма поиска таковых вершин.

# 

# Описание алгоритма программы

Для реализации алгоритма поиска изолированных вершин сформируем матрицу смежности **D[][]** графа **G=(V, E)**, в котором каждая вершина пронумерована от **1** до **|V|**. Эта матрица имеет размер**|V|´|V|**, и каждому ее элементу **D[i][j]** присвоено значение определяющее наличие связи между вершинами. По мере выполнения алгоритма, данная матрица будет анализироваться: каждая вершина будет пропускаться через алгоритм и проверяться на наличие нулевой степени связности.

Алгоритм создает одномерный массив и записывает в него степень связности проанализированных вершин. Если в этом массиве присутствует нулевое значение степени связности на экран выводится номер соответствующей вершины. Если таких вершин нет, выводится сообщение об их отсутствии.

Ниже представлен псевдокод функции :

1. Для i=0 пока i<n делать i++
2. pow\_matrix = 0
3. Для j=0 пока j<n делать j++
4. Если с arr[i][j] > 0
5. pow\_matrix++
6. Конец условия
7. Конец цикла
8. pow1[i] = pow\_matrix
9. Конец цикла
10. Для i=0 пока i<n делать i++
11. Если pow1[i] = 0
12. m = i + 1
13. Вывод m
14. Test ++
15. Конец условия
16. Конец цикла
17. Если test = 0
18. Вывод “Таких вершин нет”
19. Конец условия
20. Test = 0
21. Вывод \n

# Описание программы

Для разработки данной программы использован язык программирования Си.

Си - язык программирования общего назначения, который очень известен своей эффективностью, экономичностью ресурсов и простотой переноса на другие программные платформы. Все эти преимущества обеспечивают качество разработки программного обеспечения, их быстроту и сравнительно небольшой размер.

Проект был создан в виде консольного приложения Win32 (Visual C++),и является многомодульным, поскольку состоит из нескольких функций.

Работа программы начинается с того, что пользователь выбирает, как ему задать граф (сгенерировать в программе случайно или загрузить граф из файла).

Пример мы можем увидеть на рисунке 2.

printf("\n Введите цифру 1 для рандомного заполнения ");

printf("\n Введите цифру 2 считывания массива из файла ");

printf("\n Введите 0 чтобы выйти \n");

printf(" Ожидается ввод: —> ");

scanf("%d", &x);

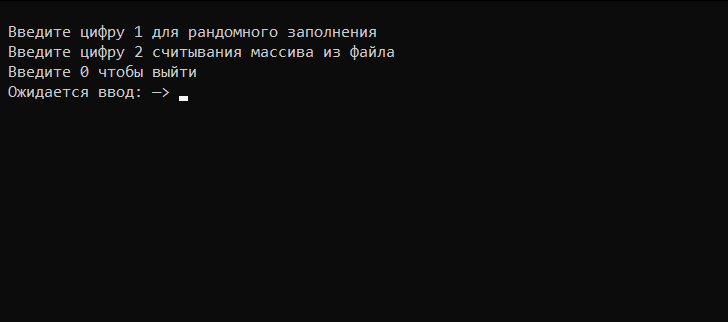


Рисунок 2 – Открытие основного меню

Если пользователь выбирает 1 пункт меню, программа выводит запрос от пользователя порядка массива который будет создан рандомно, вероятность наличия связей в графе также вводится пользователем в процентах. Далее выполняется функция malloc, отвечающая за динамическое выделение памяти. Исходя из введенного размера массива, рандомно генерируется массив и заполняется числами 1 и 0. На экран выводится заданный массив

void random() {

int test = 0;

int c;

int min = 1000;

srand(time(NULL));

printf("Введите порядок массива - ");

scanf("%d", &n);

printf("Введите вероятность выпадения 0 в процентах - ");

scanf("%d", &c);

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

for (int i = 0; i < n; i++) arr[i] = (int\*)malloc(n \* sizeof(int));

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

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

if (j > i) {

int v;

v = rand() % 100;

if (v > c) {

arr[i][j] = 1;

}

else {

arr[i][j] = 0;

}

arr[j][i] = arr[i][j];

}

else if (i == j) arr[i][j] = 0;

}

}

print(arr, n);

printf(" ");

}

За вывод массива на экран отвечает функция print. Она же автоматически сохраняет выведенный массив в созданный программой файл data.txt.

void print(int\*\* mass, int count) {

data = fopen("data.txt", "a");

printf(" ");

fprintf(data, " ");

for (int k = 1; k <= count; k++) {

printf("%d ", k);

fprintf(data, "%d ", k);

}

printf("\n —---------\n");

fprintf(data, "\n —---------\n");

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

printf("%d |", i + 1);

fprintf(data, "%d |", i + 1);

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

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

fprintf(data, "%d ", mass[i][j]);

}

printf("\n");

fprintf(data, "\n");

}

fclose(data);

Далее происходит вызов функции kurs, в которой производится поиск изолированных вершин. Полученный результат сохраняется в уже созданный файл data.txt

void kurs() {

int test = 0;

int m = 0;

int\* pow1;

int pow\_matrix;

srand(time(NULL));

pow1 = dynamic\_dist(n);

generate\_pow1(pow1, n);

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

pow\_matrix = 0;

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

if (arr[i][j] > 0) pow\_matrix++;

}

pow1[i] = pow\_matrix;

}

data = fopen("data.txt", "a");

printf("Изолированные вершины - ");

fprintf(data, "Изолированные вершины - ");

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

if (pow1[i] == 0) {

m = i + 1;

printf(" %d ", m);

test++;

fprintf(data, "%d ", m);

}

}

if (test == 0) {

printf("Таких вершин нет");

fprintf(data, "Таких вершин нет");

}

test = 0;

fprintf(data, "\n");

fclose(data);

printf("\n");

}

Для одномерного массива pow1, который используется для записи степеней каждой из вершин выделяется память с помощью функции dynamic\_dist.

int\* dynamic\_dist(int count) {

int\* dist;

dist = (int\*)malloc(count \* sizeof(int));

return dist;

}

После этого сгенерированный одномерный массив заполняется нулями с помощью функции generate\_pow1

void generate\_pow1(int\* massiv, int count) {

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

massiv[i] = 0;

}

}

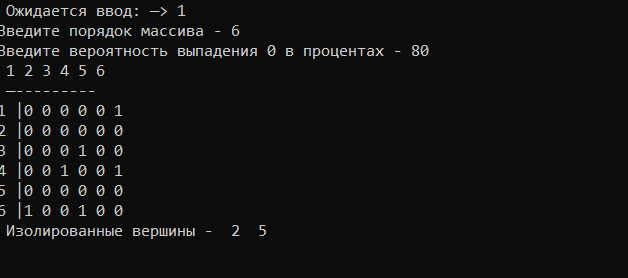


Рисунок 4 – Работа 1 пункта меню

Далее рассмотрим второй пункт основного меню. Этот пункт отвечает за считывание массива из файла. Вызывается функция file. Пользователю представляется возможность самостоятельно ввести название заранее созданного файла из которого будет считан массив. Далее программа проверяет содержание файла. Если там присутствуют символы, то на экран выводится сообщение "Неверный формат данных". После этого программа считает количество элементов в файле и записывает их в переменную n, != EOF означает что программа дошла до конца файла. После этого функция rewind переносит указатель на начало файла. Далее программа считает количество вершин полученный матрицы, путем вычисления квадратного корня из переменной n. После этого, программа безошибочно считывает матрицу из файла. Выводится текст «Массив считан из файла». И вызывается функция print отвечающая за вывод матрицы в консоль.

void file() {

char OPEN = 'A';

char name[50];

FILE\* fp;

int s, k = 0;

int v;

int num;

printf("Введите название файла для чтения значений, но не более 50 символов (Пример: vvod.txt):");

scanf("%s", name);

if ((fp = fopen(name, "r")) == NULL) {

while ((OPEN != '1')) {

printf("Данного файла не существует. Проверьте правильность ввода названия файла!\n");

printf("Для выхода из программы нажмите - 1\n");

scanf("%c", &OPEN, 1);

if (OPEN == '1') {

system("cls");

exit(0);

}

}

}

if (OPEN == 'A') {

while ((fscanf(fp, "%d", &s) != EOF)) {

v = (fscanf(fp, "%d", &s));

if (v == 0) {

printf("Неверный формат данных! Проверьте файл.\n");

exit(0);

break;

}

}

}

fclose(fp);

fp = fopen(name, "r");

if (OPEN == 'A') {

while ((fscanf(fp, "%d", &s) != EOF)) {

if (!fp) break;

n += 1;

}

rewind(fp);

num = sqrt(n);

printf("%d\n", num);

arr = dynamic(num);

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

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

fscanf\_s(fp, "%d", &arr[i][j]);

}

}

printf("Массив считан из файла.\n");

print(arr, num);

fclose(fp);

n = num;

}

}

Далее программа вновь вызывает уже упомянутую функцию kurs. Работоспособность второго пункта показана на рисунке 5.

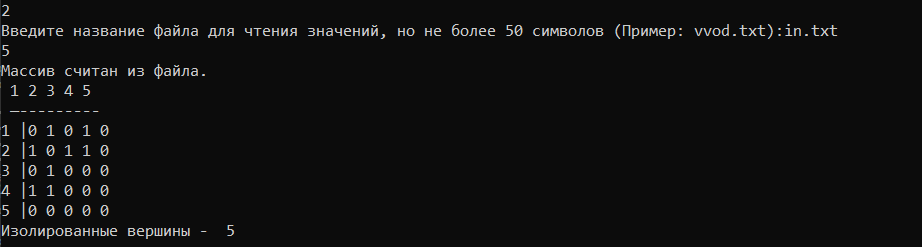


Рисунок 5– Работа 2 пункта меню

За работоспособность всех пунктов меню отвечает функция main, которая проверяет значения, введенные пользователем, и выполняет необходимые для него действия. В ней содержится цикл while, который проверяет данные введенные пользователем.

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int x;

printf("\n Введите цифру 1 для рандомного заполнения ");

printf("\n Введите цифру 2 считывания массива из файла ");

printf("\n Введите 0 чтобы выйти \n");

printf(" Ожидается ввод: —> ");

scanf("%d", &x);

data = fopen("data.txt", "w");

fclose(data);

while (x != 0) {

if (x == 1) {

random();

kurs();

}

if (x == 2) {

file();

kurs();

}

if (x > 2) {

printf("Такого пункта меню не существует \n");

}

scanf("%d", &x);

}

}

Пользователь всегда может завершить работу программы нажатием клавиши 0. Это можно рассмотреть на рисунке 6.

case 0:

return(0);

}



Рисунок 6 – Выход из программ

# Тестирование

Среда разработки MicrosoftVisualStudio 2019 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 и выводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных количеств вершин и выбор разных пунктов программы.

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Описание теста** | **Ожидаемый результат** | **Полученный результат** |
| 1 | Запуск программы | Вывод сообщения о выборе:  сгенерировать матрицу или  вывести матрицу из файла (рис. 7) | Верно |
| 2 | Ввод неверного пункта меню | Вывод сообщения на экран о некорректном вводе (рис. 8) | Верно |
| 3 | Выбор генерации матрицы | Вывод сообщения о  количестве вершин в графе  и вероятности присутствия  ребра между ними (рис. 9) | Верно |
| 4 | Генерация матрицы, и вывод на экран данных изолированных вершин | Вывод матрицы на экран, проверка условия существования и поиск цикла  (рис. 10) | Верно |
| 5 | Генерация матрицы, которая не содержит изолированные вершины | Вывод матрицы на экран, проверка условия существования и вывод соответствующего сообщения на экран (рис. 11) | Верно |
| 6 | Вывод матрицы из файла | Вывод самостоятельно заданной матрицы из файла (рис. 13, рис. 15 ) | Верно |
| 7 | Сохранение результата программы в файл | Успешное сохранение исходной матрицы с результатом поиска (рис. 12, рис. 14) | Верно |

.

В результате тестирования было выявлено, что программа успешно реализует алгоритм нахождения изолированных вершин во всех вариантах генерации матрицы.

* + - 1. Запуск программы и стартовое меню

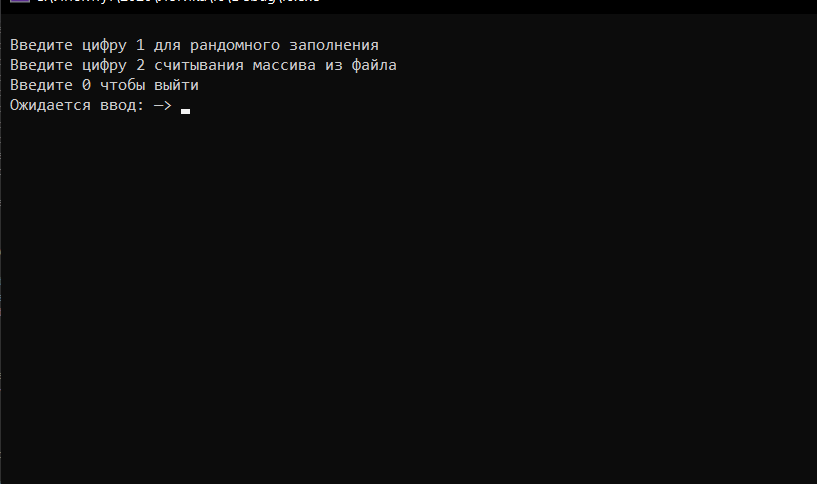


Рисунок 7 – Запуск программы

2. Ввели неверный пункт меню, программа сообщила об ошибке

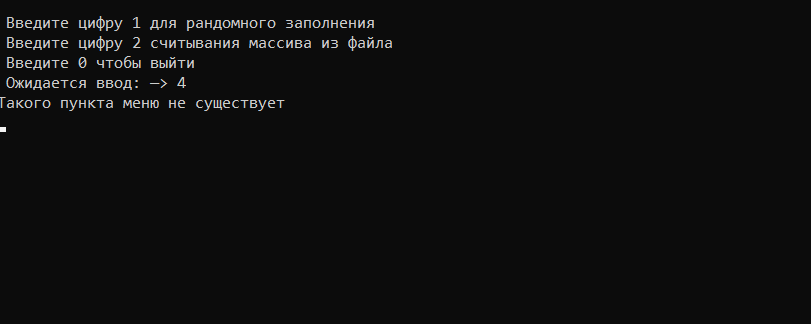


Рисунок 8 – Некорректный ввод

3.Выбран первый пункт меню с рандомной генерацией, и последующий выбор размерности массива с вероятностью возникновения связей между вершинами.

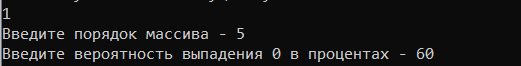


Рисунок 9 – Генерация матрицы размером 5\*5 и вероятностью 60%.

4. Сгенерировали массив, после чего программа произвела поиск изолированных вершин

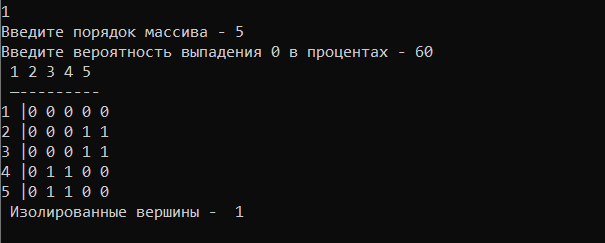


Рисунок 10 – Рандомное заполнение и нахождение изолированных вершин.

5. Сгенерированная матрица не содержит изолированные вершины, выводится соответствующее сообщение.

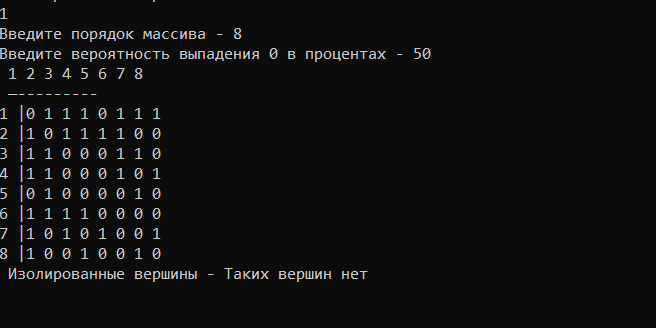


Рисунок 11 – отсутствие изолированных вершин.

1. Результат сохранен в файл.

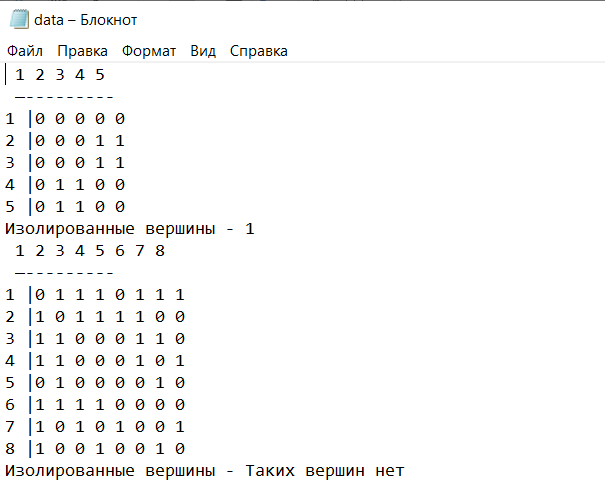


Рисунок 12 – сохранение в файл data.txt произведенное автоматически

6. Запустили программу и ввели пункт меню 2

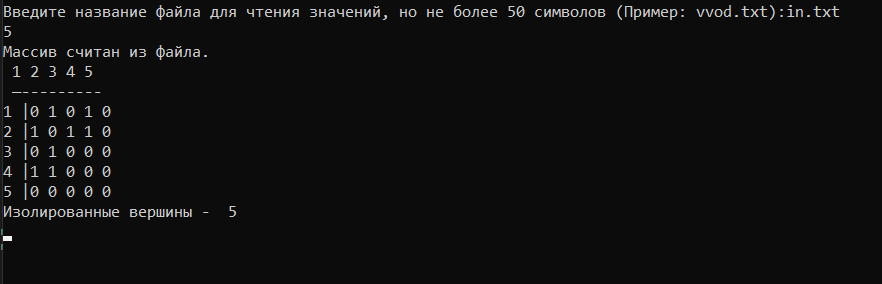


Рисунок 13 – Массив считан и записан из файла in.txt

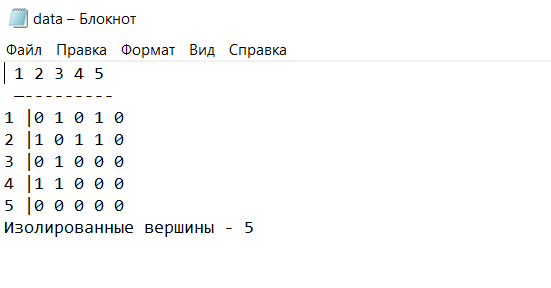


Рисунок 14 – сохранение в файл data.txt произведенное автоматически

Результат полученный при неверном названии файла

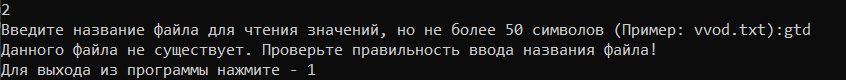


Рисунок 15 – Введено неверное название файла

Для тестирования правильной работоспособности программы, нужно провести ручной просчет алгоритма, для проверки удобно использовать 1 пункт меню программы (ввод массива рандомно).

**Ручной просчет программы.**

Проведем проверку программы посредством ручных вычислений на примере ориентированного графа с 5 вершинами (Рисунок 10).



Рисунок 16 – Проверка вручную

Таким образом, можно сделать вывод, что программа работает верно.

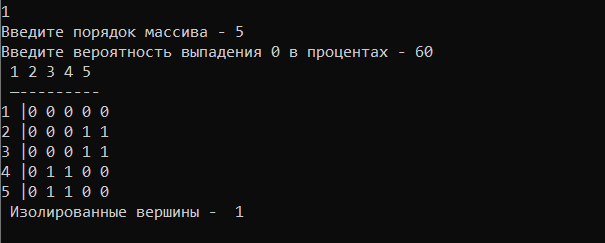


Рисунок 17 – Сравнение с выводом программы

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм для поиска изолированных вершин в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц. Приобретены навыки по осуществлению поиска с возвратом, работы с файлами, а также работы с командной строкой. Углублены знания языка программирования Cи.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

## Список литературы.

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

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

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

Кристофидес Н. Теория графов. Алгоритмический подход. Пер. с анг. - М.:Мир, 1978, 432 с.

<https://www.cyberforum.ru/>

<http://fevt.ru/load/graf/71-1-0-282>

### 

## Приложение А.

**Листинг программы.**

Файл: kurs.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <math.h>

#include <tchar.h>

#include <windows.h>

#include <conio.h>

#include <string.h>

int\*\* arr;

FILE\* data;

int n;

int\*\* array;

int\* dynamic\_dist(int count) {

int\* dist;

dist = (int\*)malloc(count \* sizeof(int));

return dist;

}

void generate\_pow1(int\* massiv, int count) {

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

massiv[i] = 0;

}

}

int\*\* dynamic(int count) {

int\*\* mass;

mass = (int\*\*)malloc(count \* sizeof(int\*));

for (int i = 0; i < count; i++) mass[i] = (int\*)malloc(count \* sizeof(int));

return mass;

}

void print(int\*\* mass, int count) {

data = fopen("data.txt", "a");

printf(" ");

fprintf(data, " ");

for (int k = 1; k <= count; k++) {

printf("%d ", k);

fprintf(data, "%d ", k);

}

printf("\n —---------\n");

fprintf(data, "\n —---------\n");

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

printf("%d |", i + 1);

fprintf(data, "%d |", i + 1);

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

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

fprintf(data, "%d ", mass[i][j]);

}

printf("\n");

fprintf(data, "\n");

}

fclose(data);

}

void kurs() {

int test = 0;

int m = 0;

int\* pow1;

int pow\_matrix;

srand(time(NULL));

pow1 = dynamic\_dist(n);

generate\_pow1(pow1, n);

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

pow\_matrix = 0;

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

if (arr[i][j] > 0) pow\_matrix++;

}

pow1[i] = pow\_matrix;

}

data = fopen("data.txt", "a");

printf("Изолированные вершины - ");

fprintf(data, "Изолированные вершины - ");

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

if (pow1[i] == 0) {

m = i + 1;

printf(" %d ", m);

test++;

fprintf(data, "%d ", m);

}

}

if (test == 0) {

printf("Таких вершин нет");

fprintf(data, "Таких вершин нет");

}

test = 0;

fprintf(data, "\n");

fclose(data);

printf("\n");

}

void random() {

int test = 0;

int c;

int min = 1000;

srand(time(NULL));

printf("Введите порядок массива - ");

scanf("%d", &n);

printf("Введите вероятность выпадения 0 в процентах - ");

scanf("%d", &c);

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

for (int i = 0; i < n; i++) arr[i] = (int\*)malloc(n \* sizeof(int));

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

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

if (j > i) {

int v;

v = rand() % 100;

if (v > c) {

arr[i][j] = 1;

}

else {

arr[i][j] = 0;

}

arr[j][i] = arr[i][j];

}

else if (i == j) arr[i][j] = 0;

}

}

print(arr, n);

printf(" ");

}

void file() {

char OPEN = 'A';

char name[50];

FILE\* fp;

int s, k = 0;

int v;

int num;

printf("Введите название файла для чтения значений, но не более 50 символов (Пример: vvod.txt):");

scanf("%s", name);

if ((fp = fopen(name, "r")) == NULL) {

while ((OPEN != '1')) {

printf("Данного файла не существует. Проверьте правильность ввода названия файла!\n");

printf("Для выхода из программы нажмите - 1\n");

scanf("%c", &OPEN, 1);

if (OPEN == '1') {

system("cls");

exit(0);

}

}

}

if (OPEN == 'A') {

while ((fscanf(fp, "%d", &s) != EOF)) {

v = (fscanf(fp, "%d", &s));

if (v == 0) {

printf("Неверный формат данных! Проверьте файл.\n");

exit(0);

break;

}

}

}

fclose(fp);

fp = fopen(name, "r");

if (OPEN == 'A') {

while ((fscanf(fp, "%d", &s) != EOF)) {

if (!fp) break;

n += 1;

}

rewind(fp);

num = sqrt(n);

printf("%d\n", num);

arr = dynamic(num);

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

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

fscanf\_s(fp, "%d", &arr[i][j]);

}

}

printf("Массив считан из файла.\n");

print(arr, num);

fclose(fp);

n = num;

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int x;

printf("\n Введите цифру 1 для рандомного заполнения ");

printf("\n Введите цифру 2 считывания массива из файла ");

printf("\n Введите 0 чтобы выйти \n");

printf(" Ожидается ввод: —> ");

scanf("%d", &x);

data = fopen("data.txt", "w");

fclose(data);

while (x != 0) {

if (x == 1) {

random();

kurs();

}

if (x == 2) {

file();

kurs();

}

if (x > 2) {

printf("Такого пункта меню не существует \n");

}

scanf("%d", &x);

}

### }