**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_

**ЗАДАНИЕ**

**на курсовое проектирование по курсу**

**Логика и основы алгоритмизации в инженерных задачах**

Студенту Портнову Никите Сергеевичу Группа 20ВВ2   
Тема проекта Реализация операции нахождения двух и более множеств

Исходные данные (технические требования) на проектирование

Разработать алгоритм в соответствии с данным заданием курсового проекта. Пояснительная записка должна содержать: 1. Постановку задачи; 2. Теоретическую часть задания; 3. Описание алгоритма программы; 4. Тестирование; 5. Ручной расчёт задачи; 6. Заключение; 7. Список литературы 8. Листинг

**Объем работы по курсу**

1. Расчетная часть

Ручной расчет работы алгоритма

1. Графическая часть

1. Экспериментальная часть

Тестирование программы; Результат работы программы;

Срок выполнения проекта по разделам

1 Исследования теоретической части курсовой 2 Разработка алгоритма программы

3 Разработка программы 4 Тестирование программы завершение разработки программы 5 Оформление пояснительной задачи

Дата выдачи задания “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты проекта “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Задание получил**      “\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**Студент\_\_\_Портнов\_ Никита Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Министерство 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 операции нахождения двух и более множеств»

Выполнил:

студент группы

Портнов Н.С.

Принял:

Митрохин М.А.

Пенза 2021

**Содержание**

[Реферат 5](#_Toc58799884)

[Введение 6](#_Toc58799885)

[Постановка задачи 7](#_Toc58799886)

[Теоретическая часть задания 8](#_Toc58799887)

[Описание алгоритма программы 10](#_Toc58799888)

[Тестирование 15](#_Toc58799890)

[Ручной расчёт задачи 18](#_Toc58799891)

[Заключение 19](#_Toc58799892)

[Список литературы 20](#_Toc58799893)

[Приложения 21](#_Toc58799894)

# 

# Реферат

ГРАФ, ТЕОРИЯ МНОЖЕСТВ, ОПЕРАЦИИ НАХОЖДЕНИЯ ДВУХ И БОЛЕЕ МНОЖЕСТВ.

Цель исследования – разработка программы, реализующую операции нахождения двух и более множеств.

В работе рассмотрен алгоритм нахождения двух и более множеств.

# Введение

*Теория множеств* — раздел математики, в котором изучаются общие свойства *множеств*.

*Множество* – математический объект, который является совокупностью (т.е. объединением в одно целое) некоторого количества определенных, вполне различимых реальных или абстрактных объектов.

Объекты, которые составляют множества называются *элементами множеств*.

 Множество может быть задано перечислением или с помощью определяющего (характеристического) свойства.

# Постановка задачи

Требуется разработать программу, которая осуществит нахождения, объединение двух и более множеств.

.

Исходный граф в программе должен задаваться матрицей смежности, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь мог выбирать способ создания матрицы смежности (ручной или автоматический метод заполнения), вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности графа. Для совершения нахождения двух и более множеств. Если пользователь введет номер несуществующей команды программа должна вывести соответствующее сообщение с просьбой заново ввести требуемые данные. Необходимо предусмотреть различные исходы, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

# Теоретическая часть задания

**Сведения о множествах и объединении**

Множество может быть задано перечислением или с помощью определяющего (характеристического) свойства.

Например, перечислением заданы следующие множества:

   -    А={1,2,3,5,7} — множество чисел;

   -    Х={x1,x2,...,xn} — множество некоторых элементов x1,x2,...,xn.

   -    Q={«Андрей», «Владимир»,..., «Ярослав»} — множество имён.

С помощью определяющего свойства задано:

 А = { x׀  x ∈ **N** и х < 43}.

Объединение:

Пусть имеются множества A и B. Объединением или суммой этих множеств называется множество, состоящее из всех элементов, принадлежащих хотя бы одному из слагаемых“:

A∪B={x∣x∈A∨x∈B}.

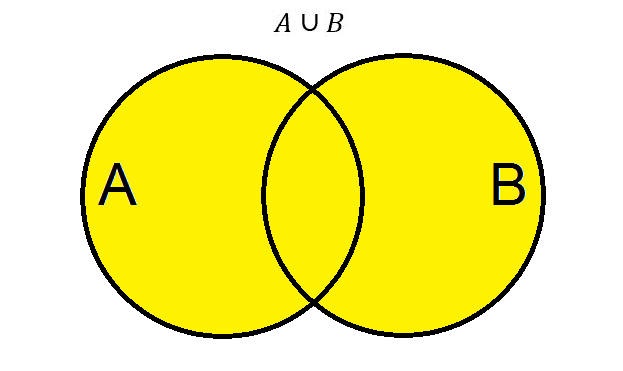
 Свойства:

   а) Если элемент x принадлежит нескольким слагаемым, то он входит в сумму лишь один раз.

   Например,  A={1,2,3,4} и B={3,4,5,6,7}.

   Тогда          A∪B={1,2,3,4,5,6,7}.

б) Объединение множества с самим собой есть это же множество:

A∪A=A. 

  в) Объединение множества со своим подмножеством есть исходное множество:

если B⊆A, то B∪A=A.

 г)  Объединение множества с пустым множеством есть само множество:

    A∪∅=A.

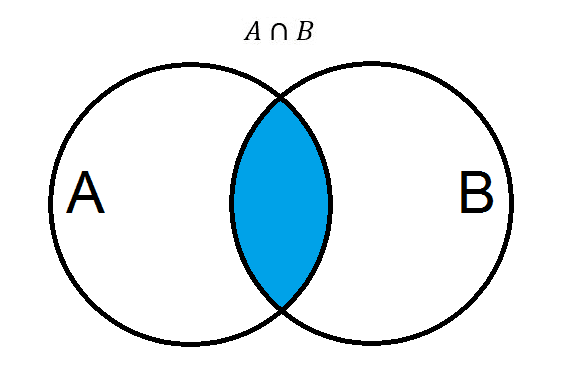
    д) Операция объединения множеств коммутативна:

    A∪B=B∪A.

    е) Операция объединения множеств транзитивна:

    (A∪B)∪C=A∪(B∪C).

2 Пересечением или общей частью множеств A и B называется множество, состоящее из всех тех элементов, которые принадлежат одновременно обоим множествам A и B:

A∩B={x∣x∈A ∧ x∈B}.

Например,  A={1,2,3,4} и B={3,4,5,6}.

   Тогда          A∩B={3,4}.

# Описание алгоритма программы

**Описание работы функции** obe **()**

Реализация состоит из подготовительной части, в которой все вершины помечаются как не посещенные и осуществляется объединение двух и более множеств.

Ниже представлен псевдокод функций BFS()

void obe(int\*\* mass\_o, int\*\* mass\_o2, int\*\* mass\_o3, int\*\* mass\_p1, int\*\* mass\_p2, int n, int n1)

{ //объединение

int i, j;

if (n > n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_o[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

if (n < n1) {

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

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

if (mass\_p1[i][j] == 1 && mass\_p2[i][j] != 1) {

mass\_p2[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n1; j++) {

mass\_o2[i][j] = mass\_p2[i][j];

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

}

if (n == n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_o3[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

}

Сначала находим какая матрица является больше, затем проходим по каждому элементу двух матриц и если в одной матрице находим 1, в другой нет, то результатом объединения будет третья матрица

# 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 VISUAL C++ .

Функция obe()код которой представлен ниже выполняет объединение двух матриц.

void obe(int\*\* mass\_o, int\*\* mass\_o2, int\*\* mass\_o3, int\*\* mass\_p1, int\*\* mass\_p2, int n, int n1)

{ //объединение

int i, j;

if (n > n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_o[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

if (n < n1) {

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

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

if (mass\_p1[i][j] == 1 && mass\_p2[i][j] != 1) {

mass\_p2[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n1; j++) {

mass\_o2[i][j] = mass\_p2[i][j];

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

}

if (n == n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_o3[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

}

Результат работы функции представлен на рисунке 1

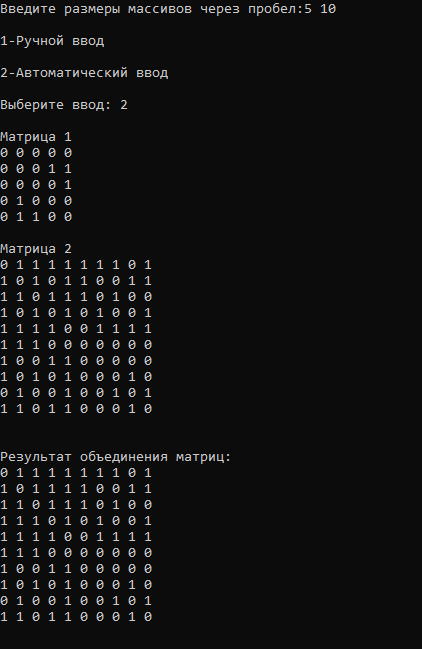


Рисунок 1 Результат объединения матриц.

Функциия oleg() сохраняет результаты объединения в файл.

void oleg(int\*\* mass, int n)

{

FILE\* result;

result = fopen("РЕЗУЛЬТАТ.txt", "w");

fprintf(result, " Результат объединения матриц: \n\n ");

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

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

fprintf(result, "%4d ", mass[i][j]);

}

fprintf(result, "\n\n");

}

fclose(result);

}

Результаты объединения сохранённые в файл.

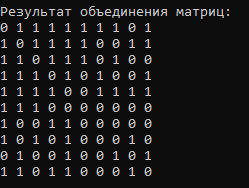
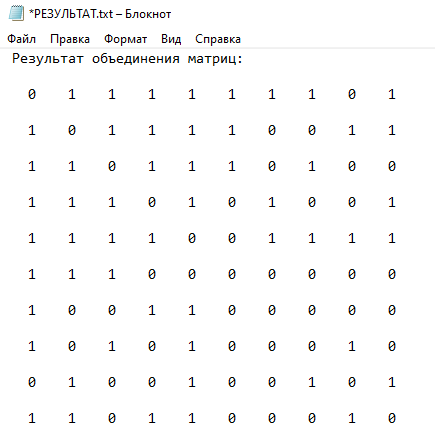
 

Рисунок 2 Результат объединения матриц сохранённый в файл.

# Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Ожидаемый результат | Выполненное действие | Полученный результат |
| Создание с помощью функции автоматического заполнения графа из 5х10 вершин | Вывод графа размера 5х10, который был создан с помощью функции автоматического заполнения | Выбрать пункт 2 – «Автоматическое заполнение» | Верно |
| Создание с помощью функции ручного заполнения графа из 10х5 вершин | Вывод графа размера 2х3, который был создан с помощью функции ручного заполнения | В главном меню выбрать «Начать работу».  .  Выбрать пункт 1 – «Ручное | Верно |
| Сохранение созданного графа в файл | Обнаружение созданного графа в файле | После вывода результата объединения матриц | Верно |
| Выход из программы | Завершение работы программы | Выбрать «Завершить работу» в главном меню. | Верно |

Таблица 2 – План тестирования

Среда разработки Microsoft Visual Studio 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 данных, изменением дизайна, алгоритмом программы, взаимодействием функций, а также с правильностью и неправильностью выводимых данных.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных данных.

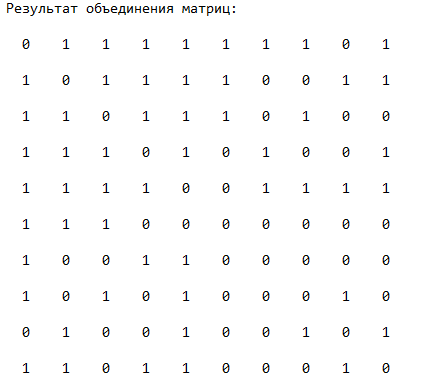
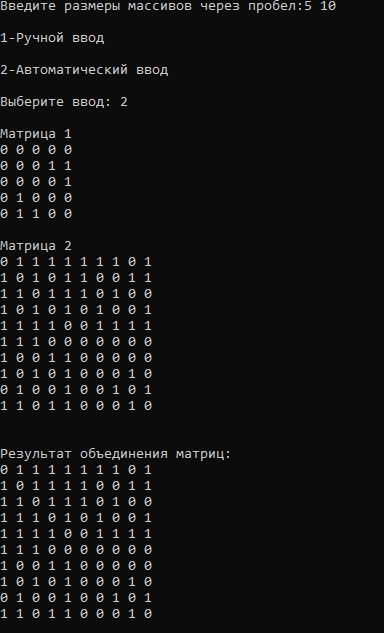


Рисунок 3 Вывод графа размера 5х10, который был создан с помощью функции автоматического заполнения.

Также при вводе неверно введённого символа, просит повторить ввод

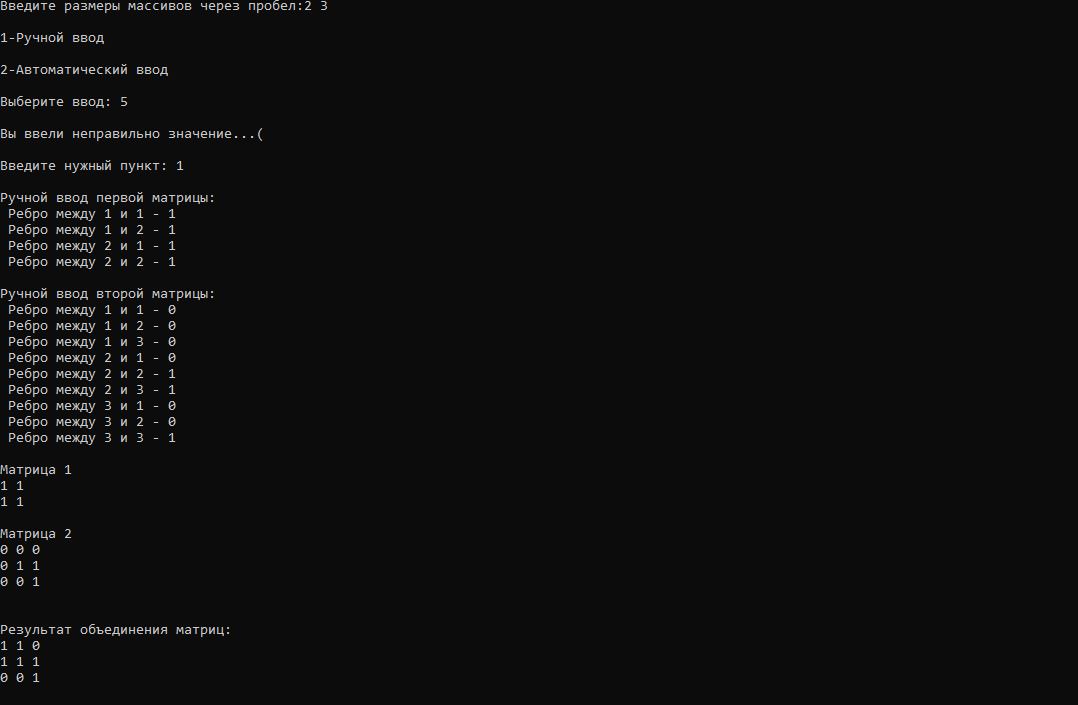


Рисунок 4 Повтор ввода при неверном символе.

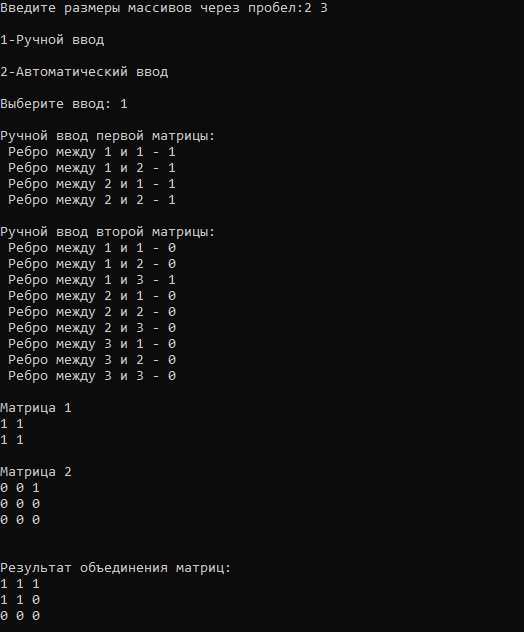


Рисунок 5 Вывод графа размера 2х3, который был создан с помощью функции ручного заполнения.

# Ручной расчёт задачи

Проведем проверку программы посредством ручных вычислений на примере графа с 3 вершинами. Граф создадим автоматически.

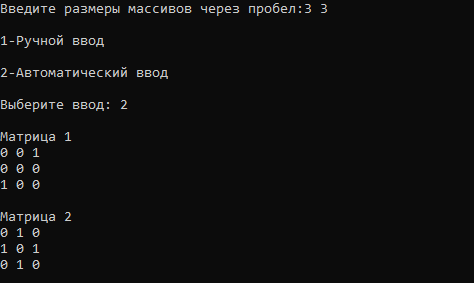


Рисунок 6 – Пример матрицы для ручного расчета

Элементы двух матриц объединятся и запишутся в третью матрицу.

В новой матрице появится связь между ребром 1 и 1, 1 и 2, 2 и 1, 2 и 2, 1 и 3,

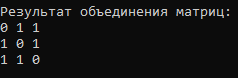


Рисунок 7 – Результат работы

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая операции нахождения двух и более множеств в Microsoft Visual Studio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей, основанных на теории графов. Углублены знания языков программирования Cи и Си++.

# 

# Список использованных источников

1. Лекции
2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978
3. Бьерн Страуструп - "Язык программирования С++"
4. Герберд Шилд "Полный справочник по C++"

## Приложения

## Приложение 1. Листинг программы

## 

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <random>

#include <time.h>

#include <locale.h>

#include <windows.h>

#include <string.h>

using namespace std;

void generate(int\*\* mass, int n) { // заполнение массивов

int i, r, c;

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

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

r = rand() % 2;

c = rand() % 2;

if (r == 1) {

mass[i][j] = 1;

}

else

mass[i][j] = 0;

if (i == j) {

mass[i][j] = 0;

}

}

}

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

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

if (mass[i][j] != mass[j][i]) {

mass[i][j] = mass[j][i];

}

}

}

}

void rr(int\*\* mass, int n)

{

int i;

printf("\nРучной ввод первой матрицы:\n");

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

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

printf(" Ребро между %d и %d - ", i + 1, j + 1);

scanf\_s("%d", &mass[i][j]);

}

}

}

void rr2(int\*\* mass, int n)

{

int i;

printf("\nРучной ввод второй матрицы:\n");

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

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

printf(" Ребро между %d и %d - ", i + 1, j + 1);

scanf\_s("%d", &mass[i][j]);

}

}

}

void obe(int\*\* mass\_olegon, int\*\* mass\_olegon2, int\*\* mass\_olegon3, int\*\* mass\_p1, int\*\* mass\_p2, int n, int n1) { //объединение

int i, j;

if (n > n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_olegon[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

if (n < n1) {

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

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

if (mass\_p1[i][j] == 1 && mass\_p2[i][j] != 1) {

mass\_p2[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n1; j++) {

mass\_olegon2[i][j] = mass\_p2[i][j];

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

}

if (n == n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

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

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

mass\_olegon3[i][j] = mass\_p1[i][j];

printf("%d ", mass\_p1[i][j]);

}

printf("\n");

}

}

}

void oleg(int\*\* mass, int n)

{

FILE\* result;

result = fopen("РЕЗУЛЬТАТ.txt", "w");

fprintf(result, " Результат объединения матриц: \n\n ");

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

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

fprintf(result, "%4d ", mass[i][j]);

}

fprintf(result, "\n\n");

}

fclose(result);

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int n, n1;

printf("Введите размеры массивов через пробел:");

scanf\_s("%d %d", &n, &n1);

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

int\*\* mass\_p1 = (int\*\*)malloc(sizeof(int) \* n);

int\*\* mass\_p2 = (int\*\*)malloc(sizeof(int) \* n1);

int\*\* mass\_olegon = (int\*\*)malloc(sizeof(int) \* n);

int\*\* mass\_olegon2 = (int\*\*)malloc(sizeof(int) \* n1);

int\*\* mass\_olegon3 = (int\*\*)malloc(sizeof(int) \* n1);

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

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

mass\_p1[i] = (int\*)malloc(sizeof(int) \* n);

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

}

for (int i = 0; i < n1; i++) {

mass\_p2[i] = (int\*)malloc(sizeof(int) \* n1);

mass\_olegon2[i] = (int\*)malloc(sizeof(int) \* n1);

mass\_olegon3[i] = (int\*)malloc(sizeof(int) \* n1);

}

int i = 0;

int j = 0;

int gg;

printf("\n1-Ручной ввод\n");

printf("\n2-Автоматический ввод\n");

printf("\nВыберите ввод: ");

scanf\_s("%d", &gg);

while (gg > 2 || gg < 1)

{

printf("\nВы выбрали какое-то другое число...(\n");

printf("\nВведите нужный пункт: ");

scanf\_s("%d", &gg);

}

if (gg == 1)

{

rr(mass, n);

rr2(mass\_p2, n1);

}

if (gg == 2)

{

generate(mass, n);

generate(mass\_p2, n1);

}

printf("\nМатрица 1\n");

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

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

mass\_p1[i][j] = mass[i][j];

}

}

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

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

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

}

printf("\n");

}

printf("\nМатрица 2\n");

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

printf("\n");

obe(mass\_olegon, mass\_olegon2, mass\_olegon3, mass\_p1, mass\_p2, n, n1);

printf("\n");

if (n > n1) {

oleg(mass\_olegon, n);

}

if (n < n1) {

oleg(mass\_olegon2, n1);

}

if (n == n1) {

oleg(mass\_olegon3, n1);

}

return 0;

}