Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Унарные и бинарные операции над графами»

**Выполнили студенты группы 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

### **Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы *M*1*, М*2 смежности неориентированных помеченных графов *G*1, *G*2. Выведите сгенерированные матрицы на экран.
2. \* Для указанных графов преобразуйте представление матриц смежности в списки смежности. Выведите полученные списки на экран.

**Задание 2:**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

**Задание 3**

1. Для матричной формы представления графов выполните операцию:

а) объединения *G* = *G*1  *G*2

б) пересечения *G* = *G*1  *G*2

в) кольцевой суммы *G* = *G*1  *G*2

Результат выполнения операции выведите на экран.

**Задание 4 \***

1. Для матричной формы представления графов выполните операцию декартова произведения графов *G = G*1X *G*2.

Результат выполнения операции выведите на экран.

**Листинг:**

**Задание 1.1;1.2:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 6;

const int m = 6;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

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

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

adj\_matrixONE[i][j] = rand()%2;

adj\_matrixTWO[i][j] = rand()%2;

}

}

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if(i == j){

adj\_matrixONE[i][j] = 0;

adj\_matrixTWO[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

if(adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i]){

adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

printf("%d: ",i);

for (int j = 1; j < n; j++) {

if (adj\_matrixONE[i][j] !=0)

printf(" %d ",j);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

printf("%d: ",i);

for (int j = 1; j < n; j++) {

if (adj\_matrixTWO[i][j] !=0)

printf(" %d ",j);

}

printf("\n");

}

return 0;

}

**Задание 2.1:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 5;

const int m = 5;

int x;

int y;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int z[7];

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

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

adj\_matrixONE[i][j] = rand()%2;

}

}

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

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

if(i == j){

adj\_matrixONE[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("Add 2 verchiny: ");

scanf("%d",&x);

scanf("%d",&y);

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

if (i!=x){

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

if (j!=y){

adj\_matrixTWO[i][j]=adj\_matrixONE[i][j];

}

}

}

}

for (int j = 0; j < n-1; j++){

if((adj\_matrixTWO[j][x]!=1)&&(adj\_matrixTWO[y][j]!=1))

z[j]=0;

else

z[j]=1;

}

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

if((x!=i)&&(y!=i)){

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

if((x!=j)&&(y!=j))

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

}

for(int i = 0; i < n-1; i++) {

if (i==3)

z[i]=0;

printf(" %d ", z[i]);

}

printf("\n\n");

for(int j = 0; j < n-1; j++) {

if (j==3)

z[j]=0;

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

}

return 0;

}

**Задание 2.2:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 5;

const int m = 5;

int x;

int y;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int z[n];

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

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

adj\_matrixONE[i][j] = rand()%2;

}

}

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

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

if(i == j){

adj\_matrixONE[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("Add 2 verchiny: ");

scanf("%d",&x);

scanf("%d",&y);

if (adj\_matrixONE[x][y] == 1){

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

if (i!=x){

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

if (j!=y){

adj\_matrixTWO[i][j]=adj\_matrixONE[i][j];

}

}

}

}

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

if((adj\_matrixTWO[j][x]!=1)&&(adj\_matrixTWO[y][j]!=1))

z[j]=0;

else

z[j]=1;

}

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

if((x!=i)&&(y!=i)){

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

if((x!=j)&&(y!=j))

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

}

for(int i = 0; i < n-1; i++) {

if (i==3)

z[i]=0;

printf(" %d ", z[i]);

}

printf("\n\n");

for(int j = 0; j < n-1; j++) {

if (j==3)

z[j]=0;

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

}

}

else

printf("Verchiny ne svyazanbI");

return 0;

}

**Задание 2.3:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 6;

const int m = 6;

int x;

int y;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int z[n];

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

adj\_matrixONE[i][j] = rand()%2;

}

}

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if(i == j){

adj\_matrixONE[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("Add 1 verchiny: ");

scanf("%d",&x);

for(int i = 1; i < n; i++) {

if(x==i){

for (int j = 1; j < n+1; j++) {

if(j==6){

adj\_matrixONE[x][j]=0;

}

printf("%d ", adj\_matrixONE[x][j]);

}

}

}

printf("\n\n");

for(int i = 1; i < n; i++) {

if(x==i){

for (int j = 1; j < n+1; j++) {

if(j==6){

adj\_matrixONE[x][j]=0;

}

printf(" %d \n", adj\_matrixONE[x][j]);

}

}

}

return 0;

}

**Задание 3.1:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 6;

const int m = 6;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int adj\_matrixRES[n][m];

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

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

adj\_matrixONE[i][j] = rand()%2;

adj\_matrixTWO[i][j] = rand()%2;

}

}

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if(i == j){

adj\_matrixONE[i][j] = 0;

adj\_matrixTWO[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

if(adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i]){

adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

printf("\n");

printf("Result: \n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if((adj\_matrixONE[i][j]==1)||(adj\_matrixTWO[i][j]==1))

adj\_matrixRES[i][j]=1;

else

adj\_matrixRES[i][j]=0;

printf(" %d ", adj\_matrixRES[i][j]);

}

printf("\n");

}

return 0;

}

**Задание 3.2:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 6;

const int m = 6;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int adj\_matrixRES[n][m];

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

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

adj\_matrixONE[i][j] = rand()%2;

adj\_matrixTWO[i][j] = rand()%2;

}

}

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if(i == j){

adj\_matrixONE[i][j] = 0;

adj\_matrixTWO[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

if(adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i]){

adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

printf("\n");

printf("Result: \n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if((adj\_matrixONE[i][j]==1)&&(adj\_matrixTWO[i][j]==1))

adj\_matrixRES[i][j]=1;

else

adj\_matrixRES[i][j]=0;

printf(" %d ", adj\_matrixRES[i][j]);

}

printf("\n");

}

return 0;

}

**Задание 3.3:**

#include "StdAfx.h"

#include <iostream>

#include <stdlib.h>

#include "Time.h"

using namespace std;

const int n = 6;

const int m = 6;

int main()

{

srand(time(NULL));

int adj\_matrixONE[n][m];

int adj\_matrixTWO[n][m];

int adj\_matrixRES[n][m];

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

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

adj\_matrixONE[i][j] = rand()%2;

adj\_matrixTWO[i][j] = rand()%2;

}

}

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if(i == j){

adj\_matrixONE[i][j] = 0;

adj\_matrixTWO[i][j] = 0;

}

if(adj\_matrixONE[i][j] = adj\_matrixONE[j][i]){

adj\_matrixONE[i][j] = adj\_matrixONE[j][i];

}

if(adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i]){

adj\_matrixTWO[i][j] = adj\_matrixTWO[j][i];

}

printf(" %d ", adj\_matrixONE[i][j]);

}

printf("\n");

}

printf("\n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

printf(" %d ", adj\_matrixTWO[i][j]);

}

printf("\n");

}

printf("\n");

printf("Result: \n");

for(int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

if((adj\_matrixONE[i][j]==1)^(adj\_matrixTWO[i][j]==1))

adj\_matrixRES[i][j]=1;

else

adj\_matrixRES[i][j]=0;

printf(" %d ", adj\_matrixRES[i][j]);

}

printf("\n");

}

return 0;

}

**Ход работы:**

**Задание 1.1:**

Создали граф путём использования функции *rand()%2* с генерацией только “0” или “1”. После этого сравнили значения зеркальных адресов графа и, если они различны – сделали их одинаковыми. Главную диагональ графа обнулили. Таким образом создали 2 графа.

**Задание 1.2:**

Из ранее созданных графов путём циклического перебора выявили все связанные вершины и вывели их в консоль.

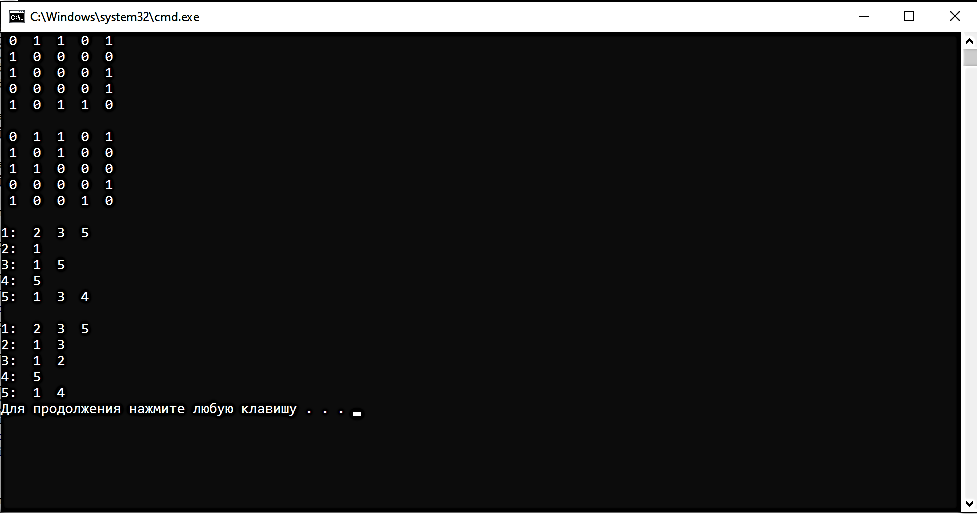
****

Рисунок 1.1-результат работы программы.

**Задание 2.1:**

Генерируется граф, пользователь выбирает вершины, которые необходимо связать. Далее эти вершины удаляются, а из графа удаляются соответствующие столбцы и строки со связями. Пользователю выводятся оставшаяся матрица смежности. Из удалённых строк и столбцов выбираются связи, которые были у удалённых вершин с остальными. Из этих связей формируется новый массив, который подставляется в первый.

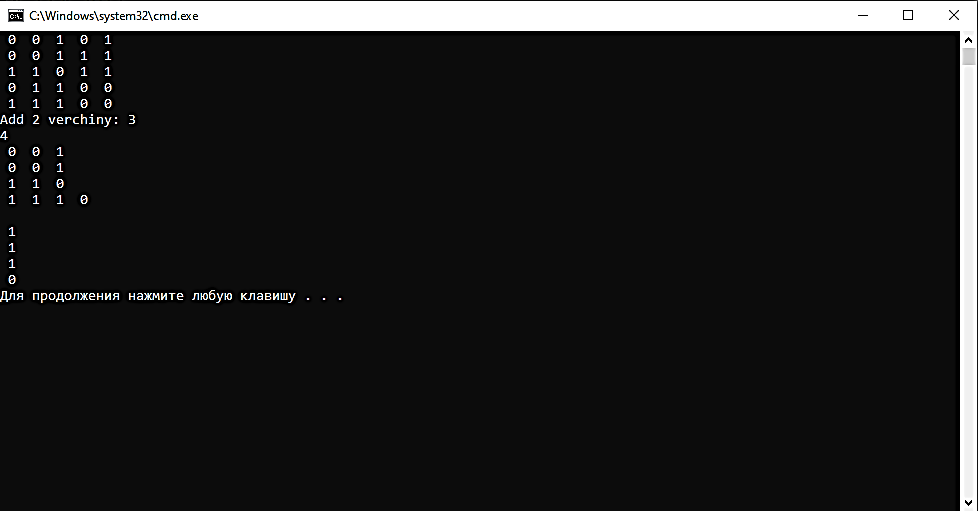


Рисунок 2.1-результат работы программы.

**Задание 2.2:**

Генерируется граф, пользователь выбирает вершины, которые необходимо связать. Затем проверяется, есть ли связь между выбранными вершинами, если есть, то вершины удаляются, а из графа удаляются соответствующие столбцы и строки со связями. Пользователю выводятся оставшаяся матрица смежности. Из удалённых строк и столбцов выбираются связи, которые были у удалённых вершин с остальными. Из этих связей формируется новый массив, который подставляется в первый. Если связи нет, то пользователю выдаётся соответствующее предупреждение.

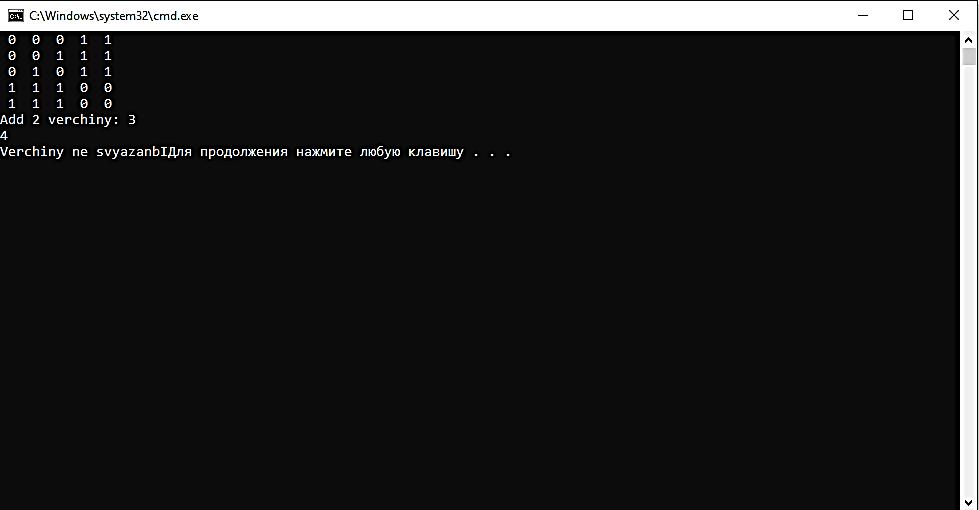
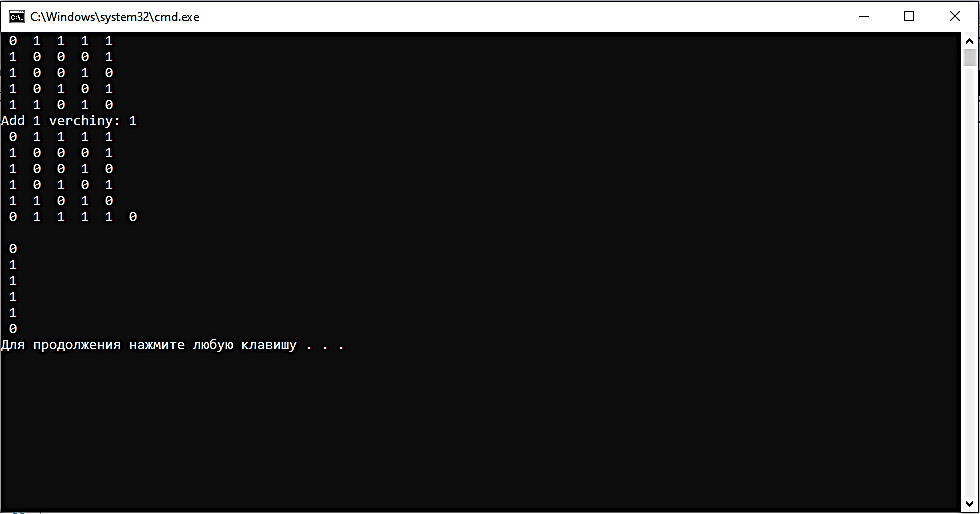
****

Рисунок 2.2-результат работы программы.

**Задание 2.3:**

Генерируется граф, пользователь выбирает вершину, которую необходимо расщепить. Далее столбик вершины копируется из матрицы смежности, после он выводится в новой матрице со всеми связями.

****

**Задание 3.1:**

Генерируется две матрицы, после каждый элемент матрицы логически складывается с соответствующим элементом другой матрицы.

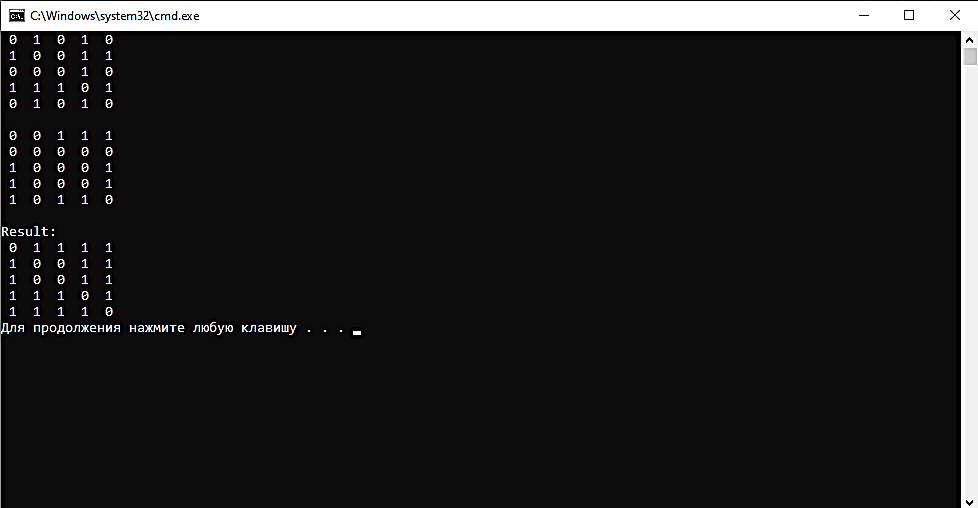


Рисунок 3.1-результат работы программы.

**Задание 3.2:**

Генерируется две матрицы, после каждый элемент матрицы логически умножается с соответствующим элементом другой матрицы.

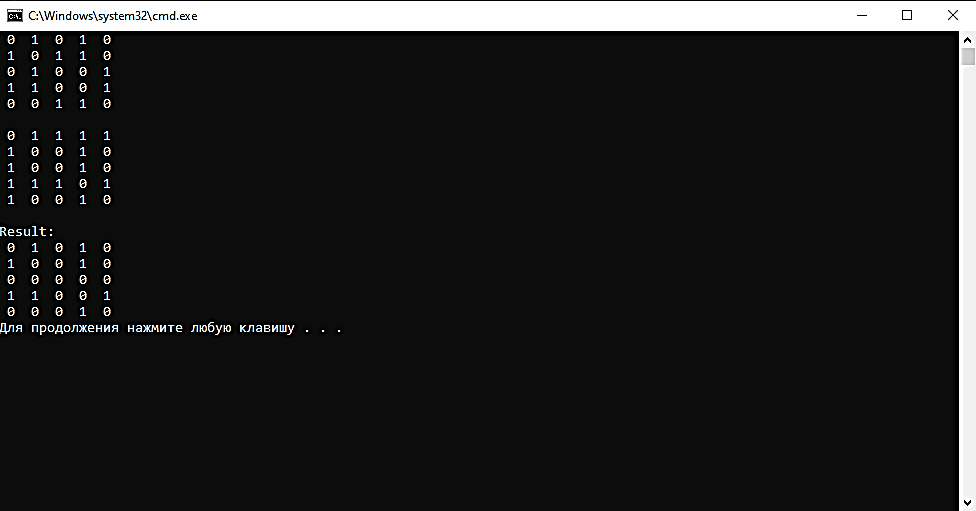
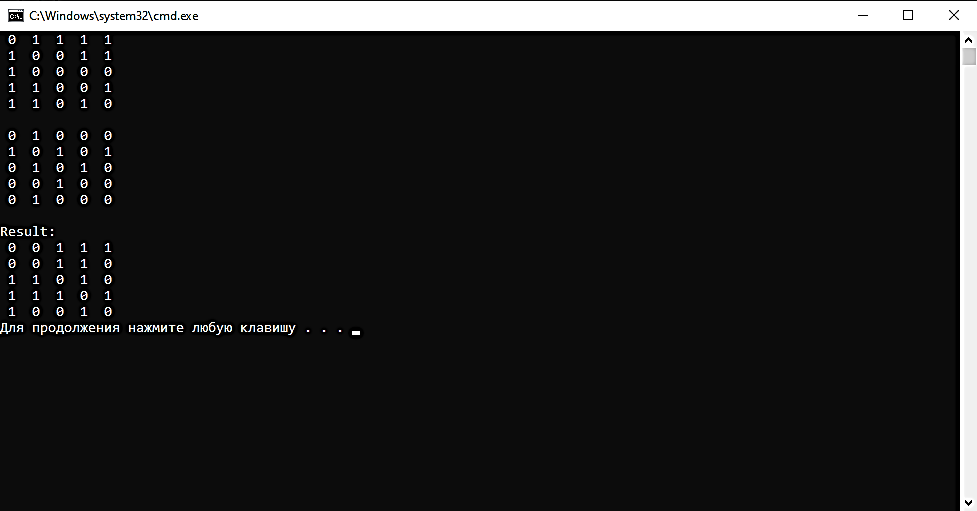
****

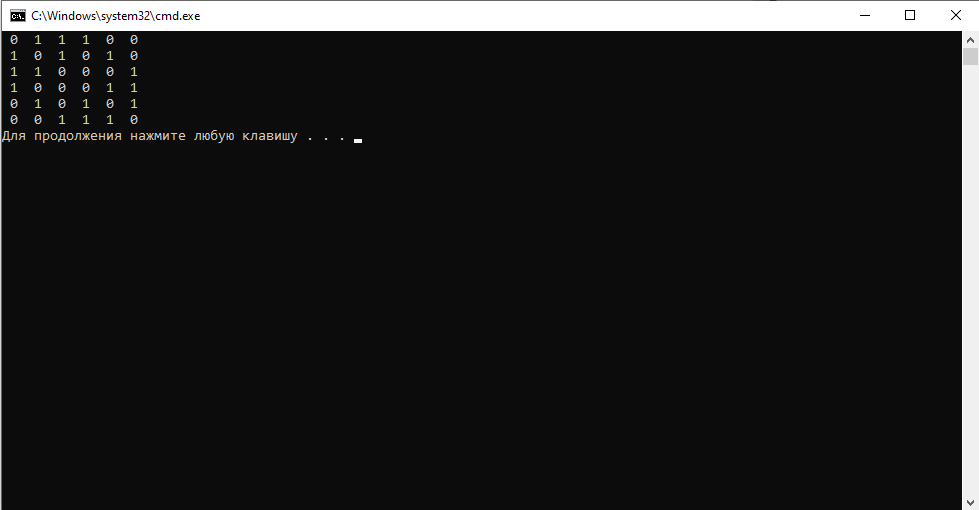
Рисунок 3.2-результат работы программы.

**Задание 3.3:**

Генерируется две матрицы, после каждый элемент матрицы разделительно дизъюнктирутеся с соответствующим элементом другой матрицы.

****

**Задание 4:**

****

**Вывод:** в ходе выполнения лабораторной работы были разработаны программы, выполняющие работу с графами.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с использованием графов на языке Си.