Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №4
по дисциплине «Информатика: Основы программирования»
на тему «Массивы. Динамическое выделение памяти»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель:
Першин Д.В.

Санкт-Петербург
2020 г.

Задача 1. Записать элементы массива C (20) в обратном порядке {C20;C19;C18;…;C2;C1}. Вспомогательный массив не использовать.

Исходные данные:

const N = 20 – размер массива C, тип int, C[N] – массив элементов размером N, тип int.

Результирующие данные:

Элементы массива C, тип int.

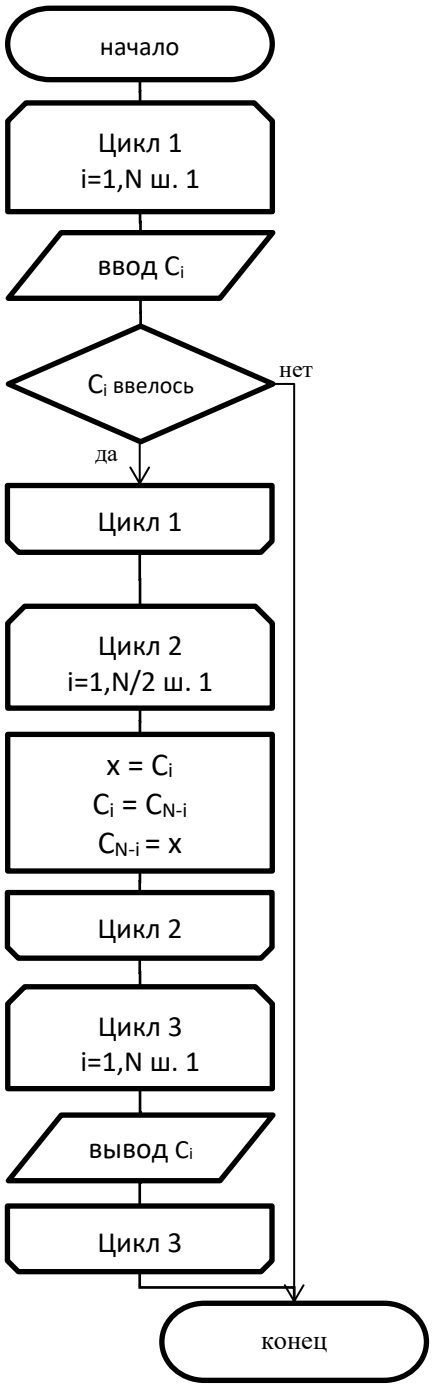
Вспомогательные переменные:

i – индекс массива C, тип int, x – переменная для замены значений элементов массива C, тип int.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 Process returned 0 (0x0) execution time : 19.226 s

Схема программы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define N 20
int main()
{
    /*объявление переменных*/
    int C[N], x, i;
    /*ввод элементов массива С с помощью цикла,
    проверяя ввелось ли число, scanf() возвращает
    кол-во прочитанных элементов*/
    for (i=0; i<N; i++)
        if (scanf("%d", &C[i])!= 1)
            return -1;
    /*запись элементов массива С в обратном порядке
    с помощью цикла, цикл до N/2, так как переставить
```

```
    местами нужно N/2 пар элементов массива*/
    for (i=0; i<N/2; i++)
    {
        /*записываем значение C[i] в x*/
        x = C[i];
        /*присваиваем C[i] значение, с которым его надо поменять*/
        C[i] = C[N-1-i];
        /*присваиваем паре C[i] значение x, то есть C[i] */
        C[N-1-i] = x;
    };
    /*выводим измененный массив C с помощью цикла*/
    for (i=0; i<N; i++)
        printf("%d ",C[i]);
    return 0;
}
```

Задача 2. Определить количество элементов линейного массива, больших среднего арифметического значения элементов этого массива.

Исходные данные:

a – указатель, тип int*, k – кол-во элементов массива a.

Результирующие данные:

x – ср. арифм. элементов массива a, тип double, c – кол-во элементов массива a больших ср. арифм., тип int,

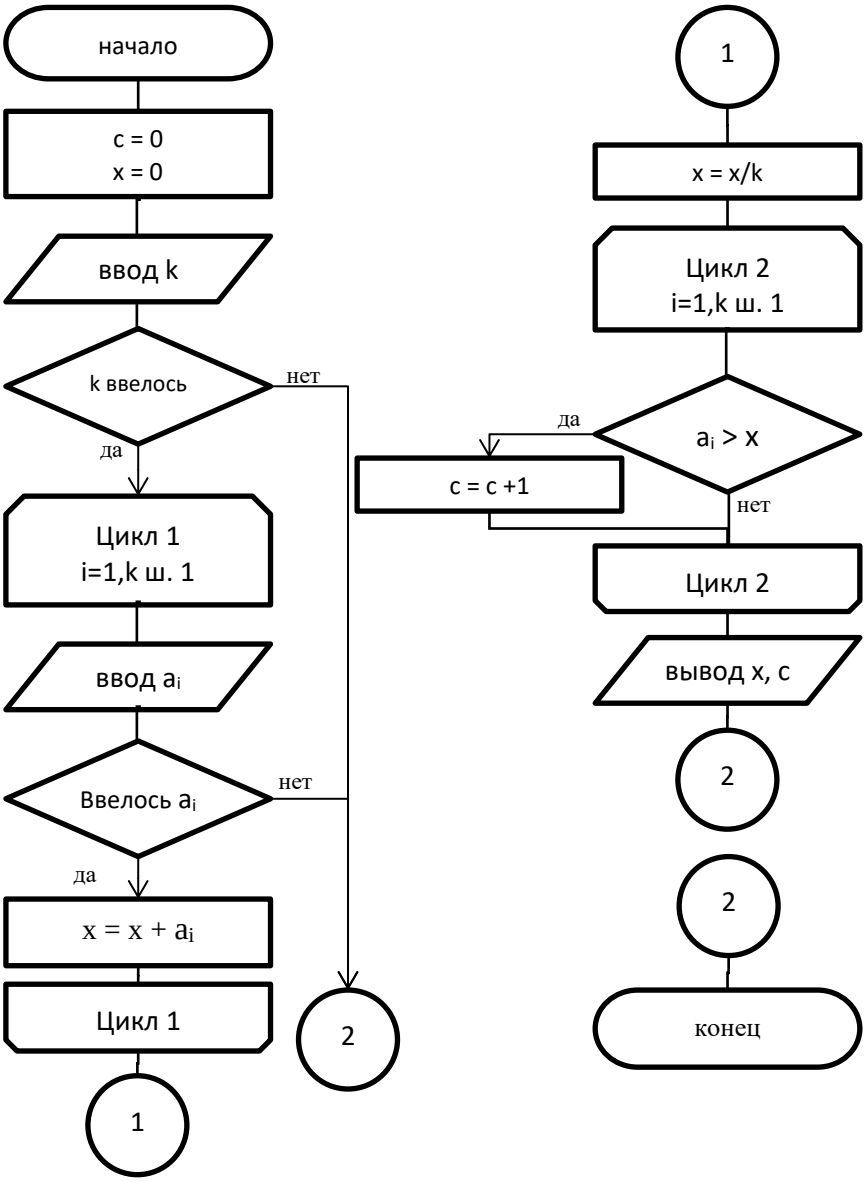
Вспомогательные переменные:

i – индекс массива a, тип int.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
5 1 2 3 4 5	2	5 1 2 3 4 5 3.000000 2 Process returned 0 (0x0) execution time : 37.984 s
5 0 0 0 0 0	0	5 0 0 0 0 0 0.000000 0 Process returned 0 (0x0) execution time : 7.870 s
5 -5 10 14 2 31	2	5 -5 10 14 2 31 10.400000 2 Process returned 0 (0x0) execution time : 25.452 s
5 -1 -2 -3 -4 -5	2	5 -1 -2 -3 -4 -5 -3.000000 2 Process returned 0 (0x0) execution time : 26.063 s

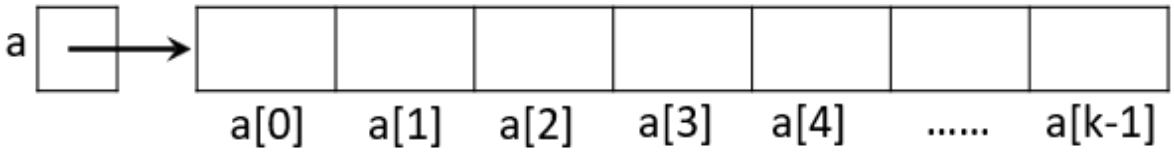
Схема программы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    /*объявление переменных*/
    int *a, i,c=0,k;
    double x=0.;
    /*ввод k, кол-ва элементов массива a,
    проверяя ввелось ли число, scanf() возвращает
    кол-во прочитанных элементов*/
    if (scanf("%d", &k)!= 1)
        return -1;
    /*выделение памяти под k элементов тип int*/
    a = malloc(k*sizeof(int));
    /*проверка выделилась ли память*/
    if (a==NULL)
        return -1;
    /*ввод элементов массива a с помощью цикла,
    проверяя ввелось ли число, scanf() возвращает
    кол-во прочитанных элементов*/
```

```
    for (i=0; i<k; i++)
    {
        if (scanf("%d", &a[i])!= 1)
            return -1;
        /*подсчет суммы всех эл-ов массива a*/
        x+=a[i];
    };
    /*вычисление ср. арифм. элементов массива a
    и запись в x*/
    x/=k;
    /*подсчет кол-ва эл-тов массива a больших ср. арифм.*/
    for (i=0; i<k; i++)
        if (a[i]>x)
            c++;
    /*вывод ср. арифм. и кол-во элементов больших его*/
    printf("%lf %d",x,c);
    /*освобождение памяти, выделенной под массив a*/
    free(a);
    return 0;
}
```



Задача 3. Заполнить матрицу A (8x8) следующим образом: на главной диагонали – «0», над диагональю – «1», под диагональю – «-1»

Исходные данные:

const N = 8, тип int, a[N][N] - двумерный массив, матрица размером 8*8, тип int.

Результирующие данные:

элементы массива a, тип int.

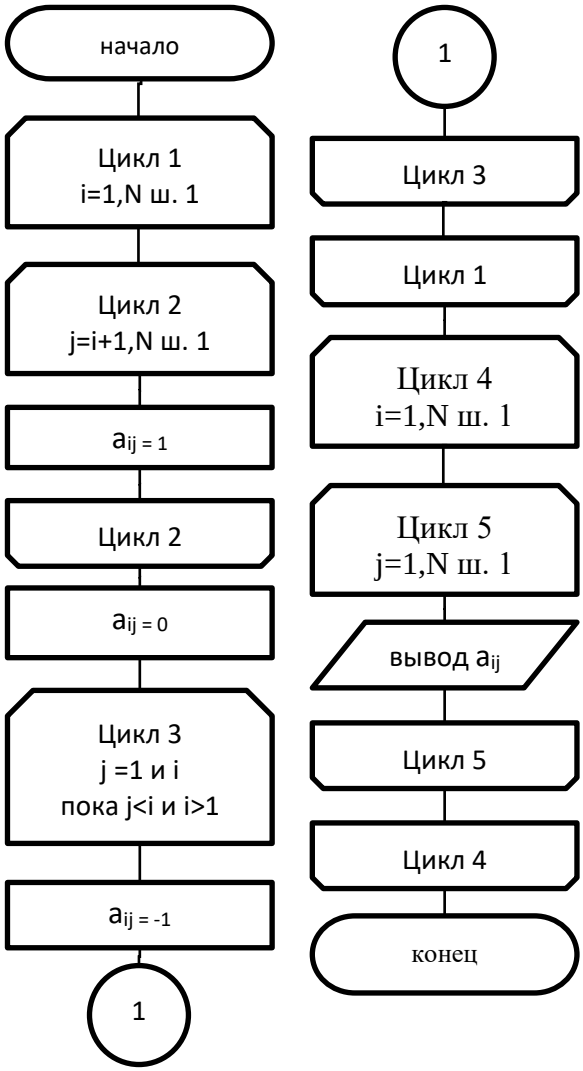
Вспомогательные переменные:

i – индекс массива a, отвечающий за строки, тип int, j – индекс массива a, отвечающий за столбцы, тип int.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
N=8	0 1 1 1 1 1 1 1 -1 0 1 1 1 1 1 1 -1 -1 0 1 1 1 1 1 -1 -1 -1 0 1 1 1 1 -1 -1 -1 -1 0 1 1 1 -1 -1 -1 -1 -1 0 1 1 -1 -1 -1 -1 -1 -1 0 1 -1 -1 -1 -1 -1 -1 -1 0	0 1 1 1 1 1 1 1 -1 0 1 1 1 1 1 1 -1 -1 0 1 1 1 1 1 -1 -1 -1 0 1 1 1 1 -1 -1 -1 -1 0 1 1 1 -1 -1 -1 -1 -1 0 1 1 -1 -1 -1 -1 -1 -1 0 1 -1 -1 -1 -1 -1 -1 -1 0 Process returned 0 (0x0) execution time : 0.016 s

Схема программы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define N 8
int main()
{
    /*объявление переменных*/
    int a[N][N],i,j;
    /*проход по матрице циклами, i - строка, j - столбец*/
    for (i=0;i<N;i++)
    {
        /*выставление 1 над диагональю, то есть в текущей строке
        происходит проход по столбцам, начиная с номера столбца i+1 до N-1*/
        for (j=i+1;j<N;j++)
            a[i][j]=1;
        /*выставление нулей на диагонали*/
        a[i][i]=0;
        /*выставление -1 под диагональю, то есть в текущей строке
        происходит проход по столбцам, начиная с номера столбца 0 до i-1,
        и этот цикл будет выполняться если строка не первая*/
        for (j=0;j<i&&i>0;j++)
            a[i][j]=-1;
    };
    /*вывод матрицы*/
    for (i=0; i<N;i++)
```

```
    {
        for (j=0;j<N;j++)
            printf("%2d ",a[i][j]);
        printf("\n");
    };
    return 0;
}
```

Задача 4. Поменять местами минимальный элемент целочисленной матрицы P (9x11) и элемент, значение которого совпадает с заданным X. Если указанный элемент в матрице отсутствует, вывести сообщение об этом. Матрица, в которой минимальное значение или X встречается неоднократно, является некорректной.

Исходные данные:

const N=9,M=11 – кол-во строк и столбцов матрицы, тип int, a[N][M]- двумерный массив, размер N*M, тип int, x – значение эл-та, который нужно заменить, тип int.

Результирующие данные:

сообщение, если x не найден в матрице, массив a, тип int.

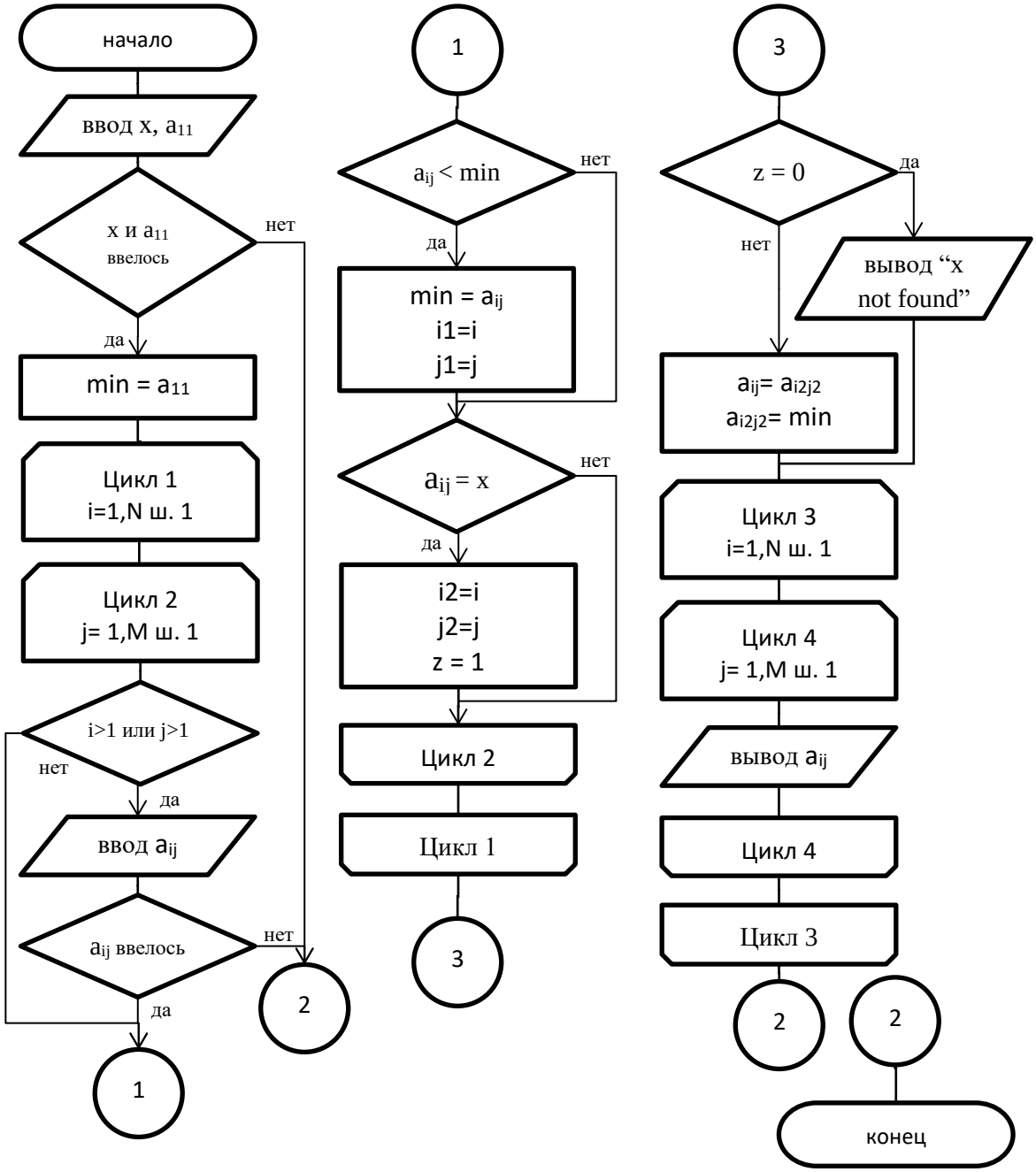
Вспомогательные переменные:

i - индекс массива a, отвечающий за строки, тип int, j – индекс массива a, отвечающий за столбцы, тип int, i1 и j1 – индексы минимального элемента массива a, тип int, min – значение минимального эл-та массива a, тип int, i2 и j2 индексы эл-та равного x, тип int, z - переменная сигнализирующая, что x был найден в массиве, если равна 1, тип int.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
При N=3 M=4 5 1 2 3 5 -1 -2 -3 -4 22 33 44 55	1 2 3 -4 -1 -2 -3 5 22 33 44 55	5 1 2 3 5 -1 -2 -3 -4 22 33 44 55 1 2 3 -4 -1 -2 -3 5 22 33 44 55 Process returned 0 (0x0) execution time : 22.782 s
При N=3 M=4 100 1 2 3 5 -1 -2 -3 -4 22 33 44 55	1 2 3 5 -1 -2 -3 -4 22 33 44 55	100 1 2 3 5 -1 -2 -3 -4 22 33 44 55 x not found 1 2 3 5 -1 -2 -3 -4 22 33 44 55 Process returned 0 (0x0) execution time : 1.480 s

Схема программы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define N 9
```

```
#define M 11
int main()
{
    /*объявление переменных*/
    int a[N][M],i,j,i1,j1,x,c,min,z=0,i2,j2;
    /*ввод x и первого эл-та массива a, проверяя ввод*/
    if (scanf("%d",&x)!=1 || scanf("%d",&a[0][0])!=1)
        return -1;
    /*присвоение min a[0][0] для поиска минимума массива a*/
    min = a[0][0];
    /*проход по массиву*/
    for (i=0;i<N;i++)
        for (j=0;j<M;j++)
        {
            /*проверка на первый элемент, его вводить уже не надо*/
            if (i>0 || j>0)
                /*ввод эл-тов с проверкой введения*/
                if (scanf("%d",&a[i][j])!=1)
                    return -1;
            /*поиск минимума*/
            if (a[i][j]<min)
            {
                min = a[i][j];
                /*запись индексов минимального эл-та*/
                i1=i;
                j1=j;
            };
            /*поиск x в массиве*/
            if (a[i][j]==x)
            {
                /*запись индексов эл-та равного x*/
                i2=i;
                j2=j;
                /*переменная сигнализирующая,
                что x был найден в массиве, если = 1*/
                z=1;
            };
        };
    /*провека, был ли найден x, если нет,
    вывод сообщения об этом, если найден,
    перестановка местами минимального и эл-та со значением x*/
    if (z==0)
        printf("x not found");
    else
    {
        a[i1][j1] = a[i2][j2];
        a[i2][j2] = min;
    };
    printf("\n");
    /*вывод матрицы*/
    for (i=0; i<N;i++)
    {
        for (j=0;j<M;j++)
            printf("%3d ",a[i][j]);
        printf("\n");
    };
    return 0;
}
```

Задача 5. Куб состоит из N^3 прозрачных и непрозрачных элементарных кубиков. Построить полностью непрозрачный куб, используя ровно N^2 непрозрачных элементарных кубиков.

Исходные данные:

const N – сторона куба, тип int, a[N][N][N] – массив элементарных кубиков, тип int.

Результирующие данные:

массив a, содержащий информацию, какой элементарный кубик прозрачный, а какой нет, тип int.

Вспомогательные переменные:

i,j,c – индексы массива a, тип int., i – уровень куба, j – строка уровня, c – столбец уровня.

Таблица тестирования:

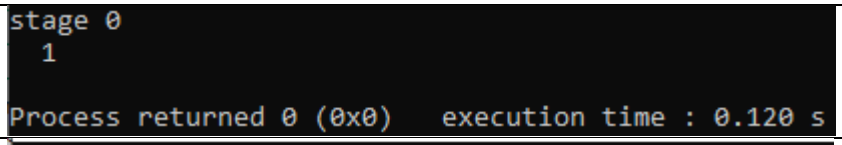
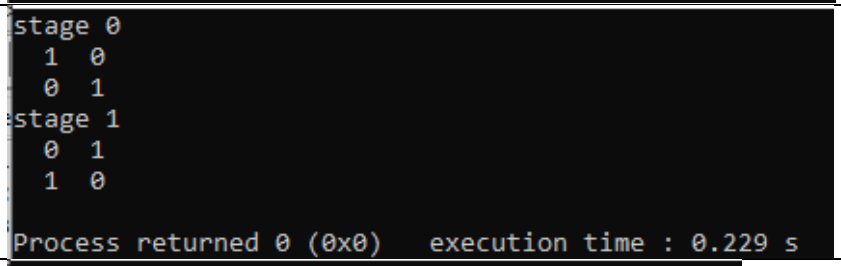
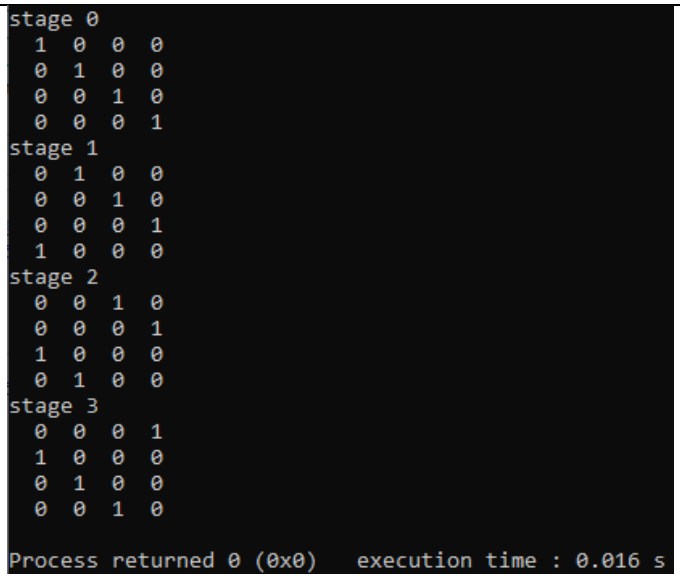
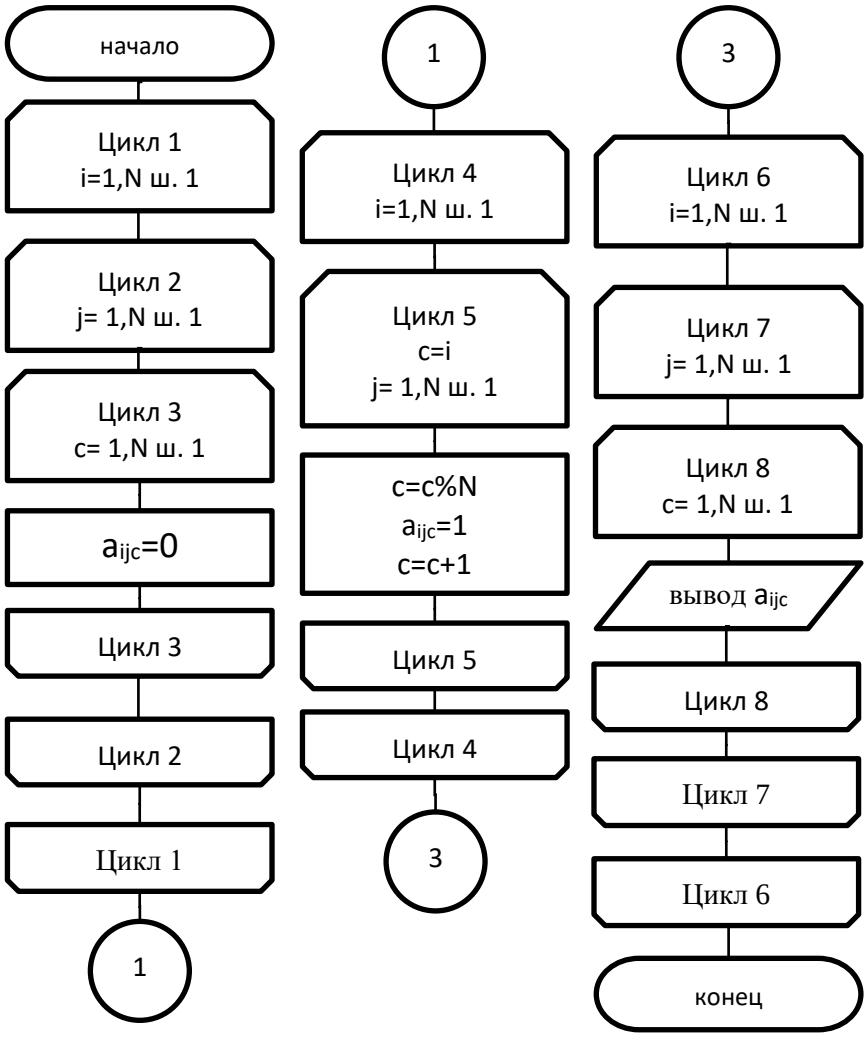
Входные данные	**Ожидаемый результат**	**Результат работы программы**
При N=1	1	
При N=2	10 01 01 10	
При N=3	100 010 001 010 001 100 001 100 010	
При N=4	1000 0100 0010 0001 0100 0010 0001 1000 0010 0001 1000 0100 0001 1000 0100 0010	

Схема программы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define N 4
int main()
{
    /*объявление переменных*/
    int a[N][N][N], i, j, c;
    /*заполняем массив а нулями, то есть прозрачными кубиками,
    чтобы потом просто заполнить нужным кол-вом непрозрачных*/
    for (i=0;i<N;i++)
        for (j=0;j<N;j++)
            for (c=0;c<N;c++)
                a[i][j][c] = 0;
    /*проход по всем уровням куба*/
    for (i=0;i<N;i++)
        /*заполнение единицами происходит построчно,
        если смотреть на весь уровень целиком,
        то по диагонали каждый раз, смещая начало
        диагонали на один вправо, насколько я понял,
```

```
        чтобы куб был непрозрачен, нужно,
        чтобы в каждой строке и каждом столбце
        каждого уровня была одна единица*/
        for (j=0,c=i;j<N;j++,c++)
        {
            /*присваивая с остаток деления с на N,
            мы не допускаем выхода за пределы массива,
            тем самым при смещении вправо, когда с
            станет равно N, мы переместимся на начало строки. */
            c%=N;
            a[i][j][c] = 1;
        };
    /*вывод куба по уровням в матричном виде*/
    for (i=0;i<N;i++)
    {
        printf("stage %d\n",i);
        for (j=0;j<N;j++)
        {
            for (c=0;c<N;c++)
                printf("%3d", a[i][j][c]);
            printf("\n");
        }
    }
    return 0;
}
```

Задача 6. Заполнить матрицу A (8x8) следующим образом: на главной диагонали – «0», над диагональю – «1», под диагональю – «-1». Память выделять динамически.

Исходные данные:

a – указатель на указатель, впоследствии двумерный массив, тип int**

Результирующие данные:

элементы массива a, тип int.

Вспомогательные переменные:

i – индекс массива a, отвечающий за строки, тип int, j – индекс массива a, отвечающий за столбцы, тип int.

Таблица тестирования:

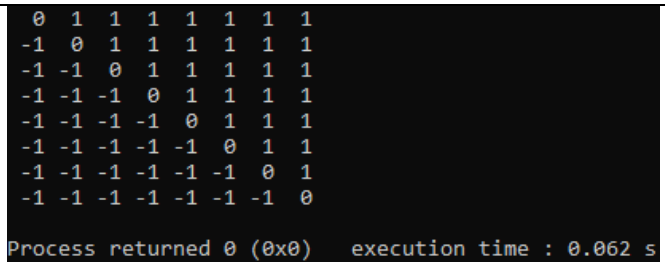
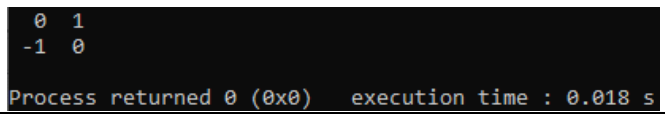
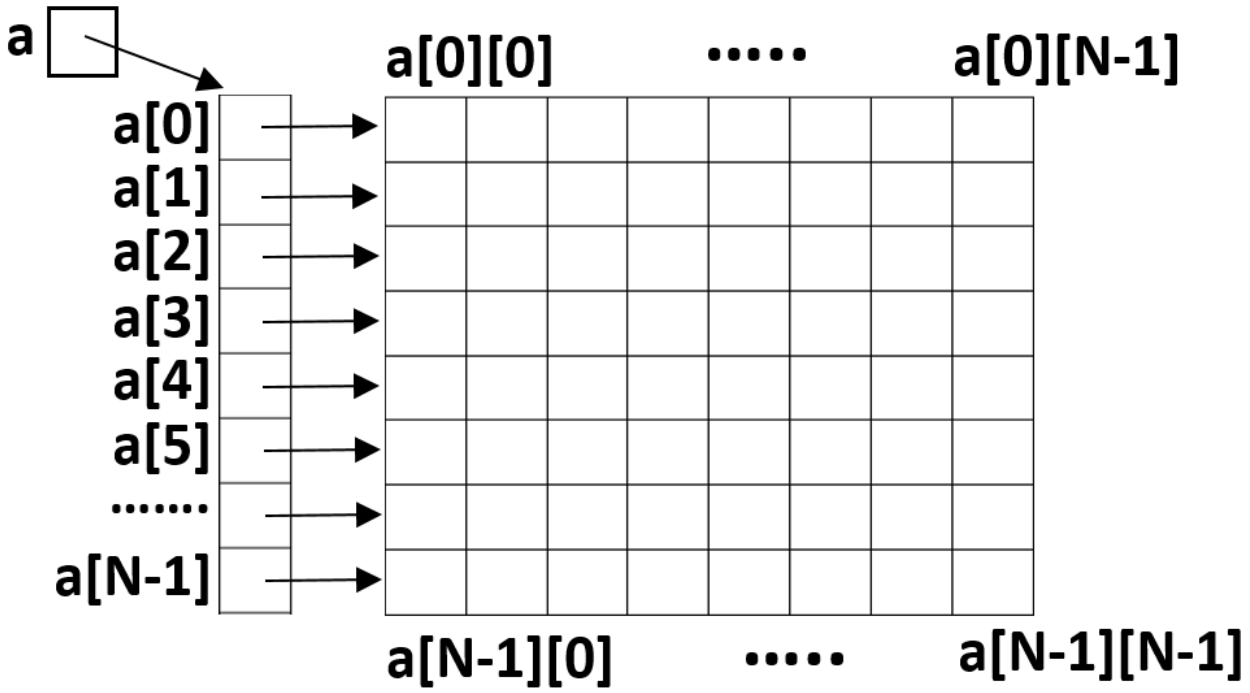
Входные данные	**Ожидаемый результат**	**Результат работы программы**
N = 8	0 1 1 1 1 1 1 1 -1 0 1 1 1 1 1 1 -1 -1 0 1 1 1 1 1 -1 -1 -1 0 1 1 1 1 -1 -1 -1 -1 0 1 1 1 -1 -1 -1 -1 -1 0 1 1 -1 -1 -1 -1 -1 -1 0 1 -1 -1 -1 -1 -1 -1 -1 0	
N = 2	0 1 -1 0	

Схема организации матрицы



Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define N 8
int main()
{
    /*объявление переменных*/
    int **a,i,j;
    /*выделение памяти под указатели на строки*/
    a=malloc(N*sizeof(int*));
    /*проверка, выделилась ли память*/
    if (a == NULL)
        return -1;
    /*выделение памяти для каждого указателя
    на строку под столбцы, и проверка, выделилась ли память*/
    for(i=0;i<N;i++)
    {
        a[i] = malloc(N*sizeof(int));
        if (a[i] == NULL)
            return -1;
    };
    /*проход по матрице циклами, i - строка, j - столбец*/
    for(i=0;i<N;i++)
    {
        /*выставление 1 над диагональю, то есть в текущей строке
        происходит проход по столбцам, начиная с номера столбца i+1 до N-1*/
        for (j=i+1;j<N;j++)
            a[i][j]=1;
        /*выставление нулей на диагонали*/
        a[i][i]=0;
        /*выставление -1 под диагональю, то есть в текущей строке
        происходит проход по столбцам, начиная с номера столбца 0 до i-1,
        и этот цикл будет выполняться если строка не первая*/
        for (j=0;j<i&&i>0;j++)
            a[i][j]=-1;
    };
    /*вывод матрицы*/
```

```
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%3d",a[i][j]);
        printf("\n");
    };
    /*освобождение памяти, выделенной под каждую строку*/
    for(i=0;i<N;i++)
        free(a[i]);
    /*освобождение памяти, выделенной под указатели на строки*/
    free(a);
    return 0;
}
```