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

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №5
по дисциплине «Информатика: Основы программирования»
на тему «Функции»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель:
Першин Д.В.

Санкт-Петербург
2020 г.

Задача 1. Отсортировать массив M (50) и каждую строку матрицы A (6x7) в порядке убывания

Исходные данные:

L - const, тип *int*, размер *M*.

N, C – const, тип *int*, размеры *b*, *N* - кол-во строк, *C*– столбцов.

M[L] – статический массив, тип *int*, размером L,

b – указатель на *int*, матрица, тип *int***, выделение памяти способом для динамической матрицы

Результирующие данные:

печать массива M и матрицы b, тип *int*

Таблица тестирования:

Случайные числа

L=20 N=6 C=7

```
41  67  34   0  69  24  78  58  62  64   5  45  81  27  61  91  95  42  27  36
91   4   2  53  92  82  21
16  18  95  47  26  71  38
69  12  67  99  35  94   3
11  22  33  73  64  41  11
53  68  47  44  62  57  37
59  23  41  29  78  16  35

95  91  81  78  69  67  64  62  61  58  45  42  41  36  34  27  27  24   5   0
92  91  82  53  21   4   2
95  71  47  38  26  18  16
99  94  69  67  35  12   3
73  64  41  33  22  11  11
68  62  57  53  47  44  37
78  59  41  35  29  23  16
```

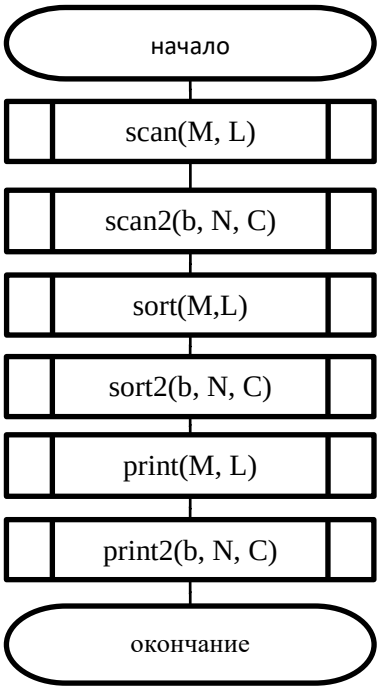
L=12 N=4 C=6

```
41  67  34   0  69  24  78  58  62  64   5  45
81  27  61  91  95  42
27  36  91   4   2  53
92  82  21  16  18  95
47  26  71  38  69  12

78  69  67  64  62  58  45  41  34  24   5   0
95  91  81  61  42  27
91  53  36  27   4   2
95  92  82  21  18  16
71  69  47  38  26  12
```

Одинаковые числа

```
 1   1   1   1   1   1   1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1

 1   1   1   1   1   1   1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1
 1   1   1   1   1   1
```

Основной алгоритм:



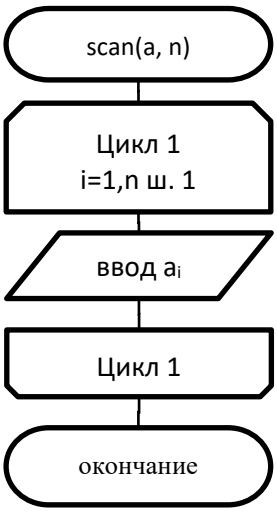
Вспомогательные алгоритмы:

1) Алгоритм ввода одномерного массива

Входные данные: имя массива, размер.

Результирующие данные: заполненный массив

Схема алгоритма:



Этот алгоритм описывается в программе функцией

void scan(int, int)*

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

Возвращаемое значение –отсутствует

Вспомогательные переменные:

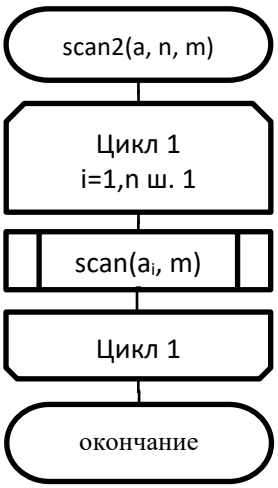
i – тип *int*, индекс переданного массива

2) Алгоритм ввода двумерной динамической матрицы

Входные данные: имя матрицы, кол-во строк, кол-во столбцов.

Результирующие данные: заполненная матрица

Схема алгоритма:



Этот алгоритм описывается в программе функцией

*void scan2(int**, int, int)*

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – кол-во строк матриц

третий параметр – кол-во столбцов матрицы

Возвращаемое значение –отсутствует

Вспомогательные переменные:

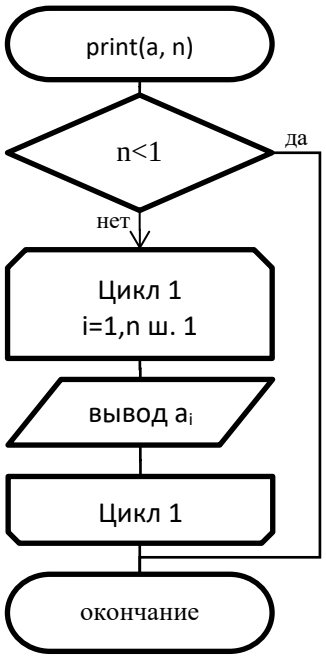
i – тип *int*, индекс переданного массива

3) Алгоритм вывода одномерного массива

Входные данные: имя массива, размер.

Результирующие данные: нет

Схема алгоритма:



Этот алгоритм описывается в программе функцией

void print(int, int)*

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

Возвращаемое значение отсутствует.

Вспомогательные переменные:

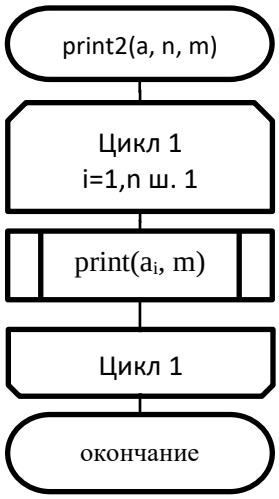
i – тип *int*, индекс переданного массива

4) Алгоритм вывода двумерной динамической матрицы

Входные данные: кол-во строк, кол-во столбцов.

Результирующие данные: нет

Схема алгоритма:



Этот алгоритм описывается в программе функцией

*void print2(int**, int, int)*

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – количество строк матрицы

третий параметр – количество столбцов матрицы

Возвращаемое значение отсутствует.

Вспомогательные переменные:

i – тип *int*, индекс переданного массива

5) Алгоритм выделения памяти для динамической матрицы, матрица хранится по строкам

Этот алгоритм описывается в программе функцией

*int** create2(int**, int, int)*

Параметры:

первый параметр – указатель, под который надо выделить память

второй параметр – количество строк

третий параметр – количество столбцов

Возвращаемое значение – адрес первого элемента в массиве указателей или NULL в случае отсутствия искомого значения.

Вспомогательные переменные:

i – тип *int*, индекс переданного массива, индекс строки

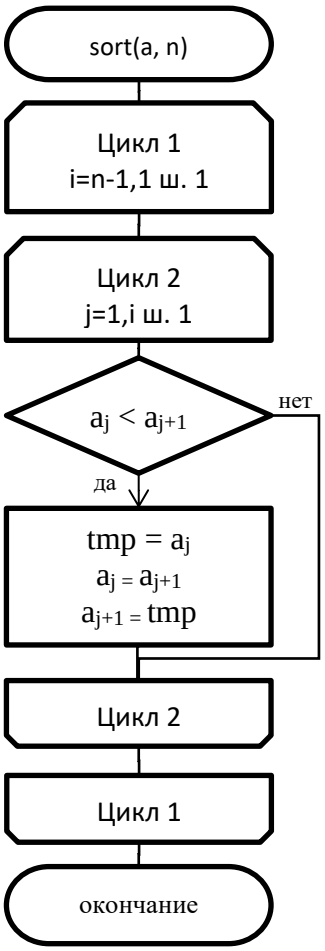
j – тип *int*, переменная для очистки памяти, выделенной под строки, в том случае если память не выделилась под строку.

6) Алгоритм сортировки одномерного массива по убыванию

Входные данные: имя массива, размер.

Результирующие данные: массив будет отсортирован по убыванию

Схема алгоритма:



Этот алгоритм описывается в программе функцией

void sort (int, int)*

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

Возвращаемое значение – отсутствует

Вспомогательные переменные:

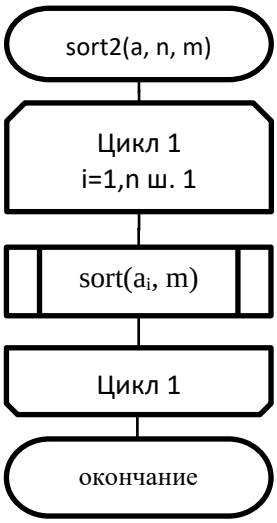
i,j – тип *int*, индексы переданного массива

7) Алгоритм сортировки строк матрицы по убыванию

Входные данные: имя массива, размер.

Результирующие данные: числа в строках будут отсортированы по убыванию.

Схема алгоритма:



Этот алгоритм описывается в программе функцией

*void sort2 (int**, int, int)*

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – кол-во строк матрицы

третий параметр – кол-во столбцов матрицы

Возвращаемое значение – отсутствует

Вспомогательные переменные:

i,– тип *int*, индекс массива a, отвечающий за строки

8) Алгоритм освобождения памяти, выделенной под динамическую матрицу

Входные данные: имя матрицы, кол-во строк.

Результирующие данные: выделенная под матрицу память будет очищена

Этот алгоритм описывается в программе функцией

*void free2(int**, int)*

Параметры:

первый параметр – адрес первой строки матрицы.

второй параметр – количество строк

Возвращаемое значение – отсутствует

Вспомогательные переменные:

i – тип *int*, индекс переданного массива, индекс строки

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define L 50
#define N 6
#define C 7
```

```
/*объявление функций*/
void scan(int*, int);
void print(int*, int);
void print2(int**, int, int);
void scan2(int**, int, int);
int** create2(int**, int, int);
void sort(int*, int);
void sort2(int**, int, int);
void free2(int**, int);
int main()
{
    int M[L], **b;
    /*выделение памяти под b*/
    b=create2(b,N,C);
    /*ввод M*/
    scan(M,L);
    /*ввод b*/
    scan2(b,N,C);
    /*сортировка M*/
    sort(M,L);
    /*сортировка b*/
    sort2(b,N,C);
    /*печать M*/
    print(M,L);
    /*печать b*/
    print2(b,N,C);
    /*освобождение памяти, выделенной под b*/
    free2(b,N);
    return 0;
}
/*фунция ввода одномерного массива*/
void scan(int* a, int n)
{
    int i;
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        //a[i] = rand()%100;

}
/*фунция печати одномерного массива*/
void print(int*a, int n)
{
    int i;

    if (n<1)
        return;
    for(i=0; i<n; i++)
        printf("%3d ", a[i]);
    printf("\n");
}
/*фунция ввода двумерной динамической матрицы*/
void scan2(int** a, int n, int m)
{
    int i;
    for(i=0; i<n; i++)
        /*используем ф-ию scan, чтобы ввести каждую строку*/
        scan(a[i], m);
}
/*фунция вывода двумерной динамической матрицы*/
void print2(int**a, int n, int m)
{
    int i;
    for(i=0; i<n; i++)
```

```
        /*используем ф-ию print, чтобы ввести каждую строку*/
        print(a[i], m);
}
/*фунция выделения памяти под двумерную матрицу*/
int** create2(int**a, int n, int m)
{
    int i,j;
    /*выделяем память под указатели на строки*/
    a = malloc(n*sizeof(int*));
    /*если память не выделилась, возвращаем NULL*/
    if (a==NULL)
        return NULL;
    for(i=0; i<n; i++)
    {
        /*выделяем память под каждую строку,
        если не выделилась, очищаем память
        под уже выделенные строки и возвращаем NULL*/
        a[i] = malloc(m*sizeof(int));
        if (a[i]==NULL)
        {
            for(j=0;j<i;j++)
                free(a[j]);
            free(a);
            return NULL;
        };
    };
    return a;
}
/*функция сортировки одномерного
массива пузырьком по убыванию*/
void sort(int*a, int n)
{
    int i,j, tmp;
    for(i=n-1; i>0; i--)
        for(j=0; j<i; j++)
            if(a[j]<a[j+1])
            {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            };
}
/*функция сортировки строк динамической
матрицы по убыванию*/
void sort2(int**a, int n, int m)
{
    int i;
    for(i=0; i<n; i++)
        /*сортируем каждую строку*/
        sort(a[i], m);
}
/*функия особождения памяти, выделенной под матрицу*/
void free2(int**a, int n)
{
    int i;
    for(i=0;i<n;i++)
        /*очищаем каждую строку*/
        free(a[i]);
    /*очищаем указатели на строки*/
    free(a);
}
```

Задача 2. Удалить из матрицы A (7x5) все строки, сумма элементов которых четная, а из матрицы

В (6x8) – строки, сумма элементов которых нечетная, передвинув на их место следующие строки без нарушения порядка их следования.

Исходные данные:

n1, m1, n2, m2 – размеры матриц *a* и *b*, *n1* и *n2* – кол-во строк, *m1* и *m2* – кол-во столбцов, тип *int*.

a, b – указатели на матрицы, тип *int***, память выделена методом динамической матрицы

Результирующие данные:

матрицы *a* и *b*, тип *int*

Вспомогательные переменные:

z – кол-во удаленных строк, тип *int*

Таблица тестирования:

```
4 4 5 5
 41  67  34   0
 69  24  78  58
 62  64   5  45
 81  27  61  91

 95  42  27  36  91
  4   2  53  92  82
 21  16  18  95  47
 26  71  38  69  12
 67  99  35  94   3

3 line(s) was deleted

a
 69  24  78  58

3 line(s) was deleted

b
 26  71  38  69  12
 67  99  35  94   3

Process returned 0 (0x0)
```

```
6 6 7 7
 41  67  34   0  69  24
 78  58  62  64   5  45
 81  27  61  91  95  42
 27  36  91   4   2  53
 92  82  21  16  18  95
 47  26  71  38  69  12

 67  99  35  94   3  11  22
 33  73  64  41  11  53  68
 47  44  62  57  37  59  23
 41  29  78  16  35  90  42
 88   6  40  42  64  48  46
  5  90  29  70  50   6   1
 93  48  29  23  84  54  56

2 line(s) was deleted

a
 41  67  34   0  69  24
 81  27  61  91  95  42
 27  36  91   4   2  53
 47  26  71  38  69  12

6 line(s) was deleted

b
 88   6  40  42  64  48  46
Process returned 0 (0x0)
```

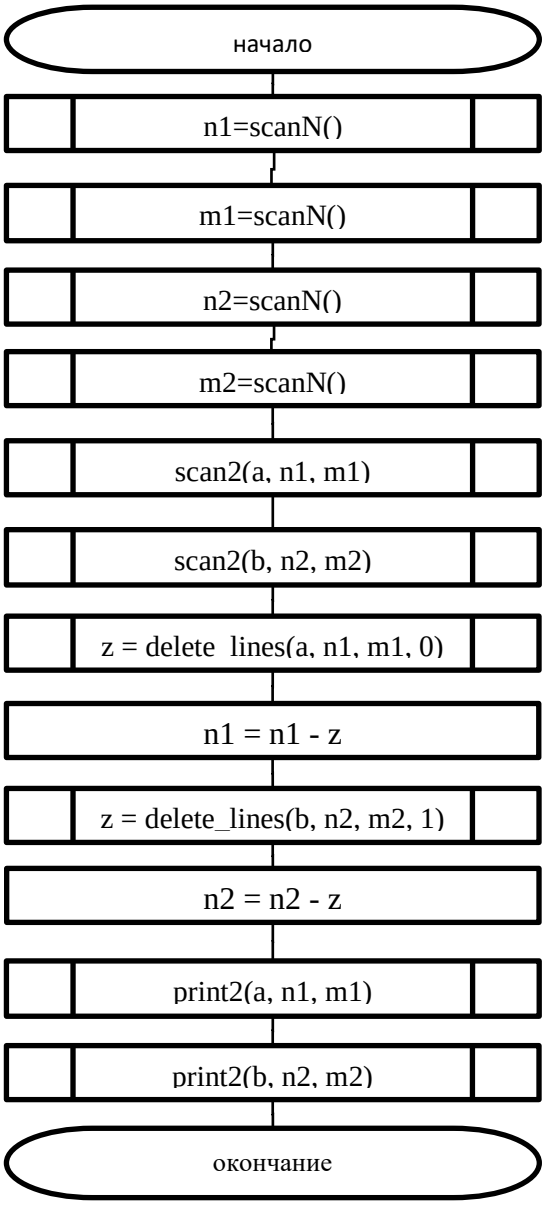
Если все строки матрицы должны быть удалены

```
1 1 1 1
 41

 67

0 line(s) was deleted

a
 41

1 line(s) was deleted

b

Process returned 0 (0x0)
```

```
2 2 1 2
 41  67
 34   0

 69  24

2 line(s) was deleted

a

1 line(s) was deleted

b

Process returned 0 (0x0)
```

```
2 2 1 1
 41  67
 34   0

 69

2 line(s) was deleted

a

1 line(s) was deleted

b

Process returned 0 (0x0)
```

Основной алгоритм:



Вспомогательные алгоритмы:

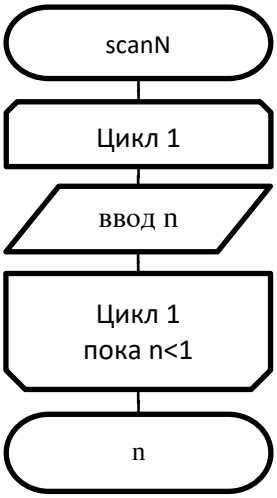
1) Алгоритм ввода одномерного массива и описывающая его функция те же, что и в предыдущей задаче
2) Алгоритм ввода двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
3) Алгоритм вывода одномерного массива и описывающая его функция те же, что и в предыдущей задаче
4) Алгоритм вывода двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
5) Алгоритм выделения памяти для двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
6) Алгоритм освобождения памяти, выделенной под двумерную матрицу и описывающая его функция те же, что и в предыдущей задаче

7) Алгоритм ввода натурального числа

Входные данные: нет.

Результирующие данные: нет

Схема алгоритма:



Этот алгоритм описывается в программе функцией

void scanN ()

Параметры: нет

Возвращаемое значение – натуральное число, введенное пользователем

Вспомогательные переменные:

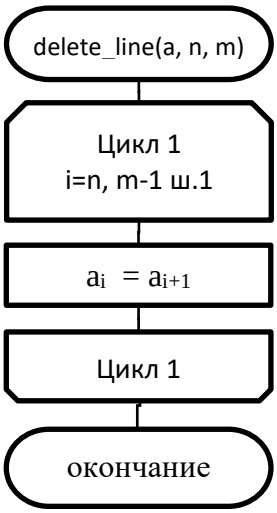
n – тип *int*, вводимая переменная

8) Алгоритм удаления строки матрицы

Входные данные: имя матрицы, индекс строки, кол-во строк матрицы.

Результирующие данные: удаление строки со смещение последующих строк на ее место.

Схема алгоритма:



Этот алгоритм описывается в программе функцией

*void delete_line(int**, int, int)*

Параметры:

первый параметр – адрес матрицы

второй параметр – индекс строки, которую надо удалить

третий параметр – кол-во строк матрицы

Возвращаемое значение – отсутствует

Вспомогательные переменные:

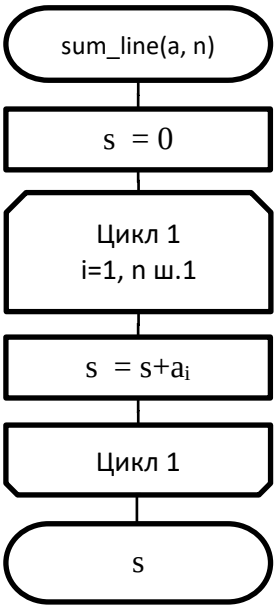
i – инедкс строки, тип *int*

9) Алгоритм суммы строки

Входные данные: имя матрицы, индекс строки, кол-во строк матрицы.

Результирующие данные: удаление строки со смещение последующих строк вперед.

Схема алгоритма:



Этот алгоритм описывается в программе функцией

int sum_line(int, int)*

Параметры:

первый параметр – адрес первого эл-та строки

второй параметр – кол-во столбцов матрицы

Возвращаемое значение – сумма эл-ов строки

Вспомогательные переменные:

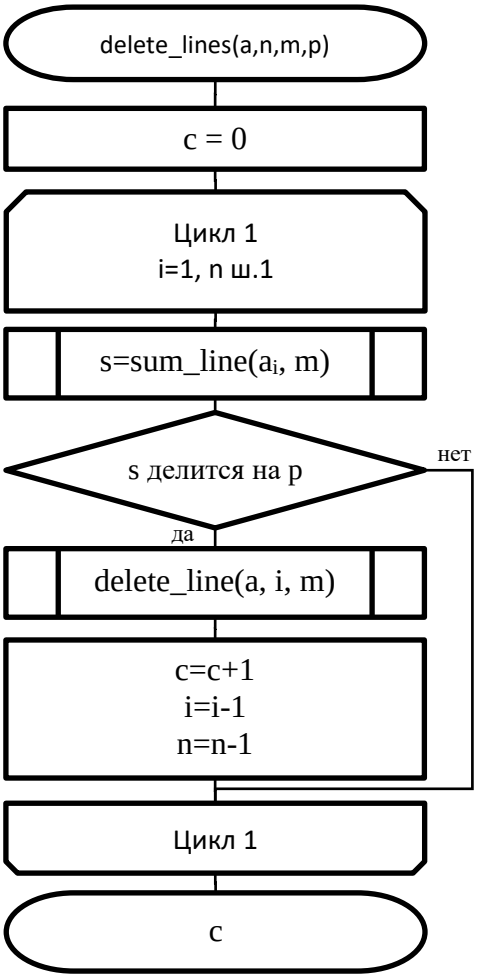
i – индекс столбца, тип *int*

10) Алгоритм удаления строк, сумма эл-ов которых четная или нечетная

Входные данные: имя матрицы, индекс строки, кол-во строк матрицы.

Результирующие данные: удаление строки со смещением последующих строк на ее место.

Схема алгоритма:



Этот алгоритм описывается в программе функцией

*int delete_lines(int**, int, int, int)*

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – кол-во строк матрицы

третий параметр – кол-во столбцов матрицы

четвертый параметр – параметр в зависимости от которого будет определяться какие строки удалять, четные (0), нечетные (1)

Возвращаемое значение – кол-во удаленных строк

Вспомогательные переменные:

i – индекс строки, тип *int*

s – сумма эл-тов строки, тип *int*

c – счетчик удаленных строк, тип *int*

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
/*объявление функций*/
int scanN();
void scan(int*, int);
void print(int*, int);
void print2(int**, int, int);
void scan2(int**, int, int);
int** create2(int**, int, int);
void free2(int**, int);
void delete_line(int**, int, int);
int delete_lines(int**, int, int, int);
int sum_line(int*, int);

int main()
{
    /*объявление переменных и ввод некторых с помощью ф-ии scanN*/
    int **a, **b, z, i, n1=scanN(),m1=scanN(),n2=scanN(),m2=scanN();
    /*выделение памяти под матрицы*/
    a = create2(a, n1, m1);
    b = create2(b, n2, m2);
    /*ввод матриц*/
    scan2(a, n1, m1);
    scan2(b, n2, m2);
    /*печать матриц*/
    print2(a, n1, m1);
    printf("\n");
    print2(b, n2, m2);
    printf("\n");
    /*удаление строк с четной суммой эл-тов из первой матрицы*/
    z=delete_lines(a, n1, m1, 0);
    printf("%d line(s) was deleted\n\na\n",z);
    /*уменьшаем общее кол-во строк первой матрицы на кол-во удаленных,
    то есть фиксируем, что строк стало меньше, чтобы не освобождать память
    одного указателя несколько раз*/
    n1-=z;
    /*печать матрицы*/
    print2(a, n1, m1);
    /*удаление строк с нечетной суммой эл-тов из второй матрицы*/
    z=delete_lines(b, n2, m2, 1);
    printf("\n%d line(s) was deleted\n\nb\n",z);
    n2-=z;
    /*печать матрицы*/
    print2(b, n2, m2);
    free2(a, n1);
    free2(b, n2);
    return 0;
}
/*ф-ия ввода натурального число*/
int scanN()
{
    int n;
    do
        scanf("%d",&n);
    while (n<1);
    return n;
}

/*фунция ввода одномерного массива*/
```

```
void scan(int* a, int n)
{
    int i;
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        //a[i] =rand()%100;

}
/*фунция печати одномерного массива*/
void print(int*a, int n)
{
    int i;
    if (n<1)
        return;
    for(i=0; i<n; i++)
        printf("%3d ", a[i]);
    printf("\n");
}
/*фунция ввода двумерной динамической матрицы*/
void scan2(int** a, int n, int m)
{
    int i;
    for(i=0; i<n; i++)
        /*используем ф-ию scan, чтобы ввести каждую строку*/
        scan(a[i], m);
}
/*фунция вывода двумерной динамической матрицы*/
void print2(int**a, int n, int m)
{
    int i;
    for(i=0; i<n; i++)
        /*используем ф-ию print, чтобы ввести каждую строку*/
        print(a[i], m);
}
/*фунция выделения памяти под двумерную матрицу*/
int** create2(int**a, int n, int m)
{
    int i,j;
    /*выделяем память под указатели на строки*/
    a = malloc(n*sizeof(int*));
    /*если память не выделилась, возвращаем NULL*/
    if (a==NULL)
        return NULL;
    for(i=0; i<n; i++)
    {
        /*выделяем память под каждую строку,
        если не выделилась, очищаем память
        под уже выделенные строки и возвращаем NULL*/
        a[i] = malloc(m*sizeof(int));
        if (a[i]==NULL)
        {
            for(j=0;j<i;j++)
                free(a[j]);
            free(a);
            return NULL;
        };
    };
    /*возращаем указатель*/
    return a;
}

/*функия особождения памяти, выделенной под матрицу*/
void free2(int**a, int n)
```

```
{
    int i;
    for(i=0;i<n;i++)
        /*очищаем каждую строку*/
        free(a[i]);
    /*очищаем указатели на строки*/
    free(a);
}

/*ф-ия удаления определенной строки по ее индексу*/
void delete_line(int**a, int n, int m)
{
    int i;
    free(a[n]);
    for(i=n; i<m-1; i++)
        a[i] = a[i+1];
}
/*функция нахождения суммы эл-тов определенной строки по индексу*/
int sum_line(int* a, int n)
{
    int i,s=0;
    for (i=0;i<n;i++)
        s+=a[i];
    return s;
}
/*функция удаления строк в зависимости от суммы их эл-тов, четных или нечетных*/
int delete_lines(int**a, int n, int m, int p)
{
    int i,s,c=0;
    /*проход циклом по всем строкам*/
    for(i=0; i<n; i++)
    {
        /*вызываем ф-ию нахождения суммы строки и записываем в s*/
        s=sum_line(a[i],m);
        /*определяем четность числа*/
        if (s%2==p)
        {
            /*удаляем строку*/
            delete_line(a,i,n);
            /*подсчитываем удаленных строки*/
            c++;
            /*уменьшаем i, чтобы не пропустить потенциальную строку к
удалению*/
            i--;
            /*фиксируем, что строк стало меньше*/
            n--;
        };
    };
    /*возвращаем кол-во удаленных строк*/
    return c;
}
```

Задача 3.

Определить функцию *Integral()* для приближенного вычисления определенного интеграла вида методом трапеций. Использовать эту функцию для вычисления значений двух интегралов, передавая подынтегральную функцию в функцию *Integral()* в качестве параметра. $\int_{-1}^{3} \frac{2x\,dx}{e^{2x}}$, $\int_{-3}^{2} \sqrt{x^2+1}\,dx$, N=40

Формула трапеций $\int_a^b f(x)dx \cong \frac{b-a}{N}\left(\frac{y_0+y_N}{2} + y_1 + y_2 + \ldots + y_{N-1}\right)$

Исходные данные:

нижний и верхний пределы интегрирования задаются константами

подынтегральные функции описываются функциями в программе.

Результирующие данные:

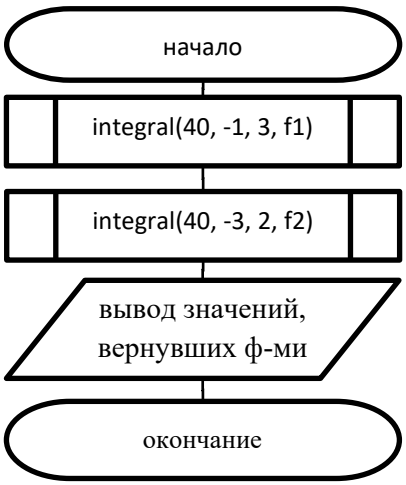
значение интеграла – вещественное число

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
$\int_{-1}^{3} \frac{2x\,dx}{e^{2x}}$	$\int_{-1}^{3} \frac{2x}{e^{2x}}\,dx = -\frac{7+e^8}{2e^6} \approx -3.70320368208366$ Computed by Wolfram|Alpha Wolframalpha	-3.741885 8.347300 Process returned 0 (0x0)
$\int_{-3}^{2} \sqrt{x^2+1}\,dx$	$\int_{-3}^{2} \sqrt{x^2+1}\,dx = \frac{1}{2}\left(2\sqrt{5} + 3\sqrt{10} + \sinh^{-1}(2) + \sinh^{-1}(3)\right) \approx 8.61052543495780$ Computed by Wolfram|Alpha Wolframalpha	

Результат работы программы не совпадает с ожидаемым из-за малого числа равных отрезков или же частей, на которые разбивается интервал для вычисления площади под кривой.

Схема программы

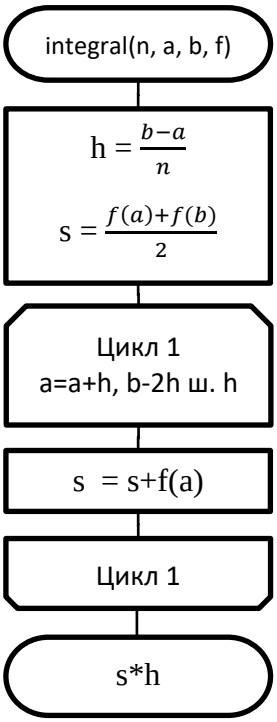


Вспомогательные алгоритмы
Алгоритм вычисления интеграла

Входные данные: кол-во частей на которые поделен отрезок, начало отрезка, конец отрезка, имя функции.

Результирующие данные: нет

Схема алгоритма:



Этот алгоритм описывается в программе функцией

double integral(int, double, double, double(*)(double))

Параметры:

первый параметр – кол-во частей

второй параметр – начало отрезка

второй параметр – конец отрезка

второй параметр – адрес на функции

Возвращаемое значение – значение интеграла

Вспомогательные переменные:

h – длина части, тип *double*

s – сумма площадей трапеций, тип *double*

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/*объявление ф-ий*/
double f1(double);
double f2(double);
double integral(int, double, double, double(*)(double));
int main()
{
    /*печать значения интеграла*/
    printf("%lf\n%lf",integral(40, -1, 3, f1),integral(40, -3, 2, f2));
    return 0;
}
```

```
double f1(double x)
{
    return 2*x/exp(2*x);
}

double f2(double x)
{
    return sqrt(x*x+1);
}
/*ф-ия вычисления интеграла методом трапеций*/
double integral(int n, double a, double b, double (*f)(double))
{
    double h =(b-a)/n, s=((f(a)+f(b))/2);
    for(a+=h; a<b-h; a+=h)
        s+=f(a);
    return h*s;
}
```