

Лабораторная работа №4

Работа с указателями. Передача параметров в функцию через указатель и ссылку. Обработка числовых массивов и текстов. Работа с библиотечными функциями.

Цель работы:

1. Изучить методы работы с указателями.
2. Продолжить изучение использования пользовательских функций :
 - Применить указатели при взаимодействии с функциями (передача параметров и возврат значений)
 - Применить указатели и функции в задачах по обработке числовых массивов.
 - Применить указатели и функции в задачах обработки текстов.
3. Применить библиотечные функции в задачах обработки текстов.

Теоретические сведения.

Указатели.

Работая с памятью, программист оперирует понятиями имя и значение переменной – это работа на высоком уровне абстракции. На машинном (низком уровне), компьютер оперирует понятиями «адрес памяти» и «содержимое памяти». Указатели позволяют программисту работать на низком уровне абстракции, оперируя адресами, а не именами объектов.

Определяя переменную, например: `int var = 1;` программа заставляет компилятор выделять память, при этом имя переменной `var` адресует участок памяти, а константа `1` определяет значение, которое запишется по этому адресу.

Имея в своем распоряжении адрес переменной, программисту необходимо с ним работать: сохранять, передавать и преобразовывать, для этой цели и служат указатели.

Указатели - это переменные, значениями которых являются адреса других объектов - переменных, массивов, функций.

Указатели служат для обращения к области памяти, отведенной под другую переменную.

Итак, для того, чтобы работать с указателем, необходимы, по крайней мере, две переменные – сам указатель и переменная на которую он будет ссылаться. Подобно любой другой переменной, указатель нужно создать (объявить) и задать начальное значение (установить указатель), после чего он будет указывать (ссылаться) на переменную. При объявлении указателя перед именем ставится знак `*`.

Пример: Объявление и установка указателей

```
int    x, *p_i;    // объявить (выделить память) переменную x и указатель p_i
double f, *p_d;    // объявить переменную f и указатель p_d
p_i = &x;          // установить указатель p_i на переменную x
p_d = &f;
```

Обратите внимание, что указатель должен быть того же типа, что и переменная, на которую он ссылается.

Операция `&x` означает: "получить адрес переменной `x`".

Оператор `p_d = &f` означает следующее:

адрес переменной `f` занести в указатель `p_d` (указатель установлен на переменную `f`).

Установка указателя на объект – это обязательный этап работы с указателем.

Будьте внимательны, неустановленный указатель главный источник неприятностей !

После того, как указатель установлен, можно обращаться к объекту, на который он указывает, для этой цели служит специальная операция – разыменование «*». Операция * рассматривает свой операнд как адрес и позволяет обратиться к содержимому этого адреса.

Пример: Использование указателей в выражениях.

```
int x, *ptr;
ptr = &x;           // ptr ссылается на x
*ptr = *ptr + 1;    // аналог: x = x+1
*ptr = *ptr*5;      // x=x*5
```

Указатели и массивы.

В языке C между указателями и массивами существует тесная связь.

Рассмотрим, что происходит, когда объявляется массив, например `int array[25]`:

- выделяется память для двадцати пяти элементов массива типа `int`
- создается указатель на начало выделенной памяти с именем `array`

Массив остается безымянным, а доступ к его элементам осуществляется через указатель `array`. Следует отметить, что указатель `array` является константным, его значение нельзя менять, то есть он всегда указывает на нулевой элемент массива (его начало).

Имя массива можно приравнять к указателю, так как оно также является указателем.

...

```
int array[25], *ptr;
ptr = array; // установка указателя ptr на начало массива
```

Оператор `ptr=array` можно заменить на эквивалентный ему: `ptr=&array[0]`.

К элементу массива можно обратиться двумя способами:

- с помощью индекса
- с помощью указателя.

Пример: Сравнение двух способов доступа к элементам массива.

// определение массива `mas` и указателя `q`

```
char mas[100] , *q;
```

```
q = mas; // установка указателя на начало массива
```

// Доступ к элементам массива (следующие записи эквивалентны):

```
mas[0];           // *(q+0)
mas[10];          // *(q+10)
mas[n-1];         // *(q+n-1)
```

Арифметика указателей.

Работая с указателем программист имеет возможность совершать действия двух различных типов :

- обращаться к объекту (к данным), на который ссылается указатель, используя операцию * (обращение по адресу указателя);
- изменять адрес, записанный в указателе, таким образом, меняя объект с которым указатель связан.

Манипуляции с адресом часто называют *арифметикой указателей* или *адресной арифметикой*. Это очень удобный инструмент, позволяющий легко манипулировать массивами

как простых, так и сложных типов данных. Арифметические операции с адресами имеют ряд ограничений, которые мы и рассмотрим подробнее.

Присваивание

Операцию присваивания можно применять :

- при занесении адреса в указатель;
- при занесении нулевого значения в указатель;
- два указателя можно приравнять друг другу.

Никакие другие типы присваивания с указателями недопустимы.

Пример:

```
int *p,*k,*z,date=2007;
p=&date;           // установка указателя на переменную date
k=p;               // значение указателя p заносится в указатель k
z=NULL;            // нулевое значение занести в указатель
```

Инкремент и декремент

Операции ++ и -- применённые к указателю перемещают ссылку на следующий (или предыдущий) объект, позволяя последовательно перебирать элементы массива в прямом или обратном направлении.

Пример:

```
float m[10], *pk;
char q[60], *ps;
ps=q;           // установить указатель на q[0] (первый элемент)
pk=&m[9];        // установить указатель на m[9] (последний элемент)
ps++;           // переместить указатель на q[1]
pk--;           // переместить указатель на m[8]
```

Заметим, что в данном примере обращение к элементам массива нет, мы лишь манипулируем с адресами в указателях. Для того, чтобы обратиться к данным, например, напечатать элемент массива, нужно применить операцию разыменования

```
printf("%c \n",*ps);    // печать q[1]
printf("%f \n",*pk);    // печать m[8]
```

Сложение

Два указателя нельзя складывать, однако к указателю можно прибавить целую величину.

Если помнить, что указатели содержат адреса, то подобные ограничения вполне объяснимы, действительно, складывать между собой два разных адреса не имеет смысла, а вот прибавить к адресу некоторое число, означает переместиться вперед, «перешагнуть» через несколько объектов.

Пример: Вывести каждый четвертый элемент массива на печать.

```
double mas[N], *pk;
... // инициализация массива
for (pk=&mas[0] ; pk<&mas[N-1]; pk=pk+3)
printf("\t%e",*pk); // печать элемента массива
```

Вычитание

Два указателя можно вычитать друг из друга, если они одного типа.

Эта операция дает расстояние между объектами. Из адреса указателя можно вычесть целую величину, таким образом, перемещаясь в сторону уменьшения адресов.

Пример:

Вывести каждый второй элемент массива в обратном порядке.

```
...
for (pk=&mas[N-1] ; pk>M; pk=pk-1)
printf("\t%f",*pk); // печать элемента массива
```

Указатели в параметрах функции.

Переменные, объявленные в функциях - это локальные (временные) переменные, которые уничтожаются при выходе из функции, а значит, их значения теряются. Когда же нужно сохранить вычисляемые значения требуется располагать переменные за пределами функции. В других случаях требуется обрабатывать данные, расположенные за пределами функции. Поэтому достаточно часто возникает необходимость обращения к объектам, определенным вне функции, которые являются внешними по отношению к функции.

Для этой цели служат указатели, и это одно из важнейших применений указателя - организация связи функции с внешней средой.

Указатели в параметрах функции имеют два важных применения:

- доступ к памяти, находящейся за пределами функции (доступ к «внешней» памяти);
- возврат из функции более одного значения.

Если функция должна обращаться к внешней памяти, то параметры следует передавать через указатели.

Пример: Функция изменяет переменную в вызывающей программе.

```
void inp_kl (int *p) // ввод числа с клавиатуры в переменную функции main()
{printf ("\nx=");
scanf ("%d",p); // второй параметр функции scanf – адрес переменной для ввода данных
}

int _tmain(int argc, _TCHAR* argv[])
{int x=0;
inp_kl(&x); // вызов функции inp_kl ()
}
printf("x=%d\n",x);
```

Функция **inp_kl()** имеет один параметр – указатель на **int**, который устанавливается при вызове функции (фактическим параметром). То есть при вызове **inp_kl(&x)** происходит создание

и установка указателя **p** (формального параметра функции) на переменную **x**, то есть происходит следующее действие : **int* p = &x;**

Таким образом, функция **inp_kl()** получает доступ к переменной **x** через указатель **p**

Подобным образом могут изменяться и внешние массивы. В функцию передаётся указатель на массив, через который и ведётся работа с его элементами.

Примеры программирования.

Пример 1: Создать массив с помощью датчика случайных чисел, распечатать на экране и найти его сумму.

```
#include "stdafx.h"
#include <stdio.h>
#include <windows.h>
#define N 10      // размер массива
#define COL 5     // число колонок при выводе массива на экран

int _tmain(int argc, _TCHAR* argv[])
{
    // область определений (выделение памяти)
    double *k;      // указатель на тип double
    double dig[N], s; // массив dig и переменные s и i
    int i;

    //-----
    // Заполнить массив случайными числами
    for (k=dig; k<dig+N; k++)
        *k=rand()%99/3.;

    // печать исходного массива
    for (k=dig,i=0; k<dig+N; k++, i++)
    {
        printf("%8.2f",*k);
        if ((i+1)%COL)    printf("\t");
        else               printf("\n");
    }

    // суммирование массива
    for (s=0,k=dig; k<dig+N; k++)
        s=s+ *k;

    // печать результата
    printf ("\n=====ns=%f\n",s);
    system ("pause");
    return 0;
}
```

Рассмотрим цикл, заполняющий массив **for (k=dig; k<dig+N; k++)** :

- перед началом цикла указатель устанавливается на начало массива, **k=dig**,
- цикл продолжается до тех пор, пока значение указателя меньше адреса последнего элемента массива, **k<dig+N**,
- на каждом проходе цикла дается приращение к адресу указателя **k++** , таким образом, указатель перемещается на следующий элемент

В цикле происходит перебор всех элементов массива от первого до последнего, тело цикла состоит из одного оператора ***k=rand()%99/3.;**

Указатель **k** ссылается на очередной элемент, то есть содержит адрес этого элемента, а операция ***k** – обращение к данным этого элемента. Функция **rand()** возвращает случайное число целого

типа, путем несложного вычисления преобразуем его в вещественное число. Поэтому оператор ***k=rand()%99/3.;** не что иное, как запись в текущий элемент массива случайного вещественного числа.

В цикле печати, кроме указателя **k**, адресующего элементы массива, используется счётчик выведенных чисел **i**, который используется для форматирования. Если **i** кратно **COL**, то необходимо закончить строку (вывести на печать символ «перевод строки»). Суммирование массива проводится в отдельном цикле.

Пример 2: Решим ту же задачу с применением функций. Потребуется три функции: для заполнения массива, для вывода массива на экран и для суммирования массива.

```
#include "stdafx.h"
#include <stdio.h>
#include <time.h>
#include <windows.h>
#include <locale>
#define N 10      // размер массива
#define COL 5     // число колонок массива на экране
//_____ область определения функций_____
// функция инициализации массива
// генерация вещественных положительных чисел
void initmas (double *p, int n)
{
    double *tp; // рабочий указатель (локальная переменная)
    for (tp=p; tp < (p+n); tp++)
        *tp=rand()%99/3.;
}
// функция печати массива
void printmas (double *p, int n, int k)
{
    int i;
    for (i=0; i<n; i++,p++)
    {
        printf("%8.2f",*p); // печать элемента массива
        if ((i+1)%k) printf("\t"); // выбор разделителя
        else          printf("\n");
    }
    printf("\n");
}
// функция вычисления суммы
double summas (double *p, int n)
{
    double fs; //сумма (локальная переменная)
    double *tp; // рабочий указатель (локальная переменная)
    for (fs=0, tp=p; tp < (p+n); tp++)
        fs=fs+ *tp; // суммирование массива
    return fs; // возврат суммы в вызывающую программу
}
//_____ конец области определения функций_____

int _tmain(int argc, _TCHAR* argv[])
{
    //----- настройки
        setlocale(0,"Russian");
        srand (time(0));
    //----- область определений (выделение памяти)
        double dig[N], s;
    //-----
```

```

initmas(dig,N);           // инициализация массива dig
printmas(dig,N,COL);      // печать в 5 колонок
s = summas(dig,N);        // вычисление суммы
printf ("s=%f\n",s);
system ("pause");
return 0;
}

```

Вся необходимая память (dig[] и s) определена в вызывающей программе, поэтому эта память является внешней, по отношению к трём пользовательским функциям : **initmas()**, **printmas()** и **summas()**. Разберем подробно работу с внешним массивом на примере функции **initmas()**.

Чтобы функция имела доступ к массиву dig(), необходимо передать указатель на массив (адрес начала) в качестве параметра функции (первый параметр) и длину массива (второй параметр). При вызове функции **initmas(dig, N)** в функции **main()** происходит инициализация формальных параметров (**p** и **n**) функции **initmas()**, при входе в функцию выполняются следующие действия :

int* p = dig, int n = N, то есть создается указатель **p** и связывается с началом массива **dig**. Также создается переменная **n** и инициализируется длиной массива.

Для текущей работы потребуется ещё один указатель (**tp**), текущая работа заключается в перемещении указателя на очередной элемент массива.

Рассмотрим заголовок цикла **for (tp=p; tp < (p+n); tp++)**

- перед началом цикла текущий указатель устанавливается на начало массива, то есть **tp=p**
- цикл выполняется до тех пор, пока текущий указатель меньше адреса последнего элемента массива (выражение **(p+n)** дает адрес последнего элемента)
- на каждом проходе цикла текущий указатель перемещается на следующий элемент, то есть **tp++**

Теперь понятно, почему потребовался дополнительный указатель **tp**, в самом деле, если на каждом проходе цикла необходимо проверять условие выхода из цикла, то следует где-то хранить адрес начала массива для того, чтобы вычислить адреса конца массива (**p+n**), а значит перемещать указатель **p** нельзя. Следует отметить, что когда мы работаем с массивами через указатель, всегда требуется как минимум два указателя – один для хранения «начала отсчета», а другой для текущей работы.

Две другие функции **printmas()** и **summas()** используют те же принципы работы с указателями. Обратите внимание, на то, как упростилась функция **main()**, логика выполнения программы стала хорошо видна, остались только основные переменные. Все «детали работы» и вспомогательные переменные отошли к функциям, или как часто говорят «скрыты в функциях», и это существенно экономит как память компьютера, так и время на разработку программы. В самом деле, стоит вспомнить, что функцию можно многократно вызывать из различных частей программы с различными параметрами. Поэтому однажды написанная функция может многократно использоваться. Следующие примеры показывают, разнообразную работу с функциями.

Пример 3: Дополним предыдущий пример функцией поиска минимума

```

//_____ область определения функций_____
...
// функция поиска минимума,
// возврат указателя на минимум
double* minmas (double *p, int n)
{
    double* tmin; //указатель на минимум (локальная переменная)
    double* tp;   //рабочий указатель
    for (tmin=tp=p; tp < (p+n); tp++)
        if (*tmin > *tp) tmin=tp;
}

```



```

        return tmin; // возврат указателя на минимум (положение минимума)
} // уничтожение всех локальных переменных: p, n, tmin, tp
// _____ конец области определения функций _____
int _tmain(int argc, _TCHAR* argv[])
{
    ...
    double* q=minmas(dig,N); // получение указателя на минимум
    printf ("окружение минимума: %8.2f\t%8.2f\n",*(q-1),*(q+1));
    system ("pause");
    return 0;
}

```

При получении из функции указателя на результат, возможности работы в вызывающей программе расширяются, так как через указатель можно работать как с адресом, так и с данными. В нашем примере мы получили указатель на минимум и можем его рассматривать как точку отсчета для дальнейшей работы, например распечатать не сам минимум, а его окружение.

В данном решении есть существенный недостаток, не учтены случаи, когда минимум находится на границе массива (на первом или последнем месте).

Внесем в программу необходимые исправления :

```

...
double* q=minmas(dig,N); // получение указателя на минимум
if (q == dig) printf ("значение после минимума: %8.2f\n",*(q+1));
else if (q == dig+N-1) printf ("значение до минимума: %8.2f\n",*(q-1));
else printf ("окружение минимума: %8.2f\t%8.2f\n",*(q-1),*(q+1));
...

```

Пример 4: Поставим задачу формирования массива результатов.

Дан исходный массив вещественных чисел, сформировать результирующий массив из отрицательных элементов исходного массива.

```

#include "stdafx.h"
#include <stdio.h>
#include <time.h>
#include <windows.h>
#include <locale>
#define N 20 // размер массива
#define COL 5 // число колонок массива на экране
// _____ область определения функций _____
// функция инициализации массива
// генерация положительных и отрицательных чисел
void initmas (double *p, int n)
{
    double *tp; // рабочий указатель
    for (tp=p; tp < (p+n); tp++)
        *tp=rand()%999/5.-50;
}
// функция печати массива
void printmas (double *p, int n, int k)
{
    int i;
    for (i=0; i<n; i++,p++)
    {
        printf("%8.2f",*p); // печать элемента массива
        if ((i+1)%k) printf("\t"); // выбор разделителя
        else
            printf("\n");
    }
    printf("\n");
}

```



```

// Функция поиска отрицательных элементов массива
// формирование массива результатов (данные)
// p - указатель на исходный массив, n - размер исходного массива
// r - указатель на результирующий массив
// возврат значения (размер массива результатов)
int minusmas (double* p, int n, double* r)
{double* tp;           // рабочий указатель для исходного массива
double* tr;           // рабочий указатель для массива результатов
for (tp=p,tr=r; tp < (p+n); tp++)
if (*tp<0) {*tr=*tp; tr++;}
return (tr-r);         // размер массива результатов
}
//_____ конец области определения функций_____

int _tmain(int argc, _TCHAR* argv[])
{
//----- настройки
    setlocale(0,"Russian");
    srand (time(0));
// область определений (выделение памяти)
    double dig[N],rez[N];
//-----
    initmas(dig,N);      // инициализация исходного массива
    printmas(dig,N,COL); // печать исходного массива
    int size=minusmas(dig,N,rez); // формирование массива отрицательных элементов
    printf ("Массив результатов -----\\n");
    printmas(rez,size,COL); // печать массива результатов
    system ("pause");
    return 0;
}

```

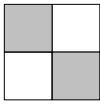

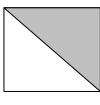
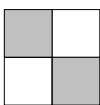
Когда нужно сформировать массив результатов, размер которого заранее не известен, можно поступить следующим образом:

- Выделить память заведомо большего размера (с запасом). В нашем случае размер результирующего массива **rez** равен размеру исходного массива **dig**
- В процессе работы вычислить истинный размер результирующего массива. В нашем случае функция **minusmas()** возвращает размер массива с результатами

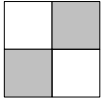
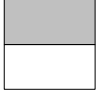
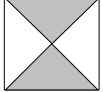
Вопросы.

1. Дайте понятие указателя. Для каких целей он служит?
2. Поясните следующие понятия: установка указателя, тип указателя.
3. Какие действия выполняют операции * и & ?
4. Какие действия необходимо выполнить, чтобы обратиться к переменной через указатель (начиная с объявления указателя).
5. Что такое адресная арифметика?
6. Какие ограничения действуют в адресной арифметике на операцию «присваивания»?
7. Какие ограничения действуют в адресной арифметике на операции «инкремента» и «декремента»?
8. Какие ограничения действуют в адресной арифметике на «сложение» и «вычитание»?
9. Что представляет собой массив данных с точки зрения указателей?
10. Перечислите три направления применения указателей в параметрах функций.
11. Как функция может обратиться к памяти, находящейся за её пределами? Приведите пример.

Варианты задания	Условия задания	
1, 16	<p>Создать функцию для поиска положения максимального элемента произвольного массива вещественных чисел. Вернуть указатель на максимальный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение максимумов в обоих массивах 3. Определить в каком массиве максимум находится ближе к началу массива 	
		<p>Дана целочисленная матрица $N \times N$</p> <p>Создать функцию, которая возвращает 2 значения: минимум и максимум заштрихованной области</p>
2, 17	<p>Создать функцию для поиска положения первого положительного элемента произвольного массива целых чисел. Вернуть указатель на положительный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение положительного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить количество элементов в первой и второй частях массива 4. Пункты 2 и 3 выполнить для массивов A и B 	
		<p>Дана целочисленная матрица $N \times M$</p> <p>Создать функцию, которая возвращает 2 значения : максимум в верхней заштрихованной области и минимум в нижней.</p>
3, 18	<p>Создать функцию для поиска положения последнего отрицательного элемента произвольного массива целых чисел. Вернуть указатель на отрицательный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение отрицательного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить среднее арифметическое второй части массива 4. Пункты 2 и 3 выполнить для массивов A и B 	
		<p>Дана целочисленная матрица $N \times N$</p> <p>Создать функцию, которая возвращает 2 значения : Сумму элементов левой и правой заштрихованной</p>
4, 19	<p>Создать функцию для поиска положения минимального элемента произвольного массива вещественных чисел. Вернуть указатель на минимальный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение минимумов в обоих массивах. Полученные указатели разделяют оба массива на 2 части. 3. Определить в каком массиве сумма элементов первой части больше. 	
		<p>Дана целочисленная матрица $N \times N$</p> <p>Создать функцию, которая возвращает 2 значения : минимум и максимум заштрихованной области</p>

5, 20	<p>Создать функцию для поиска положения первого отрицательного элемента произвольного массива целых чисел. Вернуть указатель на отрицательный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение отрицательного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить количество элементов в первой и второй частях массива 4. Пункты 2 и 3 выполнить для массивов A и B
	<p>Дана целочисленная матрица $N \times M$</p> <p>Создать функцию, которая возвращает 2 значения : суммы верхней и нижней заштрихованной области</p>
6, 21	<p>Создать функцию для поиска положения последнего положительного элемента произвольного массива целых чисел. Вернуть указатель на положительный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение положительного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить минимальный элемент в первой части массив 4. Пункты 2 и 3 выполнить для массивов A и B
	<p>Дана целочисленная матрица $N \times M$</p> <p>Создать функцию, которая возвращает 2 значения : минимум и максимум заштрихованной области</p>
7, 22	<p>Создать функцию для поиска положения первого нулевого элемента произвольного массива целых чисел. Вернуть указатель на нулевой элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение нулевого элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить сумму элементов в первой и во второй частях массива 4. Пункты 2 и 3 выполнить для массивов A и B
	<p>Дана целочисленная матрица $N \times N$.</p> <p>Создать функцию, которая возвращает 2 значения : максимальный минимальный по модулю элемент в заштрихованной области.</p>
8, 23	<p>Создать функцию для поиска положения минимального элемента произвольного массива вещественных чисел. Вернуть указатель на минимальный элемент.</p> <p>Тестовая программа:</p> <ul style="list-style-type: none"> – Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ – Найти положение минимумов в обоих массивах – Определить в каком массиве минимум находится ближе к концу массива
	<p>Дана целочисленная матрица $N \times M$.</p> <p>Создать функцию, которая возвращает 2 значения : максимум в верхней заштрихованной области и минимум в нижней.</p>

9, 24	<p>Создать функцию для поиска положения максимального элемента произвольного массива вещественных чисел. Вернуть указатель на максимальный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение максимумов в обоих массивах. Точка максимума (его положение) разделяет массив на 2 части. 3. Определить в каком массиве сумма элементов второй части меньше
	<div data-bbox="316 454 416 555"></div> <p>Дана целочисленная матрица $N \times N$</p> <p>Создать функцию, которая возвращает 2 значения :</p> <p>Найти количество нулей в верхней заштрихованной области и количество положительных элементов в нижней.</p>
10, 25	<p>Создать функцию для поиска положения последнего отрицательного элемента произвольного массива целых чисел. Вернуть указатель на отрицательный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение отрицательного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить максимальный элемент в второй части массив 4. Пункты 2 и 3 выполнить для массивов A и B
	<div data-bbox="316 992 416 1093"></div> <p>Дана целочисленная матрица $N \times M$, содержащая как положительные, так и отрицательные элементы.</p> <p>Создать функцию, которая возвращает 2 значения :</p> <p>количество положительных и отрицательных элементов в заштрихованной области.</p>
11, 26	<p>Создать функцию для поиска положения последнего положительного элемента произвольного массива целых чисел. Вернуть указатель на положительный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение положительного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить количество отрицательных элементов во второй части массива 4. Пункты 2 и 3 выполнить для массивов A и B
	<div data-bbox="316 1507 416 1608"></div> <p>Дана целочисленная матрица $N \times N$, содержащая как положительные, так и отрицательные элементы.</p> <p>Создать функцию, которая возвращает 2 значения :</p> <p>количество положительных четных и отрицательных четных элементов в заштрихованной области.</p>
12, 27	<p>Создать функцию для поиска положения последнего положительного элемента произвольного массива целых чисел. Вернуть указатель на положительный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение положительного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить минимальный элемент в первой части массив 4. Пункты 2 и 3 выполнить для массивов A и B
	<div data-bbox="316 2022 416 2123"></div> <p>Дана целочисленная матрица $N \times N$</p> <p>Создать функцию, которая возвращает 2 значения :</p> <p>минимум в левой заштрихованной области и максимум в правой</p>

13, 28	<p>Создать функцию для поиска положения первого отрицательного элемента произвольного массива целых чисел. Вернуть указатель на отрицательный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение отрицательного элемента массива. Полученный указатель разделяет массив на 2 части. 3. Определить среднее арифметическое элементов второй части массива 4. Пункты 2 и 3 выполнить для массивов A и B
	<p>Дана целочисленная матрица $N \times M$</p> <p>Создать функцию, которая возвращает 2 значения : среднее арифметическое каждой заштрихованной области (отдельно верхней и нижней).</p>
14, 29	<p>Создать функцию для поиска положения максимального элемента произвольного массива вещественных чисел. Вернуть указатель на максимальный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение максимумов в обоих массивах 3. Определить в каком массиве максимум находится дальше от начала массива
	<p>Дана целочисленная матрица $N \times M$, содержащая как положительные, так и отрицательные элементы.</p> <p>Создать функцию, которая возвращает 2 значения : среднее арифметическое положительных и отрицательных элементов заштрихованной области</p>
15, 30	<p>Создать функцию для поиска положения минимального элемента произвольного массива вещественных чисел. Вернуть указатель на минимальный элемент.</p> <p>Тестовая программа:</p> <ol style="list-style-type: none"> 1. Создать и инициализировать 2 массива $A[N1]$, $B[N2]$ 2. Найти положение минимумов в обоих массивах. Полученные указатели разделяют оба массива на 2 части. 3. Определить в каком массиве среднее арифметическое второй части больше.
	<p>Дана целочисленная матрица $N \times M$.</p> <p>Создать функцию, которая возвращает 2 значения : количество нулей в заштрихованной области каждой заштрихованной области (отдельно верхней и нижней)</p>