**Лабораторная работа №2**

**Тема: Программирование циклических алгоритмов.**

**Одномерные массивы и матрицы (двумерные массивы). Работа с индексами.**

***Цель работы:***

1. Изучить возможности операторов цикла **for, while, do-wh**ile.
2. Получить навыки построения циклических алгоритмов, разработки и отладки циклических программ.
3. Изучение методов работы с массивами данных с применением индексов.
4. Получение навыков обработки числовых массивов.
5. Познакомиться с новыми возможностями ввода-вывода, функции **cin** и **cout**

(смотри файл «ввод-вывод на консоль»)

1. Освоить русификацию текстов при вводе-выводе информации.

**Лабораторное задание**

1. Повторить теоретическую часть лабораторной работы №1:

* Циклы while, do… while, for
* Операторы break, continue и goto.

1. Изучить теоретическую часть лабораторной работы №2 :

* Определение и инициализация массивов и матриц.
* Доступ к элементу массива с использованием индексов.
* Особенности работы с многомерными массивами (матрицами)

1. Изучить примеры программирования и выполнить их на компьютере.
2. Выполнить задание (номер варианта соответствует номеру компьютера).

* Разработать и записать алгоритм решения
* Написать программу, реализующую разработанный алгоритм, используя, операторы for, while и do…while, доступ к элементу массива по его индексу.

**Теоретическая часть**

*Про массив можно сказать, что это набор однотипных безымянных переменных, для доступа к которым используются их номера, называемые индексами.* Впрочем, часто массив называют вектором (одномерный массив) или матрицей (двумерный массив). Создание одномерного массива похоже на создание переменной, только после идентификатора в квадратных скобках указывается число элементов - это размерность массива.

*Следующее объявление создаст массив из* ***10*** *элементов типа* ***double***

double M[10];

*Индексация массива начинается с нуля,* то есть первый элемент массива имеет индекс **0**, а последний – индекс **9**

*Элемента с индексом 10 не существует!*

После создания массив **М** будет содержать непредсказуемые значения («информационный мусор»), то есть значения его элементов не определены. Поэтому, как правило*, перед использованием массива требуется задать начальные значения его элементам, инициализировать массив*.

Инициализировать массив можно при его создании или же позже. В следующем примере создается массив из пяти элементов типа double, в который последовательно записываются числа: **0.0, 10.0, -1.0, 3.4, 0.75**

double М[5] = { 0.0, 10.0, -1.0, 3.4, 0.75 };

Список значений для инициализации может быть короче массива,

double М[5] = { 0.0, 10.0};

В данном примере создан массив из пяти элементов, при этом первые два получили начальные значения, а остальные остались неинициализированными.

Создать массив можно на основании списка инициализации, в этом случае его размер вычисляется автоматически. Обратите внимание на то, что размер массива не указан, только в этом случае он вычисляется автоматически.

/\* создается и инициализируется массив из трех элементов \*/

double М[] = { 0.0, 10.0, -1.0};

**Доступ к элементам массива с использованием индекса.**

После того, как массив создан, с ним можно работать, то есть обращаться к его элементам (считывать и записывать данные).

*Для обращения к элементу массива служат «квадратные скобки», которые являются оператором доступа к элементу массива по индексу.*

Пример: Создать массив из 3-х элементов и напечатать второй элемент.

…

int М[3];

printf("%d\n", М[1] );

Оператор **printf()** напечатает элемент массива **М** с индексом **1**, то есть второй элемент массива.

*Создавая массив из* ***n*** *элементов, нужно учитывать, что значения его индексов будут лежать в диапазоне от* ***0*** *до* ***n-1***

Например, при инициализации массива из **5** элементов

double М[] = { 0.0, 10.0, -1.0, 3.4, 0.75 };

последним элементом массива будет **М[4]**, со значением равным **0.75**, а при попытке обратиться к элементу **М[5]** мы выйдем за границу массива **М**

Компилятор не контролирует ошибки обращения к несуществующему элементу массива, то есть ошибки на этапе компиляции не возникнет, но при запуске такой программы реакция компьютера непредсказуема:

* возможно, произойдет ошибка времени выполнения,
* или программа выдаст неверные результаты,
* в худшем случае – компьютер зависнет и его придется перезагрузить.

*Для обработки массива необходимо в цикле последовательно обратиться к каждому элементу массива.*

С элементом массива можно работать так же, как с обычной переменной соответствующего типа, например, использовать в выражении или передавать значение элемента массива в функцию:

int x=2, y;

int M[3]; // выделить память под массив из 4-х элементов типа int

M[2] = 5; // записать число 5 в третий элемент массива M

y = x + M[2]; // y = 7

Cледующий фрагмент программы подсчитывает сумму элементов массива:

…

int i;

double М[] = { 0.0, 10.0, -1.0, 3.4, 0.75 };

double sum;

for (i=0, sum=0 ; i<5 ; i++)

sum=sum+M[i]; // или sum += M[i]

**Форматирование массива при выводе**.

Вывод массива на экран должен быть отформатирован, для наглядности его следует печатать в виде матрицы. В этом случае на экране уместится достаточно большой объём информации, и она будет представлена в удобном для восприятия виде.

Рассмотрим печать массива, например, в три колонки, для этого необходимо разделить элементы символом «табуляции», а после каждого третьего элемента «перевести строку».

Запишем это решение в виде «словесного алгоритма»:

1. Безусловно печатаем элемент массива;
2. Анализируем индекс:
   * + - если он кратен **col** (числу колонок), то выводится символ «перевод строки»;
       - если нет, то выводится символ «табуляции».

Этот алгоритм был бы верен, если бы не «нумерация с нуля», в самом деле, если ориентироваться по индексам, то нужно переводить строку не после **3,6,9…** индекса, а после **2,5,8….** индекса. Чтобы компенсировать это неудобство, при анализе индекса приходится «смещать нумерацию» на единицу, таким образом, в проверке участвует не сам индекс **i**, а выражение **(i+1)**

#include "stdafx.h"

#include <iostream>

#include <clocale>

#define col 5 // число колонок при выводе

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{int dig[]={5,3,2,4,6,7,11,17,0,13};

int i;

setlocale (LC\_ALL,"Russian");

cout<<"Распечатать массив в несколько колонок ("<<col<<")\n";

for (i=0; i<=9; i++)

{cout<<dig[i];

if ((i+1)%col) cout<<"\t";

else cout<<"\n";

}

cout<<endl;

system ("pause");

return 0;

}

В примере массив **dig[]** задан списком инициализации, а число колонок при выводе массива – именованной константой **col**. Использование именованных констант для обозначения различных параметров в программе настоятельно рекомендуется, так как в этом случае очень легко вносить изменения в программу. Например, в нашем примере изменить количество колонок очень просто, изменения вносятся в одном месте программы – в инструкции #define. Выполните пример на компьютере с разными значениями константы col, и Вы сразу почувствуете преимущество использования именованных констант в сравнении с обычными числовыми константами.

Разберем инструкцию if ((i+1)%col)**,** - это условный оператор и в круглых скобках должно стоять проверяемое условие, в нашем же случае вычисляется выражение ((i+1)%col, но условие проверки отсутствует! По соглашению языка С в этом случае подразумевается условие **!=0**  
(что означает **TRUE)**, поэтому условный оператор, приведенный в примере эквивалентен следующему**:** if ((i+1)%col !=0)

Результатом операции **%** («деление по модулю») является остаток от деления первого операнда на второй. Остаток равен нулю, если первый операнд (i+1) кратен второму(col).

В данном примере (и в дальнейшем) будем использовать средства ввода-вывода библиотеки **iostream**: cout – consol output (консольный вывод)

cin – consol input (консольный ввод)

(теорию смотрите в файле «ввод-вывод данных на консоль(iostream)»)

**Использование датчика случайных чисел при инициализации массива.**

При инициализации массивов часто используют функцию, генерирующую случайные числа, или как её чаще называют – «датчик случайных чисел» - функция **rand()** из библиотеки **windows.** Датчик случайных чисел позволяет автоматизировать процесс заполнения массива, что немаловажно при отладке программы, когда программа еще не до конца работает, и в целях эксперимента массив необходимо заполнять многократно.

**Пример:** Инициализировать массив с помощью датчика случайных чисел и вывести его на печать в n колонок, число колонок ввести с клавиатуры.

#include "stdafx.h"

#include <iostream>

#include <clocale>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale (LC\_ALL,"Russian");

const int N=100; // размер массива

int dig[N],col;

int i;

// заполнение массива с помощью датчика случайных чисел

for (i=0; i<N; i++)

dig[i]=rand();

cout<< "Введите число колонок (от 2 до 9): ";

cin>>col;

for (i=0; i<N; i++)

{cout<<dig[i];

if ((i+1)%col) cout<<"\t";

else cout<<"\n";

}

cout<<"\n";

system ("pause");

return 0;

}

В примере размер массива задан с помощью именованной константы **N**, при этом использован метод её определения, с помощью ключевого слова const.

Функция **rand()** при каждом обращении возвращает случайное число, которое и заносится в очередной элемент массива. После выхода из цикла весь массив оказывается инициализированным. Запустите программу и Вы увидите, что **rand()** генерирует целые числа с количеством цифр от **1** до **5**, такие большие цифры не всегда удобно использовать при решении задач. Поэтому, как правило, случайное число перед занесением в массив обрабатывают, согласно требованиям задачи.

Например, можно уменьшить количество цифр в числе, выполнив операцию «деление по модулю». Замените инструкцию

**dig[i]=rand()** на **dig[i]=rand()%100** и Вы увидите, что генерируемые числа находятся в диапазоне от **0** до **99**, так как именно в этом диапазоне находятся возможные «остатки от деления» произвольного числа на **100**.

Есть еще одна проблема, которую необходимо решить при использовании датчика случайных чисел – это генерация различных последовательностей при многократных запусках программы. Если Вы запустите приведенный пример несколько раз, то увидите, что последовательность случайных чисел не меняется, а это обстоятельство не дает возможности проверить алгоритм программы при различных данных.

Эту проблему можно разрешить с помощью функции **srand()** из библиотеки **stdlib**.

Функции **srand()** устанавливает стартовую точку для последовательности, генерируемой функцией **rand().** Но для того, чтобы эта начальная точка, а значит и вся генерируемая последовательность, менялась, аргумент **srand()** также должен меняться. Наиболее распространенный метод задания изменяемого аргумента заключается в использовании времени трансляции программы, которое возвращает функция **time().** В самом деле, каждый момент времени уникален, то есть гарантированно отличается от другого момента времени.

И в заключении, разберем генерацию случайной последовательности дробных чисел. Инструкция **dig[i]=rand()%100/3.5** будет генерировать дробные числа.

Внесем в программу необходимые изменения, касающиеся смены типа данных в массиве и форматирования вывода чисел:

1. В объявлении массива укажем **double dig[N];**

Запустив программу после этих изменений, обнаруживаем, что стройность вывода столбцов массива нарушилась, к тому же после запятой выводится слишком много знаков. Эти неожиданные проблемы легко снимаются форматированием выводимых чисел

1. При печати необходимо задать формат вывода самого числа, для это необходимо сделать следующее:

* Подключить библиотеку **iomanip**

#include<iomanip>

* Определить формат вывода числа, например:общее число знаков 8, из них 3-после запятой

cout<<setw (8)<<setprecision(3)<<dig[i];

Выполнив эти действия, получаем ровные столбцы.

Измененная программа будет выглядеть следующим образом:

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <clocale>

#include <time.h>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale (LC\_ALL,"Russian");

time\_t k;

srand (time(&k)); // инициализировать генератор случайных чисел

const int N=100; // размер массива

int i,col;

double dig[N];

// заполнение массива с помощью датчика случайных чисел

for (i=0; i<N; i++)

dig[i]=rand()%100/3.5; // генерация дробных чисел

cout<< "Введите число колонок (от 2 до 9): ";

cin>>col;

for (i=0; i<N; i++)

{cout<<setw (8)<<setprecision(3)<<dig[i];

if ((i+1)%col) cout<<" ";

else cout<<"\n";

}

cout<<"\n";

system ("pause");

return 0;

}

**Многомерные числовые массивы.**

Язык С позволяет работать с массивами любой размерности, устройство многомерного массива имеет некоторые особенности, которые мы рассмотрим на примере двумерного массива.

Для того, чтобы создать двумерный массив, необходимо использовать два набора квадратных скобок:

/\* Двумерный массив элементов double \*/

double m[2][4];

*Двумерный массив определяется как массив массивов,* про такой массив говорят, что у него **2** элемента по **4** элемента в каждом, если же рассматривать такой массив как матрицу, то можно сказать, что она имеет **2** строки и **4** столбца. Оба индекса, как и в одномерных массивах, начинаются с нуля. То есть, в массиве m есть строки **0** и **1**, каждая из которых состоит из элементов **0..3**.

Многомерные массивы также можно явно инициализировать при создании, заключая списки значений для строк в дополнительный набор фигурных скобок:

double m[2][4] = { { 1, 2, 3, 4 },

{-1, 1, 0, 0 }

};

Как и в случае одномерных массивов, вы можете задавать не все значения, а только часть. Например, написав

double m[2][4] = { { 1, 2 } };

здесь инициализируются только первые два элемента нулевой строки матрицы.

Размерность многомерного массива также как и одномерного, можно вычислить автоматически. Но в отличие от одномерного массива, где можно не указывать размерность, в многомерных массивах можно не указывать только первую размерность, вторая же (и прочие, если их больше двух) должна быть указана. Например, такая запись

double m[][4] = { { 1, 2, 3, 4 }, {-1, 1, 0, 0 } };

приведет к созданию матрицы **2x4**.

Работать с элементами многомерного массива можно, как с обычными переменными.

**Пример:** Программа для работы с квадратной матрицей

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <clocale>

#include <time.h>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale (LC\_ALL,"Russian");

time\_t k;

srand (time(&k)); // инициализировать генератор случайных чисел

const int N=5;

double dig[N][N]; // квадратная матрица

int i,j,col=5;

// заполнение матрицы с помощью датчика случайных чисел

for (i=0; i<N; i++)

for (j=0; j<N; j++) dig[i][j]=rand()%100/3.5;

// печать матрицы

for (i=0; i<N; i++)

{ cout<<endl;

for (j=0; j<N; j++) cout<<setw (8)<<setprecision(3)<<dig[i][j]<<' ';

}

cout<<endl;

system ("pause");

return 0;

}

**Примеры программирования.**

**Пример 1:** Программа вводит с клавиатуры одномерный массив из 5 целых чисел, после чего выводит количество ненулевых элементов. Ввод каждого элемента оформлен «с приглашением» (выводится подсказка с номером элемента).

#include "stdafx.h"

#include <iostream>

#define SIZE 5 // размер массива

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a[SIZE]; //массив

int n = 0; // кол-во ненулевых эл-тов

int i; // индекс

setlocale(0,"Russian"); // русификкация

cout<<"\nВведите массив из "<<SIZE<<" целых чисел";

cout<<"\nПосле ввода каждого числа нажимайте <Enter>\n";

cout<<"===========================================\n";

for (i = 0; i < SIZE; i++)

{

cout<<"a["<<i+1<<"] ->";

cin>>a[i];

if (a[i] != 0) n++;

}

cout<<"===============================\n";

cout<<"В массиве "<<n<<" ненулевых элемента.\n";

system ("pause");

return 0;

}

**Пример 2:** Программа проверяет, находится ли введенное с клавиатуры число в массиве. Массив также должен вводиться с клавиатуры.

#include "stdafx.h"

#include <iostream>

#define SIZE 5 // размер массива

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{int a[SIZE]; //массив

int obr=1; // образец для поиска

bool found; // признак совпадения с образцом

int i; // индекс

setlocale(0,"Russian");

while (obr != 999) { // зацикливание программы, 999- выход

cout<<"\nВведите массив из "<<SIZE<<" целых чисел"<<endl;

cout<<"===========================================\n";

for (i = 0; i < SIZE; i++)

{

cout<<"a["<<i+1<<"] ->";

cin>>a[i];

}

cout<<"==============================="<<endl;

cout<<"Введите образец для поиска ->";

cin>>obr;

// поиск простым перебором

found = false;

i = 0; // проверяем с первого элемента массива

do

{

if (a[i] == obr ) found = true; // совпадение с образцом

else i++; // переход к следующему элементу

}

while (!found && i<SIZE);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

if ( found )

cout<<"Совпадение с элементом номер "<<i+1<<endl;

else

cout<<"Совпадений с образцом нет"<<endl;

};

system ("pause");

return 0;

}

**Пример 3:** Определить максимальное число в массиве. Массив задать с помощью датчика случайных чисел.

#include "stdafx.h"

#include <iostream>

#include <time.h>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

const int N=30; // размер массива

int dig[N],col=5,max;

int i;

char s;

setlocale(0,"Russian");

srand (time(0)); // инициализировать генератор случайных чисел

cout<<"Вычисление максимального элемента массива\n";

do

{

// заполнение массива с помощью датчика случайных чисел

for (i=0; i<=N; i++)

dig[i]=rand()%100; //генерация 2-х значных чисел

// печать массива в col колонок

for (i=0; i<=N; i++)

{

cout<<dig[i];

if ((i+1)%col) cout<<"\t";

else cout<<"\n";

}

cout<<"\n";

// поиск максимума

max=dig[0]; // начальное значение для максимума

for (i=0; i<N; i++)

if(dig[i] > max) max=dig[i];

printf("max=%d\n",max);

cout<<"Продолжить или закончить? (y/n) : ";

cin>>s;

cout<<"======================================="<<endl;

} while (s=='y');

system ("pause");

return 0;

}

**Пример 4:** Программа для работы с квадратной матрицей, суммирование столбцов

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <clocale>

#include <time.h>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale (LC\_ALL,"Russian");

time\_t k;

srand (time(&k));

const int N=5;

double dig[N][N],s;

int i,j,col=5;

// заполнение матрицы с помощью датчика случайных чисел

for (i=0; i<N; i++)

for (j=0; j<N; j++) dig[i][j]=rand()%100/3.5;

// печать матрицы

for (i=0; i<N; i++)

{cout<<endl;

for (j=0; j<N; j++)

cout<<setw (8)<<setprecision(3)<<dig[i][j]<<' ';

}

cout<<endl;

// работа со столбцами (вычисление и печать результата)

cout<<"----------------------------------------------"<<endl;

for (i=0; i<N; i++)

{for (j=0, s=0; j<N; j++) s += dig[j][i];

cout<<setw (8)<<setprecision(3)<<s<<' ';

}

cout<<endl<<"---------- сумма столбцов --------------------"<<endl;

cout<<endl;

system ("pause");

return 0;

}

**Вопросы.**

1. Дайте общее определение массива данных.

- одномерного

- многомерного

1. Может ли массив содержать переменные разных типов?
2. Приведите пример объявления массива.
3. Что такое индекс массива?
4. Приведите пример обращения к элементу массива.
5. Можно ли инициализировать массив при его объявлении?

**Общие требования.**

1. Генератор случайных чисел должен выдавать различные последовательности при многократном запуске программы.
2. Предусмотреть возможность быстрого изменения размерности массива, для чего в тексте программы везде, где это необходимо использовать именованную константу в качестве размерности массива.
3. Работа программы должна быть циклической подобно примерам №2-№4

**Требования к оформлению ввода-вывода.**

1. При оформлении ввода-вывода данных вид экрана должен быть подобен примерам :

* на экран выводится тема задания (кратко);
* ввод данных и результат вычислений выводить с комментариями;
* выделять области ввода и вывода информации с помощью строк-разделителей.

1. Массивы на экран выводить в несколько столбцов, столбцы выравнивать по правому краю.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Вариант задания** | | | | |
| 1,16 | В массиве Х(N) найти сумму элементов, расположенных после  максимального элемента. | | | | |
|  | | | | Дана целочисленная матрица 8x8. Найти сумму элементов в заштрихованной области. Сформировать одномерный массив, состоящий из сумм элементов нечетных столбцов. |
| 2,17 | Упорядочить массив Х(N) по возрастанию, найти сумму положительных  элементов массива. | | | | |
|  | | | | Дана целочисленная матрица 8x12, содержащая как положительные, так и отрицательные элементы. Найти максимум в верхней в заштрихованной области и минимум в нижней. Сформировать одномерный массив, состоящий из сумм элементов четных строк. |
| 3,18 | В массиве Х(N) найти количество элементов, расположенных между максимальным и минимальным элементами. | | | | |
|  | Дана целочисленная матрица 10x10. Найти сумму элементов в заштрихованной области. Сформировать одномерный массив, содержащий количество отрицательных элементов нечетных столбцов. | | | |
| 4,19 | В массиве Х(N) найти S1 – сумму положительных элементов массива в  первой половине массива и S2 – сумму отрицательных элементов массива во  второй половине массива. | | | | |
|  | | | Дана целочисленная матрица 8x8. Найти сумму элементов в заштрихованной области. Сформировать одномерный массив, содержащий средние арифметические значения строк. | |
| 5,20 | Упорядочить массив Х(N) по возрастанию модулей элементов, найти  сумму отрицательных элементов массива. | | | | |
|  | | | Дана целочисленная матрица 10x12. Найти сумму элементов в заштрихованной области. Сформировать одномерный массив, состоящий из максимальных элементов четных столбцов. | |
| 6,21 | В целочисленном массиве Х(N) найти max – максимальный элемент среди  четных элементов массива и min – минимальный элемент среди всех элементов массива. | | | | |
|  | | Дана целочисленная матрица 6x9. Найти сумму элементов в заштрихованной области. Сформировать одномерный массив, содержащий количество нулевых элементов четных столбцов. | | |
| 7,22 | В массиве Х(N) найти S1 – сумму положительных элементов массива в  первой половине массива и S2 – сумму отрицательных элементов массива во  второй половине массива. | | | | |
|  | Дана целочисленная матрица 12x12, содержащая как положительные, так и отрицательные элементы. Найти максимальный по модулю элемент в заштрихованной области. Сформировать одномерный массив, содержащий средние арифметические значения каждого столбца. | | | |
| 8,23 | В целочисленном массиве Х(N) найти количество элементов, значения которых меньше среднего арифметического элементов массива. Найти номер минимального  положительного элемента. | | | | |
|  | | | Дана целочисленная матрица 8x12, найти максимум в верхней заштрихованной области и минимум в нижней. Сформировать одномерный массив, содержащий количество нулей в нечетных строках. | |
| 9,24 | В целочисленном массиве Х(N) среди четных значений найти максимальный элемент, среди нечетных подсчитать количество отрицательных. | | | | |
|  | | | Дана целочисленная матрица 10x10, содержащая как положительные, так и отрицательные элементы. Найти количество нулей в верхней заштрихованной области и количество положительных элементов в нижней. Сформировать одномерный массив, содержащий минимумы (по модулю) каждого столбца. | |
| 10,25 | В массиве Х(N) найти S1 – сумму положительных элементов массива,  расположенных после максимального элемента и S2 – сумму отрицательных  элементов массива, расположенных до максимального элемента. | | | | |
|  | | | Дана целочисленная матрица 10x12, содержащая как положительные, так и отрицательные элементы. Найти количество положительных элементов в заштрихованной области. Сформировать одномерный массив, содержащий количество отрицательных элементов в четных столбцах. | |
| 11,26 | В массиве Х(N) упорядочить элементы по убыванию, расположенные между максимальным и минимальным элементами. | | | | |
|  | | | Дана целочисленная матрица 8x8, содержащая как положительные, так и отрицательные элементы. Найти количество положительных четных элементов в заштрихованной области. Сформировать одномерный массив, содержащий максимальные значения четных строк. | |

|  |  |  |  |
| --- | --- | --- | --- |
| 12,27 | Упорядочить массив Х(N) по убыванию, найти среднее арифметическое значение элементов массива, стоящих на четных местах. | | |
|  | | Дана целочисленная матрица 10x10,найти минимум в левой заштрихованной области и максимум в правой. Сформировать одномерный массив, содержащий количество нулей в нечетных столбцах. |
| 13,28 | В целочисленном массиве Х(N) найти max1 – максимальный элемент  среди четных элементов массива и max2 – максимальный элемент среди  нечетных элементов массива. | | |
|  | | Дана целочисленная матрица 10x12, найти среднее арифметическое заштрихованной области. Сформировать одномерный массив, содержащий среднее арифметическое каждого столбца. |
| 14,29 | В массиве Х(N) найти сумму S1 элементов, расположенных до  максимального элемента, и сумму S2 элементов, расположенных после  максимального элемента. Вывести сообщение, какая из двух сумм больше. | | |
|  | | Дана целочисленная матрица 8x12, содержащая как положительные, так и отрицательные элементы. Найти среднее арифметическое отрицательных элементов в заштрихованной области. Сформировать одномерный массив, содержащий минимальные по модулю значения нечетных столбцов. |
| 15 | В массиве Х(N) найти k1 – количество элементов массива, расположенных  после минимального элемента и k2 – количество элементов массива,  расположенных после максимального элемента. | | |
|  | Дана целочисленная матрица 9x9. Найти количество нулей в заштрихованной области. Сформировать одномерный массив, содержащий суммы четных строк. | |