Лабораторная работа №6
Матрицы и функции

Цель работы: научиться обрабатывать двумерные массивы с использованием функций.

Теоретические сведения

Массивы

Описание массива производится с помощью обычного оператора описания, при этом за именем массива в квадратных скобках должна быть записана целая положительная константа или константное выражение, равное размеру этого массива, то есть максимально возможному числу элементов. ***Например:***

```
int a[100], ab[2*40];
double c[200], speed[100];
char name[20];
```

Имя массива без квадратных скобок за ним имеет значение, равное адресу первого элемента этого массива.

Имя массива с квадратными скобками, в которых записано индексное выражение целого типа, обозначает значение соответствующего элемента массива. В языке Си нумерация элементов массива начинается с нуля, то есть для массива d из пяти элементов допустимы следующие обозначения:

d[0], d[1], d[2], d[3], d[4].

Индексированные переменные могут использоваться в любых выражениях в тех местах, где допускается применение переменных соответствующих типов.

При работе с индексированными переменными необходимо внимательно следить за тем, чтобы индексы не вышли из допустимого диапазона, определяемого описаниями массивов. Дело в том, что компилятор не проверяет факт выхода индексов за границы массива, а при ошибочном занесении данных за пределы массива может запортиться нужная информация и, скорее всего, компьютер зависнет.

Ввод-вывод массива

Язык Си не имеет встроенных средств для ввода-вывода массива целиком, поэтому массив вводят и выводят поэлементно с помощью циклов, как, например, в следующей программе:

```
#include <stdio.h>
void main(void)
{   double a[100]; int n, i;
  printf("Введите количество чисел n = ");
  scanf("%d", &n);
  if( n>(sizeof a)/sizeof(double) )
     { printf("Слишком много элементов\n"); return; }
  for(i=0; i<n; i++)
  {
    printf("a[%d] = ", i); scanf("%lf", &a[i]);
  }
  /* Операторы, обрабатывающие массив */     }
```

Инициализация массива

Инициализация - присвоение значений вместе с описанием данных. Ранее была рассмотрена инициализация простых переменных, ***например:*** int a = 5.

Для инициализации массива за его именем располагают знак присваивания и список инициализации, который представляет собой заключенные в фигурные скобки и разделенные запятыми инициализирующие значения. Ниже приведен пример инициализации массива:

int a[4]={ 15, 21, 1, 304 }; индексы элементов -> 0 1 2 3

Констант в списке инициализации должно быть не больше, чем объявленный размер массива. Если их меньше, то элементы для которых нет констант обнуляются. Для инициализируемого массива допускается вообще не указывать размер. В этом случае размер массива определяется по количеству констант, например по описанию int c[] = { 1, 15, 18, 11, 20 }; транслятор выделит 10 байт для хранения массива из 5 двухбайтовых целых чисел.

Частный случай инициализации массива - инициализация строк. Массив символов может быть проинициализирован стандартным образом:

char s[] = { 'A', 'B', 'C', 'D' };

Строка символов дополнительно должна завершаться нуль-символом.

char s[] = { 'A', 'B', 'C', 'D', '\0' };

В связи с тем, что инициализацию строк приходится организовывать довольно часто, язык Си предусматривает для этого упрощенную форму записи:

char s[] = "ABCD";

В этом случае нуль-символ автоматически дописывается в конец строки. Два последних примера инициализации строки совершенно эквивалентны.

Двумерные массивы (массивы массивов)

Элементом массива может быть в свою очередь тоже массив. Таким образом, мы приходим к понятию двумерного массива или матрицы. Описание двумерного массива строится из описания одномерного путем добавления второй размерности, например: int a[4][3];

Анализ подобного описания необходимо проводить в направлении выполнения операций [], то есть слева направо. Таким образом, переменная a является массивом из четырех элементов, что следует из первой части описания a[4]. Каждый элемент a[i] этого массива в свою очередь является массивом из трех элементов типа int, что следует из второй части описания.

Для наглядности двумерный массив можно представить в виде таблицы с числом строк, равным первому размеру массива, и числом столбцов, равным второму размеру массива, ***например:***

Массив a	Столбец 0	Столбец 1	Столбец 2
Строка 0	18	21	5
Строка 1	6	7	11
Строка 2	30	52	34
Строка 3	24	4	67

Имя двумерного массива без квадратных скобок за ним имеет значение адреса первого элемента этого массива, то есть значение адреса первой строки - одномерного массива из трех элементов. При использовании в выражениях тип имени двумерного массива преобразуется к типу адреса строки этого массива. В нашем примере тип имени массива a в выражениях будет приведен к типу адреса массива из трех элементов типа int и может использоваться во всех выражениях, где допускается использование соответствующего адреса.

Имя двумерного массива с одним индексным выражением в квадратных скобках за ним обозначает соответствующую строку двумерного массива и имеет значение адреса первого элемента этой строки. Например, в нашем случае a[2] является адресом величины типа int, а именно ячейки, в которой находится число 30, и может использоваться везде, где допускается использование адреса величины типа int.

Имя двумерного массива с двумя индексными выражениями в квадратных скобках за ним обозначает соответствующий элемент двумерного массива и имеет тот же тип. Например, в нашем примере a[2][1] является величиной типа int, а именно ячейкой, в которой находится число 52, и может использоваться везде, где допускается использование величины типа int.

В соответствии с интерпретацией описания двумерного массива (слева-направо) элементы последнего располагаются в памяти ЭВМ по строкам.

Инициализация двумерного массива также проводится по строкам, например, для того чтобы получить вышеописанный массив a, можно было бы провести следующую инициализацию

```
int a[][3] = {
               { 18, 21,  5 },
               {  6,  7, 11 },
               { 30, 52, 34 },
               { 24,  4, 67 }
             };
```

Здесь первый размер массива будет определен компилятором. Следует отметить, что второй размер массива должен быть всегда указан. Это необходимо для того, чтобы сообщить компилятору размер строки массива, без которого компилятор не может правильно разместить двумерный массив в памяти ЭВМ.

Для инициализации двумерного массива символов можно использовать упрощенный синтаксис инициализации строк:

```
char s[][17] = {
                "Строка 1",
                "Длинная строка 2",
                "Строка 3"
               }
```

Размер памяти заказанный под каждую строку в этом случае должен быть равным длине самой длинной строки с учетом нуль-символа. При этом, для части строк (строка 1 и строка 3) будет выделено излишнее количество памяти. Таким образом, хранение строк различной длины в двумерном массиве символов недостаточно эффективно с точки зрения использования памяти.

Ввод двумерного массива осуществляется поэлементно с помощью двух вложенных циклов. Следующий фрагмент программы предназначен для ввода по строкам двумерного массива элементов типа double размером n строк на m столбцов

```
for (i=0; i<n; i++)
  for (j=0; j<m; j++)
  {
    printf("a[%d][%d] = ", i, j);
    scanf ("%lf", &a[i][j]);
  }
```

Для ввода массива по столбцам достаточно поменять местами строки программы, являющиеся заголовками циклов.

Вывод такого же двумерного массива иллюстрирует следующий фрагмент:

```
for (i=0; i<n; i++)
{
  for (j=0; j<m; j++) printf ("%9.3lf   ", a[i][j]);
  printf("\n");
}
```

В данном фрагменте после вывода очередной строки массива осуществляется переход на следующую строку дисплея.

В языке Си допускается использовать не только двумерные, но и трехмерные, четырехмерные и т. д. массивы. Их использование ничем принципиально не отличается от использования двумерных массивов, однако на практике они применяются значительно реже.

Генератор случайных чисел в си

rand

Синтаксис: int i = rand();

Эта функция генерирует любое положительное число от 0 до RAND_MAX, значение которого можно найти в подключаемом файле #include<stdlib.h>, как и саму функцию rand.

srand

Но, как ни странно, значение у нас все время будет одним и тем же. Тут все дело, что случайное число генерируется, исходя из определенных параметров. Ну, так вот, для превращения функции rand в, действительно, оператор случайных чисел, нужно в начале программы использовать функцию srand, которая в качестве аргумента просит число. И по этому числу уже будет генерироваться случайное число функцией rand.

Вот пример:

```
srand(time(NULL));
chislo = rand();
```

Теперь мы добились настоящей генерации случайных чисел. И каждый запуск программы будет выдавать различные цифры.

Синтаксис регулирования диапазона чисел:

начальное значение + rand() % конечное значение

Тут объяснений не требуется. Так же можно сделать, чтобы случайные числа были отрицательными. Для этого нужно начальное значение просто сделать отрицательным:

```
int chislo = -3 + rand() %6
```

Общие сведения о функциях

Подпрограмма - именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке Си подпрограммы реализованы в виде функций. Функция принимает параметры и возвращает единственное скалярное значение.

```
Заголовок_функции
{
тело_функции
}
```

Заголовок функции имеет вид: type имя_функции ([список параметров]), type – тип возвращаемого функцией значения; список параметров - список передаваемых в функцию величин, которые отделяются запятыми, каждому параметру должен предшествовать его тип;

В случае, если вызываемые функции идут до функции main, структура программы будет такой:

```
директивы компилятора
...
Тип_результата f1(Список_переменных)
{
Операторы
}
Тип_результата f2(Список_переменных)
{
Операторы
}
...
Тип_результата fn(Список_переменных)
{
Операторы
}
int main( )
Список переменных
{
Операторы основной функции, среди которых могут операторы
вызова функций f1, f2, ..., fn
}
```

В случае, если вызываемые функции идут после функции main, структура программы будет такой (заголовки функций должны быть описаны до функции main()). Опережающие заголовки функций называют прототипами функций.

```
директивы компилятора
...
Тип_результата f1(Список_переменных);
Тип_результата f2(Список_переменных);
...
Тип_результата fn(Список_переменных);
int main(Список_переменных)
{
Операторы основной функции, среди которых могут операторы вызова функций f1, f2, ..., fn
}
Тип_результата f1(Список_переменных)
{
Операторы
}
Тип_результата f2(Список_переменных)
{
Операторы
}
...
Тип_результата fn(Список_переменных)
```

{
Операторы
}

Для того, чтобы функция вернула какое-либо значение, в ней должен быть оператор return значение;

Для вызова функции необходимо указать имя функции и в круглых скобках список передаваемых в функцию значений.

Передача параметров в функцию

Параметры, указанные в заголовке функции, называются формальными. Параметры, передаваемые в функцию, называются фактическими.

При обращении к функции фактические параметры передают свое значение формальным и больше не изменяются. Типы, количество и порядок следования формальных и фактических параметров должны совпадать. С помощью оператора return из функции возвращается единственное значение.

Вызов функции

Выполнение вызова функции происходит следующим образом:

1. Вычисляются выражения в списке выражений, и подвергаются обычным арифметическим преобразованиям. Затем, если известен прототип функции, тип полученного фактического аргумента сравнивается с типом соответствующего формального параметра. Если они не совпадают, то либо производится преобразование типов, либо формируется сообщение об ошибке. Число выражений в списке выражений должно совпадать с числом формальных параметров, если только функция не имеет переменного числа параметров. В последнем случае проверке подлежат только обязательные параметры. Если в прототипе функции указано, что ей не требуются параметры, а при вызове они указаны, формируется сообщение об ошибке.
2. Происходит присваивание значений фактических параметров соответствующим формальным параметрам.
3. Управление передается на первый оператор функции.
4. Выполнение оператора return в теле функции возвращает управление и возможно, значение в вызывающую функцию. При отсутствии оператора return управление возвращается после выполнения последнего оператора тела функции, а возвращаемое значение не определено.

Возврат значений из функций

Возврат результата работы функции в вызывающую программу в виде единственного значения можно осуществить с помощью оператора return. При этом результат возвратится как значение самой функции и должен иметь соответствующий тип.

Типы возвращаемых значений могут быть любыми, кроме массивов. Тип void означает, что функция не возвращает никакого значения. Тип void* означает, что функция возвращает указатель на произвольный тип данных.

Пример. Найти количество положительных элементов, заканчивающихся на 1, каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define M 5
#define N 7
#define A -50
#define B 50
/* инициализация элементов матрицы a случайными числами в диапазоне от A до B */
void InitA(int a[][N], int m, int n)
{ int i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
a[i][j] = A + rand() % (B - A + 1); }
/* вывод матрицы на экран */
void PrintA(int a[][N], int m, int n)
{
```

```
int i, j;
for (i = 0; i < m; i++)
{
for (j = 0; j < n; j++)
printf("%5d", a[i][j]);
printf("\n");
}
printf("\n");
}
/* вычисление значений элементов массива b */
void InitB(int a[][N], int m, int n, int *b)
{
int i, j, count;
for(i = 0; i < m; i++)
for(j = count = 0; j < n; j++)
{
if (a[i][j] > 0 && a[i][j] % 10 == 1)
count++;
b[i] = count;
}
}
/* вывод массива на экран */
void PrintB(int *a, int n)
{
int i;
for(i = 0; i < n; i++)
printf("%5d ", a[i]);
printf("\n");
}
int main( )
{
int a[M][N], b[M];
srand(time(NULL));
InitA(a, M, N);
PrintA(a, M, N);
InitB(a, M, N, b);
PrintB(b, M);
return 0;
}
```

Задания для самостоятельной работы

Задание 1. Требуется определить двумерный массив *a* целых чисел (например, размера 5×7), заполнить его случайными числами или ввести его элементы с клавиатуры. В одномерный массив *b* записать характеристики двумерного массива *a* в соответствии с вариантом задания.

Программа должна содержать следующие функции:

- инициализация элементов двумерного массива случайными числами или вводимыми с клавиатуры;
- заполнение одномерного массива в соответствии с заданием;
- вывод одномерного массива на экран;
- вывод двумерного массива на экран.

Варианты заданий

1. Найти суммы четных положительных элементов каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
2. Найти количество четных положительных элементов каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.

3. Найти суммы нечетных положительных элементов каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
4. Найти количество нечетных положительных элементов каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
5. Найти суммы положительных элементов, делящихся на 3, каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
6. Найти количество положительных элементов, делящихся на 5, каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
7. Найти суммы положительных элементов, не делящихся на 3, каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
8. Найти количество положительных элементов, не делящихся на 5, каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
9. Найти минимальные элементы каждой строки матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
10. Найти минимальные элементы каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.
11. Найти максимальные элементы каждой строки матрицы *a*(5, 7) и со-хранить их в одномерном массиве *b*.
12. Найти максимальные элементы каждого столбца матрицы *a*(5, 7) и сохранить их в одномерном массиве *b*.

Задание 2. Написать программу для решения задачи согласно варианту. Использовать пользовательские функции аналогично заданию 1 (функция ввода, вывода матрицы, решения задачи согласно варианту).

Вариант 1. Задан двумерный массив *C* из 4-х строк и 4-х столбцов (квадратная матрица). Составить программу подсчёта суммы всех отрицательных элементов и суммы элементов по главной диагонали.

Вариант 2. Задан двумерный массив *Y* из 7-и строк и 3-х столбцов. Составить программу подсчёта суммы произведений элементов строк.

Вариант 3. Задан двумерный массив *A* из 5-и строк и 2-х столбцов. Составить программу, которая формирует одномерный массив *B*, каждый элемент которого есть произведение элементов массива *A* в строке.

Вариант 4. Задан двумерный массив *B* из 4-х строк и 4-х столбцов. Составить программу, которая организует двумерный массив, элементы главной диагонали которого равны соответствующим элементам исходного массива, а остальные элементы равны нулю.

Вариант 5. Задан двумерный массив *A* из 2-х строк и 7-и столбцов. Составить программу, которая организует одномерный массив, каждый элемент которого есть сумма элементов исходного в столбце.

Вариант 6. Задан двумерный массив *Y* из 5-и строк и 5-и столбцов. Составить программу подсчёта суммы всех положительных элементов и суммы элементов по главной диагонали.

Вариант 7. Задан двумерный массив *A* из 4-х строк и 4-х столбцов. Составить программу, которая подсчитывает произведение элементов массива, лежащих вне главной диагонали.

Вариант 8. Задан двумерный массив *C* из 6-и строк и 6-и столбцов. Составить программу, которая подсчитывает сумму всех элементов массива. Затем организовать формирование нового массива *C*, в котором элементы, лежащие на главной диагонали,

равны 1, а остальные элементы равны соответствующим элементам исходного массива *C*.

Вариант 9. Задан двумерный массив *Y* из 7-и строк и 3-х столбцов. Составить программу, которая вычисляет значение суммы произведений элементов строк

Вариант 10. Задан двумерный массив *B* из 4-х строк и 5-и столбцов. Составить программу, которая организует одномерный массив, каждый элемент которого есть сумма элементов в столбце. Вычислить произведение элементов полученного массива.

Вариант 11. Задан двумерный массив *A* из 5-и строк и 4-х столбцов. Составить программу, которая вычисляет значение произведения сумм строк.

Вариант 12. Задан двумерный массив *Y* из 4-х строк и 4-х столбцов. Составить программу, которая вычисляет *S* – сумму элементов побочной диагонали и значение суммы всех элементов массива

Вариант 13. Задан двумерный массив *C* из 6-и строк и 3-х столбцов. Составить программу, которая подсчитывает сумму всех элементов массива. Затем организовать формирование нового массива *C*, в котором элементы, лежащие не на главной диагонали, равны 1, а остальные элементы равны соответствующим элементам исходного массива *C*. Вычислить произведение всех элементов нового массива.

Вариант 14. Задан двумерный массив *A* из 6-и строк и 3-х столбцов. Составить программу, которая организует одномерный массив, каждый элемент которого есть произведение элементов массива *A* в каждой строке. Затем вычислить сумму элементов полученного одномерного массива.

Вариант 15. Задан двумерный массив *C* из 3-х строк и 5-и столбцов. Составить программу, которая вычисляет произведение всех элементов массива. Затем организовать новый массив *C*, в котором значения элементов, лежащих на главной диагонали, равны 1, а остальные элементы равны квадрату соответствующих элементов исходного массива *C*.

Вариант 16. Задан двумерный массив *B* из 4-х строк и 5-и столбцов. Составить программу, которая вычисляет сумму всех элементов массива. Затем организовать новый массив *B*, в котором заменить отрицательные элементы исходного массива на 1, а значения остальных элементов оставить без изменения. Подсчитать количество замен.

Вариант 17. Задан двумерный массив *A* из 8-и строк и 3-х столбцов. Составить программу, которая подсчитывает общее число неотрицательных элементов в массиве. Затем организовать формирование нового массива *B*, в котором значения элементов исходного массива заменить на противоположные по знаку.

Вариант 18. Задан двумерный массив *B* из 6-и строк и 3-х столбцов. Составить программу, которая организует одномерный массив *C*, элементы которого равны количеству положительных элементов в строке исходного массива *B*.

Вариант 19. Задан двумерный массив *D* из 5-и строк и 5-и столбцов. Составить программу, которая организует одномерный массив, элементы которого равны элементам массива *D*, лежащим на побочной диагонали, а затем вычисляет сумму элементов полученного одномерного массива.

Вариант 20. Задан двумерный массив *B* из 4-х строк и 5-и столбцов. Составить программу, которая подсчитывает количество положительных, отрицательных и нулевых элементов в массиве *B* и организует одномерный массив из полученных значений.