Arrays

An array is a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

An array is a collection of a fixed number of components all of the same data type.

1-D Arrays

A one-dimensional array is an array in which the components are arranged in a list form. The general form for declaring a one-dimensional array is:

dataType arrayName[intExp];

in which intExp is any constant expression that evaluates to a positive integer. Also, intExp specifies the number of components in the array.

The general form (syntax) used for accessing an array component is:

arrayName[indexExp];

in which indexExp, called the index, is any expression whose value is a nonnegative integer. The index value specifies the position of the component in the array.

An array can also be initializes while it is being declared.

The general (syntax) for declaring and initialization of array in 1D is

double sales[5] = {12.25, 32.50, 16.90, 23, 45.68};

Here, sales[0] = 12.25, sales[1] = 32.50, sales[2] = 16.90, sales[3] = 23.00, and sales[4] = 45.68.When initializing arrays as they are declared, it is not necessary to specify the size of the array. The size is determined by the number of initial values in the braces. However, you must include the brackets following the array name.

The previous statement is, therefore, equivalent to:

double sales[] = {12.25, 32.50, 16.90, 23, 45.68};

//print the numbers in reverse order. #include<stdio.h>

int main()

{ int item[5]; //Declare an array item of five components int sum; int counter,Rcounter;

printf("Enter five numbers: "); sum = 0;

for (counter = 0; counter < 5; counter++) { scanf("%d", &item[counter]); }

printf("\nThe numbers in reverse order are:" ); for (Rcounter = 4; Rcounter >=0; Rcounter--) printf("%d " "" , item[Rcounter] );return 0;}

//Program to read five numbers, find their sum #include<stdio.h>

int main()

{ int item[5]; //Declare an array item of five components int sum, counter;

sum = 0;

printf("Enter five numbers: ");

for (counter = 0; counter < 5; counter++) { scanf("%d", &item[counter]);

sum = sum + item[counter]; }

printf("\nThe sum of the numbers is %d",sum ); return 0;}

1-D Arrays syntax reference

|  |  |  |  |
| --- | --- | --- | --- |
| **Counter** | **Condition** | **Item[Counter]** | **Sum+ item[Counter]** |
| **0** | True | 1 | 0+1=1 |
| **1** | True | 2 | 1+2=3 |
| **2** | True | 3 | 3+3=6 |
| **3** | True | 4 | 6+4=10 |
| **4** | True | 5 | 10+5=15 |
| **5** | False |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Count er** | **Condition** | **Item [Counter]** | **RCounter** | **Item[RCount er]** |
| **0** | True | 1 | 4 | 5 |
| **1** | True | 2 | 3 | 4 |
| **2** | True | 3 | 2 | 3 |
| **3** | True | 4 | 1 | 2 |
| **4** | True | 5 | 0 | 1 |
| **5** | False |  |  |  |

2-D Arrays

A collection of a fixed number of components arranged in rows and columns (that is, in two dimensions), wherein all components are of the same type.

The syntax for declaring a two-dimensional array is:

dataTyp arrayName[intExp1][intExp2];

Wherein intExp1 and intExp2 are constant expressions yielding positive integer values. The two expressions, intExp1 and intExp2, specify the number of rows and the number of columns, respectively, in the array.

The syntax to access a component of a two-dimensional array is:

arrayName[indexExp1][indexExp2]

Wherein indexExp1 and indexExp2 are expressions yielding nonnegative integer values. indexExp1 specifies the row position; indexExp2 specifies the column position.

The statement:

double sales[10][5];

Declares a two-dimensional array sales of 10 rows and 5 columns, in which every component is of type double. As in the case of a one-dimensional array, the rows are numbered 0. . .9 and the columns are numbered 0.. .4.

Like one-dimensional arrays, two-dimensional arrays can be initialized when they are declared. The general (syntax) for declaring and initialization of array in 2D is

int board[4][3] = {{2, 3, 1}, {15, 25, 13}, {20, 4, 7}, {11, 18, 14}};

This statement declares board to be a two-dimensional array of four rows and three columns. The components of the first row are 2, 3, and 1; the components of the second row are 15, 25, and13; the components of the third row are 20, 4, and7; and the components of the fourth row are 11, 18, and14.

2-D Arrays syntax reference

#include<stdio.h> int main() {

int item[3][3]={{1,2,3},

{4,5,6},

{7,8,9}};

int rsum=0; int csum=0; int lsum=0; int risum=0;

for(int row=0;row<3;row++){ for(int col=0;col<3;col++){

rsum+=item[row][col]; csum+=item[col][row];}

printf("\nSum of row %d = %d",row+1,rsum); printf("\nSum of column %d = %d",row+1,csum); rsum=0;

csum=0; printf("\n");}

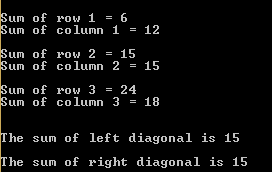
for (int i=0;i<3;i++){

lsum+=item[i][i];}

printf("\n\nThe sum of left diagonal is %d", lsum); for(int i=0; i<3; i++){

risum+=item[i][(3-1)-i];} printf("\n\nThe sum of right diagonal is %d", risum); return 0;

}



N-D Arrays

N-dimensional array is a collection of a fixed number of elements (called components) arranged in n dimensions (n>=1).

The general syntax for declaring an n-dimensional array is:

dataType arrayName[intExp1][intExp2] ... [intExpn];

where intExp1, intExp2, .. . , and intExpn are constant expressions yielding positive integer values.

The syntax to access a component of an n-dimensional array is:

arrayName[indexExp1][indexExp2] ... [indexExpn]

where indexExp1,indexExp2, . . ., and indexExpn are expressions yielding non negative integer values. indexExpi gives the position of the array component in the ith dimension.

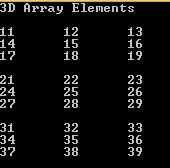
For example the statement: double carDealers[10][5][7];

declares carDealers to be a three-dimensional array. The size of the first dimension is 10, the size of the second dimension is 5, and the size of the third dimension is 7. The first dimension ranges from 0 to 9, the second dimension ranges from 0 to 4, and the third dimension ranges from 0 to

6. The base address of the array carDealers is the address of the first array component—that is, the address of carDealers[0][0][0]. The total number of components in the array carDealers is 10

\* 5 \* 7 = 350.

N-D Arrays syntax Reference



#include<stdio.h> int main()

{ int i, j, k;

int arr[3][3][3]=

{{ {11, 12, 13},

{14, 15, 16},

{17, 18, 19} },

{{21, 22, 23},

{24, 25, 26},

{27, 28, 29}},

{{31, 32, 33},

{34, 35, 36},

{37, 38, 39}},};

printf("3D Array Elements\n\n"); for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

for(k=0;k<3;k++)

{

printf("%d\t",arr[i][j][k]);

}

printf("\n");

}

printf("\n");

}

return 0;}