

m3

Arrays & strings

**Syllabus:**

**Arrays and Strings: Declaration, initialization, processing arrays and strings– two dimensional and multidimensional arrays –application of arrays. Example programs.**

**Array**

**Why we need to use an array**

if we want to process one data in a c programming,we have to declare one variable and assign data to it. if we want to process 100 data,we have to declare 100 variables and assign values one by one.

This method is not practical if we want to process large no.of variables.because this is time consuming and increase program size. So that overall efficiency of program will decrease.

Array is a perfect solution for this issue. By using an array we can process large no.of data by a single variable declaration .the only limitation is we can only process data in the same data type.

**Definition**

An array is a data structure containing a number of data values(all of which are same data type)

A data structure is a format for organizing and storing data,each data structure is designed to organize data to a suit specific purpose.

Or

An array is the collection of elements of the same data type in consecutive memory locations.

**Applications of array**

**Types of array**

* One dimensional array
* Multi dimensional array

**One dimensional array**

this is the simplest form of array. One dimensional array has only one subscript. One dimensional array is a data structure which can be used to store elements of same data type in consecutive memory locations

Which can be visualize as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

One dimensional array is used when a list of similar items need to be processed.

Eg:

* Int my[5]
* This is an array used to store integer data .
* name of this array is ’my’.
* my[0],my[1]…….my[4] are array indexes,which is necessary when we have to add elements into array and to access array elements
* 4,7,9,4,6 are elements stored in this array

My [4]

My[3]]]]

My[2]

My[1]

My[0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 7 | 9 | 4 | 6 |

* Char my[6]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **g** | **h** | **d** | **f** | **i** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **7.6** | **9.4** | **4.2** | **6.5** | **7.2** | **2.7** | **3.5** |

* Float you[7]
* Double me[3]

|  |  |  |
| --- | --- | --- |
| 5.66 | 3.55 | 2.77 |

(Array index is not mentioned in last 3 examples,and double data type is not completed)

**Declaration of one dimensional array**

In order to use an array in c programming, we have to declare it first. ie, we have to allocate some physical memory location to store elements of array.

When we declared an array,memory locations are allocated based on our required size.

When we defining an array we have to specify the following

* The type of data that we are entering to the array
* The number of elements need to be stored in array(size of array)
* Name of array

**Syntax:**

**Data type array name[size of array];**

Eg:

* Char ace[3];
* ‘Ace’ is the name of array.
* ‘Char’ is the type of data that we are entering to array.we can add character type elements to this array.
* ’3’ is the size of array (that is the max.number of elements that can be entered to an array).this array can store up to 3 elements.
* Int arm[4];
* ‘arm’ is the name of array.
* ‘int’ is the type of data that we are entering to array.we can add character type elements to this array.
* ’4’ is the size of array.this array can store up to 4 elements.

**Note**

* No.of physical memory locations allocating for an array is depends upon the size and data type of an array.

Eg:

* “Int a[2]” need 4 memory locations

‘Int’ data type is two byte.so when we have to declare an array that can hold two elements,computer must allocate 4 memory locations for it

a[0] a[1] (array index)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

2976 2977 2978 2978 (memory location)

Like this,

* “char a[2] “ need 2 physical memory locations(char data type is 1 byte)
* “Float a[2] “ need 8 physical memory locations(float data type is 4 byte)
* “double a[2]” need 16 physical memory locations(double data type is 8 byte)

**(each physical memory locations can store only one byte)**

**Note**

* We can also declare an array without specifying array size.

Eg:

**Int see[ ]={2,3,6,7,8}**

We can add any number of elements based on our requirements.array will automatically adjust its size.

**Initialization of one dimensional array**

Initialization is the process of adding elements to an array.this can be done using two methods.

* Compile time initialization
* Run time initialization

**Compile time initialization**

In compile time initialization,we are directly entering elements to an array.ie,elements are given in the program itself.this can be done using a single statement

(using this method we can declare and initialize an array at the same time.no need to do that separately)

**Syntax:**

**Data type array name[array size]={element 1,element 2,…….element n};**

Eg:

* Char haha[3]={3,6,8};
* Int hehe[4]={1,5,7,3};

Eg:

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3]={1,5,7};

printf(“%d%d%d”,a[0],a[1],a[2]);

getch();

}

Output=157

**Run time initialization**

In compile time initialization we are adding elements to array before the compiling of program takes place (ie,elements need to be entered in array is provided in program itself).

A Major draw back of compile initialization is that,we cant change values assigned to the array,once it is initialized.

Here we can use run time initialization

In this method elements are entered to array during execution of program (ie,after compiling).due to this we can change elements to array based on our needs during execution.

Eg

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3];

printf(“enter elements”);

Scanf(“%d%d%d”,&a[0],&a[1],&a[2]);

printf(“%d%d%d”,a[0],a[1],a[2]);

getch();

}

Output

We get all elements we are added to array as output.

(We can also use looping statements for run time initialization. usually used when we have to enter large no.of elements to array).

Eg

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int i,a[20];

For(i=0;i<20;i++);

{

Scanf(“%d”,&h[i]);

}

printf(“%d”,a[5]);

getch();

}

Output: This will print a[5] element entered in array

**NOTES regarding initialization**

* **The number of elements entered in array should not be higher than size of array.**

Eg:

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3]={1,5,7};

printf(“%d%d%d”,a[0],a[1],a[2]);

getch();

}

Output=157

* **If we are not entering elements to an array completely , vacant locations will fill ‘zero’ by default.**

**Eg:**

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3]={1,5};

printf(“%d%d%d”,a[0],a[1],a[2]);

getch();

}

Output=150

Bcz,we have entered only 2 elements in array,so the third element will be zero by default.

* If we enter more elements than size of array we won’t get output,it will show error

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3]={1,5,7,4};

printf(“%d%d%d”,a[0],a[1],a[2]);

getch();

}

Output=no output ,it will show error in compiling bcz,no. of elements given to array is greater than size of array

**Processing of one dimensional array**

In order to perform processes like addition ,subtraction etc using array elements ,we have to access array element.this can be done by indexing array name.this is done by placing the index of element within square bracket after name of array.

Eg:

* X=see[3]

This will assign 4th element in array ‘see’ to variable x.

Eg:

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[3]={3,5,7};

Int x,r;

x=a[2];

r=x+1;

Printf(“%d”,r);

Getch();

}

Output =8

**In the above example, We can also initialize variables along with array**

Ie,

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int x,r,a[3]={3,5,7};

x=a[2];

r=x+1;

Printf(“%d”,r);

getch();

}

**Two dimensional array**

One-dimensional arrays variable can be used to store a list of variables. There could be situations where a table of values will have to be stored.we have to use two dimensional array.

Two-dimensional arrays can also be called as a ‘matrix’

Two dimensional array has two subscripts.

Eg:-

Number[4][3]

0 1 2

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 5 |
| 5 | 6 | 7 |
| 7 | 4 | 2 |
| 8 | 4 | 1 |

number[0][0]=2

0

1

2

3

number[0][1]=3

---------------------

------------------etc

**Declaration of two dimensional array**

**Syntax:**

**Data type name of array[ i ] [ j ];**

* i and j are representing size of array,
* i is the no.of rows and j is the no.of columns
* Total number of elements = i \* j

**Initialization of two dimensional array**

Initialization is the process of adding elements into an array ,this can be done by two methods

1. Compile time initialization
2. Run time initialization

**Compile time initialization of two dimensional array**

Elements are added to the array before compiling.in compile time initialization,we are directly entering elements to an array.ie,elements need to be added is given in the program itself.this can be done using a single statement

(using this method we can declare and initialize an array at the same time.no need to do that separately)

Eg:

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[2][3]={3,5,7,2,4,1};

Printf(“%d%d%d%d%d%d”,a[0][0],a[0][1],a[0][2],a[1][0],a[1][1],a[1][2]);

Getch();

}

Output=357241

Visualization of this array

|  |  |  |
| --- | --- | --- |
| 3 | 5 | 7 |
| 2 | 4 | 1 |

**Run time initialization of two dimensional array**

Run time initialization In the runtime initialization, the values of array elements are given at the time of execution. We can use a read function such as scanf to initialize an array

Eg:

Runtime initialization using looping statements

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[2][3];

For(i=0;i<2;i++)

{

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

{

scanf(“%d”,&a[ i ][ j ]);

}

}

For(i=0;i<2;i++)

{

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

{

Printf(“%d”,a[ i ][ j ]);

}

}

getch();

}

Processing of two dimensional array

We can access array elements by using array index

Ie,

X= Array name[i][j]

This instruction will assign corresponding array element to variable x

**Programs using array**

* **Program to print elements in a 2D array**

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[2][3]={3,5,7,2,4,1};

Printf(“array elements are\n”);

For(i=0;i<2;i++)

{

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

{

Printf(“%d\n”,a[ i ][ j ]);

}

}

Getch();

}

Here we used compile time initialization and looping statement are used for printing array elements

Output=357241

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int a[2][3];

For(i=0;i<2;i++)

{

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

{

scanf(“%d”,&a[ i ][ j ]);

}

}

Printf(“array elements are\n”);

For(i=0;i<2;i++)

{

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

{

Printf(“%d\n”,a[ i ][ j ]);

}

}

getch();

}

Here we are solving same question using looping statements for both runtime initialization and printing array elements

* **Array searching(Linear search)**

This program is using to find out occurrence of an element in an array.in this program,element need to be searched is compared with element of array in linear manner(ie,first compared with 0th element then with 1st element…….up to Nth element).

if there is a match (ie,equal values) .corresponding array index will be printed and says that our elements is present in this location.

If there is no match we get an output,element not found.

#include<stdio.h>

#include<conio.h>

Void main()

{

Clrscr();

Int x,i,y,a[10];

Printf(“enter array elements\n”);

For(i=0;i<10;i++)

{

scanf(“%d”,&a[ i ]);

}

Printf(“enter element need to be searched\n”);

scanf(“%d”,&x);

y=1;

For(i=0;i<10;i++)

{

If (x==a[ i ]);

{

Printf(“%d is present in location a[%d]\n”,x,i);

Y=2;

}

}

If(y==1)

{

Printf(“%d is not present in this array\n”,x);

}

getch();

}

(here is y is used as flag)

* **Array sorting** (sorting array elements in a particular order)
* **Matrix addition**

In order to do addition of matrices,order of matrices must be same.

For this firstly we have to declare and initialize two matrices. Then we have to do addition

**#include<stdio.h>**

**#include<conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int A[2][3],B[2][3],sum[2][3];**

**Int i,j;**

**Printf(“enter elements of matrix one\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**scanf(“%d”,&A[ i ][ j ]);**

**}**

**}**

**Printf(“enter elements of matrix two\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**scanf(“%d”,&B[ i ][ j ]);**

**}**

**}**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**Sum[i][j]=A[ i ][ j ]+B[ i ][ j ];**

**}**

**}**

**Printf(“sum matrix is\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**printf(“%d”,sum[ i ][ j ]);**

**}**

**}**

**getch();**

**}**

* **Matrix subtraction**

In order to do subtraction of matrices,order of matrices must be same.

**#include<stdio.h>**

**#include<conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int A[2][3],B[2][3],diff[2][3];**

**Int i,j;**

**Printf(“enter elements of matrix one\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**scanf(“%d”,&A[ i ][ j ]);**

**}**

**}**

**Printf(“enter elements of matrix two\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**scanf(“%d”,&B[ i ][ j ]);**

**}**

**}**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**diff[i][j]=A[ i ][ j ]-B[ i ][ j ];**

**}**

**}**

**Printf(“difference matrix is\n”);**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

**{**

**printf(“%d”,diff[ i ][ j ]);**

**}**

**}**

**getch();**

**}**

* **Matrix multiplication**
* **Transpose of A Matrix**

consider matrix A of order **2X3**.this can be visualized as follows

|  |  |  |
| --- | --- | --- |
| **2** | **3** | **4** |
| **5** | **6** | **7** |

A[0][0]=2

A[0][1]=3

A[0][2]=4

A[1][0]=5

A[1][1]=6

A[1][2]=7

Transpose of this matrix can be find by interchanging rows and columns.ie, transpose of the above matrix should be in the order of **3X2**.this can be visualized as follows,T transpose matrix.

|  |  |
| --- | --- |
| 2 | 5 |
| 3 | 6 |
| 4 | 7 |

T[0][0]=2

T[0][1]=5

T[1][0]=3

T[1][1]=6

T[2][0]=4

T[2][1]=7

from the above given matrices it is clear that **A[i][j]=T[j][i]** this is our key equation to find transpose of a matrix.

**Here we are writing transpose program for a 2X3 matrix**

**#include<stdio.h>**

**#include<conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int A[2][3],T[3][2];**

**Int i,j;**

**Printf(“enter array elements\n”);**

**For(i=0;i<2;i++)**

**{**

Array is initialized using loop statement

**For(j=0;j<3;j++)**

**{**

**scanf(“%d”,&a[ i ][ j ]);**

**}**

**}**

**For(i=0;i<2;i++)**

**{**

**For(j=0;j<3;j++)**

Using loop statement Transpose is performed

**{**

**T[i][j]=A[ j ][ i ];**

**}**

**}**

**Printf(“transpose matrix”\n);**

**For(i=0;i<2;i++)**

**{**

Printing transpose matrix

**For(j=0;j<3;j++)**

**{**

**printf(“%d”, T[ i ][ j ]);**

**}**

**}**

**Getch();**

**}**

We can write another program it will support transpose without order limitations

**#include<stdio.h>**

**#include<conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int A[10][10],T[10][10];**

**Int i,j,a,b;**

**Enter(“no.of rows and columns\n”);**

**Scanf(“%d%d”,&a,&b);**

**Printf(“enter array elements\n”);**

**For(i=0;i<a;i++)**

**{**

Array is initialized using loop statement

**For(j=0;j<b;j++)**

**{**

**scanf(“%d”,&a[ i ][ j ]);**

**}**

**}**

**For(i=0;i<a;i++)**

**{**

**For(j=0;j<b;j++)**

Using loop statement Transpose is performed

**{**

**T[i][j]=A[ j ][ i ];**

**}**

**}**

**Printf(“transpose of matrix”\n);**

**For(i=0;i<a;i++)**

**{**

**For(j=0;j<b;j++)**

Printing transpose matrix

**{**

**printf(“%d”, T[ i ][ j ]);**

**}**

**}**

**Getch();**

**}**

.