**ARRAYS**

Array is a fixed size collection of consecutive memory location. All elements in the array must be of the same type (all integers, all characters …).

The advantage of an array is that; it allows the programmer to declare only a single identifier and yet obtain a large amount of space.

**ONE-DIMENSIONAL ARRAYS**

The simplest form of array is a “one-dimensional” array.

**General Form:**

data\_type identifier[size\_of\_array];

**Exp:**

int a[100]; //specifies an array of 100 integers

The smallest element of an array’s index is called its lower bound, and in C is always 0, and the highest element is called upper bound. The number of elements in the array called its range (it is size of the array).

range = upper bound – lower bound + 1

**Exp:**

In the array “a”, the lower bound is 0, the upper bound is 99 and the range is 100.

**Exp:**

|  |  |  |
| --- | --- | --- |
| **int scores[50];** | **char name[15];** | **float height[600];** |
| 4 bytes   |  |  |  | | --- | --- | --- | |  |  | scores[0] | | 50 |  | scores[1] | | memory | …  …  … |  | | allocation | |  | |  |  | scores[49] | | 1 bytes   |  |  |  | | --- | --- | --- | |  |  | name[0] | | 15 |  | name[1] | | memory | …  …  … |  | | allocation | |  | |  |  | name[14] | | 4 bytes   |  |  |  | | --- | --- | --- | |  |  | height[0] | | 600 |  | height[1] | | memory | …  …  … |  | | allocation | |  | |  |  | height[599] | |

**1-Dim Array Initialization**

float x[3]={12.5, 0.0, -3.5 };

int months[12]={31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

char grades[]={‘A’, ‘B’, ‘C’, ‘D’, ‘F’, };

**Fill Array**

int x[10], i;

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

cin>>x[i];

**Display Array Contents**

int x[10], i;

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

cout<<x[i]<<endl;

//cout<< “x[“<<i<<”]=”<<x[i]<<endl;

**TWO-DIMENSIONAL ARRAYS**

**General Form:**

data\_type identifier[size\_of\_row][size\_of\_column];

**Exp:**

int a[3][2]; // number of elements in array 3 \* 2 =6

row column

row major order:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | row 0 |  | a[0][0] | |  | a[0][1] | | row 1 |  | a[1][0] | |  | a[1][1] | | row 2 |  | a[2][0] | |  | a[2][1] | | OR  🡺 | |  |  |  | | --- | --- | --- | | 0 | | 1 | | 0 | a[0][0] | a[0][1] | | 1 | a[1][0] | a[1][1] | | 2 | a[2][0] | a[2][1] | |

**2-Dim Array Initialization**

int a[3][2]= {12, 7, 21, 0, 42, 4};

or

|  |  |
| --- | --- |
| 12 | 7 |
| 21 | 0 |
| 42 | 4 |

int a[3][2]= {{12, 7}, {21, 0}, {42, 4}}; // 42=a[2][0]

**Fill Array**

int x[10][20], i, j;

for( i=0; i<10; i++) //row

for( j=0; j<20; j++) //column

cin>>x[i][j];

**Display Array Contents**

int x[10][20], i, j;

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

for( j=0; j<20; j++)

cout<<x[i][j]<<endl;

**THREE-DIMENSIONAL ARRAYS**

int b[3][2][4];

plane row column

The array “b” contains 3\*2\*4 =24 elements.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| plane 0  plane 1  plane 2  0 1 2 3  colums  row 0  row 1 | OR  🡺 | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | | 0 |  |  |  |  |  |  |  |  |  |  |  |  | | 1 |  |  |  |  |  |  |  |  |  |  |  |  | |  | 0 | | | | 1 | | | | 2 | | | | |

**MULTI-DIMENSIONAL ARRAYS**

C also allows an arbitrary number of dimensions, for example, a six-dimensional array may be declared as

int c[7][15][3][5][8][2];

The array “c” contains 7\*15\*3\*5\*8\*2 = 25,200 elements.

**Arrays as Function Parameter**

When an array is passed as an argument, the starting address of the array is passed; the elements of the array are not copied into the array parameter.

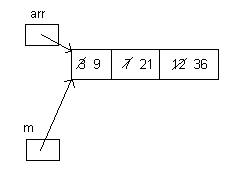
**Exp:**

#include<iostream>

using namespace std;

void mulByThree(int []); //**OR** void mulByThree(int \*);

int main(void)



{

argument

int arr[]={3, 7, 12};

mulByThree(arr);

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

cout<<arr[i]<<endl; //9 21 36

return 0;

parameter

}

void mulByThree(int m[]) //**OR** void mulByThree(int \*m)

{

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

m[i]=m[i]\*3; //**OR** \*(m+i)=\*(m+i)\*3;

}

**Note That:**

Array arguments are passed by reference rather then value.

**Exp:**

#include<iostream>

#define SIZE 10

using namespace std;

float avg(float []);

int main(void)

{

float a[SIZE];

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

cin>>a[i];

cout<<"Average is"<<avg(a)<<endl;

return 0;

}

float avg(float a[])

{

int i;

float sum=0.0;

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

sum+=a[i];

return(sum/SIZE);

}

**Exp:**

#include<iostream>

using namespace std;

int sum(int [], int);

void fillArray(int [], int);

int main(void)

{

int a[]={22, 44, 66};

int size=sizeof(a)/sizeof(int);

*// 3 = 12 / 4*

for(int i=0; i<size; i++)

cout<<"\ta["<<i<<"]="<<a[i]<<endl;

cout<<"Sum of elements in a = "<<sum(a, size)<<endl;

fillArray(a, size);

return 0;

}

int sum(int b[], int n)

{

int sum=0;

for(int i=0; i<n; i++)

sum+=b[i];

return sum;

}

void fillArray(int a[], int n) //fill one dimensional array

{

for(int i=0; i<n; i++)

{

cout<<"Enter "<<i<<". number"<<endl;

cin>>a[i];

}

}

**Exercise:**

**Write “fillArray” and “sum” functions for two dimensional array.**

**Operations on Arrays**

* + Copy
  + Sort
  + Search
  + Insert
  + Delete

**Copying**

Assume that we have following array ‘a’ and ‘b’. Lets copy 3 elements from ‘a’ into ‘b’.

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

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

m

k

b

a

|  |  |  |
| --- | --- | --- |
| for(i=2; i<=4; i++)  b[i-2]=a[i]; | for(i=3; i<=5; i++)  b[i-3]=a[i]; | for(i=k; i<=m; i++)  b[i-k]=a[i]; |

**Inserting (applied for sorted array)**

Assume that you want insert 25 into array

|  |
| --- |
| 10 |
| 20 |
| 30 |
| 40 |

|  |
| --- |
| 10 |
| 20 |
| 25 |
| 30 |
| 40 |

void insert(int a[], int **\***size, int newn)

{

if(newn>a[**\***size])

{

**\***size=**\***size+1;

a[\*size]=newn;

}

else

{

**\***size=**\***size+1;

for(int i=**\***size**-1**; i>=0;i--)

{

a[i+1]=a[i];

if(i==0)

a[i]=newn;

else if(newn>=a[i-1])

{

a[i]=newn;

break;

}

}

}

}

**Main:**

int main(void)

{

int a[10]={10,20,30,40};

int size=3, newNumber;

cout<<"Enter new number"<<endl;

cin>>newNumber;

insert(a, &size, newNumber);

return 0;

}

**Deleting (applied for sorted array)**

Assume that you want delete 20 from array

|  |
| --- |
| 10 |
| 20 |
| 30 |
| 40 |

|  |
| --- |
| 10 |
| 30 |
| 40 |

void deletion(int a[], int \*size, int oldn)

{

for(int i=0; i<=\*size; i++)

{

if(oldn==a[i])

{

for(int j=i; j<=\*size; j++)

a[j]=a[j+1];

i--; //for same numbers

\*size=\*size-1;

}

}

//a[\*size+1]=0; //automatically occurs

}