Computer Programming

Contents

- Arrays *(LL 02)*
- Types of Arrays *(LL 02)*
- One-Dimensional Array *(LL 04)*
- Declaring One-Dimensional Array *(LL 04)*
- Initializing One-Dimensional Array *(LL 04)*
- Accessing elements of One-Dimensional Array *(LL 04)*
- Inputting elements of One-Dimensional Array *(LL 04)*
- Outputting elements of One-Dimensional Array *(LL 04)*
- Summing all the elements of One-Dimensional Array *(LL 04)*
- Averaging all the elements of One-Dimensional Array *(LL 04)*

LL 02 = Learning Level 02 – Comprehension, LL 04 = Learning Level 04 – Analysis

Contents

- Multi-Dimensional Array *(LL 04)*
- Two-Dimensional Array *(LL 04)*
- Declaring Two-Dimensional Array *(LL 04)*
- Initializing Two-Dimensional Array *(LL 04)*
- Accessing elements of Two-Dimensional Array *(LL 04)*
- Inputting elements of Two-Dimensional Array *(LL 04)*
- Outputting elements of Two-Dimensional Array *(LL 04)*
- Summing all the elements of Two-Dimensional Array *(LL 04)*
- Averaging all the elements of Two-Dimensional Array *(LL 04)*

LL 02 = Learning Level 02 – Comprehension, LL 04 = Learning Level 04 – Analysis

Contents

- Searching array element using Linear Search *(LL 04)*
- Sorting array element using Bubble Sort *(LL 04)*
- Program Examples *(LL 04)*

LL 02 = Learning Level 02 – Comprehension,

LL 04 = Learning Level 04 – Analysis

Arrays

- An array is the collection homogeneous (same) data items.
- It is a collection of a fixed number of components all of the same data type.
- An array is a consecutive group of memory locations that all have the same type.
- An array is a data structure that can hold multiple values at a time unlike simple variables who can only store single value at a time.
- In an array all the elements have one or more indices through which they are accessed.

Arrays

In order to create an array we need to define the followings for an array:

- **Name**
- **Data Type**
- **Size**
- **Elements**

Arrays

- The name of the array is identifier, which follows all the rules of identifier in C++.
- The data type of array specifies the type of the data every element of its will have.
- The size specifies the total number of elements, the array can store at maximum.
- The elements are the actual data items (values) that array will hold.

Types of Arrays

- There are two types of arrays:

One-Dimensional Array

Multi-Dimensional Array

One-Dimensional Array

- One-dimensional array is just the list of items of same data type.
- All the items are stored in memory consecutively.

50
10
0
-8
73
56
74
48

Declaring One-Dimensional Array

- In order to declare one dimensional array, we need to specify:

- **Name**
- **Data Type**
- **Size**

Declaring One-Dimensional Array

```
int nums [10] ;
float temps [7] ;
char alphabets [50]
; double voltage
[250] ;
```

Initializing One-Dimensional Array

- In order to initialize one dimensional array, we need to specify:

- **Name**
- **Data Type**
- **Size**
- **Elements**

Initializing One-Dimensional Array

```
int nums [5] = {50,98,74,82,35};
```

nums

50
98
74
82
35

Initializing One-Dimensional Array

```
float temps [7] = {37.5,34.4,32.8,30.1,33.8,34.8,33.3};
```

temps

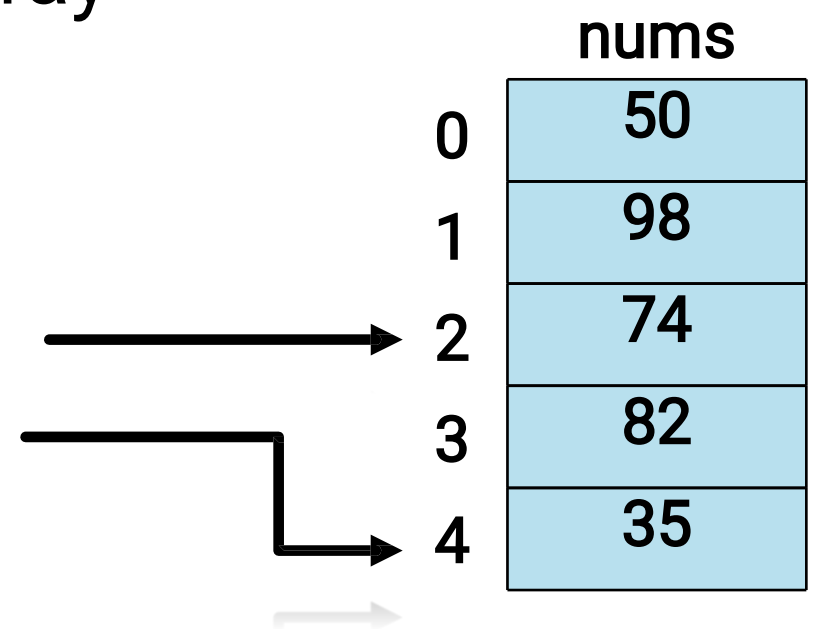
37.5
34.4
32.8
30.1
33.8
34.8
33.3

Accessing elements of One-Dimensional Array

- In one dimensional array, every element has one index number.
- The index number starts from 0.
- The index of the element is always one less than the element number. For example the index of 5th item is 4 and the index of 1st item is 0.
- For accessing any item of the array we just specify the array name along with the element index inside the square brackets.

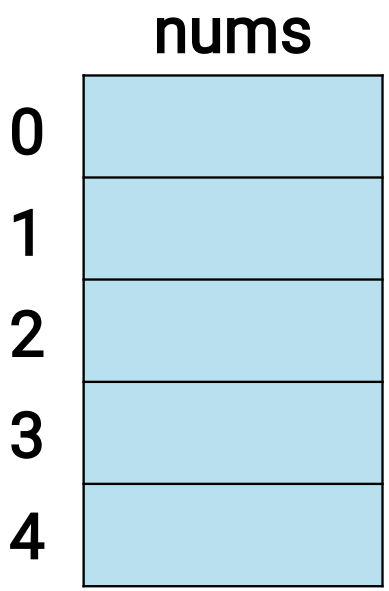
Accessing elements of One-Dimensional Array

```
int nums [5] = { 50,98,74,82,35};
cout<<nums[2] ;  // Displays 74
cout<<nums[4] ;  //Displays 35
```



Inputting elements of One-Dimensional Array

```
int nums [5];
for( int i = 0 ;    i < 5 ;    i++)
{
        cin>>nums [ i ] ;
}
```



Outputting elements of One-Dimensional Array

```
int nums [5] = { 50,98,74,82,35};
for( int i = 0 ; i < 5 ; i++)
{
        cout<<nums [ i ] << endl;
}
```

	nums
0	50
1	98
2	74
3	82
4	35

Summing all the elements of One-Dimensional Array

```
int sum = 0 ;
int nums [5] = { 50,98,74,82,35};
for( int i = 0 ; i < 5 ; i++)
{
    sum += nums [ i ] ;
}
cout<<"Sum = " << sum ; //Displays 339
```

	nums
0	50
1	98
2	74
3	82
4	35

Averaging all the elements of One-Dimensional Array

```
const int SIZE = 5 ; int sum = 0 ;
float average = 0.0 ;
int nums [5] = {50,98,74,82,35};
for( int i = 0 ;i < SIZE ;i++)
{

    sum += nums [ i ] ;
}
average = (float) ( sum / SIZE ) ;
cout<<"Average = " << average ;  //Displays 67.8
```

	nums
0	50
1	98
2	74
3	82
4	35

Multi-Dimensional Array

- Multi-dimensional array is an array having more than one dimensions.
- It require more than one index to access the items.
- Every dimension has its own index.
- A two dimensional array is the arrangement of elements in rows and columns. First index = row, second index = column.
- A three dimensional array is arrangement of data in the form of a cube. First index = cube face, second index = row, third index = column.

Two-Dimensional Array

- Two-dimensional array is the arrangement of elements in rows and columns.
- It has two indices, one for row and other for column.

nums

57	74	11
10	14	87
47	48	98

Declaring Two-Dimensional Array

- In order to declare two dimensional array, we need to specify:

- **Name**
- **Data Type**
- **Size for both the dimensions**

Declaring Two-Dimensional Array

```
int nums [3] [3] ;
float temps [5] [7];
char alphabets [2] [3] ;
```

Initializing Two-Dimensional Array

- In order to initialize two dimensional array, we need to specify:

- **Name**
- **Data Type**
- **Size for both the dimensions**
- **Elements**

Initializing Two-Dimensional Array

```
int nums [3] [3] = {{57,74,11},{10,14,87}, {47,48,98}};
```

nums

57	74	11
10	14	87
47	48	98

Initializing Two-Dimensional Array

```
float temps [3] [7] = {{34.5,33.2,38.4,36.4,37.7,34.4,38.8}, {37.4,34.1,34.1,38.1,36.5,30.4,30.8},
{38.1,38.0,30.8,37.4,38.1,33.5,34.8}};
```

temps

34.5	33.2	38.4	36.4	37.7	34.4	38.8
37.4	34.1	34.1	38.1	36.5	30.4	30.8
38.1	38.0	30.8	37.4	38.1	33.5	34.8

Initializing Two-Dimensional Array

```
char alphabets [2] [3] = {{ 'C' , 'D' , 'W' }, { '1' , '?' , 'V' }};
```

alphabets

C	D	W
1	?	V

Accessing elements of Two-Dimensional Array

- In two dimensional array, every element has two index numbers.
- One for row and other for column. Both the indices start from 0.
- The row index of the element is always one less than the row number and column index is one less than the column number of that item.
- For accessing any item of the array we just specify the array name along with the row index and column index inside the square brackets.

Accessing elements of Two-Dimensional Array

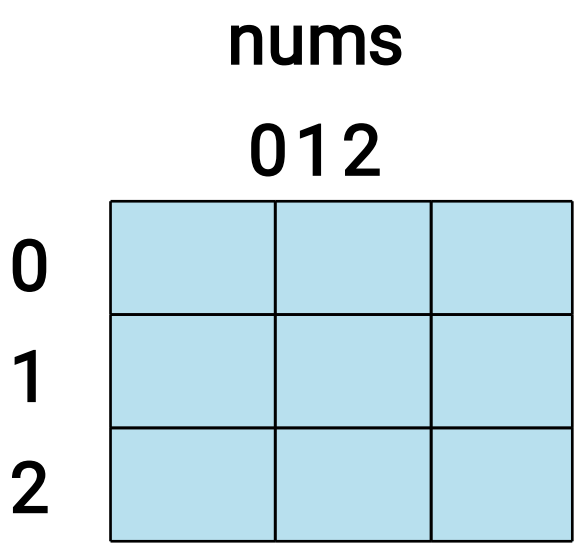
```
int nums [3] [3] = {{57,74,11},{10,14,87}, {47,48,98}};
cout<<nums [1] [2] ;   // Displays 87
cout<<nums [2] [2] ;   //Displays 98
```

nums

	0	1	2
0	57	74	11
1	10	14	87
2	47	48	98

Inputting elements of Two-Dimensional Array

```
int nums [3] [3] ; for(
int i = 0 ;  i < 3 ;          i++)
{
    for( int j = 0 ; j < 3 ;     j++)
    {
        cin>>nums [ i ] [ j ] ;
    }
}
```



Outputting elements of Two-Dimensional Array

```
int nums [3] [3] = {{57,74,11},{10,14,87}, {47,48,98}};
for( int i = 0 ;i < 3 ;  i++)
{
    for( int j = 0 ;j < 3 ;  j++)
    {
        cout<<nums [ i ] [ j ] << "  ";
    }
    cout<< endl;
}
```

nums

	0	1	2
0	57	74	11
1	10	14	87
2	47	48	98

Summing all the elements of Two-Dimensional Array

```
int sum = 0 ;
int nums [3] [3] = {{57,74,11},{10,14,87}, {47,48,98}};
for( int i = 0 ;  i < 3 ;  i++)
{
    for( int j = 0 ;  j < 3 ;  j++)
    {
        sum += nums [ i ] [ j ] ;
    }
}
cout<<"Sum = " << sum ;  //Displays 446
```

nums

	0	1	2
0	57	74	11
1	10	14	87
2	47	48	98

Averaging all the elements of Two-Dimensional Array

```
const int ROWS = 3 ;
const int COLS = 3 ;
int sum = 0 ;
float average = 0.0 ;

int nums [3] [3] = {{57,74,11},{10,14,87}, {47,48,98}};

for( int i = 0 ; i < 3 ;i++)
{
        for( int j = 0 ; j < 3 ;j++)
        {
                sum += nums [ i ] [ j ] ;
        }
}

average = (float) ( sum / (ROWS  * COLS )) ;
cout<<"Average = " << average ;  //Displays 49.55
```

nums

	0	1	2
0	57	74	11
1	10	14	87
2	47	48	98

Program Examples

Arrays

Program Example 01

Problem Statement:

Write a program in C++ that initializes two arrays A and B. The program should create, calculate and display the contents of array C as following.

A	B	C
25	87	87.25
14	11	14.11
12	10	12.10
74	81	81.74
58	67	67.58
74	94	94.74
98	74	98.74
84	82	84.82
15	15	15.15
24	87	87.24

Note that every item in array A and B is of 2 digit number.

Program Example 01

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    const int SIZE=10;

    int A[] = {25,14,12,74,58,74,98,84,15,24};
    int B[] = {87,11,10,81,67,94,74,82,15,87};
    float C[SIZE];

    for(int i=0; i<SIZE; i++)
    {
        if(A[i]>B[i])
            C[i] = A[i] + B[i] /100.0;
        else
            C[i] = B[i] + A[i] /100.0;
    }
```

Program Example 01

```
    cout<<"Matrix C:"<<endl;

    for(int j=0; j<SIZE; j++)
    {
        cout<<C[j]<<endl;
    }
}
```

Program Example 01

"D:\Google Drive\Synchronized Data\14CS\Computer Programming\Practicals\Arrays (Tasks & Sol...

```
Matrix C:
87.25
14.11
12.1
81.74
67.58
94.74
98.74
84.82
15.15
87.24

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Program Example 03

Problem Statement:

Write a program in C++ that inputs the elements of an integer array from the user and displays the number of positive, negative and zero values; sum of positive numbers and sum of negative numbers present in the array.

Numbers

85
74
-81
15
-74
0
-36
-25
54
49

Positive Numbers = 5

Negative Numbers = 4

Zeros = 1

Sum of Positive Numbers = 277

Sum of Negative Numbers =216

Program Example 03

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    const int SIZE = 10;
    int positiveCount = 0;
    int negativeCount = 0;
    int zeroCount = 0;
    int positiveSum = 0;
    int negativeSum = 0;

    int nums[SIZE];

    for(int i=0; i<SIZE; i++)
    {
        cout<<"Enter number "<<i+1<<" : ";
        cin>>nums[i];
    }
```

Program Example 03

```
for(int i=0; i<SIZE; i++)
{
    if(nums[i]<0)
    {
        negativeCount++;
        negativeSum+=nums[i];
    }
    else if(nums[i]>0)
    {
        positiveCount++;
        positiveSum+=nums[i];
    }
    else
    {
        zeroCount++;
    }

}
```

Program Example 03

```
    }

    cout<<"Positive Numbers = "<<positiveCount<<endl;
    cout<<"Negative Numbers = "<<negativeCount<<endl;
    cout<<"Zeros = "<<zeroCount<<endl;
    cout<<"Sum of Positive Numbers = "<<positiveSum<<endl;
    cout<<"Sum of Negative Numbers = "<<-negativeSum<<endl;

    getch();
    return 0;
}
```

Program Example 03

C:\Users\HP\Desktop\Program03.exe

```
Enter number 1 : 85
Enter number 2 : 74
Enter number 3 : -81
Enter number 4 : 15
Enter number 5 : -74
Enter number 6 : 0
Enter number 7 : -36
Enter number 8 : -25
Enter number 9 : 54
Enter number 10 : 49
Positive Numbers = 5
Negative Numbers = 4
Zeros = 1
Sum of Positive Numbers = 277
Sum of Negative Numbers = 216
```

Program Example 04

Problem Statement:

Write a program in C++ that performs the addition of two matrices of size 3 x 3. Program should create three matrices namely A, B and C. Matrix A and B are define below. The matrix C is the defined as C = A + B.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 3 & 1 & 9 \\ 7 & 8 & 0 \\ 5 & 3 & 6 \end{bmatrix} \quad C = A + B = \begin{bmatrix} 4 & 3 & 12 \\ 11 & 13 & 6 \\ 12 & 11 & 15 \end{bmatrix}$$

Program Example 04

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    const int ROWS = 3;
    const int COLS = 3;

    int A[ROWS][COLS] = {{1,2,3},{4,5,6},{7,8,9}};
    int B[ROWS][COLS] = {{3,1,9},{7,8,0},{5,3,6}};
    int C[ROWS][COLS];

    for(int i=0; i<ROWS; i++)
    {
        for(int j=0; j<COLS; j++)
        {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
```

Program Example 04

```
    for(int i=0; i<ROWS; i++)
    {
        for(int j=0; j<COLS; j++)
        {
            cout<<C[i][j]<<"  ";
        }

        cout<<endl;
    }

    getch();
    return 0;
}
```