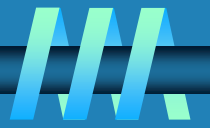
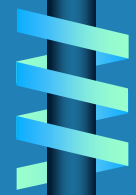
Fundamentals of Programming C++

C++ Arrays and String

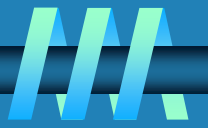
Outline



- Arrays and String
  - Array definition
  - Types of arrays
  - Array declaration
  - Array initialization
  - What are Strings?
  - Initialization of Strings
  - Assigning Values to Strings
  - Functions to Manipulate Strings

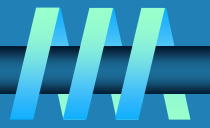


Arrays - Introduction

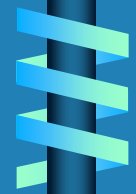


- So far we have worked primarily with primitive variables
  - One characteristic of primitive variables is that they hold one value at a time.
    - int x = 100 – can only hold one integer value at a time (100 at this time)
- Imagine that you want to hold the names of 100 people at a time.
  - What would you have to do?
    - Declare 100 variables, one for each name.
  - Better solution: Use Arrays.
    - Arrays are complex variables that can hold multiple values of the same data type.
    - Now we can declare a single array that holds all the names.

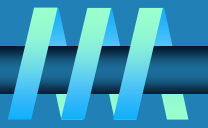
Arrays – Introduction Cont'd…



- Array: An ordered collection of values with two distinguishing characters:
  - Ordered and fixed length
  - Homogeneous. Every value in the array must be of the same type
- The individual values in an array are called elements.
- The number of elements is called the length of the array
- Each element is identified by its position number in the array, which is called index.



Arrays – Introduction Cont'd…



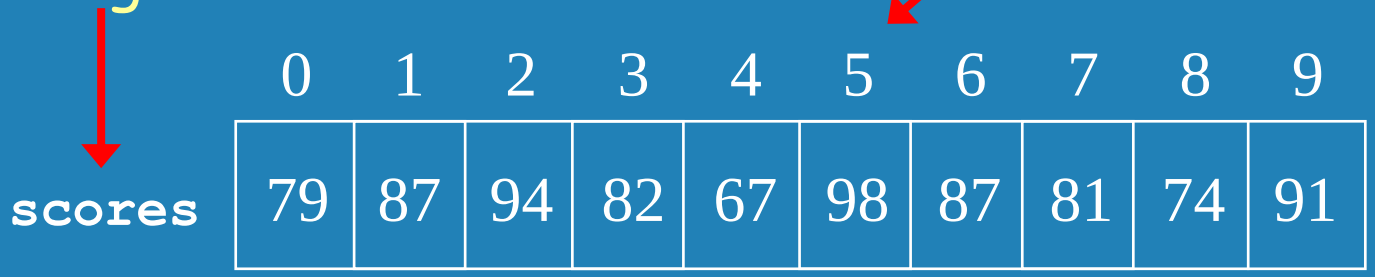
- An array stores multiple values of the same type (the *element type*)
- The element type can be a primitive type or an object reference
- Therefore, we can create an array of integers, or an array of characters, or an array of String objects, etc.

Arrays – Introduction Cont'd…

- **An *array* is an ordered list of values**

The entire array has a single name

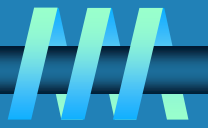
Each value has a numeric *index*



An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

Types of Arrays



1. One -Dimensional Arrays

- A one-dimensional array is a kind of linear array. It involves single sub-scripting. The [] (brackets) is used for the subscript of the array and to declare and access the elements from the array.

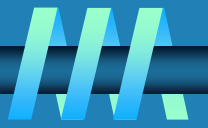
Syntax:

DataType ArrayName [size];

For example:

int a[10];

Types of Arrays Cont'd…



2. Multi -Dimensional Arrays

- In multi-dimensional arrays, we have two categories:
  - Two-Dimensional Arrays
  - Three-Dimensional Arrays

a) Two-Dimensional Arrays

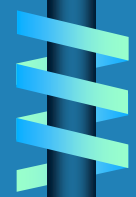
- An array involving two subscripts [] [] is known as a two-dimensional array. They are also known as the array of the array. Two-dimensional arrays are divided into rows and columns and are able to handle the data of the table.

Syntax:

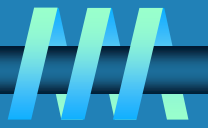
```
DataType ArrayName[row_size][column_size];
```

For Example:

```
int arr[5][5];
```



Types of Arrays Cont'd…



b) Three-Dimensional Arrays

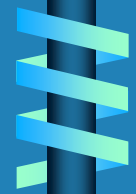
- When we require to create two or more tables of the elements to declare the array elements, then in such a situation we use three-dimensional arrays.

Syntax:

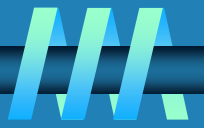
```
DataType ArrayName[size1][size2][size3];
```

For Example:

```
int a[5][5][5];
```

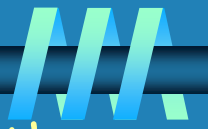


Array Declaration



- Rules for declaring a single-dimension array in C++.
  - Type: The type is the type of elements to be stored in the array, and it must be a valid C++ data type.
  - Array-Name: The array-Name is the name to be assigned to the array.
  - Array-Size: The array-Size is the number of elements to be stored in the array. It must be an integer and greater than 0.

Array Declaration Cont'd…



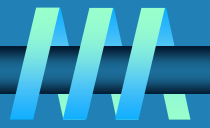
- To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array.
- Syntax

  type arrayName [arraySize];

- This is called a single-dimension array. The arraySize must be an integer constant greater than zero and type can be any valid C++ data type.
- For example, to declare a 10-element array called balance of type float, use this statement

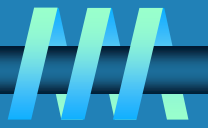
```
float balance[10];
```

Array Initialization



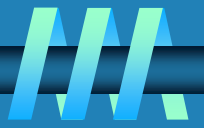
- Array initialization is the process of assigning/storing elements to an array. The initialization can be done in a single statement or one by one. Note that the first element in an array is stored at index 0, while the last element is stored at index n-1, where n is the total number of elements in the array.
- Like any other variable in C++, we can initialize arrays at the time of declaration.

Array Initialization Cont'd…



- Array Initialization by Specifying Array Size
  - int marks[7] = {5,2,9,1,1,4,7};
- Array Initialization Without Specifying Array Size
  - int marks[] = {5,2,9,1,1,4,7};
- In above declaration, compiler counts the number Of elements inside curly braces{} and determines the size of array score. Then, compiler will create an array of size 7 and initialize it with value provided between curly braces{}

Array Initialization Cont'd…



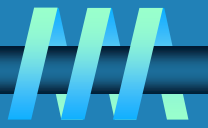
Initialization of Two Dimensional Array

- Two dimensional arrays can be initialized by specifying elements for each row inside curly braces.
  - int matrix[4][3] = {

```
{1, 2, 3}, /* Initialization of first row */
{4, 5, 6}, /* Initialization of second row */
{7, 8, 9}, /* Initialization of third row */
{10, 11, 12}, /* Initialization of fourth row */
};
```

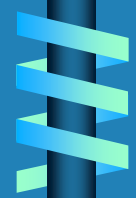
- A two dimensional matrix can be initialized without using any internal curly braces as follows:
  - int matrix[4][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

Array Initialization Cont'd…



Initialization of Three Dimensional Array

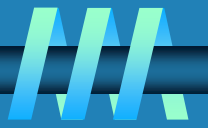
- Similarly, we can initialize any multi dimensional array.
- Similar to two dimensional array, we can also initialize 3 dimensional array as follows :
  - int rubix[2][3][2] = {4, 6, 1, 0, 6, 12, 3, 10, 0, -4, 8, 9};
- **This array *rubix* can hold a maximum of 12 elements.**
- **We can find out the total number of elements in the array simply by multiplying its dimensions:**
  - **2*3*2 = 12**



Two Dimensional Array Example

```
// C++ Program to display all elements // of an initialised two dimensional array
#include <iostream>
using namespace std;
int main()
 {
        int test[3][2] = {{2, -5}, {4, 0}, {9, 1}};
        // use of nested for loop
        // access rows of the array
        for (int i = 0; i < 3; ++i)
        {
                // access columns of the array
                for (int j = 0; j < 2; ++j)
                {
                        cout << "test[" << i << "][" << j << "] = " << test[i][j] <<
                        endl;
                }
        }
        return 0;
}
```

Accessing Array Elements



- In C++, We can access array element using array name and subscript/index written inside pair of square brackets [].
- For Example : Suppose we have an integer array of length 5 whose name is age.
  - int age[5] = {8,5,2,3,7};
- Now we can access elements of array marks using subscript followed by array name. age[0] = First element of array age = 8
  - age[1] = Second element of array age = 5
  - age[2] = Third element of array age = 2
  - age[3] = Fourth element of array age = 3
  - age[4] = Last element of array age = 7

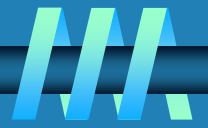
Accessing Array Elements Sample Program

```
#include <iostream>
using namespace std;
int main()
{
    int age[7] = {8,5,2,3,7,6,7};
    int i;
    // Printing array elements using loop
    for(i = 0; i < 7; i++)
        {
        // Accessing array elements using i as index
                cout << "Element at index " << i <<" is " <<
age[i];
                cout << endl;
        }
    return 0;
}
```

Output

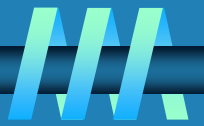
```
Element at index 0 is 8
Element at index 1 is 5
Element at index 2 is 2
Element at index 3 is 3
Element at index 4 is 7
Element at index 5 is 6
Element at index 6 is 7
```

Example: Display Largest Element of an array



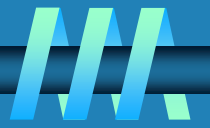
```
int main()
{
	int i, n;
	float arr[100];
	cout << "Enter total number of elements(1 to 100): ";
	cin >> n;
	cout << endl;
	for(i = 0; i < n; ++i) // Store number entered by the user
	{
		cout << "Enter Number " << i + 1 << " : ";
		cin >> arr[i];
	}
	for(i = 1;i < n; ++i) // Loop to store largest number to arr[0]
	{ // Change < to > if you want to find the smallest element
		if(arr[0] < arr[i])
		arr[0] = arr[i];
	}
	cout << endl << "Largest element = " << arr[0];
	return 0;
}
```

Example: Display the Sum of Array Elements

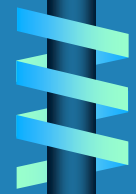


```
#include<iostream>
using namespace std;
int main()
{
	int i, n,sum=0;;
	float arr[100];
	cout << "Enter total number of elements(1 to 100): ";
	cin >> n;
	cout << endl;
	for(i = 0; i < n; ++i) // Store number entered by the user
	{
		cout << "Enter Number " << i + 1 << " : ";
		cin >> arr[i];
	}
	for(i=0;i<n;++i)
	{
		sum=sum+arr[i];
	 }
	 cout<<"\nSum of array elements is:"<<sum;
	return 0;
}
```

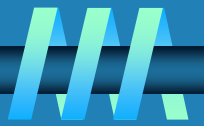
Problems



1. Write a program that reads integers, finds the largest of them, and counts its occurrences. Assume that the input ends with number 0. Suppose that you entered 3 5 2 5 5 5 0; the program finds that the largest is 5 and the occurrence count for 5 is 4. (Hint: Maintain two variables, max and count. max stores the current max number, and count stores its occurrences. Initially, assign the first number to max and 1 to count. Compare each subsequent number with max. If the number is > max, assign it to max and reset count to 1. If the number is equal to max, increment count by 1.)

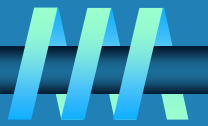


Solution



```
#include <bits/stdc++.h>
using namespace std;
int main() {
                int n=0,a[1000],c=0,val=10,ma=INT_MIN;//n is for size c for count
                of maximum,ma to store maximum.
                cout<<"Enter the values"<<endl;
                cin>>val;//taking input..
        while(val!=0)
        {
                a[n++]=val;
                cin>>val;
        }
        for(int i=0;i<n;i++)//loop for finding the maximum in the array.
        {
                ma=max(a[i],ma);
        }
        for(int i=0;i<n;i++)//loop for counting the occurence of maximum.
        {
                if(a[i]==ma)
                c++;
        }
      cout<<"The maximum value is "<<ma<<" The count of maximum value is
        "<<c<<endl;//printing the result..
return 0;
}
```

Solution



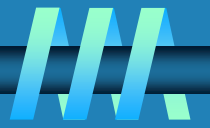
Output:-

- Enter the values

5 4 7 8 5 8 4 5 6 2 1 5 3 8 8 8 8 0

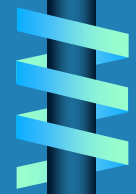
The maximum value is 8 The count of maximum value is 6

Problems

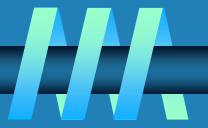


2. Write a C++ program to read a sequence of (non-negative) integers from the user ending with a negative integer and write out
   - the average of the numbers
   - the smallest number
   - the largest number
   - the range of the numbers (largest - smallest)

- Example:
  - The user enters: 3, 1, 55, 89, 23, 45, -1
  - Your program should compute the average of {3, 1, 55, 89, 23, 45} etc



C++ Strings



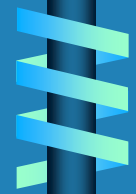
- In C++, a String is a sequence of characters.
- C++ supports two types of String representations:
  - C Strings (terminated by a null character('\0'))
  - Objects of string class (C++ Library provides a string class)

1. **C Strings**

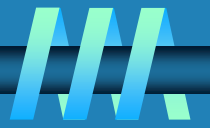
⇒ String in C programming language is a one-dimensional array of characters which is terminated by a null character('\0'). A character array which is not null terminated is not a valid C string.

⇒ In a C string, each character occupies one byte of memory including null character.

⇒ Null character marks the end of the string, it is the only way for the compiler to know where this string is ending.

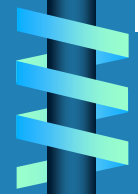


C++ Strings

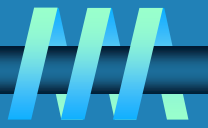


- Examples of C Strings
  - "A"
  - "TechCrashCourse"
  - "Tech Crash Course"
  - " "
- Memory representation of C string

Index	0	1	2	3	4	5	6	7
Characters	P	R	O	G	R	A	M	\0
Address	1000	1001	1002	1003	1004	1005	1006	1007



Declaration of C Strings



- Syntax of String Declaration

  char string_name[SIZE_OF_STRING];

- In the above declaration, string_name is a valid C identifier which can be used later as a reference to this string. Remember, name of an array also specifies the base address of an array. SIZE_OF_STRING is an integer value specifying the maximum number of characters which can be stored in this string including terminating null character. We must specify size of string while declaration if we are not initializing it.
  - For Example:

    char address[100];

    char welcomeMessage[200];

Initializations of C Strings

- **Initialization of string using a character array**
- Char name[16] = {'T','e','c','h','C','r','a','s','h','C','o','u','r','s','e','\0'};

Or

- char name[] = {'T','e','c','h','C','r','a','s','h','C','o','u','r','s','e','\0'};
- In the above declaration individual characters are written inside single quotes('') separated by comma to form a list of characters which is wrapped in a pair or curly braces. This list must include a null character as last character of the list.
- If we don't specify the size of String then length is calculated automatically by the compiler.

Initializations of C Strings

- **Initialization of string using string literal**

```
char name[16] = "TechCrashCourse";
```

Or

```
char name[] = "TechCrashCourse";
```

- In above declaration a string with identifier "name" is declared and initialized to "TechCrashCourse". In this type of initialization, we don't need to put terminating null character at the end of string constant. C compiler will automatically inserts a null character('\0'), after the last character of the string literal.

C++ Program for String Initialization

```
#include <iostream>
using namespace std;
int main()
{
   char nameOne[16] = {'T','e','c','h','C','r','a','s','h','C','o','u','r','s','e','\0'};
   char nameTwo[] = {'T','e','c','h','C','r','a','s','h','C','o','u','r','s','e','\0'};
   char nameThree[16] = "TechCrashCourse";
   char nameFour[] = "TechCrashCourse";
   char *nameFive = "TechCrashCourse";
   char nameSix[16];
   cout << nameOne << endl;
   cout << nameTwo<< endl;
   cout << nameThree << endl;
   cout << nameFour << endl;
   cout << nameFive << endl;
   // Taking string input from user
   cout << "Enter a string \n";
   cin >> nameSix;
   cout << "You Entered : " << nameSix;
   return 0; }
```

Output

```
TechCrashCourse
TechCrashCourse
TechCrashCourse
TechCrashCourse
TechCrashCourse
Enter a string
C++
You Entered : C++
```

C++ Program for String Initialization Cont'd…

- Another frequently used method to assign values to an array is by using directly the input stream (cin). In this case the value of the string is assigned by the user during program execution.
- When cin is used with strings of characters it is usually used with its getline method, that can be called following this prototype:

  cin.getline (char name [], int length , char delimiter = ' \n');
- where name is the address where to store the input (like an array), length is the maximum length of the buffer (the size of the array) and delimiter is the character used to determine the end of the user input, which by default.

Example

```
/*C++ program to read string using cin.getline().*/
#include <iostream>
using namespace std;
int main()
{
        char name[20], address[20];
        cout << "Name: ";
        cin.getline(name, 20);
        cout << "Address: ";
        cin.getline(address, 20);
        cout << endl << "You entered " << endl;
        cout << "Name = " << name <<endl;
        cout << "Address = " << address << endl;
        return 0;
}
```

String Handling Standard Library Functions in C++

- cstring header file contains a wide range of functions to manipulate null-terminated C strings. Below are some commonly used string handling functions of cstring header file.

Function	Description
strlen()	It returns the length of a string.
strcat()	It appends one string at the end of another string.
strcpy()	It copies characters from one string into another string.
strcmp()	It compares two strings character by character.
strtok()	It breaks a string into smaller tokens.
strchr()	It searches for the first occurrence of a character in the string.
strstr()	It searches for the first occurrence of a string in another string.

String Handling Standard Library Functions in C++

```
#include <iostream>
#include <cstring>
using namespace std;
int main(){
   char input[50];
   char copyString[60];
   cout << "Enter a string\n";
   cin >> input;
   // Printing length of string using strlen()
   cout << "Length of string : " << strlen(input) << endl;
   // copying input into copyString
   strcpy(copyString, input);
   cout << "Copied String : " << copyString << endl;
   // comparing input and copyString string
   if(strcmp(copyString, input)){
      cout << "Both Strings are Not Equal\n";
   } else {
      cout << "Both Strings are Equal\n";
   }
   return 0;}
```

Output

```
Enter a string: TechCrashCourse
Length of TechCrashCourse : 15
Copied String : TechCrashCourse
Both Strings are Equal
```

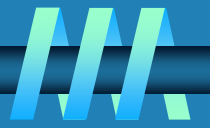
Example: How strcat() function works

```
#include <cstring>
#include <iostream>
using namespace std;

int main()

{

        char dest[50] = "Learning C++ is fun";

        char src[50] = " and easy";

        strcat(dest, src);

        cout << dest ;

        return 0;

}
```

Output

Learning C++ is fun and easy

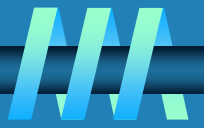
Example: How strcmp() function works



- This function returns the following three different values based on the comparison.

1. Zero(0) − It returns zero if both strings are identical. All characters are same in both strings.
2. Greater than zero(>0) − It returns a value greater than zero when the matching character of left string has greater ASCII value than the character of the right string.
3. Less than zero(<0) − It returns a value less than zero when the matching character of left string has lesser ASCII value than the character of the right string.

Example: How strcmp() function works

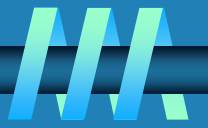


```
#include <cstring>
#include <iostream>
using namespace std;
int main()
{
        char str1[] = "Megadeth";
        char str2[] = "Metallica";
        // compare str1 and str2 lexicographically
        int result = strcmp(str1, str2);
        cout << result;
        return 0;
}
```

Output

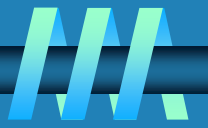
-1

Points To Remember



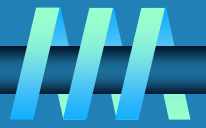
- Null character marks the end of the C strings.
- C Strings are single dimensional array of characters ending with a null character('\0').
- The ASCII value of null character('\0') is 0.
- Each character of string is stored in consecutive memory location and occupy 1 byte of memory.
- Strings constants are enclosed by double quotes and character are enclosed by single quotes.
- If the size of a C string is N, it means this string contains N-1 characters from index 0 to N-2 and last character at index N-1 is a null character.

String Class in C++



- C++ provides a String class, you can also create a objects of string class for holding strings. String class supports all the operations mentioned above, additionally much more functionality.
- At this point may not understand about classes and object, until you understand the Object Oriented Programming Concepts(which is discussed later).
- Here is a sample C++ program using string class.

String Class in C++



```
#include <iostream>
#include <cstring>
using namespace std;
int main ()
{
   // Declaring objects of string class
   string strOne = "C++";
   string strTwo = "Programming";
   string strThree;
   int length;
   // Reading string from user
   cout << "Enter a string\n";
   getline(cin, strThree);
   // Printing String
   cout << strThree<< endl;;
   // concatenating strOne and strTwo
   strOne = strOne + strTwo;
   cout << "Combined String : " << strOne << endl;
   // Printing length of string
   length = strOne.size();
   cout << "Length of Combined String : " << length << endl;
   return 0;
}
```

Output

```
Enter a string
TechCrashCourse
TechCrashCourse
Combined String : C++Programming
Length of Combined String : 14
```

THE END!