UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA

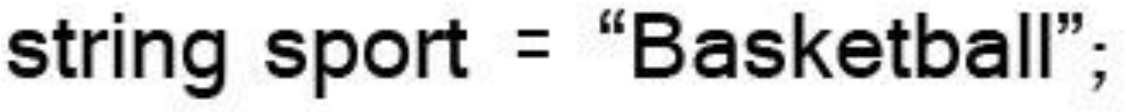
FACULTY OF TELECOMMUNICATION AND INFORMATION ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

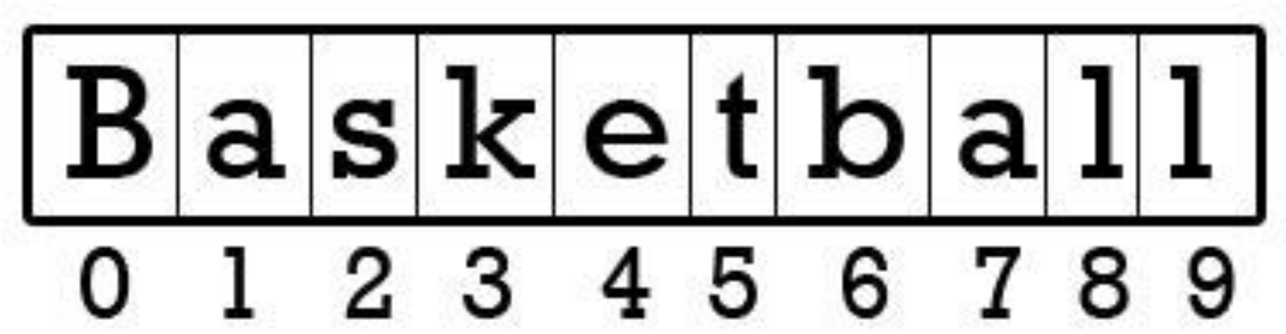
Computer Programming

Lab Manual No 06

String Arrays in C++



```
string sport = “Basketball”;
```



B	a	s	k	e	t	b	a	l	l
0	1	2	3	4	5	6	7	8	9

Semester:

Spring 2016

Objectives:-

The objectives of this session is to implement bubble, sort selection sort and learn string arrays in C++.

Sorting Arrays:-

The process of arranging data in a specified order is called sorting. Numeric type data may be arranged either in ascending or in descending order. Similarly character type data may be arranged in alphabetical order.

There are different methods to sort data into a list. The most commonly used methods are:

1. Bubble Sort
2. Selection Sort

Bubble Sort:-

Bubble sort is a simple sorting algorithm. It works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. First pass bubble out the largest element and places in the last position and second pass place the second largest in the second last position and so on. Thus, in the last pass smallest items is placed in the first position. Because, it only uses comparisons to operate on elements, it is a comparison sort.

Example:-

Let us take the array of numbers "5 1 4 2 9", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in bold are being compared.

First Pass: Considering length n

(**5 1** 4 2 9) ⟶ (**1 5** 4 2 9), Here, algorithm compares the first two elements, and swaps them.

(1 **5 4** 2 9) ⟶ (1 **4 5** 2 9), Swap since 5 > 4

(1 4 **5 2** 9) ⟶ (1 4 **2 5** 9), Swap since 5 > 2

(1 4 2 **5 9**) ⟶ (1 4 2 **5 9**), Now, since these elements are already in order (9 > 5), algorithm

does not swap them.

Second Pass: Considering length n - 1

(**1 4** 2 5 9) ⟶ (**1 4** 2 5 9)

(1 **4 2** 5 9) ⟶ (1 **2 4** 5 9), Swap since 4 > 2

(1 2 **4 5** 9) ⟶ (1 2 **4 5** 9)

Third Pass: Considering length n -2

(**1 2** 4 5 9) ⟶ (**1 2** 4 5 9)

(1 **2 4** 5 9) ⟶ (1 **2 4** 5 9)

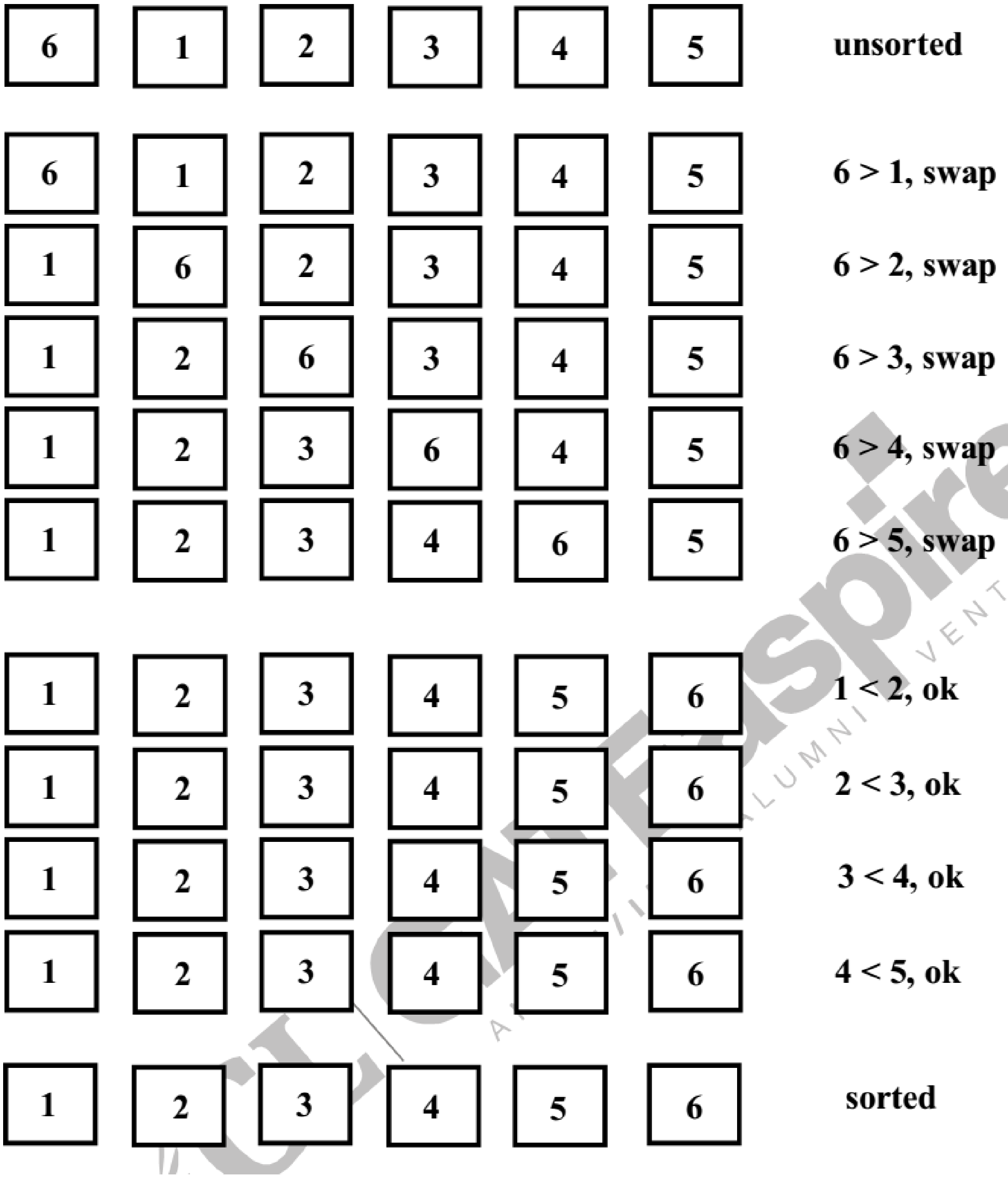
Fourth Pass: Considering length n -3

(**1 2** 4 5 9) ⟶ (**1 2** 4 5 9)

Example 4:-

Sort the following array:-

{6, 1, 2, 3, 4, 5}



6 1 2 3 4 5 unsorted

6 1 2 3 4 5 6 > 1, swap

1 6 2 3 4 5 6 > 2, swap

1 2 6 3 4 5 6 > 3, swap

1 2 3 6 4 5 6 > 4, swap

1 2 3 4 6 5 6 > 5, swap

1 2 3 4 5 6 1 < 2, ok

1 2 3 4 5 6 2 < 3, ok

1 2 3 4 5 6 3 < 4, ok

1 2 3 4 5 6 4 < 5, ok

1 2 3 4 5 6 sorted

Selection Sort:-

Selection sort is also a simple a comparison sorting algorithm. The algorithm works as follows:

Algorithm:-

1. 1st iteration selects the smallest element in the array, and swaps it with the first element.

2. 2nd iteration selects the 2nd smallest element (which is the smallest element of the remaining elements) and swaps it with the 2nd element.

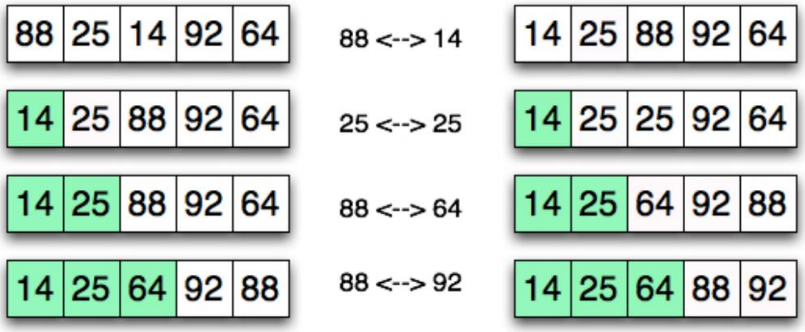
3. The algorithm continues until the last iteration selects the 2nd largest element and swaps it with the 2nd last index, leaving the largest element in the last index.

Effectively, the list is divided into two parts: the sub list of items already sorted, which is built up from left to right and is found at the beginning, and the sub list of items remaining to be sorted, occupying the remainder of the array.

Here is an example of this sort algorithm sorting five elements:

66 25 12 22 11
11 25 12 22 66
11 12 25 22 66
11 12 22 25 66
11 12 22 25 66

Example 5:-



String Arrays:-

String variables are declared in the same way as character type variables are declared. In fact a string variable is an array of character type. The length of the string is the total number of elements of the array.

The general syntax to declare a string type variable is:

```
char variable_name[n]
```

where

`variable_name` It specifies the name of the character type variable.

`n` It specifies the total number of characters of string variable.

```
char str[5],name[15],city[10];
```

The last character of every string variable is a null character. The null character is represented by ‘\0’. Thus when the information is entered into a string variable, the last character of the variable is always the null character (‘\0’). Thus if a string variable is declared with 5 characters. It can store only 4 characters and the fifth character is the null character.

For example, the variable “str” has 5 elements. In this variable, a maximum of 4 character can be entered. The fifth character will be the null character. If only 2 characters are entered in the variable then the 3rd character will be a null character. The null character is automatically added at the end of the data in the string.

Initialing String Variables:-

The data in the string variables can be assigned at the time of its declaration. For example to declare a string “str” of length 10 characters and to assign the word “Pakistan” to it the statement is written as:

```
char str[10]=”Pakistan”;
```

The above statement can also be written as:

```
char str[10]={‘P’,’a’,’k’,’i’,’s’,’t’,’a’,’n’,’\0’};
```

In the first statement the word “Pakistan” is assigned to “str” at the time of its declaration and it is given in double quotation marks. The null character ‘\0’ is automatically appended at the end of the word “Pakistan”.

In the second statement, the word “Pakistan” is assigned to the string variable “str” but it is initialized in array form. The extra null character ‘\0’ is added at the end. The null character is added because each string must have a null character at the end. If the null character is not added at the end of string then the variable of character type is handled as an array (not as a string).

Example 1:-

Write a program to initialize a string with the word “Pakistan” and then print it on the screen.

```
#include<iostream>

#include<cstdlib>

using namespace std;

int main()

{

    char str[15]="Pakistan\n";

    cout<<str;

    system("PAUSE");

    return 0;

}
```

Example 2:-

Write a program to copy one string to another string. Input the string into the first string variable and then copy this string to the second string variable by copying characters one by one.

```
#include<iostream>

#include<cstdlib>

using namespace std;

int main()

{

    char str1[15],str2[15];

    int i;

    cout<<"ENTER ANY STRING \n";
    cin>>str1;
    for(i=0;str1[i]!='\0';i++)
    str2[i]=str1[i];
    str2[i]='\0';
    cout<<str2<<endl;;
    system("PAUSE");
    return 0;
}
```

Multi-Dimensional Arrays:-

A multi-dimensional array is an array of more than one array.

For example, an array of two one-dimensional array is called two dimensional array. It is also known as table or matrix.

A two dimensional array consist of columns and rows. Each element of the two dimensional array is referenced by its index values or subscripts. The index value of a two dimensional array consist of subscripts. One subscript represents the row number and the second subscript represents the column number.

An array that requires two subscripts to identify a particular elements is also known as the double subscripted array.

For example if "temp" is a two dimensional array having 7 rows and 3 columns, it first element is temp [0][0] and the last element is temp[6][2].

The first subscript represents the row and the second subscript represents the column.

Temp[0][0]	Temp[0][1]	Temp[0][2]
Temp[1][0]	Temp[1][1]	Temp[1][2]
Temp[2][0]	Temp[2][1]	Temp[2][2]
Temp[3][0]	Temp[3][1]	Temp[3][2]
Temp[4][0]	Temp[4][1]	Temp[4][2]
Temp[5][0]	Temp[5][1]	Temp[5][2]
Temp[6][0]	Temp[6][1]	Temp[6][2]

Similarly an array of tables is called the three dimensional array. It requires three subscripts to represent its elements.

The C++ compiler support a maximum of 12 subscript arrays.

Declaration of Two Dimensional Arrays:-

A two dimensional array is declared by giving two indexed values enclosed in square brackets.

The first indexed value represents the total number of rows and the second represents the total number of columns.

The syntax to declare a two dimensional array is:

```
type array_name [r][c];
```

where

`type`	represents the data type of the array e.g. int, float, double, char etc.
`array_name`	represents the name of two dimensional array.
`r`	represents the total number of rows of table. It is an unsigned number.
`c`	represents the total number of columns of the table. It is an unsigned number.

For example to declare an integer type table “abc” having 12 rows and 3 columns, the declaration statement is written as:

```
int abc [12][3];
```

The total number of elements of the above table “abc” are 12x3=36.

The first index value for the row of the above table “abc” is 0 and the last value is 11. Similarly the first value for the column is 0 and the last value is 2.

In C++ variables one dimensional arrays and multidimensional arrays all of the same data type can be declared in a single statement.

```
float a,b,c[10],x[8][4],y,ab[12][3];
```

In the above declaration ‘a’, ‘b’and ‘y’ are simple variables, ‘x’ and ‘ab’ are two dimensional arrays and ‘c’ is a one dimensional array.

Accessing Data in Two Dimensional Array:-

Data is entered into individual elements of a two dimensional array. To enter data, the element is referenced by its index or subscript value. Similarly data is retrieved from an array from individual elements of the array.

Usually nested loops are used to access data elements of the two dimensional array. The following program example inputs data into a table and then print out the same data on the computer screen in tabular form. The table name is abc and it has two rows and 3 columns.

Nested loops has been used to enter data into the elements of the table. The upper loop has been used to change the index values of rows and the inner loop has been used to change the index of the columns.

When the upper loop is executed first time the values in the first row and all its columns are entered and the value of ‘r’ is incremented by ‘1’.

In the second repetition of the upper loop the value in the second row and all its columns are entered. This process is repeated till data in all elements of the table are entered.

Similar method is used to print out data from the elements of the table.

Example 3:-

```
#include<iostream>

#include<cstdlib>

using namespace std;

int main()

{

    int abc[2][3],r,c;

    cout<<"Input Data into the Table"<<endl;

    r=0;
```

```
while(r<=1)
{
        c=0;
        while(c<=2)
        {
                cout<<"Enter value in row = ";
                cout<<r<<" coulumn = "<<c<<"  ";
                cin>>abc[r][c];
                c=c+1;
        }
r=r+1;
}
cout<<"Printing the data from the table"<<endl;
r=0;
while(r<=1)
{
        c=0;
        while(c<=2)
        {
        cout<<abc[r][c]<<"\t";
        c=c+1;
        }
cout<<endl;
```

```
    r=r+1;
    }
    system("PAUSE");
    return 0;
}
```

Lab Task:-

Write a program to input data into a table having 5 rows and 4 columns. Find out the maximum number entered in the table and print it on the screen.

Initializing Tables:-

Assigning values to the elements of the table at the time of declaration is called its initializing of the table.

The values are assigned to the elements of a table at the time of its declaration in the same manner as they are assigned to the elements of a list.

For example, to assign values to the table abc[2][3] that has two rows and three columns values are assigned row wise.

The declaration and initialization in the above table will be.

```
     int abc[2][3]={{16,33,2},{10,12,16}};
```

Example 4:-

Write a program to initialize values to a table xyz having 4 columns and 3 rows at the time of declaration. Also print data from the table in tabular form.

```
#include<iostream>
#include<cstdlib>
using namespace std;
int main()
```

```
{
    int abc[3][4]={{2,3,10,1},{6,12,16,3},{4,3,13,17}};
    int r,c;
    for(r=0;r<=2;r++)
    {
                     for(c=0;c<=3;c++)
                     {
                                          cout<<abc[r][c]<<" \t";
                     }
    cout<<endl;
}
    system("PAUSE");
    return 0;
}
```

Initializing Character Type Tables:-

The values in a table of “char” type is also assigned in the same way as in int, float or double type table.

For example:

```
char st[5][3]= {{‘a’,‘x’,’y’},{‘p’,’e’,’t’},{‘p’,’a’,’k’},{‘a’,’b’,’c},{‘3’,’1’,’6’}};
```

Each character is enclosed in single quotation marks.

The variable “st” can also be treated as a string array. In case it is taken as an array of string type, it has a capacity to store five strings each of 3 characters including null characters. Thus only two characters can be stored in each string and the null character is automatically inserted at the end.

To initialize the above data, the variable “st” is declared and initialized as:

```
char st[5][4] ={“axy”,”pet”,”Pak”,”abc”,”316”};
```

Values assigned to string variables are enclosed in double quotation marks and are separated by commas. The length of string is 4 characters.

Example 5:-

Write a program to input data into an array of strings and then print it on the screen.

```
#include<iostream>

#include<cstdlib>

using namespace std;

int main()

{

    char str[6][15];

    int c;

    for(c=0;c<5;c++)

    {

                cout<<"Enter String "<<c+1<<" :";

                cin>>str[c];

    }

    //Print Strings

    for(c=0;c<5;c++)

    {

                cout<<str[c];

    }

    cout<<endl;
    system("PAUSE");

    return 0;

}
```

Lab Tasks:-

1. Implement the bubble sort algorithm by taking the array as input from user.
2. Implement the selection sort algorithm by taking the array as input from user.
1. Write a program in C++ to input value into a table. Find out the total number of odd and even values entered in the array. Print these on the screen.
2. Write a program in C++ to input data into a string. Insert a character into a specified location into the string. Print on the screen the entered string and the string after entering the character.
3. Write a program in C++ to input data into a string and find out the length of the string.