C++ Course Notes | Strings | Week 7

Hello everyone welcome to the weekly lecture notes

Topics to be covered:

- Introduction to strings
- Indexing of strings
- ASCII table
- User input string
- String vs character array
- Commonly used inbuilt functions
- Bucket sort and its applications
- Stringstream class

What is a String in C++?

A string in C++ is a type of object representing a collection (or sequence) of different characters. Strings in C++ are a part of the standard string class (std::string). The string class stores the characters of a string as a collection of bytes in contiguous memory locations.

To use string one must include the header the file #include or the universal header file #include.

Syntax :

`string String_Name`;

Example: ` string str = "pwskills"`;

`string subject = "C++";`

Different ways of defining a string:

1. `string str_name = "hello coders";`
2. `string str_name("physics wallah");`

Indexing of characters in a string:

Let's assume any string to be `pwcoder`. The indexing is similar to indexing in an array. It begins with 0 from the very first character and ends with a null character(\0). An extra space is for a null character after the end of any string.

0	1	2	3	4	5	6	7
p	w	c	o	d	e	r	\0

ASCII values:

Each character has an associated integer/numeric value.

For example 'a' to 'z' ranges from 97 to 122.

Where a has a value 97, b has a value 98, c for 99 and so on.

A -> 65

B ->66

|

|

|

Z->90

There are numeric values for other characters such as #, @, $ etc.

For reference one can check the ascii table for numeric value of any character whose link is provided as below.

www.cs.cmu.edu
https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Taking string input:

To provide our program's input from the user, we generally use the cin keyword along with the extraction operator (>>). By default, the extraction operator considers white space (such as space, tab, or newline) as the terminating character. So, suppose the user enters "Physics Wallah" as the input. In that case, only "Physics" will be considered input, and "Wallah" will be discarded.

Let us take a simple example to understand this:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  string str;

  cout << "Enter your string: ";

  cin >> str;
  // The user input will be stored in the str variable

  cout << "You have entered: " << str;

  return 0;
}
```

Output:

```
Enter your string: pwskills
```

```
You have entered: pwskills
```

In the above example, the user entered "Physics Wallah" in the input. As " " is the terminating character, anything written after " " was discarded. Hence, we got "Physics" as the output.

To counter this limitation of the extraction operator, we can specify the maximum number of characters to read from the user's input using the cin.get() function.

`getline(cin,str)`

Let us take an example to understand this:

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

int main() {
  char str[50];

  cout << "Enter your string: ";
  cin.get(str, 50);

  cout << "You have entered: " << str << endl;
  return 0;
}
```

Output:

```
enter your string: Physics Wallah
```

```
You have entered: Physics Wallah
```

Here we have declared a character array, of maximum size of 50 characters.

We have taken input by writing the statement:

cin.get(str_name , lengthOfString);

String vs. Character Array:

C++ supports both strings and character arrays. Although both help us store data in text form, strings and character arrays have a lot of differences. Both of them have different use cases. C++ strings are more commonly used because they can be used with standard operators while character arrays can not. Let us see the other differences between the two.

Comparison	String	Character Array
Definition	String is a C++ class while the string variables are the objects of this class	Character array is a collection of variables with the data type char.
Syntax	string string_name;	char arrayname[array_size];
Access Speed	Slow	Fast
Indexing	To access a particular character, we use "str_name.charAt(index)" or "str[index]".	A character can be accessed by its index in the character array.
Operators	Standard C++ operators can be applied.	Standard C++ Operators can not be applied.
Memory Allocation	Memory is allocated dynamically. More memory can be allocated at run time.	Memory is allocated statically. More memory can not be allocated at run time.

Commonly used inbuilt string functions:

1. **Reverse()** : This function accepts 2 parameters and reverses the string from beginning pointer to the end pointer.

Syntax: `reverse(ptr1 , ptr2)`. The first pointer ptr1 is included and the second pointer ptr2 is not included. That means the string from ptr1 to ptr2 - 1 will be reversed.

Time complexity: let n = ptr2 - ptr1. Therefore time taken by the reverse() function is O(n) where n is the number of characters involved in the reverse process.

The code showing reverse function is illustrated below:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cout<<"enter your string: ";
    cin>>s;
    reverse(s.begin()  , s.end());
    cout<<"The reversed string is: "<<s;
}
```

Output:

```
enter your string: PWskills
```

```
The reversed string is: sllikswP
```

2. **substr()**: This function is used to generate a substring of a given string.

Syntax: `str_name.substr(position , length)`

This is the general syntax where we provide the name of the string whose substring is required. The first parameter indicates the position from where the substring extraction begins, and the second parameter indicates the length till where you want the substring.

An example illustrating the function is given below:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cout<<"enter your string: ";
    cin>>s;
    string str = s.substr(3  , 5);
    cout<<"The substring obtained is: "<<str;
}
```

Output:

```
enter your string: physicsWallah
```

```
The substring obtained is: sicsW
```

Here we have given the starting index as 3 and want the substring of length 5.

To get a substring after a particular character:

Syntax 2: `str_name.substr(position)`

As per this syntax, the complete string after the mentioned “position” will be extracted as a substring as shown below in the example.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cout<<"enter your string: ";
    cin>>s;
    string str = s.substr(3);
    cout<<"The substring obtained is: "<<str;
}
```

Output:

```
enter your string: physicswallah
```

```
The substring obtained is: sicwallah
```

Here the complete string after index 3 (including index 3) is obtained as a substring.

To get a substring before any character:

Syntax 3: `str_name.substr(0 , end_position)`

Here the string from 0 to end_position - 1 is considered as the required substring.

This is a specific case of general syntax.

3. **push_back()**: This function is used to push another character or string at the end of the current string(string with which the function is used/called).

Syntax: `str_name.push_back(str2_name)`

`str2_name` string will be pushed at the end of the string `str_name` as shown in the following example.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cout<<"enter your string: ";
    cin>>s;
    cout<<"enter the new string or character to be pushed: ";
    char t;
    cin>>t;
    s.push_back(t);
    cout<<"The new string obtained is: "<<s;
}
```

Output:

```
enter your string : physics
```

```
enter the new string or character to be pushed: wallah
```

```
The new string obtained is: physicswallah
```

4. **The "+" operator**: this is directly used to concatenate 2 strings.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cout<<"enter your string: ";
    cin>>s;
    cout<<"enter the new string or character to be pushed: ";
    string t;
    cin>>t;
    s += t;
    cout<<"The new string obtained is: "<<s;
}
```

Output:

```
enter your string : physics
```

```
enter the new string or character to be pushed: wallah
```

```
The new string obtained is: physicswallah
```

Here it was written as s+=t that is equivalent to s = s + t, had it been written s = t +s then the resulting string would be `wallahphysics`.

When we write `s+=t` that signifies we are appending the string `t` at the back of string s.

Statement 2 `s = s + t` shows we have created a new copy of string s and added string t at the back of string s.

The only difference in the above two statements is that in the second case extra space will be taken by the reformation of string `s` whereas in the first case no such extra space of string `s` will be occupied.

Homework: Try t+s functionality on your own.

5. **strcat()**: This function is used to concatenate 2 character arrays.

Syntax: `strcat(s1_name , s2_name)`

This is equivalent to `s1_name  = s1_name + s2_name` as illustrated below:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    char s[20];
    cout<<"enter your string: ";
    cin>>s;
    cout<<"enter the new string or character to be pushed: ";
    char t[20];
    cin>>t;
    strcat(s,t);
    cout<<"The new string obtained is: "<<s;
}
```

Output:

```
enter your string : physics
```

```
enter the new string or character to be pushed: wallah
```

```
The new string obtained is: physicswallah
```

6. **size()**: this function is used to find the size of the string.

Syntax: `str_name.size()`

```
#include <bits/stdc++.h>
using namespace std;
int main() {
   string str;
   cin>>str;

   cout<<str.size();
}
```

Output:

```
pwcoders
```

8

Difference between **size()** and **strlen()** functions:

- **size()** function is used to know the length of the strings whereas strlen() function is used to know the length of the character array.
- **strlen()** function takes O(n) time whereas size() function uses O(1) time, where n = length of the array.

7. **to_string()**: This function is used to convert a numeric value into string type.

Syntax:

Suppose n is an integer. To convert this integer to string we have to write the following way:

`to_string(n)`

This function is used when we have to perform operations on digits of a number. Converting into string makes the digits accessible in an indexed manner which is quite easy to use.

The following code illustrates:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
   cout<<"enter the number: ";
    int n;
    cin>>n;
    string str = to_string(n);
    cout<<str<<endl;
    cout<<str[0]<<" "<<str[1]<<" "; //printing the first two digits(from left to
right / highest to lowest priority) of the entered number
}
```

Output:

```
3452
```

Bucket sort on strings:

We have 128 different types of characters available that can be A-Z, a-z, special characters such as !, @, # etc, digits from ‘0’ to ‘9’etc.

Since the maximum number of characters can be at max 128 we can use an array or vector of size 128 such that each of the index of the array represents a particular character.

Now we can use the ASCII values of the characters to mark the indices of the characters. For example,

The ASCII value of ‘a’ = 97

‘b’ = 98 and so on. . . . .

So if we created an array ‘arr’ of size 128 then arr[97] will be reserved for the frequency of ‘a’, similarly arr[98] will be reserved for frequency of ‘b’ and so on.

Once the frequencies of every character are stored we can use it to build a sorted string, because it is sure that ‘a’ will always be ahead of ‘b’, ‘b’ will always be ahead of ‘c’ and so on. This is the concept of bucket sort. Below example will clarify this more.

Example , Given a string ‘str’, sort the given string using count sort technique where the string only contains lowercase alphabetic characters (a - z).

```
#include<bits/stdc++.h>
using namespace std;

string countSort(string str) {
    vector<int>arr(26 , 0); //array to store the frequencies of all alphabets
    for(int i=0;i<str.size();i++){
        arr[str[i] - 'a']++;      //storing the required frequencies of particular
character
    }
    int n = str.size();
    int j = 0;
    for(int i=0;i<26;i++){
        while(arr[i]--){        //iterating till the frequency of a particular
character and appending to the string
            str[j++] = i + 'a';
        }
    }
    return str;     //returning the sorted string
}
int main()
{  string s;
   cout<<"Enter the string: "<<endl;
   cin>>s;
```

Stringstream class :

A stringstream associates a string object with a stream allowing you to read from the string as if it were a stream (like cin). To use stringstream, we need to include sstream header file. The stringstream class is extremely useful in parsing input.

```
#include<bits/stdc++.h>
using namespace std;

void print(string str){
   stringstream s(str);
   string word;
   while (s >> word){
       cout<<word<<endl;
    }
    return;
}
int main(){
   char s[50];
    cout<<"Enter the string : ";
    gets(s);
    cout<<"Desired output is: "<<endl;
   print(s);
   return 0;
}
```

Output:

`Enter the string : "PW is a revolution"`

Desired output is:

`PW`

`is`

a

`revolution`
