String v/s Char

* Char type of variable can store a single character.

  Eg:- char c = 'A';

* String is an array of characters used for storing a name, word, sentence etc.

* "A group of characters enclosed b/w double quotes is called a string constant".

* It is terminated by '\0'.

  Eg:- char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

  char str[] = "Hello"; ('\0' will be included at the end.)

* If null (\0) is not given at the end of character then the compiler automatically assume it.

* So, it is not necessary to giving null at last character of word.

* Class string v/s char str[] -

→ String is a built-in class in C++ to store a string.

- Internally ~~it~~ it contains an array of characters.
- It will create array of characters in heap.
- It has many functions for performing string operations.

Eg:- string str = "Hello";

(It will create a string object.)

→ char str[10] = "Hello";

(It is a 'C' style string. It contains set of characters terminated by '\0'.)

* String literal (char *s = "Hello";)

→ char *s = "Hello" may not be supported in all compilers. So change it to -

- const char *s = "Hello";
- "Hello" is a literal.
- Literal means direct value used in a program like int x = 10;
- Literals are stored in code section.
- Literals can not be modified, like s[2] = 'k' is invalid.

* char *s V/s char s[10] -

→ char s[10]; is an array of characters. It can contain string.

Eg:- char str[5] = {'H', 'e', 'l', 'l', 'o', '\0'}

'\0' also take 1 byte space.

→ char *s; is a pointer of type character. It can point of a char array or string.

Eg:- const char *s = "Hello"; // 's' is pointing to a string literal.

char str[] = "Hello";

char *s = str; // 's' is pointing to a char array.

Example -

1) const char *s = "Hello";

cout << *s << endl; → H

cout << s; → Hello

o/p

2) char s[] = "Hello";

char *ptr = s; // A char pointer pointing to s

cout << *s << endl; → H

cout << s << endl; → Hello

cout << ptr << endl; → Hello

cout << *ptr << endl; → H

* getline (cin, str) v/s cin.getline (str, 100) (100 → size)

→ getline (cin, str) -

It is used for reading a string object. It will not work for char array.

Eg:- string str;
getline (cin, str); // It is used with string class object.

→ cin.getline (str, 100) -

It is used for reading a string in char array. It will not work for string class object.

- cin.get (str, 100) is also used for char array.

Eg:- char str [10];
cin.getline (str, 10); // It is used with char array.
or, cin.get (str, 10);

* cin.ignore() -

→ When we enter any input from keyboard, it is transfered to an input buffer.

→ Program reads the data from input buffer.

→ After entering value from keyboard, we hit enter.

→ Program will read the value and ignore enter key from buffer.

→ If program doesn't ignore it then it may not read next input.

→ cin.ignore() is used for forcing the program to ignore it.

→ Usually programs don't read a string value because of enter key.

→ Use cin.ignore() before reading a string.

Example -

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

int main()
{
    char s[100];
    char s2[100];
    cout << " Enter your name : ";
     cin.get(s, 20);
     cout << "\nWelcome " << s << endl;

     cin.ignore(); // If we use cin.getline() instead
                   of cin.get() then we don't
                      need to use it.

      cout << " Enter your name again ";
       cin.get(s2, 20);
       cout << " \nWelcome " << s2 << endl;
}
```

* String (character array) library function -

#include <cstring>

↓

This is a library that provides a set of functions for manipulating c-style string which are arrays of characters terminated by '\0'.

→ These functions include string copy (strcpy), string concatenation (strcat), string comparison (strcmp) and others.

1) strlen (str1) → Used to get the length of string.

2) strcat (destination, source) - Used to add two string.

Dest = "good";
Source = "Morning"; > good Morning.

or, strncat (destination, source, length);

↓

Here 'n' tells the how many character you want to add.

→ If suppose only 3 characters we want to add then simply we write '3' in the place of length.

3) strcpy (destination, source) - used to copy the string

or strncpy (destination, source, length);

4) strcmp (S1, S2) → It is used to compare two string a/c to dictionary order.

→ When both strings are same then it returns '0'.
→ If S1 < S2 then it returns '-ve' value.
→ If S1 > S2 then it returns '+ve' value.

Eg:- 1) str1 = "Hello";
str2 = "Hello"; > returns '0'

2) str1 = "hello";
str2 = "Hello";

Here 'h' and 'H' has same dictionary order but their ASCII value different.

h = 105, H = 73

str1 > str2 → Returns '+ve' value

3) str1 = "minor"; str2 = "elder";

m = 13, e = 5

str1 > str2 Return +ve value

* String class function - (String of character)

#include <string>

↓

It provides a set of member functions for manipulating string class object.

→ Basic functions of class string -

1) s.length() → It gives the length of the string.

Here s → string name

Eg:- string str = "Hello";

str.length();

2) s.size() - It gives the size of string (same as length)

3) s.capacity() - It gives the capacity of the string and capacity of the string is always bigger than its size.

4) s.resize(30) → By using this we can resize the capacity of the string.

5) s.max_size() - It gives the max^m possible size of the string.

6) s.clear() - It clears the content of string.

7) s.empty() - It finds the string is empty or not.

8) s.append("World"); → This will add a new content to a string.

Eg:- string s1 = "Hello";

s1.append("World");

cout<<s1; → Hello World

9) s.insert(3, "kk"); → This function will insert a given string at a given index.

Eg:- string str = "Hello";
0 1 2 3

str.insert(3, "kk");

cout << str; → HelkkIO

or, str.insert(3, "APple", 2);

It means we can only insert 2 char of "APple" (AP).

If we not use it then whole word will be inserted.

cout << str; → HelAPlo

10) S.replace (3, 5, "aa"); → It is used for replacing the string.

Eg:- string str = "Programming";

str.replace (3, 4, "kk");

3 → Position for start replacing

4 → How many character want to replace.

cout << str; → Prokkming

11) S.erase() → Same as clear().

12) S.push_back('z') → It is used to add a single character at the end of string.

Eg:- string str = "Hello";

str.push_back('B');

cout << str; → HelloB

13) S.pop_back() → It is used to delete a single letter from last.

string str = "Hello";

str.pop_back();

cout << str; → Hell

14) s1.swap(s2) → It is used to swap the two string.

Eg:- string str1 = "Hello";
string str2 = "World";
str1.swap(str2);
cout<< str1; → World
cout<< str2; → Hello

15) S.copy(char dest[]) → Used to copy a string in char array.

Eg:- string s = "Welcome";
char str[10];
S.copy(str, 3); → How many character you want to copy.
str[3] = '\0';
cout << str; → Wel

16) S.compare(str); → Comparing two string (same as strcmp)

Eg:- string str1 = "Hello";
string str2 = "Hello";
cout << str1.compare(str2); → Return '0'

17) s.at() → It will give a letter at particular index.

Eg:- string str = "Holiday";

str.at(4); → d

or, str[4]; → d

18) s.front() → It gives the first letter of string.

19) s.back() → It gives the last letter of string.

* Operators in string class -

- '+' → Eg:-

```
string str1 = "Hello";
string str2 = "World";
string str3 = str1 + str2;
cout << str3; → Hello world
```

- '=' →

```
str1 = str1;
```

(contents of string 1 is copied in string 2)

* Iterator in string class -

It works like a pointer

Iterator are used for traversing or accessing all the characters of a string.

String :: iterator
- begin()
- end()

→ It allows to traverse from left to right.

String :: reverse_iterator
- rbegin()
- rend()

→ It allows to traverse from right to left.

Eg:- 1) String str = "today";

String :: iterator it; → variable name

```
for(it. str.begin(); it != str.end(); it++)
{
    cout << *it;   → today
}
```

2) String str = "today";

```
string :: reverse_iterator it;
for(it = str.~~length~~ rbegin(); it != str.rend(); it++)
{
    cout << *it;   → yadot
}
```