String Handling And I/O Functions

- Array is the effective mechanism to handle large number of same type of data
- The string array and some string built in functions are used to perform string datas in c++

String handling using arrays

- Strings are the combination of characters enclosed in a doublequates (“ ”)
- Strings are actually a one-diamensional array of characters ended with a null character ' \0 '
- A string cannot be stored in a single character variable
- Character array only used to store a string

- Eg :

  char variable_name[10];

  char a[10];

  char var[6]={'H','e','l','l','o','\0'};

  char var[6]="Hello";

String initialization & declarations

- A string declared with the help of char data type

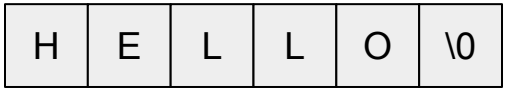
  Syntax : char array_name [size];

  Eg: char A [6];

- The size of the string depends on the size of the array

Declaration

- char A [6] = Hello ;
- char A [6] = { 'H', 'E', 'L', ‘L’, 'O', '\0' } ;



- Syntax :

char array_name [size] = string_data ;

Memory allocation for strings

- char type array is used to store a string
- the data type specify the size of the string

- Each and every string is end with a null character '\0'. so we can store size-1 characters in a array_variable

- The null character '\0’ treated as a single character and also called as the delimiter

  Eg:

  char name [4] - max of 3 characters stored

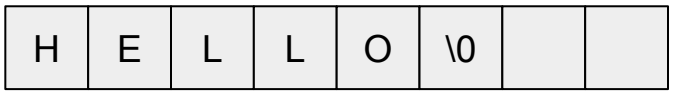
Different types of memory allocation

1. char a [8] = " HELLO " ;

   ➔ 8 byte will be allocated

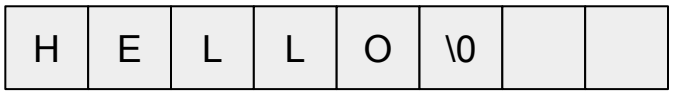
   ➔ use 5+1 = 6 bytes

   ➔ 2 byte is not used (waste)



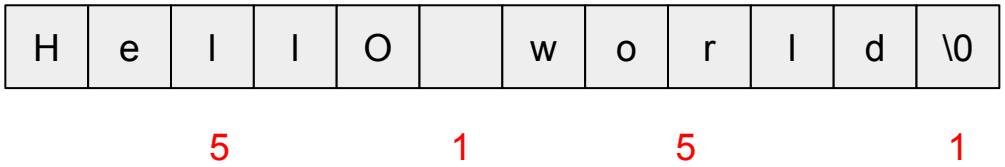
2. char a [ ] = " HELLO " ;

- ➔ size of the array not declared
- ➔ Only 5 + 1 = 6 byte will be allocated
- ➔ no wastage of memory



3. char a[ ] = " Hello world "

- ➔ size of the array not declared
- ➔ space considered as a character
- ➔ 12 byte will be allocated
- ➔ No wastage of memory



Input/output operations on strings

- The input operator " >> " and the output operator "<<" are used to perform the input / output operation in c++

- In the case of string, input operator “>>” not works properly.

Hello world

- Here the space taken as the delimiter

- ➔ To solve this problem, we use gets() - [string functions]
- ➔ gets() , puts() are defined in <stdio.h>

String Functions

1. Input functions
   - getchar()
   - getch()
   - getche()

2. Output function
   - putchar()
   - putch()

Input Functions

getchar()

- Used to access a character input through the keyboard
- Defined in <cstdio.h>

Syntax :

```
var = getchar();
```

getch()

- Used to access a character, but it will not be visible on the screen
- Define in <conio.h>

Syntax :

```
var = getch()
```

getche()

- Used to access a character, the typed character will be visible on the screen
- Define in <conio.h>

Syntax :

```
Var = getche();
```

Output Functions

putchar()

- Used to display a single characters
- Define in <stdio.h>

Syntax :

putchar (a);

putchar(“ a “);

~~putchar(“ hello “);~~

putch()

- Used to display ASCII characters
- Define in <conio.h>

Syntax :

```
putch( a );

putch( a + 2 );
```

~~putchar(“ d “);~~

Stream Function

Output Functions

- C++ provides another facility to perform input/ output operations on both character and strings
- The functions are defined in the header file <iostream.h>
- This functions are generally called as stream functions

Input functions	Output functions
1. get()	1. put()
2. getline()	2. write()

- This kinds of functions are used to transfer characters or strings between the memory and objects
- Here Keyboard & monitor are considered as the objects in c++

Input Functions

- The input functions like get() and getline() are used with cin and dot(.) operator

get()

- used to read a single character from the keyboard

  Syntax :

  ```
  var = cin.get(ch);
  ```

getline()

- Used to read a line of characters from the keyboard

  Syntax :

  cin.getline(str.len) len - limiter

  getline(str,len,ch)

Output Functions

- The input functions like put() and write() are used with cout and dot(.) operator

put()

- used to print a single character on the monitor

  Syntax :

  ```
  cout.put(ch)
  cout.put('A')
  ```

write()

- Used to print a line of character on the screen

Syntax :

```
cout.write(str, len);
cout.write(“ string ”);
```