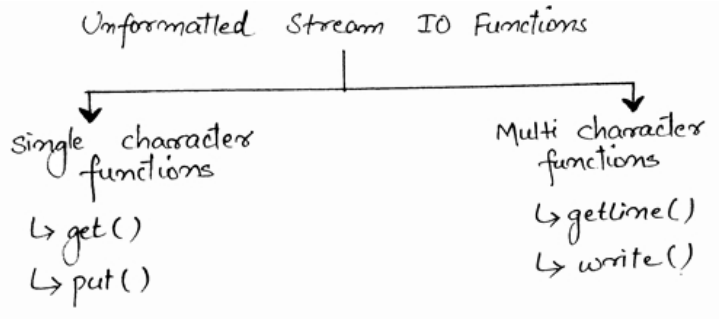
getline c, get c, put c and write c

getline c, get c, put c and write c - Unformatted stream I/O functions

getline c, get c, put c and write c - These functions are called unformatted stream IO functions where
get() and put() can handle single character input and output and
getline() and write() can handle multi character input and output.

Header files for these above functions : **"iostream.h"**



Single character functions : get() and put()

Single Character functions are used for reading and displaying a single character respectively. These functions are :

get() :
The get() function is an input function.
- It fetches a single character and store it in a character variable.

The get() is a member function of istream class and it is involved with an istream object cin.

```
char ch;
cin.get(ch); // get a character from keyboard and assign it to ch.
```

put() :
The function put() is an output function and is used to output a character at a time.
The put() function is also a member of ostream class and is involved with an ostream object cout.

```
cout.put('a');
```

Example of get() and put() :

```
char ch;
while (cin.get(ch)!='n')
cout<<ch;
```

- The above code keeps on accepting and displaying characters until a new line character (typed by pressing enter key) is encountered.

The difference between **cin>>ch** and **cin.get(ch)** is that when >> operator is used, the white spaces (e.g. tabs) and newline characters are ignored where as it is not so with **cin.get(ch).**

```
char ch;
cin.get(ch);
cout.put(ch);
cout.put(65)
```

The above code first input a character in variable ch and prints it on the screen as given by cout.put(ch).
The line cout.put(65) will display 'A' on the screen as ASCII value of A is 65.
- Thus, a number argument can also be given with put() and it will print equivalent character of that number. Multi Character Functions : getline() and write()

The **single character functions** defined above can handle single character at a time and cannot handle multiple characters. To solve this problem, there are functions that can handle strings also. These functions are :

getline() -

The getline() is an input function and needs a line of text that ends with a newline character.
- This function is used as shown below :

cin.getline(line, size);
- where line specifies the name of the variable that will store the line of text, and - size specifies that maximum size-1 characters can be stored in line; - 1 extra character in size is reserved for null character ". **write() -**

The function write() is an output function and can display an entire line.
- It takes the following form :

cout.write(line, size);
- where line specifies the string variable whose contents are to be displayed, and - size specifies the number of characters to be displayes.

Example of getline() and write() :

```
char name[30];
cin.getline(name,30);
```

- The above code reads a line of text (max 29 valid characters) and stores it under variable name.

The function getline() can read white spaces (spaces, tabs etc.) and newline characters whereas cin>>name will not read any white space or newline character.

For instance if you give following input : **"Test Program"** then
with getline() function, name will be having value "Test Program" but
when used with >>, name will be having Test only as any other character following a white space or newline character is ignored by >>.

After reading a string, cin automatically adds the terminator character '' to the string.

cout.write(name, 3);
- if name stores value "Program", then the above code will display "Pro" only.

Two write() functions can be combined together as shown below :

cout.write(string1, size1).write(string2, size2);
- This statement prints string1 followed by string2 i.e. in the same line.