getc():
It reads a single character from a given input stream and returns the corresponding integer value (typically ASCII value of read character) on success. It returns EOF on failure.

Syntax:

```
int getc(FILE *stream);
```

Example

```
#include <stdio.h>
int main()
{
   printf("%c", getc(stdin));
   return(0);
}

Input: g (press enter key)
Output: g
============================================================
```

getchar():
The difference between getc() and getchar() is getc() can read from any input stream, but getchar() reads from standard input. So getchar() is equivalent to getc(stdin).

Syntax:

```
int getchar(void);
```

Example:

```
int main()
{
   printf("%c", getchar());
   return 0;
}
Input: g(press enter key)
Output: g
============================================================
```

Gets

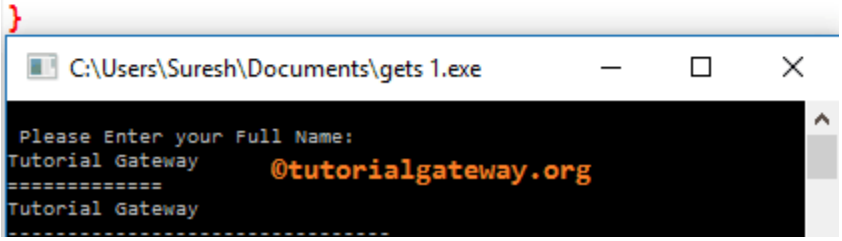
The gets method is used to read complete set of characters from the console. This program will help you to practically understand this gets function.

```
// C gets function example
#include <stdio.h>

int main()
{
        char name[50];
```

```
        printf("\n Please Enter your Full Name: \n");
        gets(name);

        printf("=============\n");
        printf("%s", name);

        return 0;
}
```

OUTPUT

```
// C gets function example

#include <stdio.h>

int main()
{
    char name[50];

    printf("\n Please Enter your Full Name: \n");
    gets(name);

    printf("=============\n");
    printf("%s", name);

    return 0;
}
```



---

putc()

putc()	Declaration: int putc(int char, FILE *fp)
	
	putc function is used to display a character on standard output or is used to write into a file. In a C program, we can use putc as below.
	
	putc(char, stdout);
	putc(char, fp);

```
#include <stdio.h>
int main()
{
  char ch;
  FILE *fp;
  if (fp = fopen("test.c", "r"))
  {
    ch = getc(fp);
    while (ch != EOF)
    {
      putc(ch, stdout);
      ch = getc(fp);
    }
    fclose(fp);
    return 0;
  }
  return 1;
}
```

Output:

```
Hi, How are you?
```

---

putchar()

putchar()	Declaration: int putchar(int char) putchar() function is used to write a character on standard output/screen. In a C program, we can use putchar function as below. putchar(char); where, char is a character variable/value.

```
#include <stdio.h>
#include <ctype.h>
int main()
{
  char c;
  printf("Enter some character. Enter $ to exit...\n");
  while (c != '$');
  {
    c = getchar();
    printf("\n Entered character is: ");
```

```
    putchar(c);
    printf("\n")
  }
  return 0;
}
```

Output:

```
Enter some character. Enter $ to exit...
A
Entered character is: A
B
Entered character is: B
$

Entered character is: $
```

Puts()

puts()	Declaration: int **puts**(const char *string) puts() function is used to write a line to the output screen. In a C program, we use puts function as below. **puts**(string); where, string – data that should be displayed on the output screen.

```
#include <stdio.h>
#include <string.h>

int main()
{
  char string[40];
  strcpy(str, "This is a test string");
  puts(string);
  return 0;
}
```

Output:

```
This is a test string
```

Ungetc()

The **ungetc()** function takes a single character and shoves it back onto an input stream. It is the opposite of the getc() function, which reads a single character from an input stream. Also, ungetc() is an input function, not an output function.
Syntax:

```
int ungetc(int char, FILE *stream)
```

```
// get 1 at the input
#include <stdio.h>
int main()
{
    int ch;

    // reads characters from the stdin and show
    // them on stdout until encounters '1'
    while ((ch = getchar()) != '1')
        putchar(ch);

    // ungetc() returns '1' previously
    // read back to stdin
    ungetc(ch, stdin);

    // getchar() attempts to read
    // next character from stdin
    // and reads character '1' returned
    // back to the stdin by ungetc()
    ch = getchar();

    // putchar() displays character
    putchar(ch);
    return 0;
}
```