Basics of Formatted Input/output in C

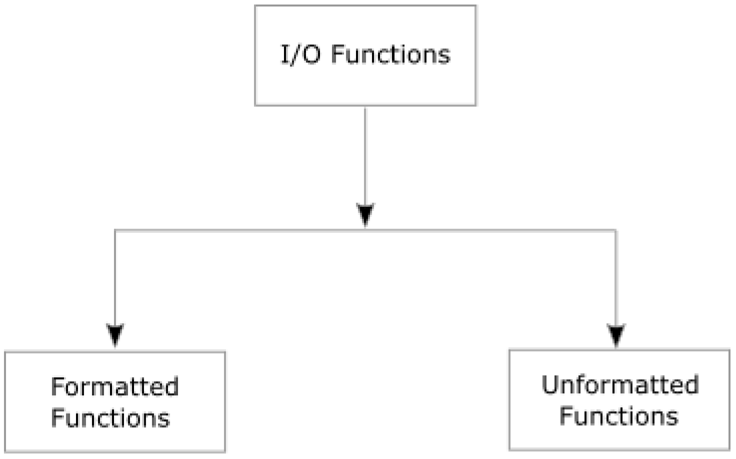
C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.
As the name says, the console input/output functions allow us to –

1. Read the input from the keyboard by the user accessing the console.
2. Display the output to the user at the console.

It is a way to tell the compiler what type of data is in a variable during taking input using **scanf**() or printing using **printf**().

Some examples are %c, %d, %f, etc.

Note: These input and output values could be of *any primitive data type*



Formatted functions	Description
scanf()	The **scanf()** function allows us to read *one or multiple* values entered by the user through the keyboard at the console.
printf()	The **printf()** function is used to print or display *one or multiple* values in the output to the user at the console.

Both **scanf**() and **printf**() function use specific format specifiers to display specific values.

Let's look at the code that allows us to read and display a single value at time using a single **scanf**() and **printf**() function.

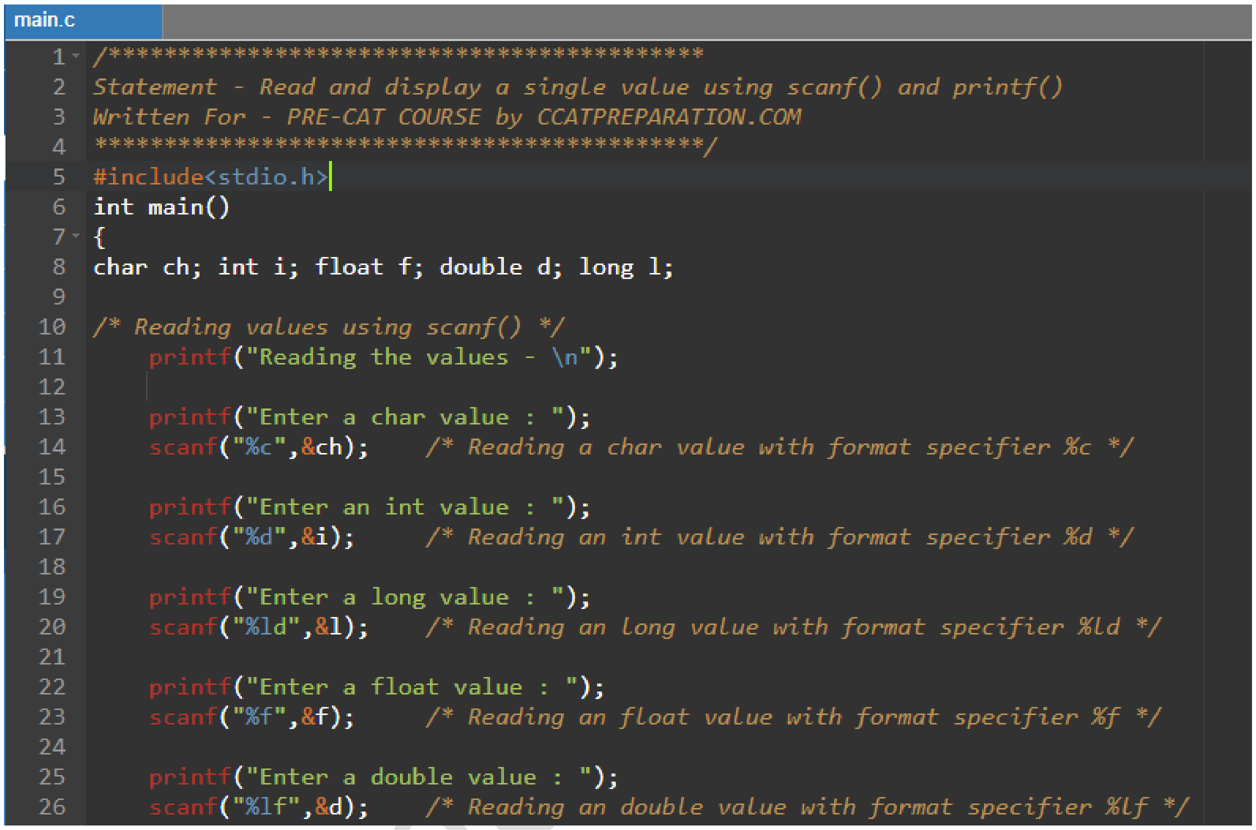
Parameter	Meaning
%d	Print an integer number printed in decimal (preceded by a minus sign if the number is negative).
%f	Print a floating point number (in the form dddd.dddddd).
%E	Print a floating point number (in scientific notation: d.dddEddd).
%g	Print a floating point number (either as f or E, depending on value and precision).
%x	Print an integer number in hexadecimal with lower case letters.
%X	Print an integer number printed in hexadecimal with upper case letters.
%c	Print a character.

We can also **limit the number of digits or characters** that can be input or output, by adding a number with the format string specifier, like **"%1d" or "%3s",** the first one means a single numeric digit and the second one means 3 characters, hence if you try to input 42, while scanf () has **"%1d",** it will take only **4** as input. Same is the case for output.
NOTE:

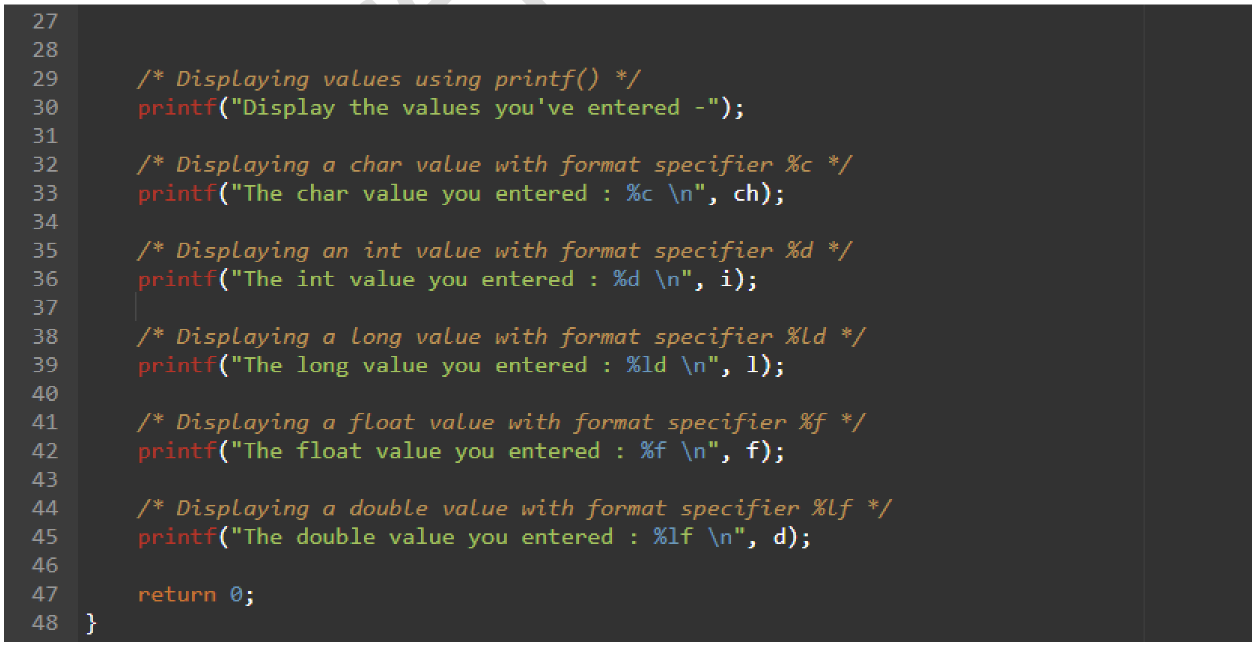
`printf ()` function returns the number of characters printed by it, and

`scanf ()` returns the number of characters read by it.

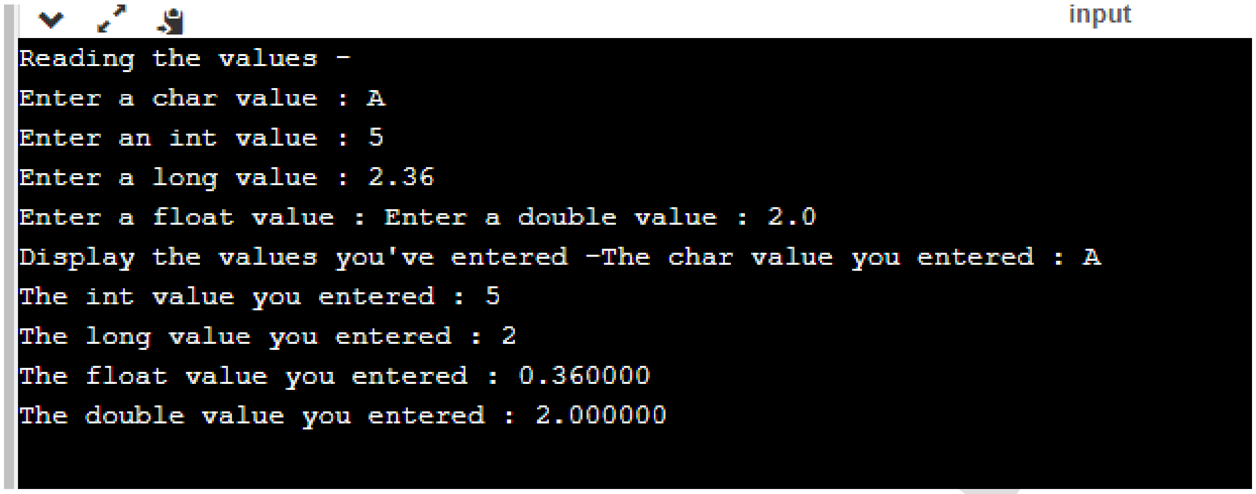
Example : Read and display a single value using `scanf ()` and `printf ()`


main.c

```c
/*********************************************
Statement - Read and display a single value using scanf() and printf()
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*********************************************/
#include<stdio.h>
int main()
{
char ch; int i; float f; double d; long l;

/* Reading values using scanf() */
    printf("Reading the values - \n");

    printf("Enter a char value : ");
    scanf("%c",&ch);    /* Reading a char value with format specifier %c */

    printf("Enter an int value : ");
    scanf("%d",&i);     /* Reading an int value with format specifier %d */

    printf("Enter a long value : ");
    scanf("%ld",&l);    /* Reading an long value with format specifier %ld */

    printf("Enter a float value : ");
    scanf("%f",&f);     /* Reading an float value with format specifier %f */

    printf("Enter a double value : ");
    scanf("%lf",&d);    /* Reading an double value with format specifier %lf */
```



```c


    /* Displaying values using printf() */
    printf("Display the values you've entered -");

    /* Displaying a char value with format specifier %c */
    printf("The char value you entered : %c \n", ch);

    /* Displaying an int value with format specifier %d */
    printf("The int value you entered : %d \n", i);

    /* Displaying a long value with format specifier %ld */
    printf("The long value you entered : %ld \n", l);

    /* Displaying a float value with format specifier %f */
    printf("The float value you entered : %f \n", f);

    /* Displaying a double value with format specifier %lf */
    printf("The double value you entered : %lf \n", d);

    return 0;
}
```



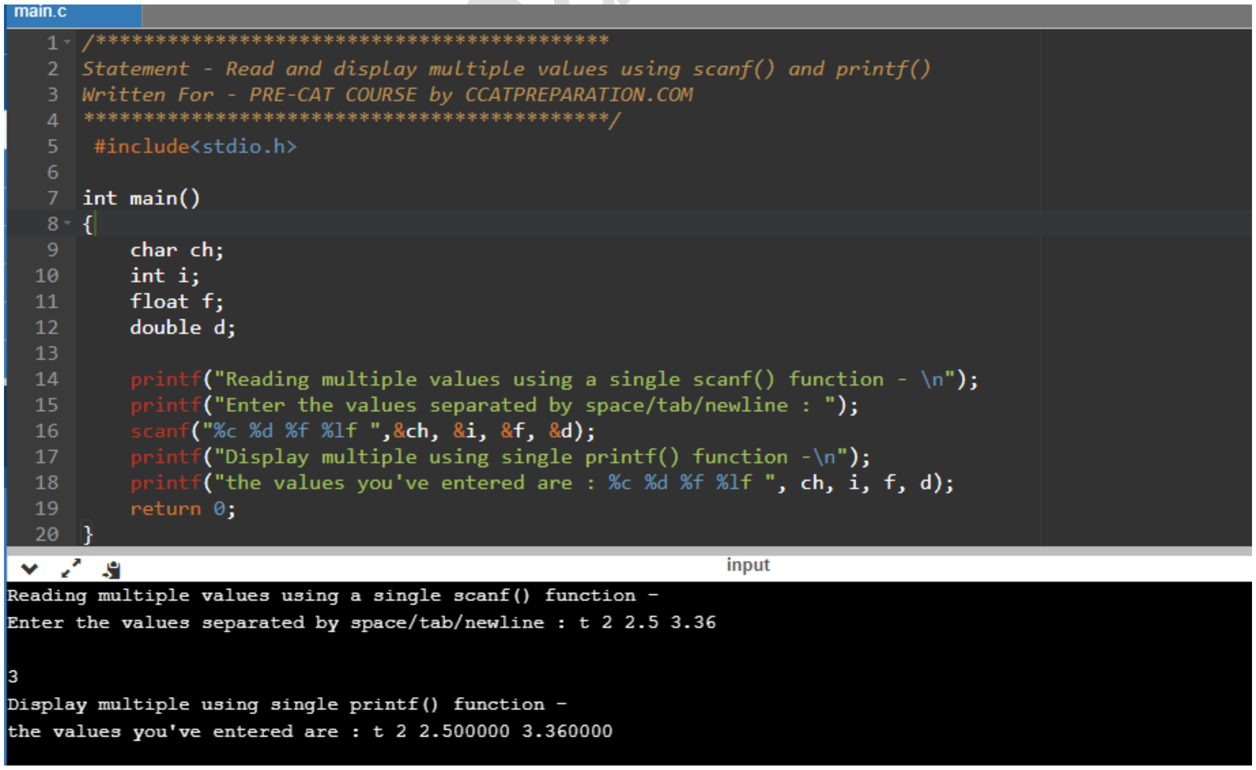
input

```
Reading the values -
Enter a char value : A
Enter an int value : 5
Enter a long value : 2.36
Enter a float value : Enter a double value : 2.0
Display the values you've entered -The char value you entered : A
The int value you entered : 5
The long value you entered : 2
The float value you entered : 0.360000
The double value you entered : 2.000000
```

As you may see in the code and its output, we have used different **format pecifiers** to *read and display* different kinds of primitive data type values.

Example : Read and display multiple values using `scanf()` and `printf ()`

Let's look at the code that allows us to read and display *multiple values* at time using a single **scanf()** and **printf()** function.



main.c

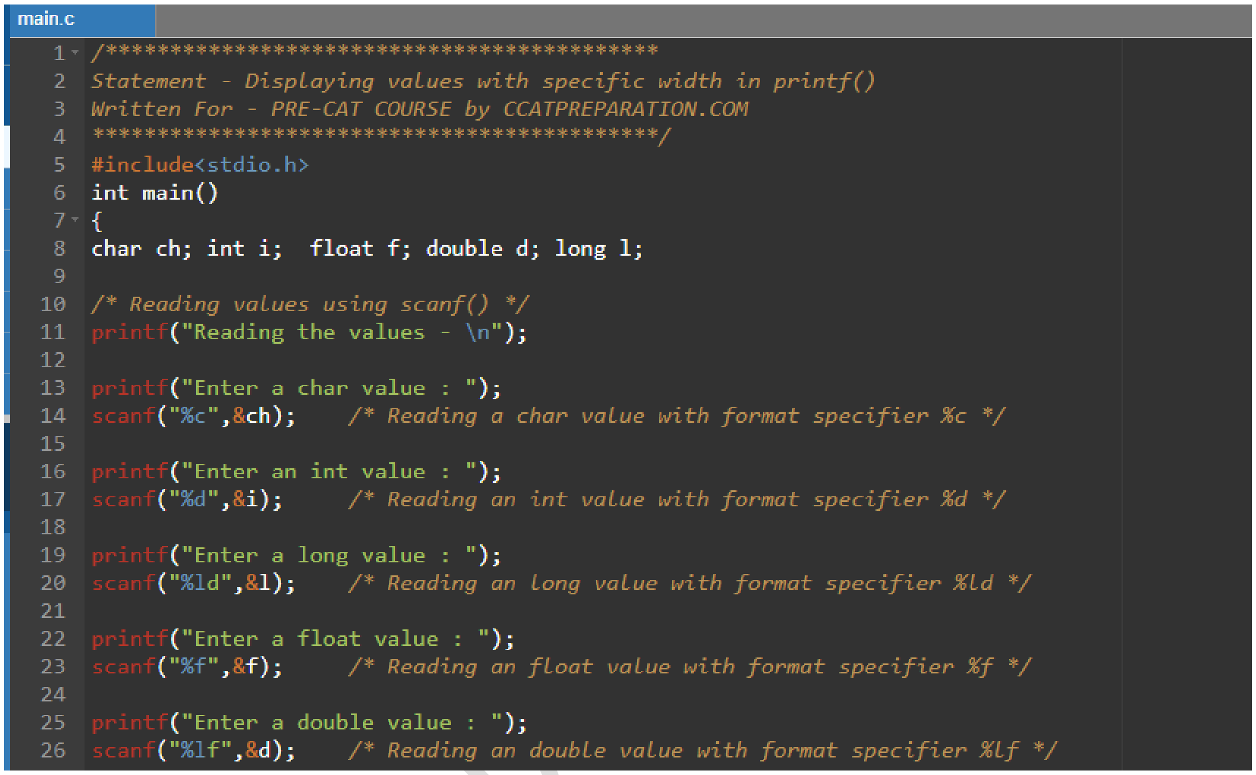
```c
/*********************************************
Statement - Read and display multiple values using scanf() and printf()
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
 *********************************************/
 #include<stdio.h>

int main()
{
    char ch;
    int i;
    float f;
    double d;

    printf("Reading multiple values using a single scanf() function - \n");
    printf("Enter the values separated by space/tab/newline : ");
    scanf("%c %d %f %lf ",&ch, &i, &f, &d);
    printf("Display multiple using single printf() function -\n");
    printf("the values you've entered are : %c %d %f %lf ", ch, i, f, d);
    return 0;
}
```

input

```
Reading multiple values using a single scanf() function -
Enter the values separated by space/tab/newline : t 2 2.5 3.36


3
Display multiple using single printf() function -
the values you've entered are : t 2 2.500000 3.360000
```

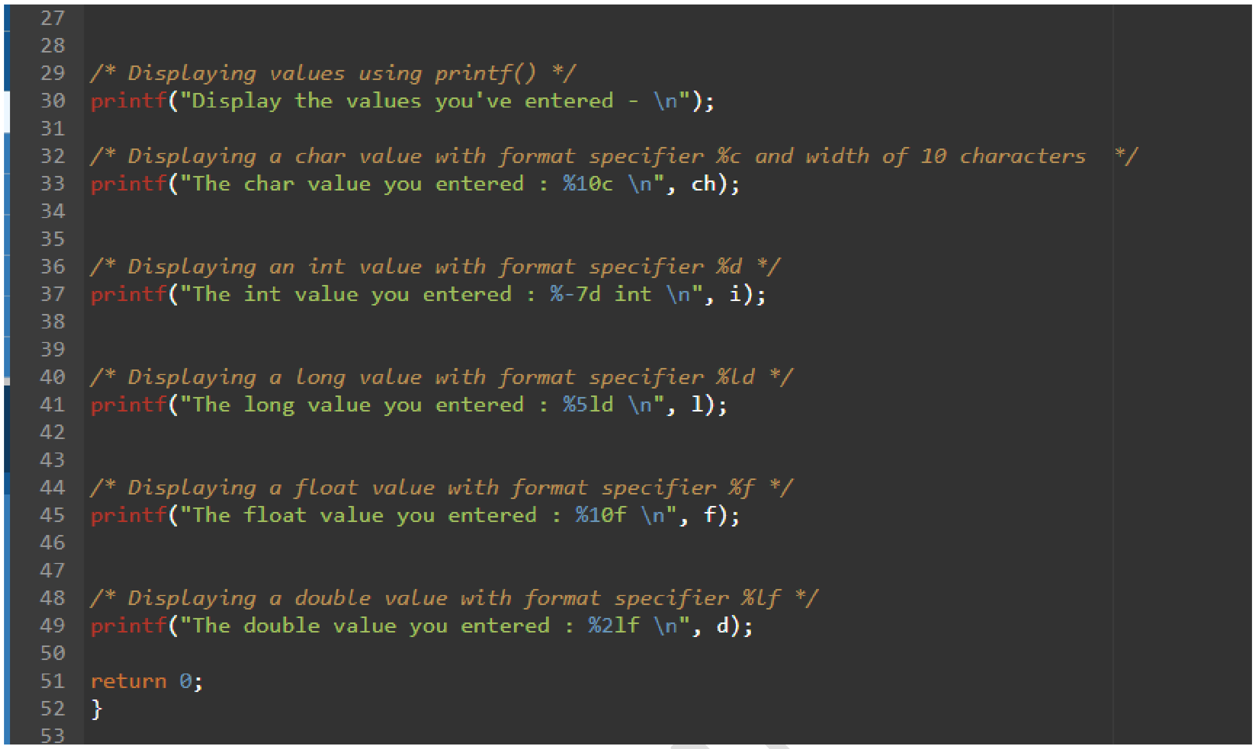
Displaying values with specific width in **printf()**

We can even optionally do the ***left* and *right justification*** of the value to be printed and have them printed in a specific ***width*** using **printf()** function.



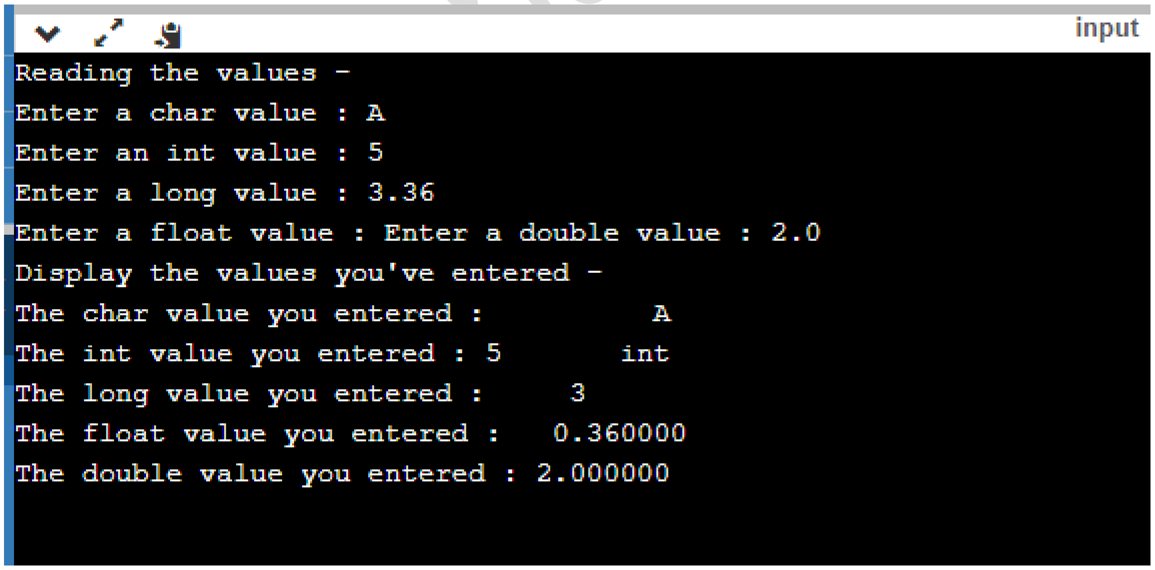
main.c

```c
/*********************************************
Statement - Displaying values with specific width in printf()
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*********************************************/
#include<stdio.h>
int main()
{
char ch; int i;  float f; double d; long l;

/* Reading values using scanf() */
printf("Reading the values - \n");

printf("Enter a char value : ");
scanf("%c",&ch);    /* Reading a char value with format specifier %c */

printf("Enter an int value : ");
scanf("%d",&i);     /* Reading an int value with format specifier %d */

printf("Enter a long value : ");
scanf("%ld",&l);    /* Reading an long value with format specifier %ld */

printf("Enter a float value : ");
scanf("%f",&f);     /* Reading an float value with format specifier %f */

printf("Enter a double value : ");
scanf("%lf",&d);    /* Reading an double value with format specifier %lf */
```



```


/* Displaying values using printf() */
printf("Display the values you've entered - \n");

/* Displaying a char value with format specifier %c and width of 10 characters  */
printf("The char value you entered : %10c \n", ch);


/* Displaying an int value with format specifier %d */
printf("The int value you entered : %-7d int \n", i);


/* Displaying a long value with format specifier %ld */
printf("The long value you entered : %5ld \n", l);


/* Displaying a float value with format specifier %f */
printf("The float value you entered : %10f \n", f);


/* Displaying a double value with format specifier %lf */
printf("The double value you entered : %2lf \n", d);

return 0;
}

```

Output:

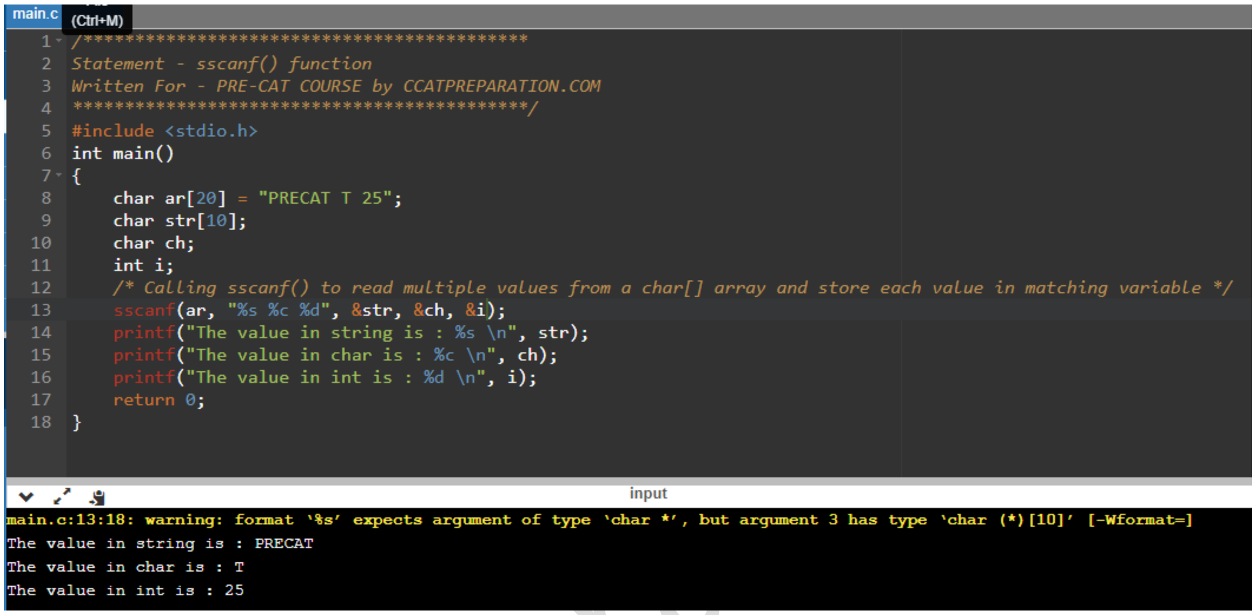


```
Reading the values -
Enter a char value : A
Enter an int value : 5
Enter a long value : 3.36
Enter a float value : Enter a double value : 2.0
Display the values you've entered -
The char value you entered :          A
The int value you entered : 5       int
The long value you entered :     3
The float value you entered :   0.360000
The double value you entered : 2.000000
```

As you may see in the code and its output, we have used multiple **format specifiers** and have also specified a specific **width** of each value to be displayed on the screen, within the **printf()** function.

we are going to explain two more formatted console input/output functions**, sscanf() and sprintf().**

*Example of **sscanf()** function*



```c
/*******************************************
Statement - sscanf() function
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*******************************************/
#include <stdio.h>
int main()
{
    char ar[20] = "PRECAT T 25";
    char str[10];
    char ch;
    int i;
    /* Calling sscanf() to read multiple values from a char[] array and store each value in matching variable */
    sscanf(ar, "%s %c %d", &str, &ch, &i);
    printf("The value in string is : %s \n", str);
    printf("The value in char is : %c \n", ch);
    printf("The value in int is : %d \n", i);
    return 0;
}
```

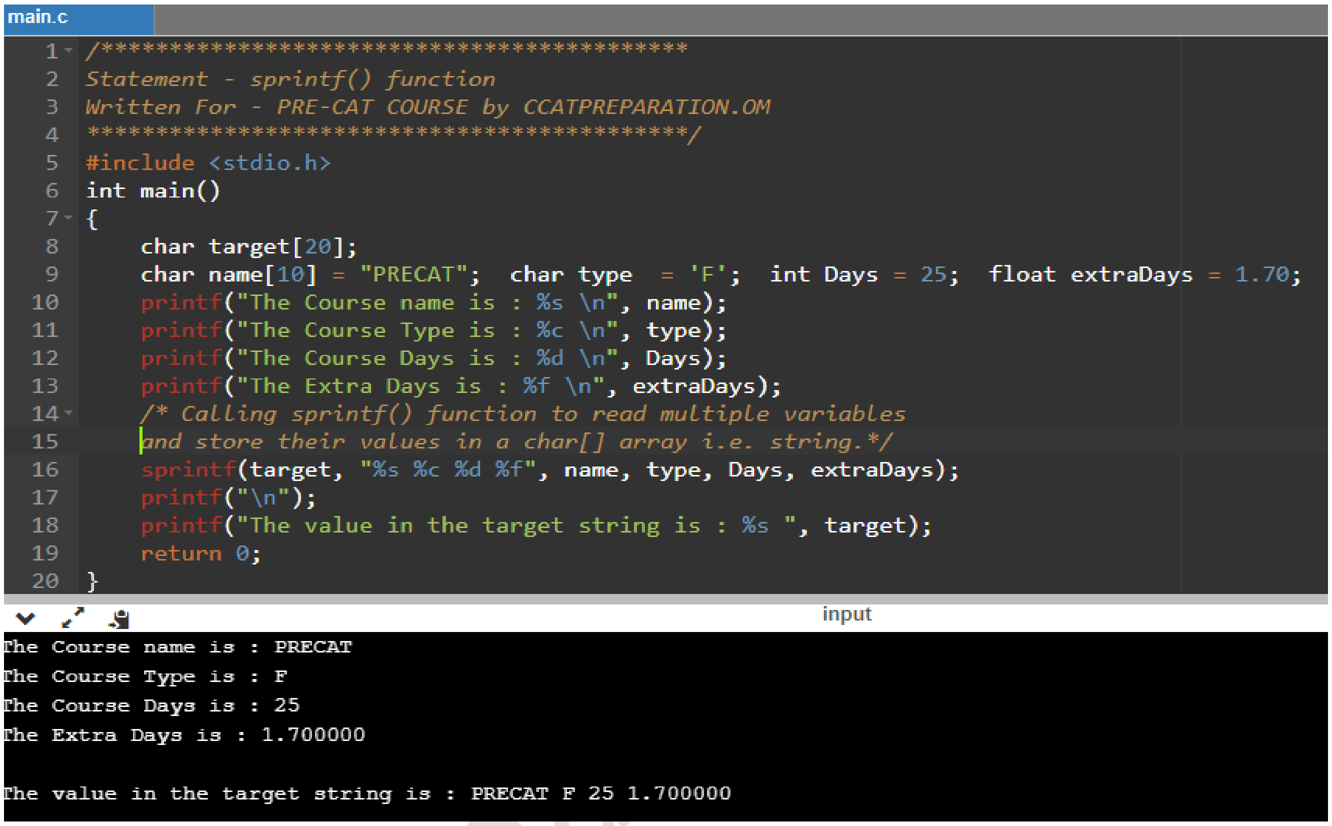
```
main.c:13:18: warning: format '%s' expects argument of type 'char *', but argument 3 has type 'char (*)[10]' [-Wformat=]
The value in string is : PRECAT
The value in char is : T
The value in int is : 25
```

Code Analysis

As you may see in the code and its output, we have called the **sscanf()** function, which linearly reads the value from a char[] array and store each value into variables of matching *data type* by specifying the matching *format specifier* in **sscanf()** function, such as

1. Reads the first value i.e. a **string** by using format specifier **%s** and stores it in **str**.
2. Reads the second value i.e. a **char** by using format specifier **%c** and stores it in **ch**.
3. Reads the third value i.e. a **int** by using format specifier **%d** and stores it in **i**.
4. Reads the fourth value i.e. **float** by using format specifier **%f** and stores it in **f**.

Example of **sprintf** () function



```c
/*********************************************
Statement - sprintf() function
Written For - PRE-CAT COURSE by CCATPREPARATION.OM
**********************************************/
#include <stdio.h>
int main()
{
    char target[20];
    char name[10] = "PRECAT";  char type  = 'F';  int Days = 25;  float extraDays = 1.70;
    printf("The Course name is : %s \n", name);
    printf("The Course Type is : %c \n", type);
    printf("The Course Days is : %d \n", Days);
    printf("The Extra Days is : %f \n", extraDays);
    /* Calling sprintf() function to read multiple variables
    and store their values in a char[] array i.e. string.*/
    sprintf(target, "%s %c %d %f", name, type, Days, extraDays);
    printf("\n");
    printf("The value in the target string is : %s ", target);
    return 0;
}
```

```
The Course name is : PRECAT
The Course Type is : F
The Course Days is : 25
The Extra Days is : 1.700000

The value in the target string is : PRECAT F 25 1.700000
```

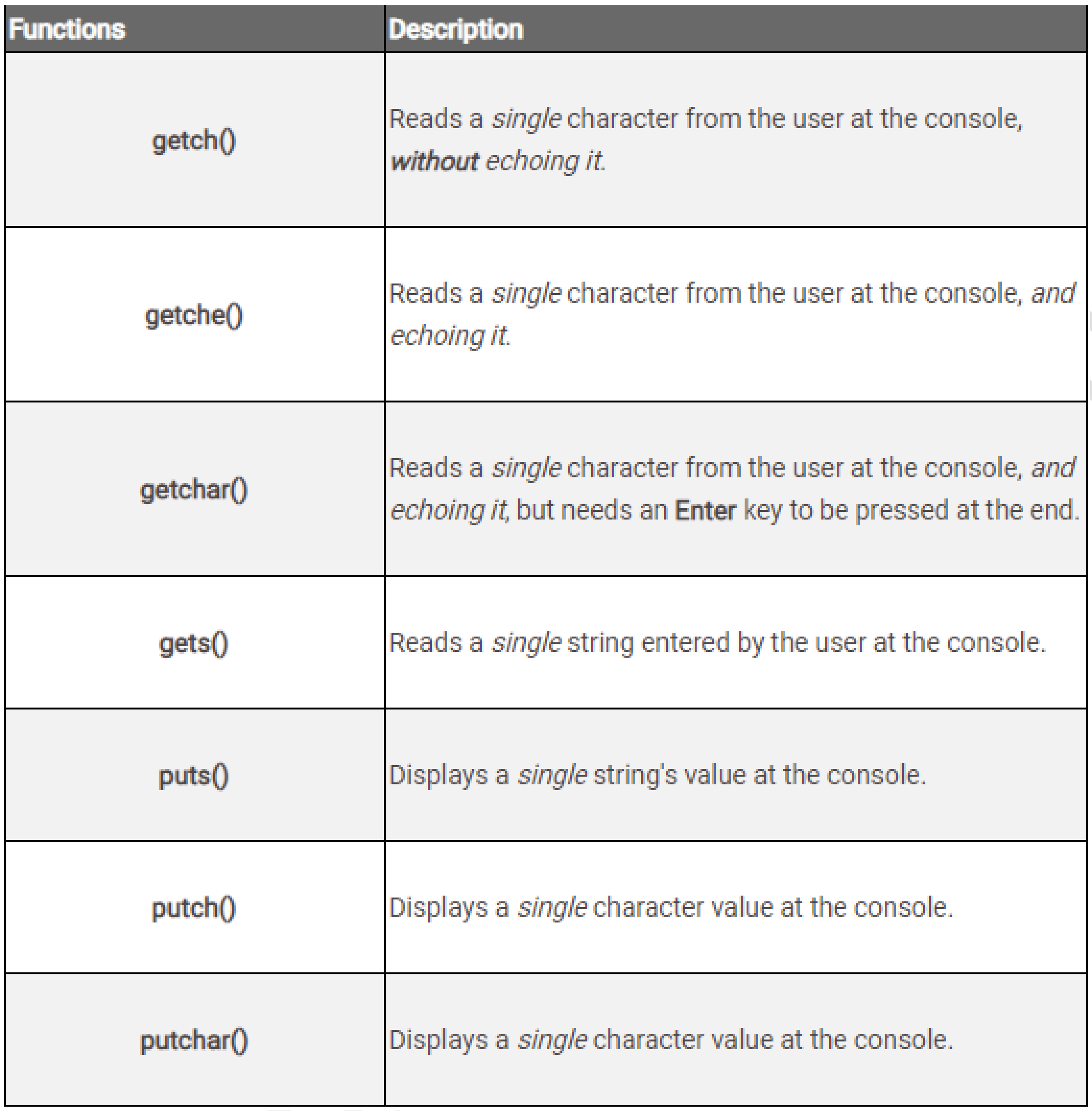
Code Analysis

As you may see in the code and its output, we have called the **sprintf()** function, which linearly reads multiple values by specifying the matching *format specifier* with their *variable names* and stores each all these values in a **char[]** array named **target**.

Unformatted input/output functions

Unformatted console input/output functions are used to read a ***single*** input from the user at console and it also allows us to display the value in the output to the user at the console.

Some of the most important formatted console input/output functions are –

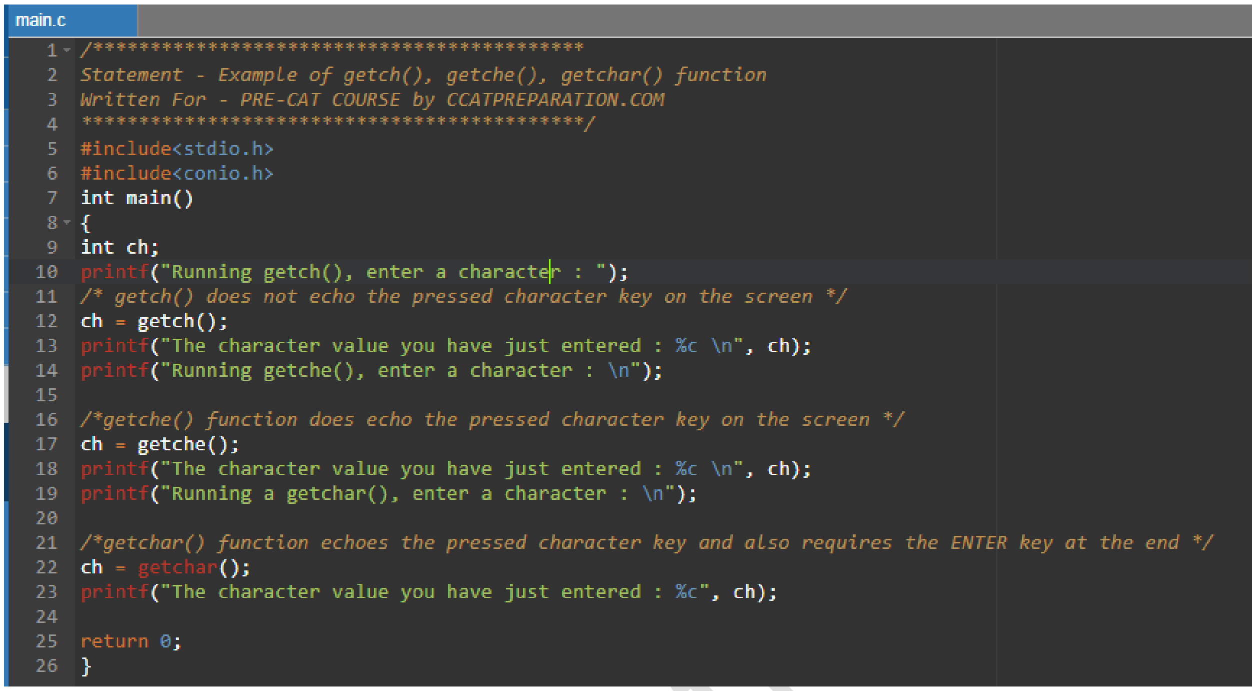


Functions	Description
getch()	Reads a *single* character from the user at the console, ***without** echoing it.*
getche()	Reads a *single* character from the user at the console, *and echoing it.*
getchar()	Reads a *single* character from the user at the console, *and echoing it*, but needs an **Enter** key to be pressed at the end.
gets()	Reads a *single* string entered by the user at the console.
puts()	Displays a *single* string's value at the console.
putch()	Displays a *single* character value at the console.
putchar()	Displays a *single* character value at the console.

Why the functions are called unformatted console I/O functions?

While calling any of the **unformatted** console input/output functions, **we do not have to use any format specifiers in them**, to read or display a value. Hence, these functions are named **unformatted** console I/O functions

Example of getch(), getche(), getchar() function



```c
/*********************************************
Statement - Example of getch(), getche(), getchar() function
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*********************************************/
#include<stdio.h>
#include<conio.h>
int main()
{
int ch;
printf("Running getch(), enter a character : ");
/* getch() does not echo the pressed character key on the screen */
ch = getch();
printf("The character value you have just entered : %c \n", ch);
printf("Running getche(), enter a character : \n");

/*getche() function does echo the pressed character key on the screen */
ch = getche();
printf("The character value you have just entered : %c \n", ch);
printf("Running a getchar(), enter a character : \n");

/*getchar() function echoes the pressed character key and also requires the ENTER key at the end */
ch = getchar();
printf("The character value you have just entered : %c", ch);

return 0;
}
```

Code Analysis

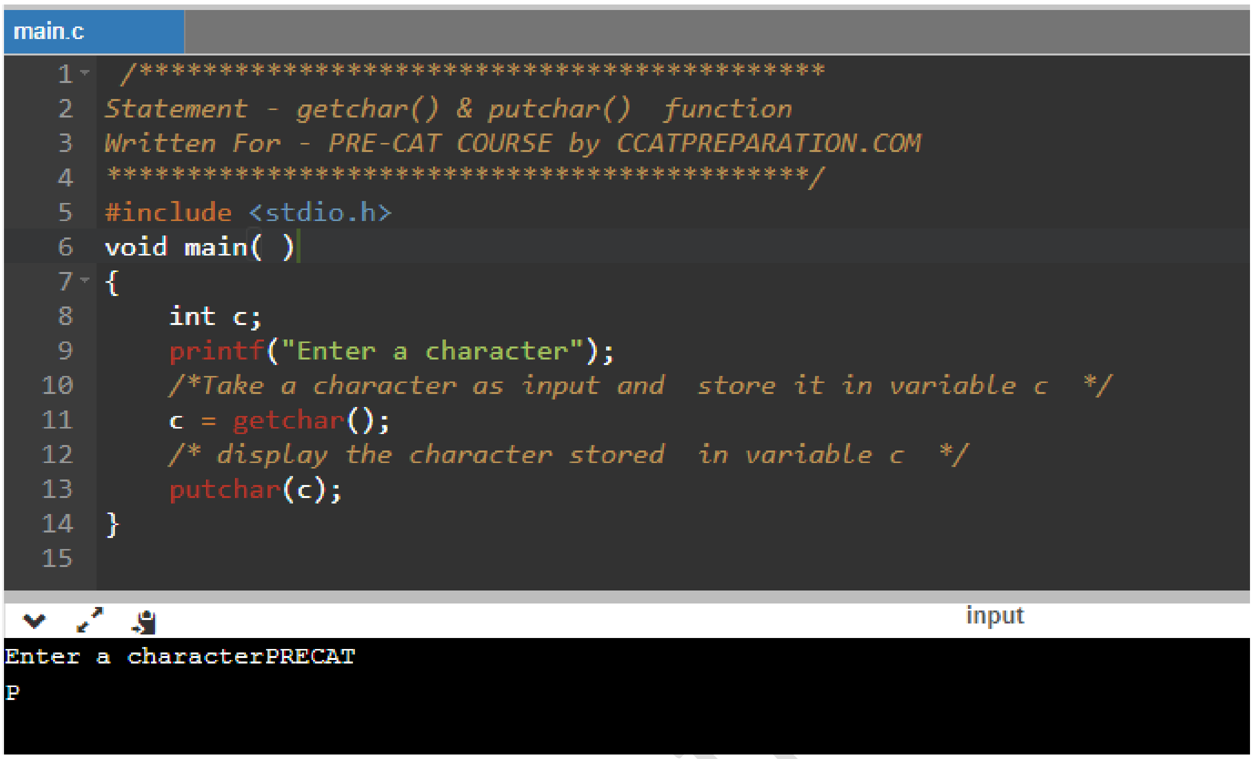
- On pressing the character key **'a'**, the **getch()** stores the character in **ch** but *does not echo it on the screen*.
- On pressing the character key **'g'**, the **getche()** stores the character in **ch** and *echoes it on the screen.*
- On pressing the character key **'w'**, the **getchar()** *echoes it on the screen* but waits for the **Enter** key to be pressed, to store the character in **ch** variable.

getchar() & putchar() functions together

The `getchar()` function reads a character from the terminal and returns it as an integer. This function reads only single character at a time. You can use this method in a loop in case you want to read more than one character.

The `putchar()` function displays the character passed to it on the screen and returns the same character. This function too displays only a single character at a time.

In case you want to display more than one characters, use `putchar()` method in a loop.


main.c

```c
/*********************************************
Statement - getchar() & putchar()  function
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*********************************************/
#include <stdio.h>
void main( )
{
    int c;
    printf("Enter a character");
    /*Take a character as input and  store it in variable c  */
    c = getchar();
    /* display the character stored  in variable c  */
    putchar(c);
}
```

```
Enter a characterPRECAT
P
```

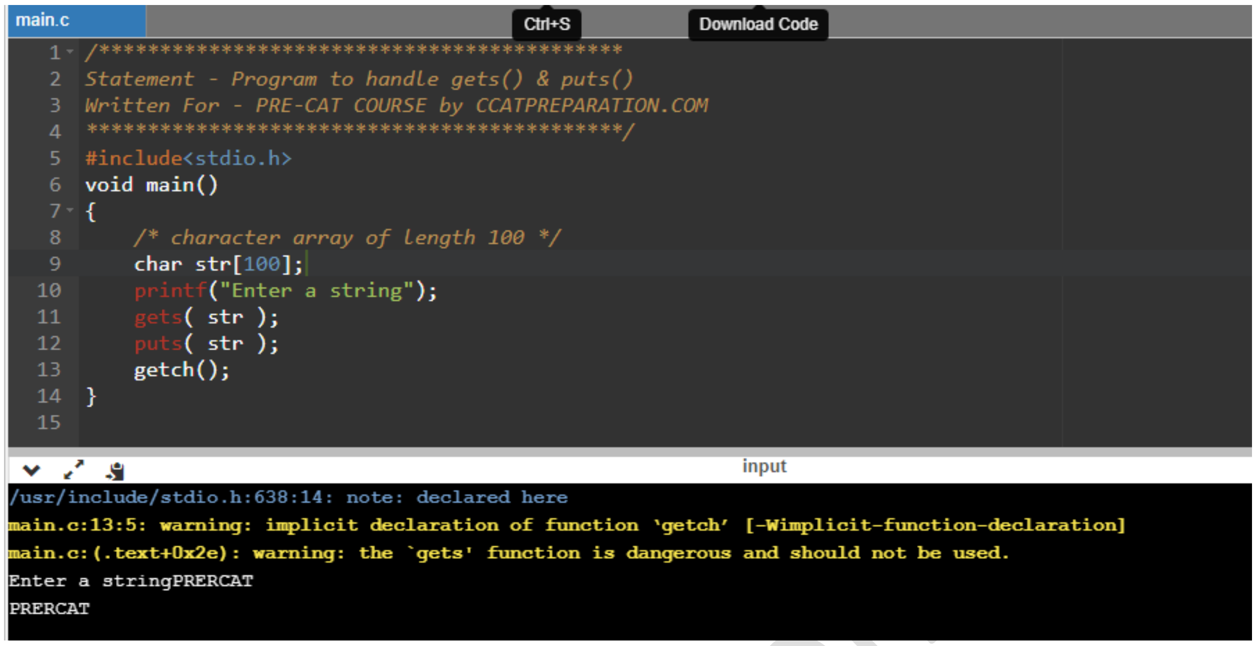
When you will compile the above code, it will ask you to enter a value. When you will enter the value, it will display the value you have entered.

gets() & puts() functions

The `gets()` function reads a line from **stdin**(standard input) into the buffer pointed to by `str` pointer, until either a terminating newline or EOF (end of file) occurs. .

The `puts()` function writes the string `str` and a trailing newline to **stdout**.

`str` → This is the pointer to an array of chars where the C string is stored. (Ignore if you are not able to understand this now.)



```
/*******************************************
Statement - Program to handle gets() & puts()
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*******************************************/
#include<stdio.h>
void main()
{
    /* character array of length 100 */
    char str[100];
    printf("Enter a string");
    gets( str );
    puts( str );
    getch();
}
```

```
/usr/include/stdio.h:638:14: note: declared here
main.c:13:5: warning: implicit declaration of function 'getch' [-Wimplicit-function-declaration]
main.c:(.text+0x2e): warning: the `gets' function is dangerous and should not be used.
Enter a stringPRERCAT
PRERCAT
```

Code Analysis

As you can in the example, the whole *multi-word* string i.e. *name of the course* entered by the user gets stored in the **char** array, using the in-built **gets()** function, which we couldn't achieve using **scanf()** function.

And unlike, **scanf()** method, the **gets()** method does not take any format specifier as a parameter but only takes the *name* of **char** array.

As, we are have read about **gets()** function. In the next tutorial, we are going to explain some **unformatted output functions** with code examples, so please, stay tuned!

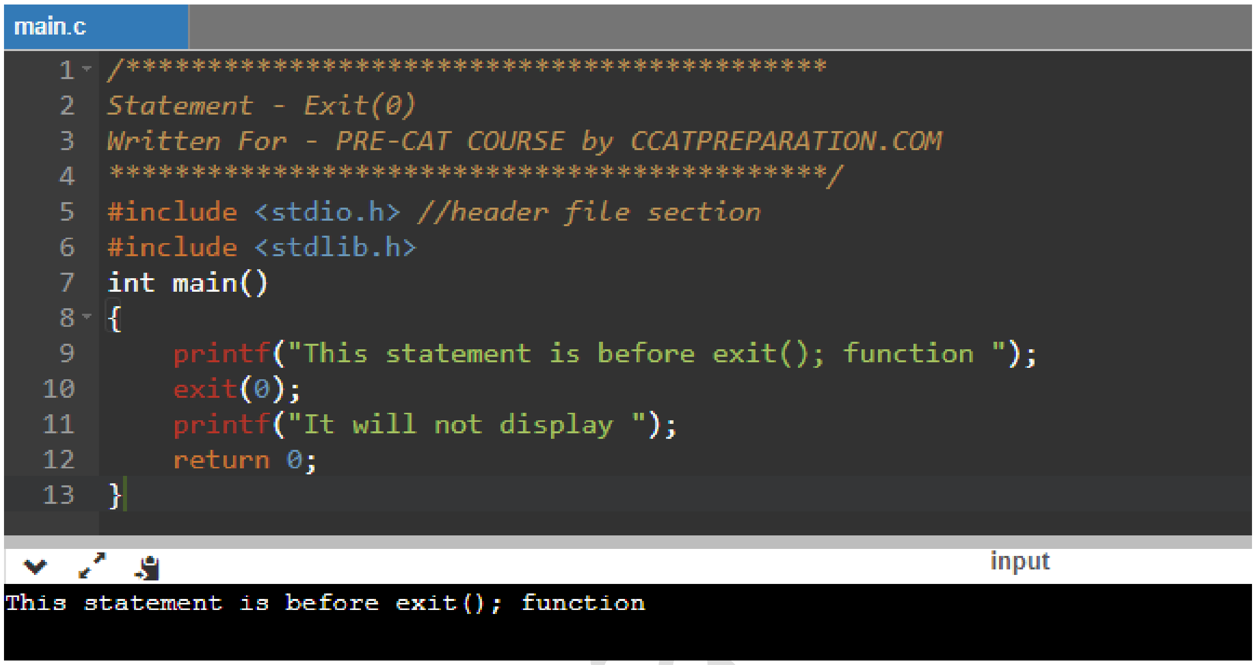
*Difference between **scanf()** and **gets()***

The main difference between these two functions is that `scanf()` stops reading characters when it encounters a space, but `gets()` reads space as character too.

If you enter name as **ccat preparation** using `scanf()` it will only read and store **ccat** and will leave the part after space. But `gets()` function will read it completely.

Example: exit (0)

exit() is an inbuilt library function which is used to terimate the program irrespective of the statements followed by it. **exit()** is defined in #include <stdlib.h> header file.



main.c

```
/*****************************************
Statement - Exit(0)
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*****************************************/
#include <stdio.h> //header file section
#include <stdlib.h>
int main()
{
    printf("This statement is before exit(); function ");
    exit(0);
    printf("It will not display ");
    return 0;
}
```

```
This statement is before exit(); function
```