C Input/Output

C Input/Output
C Input/Output
C Input/Output

Course Title :- Structured Programming Language Sessional

Course Code :- CSE-122

Level Term: 1-II-A(G1) & 1-II-B(G3,G4)

Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

Topics

- ✓Basic Input and Output in C
- ✓Format Specifiers in C
- ✓printf() in C
- ✓scanf() in C
- ✓Formatted and Unformatted Input/Output functions in C with Examples

Basic Input and Output in C

scanf()
The scanf() method, in C, reads the value from the console as per the type specified and store it in the given address.

Syntax:
scanf("%X", &variableOfXType);

where %X is the format specifier in C. It is a way to tell the compiler what type of data is in a variable and & is the address operator in C, which tells the compiler to change the real value of variableOfXType, stored at this address in the memory.

printf()
The printf() method, in C, prints the value passed as the parameter to it, on the console screen.

Syntax:
printf("%X", variableOfXType);
where %X is the format specifier in C. It is a way to tell the compiler what type of data is in a variable and variableOfXType is the variable to be printed.

How To Take Input And Output Of Basic Types In C?

- The basic type in C includes types like int, float, char, etc. Inorder to input or output the specific type, the **X** in the above syntax is changed with the specific format specifier of that type. The Syntax for input and output for these are:

Integer:
Input: scanf("%d", &intVariable);
Output: printf("%d", intVariable);

Float:
Input: scanf("%f", &floatVariable);
Output: printf("%f", floatVariable);

Double:
Input: scanf("%lf", &doubleVariable);
Output: printf("%lf", doubleVariable);

Character:
Input: scanf("%c", &charVariable);
Output: printf("%c", charVariable);

String word:
Input: scanf("%s", &stringVariable);
Output: printf("%s", stringVariable);

String sentence
Input: scanf("%[^\n]s", &stringVariable);
Output: printf("%s", stringVariable);

```
int main()
{
   int num;  float f; double d;

   printf("Enter the integer: ");
   scanf("%d", &num);
   printf("\nEntered integer is: %d", num);

   printf("\n\nEnter the float: ");
   scanf("%f", &f);
   printf("\nEntered float is: %f", f);

   printf("\n\nEnter the double: ");
   scanf("%lf", &d);
   printf("\nEntered float is: %lf", d);
}
```

```
Enter the integer: 45

Entered integer is: 45

Enter the float: 4.5

Entered float is: 4.500000

Enter the double: 4.03

Entered float is: 4.030000
```

```c
// C program to show input and output
#include <stdio.h>
int main()
{
    char ch;
    // Input the Character
    printf("\nEnter the Character: ");
    scanf("%c", &ch);
    // Output the Character
    printf("\nEntered character is: %c\n\n", ch);
    return 0;
}
```

```
Enter the Character: ritu

Entered character is: r
```

```c
// C program to show input and output

#include <stdio.h>
int main()
{
    char str[50];
    // --- String ---To read a word
    // Input the Word
    printf("Enter the Word: ");
    scanf("%s\n", str);

    // Output the Word
    printf("\nEntered Word is: %s", str);

    return 0;
}
```

```
Enter the Word: computer science

Entered Word is: computer
```

```c
// C program to show input and output
#include <stdio.h>
int main()
{
    char str[50];
    // --- String ---To read a Sentence
    // Input the Sentence
    printf("\nEnter the Sentence: ");
    scanf("%[^\n]s", str);

    // Output the String
    printf("\nEntered Sentence is: %s\n\n", str);
    return 0;
}
```

```
Enter the Sentence: we are learning c-programming language

Entered Sentence is: we are learning c-programming language
```

Format Specifiers in C

Format Specifier	Description
%c	For character type.
%d	For signed integer type.
%f	For float type.
%ld or %li	Long
%lf	Double
%Lf	Long double
%lli or %lld	Long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

- **Need for User Input in C**

- Every computer application accepts certain data from the user, performs a predefined process on the same to produce the output. There are no keywords in C that can read user inputs. The standard library that is bundled with the C compiler includes stdio.h header file, whose library function **scanf()** is most commonly used to **accept user input from the standard input stream**. In addition, the **stdio.h** library also provides other functions for accepting input.

- To understand the need for user input, consider the following C program –

```c
#include <stdio.h>
int main()
{
    int price, qty, ttl;

    price = 100;
    qty = 5;
    ttl = price*qty;

    printf("Total : %d", ttl);
    return 0;
}
```

Integer Input

The **%d** format specifier has been defined for signed integer. The following program reads the user input and stores it in the int variable num.

Example: Integer Input in C

```
#include <stdio.h>
int main()
{
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   printf("You entered an integer : %d", num);
   return 0;
}
```

```
Enter an integer: 234
You entered an integer : 234
```

Example: Multiple Integer Inputs in C

```
#include <stdio.h>
int main()
{
   int num1, num2;
   printf("Enter two integers: ");
   scanf("%d %d", &num1, &num2);
   printf("You entered two integers : %d and %d",
num1, num2);
   return 0;
}
```

Output

```
Enter two integers: 45 57
You entered two integers : 45 and 57
```

Float Input:- For floating point input, you need to use %f format specifier.

Example: Float Input in C

```
int main(){
  float num1;
  printf("Enter a number: ");
  scanf("%f", &num1);
  printf("You entered a floating-point number: %f", num1);
}
```

Output

```
Enter a number: 34.56
You entered a floating-point number: 34.560001
```

Example: Integer and Float Inputs in C

```
int main(){
  int num1;
  float num2;
  printf("Enter two numbers: ");
  scanf("%d %f", &num1, &num2);
  printf("You entered an integer: %d a floating-point number: %6.2f", num1, num2);
}
```

Output

```
Enter two numbers: 65 34.5678
You entered an integer: 65 a floating-point number:  34.57
```

Character Input

The %c format specifier reads a single character from the keyboard. However, we must give a blank space before %c in the format string. This is because the %c conversion specifier won't automatically skip any leading whitespace, so if there's a stray newline in the input stream (from a previous entry, for example) the scanf() call will consume it immediately.

scanf(" %c", &c);

The blank in the format string tells scanf to skip leading whitespace, and the first non-whitespace character will be read with the %c conversion specifier.

Example: Character Input in C

```
#include <stdio.h>
int main(){
   char ch;
   printf("enter a single character: ");
   scanf(" %c", &ch);
   printf("You entered character : %c", ch);
}
Output
enter a single character: x
You entered character : x
```

Example: Multiple Character Inputs in C

The following program reads two characters separated by a space in two char variables.

```
int main(){
   char ch1, ch2;
   printf("enter a two characters: ");
   scanf("%c %c", &ch1, &ch2);
   printf("You entered characters : %c and %c", ch1, ch2);
   return 0;
}
```

Output

```
enter a two characters: x y
You entered characters : x and y
```

Example: Character Input Using gets()

```
#include <stdio.h>
int main()
{
  char ch;
  printf("enter a character: ");
  ch = getchar();
  puts("You entered: ");
  putchar(ch);
  printf("\nYou entered character : %c", ch);
  return 0;
}
```

Output

```
enter a character: W
You entered:
W
You entered character : W
```

You can also use the unformatted putchar() function to print a single character.

String Input

There is also a %s format specifier that reads a series of characters into a char array.

Example: String Input Using scanf()

The following program accepts the string input from the keyboard –

```
#include <stdio.h>
int main()
{
   char name[20];
   printf("Enter your name: ");
   scanf("%s", name);
   printf("You entered the name : %s", name);
}
```

Output

```
Enter your name: Ravikant
You entered the name : Ravikant
```

Example: String Input Using gets()

```
#include <stdio.h>
int main()
{
  char name[20];
  printf("Enter your name: ");
  gets(name);
  printf("You entered the name : %s", name);
  return 0;
}
```

Output

```
Enter your name: Ravikant Soni
You entered the name : Ravikant Soni
```

Formatted and Unformatted Input/Output functions in C with Examples

- **Formatted I/O Functions**
- Formatted I/O functions are used to take various inputs from the user and display multiple outputs to the user. These types of I/O functions can help to display the output to the user in different formats using the format specifiers. These I/O supports all data types like int, float, char, and many more.
- **Why they are called formatted I/O?**
  These functions are called formatted I/O functions because we can use format specifiers in these functions and hence, we can format these functions according to our needs.
- The following formatted I/O functions will be discussed in this section-

1. **printf()**
2. **scanf()**
3. **sprintf()**
4. **sscanf()**

```
int main()
{
    int num1;
    // Printing a message on
    // the output screen
    printf("Enter a integer number: ");

    // Taking an integer value
    // from keyboard
    scanf("%d", &num1);

    // Displaying the entered value
    printf("You have entered %d", num1);
}
```

Unformatted Input/Output functions

- Unformatted I/O functions are used only for character data type or character array/string and cannot be used for any other datatype. These functions are used to read single input from the user at the console and it allows to display the value at the console.
- **Why they are called unformatted I/O?**
- These functions are called unformatted I/O functions because we cannot use format specifiers in these functions and hence, cannot format these functions according to our needs.
- The following unformatted I/O functions will be discussed in this section-

1. **getch()**
2. **getche()**
3. **getchar()**
4. **putchar()**
5. **gets()**
6. **puts()**
7. **putch()**

```
#include <conio.h>
#include <stdio.h>
int main()
{
	char name[50];
	puts("Please enter some texts: ");
	gets(name);
	puts(name);
	getch();
}
```

Formatted I/O vs Unformatted I/O

S No.	Formatted I/O functions	Unformatted I/O functions
1	These functions allow us to take input or display output in the user's desired format.	These functions do not allow to take input or display output in user desired format.
2	These functions support format specifiers.	These functions do not support format specifiers.
3	These are used for storing data more user friendly	These functions are not more user-friendly.
4	Here, we can use all data types.	Here, we can use only character and string data types.
5	printf(), scanf, sprintf() and sscanf() are examples of these functions.	getch(), getche(), gets() and puts(), are some examples of these functions.

Program-1:- Find summation and average of 3 number

```
/*
sum = a+b+c
avg = (a+b+c) / 3
*/
```

Program-2:- Find area of rectangle, triangle, circle, square

```
/*
area of rectangle: h * w
area of triangle:(1/2) * b * h
area of circle pi * r * r
area of square: a * a
*/
```

Program-3:- Convert F to C and C to F

```
///Fahrenheit to Celsius : C = (F - 32) × 5/9
///Celsius to Fahrenheit : F = ((9×C)/ 5) + 32
```