C Input Output (I/O)

Wee will learn to use scanf() function to take input from the user, and printf() function to display output to the user.

C Output

In C programming, `printf()` is one of the main output function. The function sends formatted output to the screen. For example,

---

Example 1: C Output

```
#include <stdio.h>
int main()
{
    // Displays the string inside quotations
    printf("C Programming, Welcome to the lab – Hello\n");
    return 0;
}
```

Output

```
C Programming
```

How does this program work?

- All valid C programs must contain the `main()` function. The code execution begins from the start of the `main()` function.
- The `printf()` is a library function to send formatted output to the screen. The function prints the string inside quotations.
- To use `printf()` in our program, we need to include `stdio.h` header file using the `#include <stdio.h>` statement.

- The `return 0;` statement inside the `main()` function is the "Exit status" of the program. It's optional.

Example 2: Integer Output

```
#include <stdio.h>
int main()
{
    int c=30,a=10,b=20;
c = 40+30;
    printf("The value of a, b and c is %d %d”,a b %d”,a,b,c+90);
    return 0;
}
```

Output

```
The value of a,b and c is 10 20 and 40
```

We use `%d` format specifier to print `int` types. Here, the `%d` inside the quotations will be replaced by the value of `testInteger`.

Example 3: float and double Output

```
#include <stdio.h>
int main()
{
    float number1 = 13.5;
    double number2 = 12.4;

    printf("number1 = %f\n", number1);
    printf("number2 = %lf", number2);
    return 0;
}
```

Output

```
number1 = 13.500000
number2 = 12.400000
```

To print `float`, we use `%f` format specifier. Similarly, we use `%lf` to print `double` values.

Example 4: Print Characters

```
#include <stdio.h>
int main()
{
    char ch;
ch = 'a';
    printf("character = %c", chr);
    return 0;
}
```

Output

```
character = a
```

To print `char`, we use `%c` format specifier.

C Input

In C programming, `scanf()` is one of the commonly used function to take input from the user. The `scanf()` function reads formatted input from the standard input such as keyboards.

Example 5: Integer Input/Output

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

Output

```
Enter an integer: 4
Number = 4
```

Here, we have used `%d` format specifier inside the `scanf()` function to take `int` input from the user. When the user enters an integer, it is stored in the `testInteger` variable.

> Notice, that we have used `&testInteger` inside `scanf()`. It is because `&testInteger` gets the address of `testInteger`, and the value entered by the user is stored in that address.

Example 6: Float and Double Input/Output

```
#include <stdio.h>
int main()
{
    float num1;
    double num2;

    printf("Enter a number: ");
    scanf("%f", &num1);
    printf("Enter another number: ");
```

```
    scanf("%lf", &num2);

    printf("num1 = %f\n", num1);
    printf("num2 = %lf", num2);

    return 0;
}
```

Output

```
Enter a number: 12.523
Enter another number: 10.2
num1 = 12.523000
num2 = 10.200000
```

We use `%f` and `%lf` format specifier for `float` and `double` respectively.

Example 7: C Character I/O

```
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c",&chr);
    printf("You entered %c.", chr);
    return 0;
}
```

Output

```
Enter a character: g
You entered g
```

When a character is entered by the user in the above program, the character itself is not stored. Instead, an integer value (ASCII value) is stored.

And when we display that value using `%c` text format, the entered character is displayed. If we use `%d` to display the character, it's ASCII value is printed.

Example 8: ASCII Value

```
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c", &chr);  a

    // When %c is used, a character is displayed
    printf("You entered %c.\n",chr);

    // When %d is used, ASCII value is displayed
    printf("ASCII value is %d.", chr);
    return 0;
}
```

Output

```
Enter a character: g
You entered g.
ASCII value is 103.
```

I/O Multiple Values

Here's how you can take multiple inputs from the user and display them.

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

```
int main()
{
    int a;
    float b;

    printf("Enter integer and then a float: ");

    // Taking multiple inputs
    scanf("%d %f", &a, &b);

    printf("You entered %d and %f", a, b);
    return 0;
}
```

Output

```
Enter integer and then a float: -3
3.4
You entered -3 and 3.400000
```

Format Specifiers for I/O

As you can see from the above examples, we use

- `%d` for `int`
- `%f` for `float`
- `%lf` for `double`
- `%c` for `char`

Here's a list of commonly used C data types and their format specifiers.

Data Type	Format Specifier
`int`	`%d`
`char`	`%c`

Data Type	Format Specifier
`float`	`%f`
`double`	`%lf`
`short int`	`%hd`
`unsigned int`	`%u`
`long int`	`%li`
`long long int`	`%lli`
`unsigned long int`	`%lu`
`unsigned long long int`	`%llu`
`signed char`	`%c`
`unsigned char`	`%c`
`long double`	`%Lf`