printf() and scanf()

Programming in C

Shirley B. Chu
shirley.chu@delasalle.ph

College of Computer Studies
De La Salle University

October 21, 2020

Input and Output

Input is to *provide* the program with some data.

Input and Output

Input is to *provide* the program with some data.

Output is to *display* data on screen or *write* data to a printer or file.

Output function in C

`printf()` statement

- one of the main output functions in C

Output function in C

`printf()` statement

- one of the main output functions in C
- sends formatted output to the screen

Output function in C

`printf()` statement

- one of the main output functions in C
- sends formatted output to the screen
- defined in the header file `stdio.h` (standard input-output header file)

printf() statements

To display a simple string,

```
printf ("Good morning!");
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
```

```
Good morning!_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

```
printf ("\tThis is tricky.\b!");
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

```
printf ("\tThis is tricky.\b!");
```

```
␣␣␣␣␣␣_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

```
printf ("\tThis is tricky.\b!");
```

```
        This is tricky._
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

```
printf ("\tThis is tricky.\b!");
```

```
        This is tricky_
```

printf() statements

To display a simple string,

```
printf ("Good morning!");
printf ("Good afternoon!");
```

```
Good morning!Good afternoon!_
```

```
printf ("Hi!\nHow are you?\n");
```

```
Hi!
How are you?
_
```

```
printf ("\tThis is tricky.\b!");
```

```
        This is tricky!_
```

printf() statements

To display a formatted string, use format specifiers.

syntax: `%specifier`

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is "

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8"

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of "

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10"

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and "

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and 2"

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and 2 is "

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and 2 is 20"

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and 2 is 20."

printf() statements

To display a formatted string, use format specifiers.

syntax: %specifier

```
printf ("The number is %d\n", 8);
```

"The number is 8\n"

```
The number is 8.
_
```

```
int nVal = 10;
printf ("The product of %d and %d is %d.",
        nVal, 2, nVal * 2);
```

"The product of 10 and 2 is 20."

```
The product of 10 and 2 is 20._
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

```
Hail, Hail, Alma Mater_
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

```
Hail, Hail, Alma Mater
_
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

```
Hail, Hail, Alma Mater
Hail to _
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

```
Hail, Hail, Alma Mater
Hail to De La Salle.
We'll hold your banner
_
```

printf() statements

```
printf ("Hail, Hail, Alma Mater\nHail to De La
Salle.\nWe'll hold your banner\nHigh and bright\nA shield
of Green and White");
```

String literals cannot span multiple lines.

```
printf ("Hail, Hail, Alma Mater\nHail to %s%s",
        "De La Salle.\nWe'll hold your banner\n",
        "High and bright\nA shield of Green and White");
```

```
Hail, Hail, Alma Mater
Hail to De La Salle.
We'll hold your banner
High and bright
A shield of Green and White
```

printf() statements

```
char cVal = 'a';
printf ("The character %c", cVal);
```

printf() statements

```
char cVal = 'a';
printf ("The character %c", cVal);
```

```
The character a
```

printf() statements

```
char cVal = 'a';
printf ("The character %c", cVal);
```

```
The character a
```

```
char cVal = '\\';
printf ("The backslash character is %c", cVal);
```

printf() statements

```
char cVal = 'a';
printf ("The character %c", cVal);
```

```
The character a
```

```
char cVal = '\\';
printf ("The backslash character is %c", cVal);
```

```
The backslash character is \
```

printf() statements

```
char cVal = 'A';
printf ("cVal = %c", cVal);
printf ("\ncVal + 5 = %c", cVal + 5);
```

printf() statements

```
char cVal = 'A';
printf ("cVal = %c", cVal);
printf ("\ncVal + 5 = %c", cVal + 5);
```

```
cVal = A_
```

printf() statements

```
char cVal = 'A';
printf ("cVal = %c", cVal);
printf ("\ncVal + 5 = %c", cVal + 5);
```

```
cVal = A
cVal + 5 = F_
```

printf() statements

```
char cVal = 'A';
printf ("cVal = %c", cVal);
printf ("\ncVal + 5 = %c", cVal + 5);
printf ("\ncVal + 5 = %d", cVal + 5);
```

```
cVal = A
cVal + 5 = F
```

printf() statements

```
char cVal = 'A';
printf ("cVal = %c", cVal);
printf ("\ncVal + 5 = %c", cVal + 5);
printf ("\ncVal + 5 = %d", cVal + 5);
```

```
cVal = A
cVal + 5 = F
cVal + 5 = 70
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
printf ("5 / 4 = %f\n", 5 / 4);
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
printf ("5 / 4 = %f\n", 5 / 4);
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
printf ("5 / 4 = %f\n", 5 / 4);
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
printf ("5 / 4 = %f\n", 5 / 4);
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
/* printf ("5 / 4 = %f\n", 5 / 4); */
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
/* printf ("5 / 4 = %f\n", 5 / 4); */
printf ("5.0 / 4 = %d\n", 5.0 / 4);
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
/* printf ("5 / 4 = %f\n", 5 / 4); */
/* printf ("5.0 / 4 = %d\n", 5.0 / 4); */
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
_
```

printf() statements

```
float fAmt = 5;
printf ("Amount = %f\n", fAmt);
printf ("5 / 4 = %d\n", 5 / 4);
/* printf ("5 / 4 = %f\n", 5 / 4); */
/* printf ("5.0 / 4 = %d\n", 5.0 / 4); */
printf ("5.0 / 4 = %f\n", 5.0 / 4);
```

```
Amount = 5.000000
5 / 4 = 1
5.0 / 4 = 1.250000
```

Trace this

```
int nNum = 5;
float fAmt = 2;
char cVal = 'a';

printf ("value = %c\n", cVal);
printf ("value = %d\n", cVal);
printf ("value = cVal\n");

printf ("number = %d\n");
printf ("number = %d\n", nNum);

printf ("amount = %d\n", fAmt);
printf ("amount = %f\n", fAmt);
```

Input function in C

`scanf()` statement

- one of the commonly used functions to take input from the user

Input function in C

`scanf()` statement

- one of the commonly used functions to take input from the user
- is used to read formatted input from the keyboard

Input function in C

`scanf()` statement

- one of the commonly used functions to take input from the user
- is used to read formatted input from the keyboard
- defined in the header file `stdio.h` (standard input-output header file)

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ?

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ?

```
Enter a number: _
```

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ?

```
Enter a number: _
```

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ?

```
Enter a number: 123
_
```

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ~~?~~ 123

```
Enter a number: 123
_
```

scanf() statement

```
int nVal;
printf ("Enter a number:  ");
scanf ("%d", &nVal);
printf ("Value entered is %d\n", nVal);
```

Tracing:

nVal = ~~?~~ 123

```
Enter a number: 123
Value entered is 123
_
```

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ?

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ?

```
Enter a real number: _
```

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ?

```
Enter a real number: _
```

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ?

```
Enter a real number: 12
_
```

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ~~?~~ 12.0

```
Enter a real number: 12
_
```

scanf() statement

```
float fVal;
printf ("Enter a real number:  ");
scanf ("%f", &fVal);
printf ("Value entered is %f\n", fVal);
```

Tracing:

fVal = ~~?~~ 12.0

```
Enter a real number: 12
Value entered is 12.000000
_
```

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ?

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ?

```
Enter a character: _
```

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ?

```
Enter a character: _
```

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ?

```
Enter a character: 1
_
```

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ~~?~~ '1'

```
Enter a character: 1
_
```

scanf() statement

```
char cVal;
printf ("Enter a character:  ");
scanf ("%c", &cVal);
printf ("Value entered is %c\n", cVal);
```

Tracing:

cVal = ~~?~~ '1'

```
Enter a character: 1
Value entered is 1
_
```

scanf() statement

```
int nOne, nTwo;
printf ("Enter two numbers:  ");
scanf ("%d%d", &nOne, &nTwo);
```

scanf() statement

```
int nOne, nTwo;
printf ("Enter two numbers:  ");
scanf ("%d%d", &nOne, &nTwo);
```

```
Enter two numbers: 1␣2
_
```

scanf() statement

```
int nOne, nTwo;
printf ("Enter two numbers:  ");
scanf ("%d%d", &nOne, &nTwo);
```

```
Enter two numbers: 1␣2
_
```

```
Enter two numbers:
1
2
_
```

% character

- % in a String, signals the start of a format specifier.

% character

- % in a String, signals the start of a format specifier.
- To have a % character as part of the String literal, write "%%"

% character

- % in a String, signals the start of a format specifier.
- To have a % character as part of the String literal, write "%%"
- % character is simply written as '%'

☺ Thank you! ☺

Format Specifiers

To display formatted output, include format specifiers

syntax: `%[flag][width][.precision]specifier`

Specifier	Output
`c`	character
`d`	integer
`f`	float
`s`	string

Flags	Description
`-`	left justify
`+`	forces `+` or `-` signs
`0`	left pads with 0

Width	Description
number	minimum number of characters to be printed

Precision	Description
`.`*number*	number of digits after decimal point for floats minimum number of characters to be printed for integers, strings

Skipped inputs

There are instances when different types of inputs (numberic and non-numeric) are collected from the user, some inputs are skipped or not read.

Skipped inputs

Try this!

```
#include <stdio.h>

int main ()
{
    int nVal;
    char cOne, cTwo;

    printf ("Enter a number:  ");
    scanf ("%d", &nVal);

    printf ("Enter a character:  ");
    scanf ("%c", &cOne);

    printf ("Enter another character:  ");
    scanf ("%c", &cTwo);

    printf ("Values entered are %d, %c, %c\n", nVal, cOne, cTwo);
    return 0;
}
```

Skipped inputs

Update your code, see what happens

```
#include <stdio.h>

int main ()
{
    int nVal;
    char cOne, cTwo;

    printf ("Enter a number:  ");
    scanf ("%d", &nVal);

    printf ("Enter a character:  ");
    scanf (" %c", &cOne);

    printf ("Enter another character:  ");
    scanf (" %c", &cTwo);

    printf ("Values entered are %d, %c, %c\n", nVal, cOne, cTwo);
    return 0;
}
```

Skipped inputs

- If inputs are skipped when reading with `%c`, use an explicit space in the format to skip white space first.
- When reading with `%c` format, by default, the usual skipping of leading white space is suppressed.