# CS102/IT102
# Computer Programming I

## Lecture 6:
## Formatted Input/Output

Bicol University College of Science
CSIT Department
1st Semester, 2023-2024

# Topics

- Streams
- Formatted input
- Formatted output

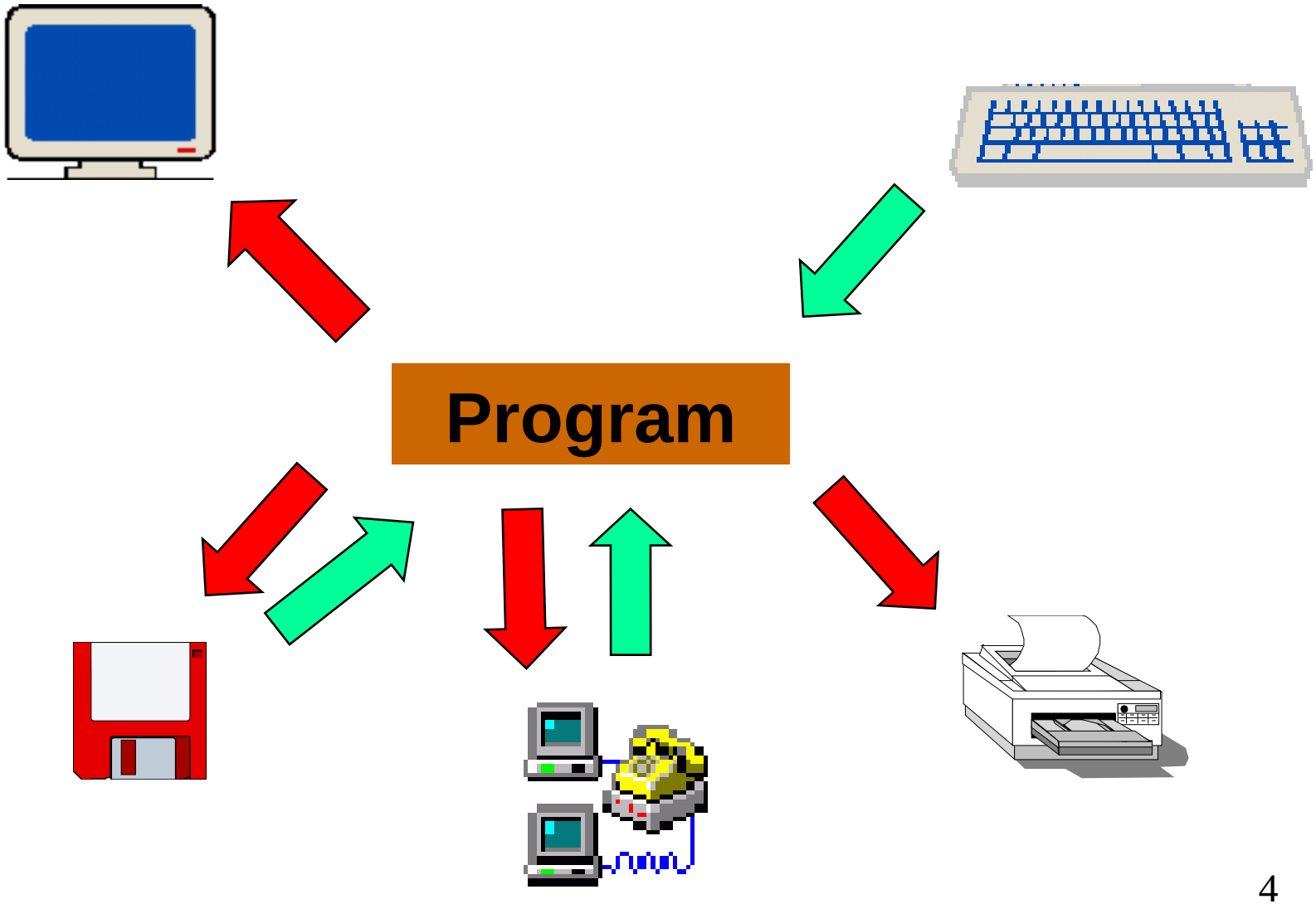
# Recall

- **`scanf()`**
  Example:
  **`scanf(“%d”, &x);`**
- **`printf()`**
  Example:

**`printf(“The value of x is %d\n”, x);`**

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

# Input/Output



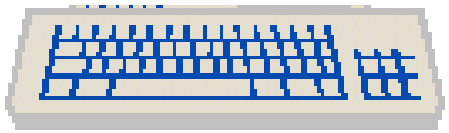
# Streams

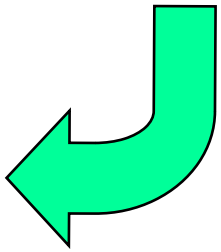
- Text input or output is dealt with as a sequence of characters.
- A stream serves as a channel to convey characters between I/O and programs.

# Streams: Input -- Example

```
int   item;
float cost;
scanf("%d %f", &item, &cost);
```

```
135 25.5
_
```



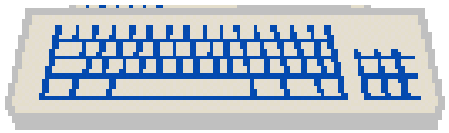



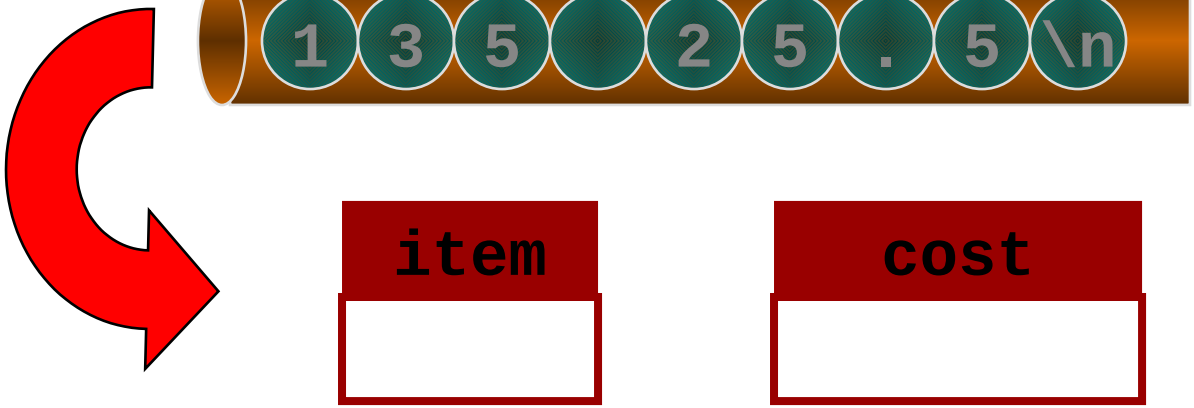
***input buffer***

# Streams: Input -- Example (cont)

```
int   item;
float cost;
scanf("%d %f", &item, &cost);
```

```
135 25.5
_
```

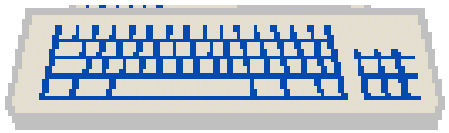


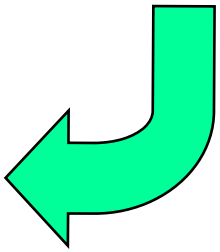


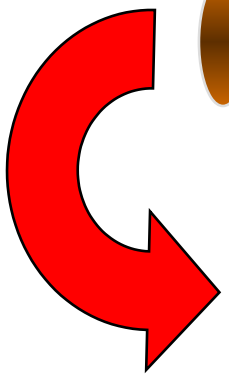
# Streams: Input – Example (cont)

```
int   item;
float cost;
scanf("%d %f", &item, &cost);
```

```
135 25.5
_
```



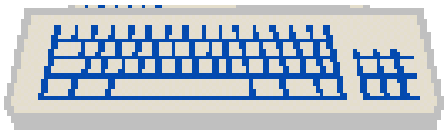


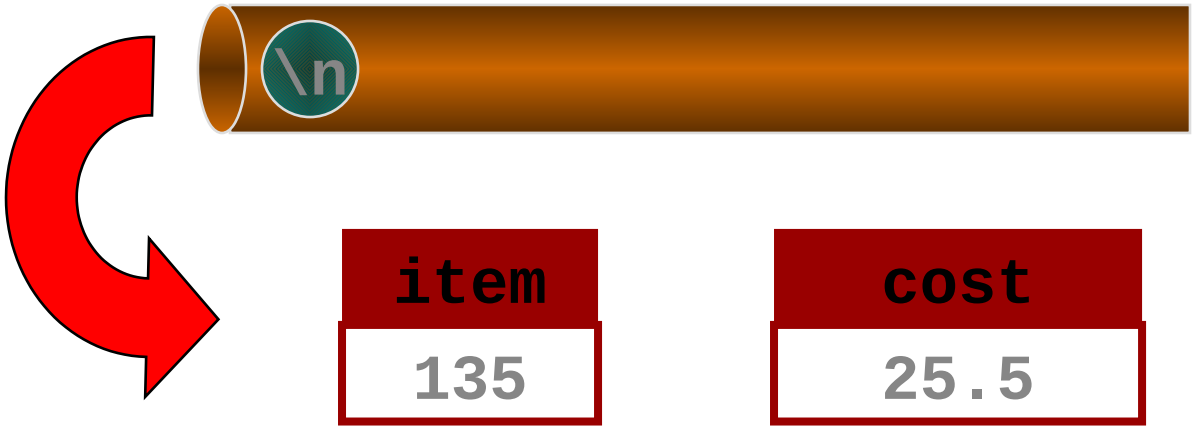
| item | cost |
|---|---|
| 135 | |

# Streams: Input – Example (cont)

```
int   item;
float cost;
scanf("%d %f", &item, &cost);
```

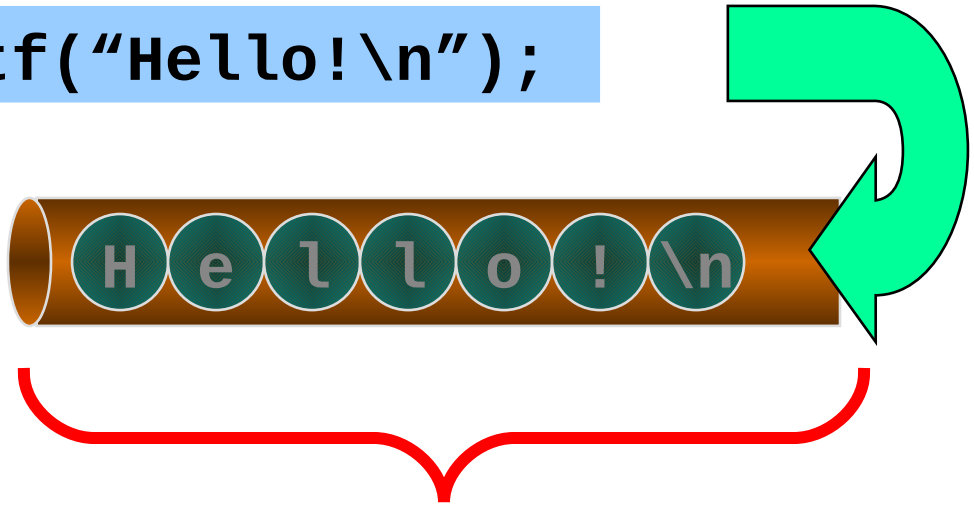
```
135 25.5
_
```





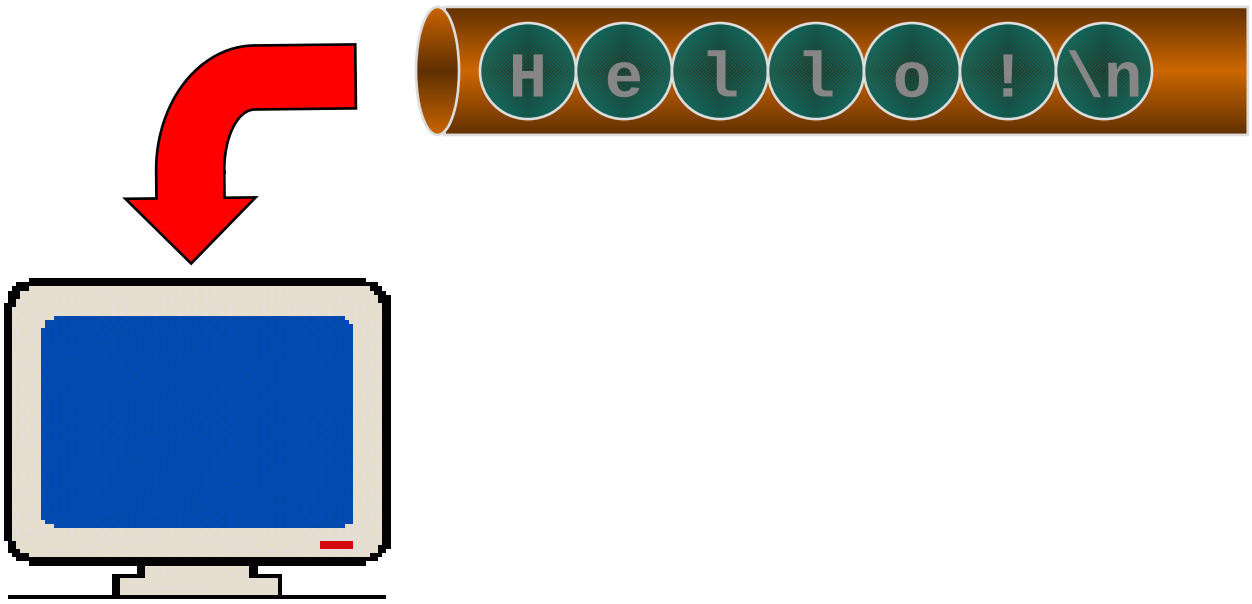
# Streams: Output -- Example

```
printf("Hello!\n");
```



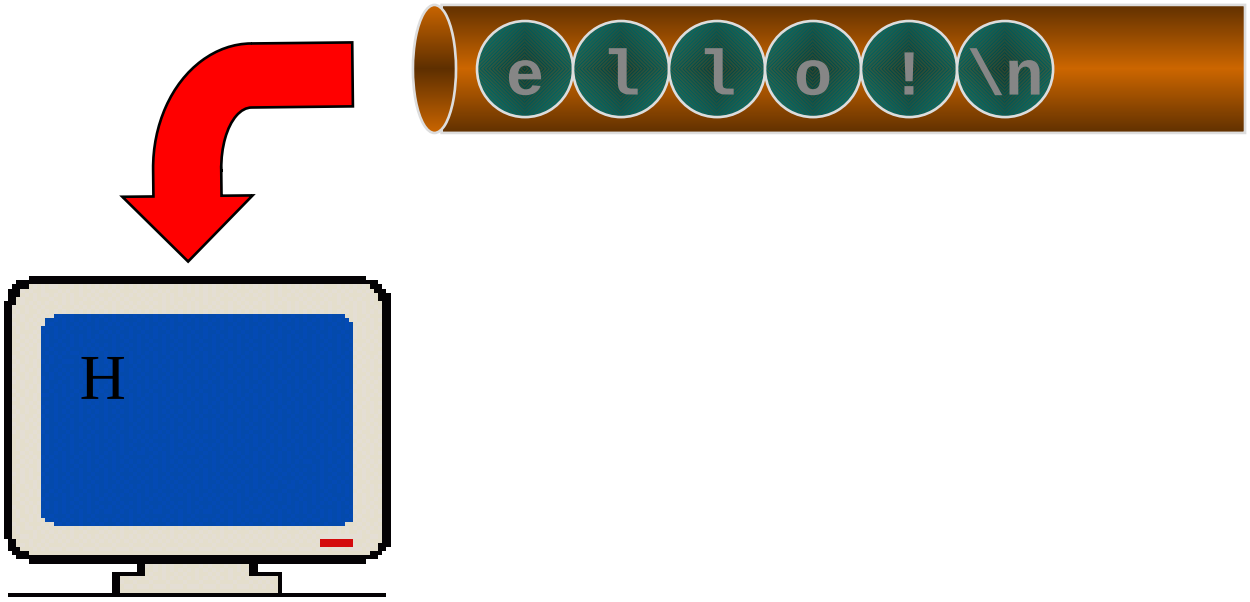
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



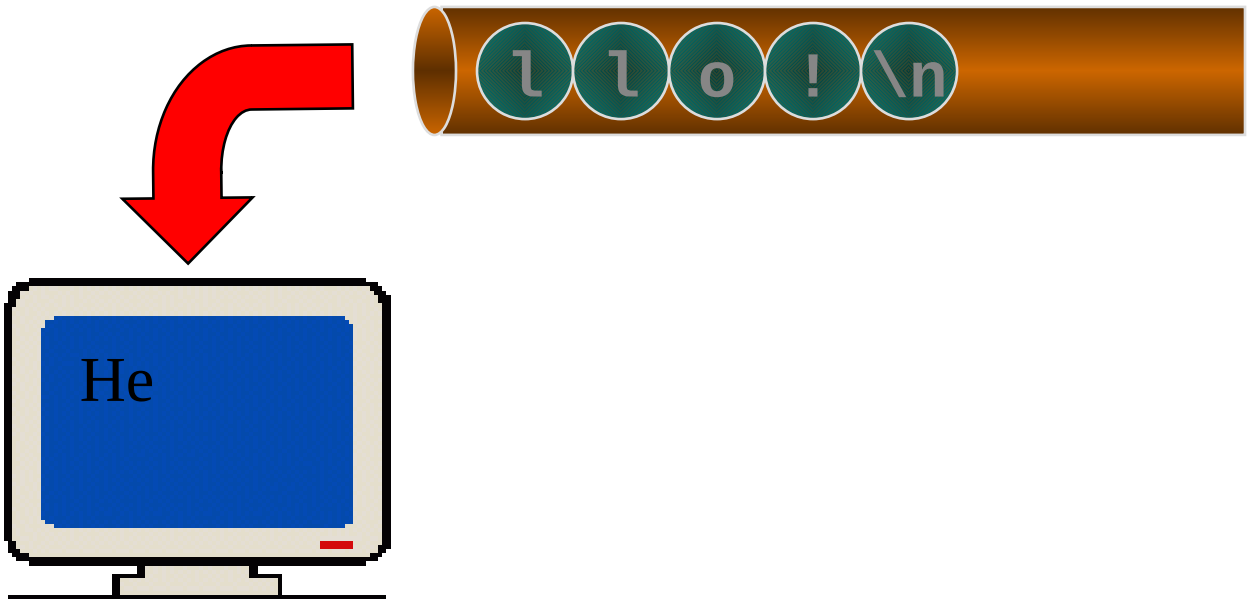
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



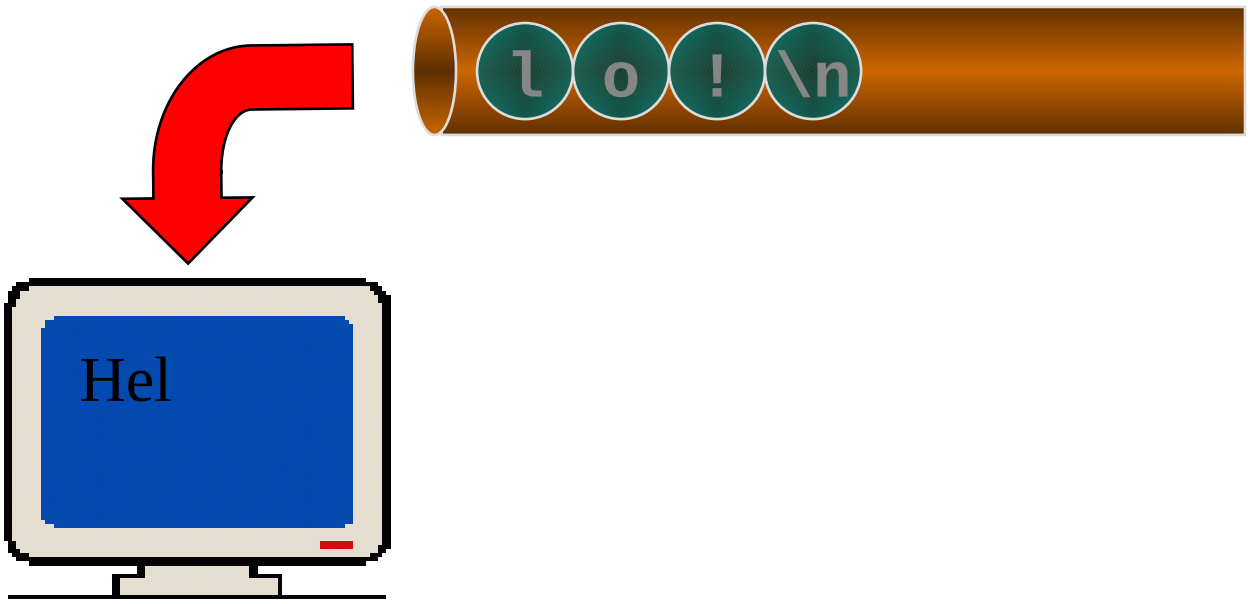
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



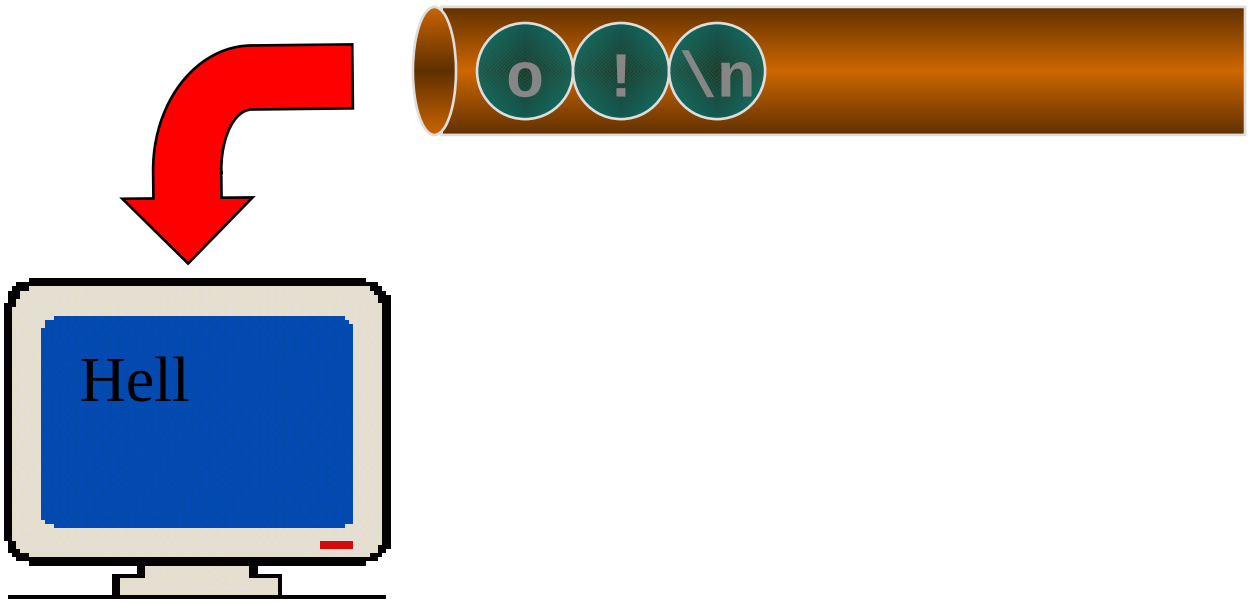
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



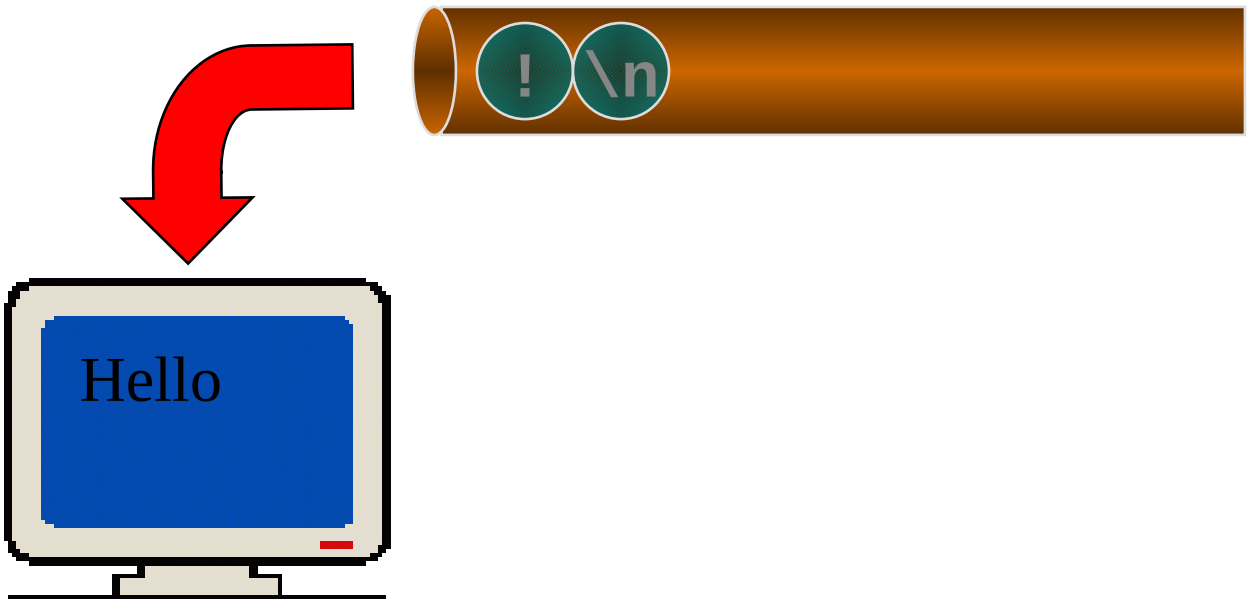
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



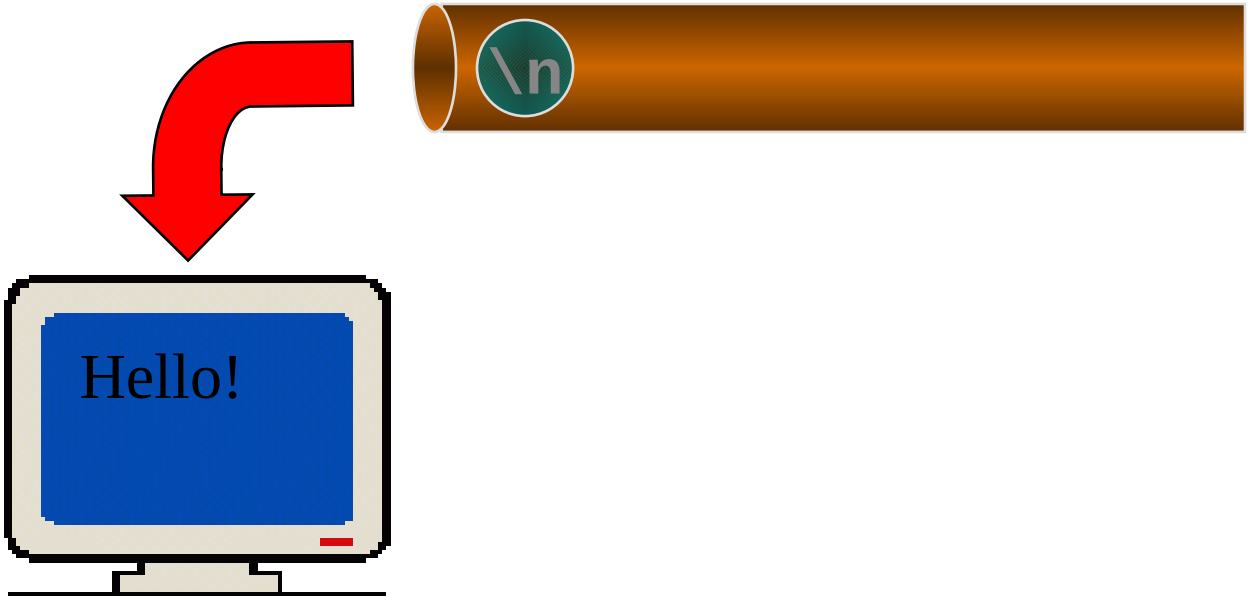
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



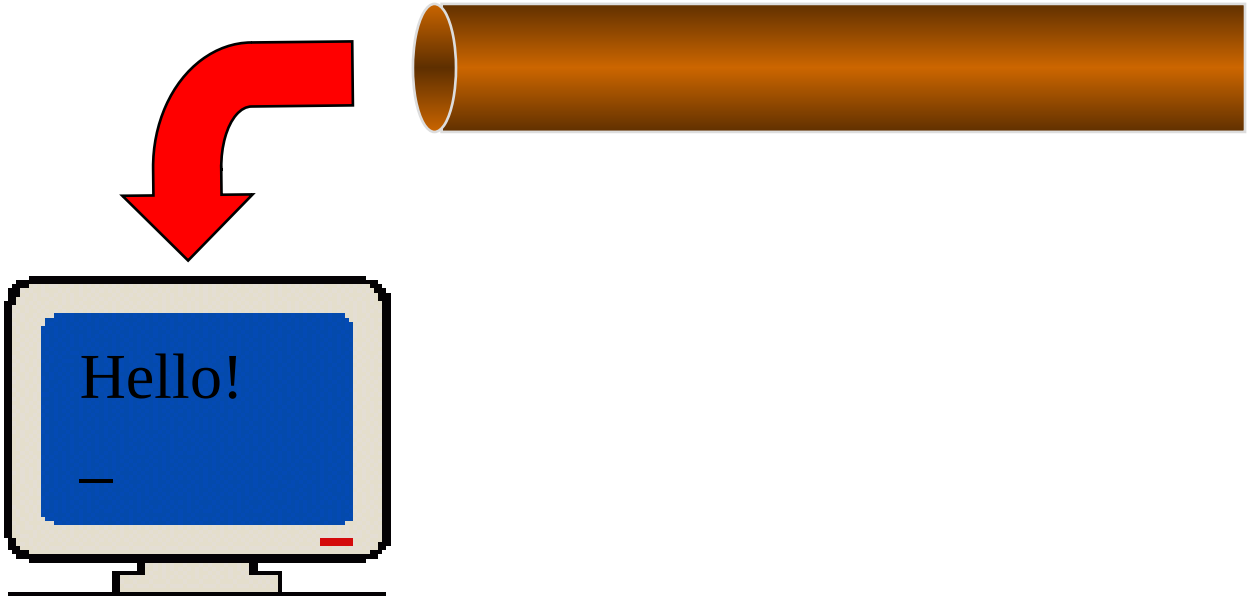
# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



# Streams: Output – Example (cont)

```
printf("Hello!\n");
```



# Streams

- From the program's point of view, the characters are queued in a pipe
- The sequence of characters is organized into lines
- Each line:
  - can have zero or more characters
  - ends with the "newline" character **`'\n'`**

# "Standard" Streams

- Standard streams:
  - **`stdin`** - standard input
    - usually from keyboard
  - **`stdout`** - standard output
    - usually to screen
  - **`stderr`** - standard error
    - usually to screen
- must have at the top of your program

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

- can be ***redirected***

# `stdin`: Input

- Data is read in from **`stdin`** (into a variable) using the **`scanf()`** function
- When input ends, the **`scanf()`** function returns a special value: **`EOF`**

# Example: ReadData

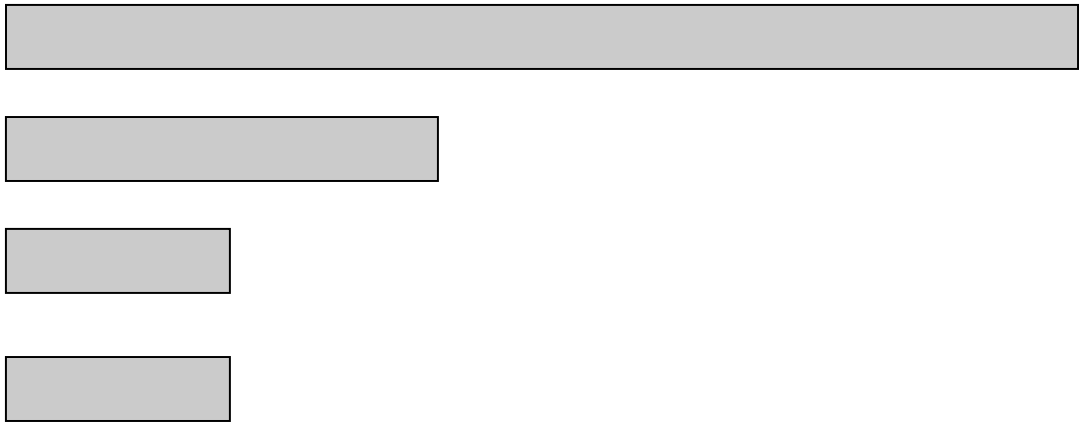
Input name, age, gender, idNumber

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

```
#include <stdio.h>
 /**************************************\
 Read in important info about a lecturer
\**************************************/
```

```
#include <stdio.h>
 /*************************************\
 Read in important info about a lecturer
\*************************************/
int main()
{









 return 0;
}
```

```
#include <stdio.h>
 /*************************************\
 Read in important info about a lecturer
\*************************************/
int main()
{
 char name[100] ;
 float age ;
 char gender ;
 int idNumber ;


 return 0;
}
```



```
#include <stdio.h>
 /*************************************\
 Read in important info about a lecturer
\*************************************/
int main()
{
 char name[100] ;
 float age ;
 char gender ;
 int idNumber ;

 scanf("%s %f %c %d", name, &age, &gender, &idNumber);
 return 0;
}
```

```
#include <stdio.h>
 /*************************************\
 Read in important info about a lecturer
\*************************************/
int main()
{
 char name[100] ;
 float age ;
 char gender ;
 int idNumber ;

 scanf("%s %f %c %d", name, &age, &gender, &idNumber);
  return 0;
}
```

```
#include <stdio.h>
 /*************************************\
 Read in important info about a lecturer
\*************************************/
int main()
{
 char name[100] ;
 float age ;
 char gender ;
 int idNumber ;

 scanf("%s %f %c %d", name, &age, &gender, &idNumber);
 return 0;
}
```

Joey

22.5

M

3825

**Input: Joey 22.5 M 3825**

# `stdout`: Output

- Data (e.g., from a variable) is written out to **`stdout`** using the **`printf()`** function.

# Example: Write Data

Set name to “Joey”

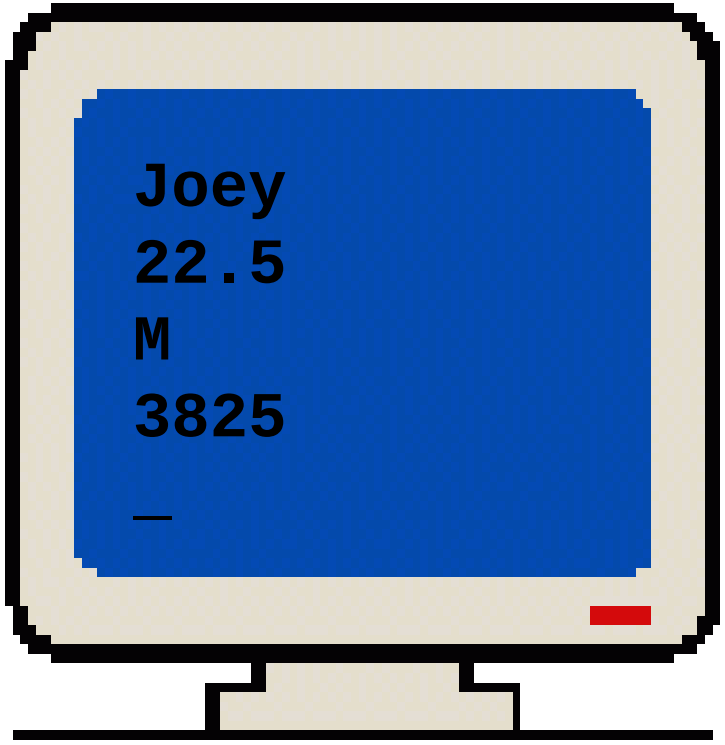
Set age to 22.5

Set gender to ‘M’

Set idNumber to 3825

Output name, age, gender, idNumber

```c
#include <stdio.h>

/*****************************************\
  Write out important info about a lecturer
\*****************************************/

int main()
{
  char   *name    = "Joey" ;
  float  age      = 22.5;
  char   gender   = 'M';
  int    idNumber = 3825 ;

  printf("%s\n%f\n%c\n%d\n", name, age, gender, idNumber);
  return 0;
}
```



# Formatted Input and Output

- General form:

```
printf(format-control-string, other-arguments);
scanf(format-control-string, other-arguments);
```

- Examples:

```
printf("%s\n%f\n%c\n%d\n", name, age, gender, idNumber);
scanf("%s %f %c %d", name, &age, &gender, &idNumber);
```

# printf -- Format-Control-String

- Describes the format of the data for output
- Contains “conversion specifiers” and “literal characters”

Example:

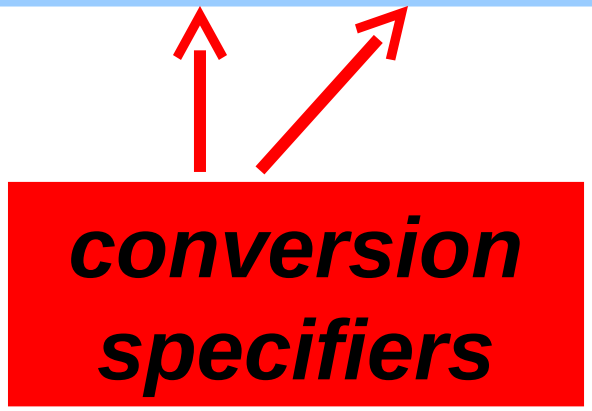
```
printf(“%s is %d years old.\n”, name, age);
```

# printf -- Format-Control-String (cont)

- Describes the format of the data for output
- Contains “conversion specifiers” and “literal characters”

Example:

```
printf(“%s is %d years old.\n”, name, age);
```

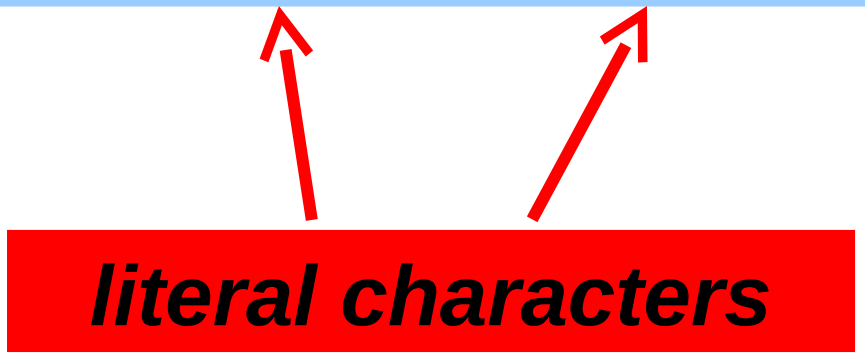


# printf -- Format-Control-String (cont)

- Describes the format of the data for output
- Contains “conversion specifiers” and “literal characters”

Example:

```
printf(“%s is %d years old.\n”, name, age);
```

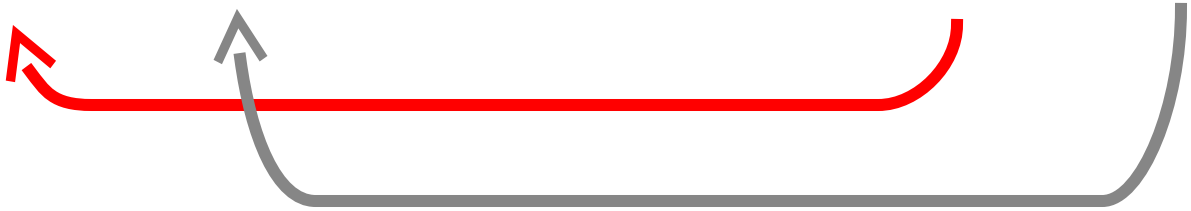


# printf -- Other-Arguments

- For printf: variables containing data for output

Example:

```
printf("%s is %d years old.\n", name, age);
```



# scanf -- Format-Control-String

- Describes the format of the data given as input
- Contains “conversion specifiers”

Example:

```
scanf("%s %f %c %d", name, &age, &gender,&id);
```

# scanf -- Other-Arguments

- For scanf: “pointers” to variables in which the input will be stored.

Example:

```
scanf("%s %f %c %d", name, &age, &gender,&id);
```

# scanf -- Other-Arguments (cont)

- For scanf: “pointers” to variables in which the input will be stored

Example:

```
scanf("%s %f %c %d", name, &age, &gender,&id);
```

- Variables of type int, float or char need ‘&’
- Do NOT use ‘&’ with strings!
- ‘&’ is for **`scanf`** only!

# Common Conversion Specifiers for Numerical Information

- decimal integer: **%d**

```
printf("What is %d plus %d?\n", x, y);
scanf("%d", &sum);
```

- float: **%f**

```
printf("%f squared is...? ", x);
scanf("%f", &ans);
```

- double:

```
printf("%f squared is...? ", x);
scanf("%lf", &ans);
```

# Conversion Specifiers for Alphanumeric Information

- char: **%c**

```
printf("What letter follows %c?\n",ch);
scanf("%c", &nextchar);
```

- string: **%s**

```
printf("Name: %s\n", name);
scanf("%s", name);
```

# printf: Conversion Specifiers

- `i` or `d`: display a signed decimal integer
- f: display a floating point value
- e or E: display a floating point value in exponential notation
- g or G: display a floating point value in either f form or e form
- L: placed before any float conversion specifier to indicate that a long double is displayed

# printf: Conversion Specifiers

%[flags][width][. precision][length]<type>

**type:**

| type | meaning | example |
|---|---|---|
| d,i | integer | printf ("%d",10); /*prints 10*/ |
| x,X | integer (hex) | printf ("%x",10); /*print 0xa*/ |
| u | unsigned integer | printf ("%u",10); /*prints 10*/ |
| c | character | printf ("%c",'A'); /*prints A*/ |
| s | string | printf ("%s","hello"); /*prints hello*/ |
| f | float | printf ("%f",2.3); /* prints 2.3*/ |
| d | double | printf ("%d",2.3); /* prints 2.3*/ |
| e,E | float(exp) | 1e3,1.2E3,1E−3 |
| % | literal % | printf ("%d %%",10); /*prints 10%*/ |

# printf: Conversion Specifiers

%[flags ][width ][. precision ][ modifier]<type>

**width:**

| format | output |
|---|---|
| printf ("%d",10) | "10" |
| printf ("%4d",10) | bb10 (b:space) |
| printf ("%s","hello") | hello |
| printf ("%7s","hello") | bbhello |

The **width** modifier is used to specify the minimum number of position in the output. (If the data require using more space to print everything, then printf will override the width.) If you do not use a width modifier, each output value will take just enough room for the data.

# printf: Conversion Specifiers

%[flags][width][. precision][ modifier]<type>

**precision:**

| format | output |
|---|---|
| printf ("%.2f,%.0f,1.141,1.141) | 1.14,1 |
| printf ("%.2e,%.0e,1.141,100.00) | 1.14e+00,1e+02 |
| printf ("%.4s","hello") | hell |
| printf ("%.1s","hello") | h |

If a floating point number is being printed, the **precision modifier** specifies the number of decimal places to be printed. It has the format .p where p is the number of decimal digits. If no precision is specified, *printf* prints *six* decimal positions.

- When both width and precision are used, the width must be large enough to contain the integral value of the number, the decimal point, and the number of digits in the decimal part.

# printf: Conversion Specifiers

%[flags ][width ][. precision ][ modifier]<type>

**flag:**

| format | output |
|---|---|
| printf ("%d,%+d,%+d",10,–10) | 10,+10,-10 |
| printf ("%04d",10) | 0010 |
| printf ("%7s","hello") | bbhello |
| printf ("%-7s","hello") | hellobb |

The **flag** can be used to specify two print modifications.

- If the flag is zero (0) and there is a width specification, then a number will be printed with leading zeros.
- If the flag is minus sign (-), then the data are formatted left justified.
- Without flag, the data are formatted right justified.

# printf: Conversion Specifiers

%[flags ][width ][. precision ][ modifier]<type>

**modifier:**

| modifier | meaning |
|---|---|
| h | interpreted as short. Use with i,d,o,u,x |
| l | interpreted as long. Use with i,d,o,u,x |
| L | interpreted as double. Use with e,f,g |

The **modifier** is used to modify the type specified by the conversion code.

# scanf: Conversion Specifiers

- `d`: read an optionally signed decimal integer
- `i`: read an optionally signed decimal, octal, or hexadecimal integer

`i` and `d`: the argument is a “pointer” to an integer

```
int idNumber;
scanf("%d", &idNumber);
```

# scanf: Conversion Specifiers (cont)

- `h` or `l`: placed before any integer conversion specifiers to indicate that a short or long integer is to be input

```
long int idNumber;
scanf("%ld", &idNumber);
```

- `l` or `L`: placed before any float conversion specifiers to indicate that a double or long double is to be input

# Conversion Example

Input octal integer

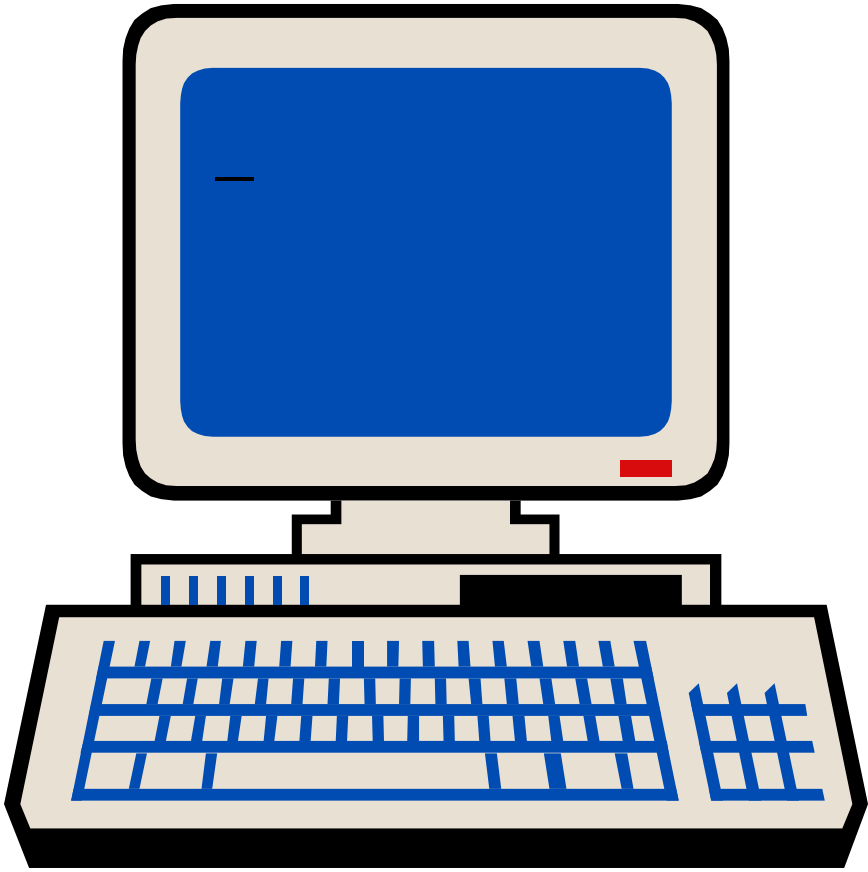
Output integer as decimal

# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
   int i ;

   scanf("%o", &i);
   printf("%d\n", i);
   return 0;
}
```

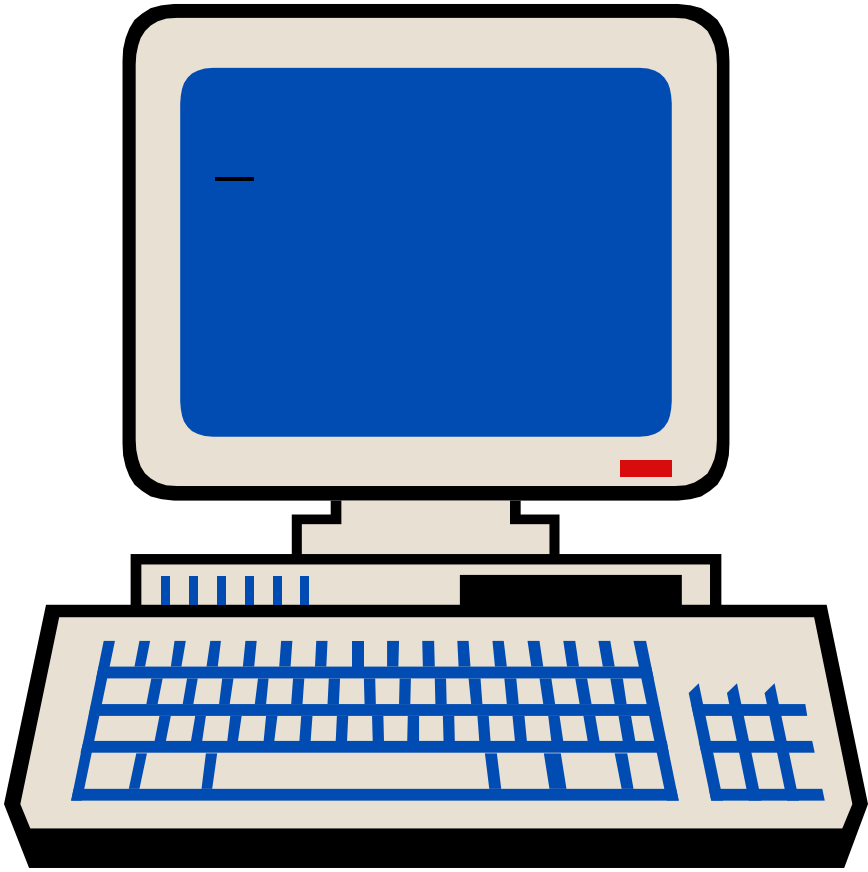
# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
   int i ;

   scanf("%o", &i);
   printf("%d\n", i);
   return 0;
}
```



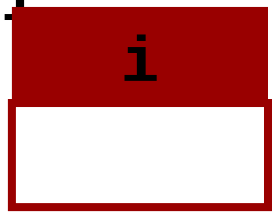
# Conversion Example (cont)

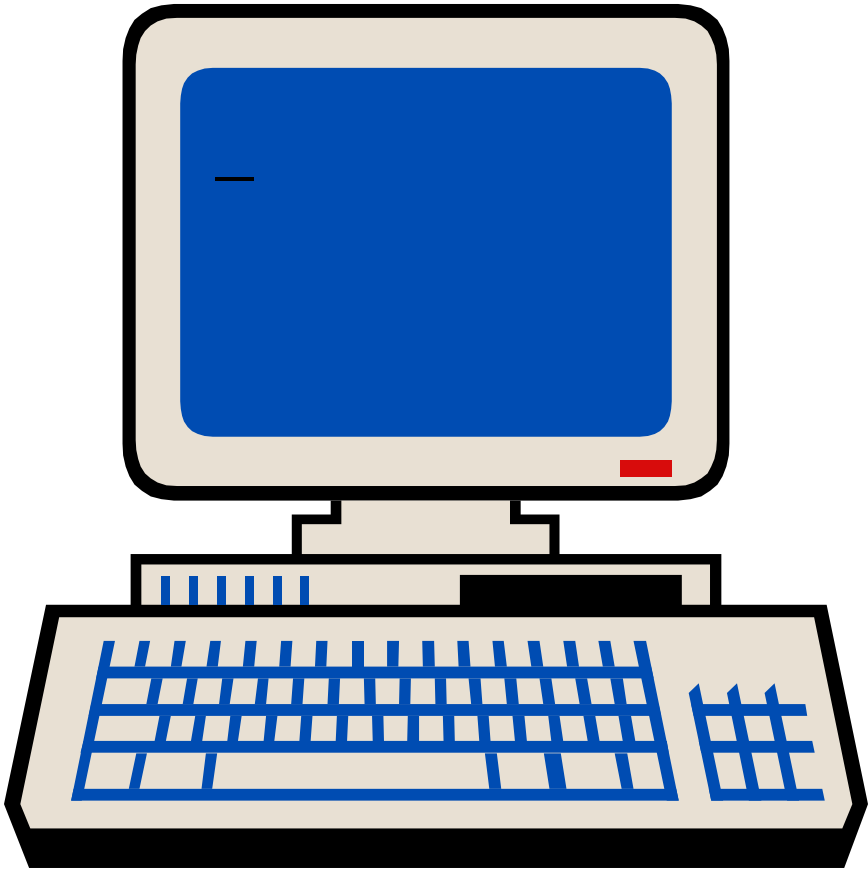
```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```



# Conversion Example (cont)

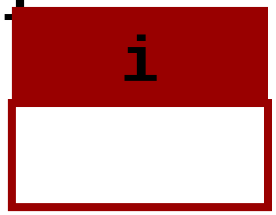
```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

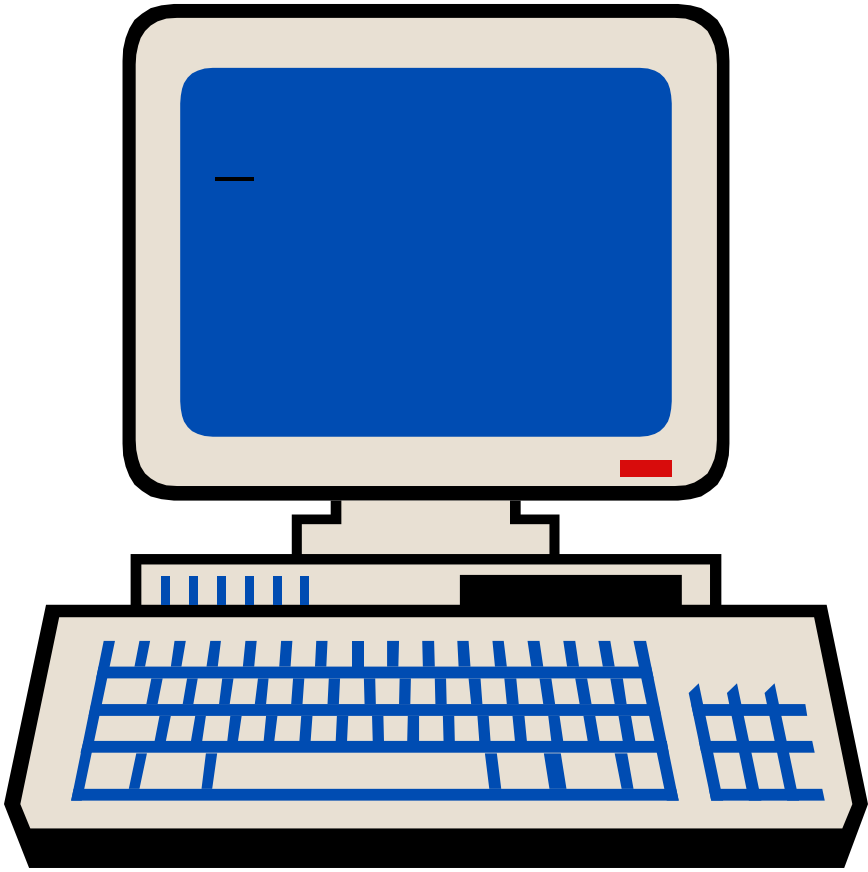


# Conversion Example (cont)

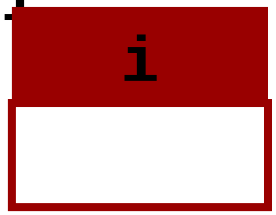
```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

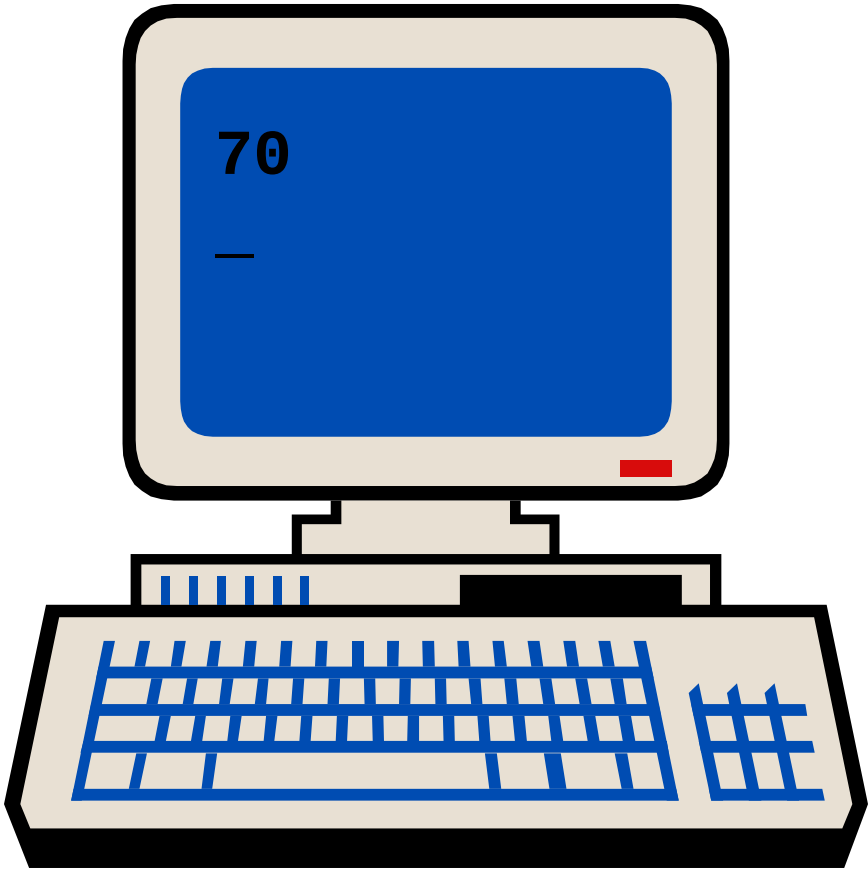


# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

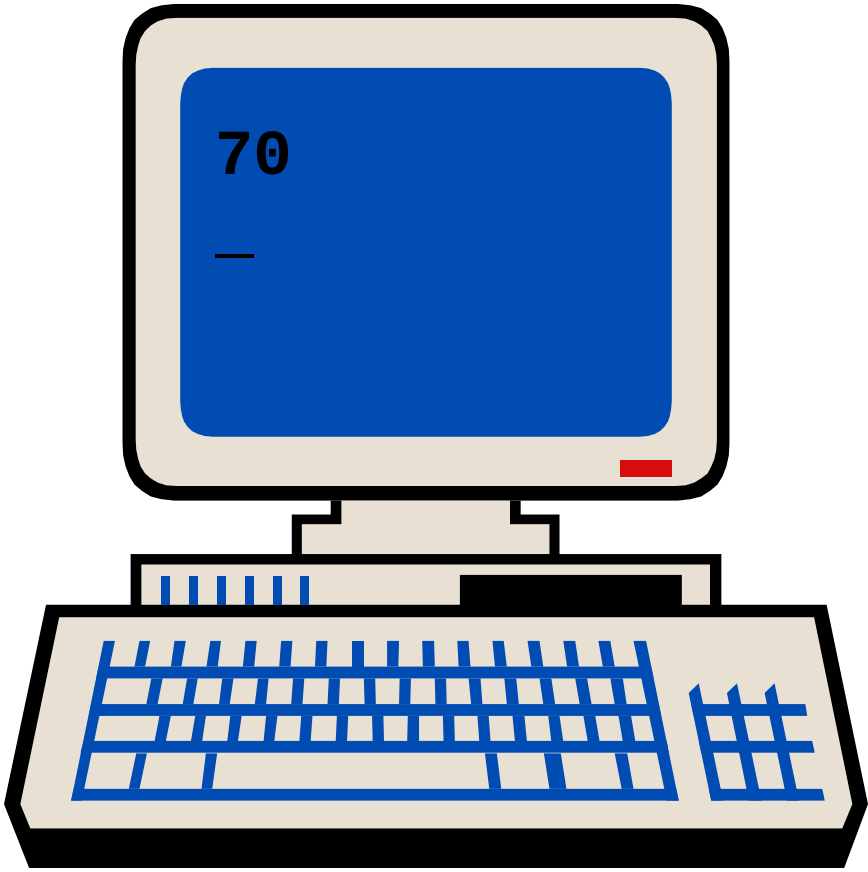


# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

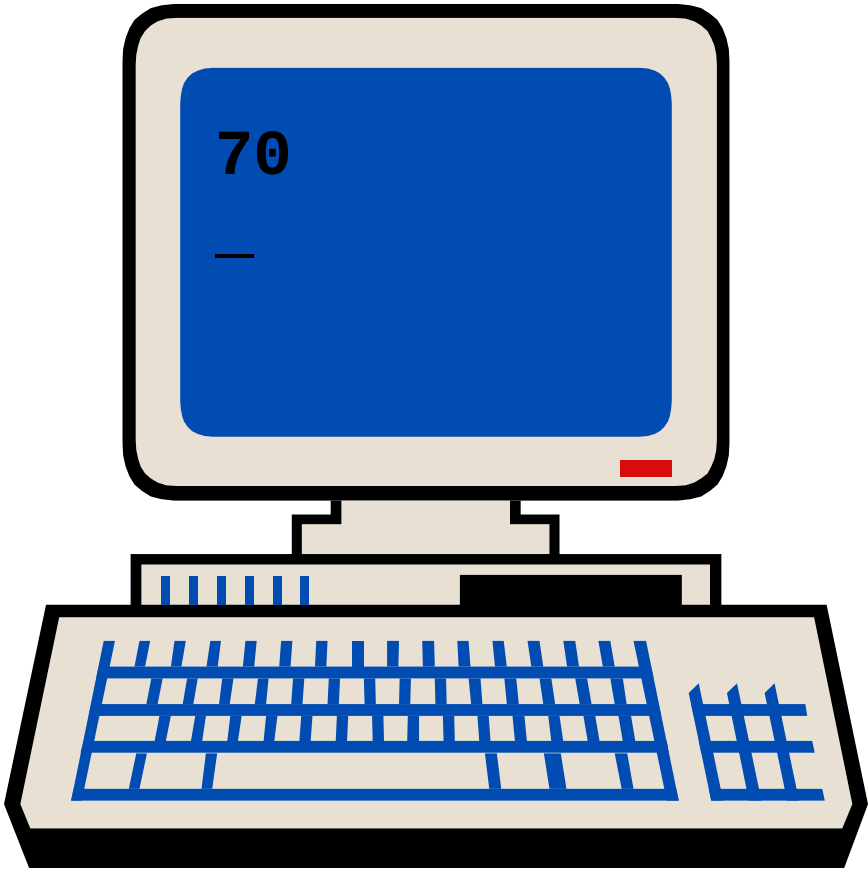
| i |
|---|
| 56 |



# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

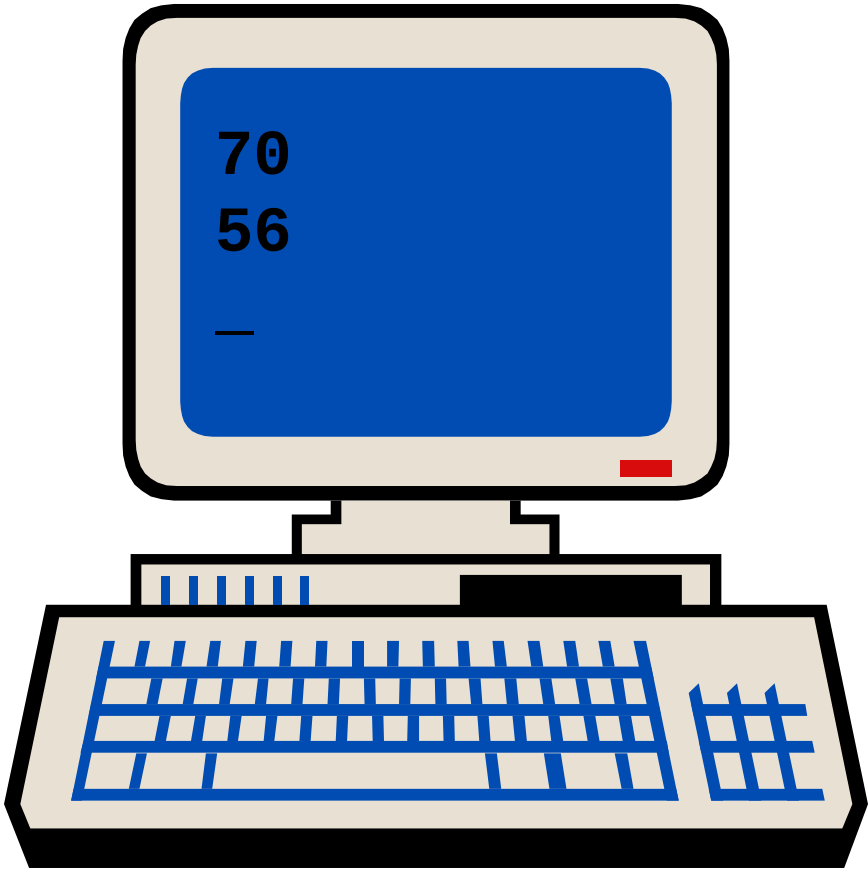
| i |
| --- |
| 56 |



# Conversion Example (cont)

```
#include <stdio.h>
int main()
{
    int i ;

    scanf("%o", &i);
    printf("%d\n", i);
    return 0;
}
```

| i |
| --- |
| 56 |



# Skipping Characters in Input Stream

- Skipping blank spaces

```
scanf("%d %d %d", &day, &month, &year);
```

- An alternative
  - Enter data as dd-mm-yyyy: 16-3-1999
  - Store each number in date variables

```
scanf("%d-%d-%d", &day, &month, &year);
```

# Summary

- Input from keyboard is via the stdin stream
- Output to the screen is via the stdout stream
- Streams carry characters
  - divided into lines with `'\n'` character
  - input ends with special value: **`EOF`**
- To use the C library functions, you must include the stdio.h header file
- Input and output can be formatted and converted between data types

# Reading

Deitel & Deitel: Section 9.1 to 9.6, Section 9.11