CSE103
Structured Programming
Lecture-2

Dr. Maheen Islam
Associate Professor
Dept. of CSE
East West University

Introduction to C

- C language
  - Facilitates a structured and disciplined approach to computer program design
  - Provides low-level access
  - Highly portable

Program Basics

- The ***source code*** for a program is the set of instructions written in a high-level, human readable language.

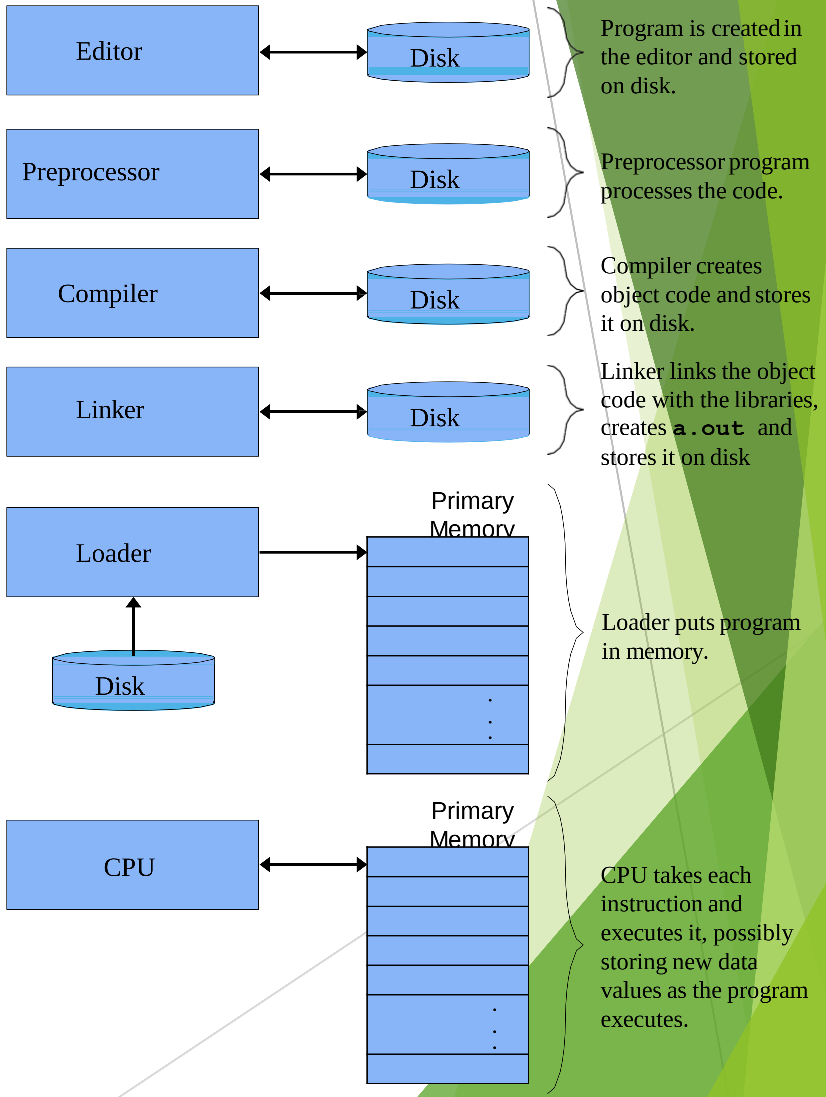
```
X = 0;
MOVE 0 TO X.
X := 0
```

- The source code is transformed into ***object code*** by a ***compiler***. Object code is a machine usable format.
- The computer ***executes*** a program in response to a command.

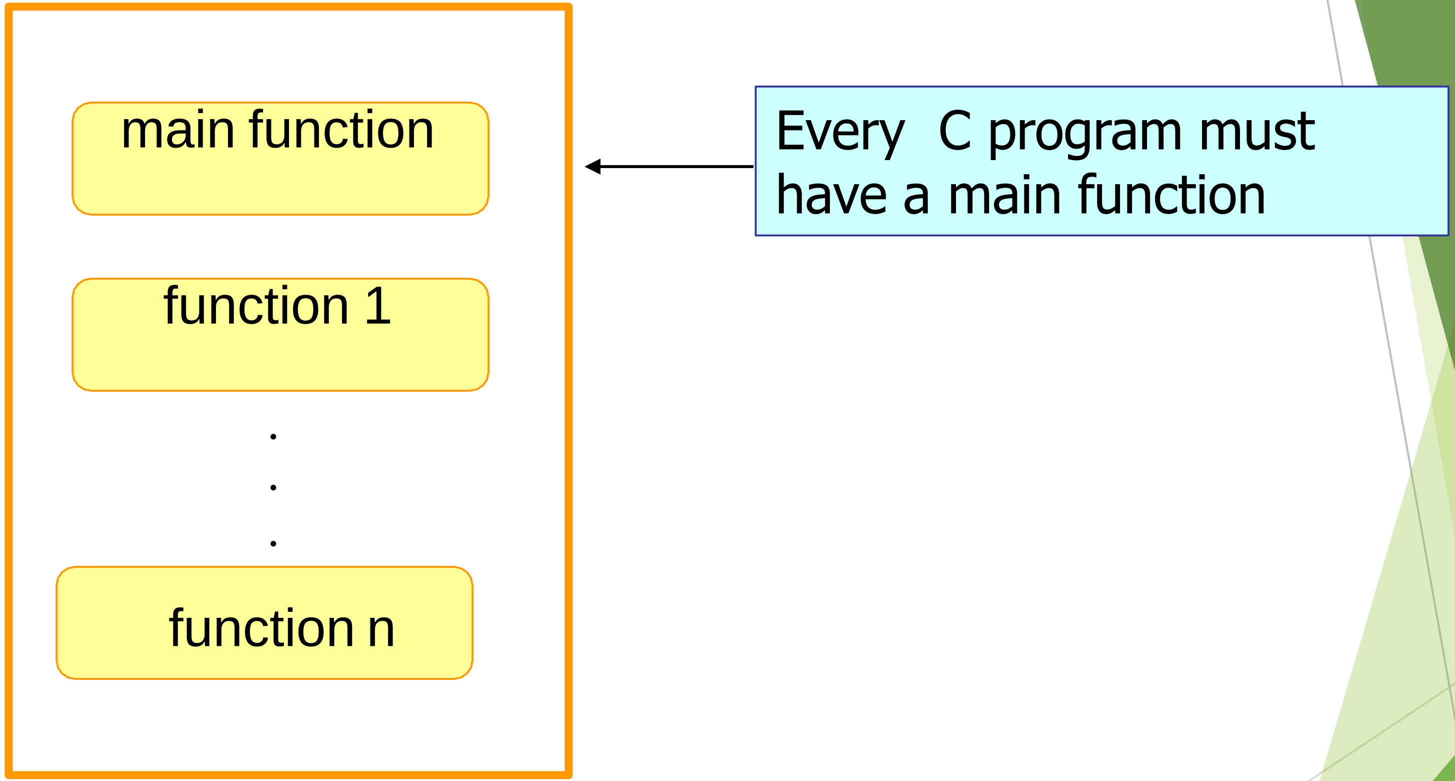
Basics of a Typical C Environment

Phases of C Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute


Editor
Disk
Program is created in the editor and stored on disk.
Preprocessor
Disk
Preprocessor program processes the code.
Compiler
Disk
Compiler creates object code and stores it on disk.
Linker
Disk
Linker links the object code with the libraries, creates a.out and stores it on disk
Loader
Disk
Primary Memory
Loader puts program in memory.
CPU
Primary Memory
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Structure of a C Program



Functions

- Each function consists of a **header** followed by a **basic block**.
- General format:

<return-type> fn-name (parameter-list) $\longleftarrow$ $\boxed{\textit{header}}$

basic block

The Basic Block

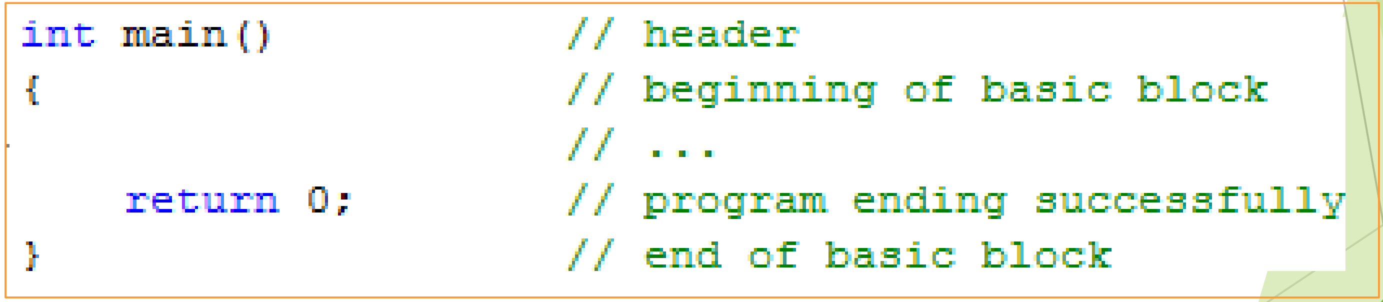
```
{
      declaration of variables
      executable statements
}
```

- A semi-colon (;) is used to terminate a statement
- A block consists of zero or more statements
- Nesting of blocks is legal and common
  - Each interior block may include variable declarations

Return statement

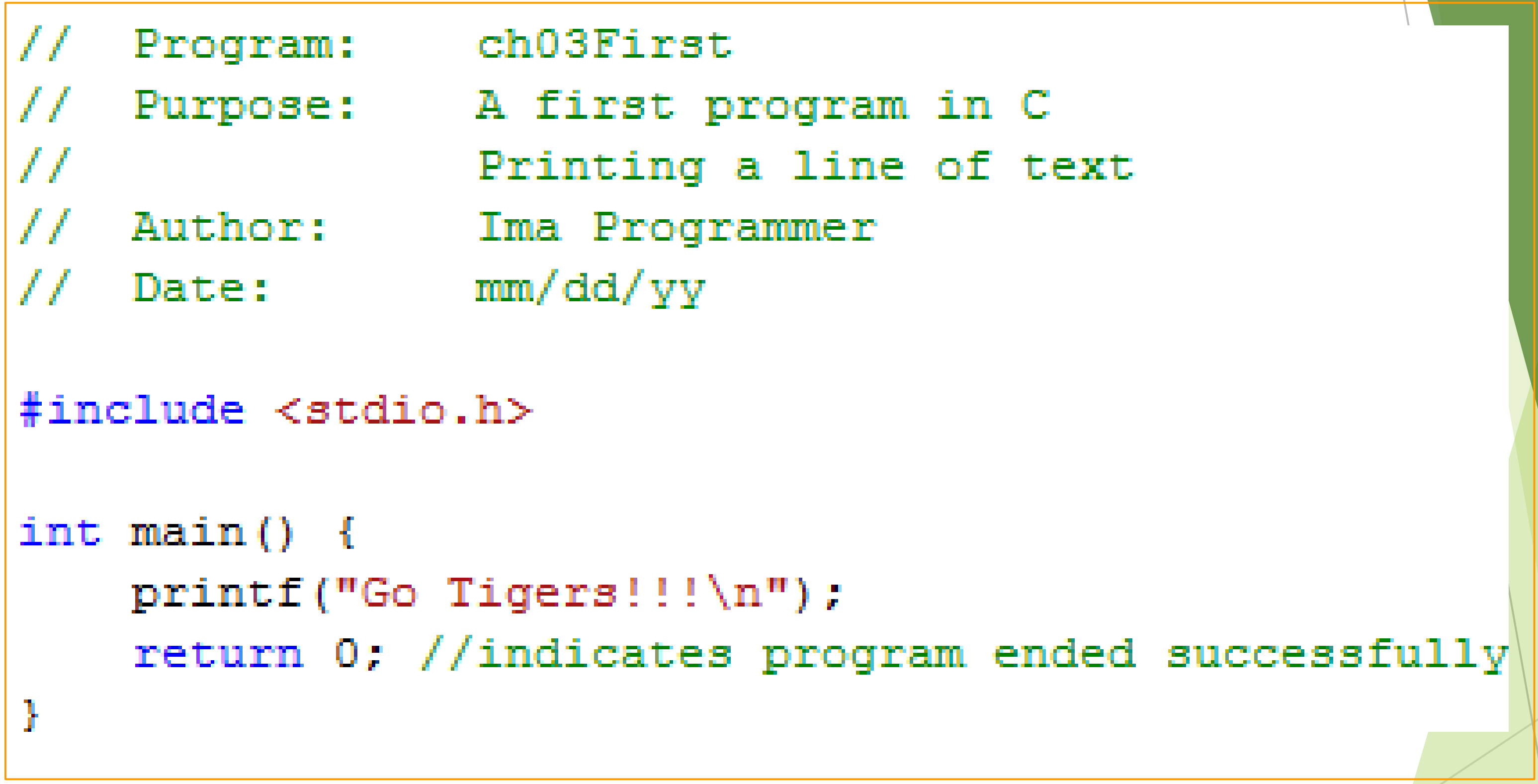
- return *expression*
  1. Sets the return value to the value of the expression
  2. Returns to the caller / invoker

- Example:



```
int main()                  // header
{                           // beginning of basic block
                            // ...
     return 0;              // program ending successfully
}                           // end of basic block
```

Our First Program

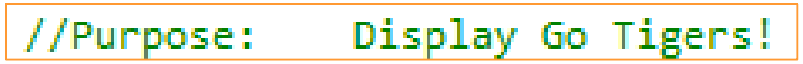


```c
//   Program:     ch03First
//   Purpose:     A first program in C
//                Printing a line of text
//   Author:      Ima Programmer
//   Date:        mm/dd/yy

#include <stdio.h>

int main() {
    printf("Go Tigers!!!\n");
    return 0; //indicates program ended successfully
}
```

```
Go Tigers!!!
```

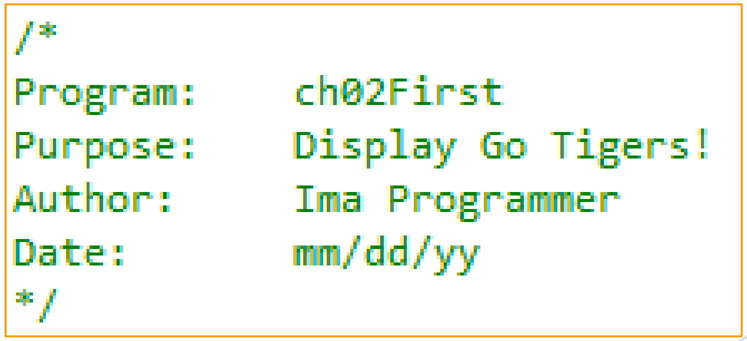
Comments

- Make programs easy to read and modify
- Ignored by the C compiler
- Two methods:
  1. // - line comment
     - everything on the line following // is ignored



```
//Purpose:     Display Go Tigers!
```

  2. /*   */ - block comment
     - everything between /*   */ is ignored



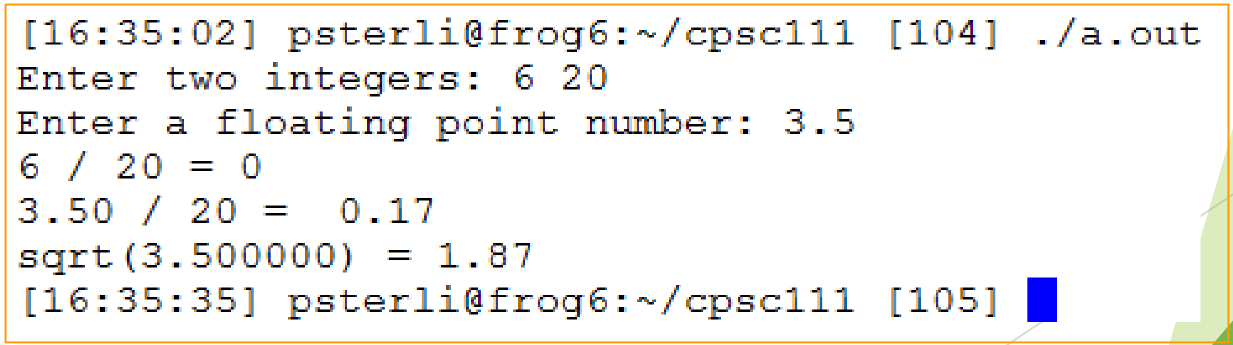
```
/*
Program:     ch02First
Purpose:     Display Go Tigers!
Author:      Ima Programmer
Date:        mm/dd/yy
*/
```

Preprocessor Directive: #include

- A C program line beginning with # that is processed by the compiler before translation begins.
- #include pulls another file into the source
  - `#include <stdio.h>` causes the contents of the named file, stdio.h, to be inserted where the # appears. File is commonly called a header file.
    - <>'s indicate that it is a compiler standard header file.
  - `#include "myfunctions.h"` causes the contents of myfunctions.h to be inserted
    - ""s indicate that it is a user file from current or specified directory

Introduction to Input/Output

- *Input data* is read into variables
- *Output data* is written from variables.
- Initially, we will assume that the user
  - enters data via the terminal keyboard
  - views output data in a terminal window on the screen



```
[16:35:02] psterli@frog6:~/cpsc111 [104] ./a.out
Enter two integers: 6 20
Enter a floating point number: 3.5
6 / 20 = 0
3.50 / 20 =  0.17
sqrt(3.500000) = 1.87
[16:35:35] psterli@frog6:~/cpsc111 [105]
```

Program Input / Output

- The C run-time system automatically opens two files for you at the time your program starts:
  - **stdin** – standard input (from the keyboard)
  - **stdout** – standard output (to the terminal window in which the program started)
- Later, how to read and write files on disk
  1. Using stdin and stdout
  2. Using FILE's

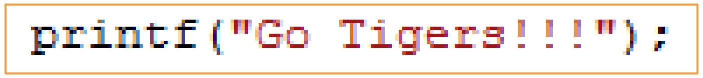
Console Input/Output

- Defined in the C library included in <stdio.h>
  - Must have this line near start of file:
    ```
    #include <stdio.h>
    ```
  - Includes input functions scanf, fscanf, …
  - Includes output functions printf, fprintf, …

Console Output - printf

- Print to standard output,
  typically the screen
- General format (value-list may not be required):

```
printf("format string", value-list);
```



```
printf("Go Tigers!!!");
```

Console Output

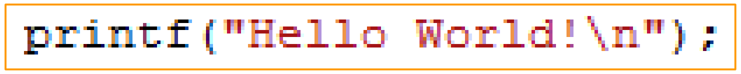
What can be output?

- Any data can be output to display screen
    - Literal values
    - Variables
    - Constants
    - Expressions (which can include all of above)
- Note
    - Values are passed to printf
    - Addresses are passed to scanf

Console Output

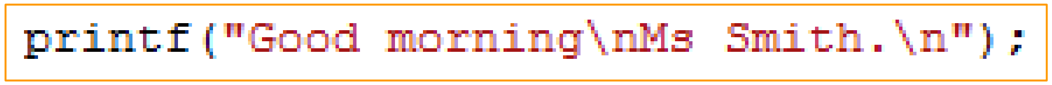
- We can
  - Control vertical spacing with blank lines
    - Use the escape sequence "\n“, new line
      - Should use at the end of all lines unless you are building lines with multiple printf’s.
      - If you printf without a \n and the program crashes, you will not see the output.
  - Control horizontal spacing
    - Spaces
    - Use the escape sequence “\t”, tab
      - Sometimes undependable.

Terminal Output - Examples



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

- Sends string "Hello World" to display, skipping to next line



```
printf("Good morning\nMs Smith.\n");
```

- Displays the lines

```
Good morning
Ms Smith.
```

Program Output:

- Indicates that a "special" character is to be output

Escape Sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\\"	Double quote. Used to print a double quote character.