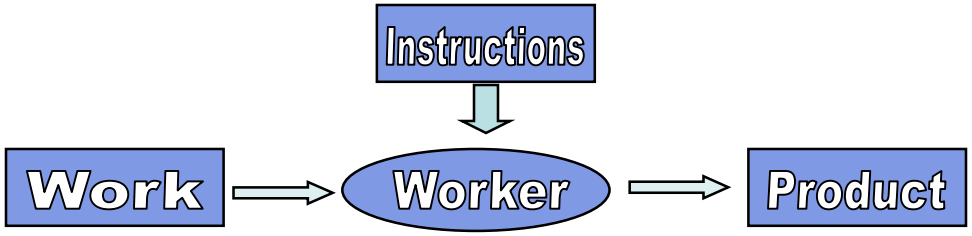
Structured Programming Language

Program

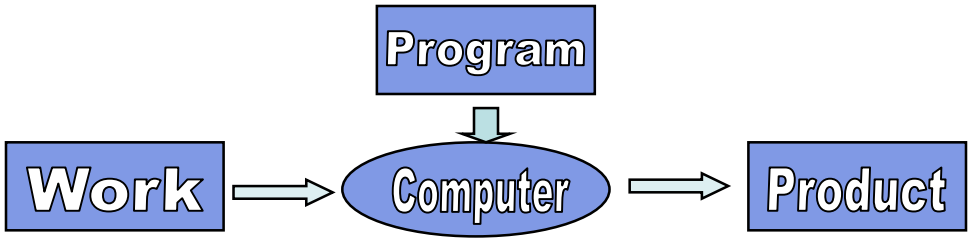
- A program is a specific set of operations to perform.
- A computer program is a collection of *instructions* that performs a specific task when executed by a computer.
- A computer program is usually written by a computer programmer in a programming language.

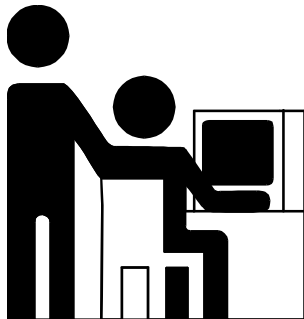
Program Cont.

- A computer program performs a specific task, and may interact with the user and the computer hardware.
  - Human work model:



  - Computer work model:





Language Types

- Three types of programming languages
  1. Machine languages
     - Strings of numbers giving machine specific instructions
     - Example:

```
+1300042774
+1400593419
+1200274027
```

  2. Assembly languages
     - English-like abbreviations representing elementary computer operations (translated via assemblers)
     - Example:

```
load    BASEPAY
add     overpay
store   GROSSPAY
```

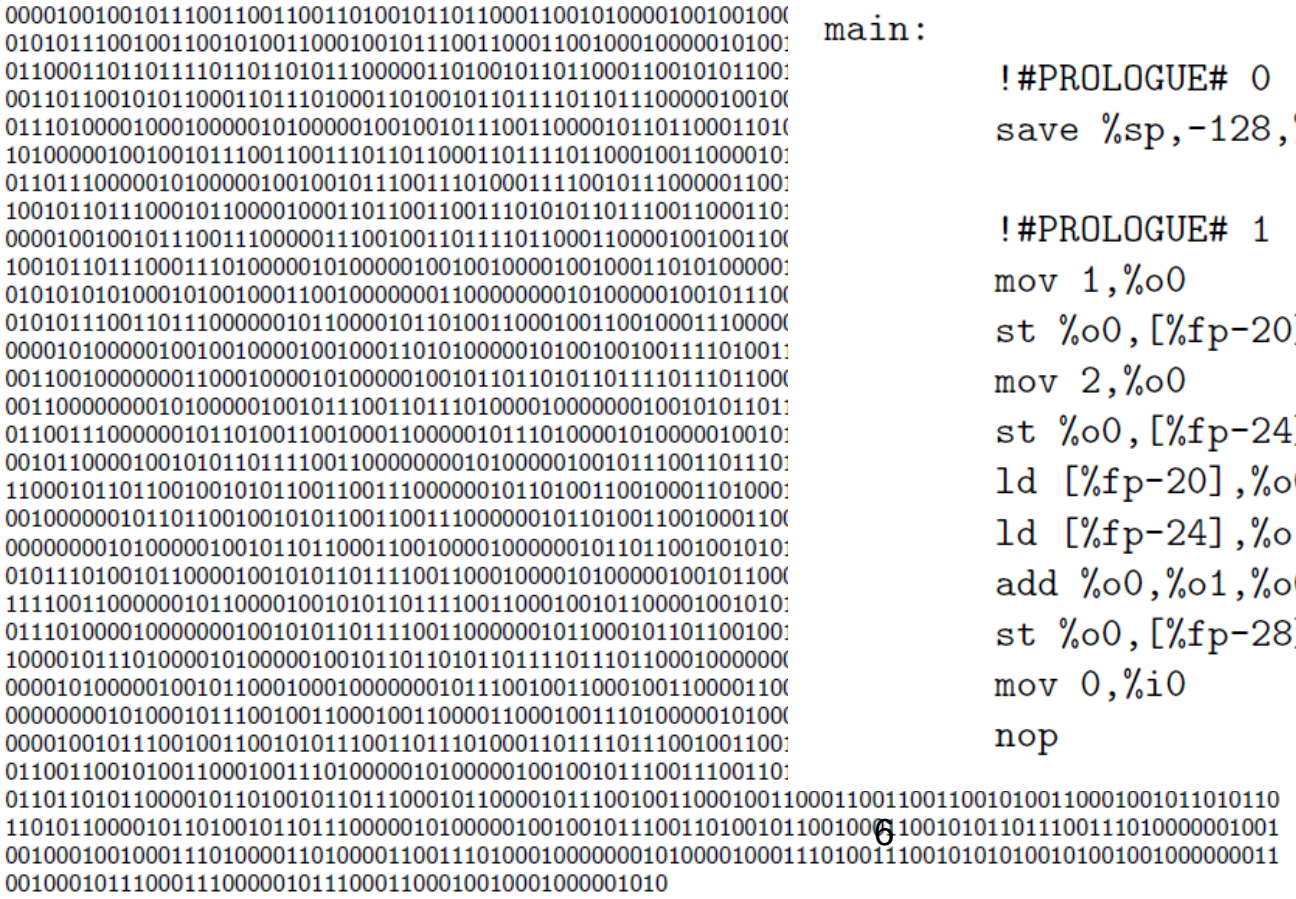
Language Types, Cont.

3. High-level languages
   - Codes similar to everyday English
   - Use mathematical notations (translated via compilers)
   - Example:

```
grossPay = basePay + overTimePay
```

An example:

Machine binary language



Low-level assembly

```
main:
        !#PROLOGUE# 0
        save %sp,-128,%sp

        !#PROLOGUE# 1
        mov 1,%o0
        st %o0,[%fp-20]
        mov 2,%o0
        st %o0,[%fp-24]
        ld [%fp-20],%o0
        ld [%fp-24],%o1
        add %o0,%o1,%o0
        st %o0,[%fp-28]
        mov 0,%i0
        nop
```

High-level

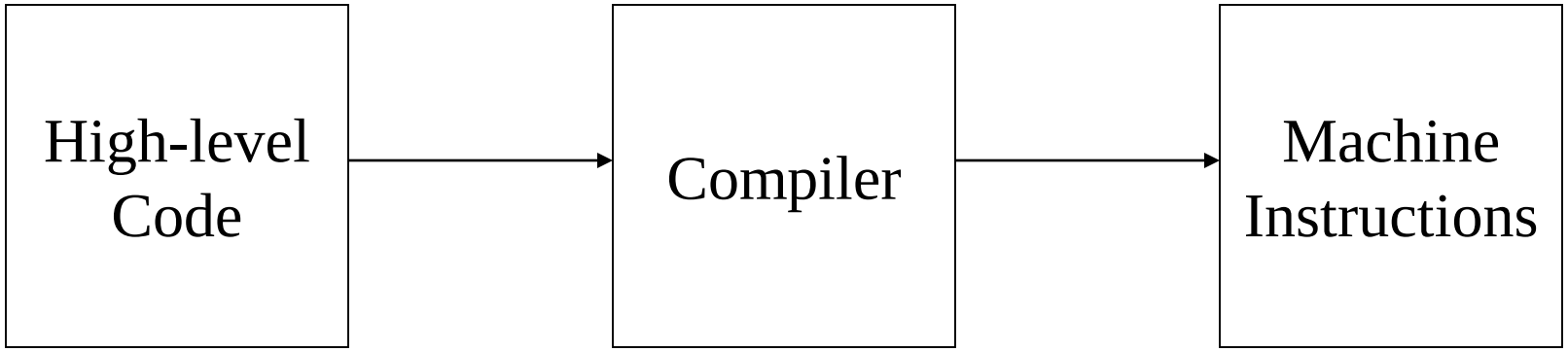
```
int main()
{
    int x, y, z;

    x = 1;
    y = 2;
    z = x+y;

    return 0;
}
```

High-level Languages

- “high-level” is a relative term
- C is a relatively low-level high-level language
- Pascal, Fortran, COBOL are typical high-level languages
- Java, Python, Perl, VB are examples of high-level high-level languages
- Application specific languages (Matlab, Javascript, VBScript, Scratch) are even higher-level.

Translation

- High level language must be translated into a language the computer can understand



How to translate?

A program written in high-level programming language (for example, C program)

COMPILER (for example, gcc)

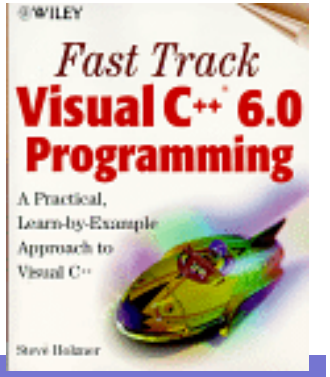
A low-level (machine language) program that is understandable by a computer (for example, a PC)

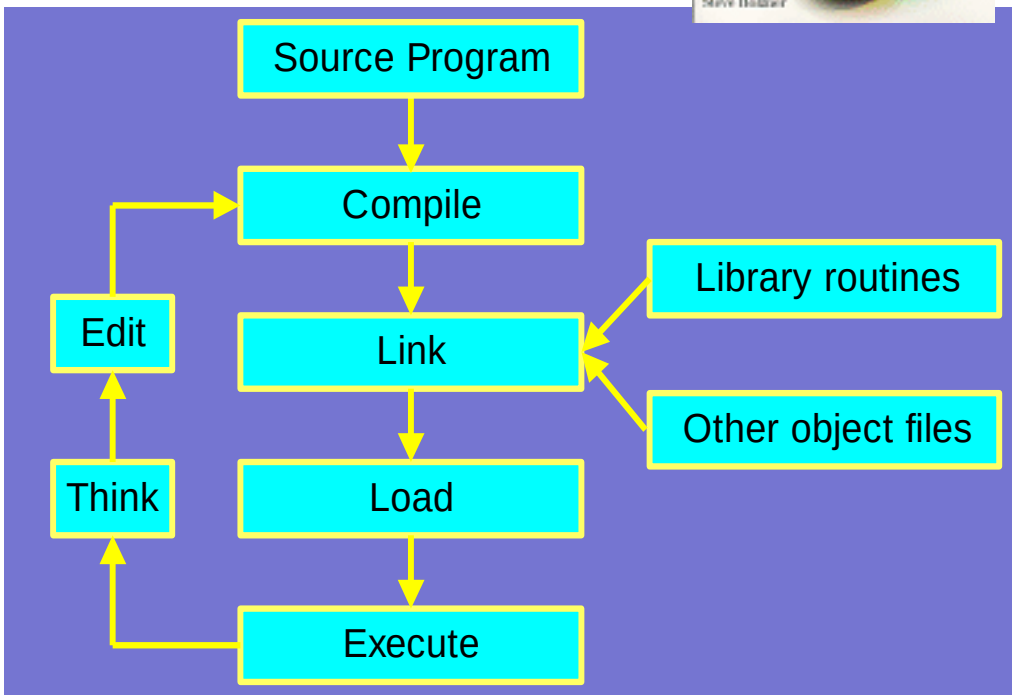
Examples of compilers:

- gcc, g++, Microsoft Visual C++

Software Development

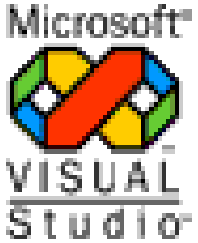
- Major activities
  - Editing (writing the program)
  - Compiling (creates .obj file)
  - Linking with compiled files (creates .exe file)
    - Object files
    - Library modules
  - Loading and executing
  - Testing the program

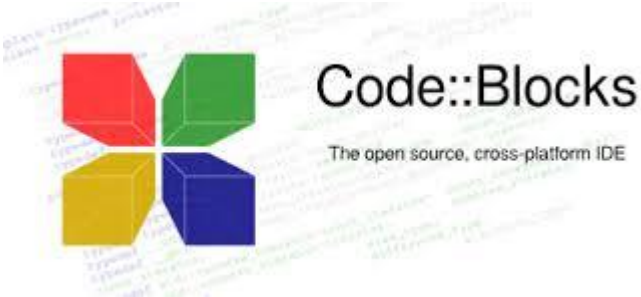




Integrated Development Environments

- Combine all of the capabilities that a programmer would want while developing software (VC++ 2008, Eclipse, CodeBlocks)
  - Editor
  - Compiler
  - Linker
  - Loader
  - Debugger
  - Viewer





Writing Programs

- While writing a program we have to-
  - Understand requirements
  - Write an *algorithm*
  - *Implement* your algorithm
  - *Test* your code

What is an algorithm?

- A *specific* set of instructions

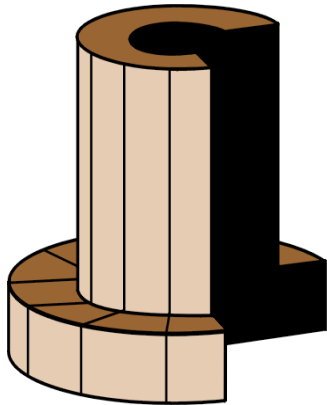
- An algorithm to make a cup of coffee?

C Programming Language

- What is C?
  - C is a structured, relatively low-level, portable programming language.
- Why study C?
  - Many popular software tools are written in C.
  - First programming language that came up with many diverse features.
  - Has strongly influenced many other languages.
    - C-shell, java, C++, Perl, etc.
  - Forces the user to understand fundamental aspects of programming.
  - Very concise language.

Most Important Feature of C

- Most important feature of C: its strong and efficient support of Structured Programming .
- C runs on: All Computers
  - PC
  - Macintosh
  - Unix workstations (also, Unix versions of C are free!)
  - supercomputers

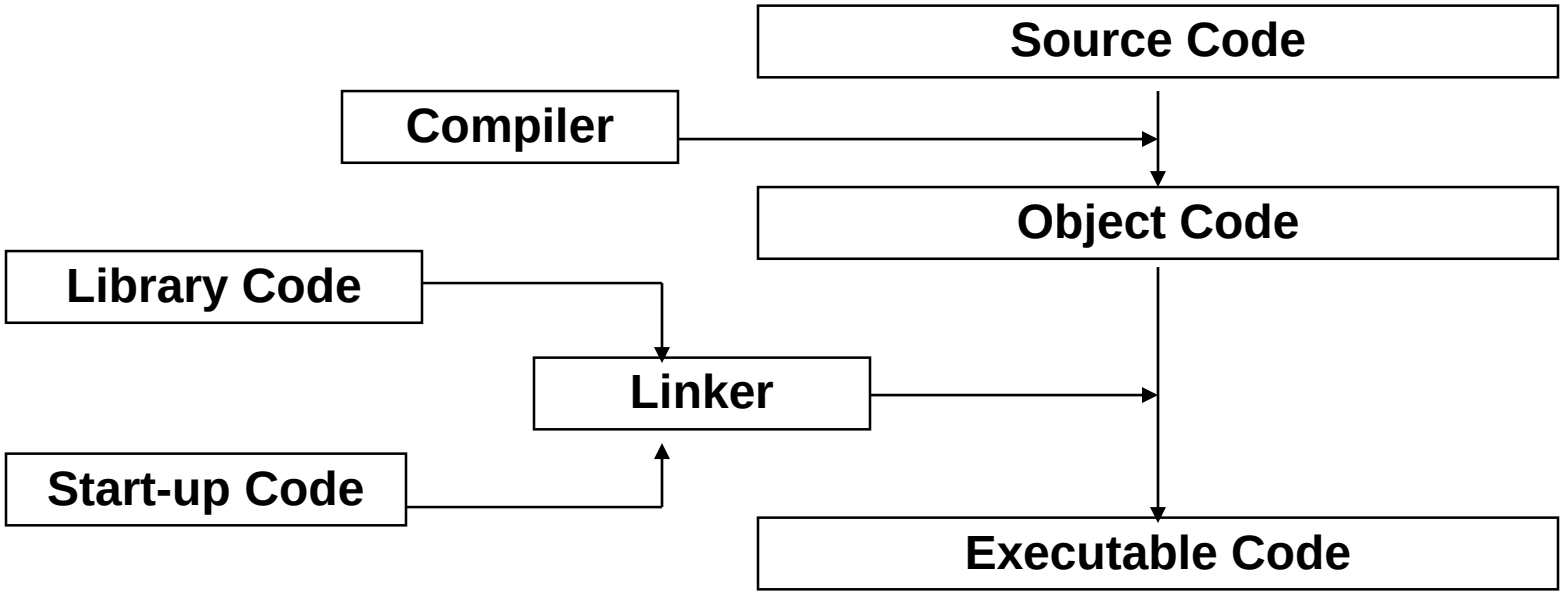


Basics of a Typical C Environment

- C systems
  - Program-development environment
- C program names extensions
  - .c
- Executable file extensions
  - .exe

C Basics

- Case Sensitive: UPPERCASE vs. lowercase
- NOT Strongly typed: Checking left to Programmer
- Each Statement ends with a semicolon(;)
- Ellipses and Parentheses MUST match: {…. } (….)
- Each program must have a function named main.
- Each functions scope is defined by {…. }
- Variables declared inside of {…. } is only visible to that scope.
- C is a translated language (vs. interpreted):



C Basics Cont…

- A C program basically consists of the following parts −
  - Preprocessor Commands(begin with # symbol)
  - Functions(exactly one main function in each program)
  - Variables
  - Statements & Expressions
  - Comments(/*…..*/ or //…….)

C Basics Cont…

- Precompiler Instructions:

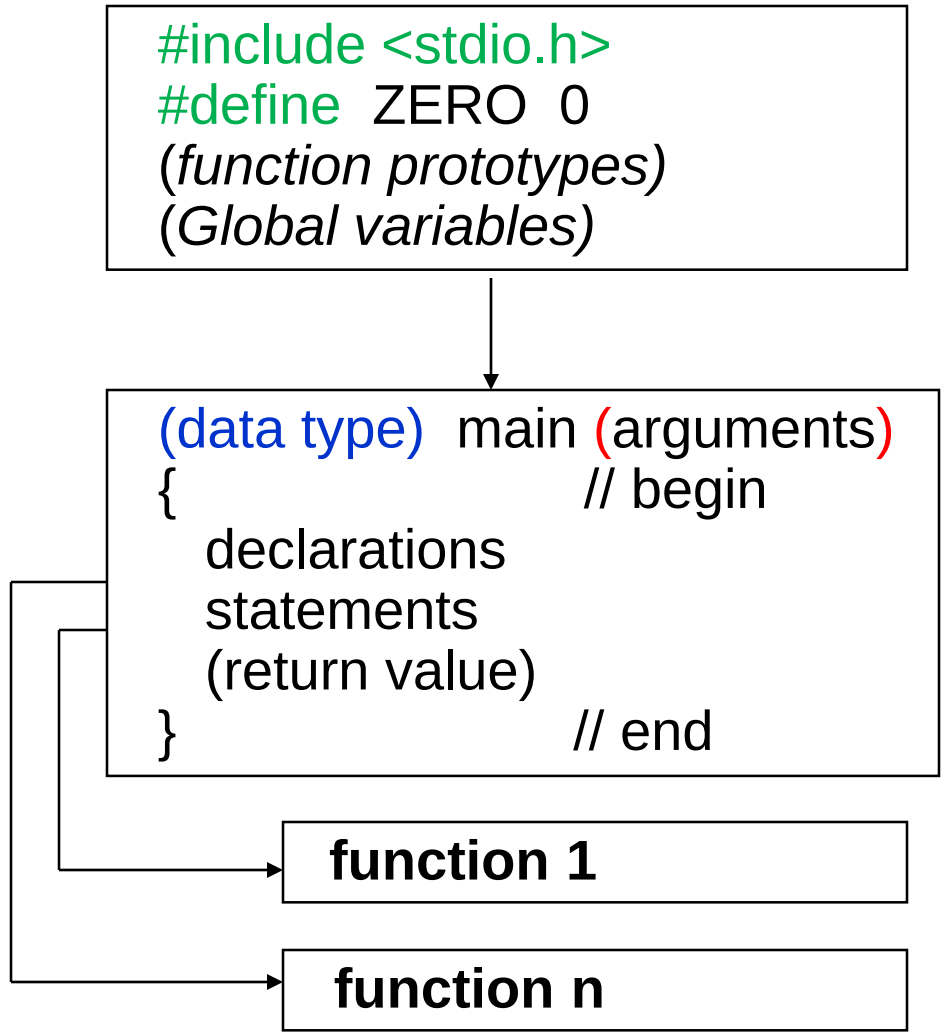
  Additional code/shorthand/ definitions

- Main Function:

  A C program consists of one or more function. The primary function MUST be named **main**

- Additional Functions:

  (optional) functions are the building blocks of C



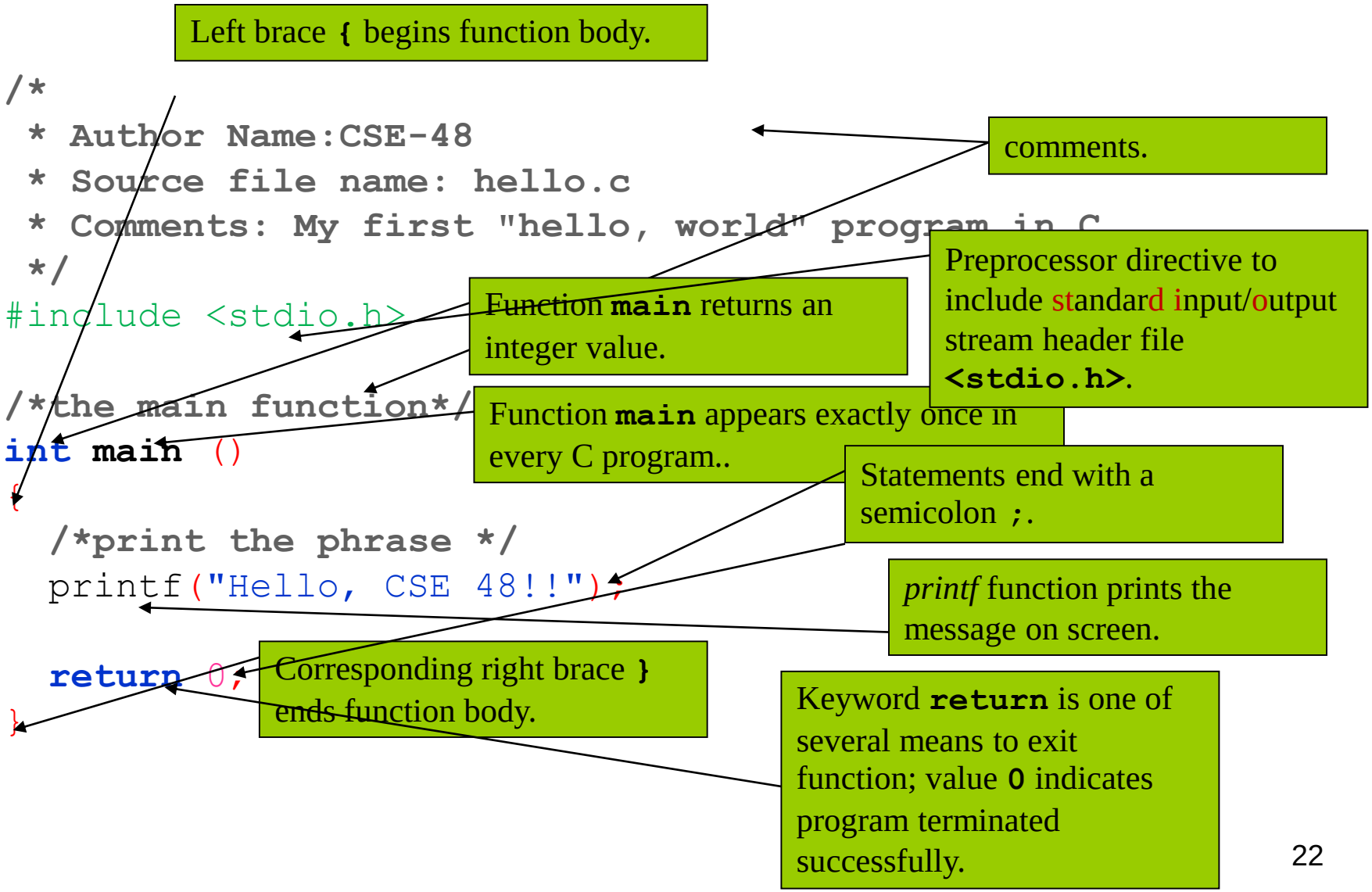
C Basics Cont…

- Common Input/output functions
  - **`scanf("")`**
    - Standard input stream
    - Normally keyboard
  - **`printf("")`**
    - Standard output stream
    - Normally computer screen
  - **`fprintf(stderr,"")`**
    - Standard error stream
    - Display error messages

C Basics Cont…

- Comments
  - Document programs
  - Improve program readability
  - Ignored by compiler
  - Single-line comment
    - Use C's comment /* .. */ OR Begin with **`// or`**

First C Program: Hello CSE 48!!



C Basics Cont…

```
/*
 * Author Name:CSE-48
 * Source file name: hello.c
 * Comments: My first program in C.
 */
```

Comment

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

Preprocessor

```
/*the main function*/
```

Comment

```
int main ()
{
  /*print the phrase */
  printf("Hello, CSE 48!!");

   return 0;
}
```

Main Function

Comment

Function

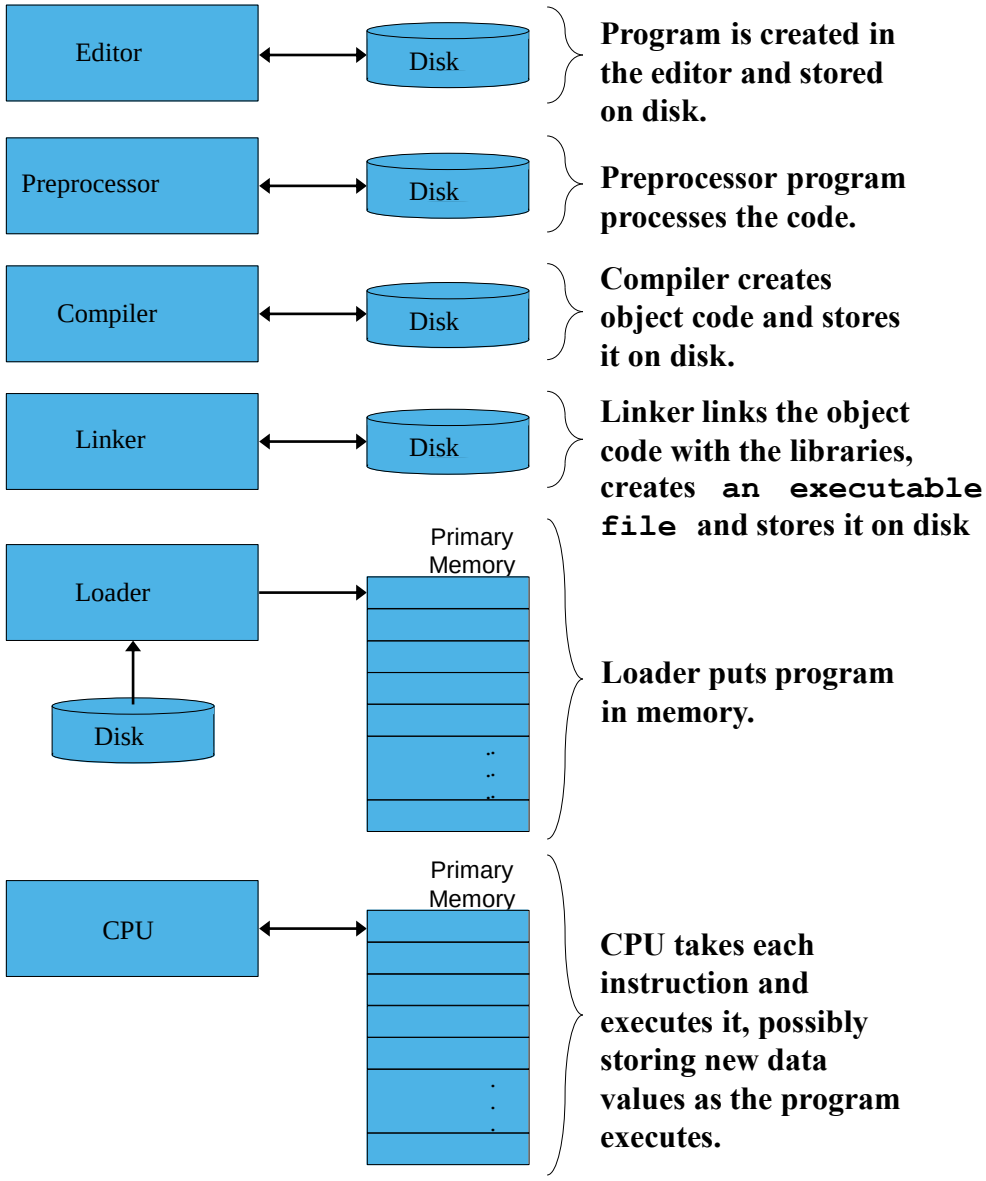
Errors

- You WILL have errors in your programs
- Syntax Errors
  - Compiler doesn't understand code
  - Ex. –`print("Hello, CSE 48!!!");`
  - Ex. –`printf("Hello, CSE 48!!!");`
  - Sometimes error messages don't match problem
- Logical Error
  - Program runs, but doesn't do what you want
  - Ex. – `printf("Hello, CSE!!!");`
  - Can be hard to track down

Basics of a Typical C Environment

Phases of C Programs:

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



C: Dangers

- C is not object oriented!
  - Can't "hide" data as "private" or "protected" fields
  - You can follow standards to write C code that looks object-oriented, but you have to be disciplined – will the other people working on your code also be disciplined?
- C has portability issues
  - Low-level "tricks" may make your C code run well on one platform – but the tricks might not work elsewhere
- The compiler and runtime system will rarely stop your C program from doing stupid/bad things
  - Compile-time type checking is weak
  - No run-time checks for array bounds errors, etc. like in Java