Getting Started with C

Summer 2022

Goal for this lecture

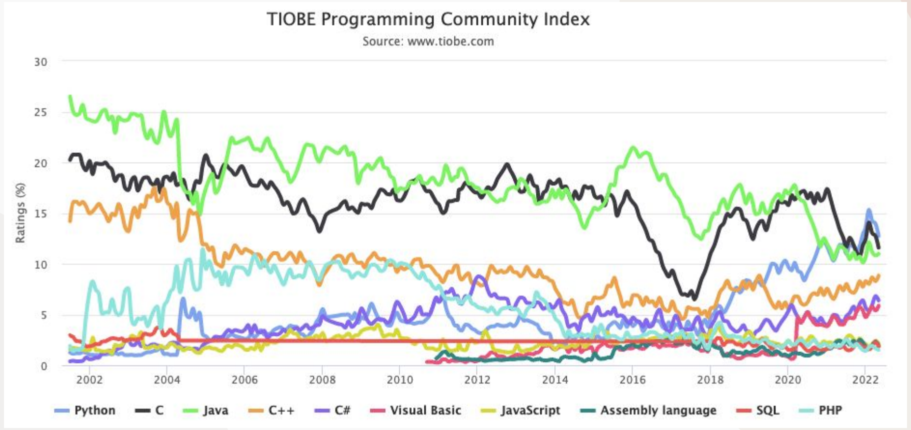
- Get the gcc compiler working
- Write, compile and run a simple C program
- Fix common compiler errors.
- Use of printf

Common Questions I received

- Is testing in Summer term going to be similar to winter term?
    - Same course so….
- How to get the gcc compiler?
    - Read the handout
    - Look for online resources
    - One good link to get GCC for mac : https://www.youtube.com/watch?v=0z-fCNNqfEg
- GDB for mac isn't working?
    - Multiple online resources to get the GDB working
    - Alternate option: use lldb
    - Similar to gdb
    - Commands might differ somewhat
    - Refer to: https://lldb.llvm.org/use/map.html

C Programming Language

- Widely used and important.
- Impressive for a language that was invented in 1972!
- Simple language.
- Many standards (by ISO):
    - K&R C through ANSI C, C99, C11 and Embedded C.
- We will use the standard **c99.**
- Used in a huge range of disciplines, from OS, controllers, upto applications in scientific disciplines.



https://www.tiobe.com/tiobe-index/

C99 standard

- C99 is the C standard ratified by the ANSI and ISO standardization groups
  - Result of sibling competition between C and C++
- Came after K&R C and C89 (also known as ANSI C)
- Some features added in C99 standard:
  - Boolean data types <stdbool.h>
  - Increased identifier size limits
  - C++ style/line comments
  - Inline functions
  - Restricted pointers
  - Variable Declarations
  - Variable length arrays
  - New long long type

Why C?

- C runs on everything!
- C imposes very few constraints on programming style: unlike higher-level languages, C doesn’t have much of an ideology
- Very few programs you can’t write in C.
- C lets you write programs that use very few resources.
- C gives you near-total control over the system.
- Many of the programming languages people actually use are executed by interpreters written in C.
- Python is written in C !!!!!!
- Other examples:
  - Visual Basic, Perl, PHP, Ruby

C vs. Python?

- Most of you (if not all!) have learned Python
- and, you may (fairly) wonder, why C and not Python?!
- There are similarities but many differences…

C	Python

C vs. Python?

- Most of you (if not all!) have learned Python
- and, you may (fairly) wonder, why C and not Python?!
- There are similarities but many differences…

C	Python
modular, imperative	modular, imperative
static variables and types declarations	Dynamic variables and typing
compiled language	interpreted language

Compiled language vs. interpreted language

- Most applications are written in a higher-level programming language, such as C/C++, C#, Java, or one of the many scripting languages including Python.
- When an application is developed, the raw language is its source code.
- Unfortunately, computers don't understand source code, so the high-level language must be converted into machine code – the native instructions the computer processor executes
- Two common ways of developing and executing programs is by **interpreting** the original source code or by **compiling** a program to native code.

Compiler vs Interpreter

Compiler	Interpreter
Translates entire program at once to produce machine executable file	Translates one line at a time
Object code is generated	No object code is generated
Prog. Languages: C, C++, Java	Python, Ruby, JavaScript

How does a computer ’run’ code?

- CPU can only carry out a number of very simple but specialized instructions
  - Eg. adding or comparing two numbers
- Not in Python or C, but in machine code
- CPU runs a “fetch/execute cycle”
  - fetch one instruction in sequence
  - execute (run) that instruction, e.g. addition
  - fetch the next instruction, and so on
- Run a program = Start CPU running on its 1st instruction
  - it runs down through all of the machine code, running the program
  - the program will have instructions like "return to step 3" to keep it running
- Super simple machine code instructions run at the rate of 2 billion per-second

Programs/Code

- We write programs as text, which we can read and understand
- but, these have to first be translated to instructions that the CPU can understand
- Python does this at the moment -
  - it needs to run each instruction – it is *interpreting* the program as it runs, much the same way an interpreter translates a person’s foreign speech as they are talking
- C does the translation once, before the program runs, and stores the translated instructions in an ’executable program’ that the machine can run

Three step Mantra for Creating and Running a C Program

- Write (or edit/change) the program in text editor
  - Choices of text editors: notepad++, vi, vim, emac,VSCode, atom, sublime
- Compile our program using GCC compiler
  - gcc hello.c
- Run the program
  - ./a.out(mac) or a.exe(windows)

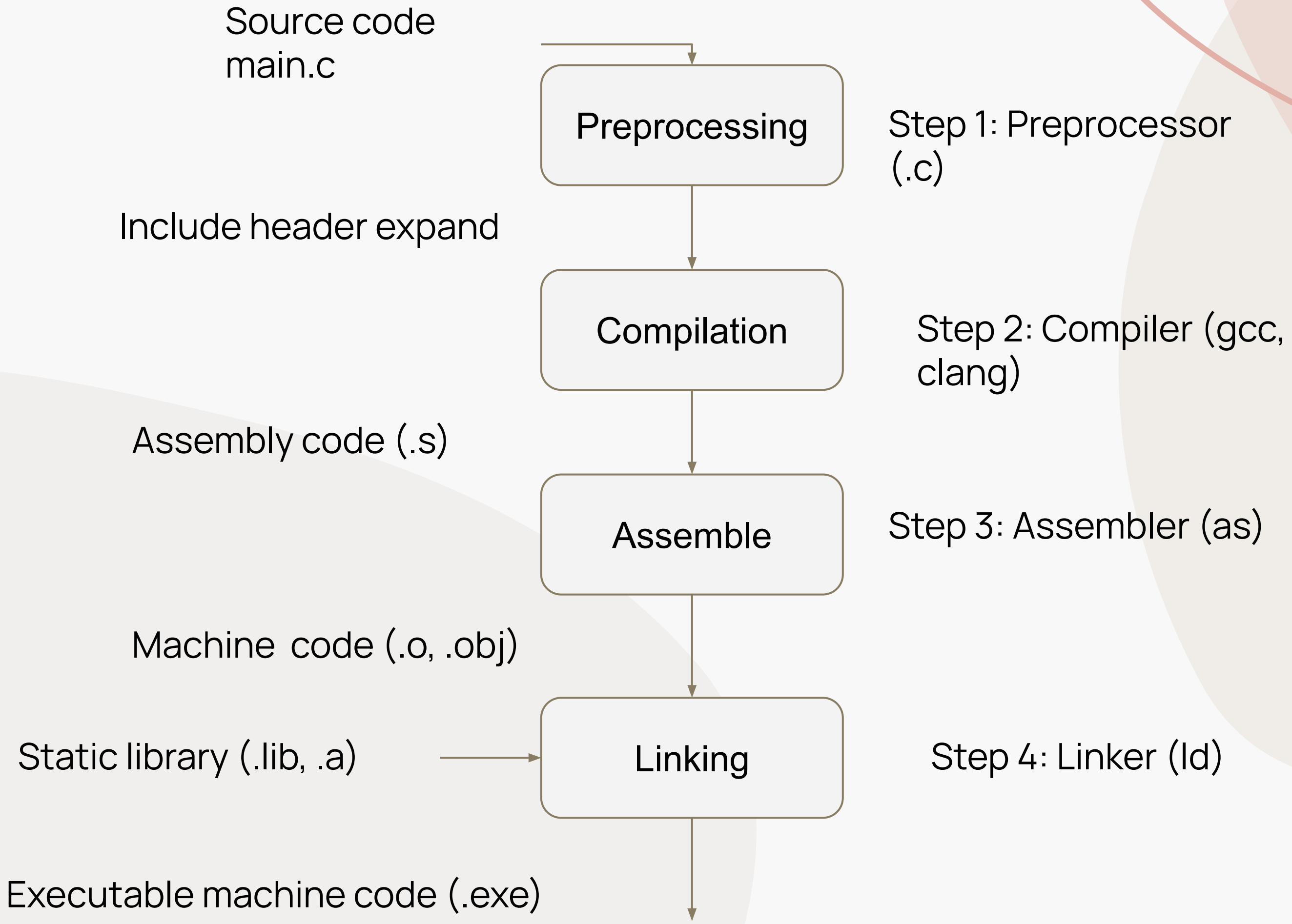
Editors preferred

- VERY SIMPLE EDITORS - Notepad++, Vim, Vi etc
- What about IDEs (Integrated Development Environment)?
- People like them because
  - they have GUIs
  - downside they have GUIs!

GCC

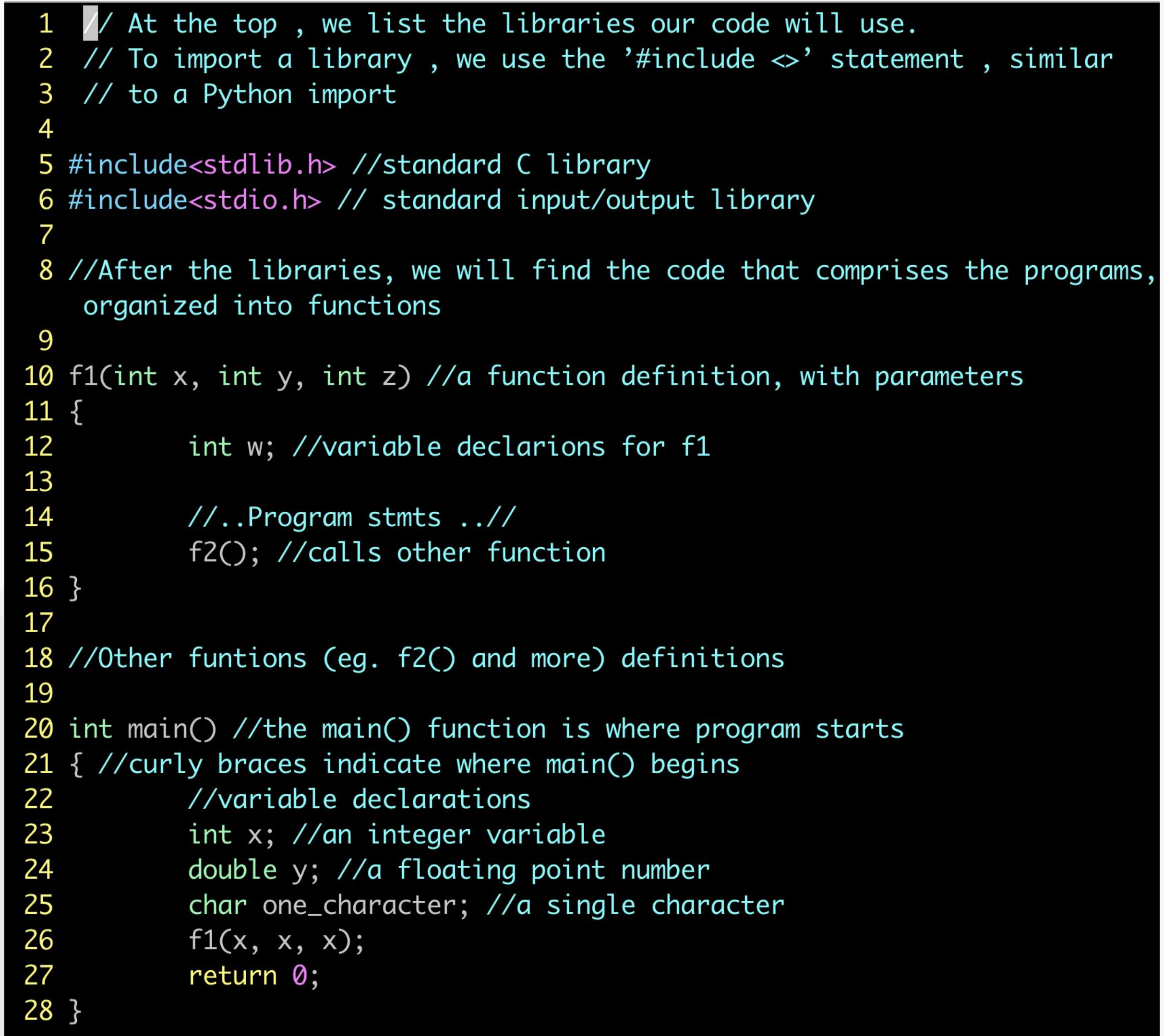
- C compiler that can handle C99 standard.
- The GNU Compiler Collection (GCC) is one such compiler for the C language
- Other languages supported by GCC:
    - C++, Fortran, and Go
- Compilers translate source code through a four-step process — **preprocessing, compiling, assembly and linking**.

What happens when you type gcc main.c



Structure of a C Program

- a program is split into functions
- all programs start with a function called main()
- Programs can include and use external libraries
- Code delimiters are the curly braces: {}
- Comments: /* ... */ –or– //
- Every variable has an associated data type, the type can not be changed

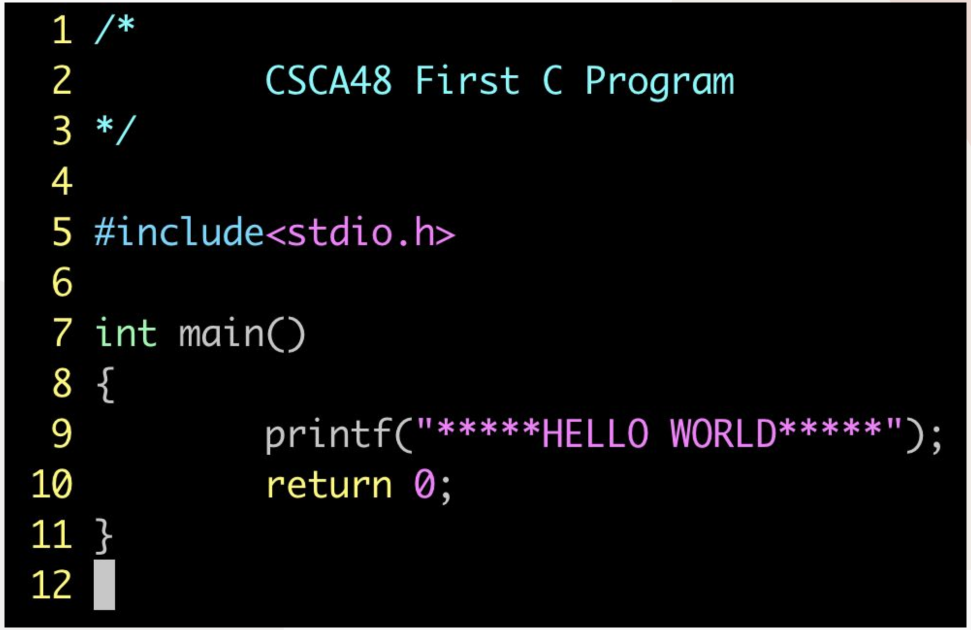


```c
// At the top , we list the libraries our code will use.
// To import a library , we use the '#include <>' statement , similar
// to a Python import

#include<stdlib.h> //standard C library
#include<stdio.h> // standard input/output library

//After the libraries, we will find the code that comprises the programs,
 organized into functions

f1(int x, int y, int z) //a function definition, with parameters
{
        int w; //variable declarions for f1

        //..Program stmts ..//
        f2(); //calls other function
}

//Other funtions (eg. f2() and more) definitions

int main() //the main() function is where program starts
{ //curly braces indicate where main() begins
        //variable declarations
        int x; //an integer variable
        double y; //a floating point number
        char one_character; //a single character
        f1(x, x, x);
        return 0;
}
```

- Variables declared within a function, are local to the function
- **main() always returns something**
- comment your code!!!
- **#include** is used to import libraries

Demo

1. edit your code
2. compile your code
   gcc -Wall -std=c99 filename.c -o executable
3. run your code
   ./executable
4. Repeat...



```c
/*
        CSCA48 First C Program
*/

#include<stdio.h>

int main()
{
        printf("*****HELLO WORLD*****");
        return 0;
}
```

Fundamental types in C

Fundamental data types supported in C

- int – integer numbers
- float/double – floating point numbers
- char – one character
- void – no data type attached
- other data types can be defined based on these primitive ones

Variables

- Variables in C are a direct abstraction of physical memory locations.
- In C, variable names are called identifiers.
- Rules for naming a variable
  - A variable name can only have letters (both uppercase and lowercase letters), digits and underscore.
  - The first letter of a variable should be either a letter or an underscore.
  - There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.
  - It helps if the variable is named logically
  - Valid: AlbertEinstein, aAaAaAaAaAAAAAa, , ____a___a_a_11727_a, and ______ are all legal identifiers in C
  - Invalid: 01, $dollar

How to deal with compiler errors

- Look at the output from the compiler.
- Locate the line that seems to be causing the problem and check it carefully in the editor (often it's a simple typo easy to see/fix)
- Not an easy fix? copy/paste the error message (without line numbers) into Google
- Look at the top 3 or so links, usually these will be places like stack overflow, where someone with the same error has asked how to fix it and the cause of the error as well as the solution is presented - fix the code
- Google did not result in a solution? at that point create an informative piazza post as discussed before and we can try to help you, or bring it to an office hour.

The Sequences

- \n for newline
- \t for tab
- \b for backspace
- \" for double quotes
- \\ for backslash
- %% for % symbol

Things to do this week

- Compiler Error Journal
  - Create a note to document common compiler errors you have encountered so far and note what you did wrong t cause those errors
- Complete exercise 1 and **optional** ex1b
  - Update your error journal based on compiler errors you encountered while finishing exercise 1
- Complete questionnaire 1
- Complete reading of handout 1