C Language Basics

COP 3223C – Introduction to Programming with C

Fall 2025

Yancy Vance Paredes, PhD

Terminologies

Random Access Memory (RAM)

Main memory that *temporarily* stores instructions and data

Central Processing Unit (CPU)

The “brain” of the computer

Processes instructions via fetch-decode-execute-store cycle

Motivation Behind Programming Language

- We human beings have **our own languages** (e.g., English)
  - Example: “Hello”, “How are you?”
- We want to “tell” (i.e., **instruct**) computers to do something
- However, computers have their “own language”…
  - Example: `010101010000`
- So, there is a barrier

C Language /1

- A foundational programming languages in computer science
- Developed by Dennis Ritchie in the early 1970s at Bell Labs
- Design emphasized on **efficiency** and **control**
- Programming in C is like driving a manual car: more control, more speed, but more responsibility

C Language /2

- Allows us to write English-like instructions (i.e., **source code**) that could **eventually be converted** into something that the computer can understand

- A formal language because we must **follow a set of well-defined rules** on how to “write” these instructions (i.e., **syntax**)

- It is a **compiled** language; the **gcc** compiler compiles the source code into an **executable program**

Obligatory: Hello World! Program

- For now, simply memorize them…
- Make sure to copy everything **EXACTLY**
- Now, let’s look into the different **elements** of the source code
- Save it as hello.c

```c
#include <stdio.h>

int main(void) {
    // display a message
    printf("Hello, World!\n");

    // program is done
    return 0;
}
```

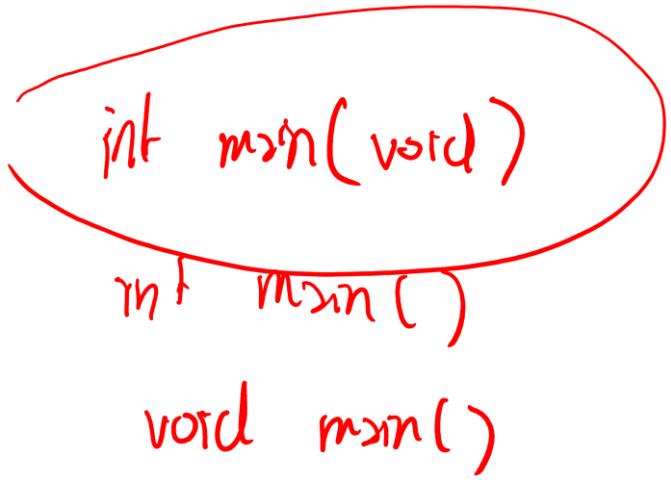
Pre-processor Directives

- Set of instructions for the **preprocessor**
  - A system program that modifies a C program prior to its compilation
  - Located at the top of the source code

- **`#include <stdio.h>`** means that we want to include the **`stdio.h`** header file
  - Contains pre-defined functions that we can use, especially for input/output
  - Many other header files available to use!

The `main()` function /1

- **Required function** in all C programs
  - Note the variation in different textbooks!



- Serves as the **entry point** of all C programs
  - Code is read and executed from top to bottom

The `main()` function /2

- The last statement should be a **`return`** statement
  - Terminates the function
  - Always **0** (*as in the number zero*) to tell the system that everything went well or that the **program ran successfully**!
  - Otherwise, system will think an error has occurred

- **Tip:** Ensure that it does not have any “mistakes” or **syntax errors**
  - Otherwise, your solution will not compile AND execute at all!

The `printf(" ")` function

- A **pre-defined function** from the **`stdio.h`** library
- Function used to **display information** to the user
- Anything **inside the double-quotes** is printed* (notice the `\n`)
- We often use the term **print** but what we really mean is **display**

Comments /1

- **Ignored by the compiler**, they are meant for human beings

- It is **best practice** to include good comments on your code. Why?
  - Serves as a **documentation** that can be referred to in the future
  - Your future self (or whoever takes over) will thank you for doing so
  - Can also be used when **debugging**

Comments /2 - Types

- **One-line** – use **//**
  - Anything after that will be a comment
  - Example: `// This is a comment`

- **Multi-line** – use **/* */**
  - Anything in between the opening `/*` and closing `*/` will be a comment
  - Example:
```
/* This comment
   has two lines */
```

```c
#include <stdio.h>

int main(void) {
    // display a message
    printf("Hello, World!\n");

    // program is done
    return 0;
}
```

Important Notes /1

- If you are coming from Python, you should notice that:

- We terminate **all statements** with a `;` (**semicolon**)

- We are using **curly braces** to **group a set of statements***
  - Called as **compound statements**
  - Basically, if you want to “treat” these statements as if they are a single one

Important Notes /2

- If you miss any of them, you are violating the rules… thus, you have a **syntax error** (i.e., hide and seek)
  - Yes, the `;`
- Unlike Python, **indentation** doesn't matter in C
  - To be more specific, **whitespaces are ignored** by the compiler
- However, it is **best practice** to always properly indent codes to **improve readability**

Basic Program Tracing of Hello World!

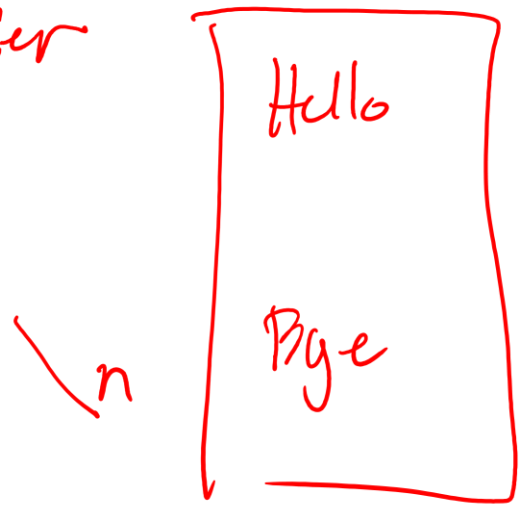
Try tracing the Hello World! source code and determine the output

```c
#include <stdio.h>

int main(void) {
    // display a message
    printf("Hello, World!\n");

    // program is done
    return 0;
}
```

escape character



What is the purpose of the `\n` character in the `printf` statement?

Demo: Compiling your source code

- You can use any IDE or online editor*
- But, can you do it manually?
- Make sure you have **`gcc`** installed on your machine

  `gcc [source_code] -o [executable_file_name]`

  Example: **`gcc hello.c -o hello`**

  ./hello

Your Turn!

- Try it on the Eustis server (attend the lab session)
- You should be able to connect to it via SSH
- Username is your NID while the password is your NID password
  - **Note:** As you type your password, it is normal that you don’t see anything

Activity

```
Hello, World!

My name is <your name>. This is my first C program.

I compiled it myself.
```

Final Notes

- The C compiler ignores whitespaces (spaces, tabs, newlines)
- However, using whitespaces improves readability of your code
- When combined with clear comments, it makes your code easier to understand and maintain
- The next slide shows an example of poor formatting

Do Not Do This!

```
#include <stdio.h>
int main(){printf("Hello, World!\n"); return 0;}
```

Questions?