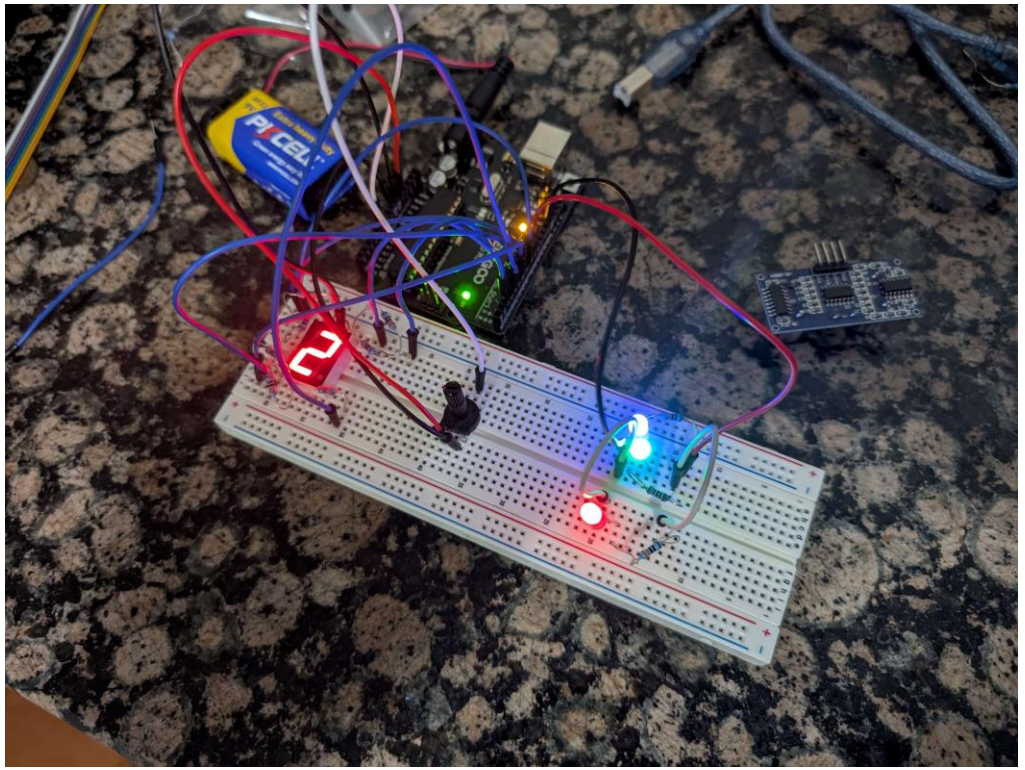
CS 262: Introduction to Low-Level Programming



GEORGE MASON UNIVERSITY

Overview of Lecture 1

Course Logistics

- Instructor Information
- Syllabus
- Honor Code

What is 'Low-Level Programming?'

- How does C differ from Java and Python?

Why are we learning C?

- IoT
- Operating Systems and Firmware
- Robotics

A Simple Program in C

- Creating, compiling, running programs in C

Wrapping Everything Up

- Recap of what we learned
- Some live-coding if we have time

People and Meeting Times

Instructor

Bobby Chab – rchab@gmu.edu

GTA

TBD

Class Meeting Times

Monday, Wednesday from 9:00 AM to 10:15 AM

Innovation Hall Room 136

Labs

Friday from 11:30 AM to 12:20 PM

Enterprise Hall room 276

Grading

Projects

3 projects worth 10% each, for a total of 30%

Quizzes

5 Quizzes, Total of 20%, **lowest quiz is dropped**

Lab Assignments

Total of 10%, **lowest lab is dropped**

Exams

Midterm exam: 15%

Final exam: 25%

Getting Help

If you feel lost, behind, overwhelmed

Come to any of our office hours

Post on Piazza

Email me – rchab@gmu.edu – I'll do what I can to provide supplemental material

Make sure you do something before you get more behind

Not all academic challenges have academic reasons

If anything is affecting you as a student, email me or come to office hours

What To Expect for Quizzes/Exams

My Testing Philosophy

Quizzes and exams should accurately assess what you've learned in the course

Quizzes and exams shouldn't have surprises

If too many students get a question wrong, that's my fault

Small mistakes should not incur large penalties

Quizzes

Given in class during the last 20 minutes, announced 1+ week ahead of time

Exams

The midterm will take place in this room, and last for the duration of the class

The final will be scheduled for a 2-and-a-half-hour block

Course Communications

Piazza

The main forum we use for discussions and questions – monitored regularly by TAs

If your question requires posting code, **make sure it is a private post** for only instructors and TAs

We post important stuff here – please do check it regularly

Email

You can email me at any time with questions on the material

For grading questions, these must be handled via email

Contacting Me

Email: rchab@gmu.edu

If I don't respond within 24 hours, **please email me again**

Office hours will be in Buchanan D215G from 2:00 to 3:00, Tuesdays and Thursdays

If these times don't work for you, feel free to email me for an appointment

Who Grades What?

Professor

- Exams
- Quizzes

GTA

- Projects
- Labs

Grade contests/questions

- If you have any questions about a grade you received, please email us so we can make sure we are all on the same page

Projects

We will have three projects

Each is worth 10% of your total course grade

Late projects will be penalized 10% per day

Everyone gets 3 late tokens

Projects can be developed on any environment, but they must compile and run on Zeus

C isn't the same on every system. Make sure your output is correct on Zeus as this is what will be considered for correctness

Quizzes

We will have a total of 5 quizzes

Quizzes will be given in class, in the last 20-25 minutes

Quizzes are worth a total of 20% of your grade (lowest score is dropped)

Each quiz will be announced at least one week in advance

Quizzes are designed to prepare you for exams

If you must miss a quiz, **please contact me as early as possible**

Labs

We will have two labs per week

Labs meet in Enterprise Hall room 276 on Fridays, from 11:30 AM to 12:20 PM

Total of 11 lab assignments, worth 10% of your total grade (lowest dropped)

Labs will be due 1-2 weeks after the lab session takes place

Exams

Midterm exam

Worth 15% of your total grade

Final exam

Worth 25% of your total grade

To pass this course, you **must** either

- Score >= 60% on the final, OR
- Have an average exam grade >=65%
- This is department policy, and I have no control over it

Academic Integrity

You aren't allowed to

- Share code with any other student
- Receive code from any other student
- Use code from online sources (Github, etc)

Please do protect your code from being shared

- Do not put your project code in any **public** repository online
- Use common sense about things like leaving your computer unattended

ChatGPT – My Thoughts

Why were you allowed to use calculators on math tests in high school, but not elementary school?

Automating tasks that are beneath your current level

An incredible tool, and a potentially slippery slope

All I have to say is - please do not outsource your thinking

ChatGPT as a learning tool

An excellent way to get a start on learning new topics

The fact it can be wrong necessitates critical thinking

Lecture Style

During class, we will:

Discuss the topic(s) we will be learning during the class

I'll do live-coded examples so we can explore the fascinating (and often unexpected) behavior programming in a low-level language

In some class sessions we might go through two lectures, and in some we may only get through part of one. Certain topics take much longer than others to explore thoroughly

Why Are We Learning C When Python Exists?

Robotics

C is the language of choice for robotics because it allows for precise control over hardware

IoT

Simple devices that don't even have an OS can still run C

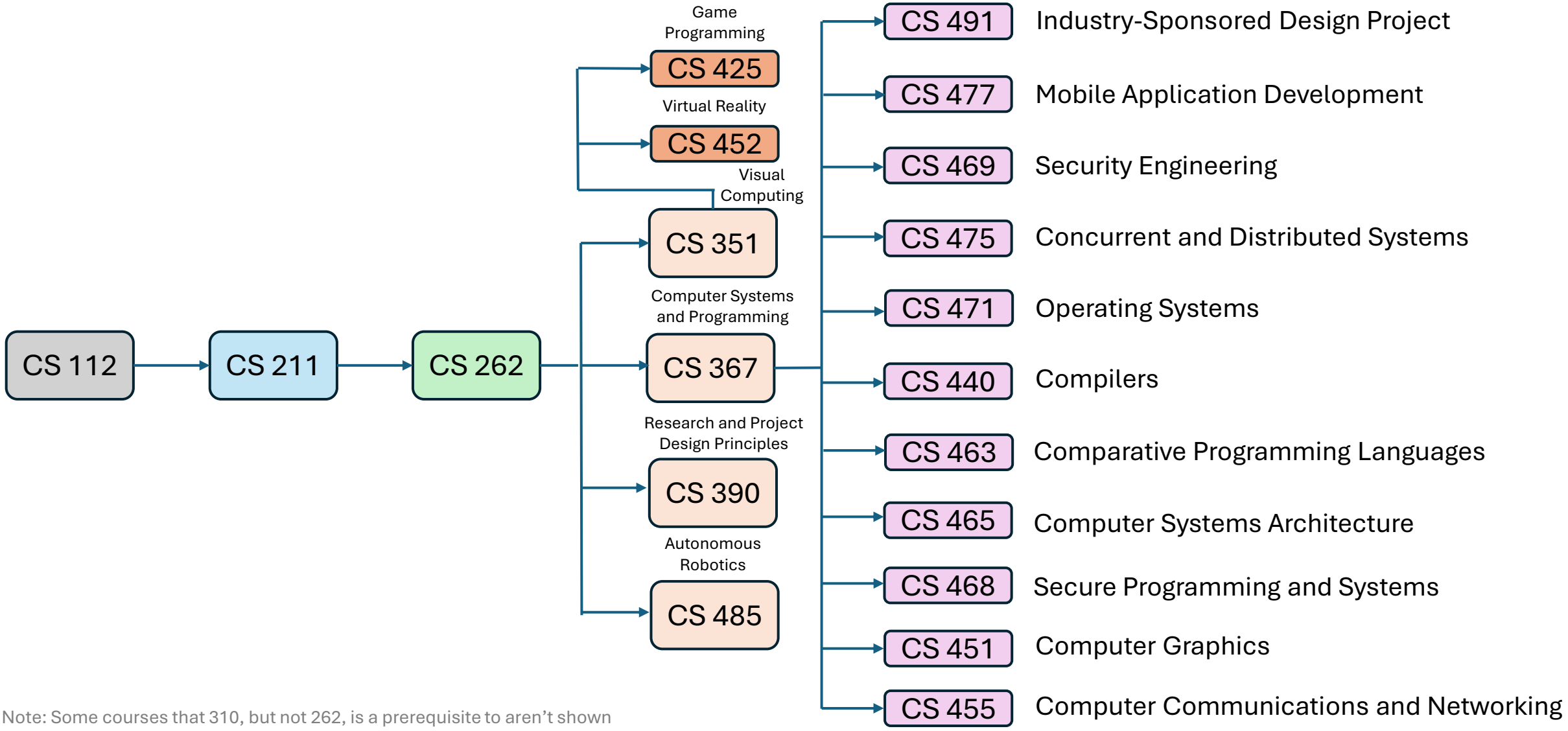
Efficiency

When it is crucial for things run as quickly as possible, C has other languages bested

Fun

It's genuinely enjoyable to see how things happen at the 'low level'

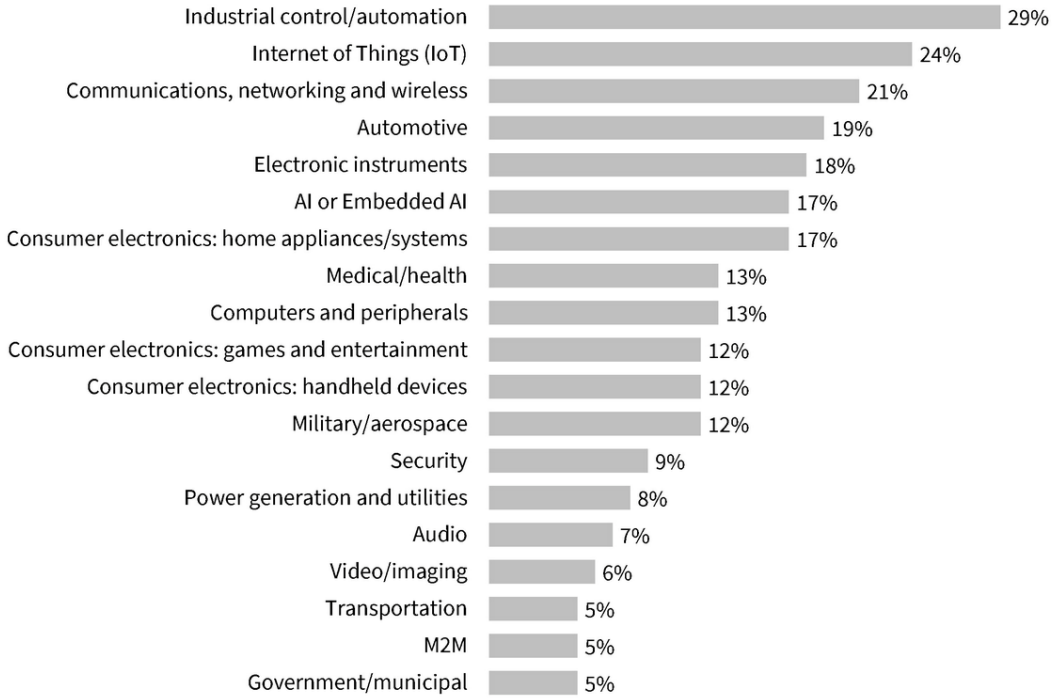
What is 262 A Gateway To at Mason?



Note: Some courses that 310, but not 262, is a prerequisite to aren't shown

How About The Job Market?

How much of different market segments involve embedded systems?



Programming languages used in the embedded systems market

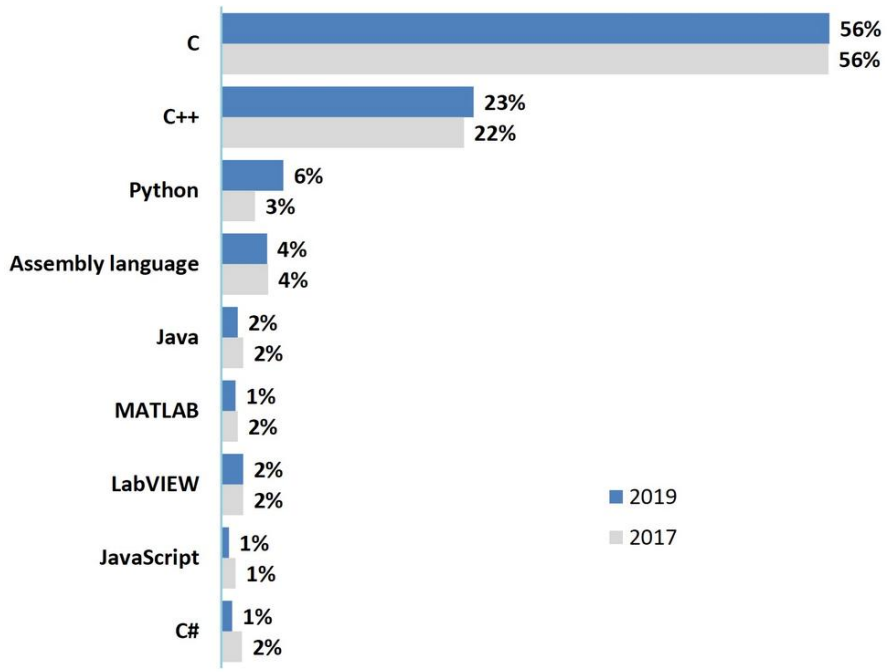


Figure 4. Programming language usage. (Source: 2019 Embedded Market Study)

Sources: https://www.embedded.com/wp-content/uploads/2023/06/Embedded-Market-Study-Report-2023-applications.jpg
https://www.embedded.com/2019-embedded-markets-study-reflects-emerging-technologies-continued-c-c-dominance/

What Will You Be Able To Do After Taking This Course?

Write fairly advanced programs in C, which means you will

- Have knowledge of all the basics of C
- Have knowledge of important C libraries
- Most importantly, you will be able to manipulate data in a way not possible with Java or Python

Understand what is going on at the hardware level, like

- How things are stored in memory
- How to ‘talk to’ the actual hardware of the computer

Be able to use UNIX tools for debugging

- Modern debuggers can fall short in specialized settings

What Does ‘Low-Level Programming’ Mean?

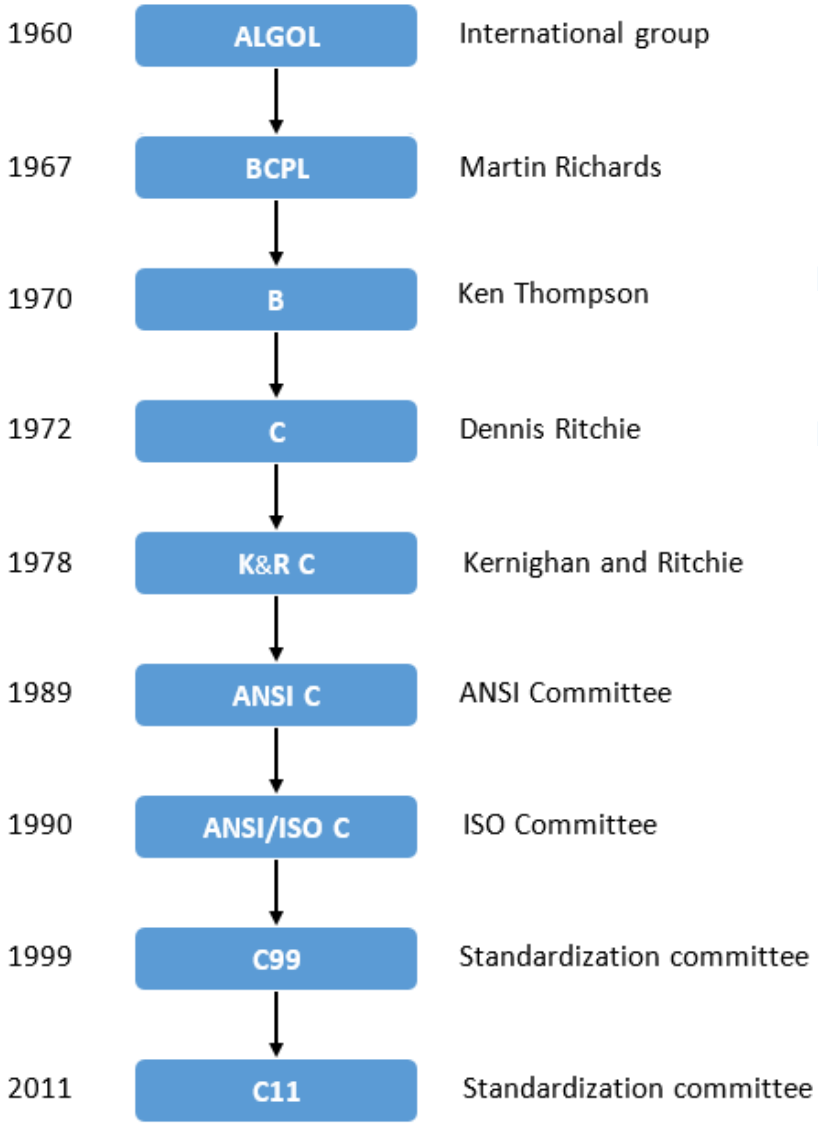
Python, the penthouse: High-level, abstract, and elegant

Java, the mid-level: structured, efficient, and with a clear view of the high and low levels

C, the ground-level: close to the machinery, and where the foundation connects to the building

We won’t talk about what’s beneath the ground floor just yet. That’ll happen in 367

History of C



B was slow as it was an interpreted language (runs on software on top of the CPU)

Ken Thompson supported Dennis Ritchie in the development of C

This is the version we use

https://codeforwin.org/c-programming/introduction-c-programming

A Few More Details about C

C is technically a ‘medium-level’ language

We can use it as a low-level language and directly access hardware

It still has structures of a high-level language

C is statically typed

All variables have a set type

C is a compiled language

C code must be translated to machine code before it’s executed

Interpreted languages are instead executed line-by-line by an interpreter at runtime

C is an imperative language

This just means we write a sequence of statements to perform tasks

Some Challenges and Downsides of C

It's easy to write spaghetti code in C

No 'graceful termination' via exception handling

It is 'unsafe' since it allows direct manipulation of memory

Much of what makes the language powerful also makes it unsafe

Basic Anatomy of a C Program

```
#include <stdio.h>

int main() {
    printf("Hello CS262 !\n");
    return 0;
}
```

This includes the standard input output header file, *similar* to an import in Python or Java

The main function is the "entry point" of our program

This prints out "Hello CS262 !", followed by a line break to the console

This returns 0, indicating our program exited successfully

Comments

Comment Types

Single-line comments are done with a double slash: `//`

Block comments must be enclosed in: `/* ... */`

Comments should

State **what** a block of code is doing

State **why** a block of code is there

Help others (or you in the future) read your code and understand your logic

```
int main() { // This is a single line comment
   printf("Hello CS262 !\n");
   /* And this is a block comment
   printf("Hello again CS262 !\n");
   printf("Hello once again CS262 !\n"); */
   return 0;
}
```

How Is A C Program Organized?

Functions are like methods in Java

Local Definitions

Start with local variables.
(Always initialize each variable)

Statements

Statements always end with **;** in C.
These will look a lot like Java
End with a **return** statement.

How Is A C Program Organized?

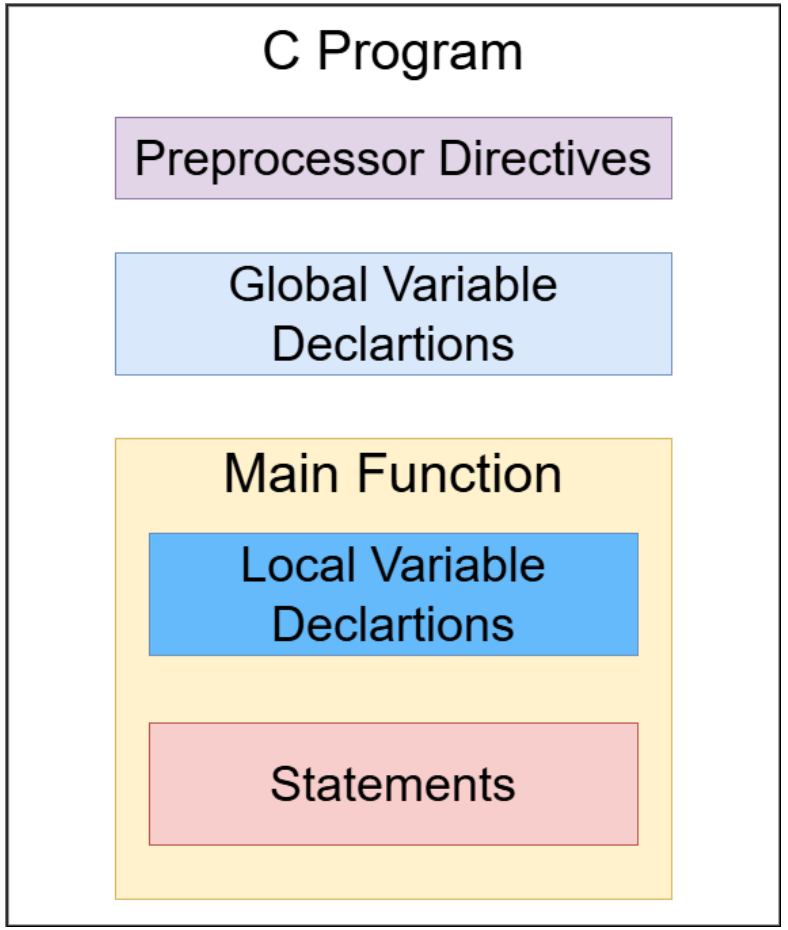
```c
#include <stdio.h>

int global_var = 262;

int main() {
    int local_var = 211;

    printf("Global variable: %d\n", global_var);
    printf("Local variable: %d\n", local_var);

    return 0;
}
```



How Is A C Program Organized?

```c
// Preprocessor Directives
#include <stdio.h>

// Global Declarations
int global_variable = 10;

int main(void)
{
    // Local Definitions
    int local_variable = 20;

    // Statements
    printf("Global Variable: %d\n", global_variable);
    printf("Local Variable: %d\n", local_variable);

    return 0;
}
```

Including the standard input/output system library

A global variable, accessible by all functions in the file

Main function – entry point for the program

A local variable, accessible only within the function where it is declared

Print the value of the global variable

Return exit code 0 to the operating system

The format specifier for an integer

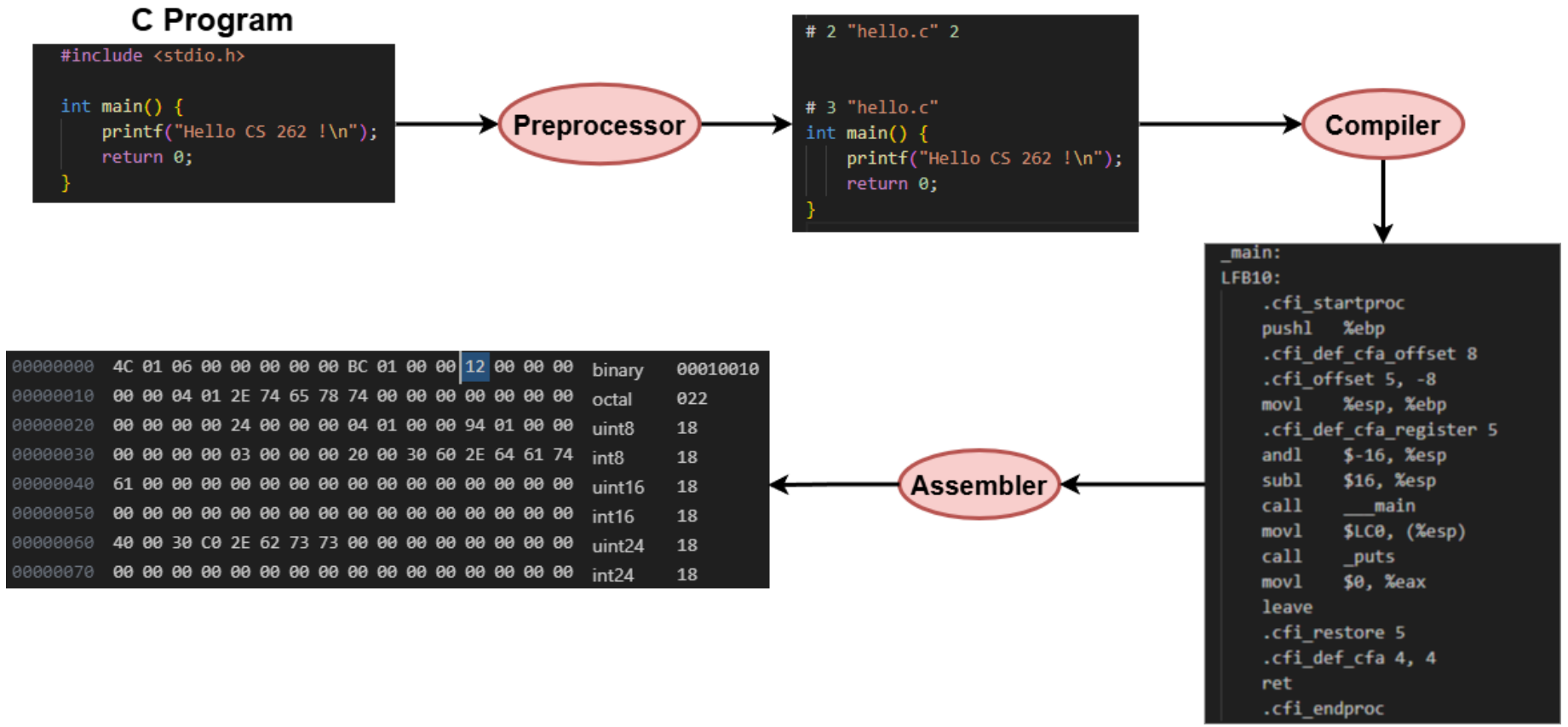
Common Format Specifiers

Specifier	Meaning	Example Input	Example Output
`%d`	Integer	`50`	`50`
`%i`	Also integer	`-20`	`-20`
`%u`	Unsigned integer	`100`	`100`
`%f`	Floating point	`3.1416`	`3.1416`
`%.2f`	Floating point, 2 decimals	`3.1416`	`3.14`
`%s`	String	`"Hello"`	`Hello`
`%c`	Single Character	`'A'`	`A`
`%X`	Hexadecimal	`255`	`FF`

Format specifiers tell `printf()` how to format and display what is passed to it. They are essentially placeholders to specify how values should be printed

What is Happening When We Do This?

Our computer can't natively recognize the C code, so it must be turned into some form it can recognize



Compiling and Running a C Program

Compiling C on a Unix-based System

```
gcc -o <output> <input>
```

```
gcc -o hello hello.c
```

Running a C program on a Unix-based system

```
./<name_chosen_for_output>
```

```
./hello
```

Zeus Refresher

If you are new to Zeus (or have repressed the memories of using it)

If you are on-campus, you must connect to either the Mason Secure or Eduroam Wifi networks. It will not work if you are connected to Mason

If you are off-campus, you must connect to the Cisco VPN before connecting to Zeus

VPN: https://its.gmu.edu/service/virtual-private-network-vpn
SSH Connections: https://labs.vse.gmu.edu/index.php/FAQ/SSH
General Zeus Info: https://labs.vse.gmu.edu/index.php/Systems/Zeus

Using VIM on Zeus (Or Any Linux System)

VIM Basics Quick Reference

Feature	Command
Edit Mode	`i`
Command Mode	`esc`
Save and Exit	`:wq` while in command mode
Exit Without Saving	`:q!` while in command mode
Tutorial	Type `vimtutor` into terminal

Building a C Program on Zeus – Start to Finish

Step 1: Connect to Zeus

```
ssh rchab@zeus.vse.gmu.edu
```

Step 2: Create the .c file

```
[rchab@zeus-1 CS262]$ vi hello.c
[rchab@zeus-1 CS262]$
```

Step 3: Write the program

```c
#include <stdio.h>

int main() {
        printf("Hello World !\n");
        return 0;
}
```

Step 4: Compile and run

```
[rchab@zeus-1 CS262]$ gcc -o helloworld hello.c
[rchab@zeus-1 CS262]$ ./helloworld
Hello World !
[rchab@zeus-1 CS262]$
```

Moving Files To and From Zeus

Copying from your computer to Zeus

```
scp file_name username@zeus.vse.gmu.edu:~/.
```

Source

Destination

./ is your home directory

Copying from Zeus to your computer

```
scp username@zeus.vse.gmu.edu:~/dir/filename .
```

Source

Destination: the . Means your local directory

Moving Files To and From Zeus

Alternatively, you can use a client like PuTTY or WinSCP

https://www.putty.org/

https://winscp.net/eng/index.php

Practice

Let's Practice

Pay close attention – These resemble what you will see on quizzes and exams

True or False: A C program can be run on any system **without** being compiled first

Answer: _____

The job of the linker is:

a) Turning your C code into assembly language

b) Connecting you to Zeus

c) Combining what the compiler generates with libraries to form an executable file

Unofficial Homework

Read through chapter 1 of the book

- 1.1 – Hello World in C
- 1.2 – Basic types
- 1.2, 1.3 – Expressions and loops
- 1.6, 1.9 – Arrays, Strings
- 1.7 - Functions

Familiarize yourself with the syllabus

Play around with the code examples

Remember - the only way to learn programming is by programming

Any Questions/Comments?

Any feedback is always welcome