

Why C Programming?

2024 Winter APS 105: Computer Fundamentals

Jon Eyolfson

Lecture 1

1.0.1

I'm Jon, Your Instructor

Eyolfson

I'm Jon, Your Instructor

Eyolfson

I'm Jon, Your Instructor

Elf

I'm Jon, Your Instructor

Elf son

I'm Jon, Your Instructor

E Ifson

I'm Jon, Your Instructor

Eyolfson

There's 4 Course Entries in Your Timetable

Lectures

Where we learn concepts

Plenary Lectures

Additional practice for labs

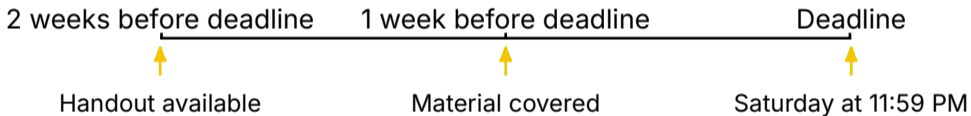
Tutorials

Additional practice for exams

Labs

In-person lab help from TAs

Start the Labs Early!



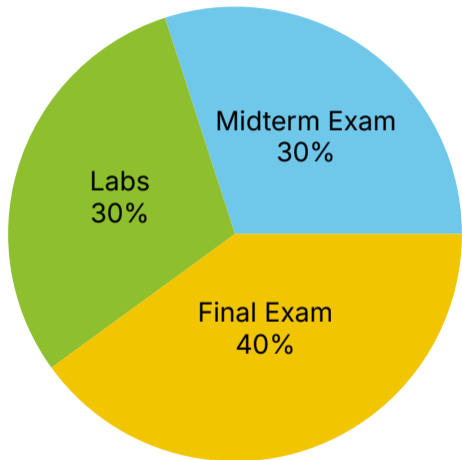
Academic Integrity is Serious

You may discuss course content, you do not copy it!

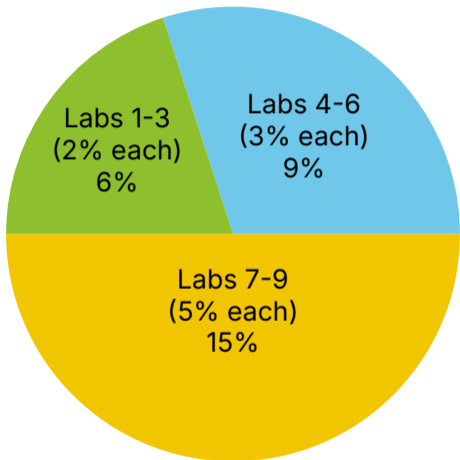
You're primarily harming your own learning

It's much easier to detect copying software

The Majority of Your Grade is Exams



There are 9 Labs, Progressively Weighted More



Officially, We Use Quercus and an Online Textbook

Direct Quercus link: <https://q.utoronto.ca/courses/330896>

Online Textbook: <https://learningc.org/> (no other materials required)

Piazza: <https://piazza.com/utoronto.ca/winter2024/aps105/home>

Additional Resources for Section 3

Lecture livestreams and recordings: <https://youtube.com/@eyolfson>

Lecture slides (and YouTube links): <https://eyolfson.com/courses/aps105/>

Discord community: <https://compeng.gg/discord/join/aps105/>

In This Course, You Tell Your Computer What to Do

Computers execute (or run) a **program**

The art of writing a program is **programming**

Why C Programming, Specifically?

Humans write programs in a [programming language](#)

C is a programming language, first appearing in the 1970s

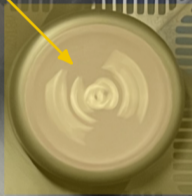
C is a small language, with few features, it's closer to how a computer works

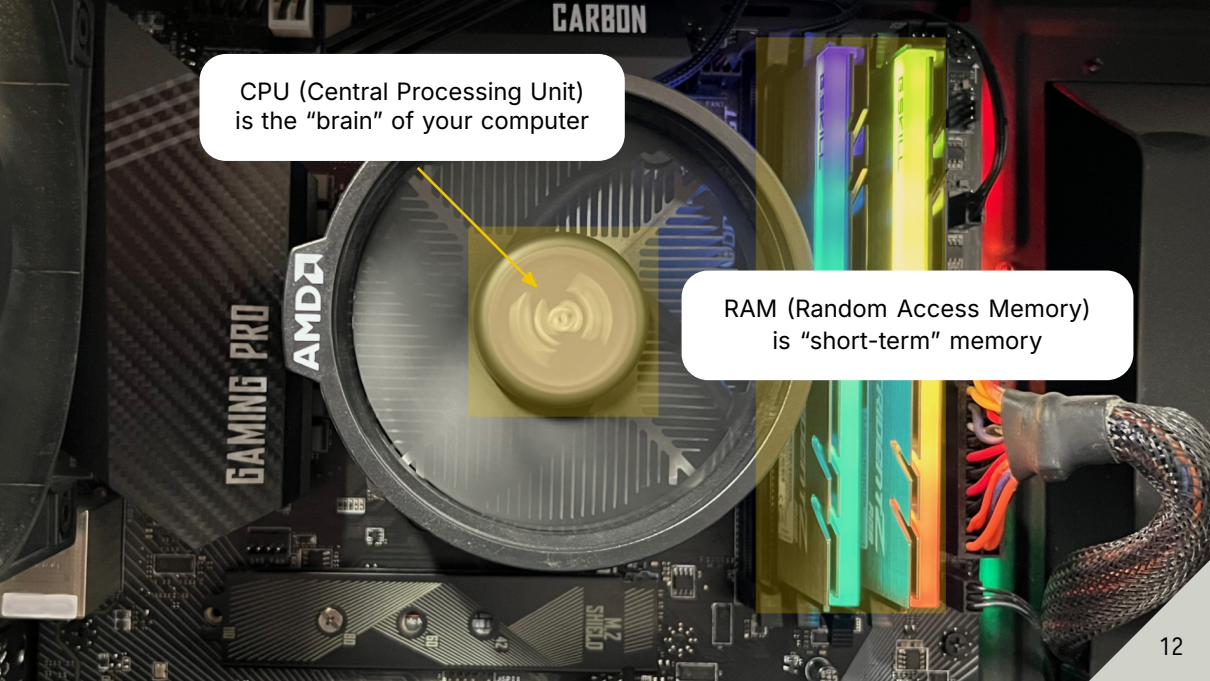
What is a Computer?

Hardware is the physical components of a computer



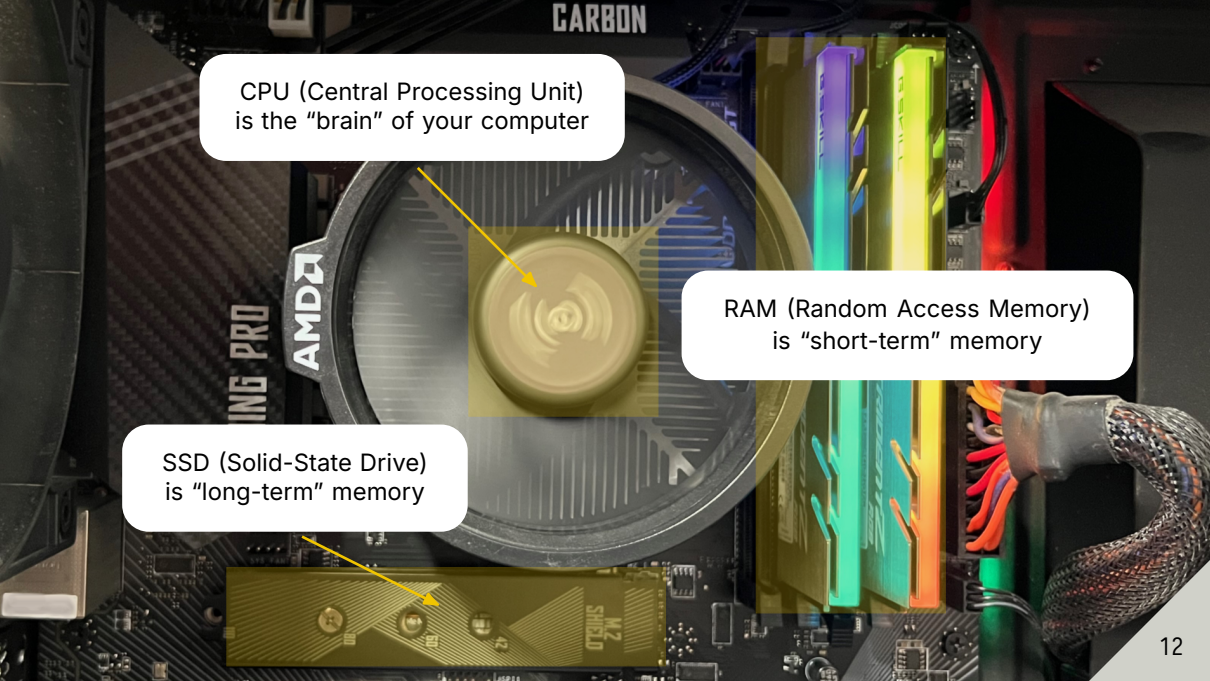
CPU (Central Processing Unit)
is the "brain" of your computer





CPU (Central Processing Unit)
is the "brain" of your computer

RAM (Random Access Memory)
is "short-term" memory

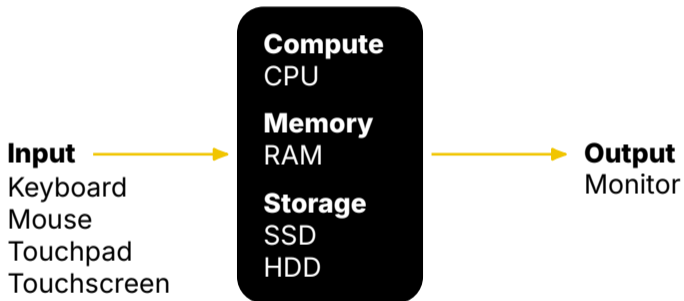


CPU (Central Processing Unit)
is the "brain" of your computer

RAM (Random Access Memory)
is "short-term" memory

SSD (Solid-State Drive)
is "long-term" memory

We Model a Computer as a Black Box



What's the Other Part of a Computer?

Software is the information the computer needs to run

The most important information is **instructions** that tell the CPU what to do

How Do Computers Store Information?

Computers use numbers and humans assign meaning to them

At the most basic level, they can only store 0 or 1



ATTENTION: NE PAS
LUMINAIRE ENFERMÉ
LUMINAIRE SUIVABLE
FOR DAMP LOCATIONS
EMPLACEMENTS HUMIDES

Off
0

CAUTION: NOT FOR
TOTALY ENCLOSED LUMINAIRE ENTRE
LUMINAIRE SUITABLE FERME COMMENTA
FOR DAMP LOCATIONS. EMPACEMENTS HUM

On
1

We Usually Represent Numbers in Decimal

We represent numbers as a sequence of digits

Digits are numbers between 0-9 (10 options)

Computers represent numbers as a sequence of bits

A **bit** (binary digit) is either 0 or 1 (2 options)

Binary is a numeral system that only uses bits

Let's Represent the Decimal 176

100	10	1
1	7	6

Let's Represent the Decimal 176

10^2	10^1	10^0
1	7	6

Let's Represent the Decimal 176

2^7 2^6 2^6 2^4 2^3 2^2 2^1 2^0

Let's Represent the Decimal 176

128 64 32 16 8 4 2 1

Let's Represent the Decimal 176

128	64	32	16	8	4	2	1
1	0	1	1	0	0	0	0

Let's Represent the Decimal 176

128	64	32	16	8	4	2	1
1	0	1	1	0	0	0	0

Verify, $128 + 32 + 16 = 176$

Computers Store Information in a Finite Amount

A **byte** is a binary number that's 8 bits long

It can represent 256 (2^8) different things

In math we'd say $(5)_{10} = (101)_2$, in most programming languages

5 is the same as `0b101`

`0b` is a prefix that says the number that follows is in binary

Note, there are implicit leading zeros so

`0b0000101` is the same as `0b101`

We Can Represent Letters and Numbers in Binary

We don't need a full byte, assume the most significant bit is always 0

We have 128 different representations (2^7)

A **character** is how a computer represents English
such as: letters, digits, punctuation, and spaces

For instance, "A" is represented as `0b01000001` or `65`

Your Program (or Applications) Run in an Operating System

The operating system (OS) is software that interacts directly with hardware
(You'll learn much more about this in ECE 344)

Code is text written in a programming language

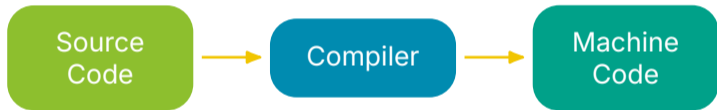
Source code is code for particular software

A library is code written by someone else that you can use

C provides a library called: the standard C library (libc)

We Need to Use a Program to Create Programs

A **compiler** transforms your source code into a program your OS can run



Machine code is the binary representation of instructions the CPU can run

We often use **compile** as a verb: "Let's compile our program."

We Need Another Program to Handle Input and Output for Us

A **terminal** uses a keyboard for input and a monitor for output

It is a text interface (it uses characters you see on your keyboard)

For this course, all the programs we create use a text interface

Your First Program

```
#include <stdio.h>


int main(void) {
    printf("Hello world\n");
    return 0;
}
```

Your First Program

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

```
int main(void) {  
    printf("Hello world\n");  
    return 0;  
}
```

`#include` reads the contents of another file,
`stdio.h` contains the **declaration** of `printf`



Your First Program

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

```
int main(void) {  
    printf("Hello world\n");  
    return 0;  
}
```

`#include` reads the contents of another file,
`stdio.h` contains the `declaration` of `printf`

`defines` a `function` called `main` that outputs a
number (OS starts executing `main` always)

Your First Program

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

```
int main(void) {  
    printf("Hello world\n");  
    return 0;  
}
```

`#include` reads the contents of another file, `stdio.h` contains the **declaration** of `printf`

`int main` defines a **function** called `main` that outputs a number (OS starts executing `main` always)

`printf` calls a function called `printf` that outputs a **string** (sequence of characters) to the terminal

Your First Program

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

```
int main(void) {  
    printf("Hello world\n");  
    return 0;  
}
```

`#include` reads the contents of another file, `stdio.h` contains the **declaration** of `printf`

`main` defines a **function** called `main` that outputs a number (OS starts executing `main` always)

`printf` calls a function called `printf` that outputs a **string** (sequence of characters) to the terminal

`return 0` stop execution of `main` and output the value `0` (OS interprets `0` as "no errors")