`main` function

When you run a program, it always runs your `main` function

- Needs to be written a specific way
- Should always be at the bottom of your file

```c
int main() {
    // The code between the curly brackets is what gets executed
    // when the program runs.

    printf("hello world!");
}
```

- CodeBlocks should provide this to you already in a `main.c` file

Variables

Declaring a variable

```
// variable declaration
int x = 2;
```

- `<stuff> = <stuff>`
- On LHS: `<type> <name of variable>`
- On RHS: value of variable

Exercise: what are the types, names, and values of each variable?

```
int y = -3;
int z = 10000;
double d = 2.0;
float f = 1.417;
char c = 'a';
```

What is a type?

- Cambridge dictionary definition: “a particular group of people or things that share similar characteristics and form a smaller division of a larger set”
- Simple definition: a group / style of similar things

- In programming, used to define what a variable is and what you can do with it

You can only use **primitive** types unless you do more work

- This will not work because `asdfdafjlksdjfksadjf` and `fruit` are not primitive types.

```
asdfdafjlksdjfksadjf failure = 2;
fruit banana = yellow;
```

- More on creating custom types (structs) in a later lesson.

Table of primitive types (not including pointers)

Data type	Size(bytes)	Range	Format String
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short	2	-32,768 to 32,767	%d
unsigned short	2	0 to 65535	%u
int	2	32,768 to 32,767	%d
unsigned int	2	0 to 65535	%u
long	4	-2147483648 to +2147483647	%ld
Unsinged long	4	0 to 4294967295	%lu
float	4	-3.4e-38 to +3.4e-38	%f
double	8	1.7 e-308 to 1.7 e+308	% lf
long double	10	3.4 e-4932 to 1.1 e+4932	%lf

https://miro.medium.com/v2/resize:fit:712/1*cemNFCrMA3MK27nCuUuG_Q.png

Mixing up types is not recommended for now

- With primitives, this makes the compiler do something called **casting**
  - Attempts to convert variable's value to the type on LHS

```c
int x = 1.4;
printf("%d\n", x); // prints 1, not 1.4 -- why?
```

- With custom types, causes a compiler error

Arithmetic demo

References for arithmetic

- Adding, subtracting, and multiplying work as expected
- **Integer** division vs **floating point** division

```
int truncated = 5 / 2; // evaluates to 2
float not_truncated = 5.0 / 2.0; // evaluates to 2.5
```

- Modulus operator % gives remainder of division operation

```
int remainder = 84 % 5; // evaluates to 1
```

Printing demo

References for printing

- Print functions need to be **imported** because they reside in **headers** in the standard C libraries
  - This just means you need to have `#include <header name>` at the top of your file

```c
#include <stdio.h>

int main() {
    printf("hello world!");
}
```

- For this class, use `<stdio.h>`
- In C++ you use `<iostream>`

`printf` formatting reference

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

https://i.stack.imgur.com/VRH1V.png

Conditionals demo

References for conditionals

- Variables resolve to `true` or `false` values
  - 0 is false, not 0 is true
  - In C++ you have bool type for dedicated true / false values
- `if` is the most basic way to check for a variable's true/false value
  - Checks value of what is inside parentheses

```c
if (0) {
    printf("This does not get printed because 0 evaluates to false.");
}

if (1) {
    printf("This gets printed because not 0 evaluates to true.");
}

if (12348) {
    printf("This also gets printed...");
}

if ('B') {
    printf("...and so does this!");
}
```

Relational operators

- These expressions evaluate to 1 or 0, i.e. true or false

==	Equal to	a==b returns 1 if a and b are the same
>	Greater than	a>b returns 1 if a is larger than b
<	Less than	a<b returns 1 if a is smaller than b
>=	Greater than or equal to	a>=b returns 1 if a is larger than or equal to b
<=	Less than or equal to	a<=b returns 1 if a is smaller than or equal to b
!=	Not equal to	a!=b returns 1 if a and b not the same

https://fastbitlab.com/wp-content/uploads/2022/07/Figure-1-26.png

Loops demo

References for loops

- Use these to do something many times
- **while** loops

```c
int x = 0;
while (x < 5) {
    printf("x is now equal to %i", x);
    x++;
}
```

- **for** loops

```c
for (int x = 0; x < 5; ++x) {
    printf("x is now equal to %i", x);
}
```

Further reading and exercises

If you want more English-language materials...

- In C++:
  - University of Michigan: EECS 183
- In Python:
  - Carnegie Mellon University: CS 15-112
  - Stanford University: CS106A

- Keep in mind: C++ and Python have very different rules...
  - ...but the foundations for both languages are similar

Complete basic coding exercises

- Google “GitHub”
- In GitHub, search for `michigan-musicer`
- Search for exercise_1 repository
- Download the files in the repository and create a CodeBlocks project with them
- Fill in the blanks where it says “YOUR CODE HERE”

Contact me

email: kwang@al.edu.pl

Office hours: by appointment (send an email!)

Please reach out with any questions about assignments, computer science at an American university, the tech industry, etcetera.

See you next week!