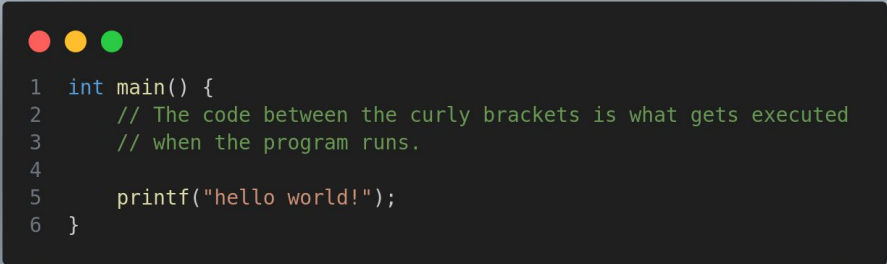


main **function**

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

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a C program's main function with line numbers 1 through 6 on the left. The code is: 1 int main() {, 2 // The code between the curly brackets is what gets executed, 3 // when the program runs., 4, 5 printf("hello world!");, 6 }.

```
1 int main() {
2     // The code between the curly brackets is what gets executed
3     // when the program runs.
4
5     printf("hello world!");
6 }
```

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

Variables

Declaring a variable



```
1 // variable declaration
2 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?




```
1  int y = -3;  
2  int z = 10000;  
3  double d = 2.0;  
4  float f = 1.417;  
5  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 `asdfdafjlkdsjfkksadjf` and `fruit` are not primitive types.



```
1 asdfdafjlkdsjfkksadjf failure = 2;
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
Unsigned 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

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains two lines of C code:

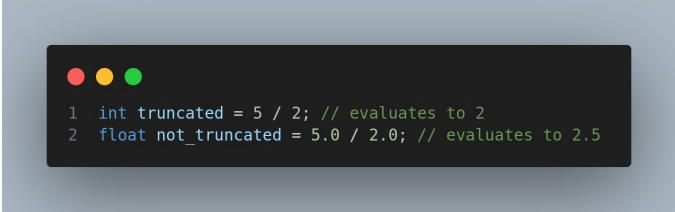
```
1 int x = 1.4;  
2 printf("%d\n", x); // prints 1, not 1.4 -- why?
```

- 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



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

- Modulus operator % gives remainder of division operation




```
1 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
- For this class, use `<stdio.h>`
- In C++ you use `<iostream>`



```
1  #include <stdio.h>
2
3  int main() {
4      printf("hello world!");
5  }
```

printf formatting reference

<i>specifier</i>	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

```
1  if (0) {  
2      printf("This does not get printed because 0 evaluates to false.");  
3  }  
4  
5  if (1) {  
6      printf("This gets printed because not 0 evaluates to true.");  
7  }  
8  
9  if (12348) {  
10     printf("This also gets printed...");  
11 }  
12  
13 if ('B') {  
14     printf("...and so does this!");  
15 }
```


Relational operators

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

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



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

- for loops



```
1 for (int x = 0; x < 5; ++x) {
2     printf("x is now equal to %i", x);
3 }
```

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!