C++ Programming

Week 2:

Basic Types

Dr. Owen Chen
Cary Chinese School
Director of Math and Computer Science

2023 Summer

Week 2: Agenda

- Review software installs and resolve any installation issues
- Create, Compile and Run first C++ program – Hello World
- C++ Variables
- C++ Basic Types (integer, char, bool, float)
- C++ operators

Review Software Installs

- Install WSL on Windows
- Install VS Code
- Install C++ Compiler
- Configure VS Code

Install WSL on Windows

- Windows Subsystem for Linux (WSL) is required for this course
- Please install it by following this page:
- https://learn.microsoft.com/en-us/windows/wsl/install
- Follow this page to set up a username and a password for your Linux account:
- https://learn.microsoft.com/en-us/windows/wsl/setup/environment#set-up-your-linux-username-and-password
- Write down your password – if you will need later when you run “sudo” command

• Install VS Code

- Download VS Code here: https://code.visualstudio.com/
- Follow the setup instruction:
- Windows:https://code.visualstudio.com/docs/setup/windows
- Mac: https://code.visualstudio.com/docs/setup/mac

Install the Compilers

Install the last gcc/g++ on WSL

```
$ sudo apt update
$ sudo apt install gcc g++ make gdb
$ gcc --version
$ g++ --version
```

Install the last clang on Mac

Run the following command from a terminal (iTerm):

```
$ xcode-select --install
```

The above command will install C++ compiler clang

How to Compile?

Compile C++ programs in g++ (for WSL):

```
$ g++ <program.cpp> -o  program
```

Compile C++ programs in clang (for Mac):

```
$ clang  <program.cpp> -o  program
```

Hello World

Organize our files in a new folder under your home: ~/cpp

- **Open your terminal:**
  - Ubuntu App in Windows
  - iTerm in Mac

- **Go to home directory (cd)**
- **Create a folder called "cpp" using mkdir:**

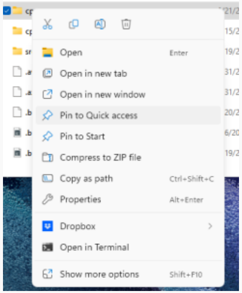
```
$ cd
$ mkdir cpp
```

Create a shortcut on your Desktop (for Windows)

- **Open File Explorer, go to:**

  Linux -> Unbuntu -> home -> <your username>

- **Right Click on “cpp” folder,**
  - Select “Pin to Quick access”
  - Select “Pin to Start”



Start VS Code from your cpp folder for both Windows and Mac

- **You can always go to cpp folder anywhere with this command:**

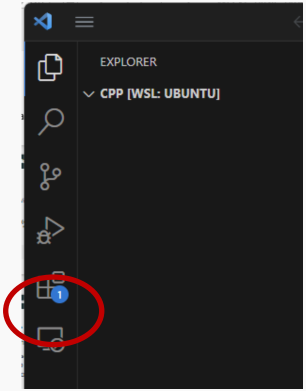
```
$ cd ~/cpp
```

- **From your terminal, type the following commands to start VS Code from your cpp folder:**

```
$ cd ~/cpp
$ code .
```

Install C++ extensions in VS Code

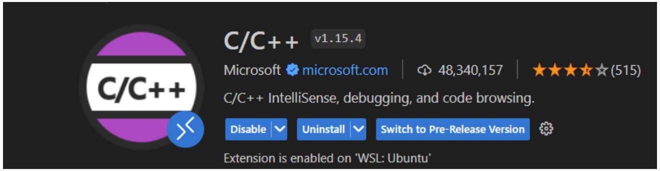
- **Click on Extensions Icon on Left Bar**
- **or type Ctrl+Shit+X**



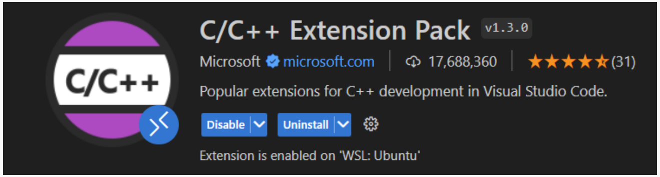
- **It will Extensions**
- **Type “C++” in the search**
- **Install**

Install C++ extensions in VS Code

- **Install these two packages:**
    - C/C++ by Microsoft
    - C/C++ Extension Pack by Microsoft


C/C++ v1.15.4
Microsoft microsoft.com | 48,340,157 | ★★★★☆ (515)
C/C++ IntelliSense, debugging, and code browsing.

Extension is enabled on 'WSL: Ubuntu'


C/C++ Extension Pack v1.3.0
Microsoft microsoft.com | 17,688,360 | ★★★★★ (31)
Popular extensions for C++ development in Visual Studio Code.

Extension is enabled on 'WSL: Ubuntu'

Hello World

Open VS Code and create our first C++ program: hello_world.cpp

```cpp
#include <iostream>
int main() {
    std::cout << "Hello World!\n";
}
```

std::cout
represent the standard output stream

Compile and Execute Programs

Open your VS Code Terminal: Top Menu -> View ->Terminal

```
$ g++  hello_world.cpp  -o  hello_world
$ hello_world
```

Compile C++ programs in clang (for Mac):

```
$  clang hello_world.cpp  -o  hello_world
$ hello_world
```

Hello World

The previous example can be written with the global `std` namespace:

```cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!\n";
}
```

Hello World

Open VS Code and create a C version of hello world program: hello_world.c

```
#include <stdio.h>

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

`printf`

prints on standard output

I/O Stream (std::cout)

`std::cout` is an example of *output* stream. Data is redirected to a destination, in this case the destination is the standard output

C:

```c
#include <stdio.h>
int main() {
    int    a   = 4;
    double b   = 3.0;
    char   c[] = "hello";
    printf("%d %f %s\n", a, b, c);
}
```

C++:

```cpp
#include <iostream>
int main() {
    int    a   = 4;
    double b   = 3.0;
    char   c[] = "hello";
    std::cout << a << " " << b << " " << c << "\n";
}
```

Variables and Basic Types

Integter Data Types

int

- **int is the most frequently used integer type**

```
int i; //declare a variable
int j = 10; //declare and initialize
int k;
k = 20; //assign a value
```

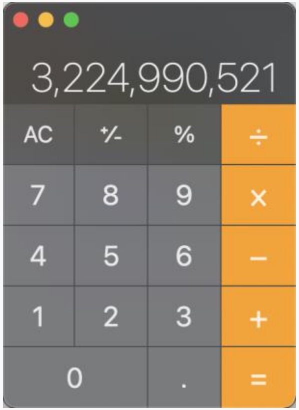
- **Remember to initialize a variable!**
- **Will the compiler give an error?**

```
int i;
cout << i; //what is i's value?
```

How to initialize a variable

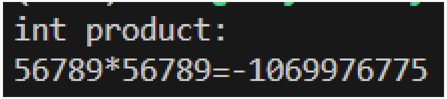
```
int num;
num = 10;//do not forget this line

int num = 10;

int num (10);

int num {10};
```

Overflow



Int_overflow.cpp

```
int main(){
    int a = 56789;
    int b = a;
    int c = a * b;
    cout << "int product:" << endl;
    cout << a << "*" << b;
    cout << "=" << c << endl;
```



```
int product:
56789*56789=-1069976775
```

Arithmetic Types

Type	Bytes	Range	Fixed width types
bool	1	true, false	
char [†]	1	-127 to 127	
signed char	1	-128 to 127	int8_t
unsigned char	1	0 to 255	uint8_t
short	2	-2^{15} to 2^{15}-1	int16_t
unsigned short	2	0 to 2^{16}-1	uint16_t
int	4	-2^{31} to 2^{31}-1	int32_t
unsigned int	4	0 to 2^{32}-1	uint32_t
long int	4/8		int32_t/int64_t
long unsigned int	4/8*		uint32_t/uint64_t
long long int	8	-2^{63} to 2^{63}-1	int64_t
long long unsigned int	8	0 to 2^{64}-1	uint64_t
float (IEEE 754)	4	$\pm 1.18 \times 10^{-38}$ to $\pm 3.4 \times 10^{+38}$	
double (IEEE 754)	8	$\pm 2.23 \times 10^{-308}$ to $\pm 1.8 \times 10^{+308}$	

* 4 bytes on Windows64 systems, † one-complement

Arithmetic Types - Short Name

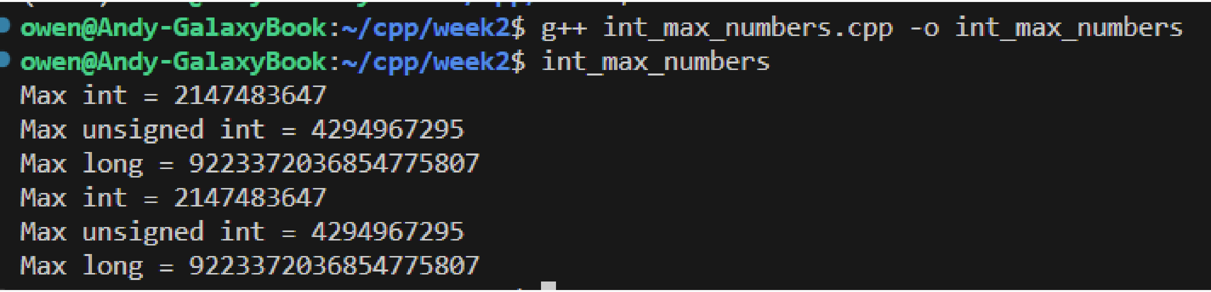
Signed Type	short name
signed char	/
signed short int	short
signed int	int
signed long int	long
signed long long int	long long

Unsigned Type	short name
unsigned char	/
unsigned short int	unsigned short
unsigned int	unsigned
unsigned long int	unsigned long
unsigned long long int	unsigned long long

Max Integers

```
#include <climits>
#include <cmath>
#include <iostream>
using namespace std;
int main(){
    // Max numbers from <climits>
    int n = INT_MAX;
    unsigned un = UINT_MAX;
    long l = LONG_MAX;
    cout << "Max int = " << n << endl;
    cout << "Max unsigned int = " << un << endl;
    cout << "Max long = " << l << endl;

    // Calculated limits;
    int n2 = (long)(pow(2, 31) - 1);
    unsigned un2 = (long)(pow(2, 32) - 1);
    long l2 = (long)(pow(2, 63) - 1);

    cout << "Max int = " << n2 << endl;
    cout << "Max unsigned int = " << un2 << endl;
    cout << "Max long = " << l2 << endl;
}
```

Max Integers



```
owen@Andy-GalaxyBook:~/cpp/week2$ g++ int_max_numbers.cpp -o int_max_numbers
owen@Andy-GalaxyBook:~/cpp/week2$ int_max_numbers
Max int = 2147483647
Max unsigned int = 4294967295
Max long = 9223372036854775807
Max int = 2147483647
Max unsigned int = 4294967295
Max long = 9223372036854775807
```

Arithmetic Types - Suffix and Prefix

Type	SUFFIX	example
int	/	2
unsigned int	u	3u
long int	l	8l
long unsigned	ul	2ul
long long int	ll	4ll
long long unsigned int	ull	7ull
float	f	3.0f
double		3.0

Representation	PREFIX	example
Binary C++14	0b	0b010101
Octal	0	0308
Hexadecimal	0x or 0X	0xFFA010

C++14 allows also *digit separators* for improving the readability `1'000'000`