Genome 540 Discussion

January 4th, 2024

Clifford Rostomily

Genome Sciences UNIVERSITY OF WASHINGTON

Introductions

- Who am I?
  - 2nd year Genome Sciences
  - Trapnell lab
  - Took this course last year
  - Gene regulation/development
  - Single-cell genomics
  - Zebrafish
  - I like to ski/trail run/mountain bike/fish

- Who are you?
  - Name?
  - Department?
  - What you hope to take away from this course?

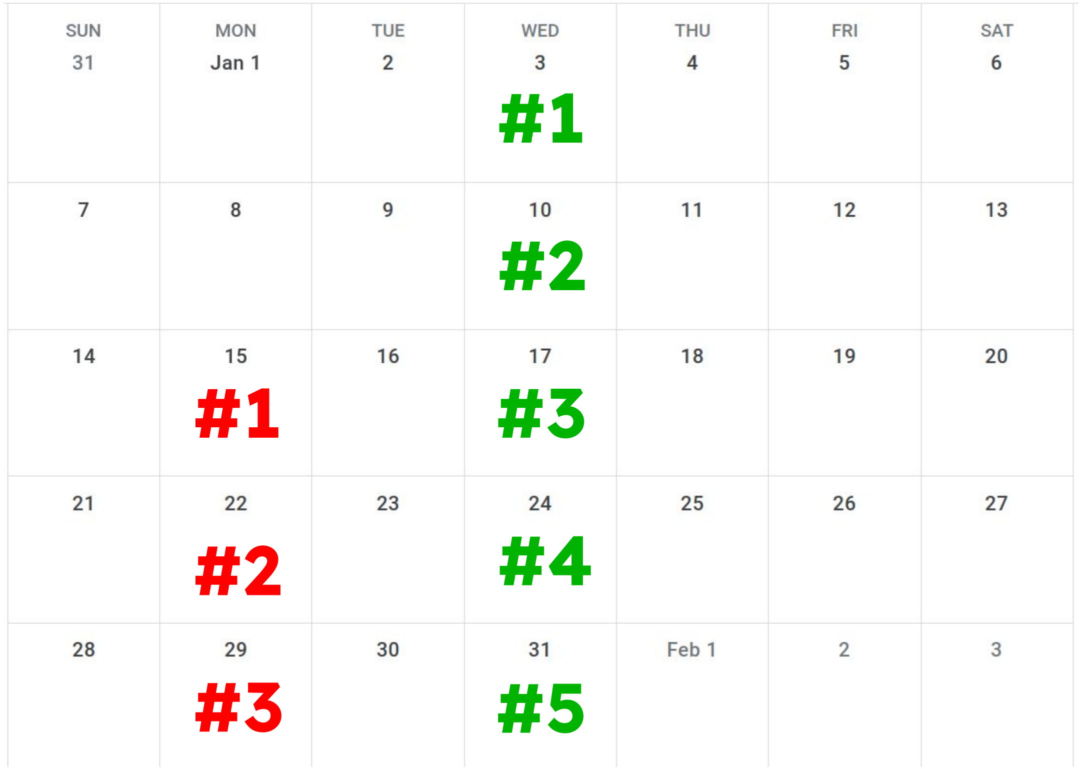
Agenda

- Homework advice
- Choosing a language
- C++ tips

Homework advice

Start Early!

- Start early
- Submit early
    - Ideally before the weekend it's due



SUN	MON	TUE	WED	THU	FRI	SAT
31	Jan 1	2	3 **#1**	4	5	6
7	8	9	10 **#2**	11	12	13
14	15 **#1**	16	17 **#3**	18	19	20
21	22 **#2**	23	24 **#4**	25	26	27
28	29 **#3**	30	31 **#5**	Feb 1	2	3

Using A.I.

- Do use it as a tool
  - Translating python to C++
  - Learning a new language
  - Debugging specific problems
    - What does this error mean?
  - Use it like a quicker version of stack overflow
- Don’t ask it to do your assignment
  - You won’t learn anything
  - If it’s wrong, debugging might be harder than doing the assignment

Write readable code - help me help you

- Use intuitive variable/function names

  x = 0 vs. number_of_friends = 0

- Comments
  - Big picture

    ```
    # Function to compute number of friends from comment quality
    ```

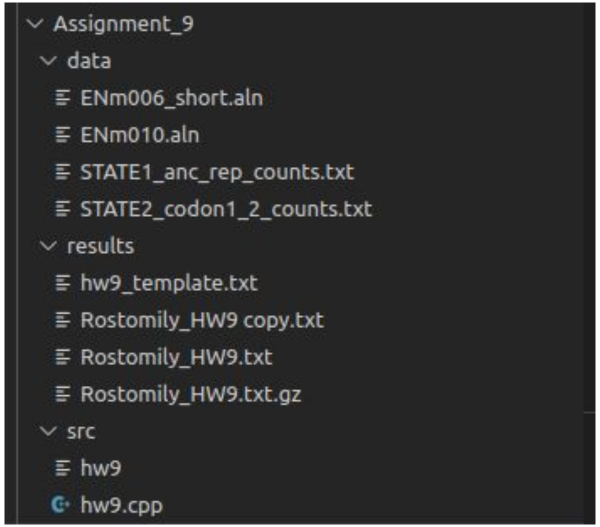
  - Confusing stuff

    ```
    # this makes me feel like I have friends
    num_friends = (a^-exp(24*b))/5 - (a^-exp(24*b))/5
    ```

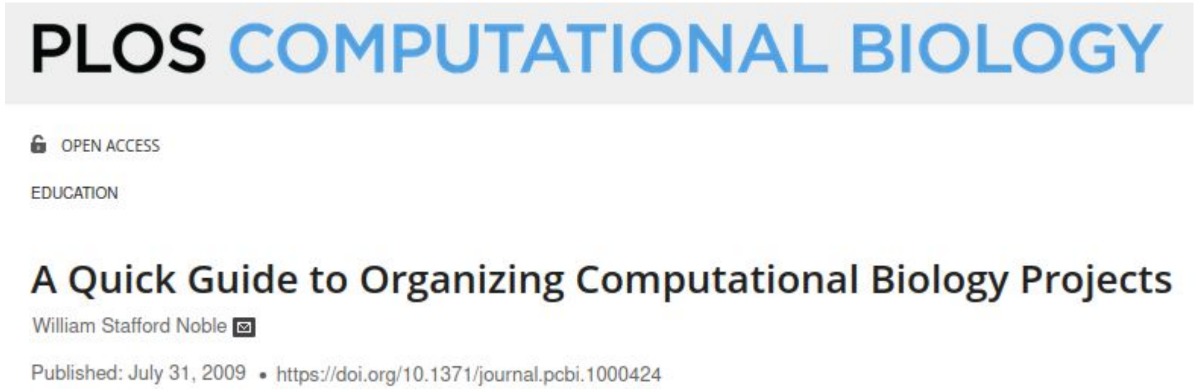
- Use lots of functions
- Don’t make code hard to read for a negligible speed up

Also keep your code organized

- Github for easy sharing etc.
- Keep a nice file structure for your assignments



```
Assignment_9
  data
    ENm006_short.aln
    ENm010.aln
    STATE1_anc_rep_counts.txt
    STATE2_codon1_2_counts.txt
  results
    hw9_template.txt
    Rostomily_HW9 copy.txt
    Rostomily_HW9.txt
    Rostomily_HW9.txt.gz
  src
    hw9
    hw9.cpp
```



PLOS COMPUTATIONAL BIOLOGY

OPEN ACCESS

EDUCATION

A Quick Guide to Organizing Computational Biology Projects

William Stafford Noble

Published: July 31, 2009 • https://doi.org/10.1371/journal.pcbi.1000424

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424

How to approach assignments

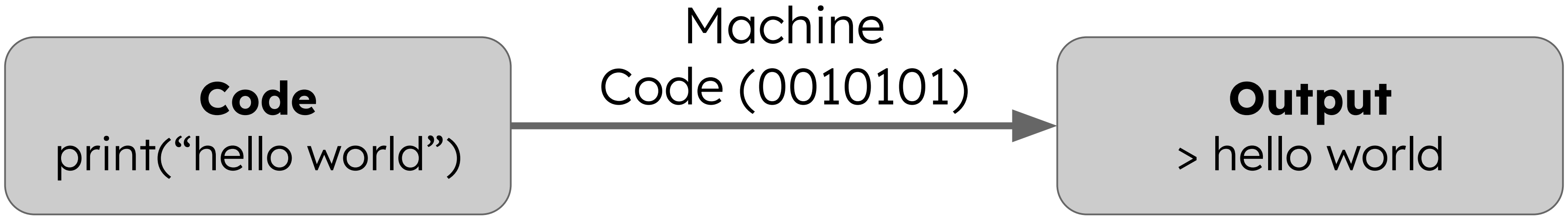
1. Understand the algorithm
2. Outline your code
    a. Write skeleton code
3. Fill it in
4. Evaluate if things are working with small tests
    a. E.g. Test that a fasta loads by creating a small fasta and printing it, or test code on a small substring you know the answer to.
    b. Try to include edge cases in your tests
5. Compare your results on the test data with diff

Choosing a language

Which language should I use?

- You are free to choose
- Most people use C++, C, or Python
- What’s the difference…

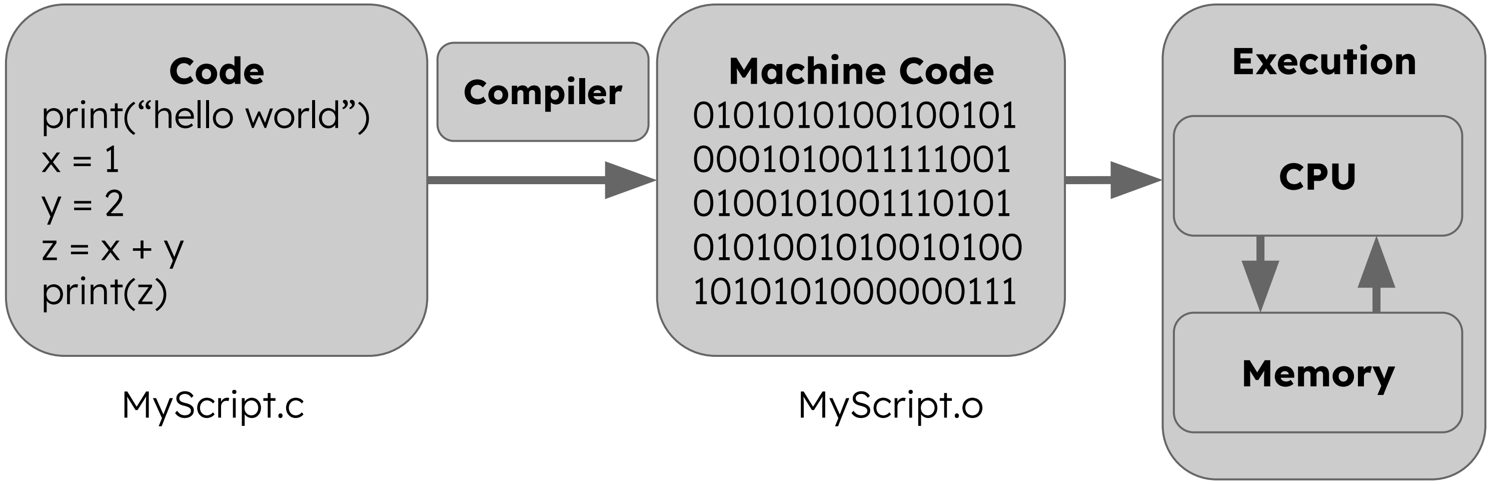
Compiled vs. Interpreted Languages



https://www.youtube.com/watch?v=_C5AHaS1mOA

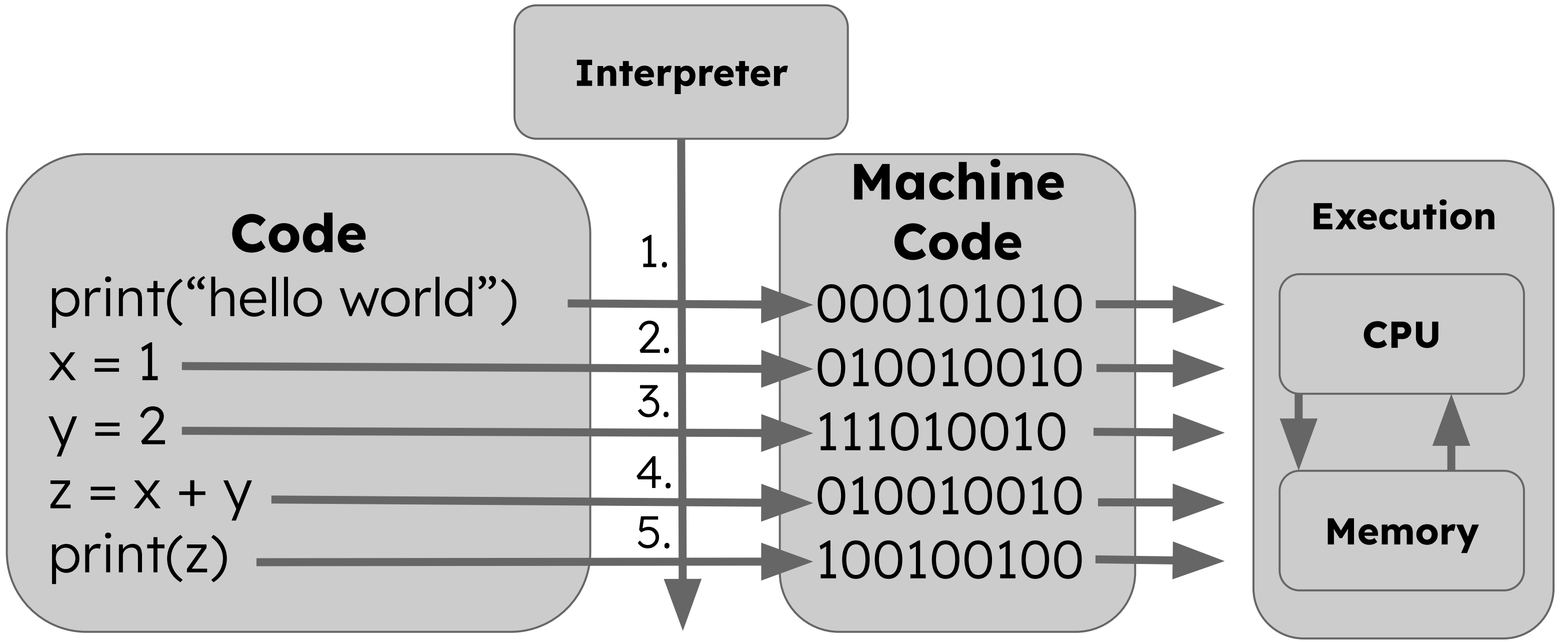
*** The following explanations are gross oversimplifications

Compiled vs. Interpreted Languages



- Compiler translates code into machine code
- Machine code can be run over and over (assuming correct OS/architecture)

Compiled vs. **Interpreted** Languages



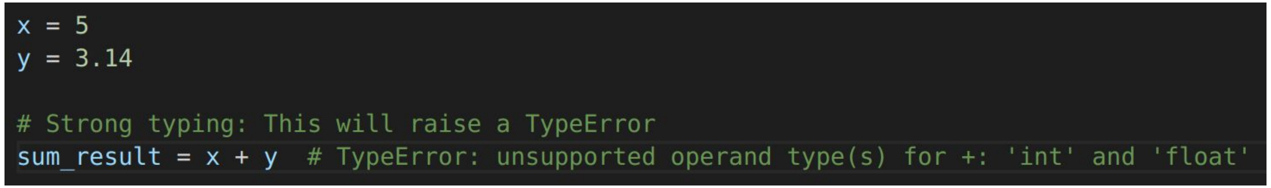
- Program executed line by line at runtime
- Need an interpreter to run program

Compiled vs. Interpreted Languages

	Compiled	Interpreted
Input	Entire program	A single statement
Intermediate	Machine code	None
Speed	Faster	Slower
Debugging	Errors reported after compiling and running	Errors reported at runtime

Static vs. Dynamic, Strong vs. Weak

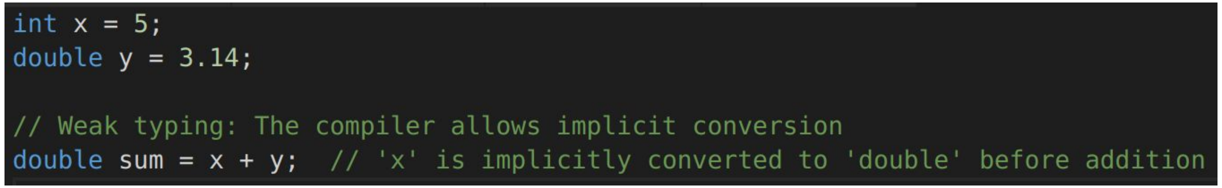
- Python is a dynamic strongly typed language
  - Don’t need to declare type: x = 5



```python
x = 5
y = 3.14

# Strong typing: This will raise a TypeError
sum_result = x + y  # TypeError: unsupported operand type(s) for +: 'int' and 'float'
```

- C++ is a static weakly typed language
  - Need to declare type: int x = 5;



```cpp
int x = 5;
double y = 3.14;

// Weak typing: The compiler allows implicit conversion
double sum = x + y;  // 'x' is implicitly converted to 'double' before addition
```

Which language should I use?

- Your choice
- C++ is my recommendation
- C++ will give the biggest improvement on the 1st assignment
- Python can work but you have to be careful with memory
- Python will be ~10x slower even if everything is perfect
- Python will be easier to learn/write/debug

C++ Tips

C++

Created by Bjarne Stroustrup in 1983.

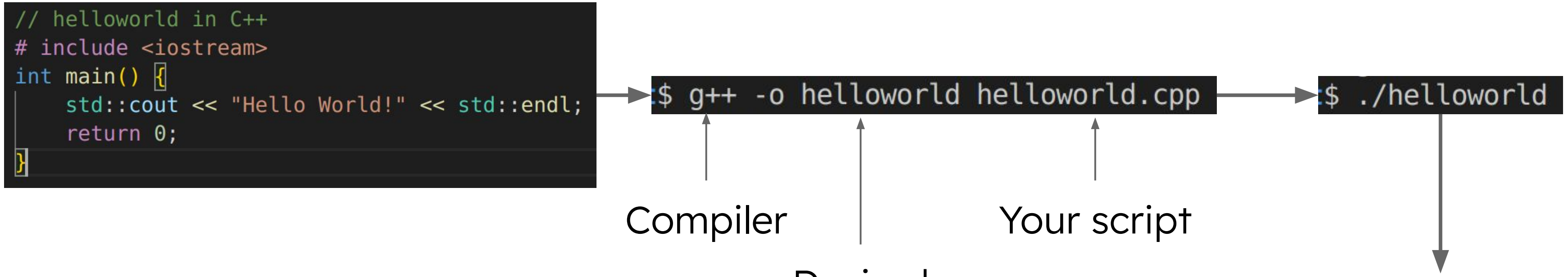
- Derived from C
- Supports classes and objects
- Standardized by the International Organization for Standardization (ISO)
- Used everywhere

“Hello World” in C++

helloworld.cpp

In terminal compile helloworld.cpp to an executable

Run the executable



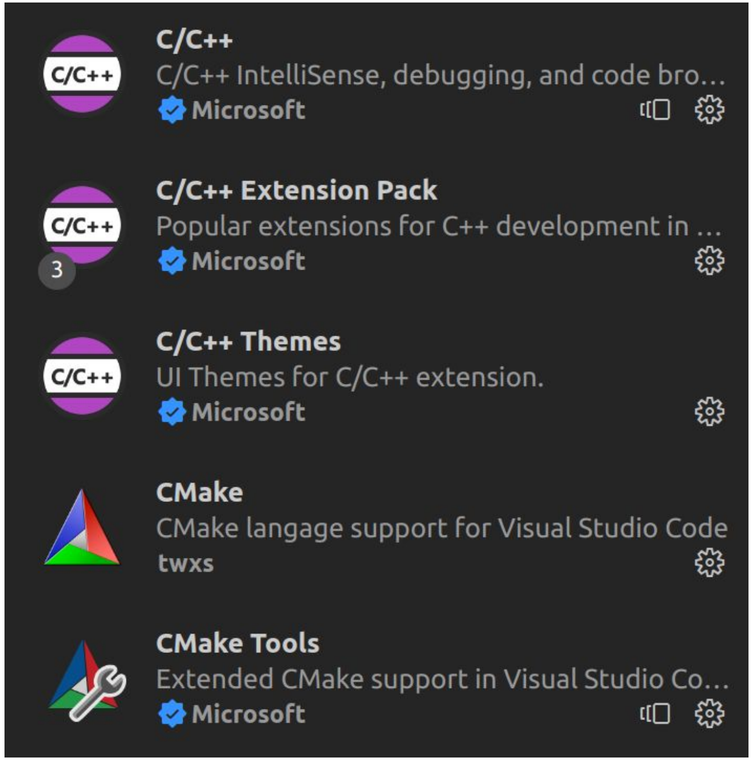
```cpp
// helloworld in C++
# include <iostream>
int main() {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

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

```
$ ./helloworld
```

Output:
Hello World!

Using an IDE



C/C++
C/C++ IntelliSense, debugging, and code bro…
Microsoft

C/C++ Extension Pack
Popular extensions for C++ development in …
Microsoft

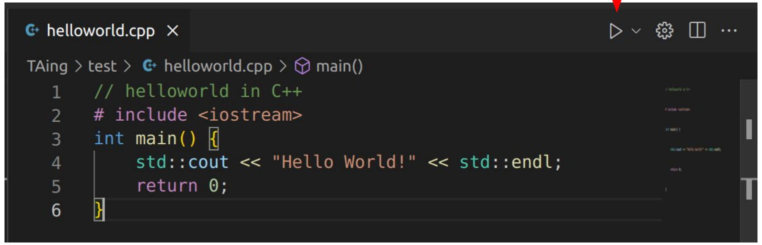
C/C++ Themes
UI Themes for C/C++ extension.
Microsoft

CMake
CMake langage support for Visual Studio Code
twxs

CMake Tools
Extended CMake support in Visual Studio Co…
Microsoft

- VSCode extensions can handle compilation and execution

All you have to do is hit play!



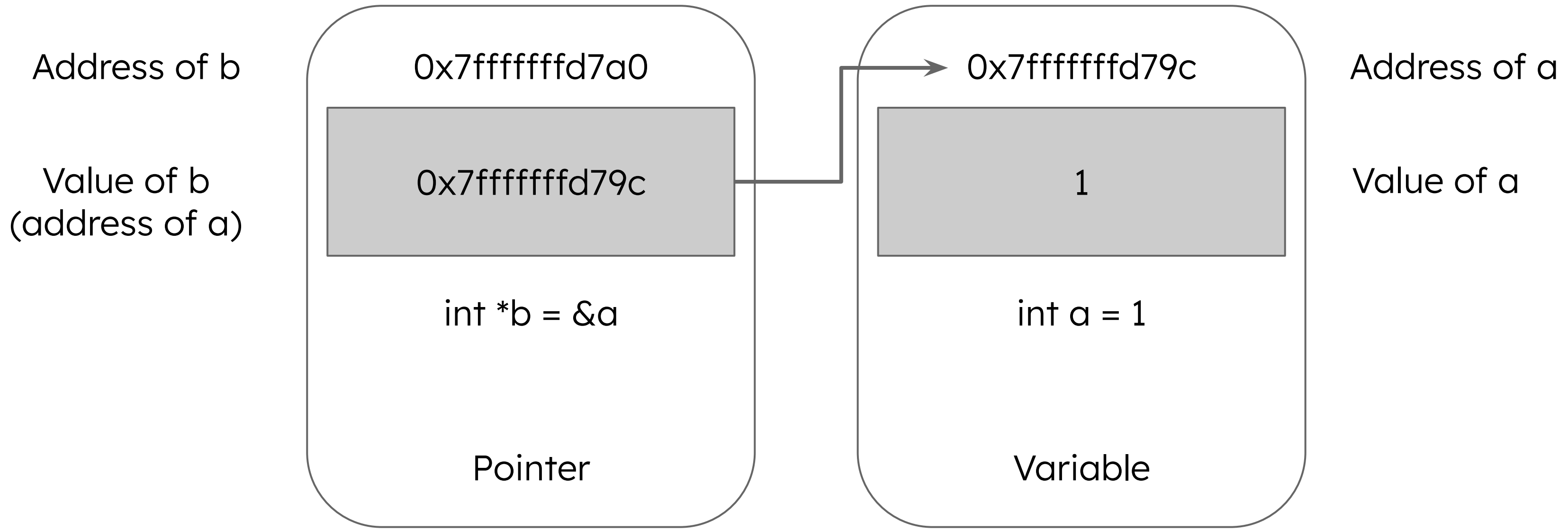
helloworld.cpp

TAing > test > helloworld.cpp > main()

```cpp
// helloworld in C++
# include <iostream>
int main() {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

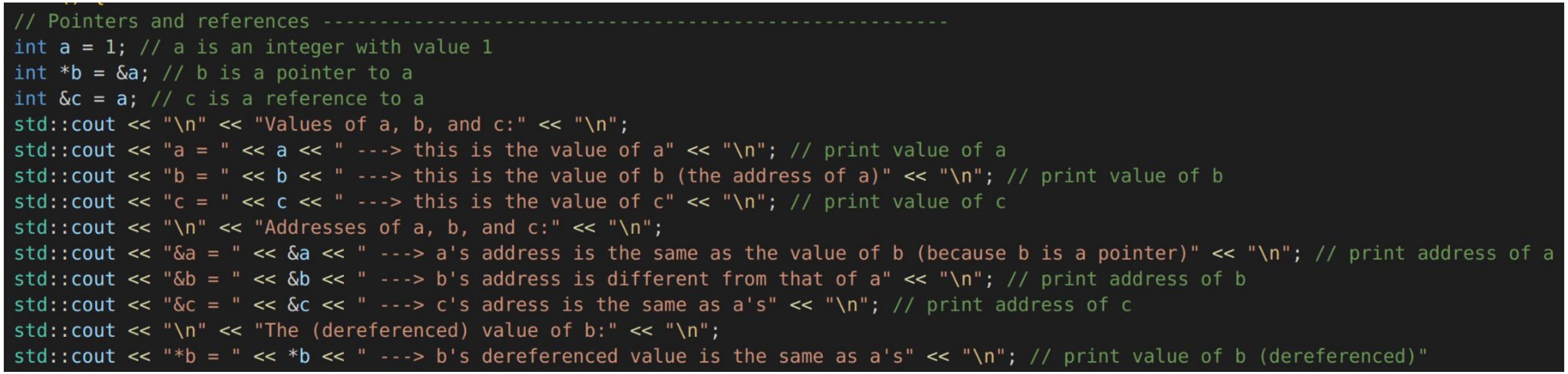
Pointers

- Pointers are memory addresses, which point to variables



Pointers

- Use **&** to reference an address
- Use ***** to dereference an address or declare a pointer



```
// Pointers and references ------------------------------------------------------
int a = 1; // a is an integer with value 1
int *b = &a; // b is a pointer to a
int &c = a; // c is a reference to a
std::cout << "\n" << "Values of a, b, and c:" << "\n";
std::cout << "a = " << a << " ---> this is the value of a" << "\n"; // print value of a
std::cout << "b = " << b << " ---> this is the value of b (the address of a)" << "\n"; // print value of b
std::cout << "c = " << c << " ---> this is the value of c" << "\n"; // print value of c
std::cout << "\n" << "Addresses of a, b, and c:" << "\n";
std::cout << "&a = " << &a << " ---> a's address is the same as the value of b (because b is a pointer)" << "\n"; // print address of a
std::cout << "&b = " << &b << " ---> b's address is different from that of a" << "\n"; // print address of b
std::cout << "&c = " << &c << " ---> c's adress is the same as a's" << "\n"; // print address of c
std::cout << "\n" << "The (dereferenced) value of b:" << "\n";
std::cout << "*b = " << *b << " ---> b's dereferenced value is the same as a's" << "\n"; // print value of b (dereferenced)"
```

Arrays vs. Vectors

- Vectors are like arrays, but they are dynamic
- Vectors can be resized, arrays cannot
- Adding new elements to a vector is slow and dynamic resizing may take up more memory than is needed
  - You should reserve the amount of memory you need when you declare a vector!!!

```
int my_array[3] = {1,2,3}; // d is an array of integers
std::vector<int> my_vector = {1,2,3}; // e is a vector of integers
my_vector.push_back(4); // add 4 to the end of my_vector
my_vector.pop_back(); // remove the last element of my_vector so that it is the same size as my_array
my_vector.reserve(100); // reserve space for 100 integers in my_vector
```

Pointers to arrays, and arrays of pointers

- Pointer to an array
  - int (*pntr_array)[5]; // a pointer to an array of 5 ints
- Array of pointers
  - int *pntr_array[5]; // an array of 5 pointers to integers
- Pointer to a vector
  - std::vector<int>*
- Vector of pointers
  - std::vector<int*>

Arrays are pointers to blocks of memory

- Arrays just point to the start of a memory block
- Array indices are just pointer arithmetic and dereferencing combined
  - a[12] is the same as *(a + 12)
  - &a[3] is the same as a + 3
- Large arrays should be dynamically allocated (on the heap)
- Make sure you delete them

```
const char *word = "hello";
word = hello
(word + 1) = ello
word[0] = h
*word = h
word[1] = e
*(word + 1) = e
```

```
int n = some_large_number;
double * d = new double[n];
```

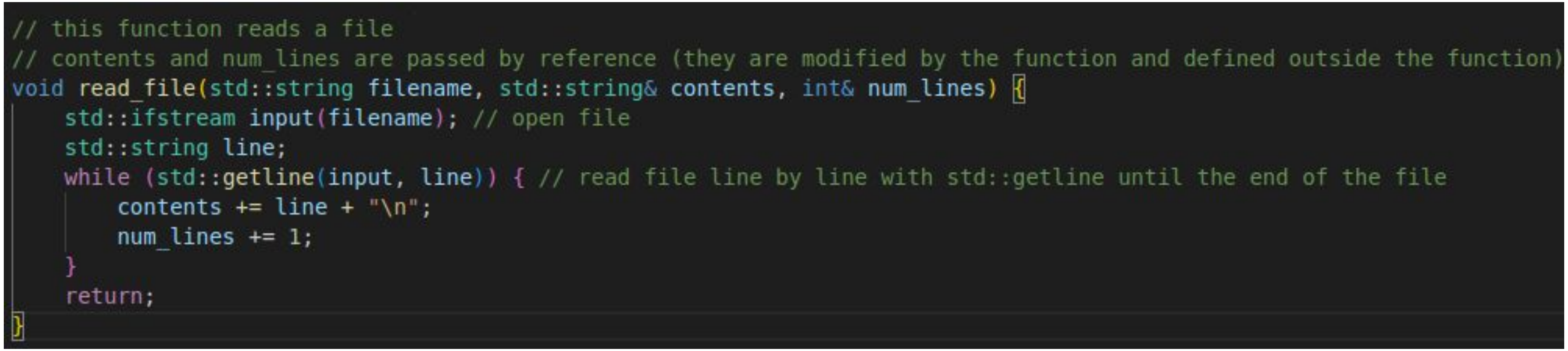
```
delete[] d;
```

Structs are a custom data type in C++

- Structs are like a very simple class
- Used to store data
- Can contain variables of any type (including pointers and other structs)

```
struct my_struct {
        int my_int;
        double my_double;
        std::string my_string;
        std::vector<int> my_vector;
        };
```

Reading Files



```
// this function reads a file
// contents and num_lines are passed by reference (they are modified by the function and defined outside the function)
void read_file(std::string filename, std::string& contents, int& num_lines) {
    std::ifstream input(filename); // open file
    std::string line;
    while (std::getline(input, line)) { // read file line by line with std::getline until the end of the file
        contents += line + "\n";
        num_lines += 1;
    }
    return;
}
```

Namespaces and libraries

- A namespace is a collection of libraries
- The standard (std) namespace is the most commonly used
  - Many other namespaces (e.g. boost, Qt, Eigen, OpenCV)
- You shouldn’t need anything other than the standard namespace for this course

Debugging

- Print intermediate to the terminal to see why something is breaking
  - Poor man’s debugger
  - std::cout << “value of x = “ << x << std::endl
- …or you can use a debugger
  - VSCode has a decent debugger for C++ and you can step through functions

Python Tips

Python tips

- Numpy
- Pandas
- Cython
- Faking pointers
  - Mutable types - https://realpython.com/pointers-in-python/
- Slack me for other questions

What do you want to learn about?

Topics for future discussion sections?

- Scalable and reproducible bioinformatics pipelines (Snakemake)
- General programming tips
- Specific languages: Python, C++, Unix tools
- Additional applications of HMMs
- Dynamic programming
- Machine learning
- Version Control/Github
- Jupyter Notebooks/Reproducibility