a C++ primer

ese2025

let’s start with Hello World! :)

```
#include <iostream>

int main()
{
	std::cout << “Hello, world!\n” <<  “that is what it said!” << std::endl;
	return 0;
}
```

this make look simple enough, and even familiar, but it’s important to note that this is a C++ program and not a C program!

when coding in C++, don’t “think in C”

- you should regard C++ as its own language
- while C remains the core language of C++, C++ has its own efficient way of doing things
- use C++ methods and containers whenever possible: your code will run faster, and other programmers will know that you know what you are doing!
- don’t be in a rush to use short-cuts like “using namespace std” (should definitely be avoided in library code)

<< and left-associativity

- consider the line:

  std::cout << “Hello, world!\n” << “that is what it said!” << std::endl;

- because << is left associative, this could be written as:

  (((std::cout << “Hello, world!\n”) << “that is what it said!”) << std::endl);

- << has a return value (**result**) EQUAL TO ITS LEFT OPERAND, and has the **side-effect** of writing its right operand to the stream represented by its left operand

std::cout and std::endl

- std denotes C++ standard library elements; in this case, we are using part of the C++ standard library--- **iostream**
- Notes to keep in mind:
    - std::cout has the type std::ostream
    - std::endl is a C++ *manipulator*
        - in this case, it says what to do with the output stream, std::cout, which is to end the line
- when C++ encounters the semi-colon, it knows to discard the result of the expression (when we use the “;” we are saying, in effect, that we are only interested in the side effect of the expression)

this was a rather in-depth look at Hello World … :)

- the above is not meant to bore you (or intimidate you!), but to deepen your understanding of what goes on “under the hood” in computer languages
- while work can be done with a superficial understanding of things, it is impossible to produce work with quality without a deeper understanding
- most of the time, we won’t be dwelling on such details, and using C++ in a much more intuitive way
- because, ultimately, C++ is meant to be an intuitive yet powerful language!

C++ strings --- *not* just a string literal!

```
#include <iostream>
#include <string>

int main()
{
	// request name
	std::cout << “please enter your name: ”;

	// read the name
	std::string name; // define the name
	std::cin >> name; // read the name

	// write the greeting
	std::cout << “Hello, ” << name << “! ” << “Your name has ” << name.size() << “ characters.” << std::endl;

	return 0;
}
```

the C++ vector container

offers the best of what C arrays and linked lists have to offer!

```
/*
 * fl.cpp
 *
  */
#include <iostream>
#include <vector>

using std::vector;
using std::cout;
using std::endl;
using std::cin;

/*
 * main code begins
 */
int main()
{
		/* store integers in a vector from standard input */
		vector<int> myarr; // our container
		int token; // our container content variable
		cout << "Please enter integers, followed by <CTRL><D>:" << endl;

		// invariant: fill is the integer to be stored
		while (cin >> token)
		{
				myarr.push_back(token);
		}
		cout << endl << endl;

		/* print integers from vector */
		cout << "Your integers are:" << endl;
		for (vector<int>::size_type j=0; j != myarr.size(); ++j)
		{
				cout << myarr[j] << endl;
		}
		cout << endl;

		/* find the largest element, manually */
		int max = myarr[0];
		int next;
		vector<int>::size_type i=1;
		while (i != myarr.size())
		{
				next = myarr[i];
				if (max < next)
				{
						max = next;
				}
				++i;
		}
		cout << endl << "... and your largest integer is: " << max << endl << endl;

		/* exit happily */
		return 0;
}
```