PIC 10B Discussion
Week 1

Professor: Michael Lindstrom
TA: Thomas Tu
Exercise

• Create a program that determines if a number is prime
  • Prime: not evenly divisible by any integer but 1 and itself
• Output: Each number, followed by “PRIME” or “NOT PRIME”
• Test this program using the following vector:
  • 2
  • 0
  • 1
  • 6
  • 2,147,483,647 (it’s prime)
  • 2,145,483,647 (it’s not prime)
Coding Style Rubric

• Using size_t to index containers
• Const correctness: isPrime(x) should be const and take const argument
• return 0; in main()
• Unsigned variable types when appropriate
• Header/CPP separation and declarations
• Commenting
Why do we even care about comments?

• Why include them? You already know what you’re doing....
  • Other people don’t
  • You won’t remember later
  • You might not even know now; helps you get your thoughts in order

• Syntax:
  • // This is a comment
  • /* This is a
     multiline comment */
  • Ctrl-K Ctrl-C, Ctrl-K Ctrl-U
Vectors

• `std::vector<T>, #include <vector>`
• An automatically resizing array that has a few other nice properties
• Takes care of memory management for you
  • Both good and bad
• Iterable, indexable, ideally behaves like an array
• Can get size()
• Front/back iterators: `begin()` and `end()`
• `push_back()`, `pop_back()`, `insert(iterator)`, `erase(iterator)`
Pointers

• Address in memory of an actual object
• Much faster to store and copy than the object
• Syntax: `int* ptr = &object;`
  • `&`: When on left, address-of operator, get the memory address of this object
• Method calls/attributes: `ptr->method()`, `ptr->attribute`
• Also `(*ptr).method()`
References

• “Another name” for a variable
• Define and initialize simultaneously
• Syntax: int& reference = original;
• Can use any syntax you would use for the original
  • ref.method(), ref.attribute
• Dereferencing:
  • (*ptr) becomes a reference to object
• Use references when you can, and pointers when you must
  • References are clearer and easier to understand
Spiral Rule

• Mnemonic device
  • Don’t use if it confuses you; it’s meant to make things easier
• Useful for decoding complicated pointer/reference declarations
• char* c
• char* c[10]
• const char* c
• char* const c
Classes

• Classes define types for new objects and methods they share
  • Objects sharing a class are called members of that class
• Functions attached to object types are member functions a.k.a methods
• Constructors: methods run after the object is first allocated
  • Initializer lists
• Other member functions: Inherited by all members of the class
Public/Private Member Functions

• Public – can be called from anywhere
• Private – can only be called from within the object
• Security: Assign the minimum permissions possible
  • Avoid public object variables – use getters and setters instead
  • If a method does not need to be public, make it private