fxns that require std::

- cin (uses >>)
- cout (uses <<)
- ifstream (uses >>)
- ofstream (uses <<)
- endl

Syntax for opening files:

```
std::ifstream infile("file_name");
        or
std::ifstream infile;
infile.open("file_name");
```

∴ same for ofstream

---

# includes

for strings → <string>

for vecs → <vector>

for in/out → <fstream>

for homemade classes → "class_name.h"

---

Pointers:

basic form → type *ptr_name;

* is a dereference operator

- used when defining the ptr to tell the computer that the var is a ptr
- used later to reference the data stored @ that ptrs location

& is a reference operator

- gets the memory address of a var

ex)

```
int num = 10;
int *num_ptr = &num
// num_ptr is now a ptr to the mem address of num, and *num_ptr == 10
// if I then write this line
*num_ptr = 30
// printing num will output 30
```

---

Dynamic Memory

allocated using 'new' keyword referenced w/ a ptr

ex) `int *array = new int[10];`

// array is now a pointer to the first position in the integer array

∴ to delete or (deallocate) the memory on the heap, use ~~delete~~ the keyword delete

→ for arrays, use delete[] in order to remove all elements

ex) `delete[] array`

You MUST ~~del~~ deallocate all memory!

---

When defining arrays in the heap that store pointers, place ref operators after type and before the brackets

ex) `int* ptr = new int*[10]`

---

custom stl sort fxns:

- the std::sort fxn take 2 args typically

→ std::sort(arr.begin(), arr.end())

- can take a third var if elements need to be sorted in a specific way
- this fxn Must return a bool & compare 2 args of the type of the elements in arr

→ std::sort(arr.begin(), arr.end(), comparison)

∴ DO NOT add extra parens w/ this fxn

Lab Section 7, 4:00 - 5:50 PM, Lally 102

TA: Ohad
Mentors: Zach, Anna, Kaelan, Niels
Terminal Compilation: g++ all_cpp_files.cpp -o name_of_out.out -g -Wall -Wextra
Example code:

- Header file ex → ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
#ifndef __line_h__
#define __line_h__
#include "point.h"
class Line {
public:
   Line(const Point &a_, const Point &b_) : a(a_),b(b_) {}
   const Point& get_a() const { return a; }
   const Point& get_b() const { return b; }
private:
   Point a,b;
};
// functions outside of the class
std::ostream& operator<< (std::ostream &ostr, const Line &l);
double gradient(const Line &ln);
bool steeper_gradient(const Line &m, const Line &n);

#endif
```

- Sort fxn ex → ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
bool compare(int a, int b) {
   return a > b;
}
int main() {
   std::vector<int> numbers = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5};
   std::sort(numbers.begin(), numbers.end(), compare);
   return 0;
}
```

- Pointers and dynamic mem ex → ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
#include <iostream>
#include <cstdlib>

int main() {
   int*** first = new int**;
   *first = new int*;
   **first = new int;
   ***first = 1;

   return 0;
}
```