Week 14 - Monday

CS222
Last time

- What did we talk about last time?
- Function pointers
- Lab 13
Questions?
Project 6
Computer Science is no more about computers than astronomy is about telescopes.

Attributed to Edsger Dijkstra
(But almost certainly not said by him)
Overview

- C++ is based on C and easier to use
  - You can declare variables anywhere
    - Not the case in the C89 standard (where all variables had to be declared right after a block starts), but our gcc is following the C99 standard
  - It has function overloading
  - Most people think the I/O is cleaner
- The big addition is OOP through classes
- It's an approximate superset of C that includes most C structures
Compiling C++

- **gcc** used to stand for the GNU C Compiler
  - When it became a suite of tools used for things other than C, they changed the name to the GNU Compiler Collection
- The compiler for C++ is called **g++** and is part of **gcc**, but it may need to be installed separately and doesn't come on all Linux installations by default
- C++ files have the extensions .cc, .cpp, .cxx, .c++, and .C
  - I prefer .cpp, but .cc is also common

```
g++ thing.cpp -o program
```
C++ is kind of an abomination

- C has too many ways to do things, but C++ is an order of magnitude worse
- Syntax is a big mess of overlapping, ambiguous ideas
  - Which only got worse in the C++11 standard, which we aren't talking about
- C++ tried to be reverse compatible with C, but not strictly true
- It tried to be object-oriented, but not strictly true
- The Standard Template Libraries are hideous compared to the Java Collection Framework
- At the time, it was the best choice available for OOP, and now we're stuck with it
Hello, World in C++

- It's not too different from C
- We need different headers for C++ I/O

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```
Output in C++

- Output uses the `cout` object (of type `ostream`)
- Instead of using formatting strings, `cout` uses the idea of a stream, where objects are placed into the stream separated by the extraction operator `<<`
- The `endl` object adds a newline to the stream
  - Of course, "\n" works too

```cpp
int x = 50;

cout << "There are " << x << " ways to leave your lover." << endl;
```
Basic output is easier
What about setting the width or precision?
You need to include the `iomanip` header
Put `setw(width)` in the stream to make the items take up the specified width
Put `setprecision(precision)` in the stream to show a certain number of decimal places
Put `fixed` to force it to pad with zeroes when there isn't enough precision

```cpp
double dollars = 2.0;
cout << "Give me $" << setw(10) << fixed << setprecision(2) << dollars << setw(0) << "!" << endl;

//printf equivalent
printf("Give me $%10.2f!\n", dollars);
```
Input in C++

- Input uses the `cin` object (of type `istream`)
- `cin` also uses the idea of a stream, where items are read from the stream and separated by the insertion operator `>>`
- It reads items using whitespace as the separator, just like `scanf()`

```cpp
int x = 0;
int y = 0;
int z = 0;

// Enter the x, y, and z values: 
cout << "Enter the x, y, and z values: ";
cin >> x >> y >> z;
```
Like Java, C++ has a class for holding strings, which makes life much easier

- It's called `string` (with a lower case 's')
- You must include `<string>` to use it
- Unlike `String` in Java, `string` is mutable
  - You can use array-style indexing to get and set individual characters

```cpp
string a = "Can I kick it?";
string b = "Yes, you can!";
string c = a + " " + b;
c[0] = 'D';
c[1] = 'i';
c[2] = 'd';
cout << c << endl;
//prints Did I kick it? Yes, you can!
```
Java uses packages to keep different classes with the same name straight.

C++ uses namespaces.

The standard library includes I/O (<iostream>), the string class (<string>), STL containers (<vector>, <list>, <deque>, and others).

If you use these in your program, put the following after your includes:

```cpp
using namespace std;
```

The alternative is to specify the namespace by putting the it followed by two colons before the class name:

```cpp
std::string name = "Ghostface Killah";
```
Functions in C++

- Regular C++ functions are very similar to functions in C.
- A big difference is that prototypes are no longer optional if you want to call the function before it's defined.
- Unlike C, function overloading allowed:

```cpp
int max(int a, int b)
{
    return a > b ? a : b;
}

int max(int a, int b, int c)
{
    return max(a, max(b, c));
}
```
Pass by reference

- In C, all functions are pass by value
  - If you want to change an argument, you have to pass a pointer to the value
- In C++, you can specify that a parameter is pass by reference
  - Changes to it are seen on the outside
  - You do this by putting an ampersand (&) before the variable name in the header

```c
void swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}
```
Pass by reference is a great tool
You don't have to pass nearly as many pointers
If you want to change a pointer, you can pass it by reference instead of passing a pointer to a pointer
It does allow more mistakes
  - Leave off the ampersand and your function does nothing
  - Change things you didn't intend to change
You cannot pass a literal by reference

```c
swap(3, 9); //doesn't compile
```
C++ also allows you to specify default values for function parameters. If you call a function and leave off those parameters, the default values will be used. Default parameters are only allowed for the rightmost grouping of parameters.

```cpp
void build(int width = 2, int height = 4) {
    cout << "We built this house with " << width << " by " << height << "s."
;
}

build();    //We built this house with 2 by 4s.
build(3);   //We built this house with 3 by 4s.
build(6, 8); //We built this house with 6 by 8s.
```
C++ example

- Let's write a complete C++ program that reads in:
  - A string
  - An integer
- Then, it prints out the string however many times the integer specified
When you want to dynamically allocate memory in C++, you use `new` (instead of `malloc()`)

- No cast needed
- It "feels" a lot like Java

```cpp
int* value = new int(); //make an int
int* array = new int[100]; //array of ints
Wombat* wombat = new Wombat(); //make a Wombat
Wombat* zoo = new Wombat[100]; //makes 100 Wombats with the default constructor
```
The delete keyword

- When you want to free dynamically allocated memory in C++, use `delete` (instead of `free()`)
  - If an array was allocated, you have to use `delete[]`

```cpp
int* value = new int(); // make an int
delete value;

Wombat* wombat = new Wombat();
delete wombat;

Wombat* zoo = new Wombat[100];
delete[] zoo; // array delete needed
```
You can compile C code with C++

- Weird things can happen, but we aren't going into those subtle issues

However, you now know and love the standard C libraries

You can use them in C++ too

You just have to include different header files

<table>
<thead>
<tr>
<th>C Library Header</th>
<th>C++ Equivalent</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctype.h</td>
<td>cctype</td>
<td>Character manipulation</td>
</tr>
<tr>
<td>limits.h</td>
<td>climits</td>
<td>Constants for integer limits</td>
</tr>
<tr>
<td>math.h</td>
<td>cmath</td>
<td>Math functions</td>
</tr>
<tr>
<td>stdio.h</td>
<td>cstdio</td>
<td>C I/O functions</td>
</tr>
<tr>
<td>stdlib.h</td>
<td>cstdlib</td>
<td>Random values, conversion, allocation</td>
</tr>
<tr>
<td>string.h</td>
<td>cstring</td>
<td>Null-terminated string manipulation</td>
</tr>
<tr>
<td>time.h</td>
<td>ctime</td>
<td>Time functions</td>
</tr>
</tbody>
</table>
A `struct` in C++ is actually just a class where all the members are public.

You can even put methods in a `struct` in C++.

Otherwise, it looks pretty similar.

You don't have to use the `struct` keyword when declaring `struct` variables.

Except in cases when it is needed for disambiguation.
Example

Here's a `TreeNode` struct in C++

```cpp
struct TreeNode
{
    int value;
    TreeNode* left;
    TreeNode* right;
};
```

Write a tree insertion with the following signature

```cpp
void insert(TreeNode* &root, int data);
```
Upcoming
Next time...

- OOP in C++
- C++ madness
- Templates in C++
Reminders

- Keep working on Project 6