CS221
C and Systems Programming

1

C++ whirlwind

Why use C++?

- Backward-compatible with C
- Able to call C library functions directly…
- …so it's good for Systems Programming
- Object oriented
- Very flexible and powerful (i.e. "dangerous like C")

I wanted to teach whole course in C++ but decided not to—too much to cover. Today and next are an intro. You are responsible to know this intro. You can investigate more on your own if you want.
PROFESSIONALLY, C HAS BEEN REPLACED BY C++.

C++ Basics

```
class MancalaBoard {
   // member variables and member functions…
};
```

Have constructors and destructors

- Called automatically, as in Java

```
class MancalaBoard {
   MancalaBoard(int numPlayers) { … }
   ~MancalaBoard() { … } // cannot have input arguments
};
```


Need closing semicolon
Any memory allocated in constructor should be freed in destructor. Can close files, etc.
SHOW EXAMPLE: allocating a Mancala board

C++ Basics

Class member variables and initialization:

```
class MancalaBoard {
  int whoPlaysNext;

  MancalaBoard() : whoPlaysNext(0) {
    // code
  }
};
```

Object Oriented Programming

Classes let us encapsulate important functionality and hide internal methods and variables

`public` and `private` syntax is different than in Java

```
class Pet {
private:
  char petName[64];
  int age;
public:
  Pet(const char* n) { strncpy(petName, n, 63); }
};
```


Encapsulation makes our SW safer to use by non-expert users

C++ Basics

Put declarations in a ***.h** file, code in a ***.cpp** file

- Compile via **g++**

Still have global methods and global variables

- `main()` is same as in C (not inside a class)

C++ memory management

Use `new` and `delete`, not `malloc()` and `free()`

`new` returns a pointer to a new object

Can call `new` on built-in or user-defined types

```
int main() {
  int* myArray = new int[100];
  float* myFloat = new float;
  Pet* petPointer = new Pet("Sparky");
  delete petPointer;
  delete myFloat;
  delete [] myArray; // syntax to delete an array
}
```


"new" better b/c easier to compute size, calls constructor, better error handling

Special array-delete syntax calls destructor on every object in array and frees all memory

Write and compile a sample program: simple.cpp

Note

Type of `"Am I a char string?"` is not `char*`

- It is `const char[20]`
- Cast to `const char*` if needed

Longer programs…

Write a class to help with Random Numbers

- Uniform and Exponential
- Exponential uses Uniform
- Main program generates and prints random numbers

Write a class to read "words" from a file

- Constructor input is file pathname
- `const char* word()` method returns next word, or NULL if end of file

see random.cpp if you need help
see readwords.cpp

Small in-class Code-along

Make a `class NumFile`

- Constructor takes name of file of `double` values, reads them all into an array
- Start with array size 10
- Make a resize() method to help the constructor: if the file has more entries than the array size, create a new array 2x larger than previous one, copy existing data into new array, and continue reading
  - Don't forget to `delete []` the old array
- Give `average()` and `valueAt(int index)` methods
- `main()` should get filename from command line, print average value, and print value at index 40

More C++ next time.
I will push a file numbers.txt with > 40 numbers
DO THE CODE-ALONG, SINCE THERE WILL BE AN IN-CLASS LAB NEXT TIME. GET HELP!!