CS 230
Programming Languages

11 / 17 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- Pointers – solutions to memory leakage problem
- C++ vs. Java

Pointers

- Problems
  - Memory leakage
    - When a pointer's value changes (that is, a different address is stored in it) and the address it was pointing to didn't have its memory deallocated
    - Garbage collection (frees deallocated memory for use)
      - C++ (done explicitly by programmer)
      - Java (done automagically)
    - Let me put a drawing on the board to represent the memory leakage problem.

Pointers

- Solutions to memory leakage problem
  - Reference counters
    - Store a reference counter for each memory cell whose value is the number of pointers currently pointing to it.
    - When pointers change value or are destroyed, the counter is decremented. It is also checked to see if it is zero. If it is, then the memory can be reclaimed.
      - Any drawbacks to this method?

Pointers

- Solutions to memory leakage problem
  - Reference counters (continued)
    - Any drawbacks to this method?
      - Space
      - Time
      - Circular references

Pointers

- Solutions to memory leakage problem (continued)
  - Garbage collection
    - Every heap cell needs a field as a flag to indicate whether or not it contains garbage.
    - When want to reclaim memory the garbage collector will
      - 1. Set all the flags as “containing garbage”
      - 2. Go through the program and determine all the memory that is being pointed to --- and changes the flags for those cells as “not containing garbage”
      - 3. All the ones still marked as “containing garbage” are reclaimed.
  - All preceding discussion assumed that the heap cells were all the same size.

C++ vs. Java

- In C++ but not in Java
  - can use pointers to memory locations
  - have a pointer to an object that has been returned to the heap (aka destroyed, deallocated)
    - dangling pointer problem
    - b/c explicit programmer deallocation of memory allowed
  - memory leaks
    - no garbage collection
  - array indices are not checked
    - unpredictable behaviour when accessing array elements beyond end of array

C++ vs. Java

- In C++ but not in Java
  - allow use of uninitialized variables (causes runtime error)
  - allowed to not return a value from a non-void function
  - can overload operators
    - e.g. for a class I can overload the * operator to perform some operation on objects of that class. more flexible/writable when used, than requiring/calling a function.
  - can have global functions that are not part of any class

C++ vs. Java

- In C++ but not in Java
  - has a preprocessor which can be used for (among other things) conditional compilation
    - e.g. code to do if compiling under linux vs. compiling under windows
  - has the Standard Template Library (STL) which contains things similar to what is found in Java Collections API
  - has templates whereas Java uses generics for same purpose

C++ vs. Java

- c++ has fewer compile time checks (which generate errors) than Java
- c++ has fewer run time checks
- c++ has no standard GUI library
- c++ has fewer standard library functions vs. the large Java API
- c++ is not as portable as Java, but there are guidelines out there to which one should try to adhere for more portable c++

C++ from C

- There are programming features leftover from C that can be used in c++.
  - c-style strings, c-style dynamic memory management, c-style printing
- we will look at the c-style way of doing things and compare and contrast to the c++ way.
- we need to know the C ways because they are part of valid c++ programs and we need to be able to read code not written by us that may use the C style stuff and sometimes we might want to call a C function in some C library ...

C++

- Some types
  - int, short, long
  - float, double, long double
  - any of the above numeric types can be prefaced with signed or unsigned
  - char (typically 8 bits)
  - wchar_t (wide = larger than a char)
  - bool (0 = false, 1 = true)
    - ints can be interpreted as bool (holdover from C which had no bool), 0=false, any other int is true.
  - void
  - enum (creates an integer type and named constants)

C++

```
// Hello.cpp

#include <iostream>
using namespace std;

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

C++

```
// #include logically inserts the code from the iostream file into this file
//   (Hello.cpp)
// iostream is a system header file so we use the < >
// if we wanted to include our own user-defined header file then we use " "

// int main() is the function declaration (all programs need one main)
//   can take 2 arguments if defined with them (int argc, char **argv)
// returns an int

// cout is like System.out.println
// << is an operator for writing
// >> is an operator for reading
// endl is a newline character
```

C++

```
// note: cout and endl are in the std namespace (like a package in java)
// without using namespace std, we could:
// std::cout << "Hello, world!" << std::endl;
```

C++

- The C++ specification does not specify the exact sizes of the types therefore they are implementation dependent. If you want to know the size of a type on a particular system, you can use the sizeof operator in C++ to get this info.
- Let's see a program that uses this sizeof operator and prints the sizes of the various types.
- You can write C++ programs in many different IDE's, including Eclipse, but you can also write code in a text editor and compile it (using GNU C++ compiler) at the command line like:
- *g++ mycode.cpp -o mycode.o*
- then to execute your program do:
- *./mycode.o*

C++

- Like Java,
  - C++ has for, while and do-while loops,
  - C++ has if's and else's
  - C++ has switch statements with cases
  - C++ has same arithmetic operators
  - C++ has same relational operators

C++

- >> and << when used on ints do bit shifting
- int x = 10;
- x = x << 2; // shift left by 2 bits multiplies by 4
- cout << x << endl; // will be 40

- >> is a shift right (divides by 2 to the right operand power)
- padding high bits with 0's?

C++

- More types
  - array
    - similar to Java
      - int nums[30]; // array of 30 ints, index starts at 0
      - int ages[] = {21, 18, 35, 52, 65};
      - can't put square brackets before the variable name when declaring (but in Java can)

C / C++

- C style memory allocation
  - malloc / calloc
  - free
- C++ style memory allocation
  - new
  - delete

C++

- More types
  - objects
    - a simple declaration in Java creates a reference to a class
      - Card a; // a is a Card reference
    - a simple declaration in C++ creates an object of that class by calling the default constructor
      - string a; // actually calls string's default constructor
    - string s1(“Hello”); // calls a different constructor
    - string s2 = string(“Bye”); // also calls a constructor
    - s1 = s2; // actually makes a COPY

C++

- More types
  - objects
    - string s1(“Hello”); // calls a different constructor
    - string s2 = string(“Bye”); // also calls a constructor
    - s1 = s2; // actually makes a COPY
    - s2 = “So long”; // doesn't affect s1

C++

- More types
  - objects/references
  - In Java we can have:

```
Card a;  // a reference to a Card (does NOT call Card's default constructor)
Card b = new Card(2, 2);
a = b; // a and b now both refer to the same object
b.setRank(5); // the ONE object has rank 5 and both a and b refer to it
```

  - This is the typical behavior in Java for classes