CS 230
Programming Languages

11 / 22 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- More C++

C++

- A couple of things from yesterday
  - Compile-time error when I forgot to put const on the toString in the definition, but it was there in the header file
  - Compiling problem when I didn't use -c
  - From the g++ manual:

    -c Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.

    Without -c it means to do the linking

    -o file Place output in file file. This applies to whatever sort of output is being produced, whether it be an executable file, an object file, an assembler file or preprocessed C code. If -o is not specified, the default is to put an executable file in a.out, the object file for source.suffix in source.o, its assembler file in source.s, a precompiled header file in source.suffix.gch, and all preprocessed C source on standard output.

C++

- classes in C++
  - destructors (class name preceded by ~, automatically called when object goes out of scope, usually put code in there for deallocating memory for data members if necessary)

C++

- classes in C++
  - If you don't create any constructors C++ automatically provides:
    - A default constructor (no parameters)
  - If you don’t create a copy constructor and/or a destructor then C++ provides
    - a copy constructor which provides shallow copy access

      if you need deep copy, write your own
    - a destructor
      - may need your own to release allocated memory

C++

- pointers in C++
  - I may have said that c1 below has the value NULL, but that is not true.

```
Card *c1;
Card *c2 = NULL;
//Apparently c1 is not NULL, but c2 is NULL.
//So, NULL is not the default value of a pointer.
cout << c1 << endl; // prints some non-zero number
cout << c2 << endl; // prints 0 because c2 is NULL
```

C++

- pointers in C++
  - a pointer is a variable that holds a memory address
    - the size of a pointer is the size of a memory address for your system
  - the memory address stored in a pointer is an address in the heap
  - a pointer can refer to different places during a program
  - a pointer can point to data structures that change in size or whose size is not known at compile-time
  - NULL == 0 is the value of a pointer that does not reference a memory location (doesn't point to a memory address)
  - a pointer can be explicitly dereferenced to refer to the data at which the pointer points (use the * operator for dereference)

C++

- pointers in C++
  - a pointer to a class can refer to members of the class by using the pointer operator ->
    - Card *c = new Card(3,3); // note use of new
    - c->setRank(5); // same as (*c).setRank(5);
  - a pointer can be subscripted if it is pointing to an array
  - can do pointer arithmetic (again useful if pointing to an array), e.g. adding 1 to a pointer adds the size of the type to which the pointer points

C++

- pointers in C++
  - from Budd's book:

```
int i = 7;
int j = 11;
int *p = &i; // & is address of, p now points to i
// p holds the address of i
// i holds the value 7

*p = *p + j; // p is being dereferenced, i now has value 18

p = &j;  // p now holds the address of j
```

C++

- const is used to make something constant (like Java final)
- e.g.
- const int x = 10; // like Java's final int x = 10;
- when used with pointers though be careful which is being declared constant (the pointer or the thing being pointed to)
- from Budd's book p.33:
  - int i = 7;
  - const int *p = &i; // pointer to a constant
  - int * const q = &i; // constant pointer
  - const int * const r = &i; // a const pointer to a const

C++

- void *
  - can be used to hold any type of pointer
  - think of Object references in Java

  - double d = 3.5;
  - double *dp = &d;
  - void *p = dp; // make p point to d (the thing dp points to)
  - double dp2;
  - dp2 = *((double *)p); // cast p to be a double pointer and deref
  - // can't simply do: dp2 = *p;

C++

- function pointers

```
double fdiv(int i, int j)
{
        return i / (double)  j;
}


double (*fptr) (int, int);

// fptr is a pointer to a function whose
// arguments are two ints and has a return type of double
// can assign to fptr the address of a function that has those
// properties

fptr = &fdiv;
```

C++

- function pointers
  - useful when a function can be generically written to take a pointer to a function which it then calls internally
  - However, newer code should use STL instead for generic functions
  - see qsort example in Budd p. 34

C++

- references used as parameters (for pass by reference)
  - use & in parameter
  - no need to do anything else

```
void passByRef(int &i) {
        i++;
}

int x = 7;
passByRef(x);
```

  - // x is now 8

C++

- Let's look at a program that uses malloc and calloc to allocate space on the heap in c-style and uses free to deallocate that memory. Let me run it as well so you can see that one defaults to all 0's.

C++

- pointers in C++
  - We have seen some uses of pointers not all were useful. Let's see one when pointers are useful/necessary:
  - a pointer can point to data structures that change in size or whose size is not known at compile-time
  - To create a linked list structure or any data structure that shrinks and grows during runtime, pointers are required.
  - Linked lists grow as you add nodes to it, so let's implement a linked list class that stores strings and allows the ability to add nodes