Andrew Plum

Cs 121

2/7/22

Week 5 Notes

* Overloading Operators
  + Multiplication
    - fraction operator\* (fraction);
    - f3 = f1 \* f2;
      * Same as f1.operator\*(f2);
    - fraction operator\* (int);
    - x = 7;
    - f3 = f2 \* x;
    - f3 = f1 \* f2;
      * Same as f1.operator\*(x + 5 / 2);
  + F3 = x \* f1;
    - F3 = x.operator\*(f1);
      * This doesn’t work
  + Friend of a class
    - friend fraction operator\* (int,fraction); // multiplication integer and fraction
* Destructor
  + Frees up parts you won’t be using anymore in memory
* Garbage
* Destructors
  + The system frees up the memory
  + YOU DO NOT CALL THE DESTRUCTOR
    - occurs when garbage goes out of scope
  + Garbage destructor
    - Garbage(){data = new double[1000];}
      * This is a constructor and data is a pointer to a double
    - ~garbage(){

Delete data;

“destructing\n”;

}

* Linked List Front Objects
  + Could be a queue
    - Treats the link list like a queue
  + Could be a stack
    - Treats the link list like a stack
  + Could be other object types
* Structure
  + Keyword is a struct
  + From c and c++ whereas classes are only c++
  + typedef struct node{ // means you don’t have to type struct every time a object is created

node \*next;

node\_entry entry; // node\_entry is the type we wish to enter

node(); // constructor

node(node\_entery item, node \*add\_on = NULL); // tell me the data you want to enter and tell me what to point to; = NULL is a default value – means that if you don’t enter a value for add\_on the function will still work, in this case the function can take 1 or 2 items

}

typedef int node\_entry;

#include”node.h”

typedef struct node node; // required to type struct every time a object is created unless this is done

int main(){

struct node n; // required to type struct every time a object is created

struct node n2(7,&n);

}