Andrew Plum

Cs 121

1/31/22

Week 4 Notes

* Stacks
  + First in last out
* Queues
  + Like stacks except they are first in first out
  + Circular arrays
    - Not really using a circular array but we think of it as circular
    - As soon as we get to the end of the array, we start reusing the start again.
      * Loop around to the start
    - Keep track of front and back to do a circular array
      * Need three variables for the circular
        + The front, back, and size of the circular array.
    - If you give the queue and an object you should decide whether or not the function pushes the object or constructs and pushes the object from raw data pass in for example.
    - Functions
      * errortype list::top(queuetype &d){

d = thelist[front];

return ec; // error code

}

* + - * errortype list::pop(queuetype &d){

front = front + 1;

thelist[front] = 0;

if(front == maxsize){

front = 0;

}

size--;

return size;

}

* + - * errortype list::push(queuetype &d){

back++;

back =back % maxsize;

thelist[back] = d;

size++;

return size;

}

* + - * if(front == maxsize){

front = 0;

}

* + - * front = front % maxsize;
* Pointers
  + Stores memory address of other variables
    - char C = ‘a’;
    - char\* P = &C;
    - \*P = ‘d’; // \* is a dereference operator, ‘a’ stored in C changes to ‘d’
  + Malloc
    - Memory allocation
    - Int \*q
    - Q = malloc(…) // points to a memory address returned
    - Q = calloc(…) // allocatios more memory
    - Equivalent of making an array when using a pointer
* {} redefining variables
  + Examples
    - Int x = 7

…

{

X = 8

…

cout << x << endl // prints 8

}

Cout << x << endl // prints 7

* Typedef can be used for linked lists to be used for linked lists so any variable can be used
  + Void pointers could also be used to point to any type you want
    - Because pointers are the same size because they just store memory addresses.
    - Void pointers could be used for arrays and linked lists
      * void \*p;

pet p1;

p = &x;

mylist.insert(p);

int x = 99;

p = &x;

mylist.insert(p);

* + - * + This works and can have multiple types of data
* Any times new, malloc, or alloc
  + You should use a destructor
* Typedef
  + When using curly braces, you put it in a new scope which means you can use the same name for
    - However, you can store a type def outside of scope in the curly braces using a variable