CSCI 2270 Lecture Notes

1/25/2019

* User Defined types for pointers
  + struct Person{

string name;

int age;

}

* + Person p1; //instance of person
  + Person \*ptr; // pointer to Person type
  + ptr = &p1; // sets pointer to address of the struct instance p1
  + Accessing members of a struct via pointer
    - (\*ptr).name = “Jacob”;
    - (\*ptr).age = 27;
    - OR the more common form:
    - ptr -> name = “Jacob”;
    - ptr -> age = 27;
* Dynamically Allocated Memory
  + so far we have been using automatic variables which are allocated on the stack at compile time
  + Dynamically allocated variables get allocated at runtime and use the heap (free-store)
  + Two important keywords
    - “new” and “delete”
  + Must use a pointer to manually allocate memory dynamically
    - int \*p1; //declare a pointer

p1 = new int; // “nameless variable”

* + - When you are finished, you must also deallocate manually. If you do not, this is called a memory leak

delete p1; // free the memory back to the heap

p1 = NULL;

* + - You are NOT deleting the pointer, you are only deallocating the memory the pointer refers to
* Dynamically Allocated Arrays
  + e.g. length 5 array

int n;

int \*ptr;

cin >> n;

ptr = new int[n];

for(int I = 0; i < n; i++)

cin >> ptr[i];

delete [] ptr; //deallocate entire array

* Array Doubling Example (double the length of the array)
  + create N-length dynamic array
  + read in N values to fill the array
  + “double” the array length keeping the first N elements in place
* Approach to above example
  + create a new array of 2N length
  + copy over values from first array to the second array
  + clean up
    - free up memory of “a”
    - make “a” point to the same address as temp
    - temp = NULL;