Matthew Lonis

Jeff Whitmer

CSCI-A 290

September 26th, 2016

MINI 3

**Pointers in C**

URLs

1. <http://www.tutorialspoint.com/cprogramming/c_pointers.htm>
2. <http://www.cprogramming.com/tutorial/c/lesson6.html>
3. <http://www.programiz.com/c-programming/c-pointers>

Summaries of URLs

URL 1

This URL starts off by mentioning that pointers make some tasks easier. On the other hand, some tasks cannot be done without pointers in C such as dynamic memory allocation. A pointer is a variable whose value is the address of another variable. Variables have to be declared with a c data type and then a name with a \* as a prefix. The \* denotes the declaration of a pointer. The URL continued with an example on declaring a pointer, assigning a value to a pointer (assigning an address), and printing the value that the pointer points to. The URL ends with stating that it’s always a good idea to assign NULL to a pointer when we don’t know what we will be doing with it.

URL 2

This URL covered a lot of the same things the first URL covered. It talked about what pointers are and why/when we should use them. This URL elaborated more on why using pointers saves memory when you have a lot of data to pass to a function. This URL also mentioned how you can ask for more data from the system but didn’t go into too much detail about it. I’d love to know more about that topic. The URL discussed the correct syntax for a pointer (<data type> \*pointer\_name). The URL finished up by showing an example much like the example shown in URL1.

URL 3

This URL starts off by discussing how & is a reference operator to talk about addresses. Continues to show us how to properly declare a pointer variable. The only really new information was found in the common mistakes section at the end of this URL. It showed how pointers can only be assigned an address.

For example:

/\* Begin code \*/

int c, \*pc;

pc = c; /\*pc is address whereas, c is not an address. \*/

\*pc=&c; /\* &c is address whereas, \*pc is not an address. \*/

So this was very useful to prevent future mistakes.

Summary

I think the most important things in all the resources are to declare a pointer variable with the \* symbol and assign the addresses using & symbol.

JIT 3

What is Dynamic Memory Allocation?

**Structures in C**

URLs

1. <http://www.tutorialspoint.com/cprogramming/c_structures.htm>
2. <http://www.cprogramming.com/tutorial/c/lesson7.html>

Summaries of URLs

URL 1

A structure is a type of user defined data type available in C. Structure are used to represent a record of things such as attributes of an employee like first name, last name, position, etc. To define a struct you use the struct tag and then flesh out the members and variables inside the { } brackets. To access members in a struct you call Struct\_name.Member\_name sort of like calling a method in a different java class. Structures can be arguments in function or they can be pointers.

URL 2

This URL was much like URL1 however it gave a lot more examples. This URL showed me how to correctly define a struct and its members. The example used in the URL at the bottom showed how a structure can be used in a pointer and introduced me to the -> sign used in printf statements to get the value from a pointer that contains a struct. Other than that this URL covered relatively nothing new.

Summary

The most important aspects from the URLs was learning that structs are used when you need a record of things or to represent things like attributes of a person.

JIT 3

How are structures used as arguments in functions?

**Linked Lists in C**

URLs

1. <http://www.cprogramming.com/tutorial/c/lesson15.html>
2. <https://www.tutorialspoint.com/data_structures_algorithms/linked_list_program_in_c.htm>

Summaries of URLs

URL 1

This URL starts put by stating that linked lists are a way to store data with structures so that the programmer can automatically create a new place to store data whenever necessary. The URL gave an example of who to create a linked list using a sruct called a node where in the node a member of the node is a struct pointing to another node. This allows the linked list to be modifiable, expandable, and variable sized. This URL also showed how to use malloc to call more memory as to not run out.

URL 2

This URL states that a linked-list is a sequence of data structures which are connected together via links within the list. Linked lists are apparently the second most used data structure after arrays. The URL continues with a massive example in order to show and explain how linked list works and it covers most of what URL 1 covers. This example really demonstrated how linked lists are lists that can be modified very easily.

Summary

Linked lists are lists that allow a programmer to modify and expand current lists in a dynamic way.

JIT 3

Are linked lists our first glimpse at Dynamic Memory Allocation?