C++ Linked Lists and Arrays

Michael Brodskiy
Professor: S. Shazli

March 22, 2023

- If `a` is a pointer to a `struct`, then to access the `struct`'s members, we use the `->` operator, as in `a -> x`
- A linked list is a series of connected nodes
  - Each node contains at least:
    - A piece of data (any type)
    - Pointer to the next node in the list
    - The head is the pointer to the first node
- Adding a node
  - There are four steps to add a node to a linked list:
    1. Allocate memory for the new node
    2. Determine the insertion point (you need to know only the new node's predecessor or previous node (`prevNode`))
    3. Point the new node to its successor
    4. Point the predecessor (`prevNode`) to the new node
  - Find the node you want to insert after
  - First, point the new node (`newNode`) to its successor
  - Second, point the predecessor (`prevNode`) to the new node
  - Deleting an element
    - To delete the first element, change the link in the header
    - To delete some other element, change the link in its predecessor
  - Notice that both the insert and delete operations on a linked list must search the list for either the proper insertion point of to locate the node corresponding to the logical data value that is to be deleted
  - For Lab 7:
    - In this lab we will practice the use of gdb as a tool to debug your programs through step-by-step execution and memory inspection. We will also continue to work with linked lists as an alternative data structure to store sequence of elements, where insertions and deletions have a constant cost. Finally, we will use gdb to explore the execution of a main program using linked lists.
    - A debugger is a tool used to inspect the memory of a program in a controlled execution environment, with the common objective of identifying the presence of bugs in the program. The gdb (GNU debugger) tool is shipped together with gcc in most Linux distributions, and can be accessed both on the DE1-SoC and the COE machines. Consider the following program, which assigns predefined values to a variable of type `Person`, and prints them through an invocation to function `PrintPerson`

* In order to debug this program with gdb, you need to make sure that it was compiled with additional debug information, required by gdb to associate instructions in the binary file with lines of your source code. This is done by adding flag `-g` to your `g++` command line, as such:

```
>_ g++ person.cpp -o person -g
```