Lab 4

Table of Contents

Review + Dynamic Memory Allocation

Dynamic Memory Allocation in C

- void *malloc(size_t size);
  - Allocates size bytes for your object.
- If Success
  - returns a pointer that matches the pointer type
- Else
  - returns a NULL pointer

Dynamic Memory Allocation in C++

- pointer-variable = **new** data-type;
  - No Parameters, C++ decides the size of the allocation for you
- Examples
  - int * a = new int(20) // create an integer pointer with value of 20 in Heap
  - double * b = new double[20] // create a double array pointer with 20 elements in Heap
- In This Lab
  - we have a Roster Pointer, the holds a collection of student, that we need to do memory allocation on

Why we need Dynamic Allocation?

- Why Can’t we just use Stack?
  a. flexibility and efficiency
     - allows on-demand and just-enough space allocation and deallocation
     - no storage waste (UNLESS memory leak)
  b. Remember Lab 3
     - we define CAPACITY to be 30
     - we create our Roster with an array of CAPACITY students on stack
     - what happens if we are growing our roster?

new vs. malloc(), delete vs. free

- This only applies if the class doesn’t overload new or delete.
- New:
  - Call Constructor
  - malloc()
- Delete:
  - Call Destructor
  - free()

Lab 4 Description

Project Structure

- Build upon Lab3
- Functions to be added
  - Destructor
- Functions to be modified (or not)
  - Constructor
  - insert
  - erase
  - (Add more functions if you would like to)
- Test Plan needs to be updated
  - What should change

Insert

- Involves pointers ➔ dynamic allocation
- How:
    - Create a new temp pointer w/ more allocated space
    - Copy old ➔ temp
    - delete[] old
    - Let old = temp

rostermain.cxx (rostertest)

- Main Function For Test
    - takes user input from input file
        - `rosterTest < input_file > outout_file` takes care of it
        - Or you can use ifstream
    - reads line by line
        - already done in sample main
    - format for line input
        - <command key> <params (depend on key)>
        - <command key> <params> format
            - <A> <ID> <first> <last> – insertion of a student with the input ID, first, last
                - ex: A 1234567 John Doe
            - <X> <ID> – deletion of student with ID
                - ex: X 7654321
            - “L” – list roster of students
                - ex: L

rostermain.cxx testing tips (rostertest)

- Edge Cases
  - insertion
    - under what cases we can not insert a student into the roster?
  - deletion
    - under what cases we can not delete a student into the roster?
  - list
    - under what cases we can not list the roster?

Test Plan

- Talk about the expected difference between the 3 implementations
- Which one is better?
    - in term of efficiency, complexity, and flexibility
    - Hint: try to find a way to record the time it takes for the program to run (sounds familar) and compared to lab 3 and lab 4

Deliverables

- All .cpp files
- All .h files
- Makefile
- output file
- test plan

Demo

- Test Plan
- Code compilation/run

Other Tips

- Test code frequently
- Test your code comprehensively
    - Think about what needs to be tested
    - Points will be deducted if you missed critical test cases

Don’t Forget

- Submit the code before next week’s deadline
- File with guide to implement and hints are in Camino
  - Make sure your code can run on school Linux server