Computer Science 2510 - Lab 6

Readings

- Class Notes
- Textbook: Chapter 17

Objectives

- To become familiar with vectors, dynamically-allocated arrays and pointers.

Notes

- Most of the exercises in this lab were taken from the "Drill" section of Chapter 17 of the textbook (Bjarne Stroustrup, *Programming - Principles and Practice Using C++*, Second edition, Addison-Wesley, 2014, ISBN 978-0-321-99278-9.)

Lab Exercises

1. **Chapter 17 Drills, Part 1**

   Part 1 exercises/builds your understanding of dynamically-allocated arrays and contrasts arrays with `vectors`:

   1.1. Allocate an array of ten `int`s on the free store using `new`.

   1.2. Print the values of the ten `int`s to `cout`.

   1.3. Deallocate the array (using `delete[]`).

   1.4. Write a function `print_array10(ostream& os, int* a)` that prints out the values of `a` (assumed to have ten elements) to `os`.

   1.5. Allocate an array of ten `int`s on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.

   1.6. Allocate an array of 11 `int`s on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.

   1.7. Write a function `print_array(ostream& os, int* a, int n)` that prints out the values of `a` (assumed to have `n` elements) to `os`.

   1.8. Allocate an array of 20 ints on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.

   1.9. Did you remember to delete the arrays? (If not, do it.)

1.10. Do 5, 6, and 8 using a `vector` instead of an array and a `print_vector()` instead of `print_array()`.

2. Chapter 17 Drills, Part 2

Part 2 focuses on pointers and their relation to arrays. Using `print_array()` from the last drill:

2.1. Allocate an `int`, initialize it to 7, and assign its address to a variable `p1`.

2.2. Print out the value of `p1` and of the `int` it points to.

2.3. Allocate an array of seven `int`s; initialize it to 1, 2, 4, 8, etc.; and assign its address to a variable `p2`.

2.4. Print out the value of `p2` and of the array it points to.

2.5. Declare an `int*` called `p3` and initialize it with `p2`.

2.6. Assign `p1` to `p2`.

2.7. Assign `p3` to `p2`.

2.8. Print out the values of `p1` and `p2` and of what they point to.

2.9. Deallocate all the memory you allocated from the free store.

2.10. Allocate an array of ten `int`s; initialize it to 1, 2, 4, 8, etc.; and assign its address to a variable `p1`.

2.11. Allocate an array of ten `int`s, and assign its address to a variable `p2`.

2.12. Copy the values from the array pointed to by `p1` into the array pointed to by `p2`.

2.13. Repeat 2.10-2.12 using a `vector` rather than an array.

Author: Department of Computer Science, MUN (BE220531)