C++ Exercises

CPSC 457 Week 1

Jesse Francis

January 23, 2020

Administrative Stuff

Plan For Today

- Pointers
- Buffers
- Exercises

Learning Resources

- Available from the library
  - The C++ Programming Language, 4th Edition
    - More of a reference. I would recommend skimming Part II.
  - Programming: Principles and Practice Using C++, 2nd Edition
    - A tutorial, lots of exercises.
    - Recommended Reading: Part I and Part III
  - C++ Primer, 5th Edition
    - An alternative to *Programming: Principles and Practice Using C++*
    - Recommended Reading: Part I and Part II
- Others
  - C++ Crash Course
    - Not available from the library, so I would recommend one of the above over this.
  - cppreference.com
    - Good reference for the C++ language.
    - Helpful for finding which Standard Library function to use.

Pointers

Pointers

Pointers hold the memory addresses. We can get the address of a variable using the 'address of' operator: `&`

Suppose we have:

```
int i = 1234;
```

Then, `&i` is the address of `i` in memory.

Pointers

Pointers hold the memory addresses. We can get the address of a variable using the 'address of' operator: `&`

Suppose we have:

```
int i = 1234;
```

Then, `&i` is the address of `i` in memory.

We can store it into a variable, using

```
int* p = &i;
```

Pointers

Pointers hold the memory addresses. We can get the address of a variable using the 'address of' operator: `&`

Suppose we have:

```
int i = 1234;
```

Then, `&i` is the address of `i` in memory.

We can store it into a variable, using

```
int* p = &i;
```

Finally, we can get the value of the variable by dereferencing the address. We do this by applying the 'dereference' operator: `*`

```
int n = *p;
```

Pointers (Cont'd)

Full program example:

```
1  #include <iostream>
2  int main() {
3      int i = 1234;
4      int* p = &i;
5      std::cout << "Address of 'i':\t" << p << "\n"
6                << "Value of 'i':\t" << *p << std::endl;
7  }
```

Outputs something similar to:

```
Address of 'i':  0x7ffe31acbb5c
Value of 'i':    1234
```

nullptr

A pointer of any type that does not point to an object can be assigned `nullptr`.

```
int *p = nullptr;
```

Buffers

Buffers

Buffers are typically arrays or other containers that temporarily store data between expensive/time consuming operations.

Buffers

Buffers are typically arrays or other containers that temporarily store data between expensive/time consuming operations.

For example, when printing to standard out, `printf` and `cout` store the output in a buffer and then send it after so much has accumulated or certain characters have been buffered, unless overridden.

For example, output is likely buffered in the below example and nothing is sent to `stdout` until the `printf` call after the `for` loop.

```
1  for (int i = 0; i < 100; i++) {
2    printf("%d ", i);
3  }
4  printf("\n");
```

Buffers Example

```cpp
#include <cstdio>

int main() {
  for(int i{0}; i < 100000; i++) {
    printf("\n");
  }
}
```

When run with

```
time ./bufferEx
```

we get something similar to

```
real    0m1.737s
user    0m0.063s
sys     0m0.217s
```

Buffers Example (Cont'd)

What do we get when we run the previous example using the following command? Why?

```
time ./bufferEx > ./output.txt
```

Exercises

Exercises

1. Compile `countLines.cpp` from the assignment page and run it on `romeo-and-juliet.txt` using
   `strace -c ./countLines ./romeo-and-juliet.txt`
2. Run
   `strace -c wc -l ./romeo-and-juliet.txt`
   and compare it to the output of Exercise 1.
3. Write a program to find and print the average of two integers from the command line.
   i.e., `./ex3 4 5` should print `4.5`.
4. Write a program to find and print the average of an arbitrary number (within reason) of integers from the command line.
   i.e., `./ex4 1 2 3 4 5` should print `3`.
5. Write a function that swaps two arrays using pointers and references.