Discussion 8!

More on pointers, debugging strategies, p3 tips

Project 3 Tips!

Tips

- Error checking: P1 and P2 didn't require strict error checking. P3 is not the same!!!
  - Think about all the possible ways that a user could provide faulty input. Be thorough!
- Start early!!! This project is significantly more workload than the last two
  - The spec comes out tomorrow, you have two weeks to do the whole project
- Read the spec carefully - there's lots of specifications you need to meet
- Review lectures and discussions
  - We did example error checking, example makefile, example dynamic allocation of 2D arrays
- Come to office hours to check design

More on Pointers

-> operator

Dereference and dot in one step!

Lines 19/20 do the same thing

```
6   class TestClass {
7       public:
8           int testInt;
9
10          TestClass() {
11              testInt = 5;
12          }
13  };
14
15
16  int main() {
17
18    TestClass *testptr = new TestClass();
19    cout << testptr->testInt << endl;
20    cout << (*testptr).testInt << endl;
21
22    return 0;
23  }
```

-> Operator

*ptr.hello() doesn’t do what you think it does!

- Does *(ptr.hello()) instead of (*ptr).hello()

Arrow operator helps make this cleaner

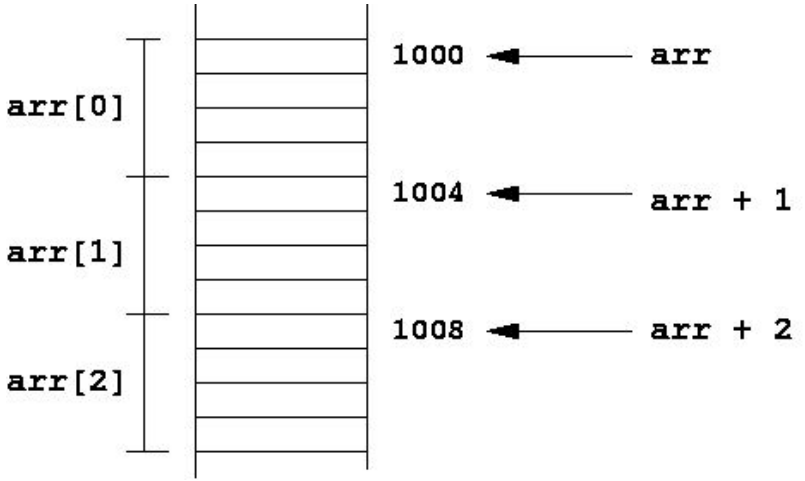
```
// From this...
(*ptr).hello();

// To this...
ptr->hello();
```

Pointer Arithmetic

Arrays and pointers are almost identical

- “arr[0]” is equivalent to “*(arr + 0)”
- Allows for a new way to traverse an array



Pointer Arithmetic

This is known as “traversal by pointer”

- Usually used when iterating over C-strings
- Also mimics using iterators (out of class scope)

```
int main(int argc, char *argv[]) {
    int arr[5] = {1, 2, 3, 4, 5};

    // Array decay!
    // Equivalent to "int *ptr = &arr[0];"
    for(int *ptr = arr; ptr < arr + 5; ptr++) {
        cout << *ptr << endl;
    }

    return 0;
}
```

File Redirect

Super useful for testing

If your program uses cin/cout, you can redirect the stream to a file

To use:

./programname.exe {anyCommandLineArguments} < inFile.txt > outFile.txt

Warning: it will overwrite your outFile!!!

Before Redirect

Have to type in every integer:(

```
 8    int temp;
 9
10    // finding average of 100 ints
11    double sum = 0;
12
13    for(int i = 0; i < 100; ++i) {
14        cout << "Enter an int: ";
15        cin >> temp;
16        sum += temp;
17    }
18
19    cout << "Average is: " << sum / 100;
20
```

Input Redirect

./test.exe < data.txt

Don’t have to type 100 ints!

```
 8    int temp;
 9
10    // finding average of 100 ints
11    double sum = 0;
12
13    for(int i = 0; i < 100; ++i) {
14        cout << "Enter an int: ";
15        cin >> temp;
16        sum += temp;
17    }
18
19    cout << "Average is: " << sum / 100;
20
```

Input and Output Redirect

./test.exe < data.txt > output.txt

Will create new file if it doesn't exist and WRITE OVER EXISTING!!!!

output.txt

```
1   Enter an int: Enter an int: Enter an int: Enter an int: Enter an int: Enter an int: En
2   Average is: 22.52
3
```

Diff Checker

https://www.diffchecker.com/

Super nice to figure out where you outputs are different

Example

Stats Class

git clone https://github.com/emolson16/stats

https://drive.google.com/file/d/1kCTB7AwToT4u0AnzHac_fwBWXM8YBb_i/view?usp=sharing

Note: this is purposefully buggy!

How to find bugs in your code

First problem: segmentation fault

Prints nothing- where could the bug be?

```
45  int main() {
46
47    Stats myStats(3);
48
49    // set array to 3, 2, 1
50    myStats.setVal(0, 3);
51    myStats.setVal(1, 2);
52    myStats.setVal(2, 1);
53
54    cout << "Find max of array. Expect 3" << endl;
55    cout << "Array max: " << myStats.max() << endl;
56
57    // set array to 1, 2, 3
58    myStats.setVal(0, 1);
59    myStats.setVal(1, 2);
60    myStats.setVal(2, 3);
61
62    cout << "Find max of array. Expect 3" << endl;
63    cout << "Array max: " << myStats.max() << endl;
64
65    return 0;
66  }
67
```

How to find bugs in your code

First problem: segmentation fault

Prints nothing- where could the bug be?

- Must be in constructor or setVal

```
45  int main() {
46
47      Stats myStats(3);
48
49      // set array to 3, 2, 1
50      myStats.setVal(0, 3);
51      myStats.setVal(1, 2);
52      myStats.setVal(2, 1);
53
54      cout << "Find max of array. Expect 3" << endl;
55      cout << "Array max: " << myStats.max() << endl;
56
57      // set array to 1, 2, 3
58      myStats.setVal(0, 1);
59      myStats.setVal(1, 2);
60      myStats.setVal(2, 3);
61
62      cout << "Find max of array. Expect 3" << endl;
63      cout << "Array max: " << myStats.max() << endl;
64
65      return 0;
66  }
67
```

How to find bugs in your code

Second problem: incorrect max on line 63

Prints 1- where could the bug be?

```
45  int main() {
46
47    Stats myStats(3);
48
49    // set array to 3, 2, 1
50    myStats.setVal(0, 3);
51    myStats.setVal(1, 2);
52    myStats.setVal(2, 1);
53
54    cout << "Find max of array. Expect 3" << endl;
55    cout << "Array max: " << myStats.max() << endl;
56
57    // set array to 1, 2, 3
58    myStats.setVal(0, 1);
59    myStats.setVal(1, 2);
60    myStats.setVal(2, 3);
61
62    cout << "Find max of array. Expect 3" << endl;
63    cout << "Array max: " << myStats.max() << endl;
64
65    return 0;
66  }
67
```

How to find bugs in your code

Second problem: incorrect max on line 63

Prints 1- where could the bug be?

- Must be in setVal or max

```
45  int main() {
46
47    Stats myStats(3);
48
49    // set array to 3, 2, 1
50    myStats.setVal(0, 3);
51    myStats.setVal(1, 2);
52    myStats.setVal(2, 1);
53
54    cout << "Find max of array. Expect 3" << endl;
55    cout << "Array max: " << myStats.max() << endl;
56
57    // set array to 1, 2, 3
58    myStats.setVal(0, 1);
59    myStats.setVal(1, 2);
60    myStats.setVal(2, 3);
61
62    cout << "Find max of array. Expect 3" << endl;
63    cout << "Array max: " << myStats.max() << endl;
64
65    return 0;
66  }
67
```

Valgrind Actually Doing its job

valgrind --leak-check=full ./fileName.exe

```
==1817== HEAP SUMMARY:
==1817==     in use at exit: 72 bytes in 1 blocks
==1817==   total heap usage: 3 allocs, 2 frees, 73,800 bytes allocated
==1817==
==1817== 72 bytes in 1 blocks are definitely lost in loss record 1 of 1
==1817==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-
linux.so)
==1817==    by 0x1094CE: Garage::Garage(int) (in /mnt/c/users/emily/OneDrive/eecs402/code/studentClass/car.exe)
==1817==    by 0x109234: main (in /mnt/c/users/emily/OneDrive/eecs402/code/studentClass/car.exe)
==1817==
==1817== LEAK SUMMARY:
==1817==    definitely lost: 72 bytes in 1 blocks
==1817==    indirectly lost: 0 bytes in 0 blocks
==1817==      possibly lost: 0 bytes in 0 blocks
==1817==    still reachable: 0 bytes in 0 blocks
==1817==         suppressed: 0 bytes in 0 blocks
==1817==
==1817== For lists of detected and suppressed errors, rerun with: -s
==1817== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
emolson@DESKTOP-SDRJGK7:/mnt/c/users/emily/OneDrive/eecs402/code/studentClass$ g++ -std=c++98 car.cpp -o car.exe
```

Not what you want!

Valgrind Actually Doing its Job

```
==1831== HEAP SUMMARY:
==1831==     in use at exit: 0 bytes in 0 blocks
==1831==   total heap usage: 3 allocs, 3 frees, 73,800 bytes allocated
==1831==
==1831== All heap blocks were freed -- no leaks are possible
==1831==
==1831== For lists of detected and suppressed errors, rerun with: -s
==1831== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Good!

Example Solution

```
class Stats {
    private:
        int size;
        int *arr;

    public:

        Stats(int sizeIn) {
            size = sizeIn;
            arr = new int[size];
        }

        void setVal(int index, int num) {
            arr[index] = num;
        }

        int max() {
            int maxElt = arr[0];

            for(int i = 0; i < size; ++i) {
                if(arr[i] > maxElt) {
                    maxElt = arr[i];
                }
            }

            return maxElt;
        }

        void printArr() {
            for(int i = 0; i < size; ++i) {
                cout << arr[i] << endl;
            }
        }

        ~Stats() {
            delete[] arr;
            arr = NULL;
        }

};
```