Programming Practicum Report: Meeting #5

R. Ethan Halim

September 22nd, 2024

1 Fizz Buzz

The entire source file is hosted on a GitHub repository **here**.

1.1 Explanation

As it is required to have the utilization of `std::vector` in the code, the vector used shall be to store the string of each value from 1 to 100 through the fizz-buzz algorithm. For performance, the `.reserve` method of `std::vector` is utilized, in order to avoid reallocations of the vector elements in the heap.

```
int program(std::istream& cin, std::ostream& cout) {
    // Requirement: the use of std::vector
    std::vector<std::string> result;
    result.reserve(100); // So that the vector doesn't need to
    ↪  reallocate

    ...
}
```

The for-loop below iterates from 1 to 100. In order to check whether the iterator `i` is a multiple of 3, 5, or both, the modulo operator `%` is utilized to check the division remainder, whereby the result `0` signifies that the nominator is a multiple of the denominator. As the fizz-buzz algorithm needs to check the case where it is a multiple of both 3 and 5, the said condition shall precede the other two. `"FizzBuzz"`, `"Fizz"`, `"Buzz"`, or the decimal presentation of the number itself is then pushed into the `result` array.

```
int program(std::istream& cin, std::ostream& cout) {
    ...
```

```
    for (int i = 1; i <= 100; i++) {
        // Both multiples of 3 and 5
        if (i % 3 == 0 && i % 5 == 0) {
            result.push_back("FizzBuzz");
        }
        // Only a multiple of 3
        else if (i % 3 == 0) {
            result.push_back("Fizz");
        }
        // Only a multiple of 5
        else if (i % 5 == 0) {
            result.push_back("Buzz");
        }
        // If neither
        else {
            result.push_back(std::to_string(i));
        }
    }

    ...
}
```

Each element of the `result` array is then printed out one by one upon the conclusion of the algorithm.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    // Prints each element in the final vector
    for (std::string& str : result) {
        cout << "- " << str << '\n';
    }

    return 0;
}
```

1.2 Manual Testing

Below is the compilation and the testing of the source code. The output is cut off to the 31st line, as the screenshot is unable to fit all 100 lines of the output.

```
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/01_fizz_buzz$ make
g++ -Wall fizz_buzz.cpp -o fizz_buzz
./fizz_buzz
- 1
- 2
- Fizz
- 4
- Buzz
- Fizz
- 7
- 8
- Fizz
- Buzz
- 11
- Fizz
- 13
- 14
- FizzBuzz
- 16
- 17
- Fizz
- 19
- Buzz
- Fizz
- 22
- 23
- Fizz
- Buzz
- 26
- Fizz
- 28
- 29
- FizzBuzz
- 31
```

1.3 Test Cases

1.3.1 Tests

Below is copied directly from the `tests.txt` file.

```
%INPUT
%OUTPUT
- 1
- 2
- Fizz
- 4
- Buzz
- Fizz
- 7
- 8
- Fizz
- Buzz
- 11
- Fizz
- 13
- 14
- FizzBuzz
- 16
- 17
- Fizz
- 19
- Buzz
- Fizz
- 22
- 23
- Fizz
- Buzz
- 26
- Fizz
- 28
- 29
- FizzBuzz
- 31
- 32
- Fizz
- 34
- Buzz
- Fizz
- 37
```

```
- 38
- Fizz
- Buzz
- 41
- Fizz
- 43
- 44
- FizzBuzz
- 46
- 47
- Fizz
- 49
- Buzz
- Fizz
- 52
- 53
- Fizz
- Buzz
- 56
- Fizz
- 58
- 59
- FizzBuzz
- 61
- 62
- Fizz
- 64
- Buzz
- Fizz
- 67
- 68
- Fizz
- Buzz
- 71
- Fizz
- 73
- 74
- FizzBuzz
- 76
- 77
- Fizz
- 79
- Buzz
- Fizz
```

```
- 82
- 83
- Fizz
- Buzz
- 86
- Fizz
- 88
- 89
- FizzBuzz
- 91
- 92
- Fizz
- 94
- Buzz
- Fizz
- 97
- 98
- Fizz
- Buzz
%END
```

1.3.2 Execution

Below are the results of the test cases. No test cases failed.

```
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/01_fizz_buzz$ make clean
rm -f fizz_buzz fizz_buzz_test
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/01_fizz_buzz$ make test
g++ -Wall -g fizz_buzz.cpp -o fizz_buzz_test -DTEST
./fizz_buzz_test
[*] The program is currently in test mode!

[*] Running test #1 with the input...
[*] Test ran successfully.

[*] All tests passed.
```

2 Reversing a Vector

The entire source file is hosted on a GitHub repository **here**.

2.1 Explanation

First and foremost, the program has to accept a vector of integers as input from the user. This part of the code which comprises most of the source file handles the parsing of the vector input. As per what's desired, the vectors that are fed in are to be formatted as follows.

```
[1, 2, 3, 4, 5, ..., n]
```

... where the vector begins with an opening square bracket and terminates with a closing square bracket, and the integer elements are comma-separated. Additionally, the algorithm to handle the parsing of such input string must take into factor extra whitespace. Therefore, it must also handle cases like as follows.

```
[    1    ,    2    ,    3    ,    4    ,    5    ]
```

I have provided brief explanations to each statement clause as comments in the code below. As an added auxiliary, the code checks for invalid syntax and informs the user thereof.

```cpp
int program(std::istream& cin, std::ostream& cout) {
    cout << "Input: ";
    std::vector<int64_t> vec;
    char inp;

    // Clears any whitespace
    do {
        cin >> inp;
    } while (std::isspace(inp));

    // Expects an opening square bracket for the vector
    if (inp != '[') {
        cout << "Expected an opening square bracket!\n";
        return 1;
    }

    // Clears any whitespace
    do {
        cin >> inp;
    } while (std::isspace(inp));
```

```
    // Loops until the array is closed
    while (inp != ']') {
        // The latter condition also handles negative numbers
        if (!('0' <= inp && inp <= '9') && inp != '-') {
            cout << "Expected a number, but instead saw '" <<
            ↪  inp << "'!\n";
            return 1;
        }

        // Returns the checked character and reads for the
        ↪  integer
        cin.unget();
        vec.push_back(0);
        cin >> vec[vec.size() - 1];

        // Clears any whitespace
        do {
            cin >> inp;
        } while (std::isspace(inp));

        // Hints that the vector has ended
        if (inp == ']') {
            break;
        }

        if (inp != ',') {
            cout << "Expected a comma, but instead saw '" <<
            ↪  inp << "'!\n";
            return 1;
        }

        // Clears any whitespace
        do {
            cin >> inp;
        } while (std::isspace(inp));
    }

    ...
}
```

Below is the actual part which deals with the reversal of the vector. The reversed content of the vector shall be stored in the variable `reversed`. The `.reserve` method of `std::vector` is utilized, in order to cull unnecessary reallocations of the elements in the heap upon appendage. The for-loop iterates backwards from `n - 1` (the index of the last element) to `0` (the index of the first element), in order to access the elements of `vec` in a similarly backwards manner, each of which is appended to the `reversed` vector.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    // Reverses the vector
    std::vector<int64_t> reversed;
    reversed.reserve(vec.size()); // So that the vector
    ↪ doesn't need to reallocate
    for (size_t i = vec.size() - 1; i < vec.size(); i--) { //
    ↪ Iterates backwards
        reversed.push_back(vec[i]);
    }

    ...
}
```

Finally, the program prints the resulted `reversed` vector and formats the vector properly, where the elements are enclosed in square brackets, and each element is separated in between each other with a comma.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    // Prints the content of the reversed vector
    cout << "Output: [";
    for (size_t i = 0; i < reversed.size(); i++) {
        cout << reversed[i];
        if (i < reversed.size() - 1) {
            cout << ", ";
        }
    }
    cout << "]\n";

    return 0;
}
```

2.2 Manual Testing

Below is the compilation and the testing of the source code.

```
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/02_reversal$ make
./reversal
Input: [132, 543, 763, 324, 684, 328, -342, 567, 564, -231, -546, -344, -12, 1, 0]
Output: [0, 1, -12, -344, -546, -231, 564, 567, -342, 328, 684, 324, 763, 543, 132]
```

2.3 Test Cases

2.3.1 Tests

Below is copied directly from the `tests.txt` file. The test cases test for scenarios, such as the lack of elements, a single element, multiple elements, multidigit elements, negative elements, and mixed blends of the said circumstances.

```
%INPUT
[]
%OUTPUT
Input: Output: []
%END

%INPUT
[69420]
%OUTPUT
Input: Output: [69420]
%END

%INPUT
[1, 2, 3, 4, 5]
%OUTPUT
Input: Output: [5, 4, 3, 2, 1]
%END

%INPUT
[123, 234, 345, 456, 567]
%OUTPUT
Input: Output: [567, 456, 345, 234, 123]
%END

%INPUT
[-69420]
%OUTPUT
Input: Output: [-69420]
%END
```

```
%INPUT
[-1, -2, -3, -4, -5]
%OUTPUT
Input: Output: [-5, -4, -3, -2, -1]
%END

%INPUT
[-1, 2, -3, 4, -5, 6, -7, 8, -9, 10, -11, 12, -13, 14, -15, 16,
 ↪  -17, 18, -19, 20, -21, 22, -23, 24, -25, 26, -27, 28, -29,
 ↪  30, -31, 32, -33, 34, -35, 36, -37, 38, -39, 40, -41, 42,
 ↪  -43, 44, -45, 46, -47, 48, -49, 50]
%OUTPUT
Input: Output: [50, -49, 48, -47, 46, -45, 44, -43, 42, -41,
 ↪  40, -39, 38, -37, 36, -35, 34, -33, 32, -31, 30, -29, 28,
 ↪  -27, 26, -25, 24, -23, 22, -21, 20, -19, 18, -17, 16, -15,
 ↪  14, -13, 12, -11, 10, -9, 8, -7, 6, -5, 4, -3, 2, -1]
%END
```

2.3.2 Execution

Below are the results of the test cases. No test cases failed.

```
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/02_reversal$ make clean
rm -f reversal reversal_test
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_5/02_reversal$ make test
g++ -Wall -g reversal.cpp -o reversal_test -DTEST
./reversal_test
[*] The program is currently in test mode!

[*] Running test #1 with the input...
[]
[*] Test ran successfully.

[*] Running test #2 with the input...
[69420]
[*] Test ran successfully.

[*] Running test #3 with the input...
[1, 2, 3, 4, 5]
[*] Test ran successfully.

[*] Running test #4 with the input...
[123, 234, 345, 456, 567]
[*] Test ran successfully.

[*] Running test #5 with the input...
[-69420]
[*] Test ran successfully.

[*] Running test #6 with the input...
[-1, -2, -3, -4, -5]
[*] Test ran successfully.

[*] Running test #7 with the input...
[-1, 2, -3, 4, -5, 6, -7, 8, -9, 10, -11, 12, -13, 14, -15, 16, -17, 18, -19, 20, -21, 22,
 -45, 46, -47, 48, -49, 50]
[*] Test ran successfully.

[*] All tests passed.
```