Meeting 5

Avicena Yosanta Muchtar 24/536710/PA/22764

26 September 2024

Github: https://github.com/TheKurusUGM/Praktikum-Pemograman-UGM-/tree/23ffdb526ca857effa4c300797bb36f6e61094e8/Week$_5$

1 First Problem

Question:
The FizzBuzz problem is defined as follows: Write a C++ program that prints the numbers from 1 to 100. However, for multiples of three, print "Fizz" instead of the number, and for multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz". You must store the result of this computation in a std::vector before printing the output.

Answer:

```cpp

#include <iostream>
#include <vector>
#include <string>

int main() {
    std::vector<std::string> FizzBuzzResults;

    for (int i = 1; i <= 100; ++i) {
        if (i % 3 == 0 && i % 5 == 0) {
            FizzBuzzResults.push_back("FizzBuzz");
        } else if (i % 3 == 0) {
            FizzBuzzResults.push_back("Fizz");
        } else if (i % 5 == 0) {
            FizzBuzzResults.push_back("Buzz");
        } else {
            FizzBuzzResults.push_back(std::to_string(i));
        }
    }

    for (const std::string& result : FizzBuzzResults) {
        std::cout << result << std::endl;
    }

    return 0;
}
```

Explanation:

```
    #include <iostream>
#include <vector>
#include <string>
```

#include <vector>: Provides the std::vector container class, which allows storing a sequence
of results (strings in this case).
#include <string>: Provides the std::string class, which is used to handle text (\Fizz", \Buzz", and numbers as strings).

```
    std::vector<std::string> FizzBuzzResults;
```

A std::vector of type std::string named FizzBuzzResults is written.
This vector will store the results for each number from 1 to 100, where each result is either \Fizz", \Buzz", \FizzBuzz", or the number itself as a string.

```

    for (int i = 1; i <= 100; ++i) {

```

This is a for loop that show over the numbers from 1 to 100 . The loop index variable i starts at 1 and increments by 1 until it reaches 100.

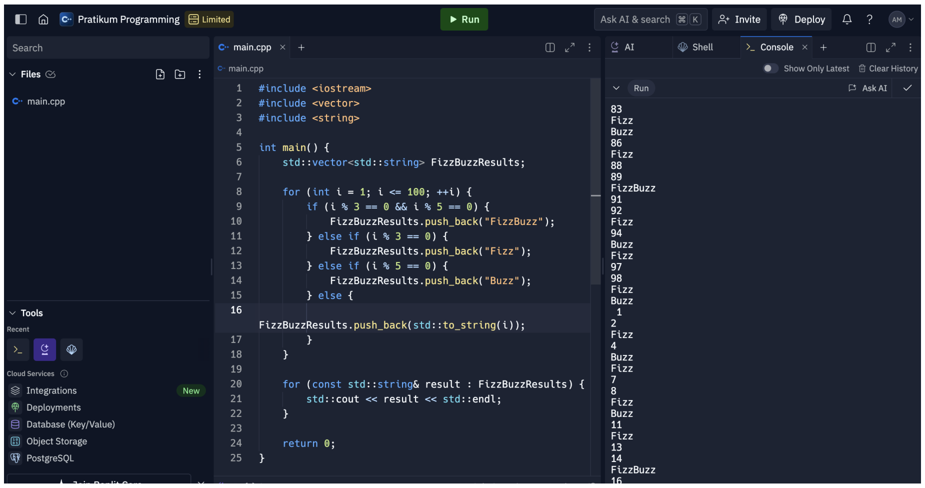
```
        if (i % 3 == 0 && i % 5 == 0) {
            FizzBuzzResults.push_back("FizzBuzz");
        } else if (i % 3 == 0) {
            FizzBuzzResults.push_back("Fizz");
        } else if (i % 5 == 0) {
            FizzBuzzResults.push_back("Buzz");
        } else {
            FizzBuzzResults.push_back(std::to_string(i));
        }
```

Conditional Statements:
- if (i % 3 == 0 && i % 5 == 0): This checks if the current number i is divisible by both 3 and 5. If true, it means the number is a multiple of both 3 and 5, so the string "FizzBuzz" is added to the vector.
- else if (i % 3 == 0): If the number is divisible only by 3, the string "Fizz" is added to the vector.
- else if (i % 5 == 0): If the number is divisible only by 5, the string "Buzz" is added to the vector.
- else: If none of the above conditions are met , the number itself is added to the vector. The number is converted to a string using std::to_string(i) before being stored.

```cpp
    for (const std::string& result : FizzBuzzResults) {
        std::cout << result << std::endl;
    }
```

This loop iterates over each element in the fizzBuzzResults vector.
const std::string& result: The loop retrieves each element from the vector as a constant reference (to avoid unnecessary copying).

The value of result (which could be \Fizz", \Buzz", \FizzBuzz", or a number) is printed to the console using std::cout.



```cpp
#include <iostream>
#include <vector>
#include <string>

int main() {
    std::vector<std::string> FizzBuzzResults;

    for (int i = 1; i <= 100; ++i) {
        if (i % 3 == 0 && i % 5 == 0) {
            FizzBuzzResults.push_back("FizzBuzz");
        } else if (i % 3 == 0) {
            FizzBuzzResults.push_back("Fizz");
        } else if (i % 5 == 0) {
            FizzBuzzResults.push_back("Buzz");
        } else {
FizzBuzzResults.push_back(std::to_string(i));
        }
    }

    for (const std::string& result : FizzBuzzResults) {
        std::cout << result << std::endl;
    }

    return 0;
}
```

```
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
 1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
```

Figure 1: FizzBuzz output

2 Second number

Question:
In this problem, you will write a function that takes a std::vector<int> as input and returns a new vector that contains the elements of the input vector in reverse order.
For example:
Input: [1, 2, 3, 4, 5]
Output: [5, 4, 3, 2, 1]

```cpp
#include <iostream>
#include <vector>


std::vector<int> ReverseArray(const std::vector<int>& input) {
    std::vector<int> reversed;


    for (int i = input.size() - 1; i >= 0; --i) {
        reversed.push_back(input[i]);
    }

    return reversed;
}

int main() {
    std::vector<int> input;
    int n, element;


    std::cout << "Enter the number of elements in the array: ";
    std::cin >> n;

    std::cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; ++i) {
        std::cin >> element;
        input.push_back(element);
    }

    std::vector<int> reversed = ReverseArray(input);

    std::cout << "Reversed array: ";
    for (int num : reversed) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Explanation:

```cpp
std::vector<int> reverseArray(const std::vector<int>& input) {
    std::vector<int> reversed;

    // Iterate over the input vector from the end to the
    ↪  beginning
    for (int i = input.size() - 1; i >= 0; --i) {
        reversed.push_back(input[i]);
    }

    return reversed;
}
```

ReverseArray Function:
std::vector<int> reverseArray(const std::vector<int>& input)

This function takes a const std::vector<int>& as an argument.

It returns a new vector (std::vector<int>) that contains the elements of the input vector in reverse order.

Reversed vector: A new empty vector reversed is created to store the elements in reverse

order.
The loop iterates from the last index of the input vector (i.e., input.size() - 1) to index 0. Each element of the input vector is appended to the reversed vector using reversed. push_back(input[i]).

```
int main() {
    std::vector<int> input;
    int n, element;
```

The input vector will hold the user-provided elements.
- The variables n and element are declared
- n: Stores the number of elements the user wishes to input.
- element: Stores each individual element entered by the user.

after all the proccess above, we then use this to ask for the user input for the array

```
    std::cout << "Enter the number of elements in the array: ";
    std::cin >> n;

        std::cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; ++i) {
        std::cin >> element;
        input.push_back(element);
    }
```

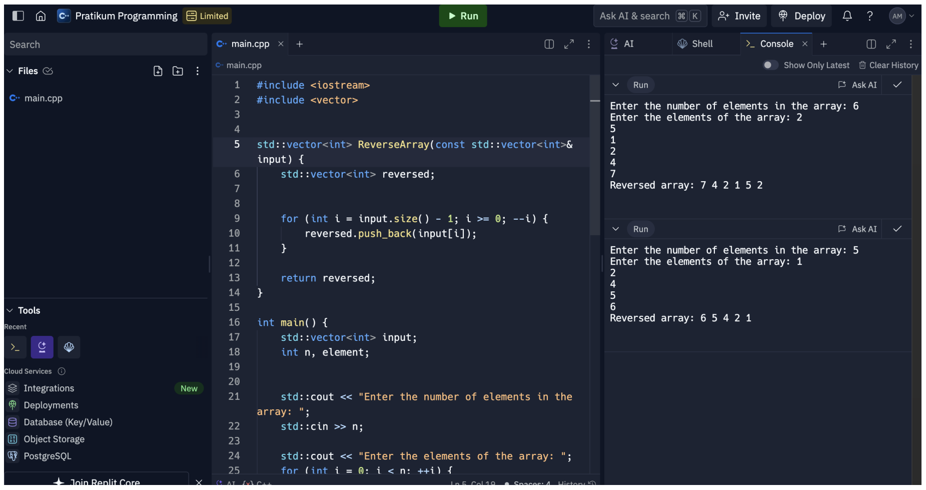
The user is prompted to enter how many elements they want in the array (n).
- std::cin >> n reads the input number, which specifies the number of elements.

The user is prompted to input the elements of the array.
- A for loop runs n times (based on the number entered by the user). During each iteration:
- The value entered by the user is stored in the variable element.
- input.push_back(element) appends the entered element to the input vector.

Here's how it will print the output:

```
    std::cout << "Reversed array: ";
    for (int num : reversed) {
        std::cout << num << " ";
    }
    std::cout << std::endl;
```

The program prints the reversed array.
- A range-based for loop is used to iterate over each element in the reversed vector, printi
the elements with spaces in between.



```cpp
#include <iostream>
#include <vector>


std::vector<int> ReverseArray(const std::vector<int>& input) {
    std::vector<int> reversed;


    for (int i = input.size() - 1; i >= 0; --i) {
        reversed.push_back(input[i]);
    }

    return reversed;
}

int main() {
    std::vector<int> input;
    int n, element;


    std::cout << "Enter the number of elements in the array: ";
    std::cin >> n;

    std::cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; ++i) {
```

```
Enter the number of elements in the array: 6
Enter the elements of the array: 2
5
1
2
4
7
Reversed array: 7 4 2 1 5 2
```

```
Enter the number of elements in the array: 5
Enter the elements of the array: 1
2
4
5
6
Reversed array: 6 5 4 2 1
```

Figure 2: Output of ReverseArray code