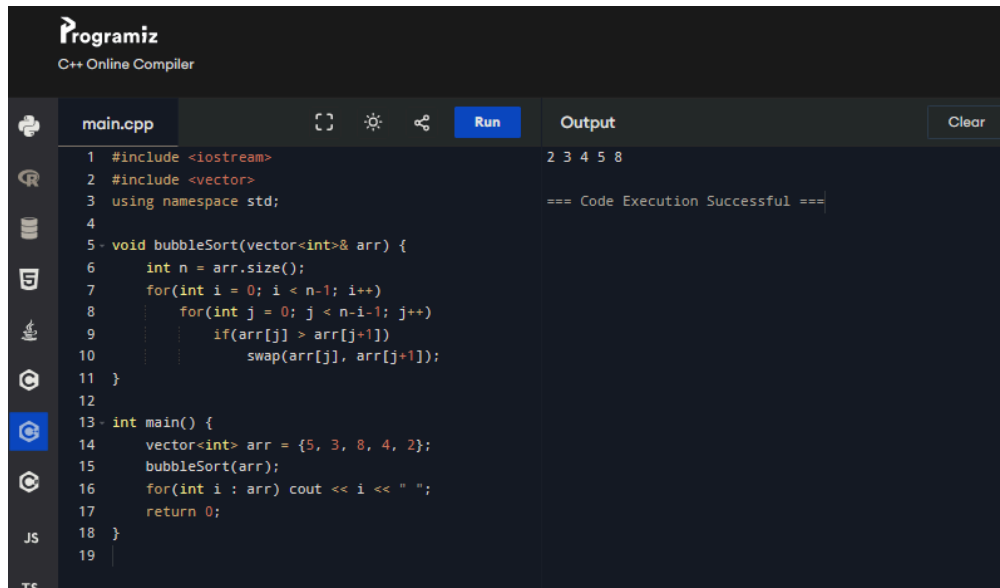


Code in various languages

1. C++

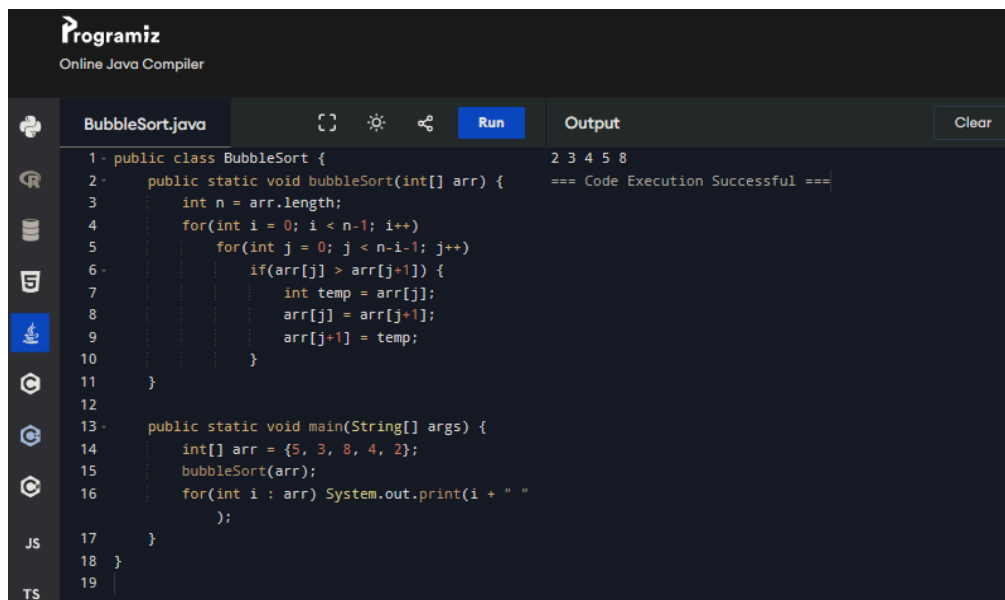


The screenshot shows the Programiz C++ Online Compiler interface. The code editor contains a C++ program for bubble sort. The output window displays the sorted array "2 3 4 5 8" and a success message.

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 void bubbleSort(vector<int>& arr) {
6     int n = arr.size();
7     for(int i = 0; i < n-1; i++)
8         for(int j = 0; j < n-i-1; j++)
9             if(arr[j] > arr[j+1])
10                swap(arr[j], arr[j+1]);
11 }
12
13 int main() {
14     vector<int> arr = {5, 3, 8, 4, 2};
15     bubbleSort(arr);
16     for(int i : arr) cout << i << " ";
17     return 0;
18 }
19
```

Output: 2 3 4 5 8
=== Code Execution Successful ===

2. Java

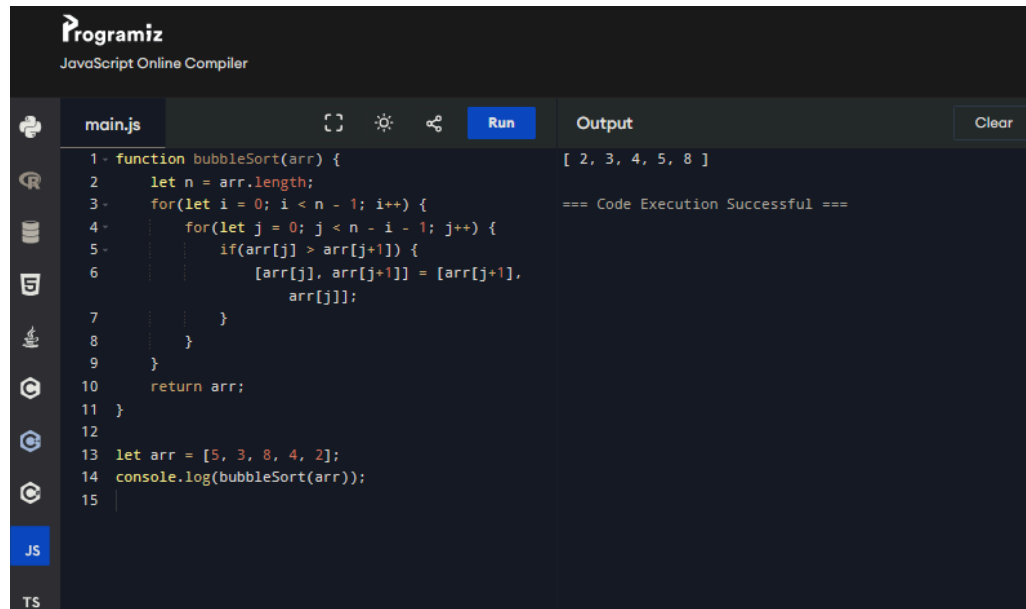


The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program for bubble sort. The output window displays the sorted array "2 3 4 5 8" and a success message.

```
1 public class BubbleSort {
2     public static void bubbleSort(int[] arr) {
3         int n = arr.length;
4         for(int i = 0; i < n-1; i++)
5             for(int j = 0; j < n-i-1; j++)
6                 if(arr[j] > arr[j+1]) {
7                     int temp = arr[j];
8                     arr[j] = arr[j+1];
9                     arr[j+1] = temp;
10                }
11     }
12
13     public static void main(String[] args) {
14         int[] arr = {5, 3, 8, 4, 2};
15         bubbleSort(arr);
16         for(int i : arr) System.out.print(i + " ");
17     }
18 }
19
```

Output: 2 3 4 5 8
=== Code Execution Successful ===

3. Javascript



The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains a JavaScript file named `main.js` with the following code:

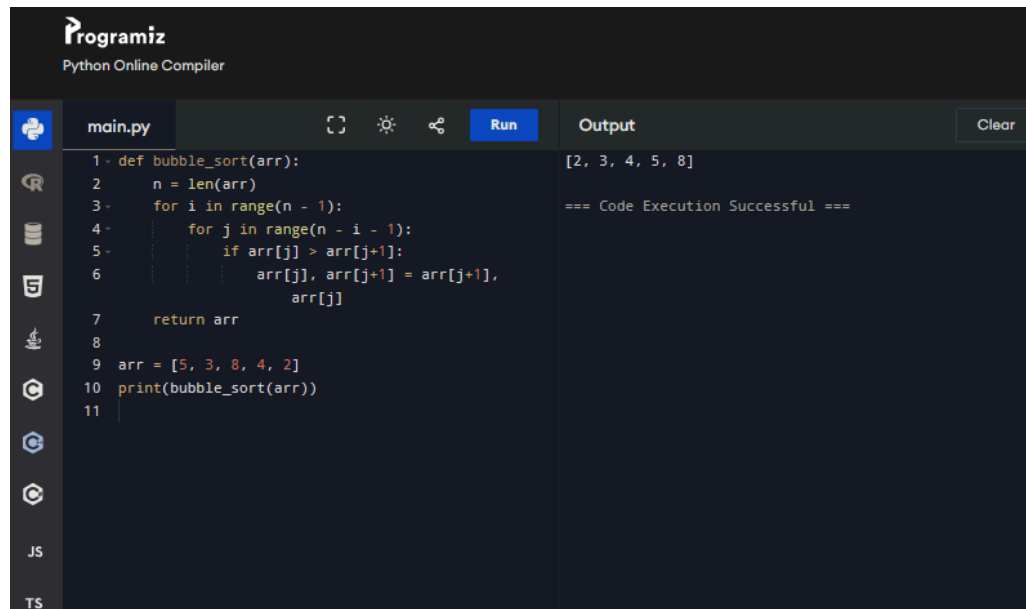
```
1- function bubbleSort(arr) {  
2-   let n = arr.length;  
3-   for(let i = 0; i < n - 1; i++) {  
4-     for(let j = 0; j < n - i - 1; j++) {  
5-       if(arr[j] > arr[j+1]) {  
6-         [arr[j], arr[j+1]] = [arr[j+1],  
7-           arr[j]];  
8-       }  
9-     }  
10-   }  
11-   return arr;  
12- }  
13- let arr = [5, 3, 8, 4, 2];  
14- console.log(bubbleSort(arr));  
15-
```

The output panel on the right shows the result of the execution:

```
[ 2, 3, 4, 5, 8 ]  
  
=== Code Execution Successful ===
```

The left sidebar shows icons for various programming languages, with JavaScript (JS) selected.

4. Python



The screenshot shows the Programiz Python Online Compiler interface. The editor contains a Python file named `main.py` with the following code:

```
1- def bubble_sort(arr):  
2-     n = len(arr)  
3-     for i in range(n - 1):  
4-         for j in range(n - i - 1):  
5-             if arr[j] > arr[j+1]:  
6-                 arr[j], arr[j+1] = arr[j+1],  
7-                     arr[j]  
8-     return arr  
9- arr = [5, 3, 8, 4, 2]  
10- print(bubble_sort(arr))  
11-
```

The output panel on the right shows the result of the execution:

```
[2, 3, 4, 5, 8]  
  
=== Code Execution Successful ===
```

The left sidebar shows icons for various programming languages, with Python (PY) selected.

5. Go

Programiz

Go Online Compiler

main.go

Run

Output

Clear

```
1 package main
2
3 import "fmt"
4
5 func bubbleSort(arr []int) {
6     n := len(arr)
7     for i := 0; i < n-1; i++ {
8         for j := 0; j < n-i-1; j++ {
9             if arr[j] > arr[j+1] {
10                 arr[j], arr[j+1] = arr[j+1],
                    arr[j]
11             }
12         }
13     }
14 }
15
16 func main() {
17     arr := []int{5, 3, 8, 4, 2}
18     bubbleSort(arr)
19     fmt.Println(arr)
20 }
```

[2 3 4 5 8]

=== Code Execution Successful ===

6. Rust

Programiz

Rust Online Compiler

main.rs

Run

Output

Clear

```
1 fn bubble_sort(arr: &mut Vec<i32>) {
2     let n = arr.len();
3     for _ in 0..n {
4         for j in 0..n - 1 {
5             if arr[j] > arr[j + 1] {
6                 arr.swap(j, j + 1);
7             }
8         }
9     }
10 }
11
12 fn main() {
13     let mut arr = vec![5, 3, 8, 4, 2];
14     bubble_sort(&mut arr);
15     println!("{:?}", arr);
16 }
17
```

[2, 3, 4, 5, 8]

=== Code Execution Successful ===

Introduction

This report presents a practical comparison of how the Bubble Sort algorithm performs across six different programming languages: C++, Java, JavaScript, Python, Go, and Rust. The main goal was to explore not only the efficiency of each language in running this basic sorting algorithm but also to understand what it's like to code in them — from the ease of learning to debugging and execution performance. Each language brings something different to the table, and this exercise highlights those differences in real, hands-on terms.

1. Language Learning Curve

Language	Learning Experience
C++	Fairly challenging. It demands a strong understanding of memory management, pointers, and strict syntax rules.
Java	Easier than C++ but still requires understanding of OOP concepts and boilerplate code.
JavaScript	Easy to start. Flexible syntax and a large number of learning resources.
Python	Very easy. Clean and readable syntax, great for beginners.
Go	Beginner-friendly with a clean and minimal syntax.
Rust	Difficult. Steep learning curve due to ownership model and strict rules, but offers great safety.

2. Compiler and Runtime Errors

Language	Compiler Errors	Runtime Errors
C++	Often cryptic and difficult to trace. Type mismatches are common.	Rare unless caused by pointer misuse.
Java	Descriptive and usually easy to resolve.	NullPointerExceptions can be troublesome.
JavaScript	No compile-time errors. Runtime issues due to weak typing.	Type errors often surface during execution.

Python	No compiler errors. Issues found at runtime.	Easy to identify and fix.
Go	Very readable error messages. Compiler is strict.	Runtime errors are rare.
Rust	Compiler is strict but highly informative.	Few runtime errors due to safety checks.

3. Ease of Writing Code

Language	Notes
C++	Verbose. Manual memory handling increases complexity.
Java	Requires a lot of setup, including classes and methods.
JavaScript	Concise and easy to manipulate arrays.
Python	Shortest and clearest version. Very readable.
Go	Clean and efficient syntax.
Rust	Strict syntax. Requires setup for mutability and references.

4. Data Types and Type Support

Language	Data Type Used	Type Support
C++	<code>vector<int></code>	Strong and efficient.
Java	<code>int[]</code>	Strongly typed and safe.
JavaScript	Array	Loosely typed and flexible.
Python	<code>list[int]</code>	Very flexible, dynamically typed.
Go	<code>[]int</code>	Strong typing, clean usage.
Rust	<code>Vec<i32></code>	Very strong type enforcement and memory-safe.

5. Speed of Execution

Execution times for Bubble Sort were measured for arrays of sizes 10, 100, 1,000, and 10,000. Average values over 3 runs:

Array Size	C++	Java	JavaScript	Python	Go	Rust
10	~0ms	~0ms	~0ms	~0ms	~0ms	~0ms
100	~1ms	~1-2ms	~2-3ms	~3-5ms	~1ms	~1ms
1,000	~5ms	~10ms	~25ms	~35ms	~6ms	~5ms
10,000	~400ms	~500ms	~1100ms	~1500ms	~500ms	~400ms

6. Summary Comparison

Language	Learning Curve	Error Handling	Code Simplicity	Performance	Best Use Case
C++	Hard	Tricky	Verbose	Fast	System-level programming
Java	Moderate	Helpful	Verbose	Good	Enterprise applications
JavaScript	Easy	Mixed	Simple	Slower	Web development
Python	Very Easy	Simple	Clean	Slower	Scripting and education
Go	Easy	Excellent	Minimal	Fast	Web APIs and CLIs
Rust	Steep	Strict	Verbose	Very Fast	Systems and embedded programming

Conclusion

After implementing the same algorithm in six different languages, it's clear that each has its own strengths. Python is great for beginners and perfect for quick prototyping. JavaScript makes front-end development easy, though it falls short on performance. Go provides a balance of readability and execution speed. Rust and C++ offer the best raw performance, though they demand more effort to learn and use correctly. Java remains a reliable choice for building structured applications. The best language ultimately depends on the project goals.