1141 CS321B Compiler Final Project

itzCode

Presented by:

1123318 王嘉成
1123323 林蔚瑄
1123334 蔡旻珈

Table of Contents

Abstract

- A Tiny Compiler for our custom programming language, **itzCode**.
- Frontend in Python, which transpiles .itz source code into C++
- Utilize g++ to compile the generated .cpp files into executable binaries

System Architecture

Lexer

- Space Elimination
- Tokenization

Parser

- Recursive Descent Parsing
- Driving Code Generation

Emitter

- Code mapping
- Buffer Management

Driver

- Pipeline Automation

the structure of our project

```
├── demo.py                # Program driver
├── examples/              # itzCode source code examples (*.itz)
│   ├── algorithm.itz      # Algorithm implementations (Bubble sort, max/min values)
│   ├── comments.itz       # Comment processing tests
│   ├── expressions.itz    # Math operations and type inference tests
│   ├── file.itz           # File I/O tests (read/write/append)
│   ├── function.itz       # Function and recursion tests
│   ├── hello-world.itz    # Basic string interpolation test
│   ├── input.itz          # User input (cin) tests
│   ├── logic.itz          # Logic gates and comparison operations tests
│   ├── loop.itz           # Loops (For/While) tests
│   └── random.itz         # Random number generation tests
├── src/                   # Compiler Core
│   ├── token.py           # Define Language Tokens (Token Enums)
│   ├── lexer.py           # Lexical Analyzer (Convert Raw Text to Tokens)
│   ├── parser.py          # Syntax Analyzer (Converts Tokens to C++ logic)
│   └── emitter.py         # Code Generator (Manages C++ output buffer)
└── results/               # Compilation Output (Generated .cpp and .exe files)
```

syntax

Variables & Types

- “DEF” → dynamic definition
- C++20 auto keyword

→ int, double, string, and arrays

Control Flow

- Supports

→ IF, ELSE IF, ELSE, THEN, ENDIF
→ FOR TO NEXT, WHILE REPEAT ENDWHILE loops

Functions & Recursion

- definition

→ FUNC name args ... ENDFUNC

- Supports

→ recursion and return values

I/O & File System

- Standard I/O (INPUT&ECHO)

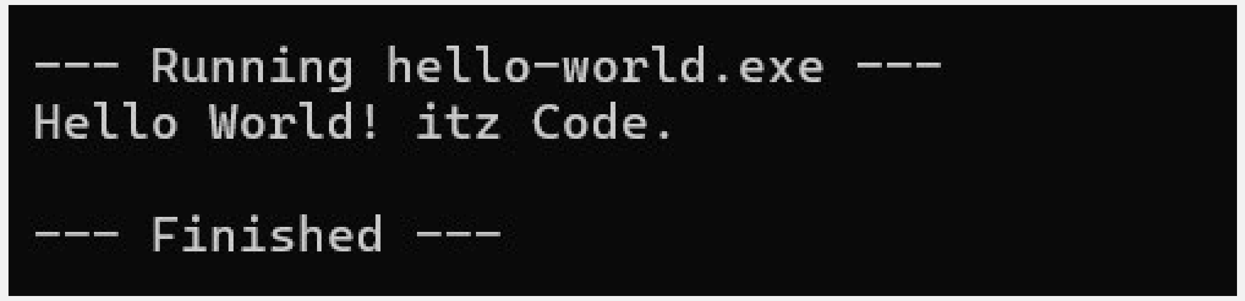
→ ECHO "Val: X”, INPUT

- File operation

→ FWRITE, FAPPEND, FREAD

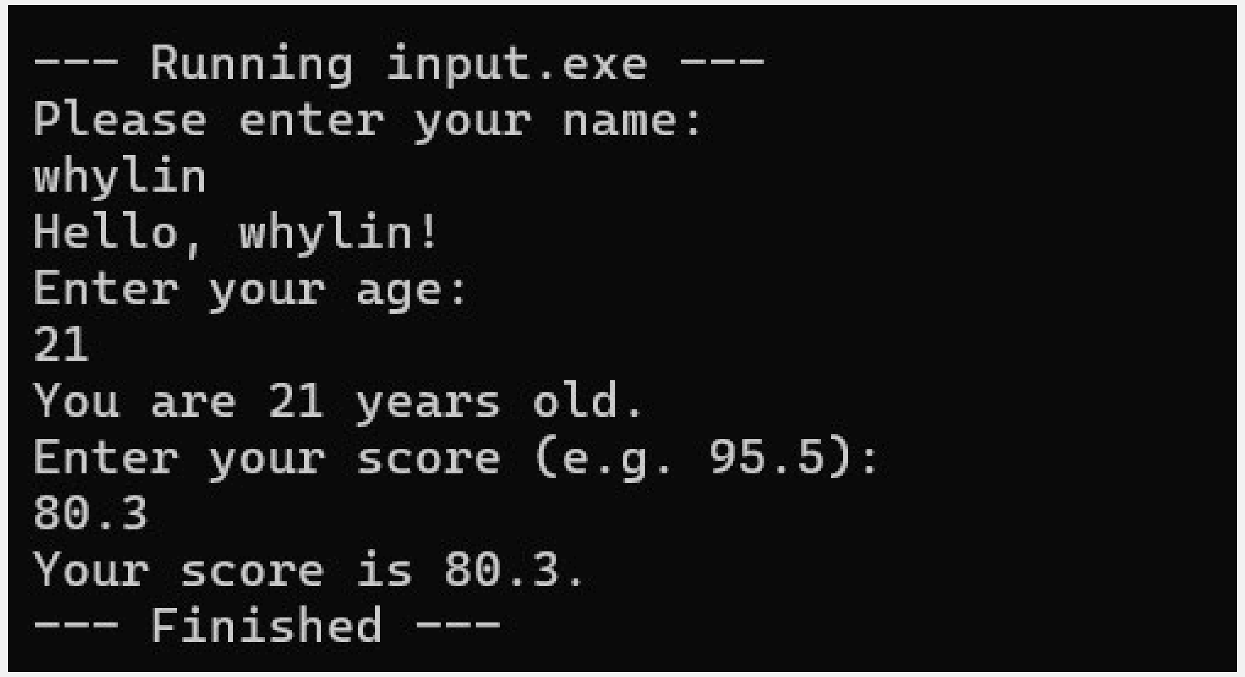
function

hello-world.itz



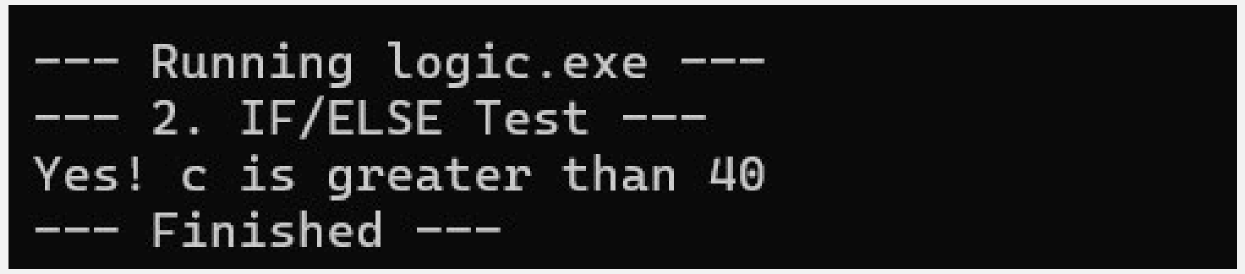
```
--- Running hello-world.exe ---
Hello World! itz Code.

--- Finished ---
```

input.itz



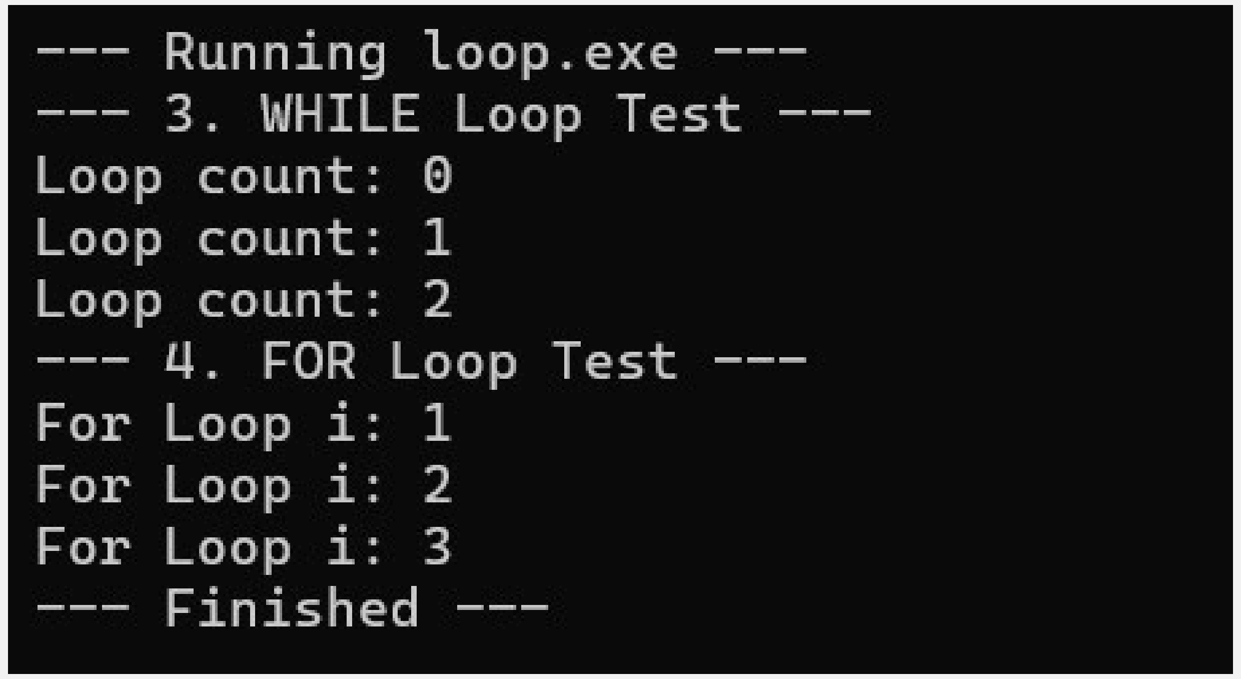
```
--- Running input.exe ---
Please enter your name:
whylin
Hello, whylin!
Enter your age:
21
You are 21 years old.
Enter your score (e.g. 95.5):
80.3
Your score is 80.3.
--- Finished ---
```

logic.itz



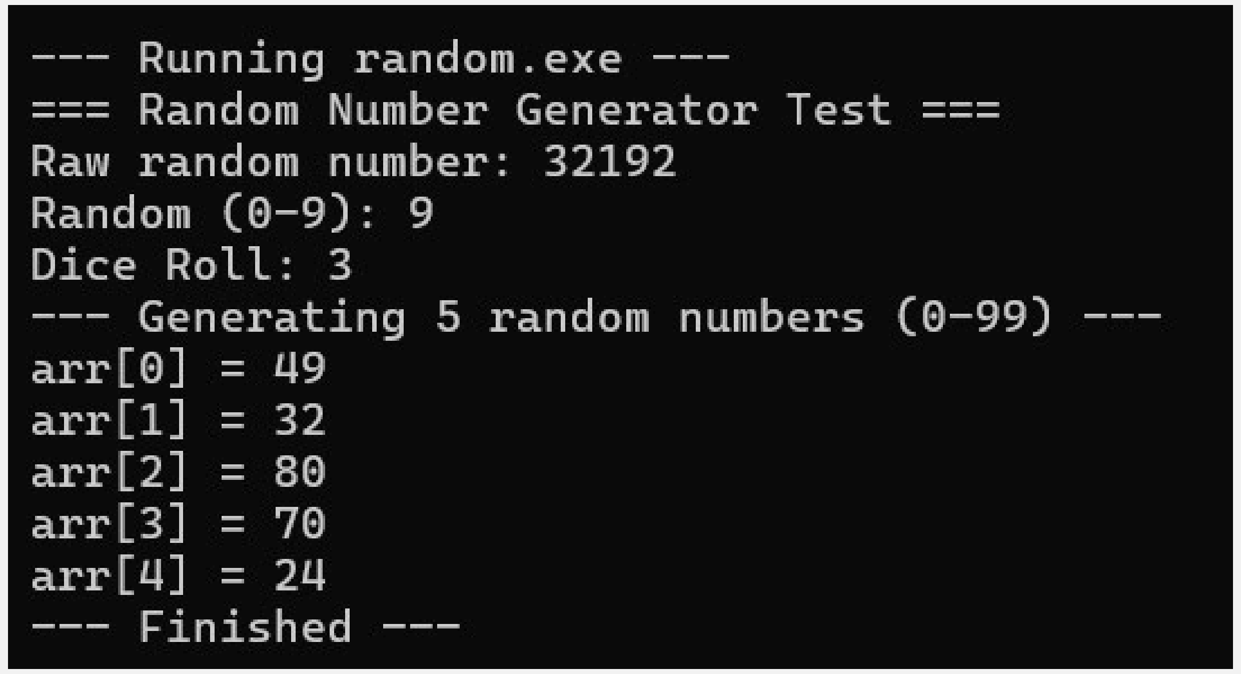
```
--- Running logic.exe ---
--- 2. IF/ELSE Test ---
Yes! c is greater than 40
--- Finished ---
```

loop.itz



```
--- Running loop.exe ---
--- 3. WHILE Loop Test ---
Loop count: 0
Loop count: 1
Loop count: 2
--- 4. FOR Loop Test ---
For Loop i: 1
For Loop i: 2
For Loop i: 3
--- Finished ---
```

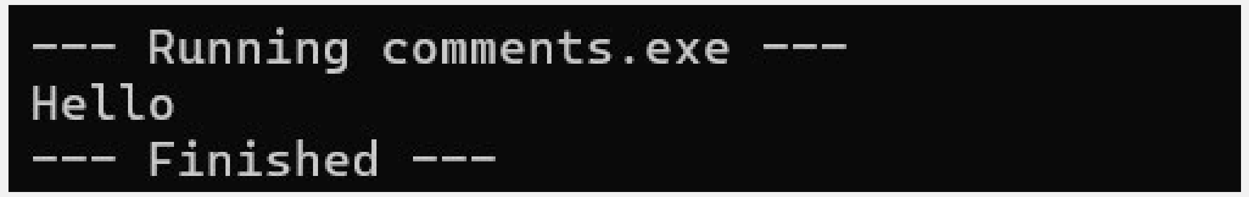
random.itz



```
--- Running random.exe ---
=== Random Number Generator Test ===
Raw random number: 32192
Random (0-9): 9
Dice Roll: 3
--- Generating 5 random numbers (0-99) ---
arr[0] = 49
arr[1] = 32
arr[2] = 80
arr[3] = 70
arr[4] = 24
--- Finished ---
```

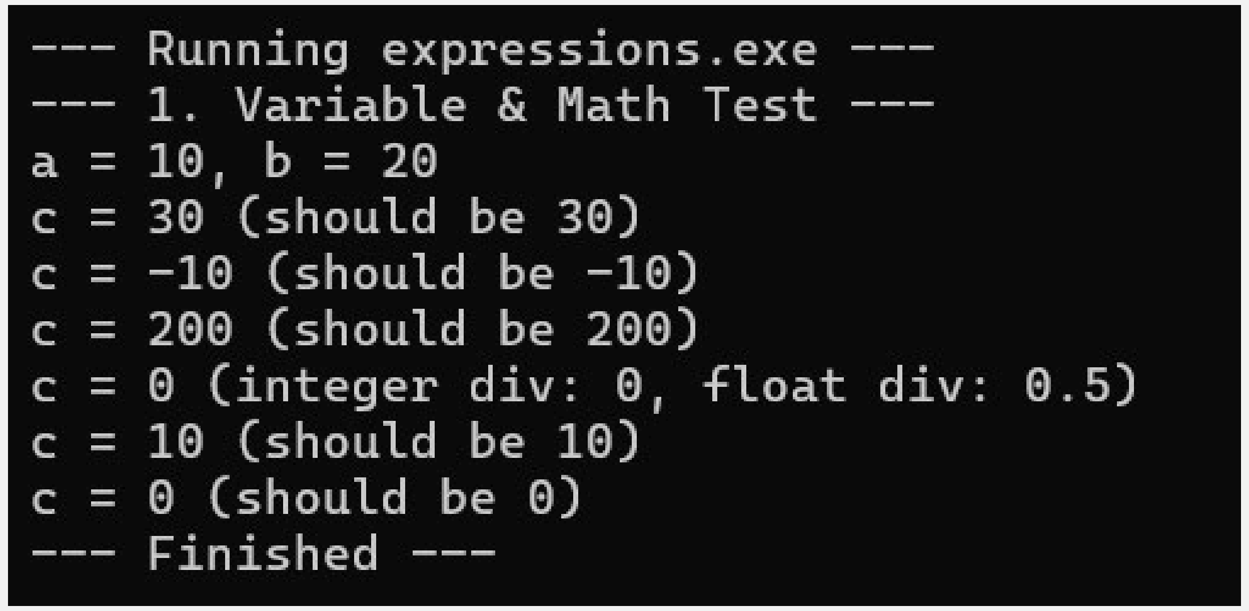
function

comments.itz



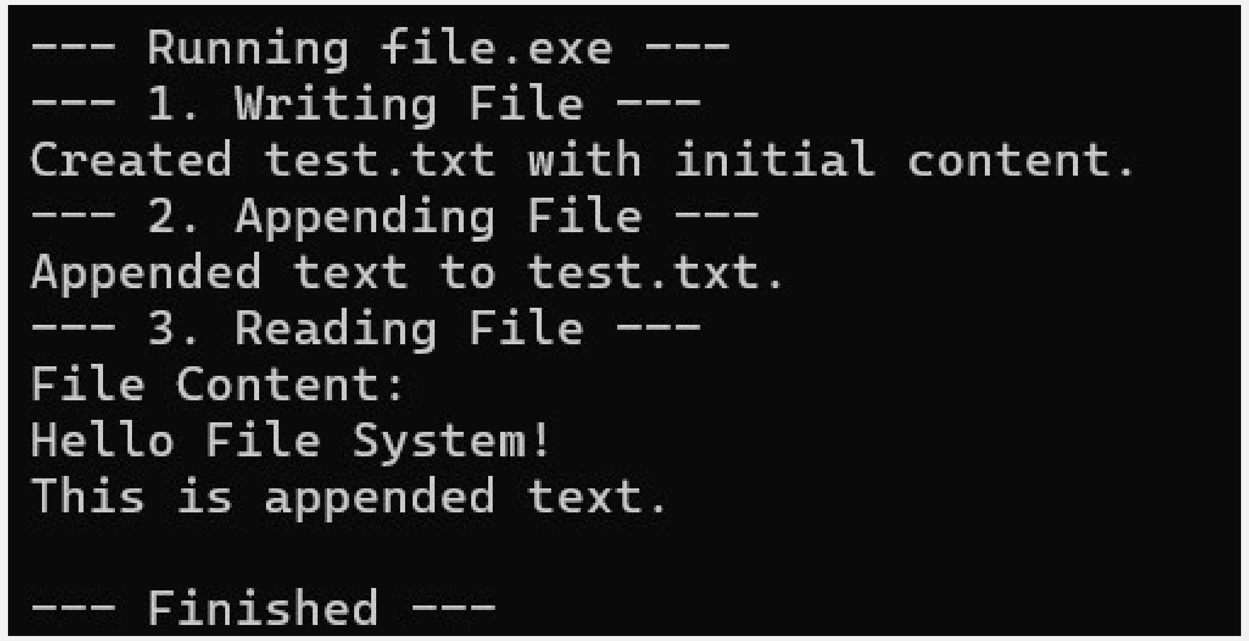
```
--- Running comments.exe ---
Hello
--- Finished ---
```

expressions.itz



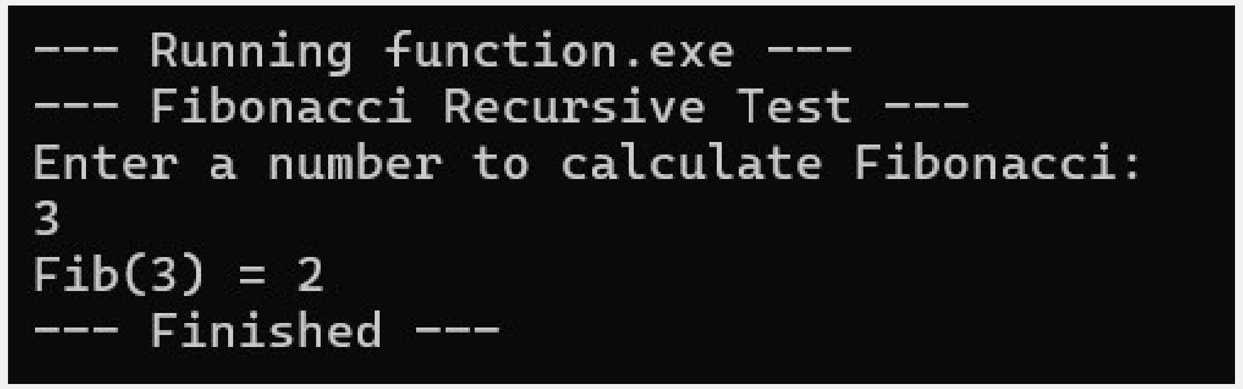
```
--- Running expressions.exe ---
--- 1. Variable & Math Test ---
a = 10, b = 20
c = 30 (should be 30)
c = -10 (should be -10)
c = 200 (should be 200)
c = 0 (integer div: 0, float div: 0.5)
c = 10 (should be 10)
c = 0 (should be 0)
--- Finished ---
```

file.itz



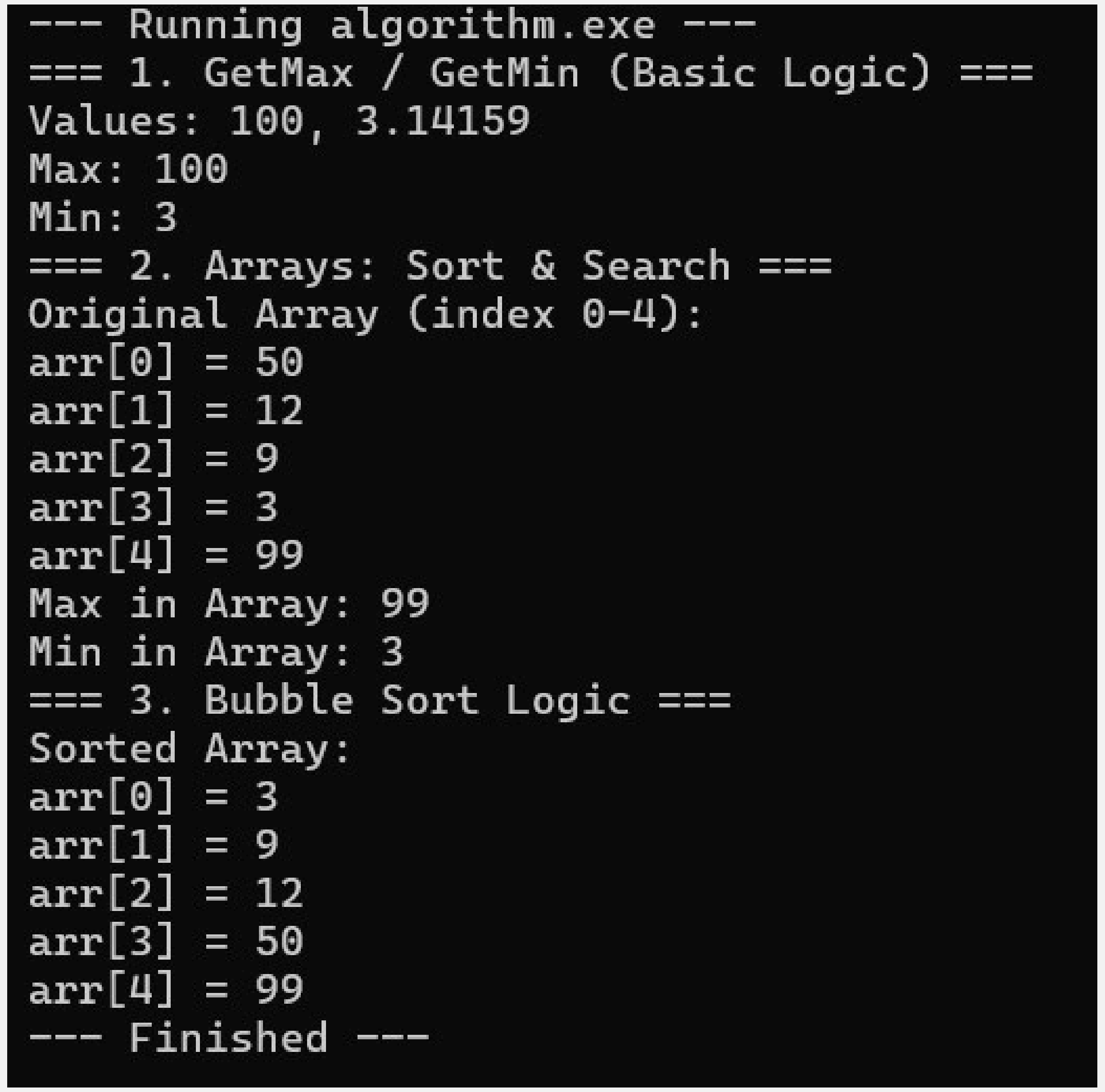
```
--- Running file.exe ---
--- 1. Writing File ---
Created test.txt with initial content.
--- 2. Appending File ---
Appended text to test.txt.
--- 3. Reading File ---
File Content:
Hello File System!
This is appended text.

--- Finished ---
```

function.itz



```
--- Running function.exe ---
--- Fibonacci Recursive Test ---
Enter a number to calculate Fibonacci:
3
Fib(3) = 2
--- Finished ---
```

algorithm.itz



```
--- Running algorithm.exe ---
=== 1. GetMax / GetMin (Basic Logic) ===
Values: 100, 3.14159
Max: 100
Min: 3
=== 2. Arrays: Sort & Search ===
Original Array (index 0-4):
arr[0] = 50
arr[1] = 12
arr[2] = 9
arr[3] = 3
arr[4] = 99
Max in Array: 99
Min in Array: 3
=== 3. Bubble Sort Logic ===
Sorted Array:
arr[0] = 3
arr[1] = 9
arr[2] = 12
arr[3] = 50
arr[4] = 99
--- Finished ---
```

demo & result

.itz .cpp

```
python .\demo.py --all
```

.cpp .exe

```
Get-ChildItem .\results\*.cpp | ForEach-Object {
  $out = Join-Path (Resolve-Path .\results) ($_.BaseName + ".exe")
  g++ -std=c++20 $_.FullName -o $out
}
```

Challenges and Solutions

Q1: Infinite Loop in Lexer

Problem: Unclosed " or ` makes lexer loop at EOF → compiler hangs.

When the source code contained an unclosed string (") or variable identifier (```), the Lexer would hit the End Of File (EOF) but continue looping because the termination character was never found, causing the compiler to hang.

Solution: Add EOF guard in scan loops (curChar != '\0').

Added an EOF check (self.curChar != '\0') inside the string and identifier parsing loops. If EOF is reached before the closing quote, the compiler now aborts with a clear error message.

Challenges and Solutions

Q2: Parser Stuck on Invalid Syntax

Problem: Unexpected token isn’t consumed → statement() repeats forever.

The statement() function in the Parser relied on if-elif blocks to match tokens. If an unexpected token (syntax error) appeared, none of the blocks executed, but the token wasn't consumed. This caused the main loop to call statement() repeatedly on the same token, freezing the program.

Solution: Add final else to catch invalid syntax and throw Parsing Error.

Implemented a final else block in the statement() function. This acts as a "catch-all" for invalid syntax, raising a specific Parsing Error and terminating the process immediately instead of looping.

Challenges and Solutions

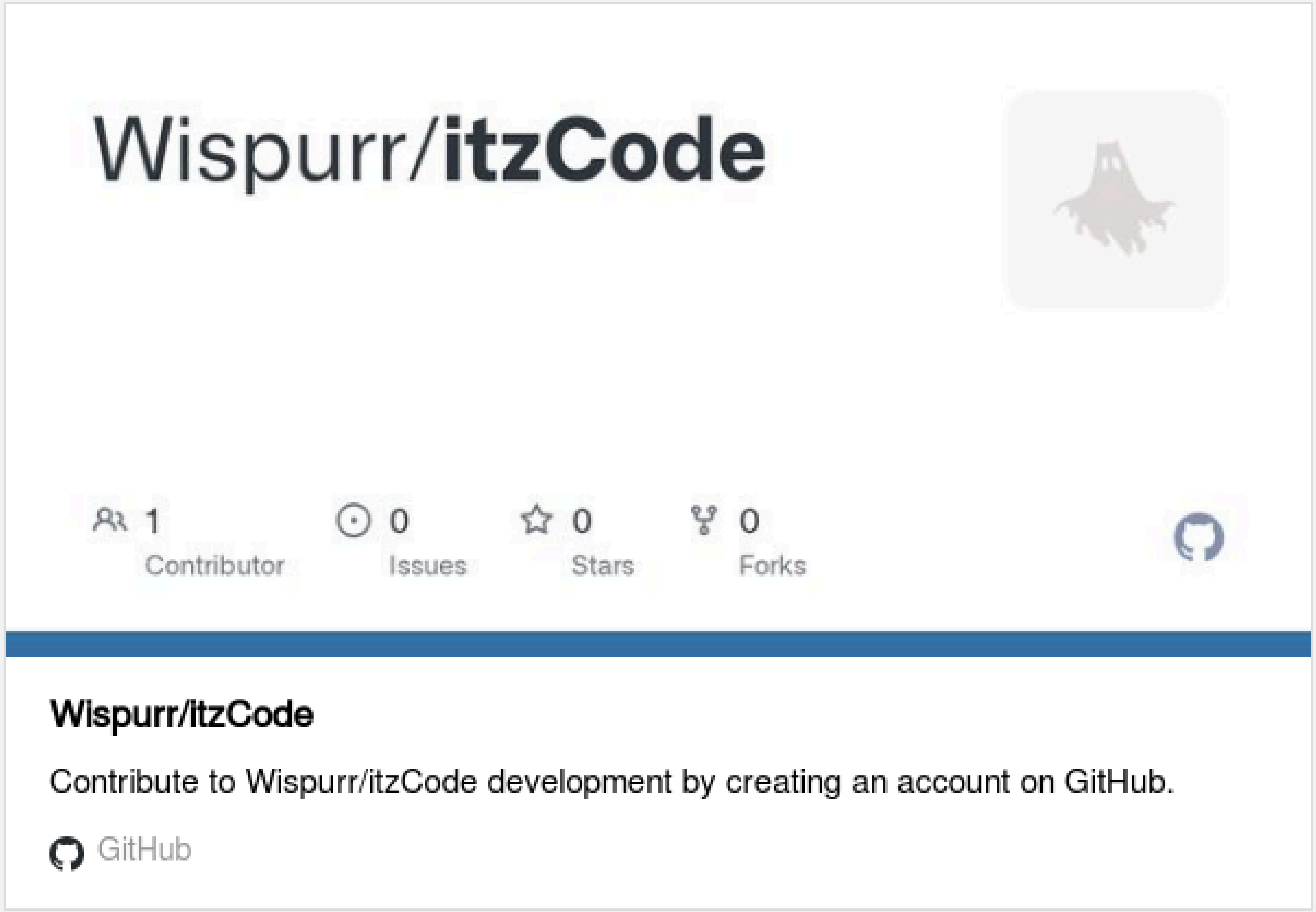
Q3: C++ Macro Conflicts

Problem: #define endl '\n' + transpiled \n strings → g++ compile error.

Initially used #define endl '\n' for optimization. However, when transpiling python strings containing \n to C++, this caused a "missing terminating character" error in G++.

Solution: Remove macro; use standard libs and std::endl.

Removed unstable macros and switched to standard C++ libraries (<iostream>, <fstream>) and std::endl to ensure cross-platform compatibility and stability.



Wispurr/**itzCode**

1 Contributor

0 Issues

0 Stars

0 Forks

Wispurr/itzCode

Contribute to Wispurr/itzCode development by creating an account on GitHub.

GitHub

`</slide>`