X0 Test Report

Yingzhe Lyu, 10152130255

November 30, 2018

1 Introduction

1.1 Test Description

This report is for my X0 compiler as the course project in Compiler Principle Practice. We did thorough tests for the compiler, and we only choose a part of them here which covers the major functionality of X0 language and can serve as representatives. The test $1-12$ are unit tests focused on specific functionality and test cases $13-15$ are advanced ones which combine a lot of features and complete some meaningful works, and specially, test 6 is the first obligatory test case and test 13 is the second one.

1.2 Test Environment

OS	Microsoft Windows 10 Professional 64-bit
Development Tools	Bison 2.4.1 & Flex 2.5.4a-1 & make for Windows 3.8.1
C Environment	TDM GCC 5.1.0-3 for Windows x86-64
Python Environment	Anaconda3 5.1.0 (Python 3.6.4) for Windows x86-64

2 Test Cases

2.1 Test01

Source File	`Test01_operators.x0`
Subject	Test all kinds of operators (except increment and decrement)
Expected Result	Correct operation result
Validation	Test passed

2.2 Test02

Source File	`Test02_nested.x0`
Subject	Test nested loop (two layers) with break/continue flow control
Expected Result	Output correct graph pattern
Validation	Test passed

2.3 Test03

Source File	`Test03_break.x0`
Subject	Test multiple break/continue statements in a multi-layer loop, note that this program contains infinite loop and should never be run. The only thing need to do is compile and observe the intermediate code.
Expected Result	Generate intermediate code with correct jump addresses
Validation	Test passed

2.4 Test04

Source File	`Test04_func.x0`
Subject	Test function declaration and recursive function call. This test case is actually a recursive Fibonacci sequence calculation. Please note that recursive calculation of this sequence is time and space consuming, you'd better only input numbers less than 20, or the stack may be overflowed.
Expected Result	Output correct n-th (user designated) Fibonacci term repetitively, terminate after an negative input.
Validation	Test passed

2.5 Test05

Source File	`Test05_array.x0`
Subject	Test declaration and reference of array. This test case is actually a linear Fibonacci sequence calculation (usually called memorize strategy in ACM-ICPC contests).
Expected Result	Output correct Fibonacci terms, from the second term to 49th term.
Validation	Test passed

2.6 Test06

Source File	`Test06_globals.x0`
Subject	Test the declaration and reference of constant variable and global variables. This test case is actually the **Sieve of Eratosthenes algorithm** of calculating primes, and is also **the first obligatory test program**.
Expected Result	Output all prime numbers in range $1 - 100$, and output the count of primes in this range.
Validation	Test passed

2.7 Test07

Source File	`Test07_dangling_else.x0`
Subject	Test our solution of **dangling else problem**.
Expected Result	The expected result is given in source file comment.
Validation	Test passed

2.8 Test08

Source File	`Test08_incre_decre.x0`
Subject	Test the prefix and postfix self increment and decrement operators.
Expected Result	The return value for `a++` is the old value of `a`, and for `++a` is `a+1`. Therefore, the expected results are $a = 1, b = 2$ $a = 3, b = 3$ respectively.
Validation	Test passed

2.9 Test09

Source File	`Test09_do_while_repeat_until.x0`
Subject	Test the do-while and repeat-until loop structures.
Expected Result	The first line of output is 54321, and the second line is 0123.
Validation	Test passed

2.10 Test10

Source File	`Test10_complex_array_access.x0`
Subject	Test some really complex array declaration and access.
Expected Result	The expected result is 12.
Validation	Test passed

2.11 Test11

Source File	`Test11_types.x0`
Subject	Test the declaration of variables in different types, and automatic and manual type conversion.
Expected Result	The expected type conversion behavior just like in C++ programming language.
Validation	Test passed

2.12 Test12

Source File	`Test12_switch.x0`
Subject	Test the switch statement.
Expected Result	Expect correct case choices. Since we used the back-patch-in-time strategy, multiple same case statements and nested switch statements are allowed in our implementation, and additional tests on these features also passed.
Validation	Test passed

2.13 Test13

Source File	`Test13_gcd.x0`
Subject	This is an advanced test case, tested function and loop functionalities together. Actually, it's a **recurrent GCD calculation** and also **the second obligatory test case**.
Expected Result	Repetitively read two numbers and output their LCM (calculated based on their GCD), terminates when the first number is negative.
Validation	Test passed

2.14 Test14

Source File	`Test14_fst_matrix_mul.x0`
Subject	This is an advanced test case, tested multi-dimension arrays, multiple functions and global variables. Actually it is the calculation of n-th Fibonacci term with **Fast Matrix Multiplication algorithm** which can solve it in $O(\log(n))$ time and $O(1)$ space. Please also note the loop condition in main function, which illustrate that our read feature also have a return value same to the number read in.
Expected Result	Output correct n-th (user designated) Fibonacci term repetitively, terminate after an negative input.
Validation	Test passed

2.15 Test15

Source File	`Test15_quick_sort.x0`
Subject	This is an advanced test case, tested global variables and arrays, loops, recursive function calls among multiple functions. Actually it implemented the **Quick Sort algorithm**, which can sort n elements in $\Theta(n \log n)$ time.
Expected Result	Input a positive number n less than 1010, and then input n numbers, the program will output the n numbers in ascending order.
Validation	Test passed