Lab08 Report

崔士强 PB22151743

January 14, 2024

1 Purpose

The purpose of the program is to implement Lab01 through Lab04 using C++.

Anticipated outcomes: Consistent with outcomes in corresponding lab.

2 Principles

Approaches to solve the problems have been discussed in previous lab reports. So here we only focus on the high-level language implementation of those approches.

2.1 Lab01 Counting Zero

Initialize an integer `singleBit` with 1. Then obtain a number with only the $(n+1)^{th}$ bit being 1 in binary representation. Perform bit-wise AND with the given number to get the information on the corresponding bit.

Relevant code:

Listing 1: Lab01

```
int16_t lab1(int16_t n) {
    // initialize
    int16_t singleBit = 1;
    int count = 0;

    // calculation
    if((n & 1) == 0)   n = -n;    // odd number
    for(int i = 0; i < 16; i++){
        count = count + !(n & singleBit);
        singleBit = singleBit + singleBit;
    }

    // return value
    return count + STUDENT_ID_LAST_DIGIT;
}
```

2.2 Lab02 The PingPong Sequence

Use an indicator `operator_d_n` to indicate the current operator(i.e. d_n). Check for divisibility by 8 and the last digit by repeatedly substracting 8 and 10 respectively, as in Lab02.

Relevant code:

Listing 2: Lab02

```
int16_t lab2(int16_t n) {
   // initialize
   int16_t result = 3;
   int operator_d_n = 1; // 1 for +, -1 for -

   // calculation
   for( ; n>1; n--){
      if(operator_d_n == 1) result = result + result + 2;
      else   result = result + result - 2;
      result = result & 0x0FFF;
      if(Check4LastDigit(result)||Check4Divisibility(result)) operator_d_n =
          -operator_d_n;
   }

   // return value
   return result;
}

bool Check4LastDigit(int16_t result){
   while(result>0) result = result - 10;
   return result == -2;
}

bool Check4Divisibility(int16_t result){
   while(result>0) result = result - 8;
   return result == 0;
}
```

2.3 Lab03 String Compare

Use a loop to check the characters at the same position in two given strings.

Relevant code:

Listing 3: Lab03

```
int16_t lab3(char s1[], char s2[]) {
   // initialize
   int i = 0;

   // calculation
   while(s1[i] == s2[i]){
        if(s1[i++] == '\0') return 0;
    }

    // return value
    return s1[i]-s2[i];
}
```

2.4 Lab04 Baguenaudier

Implement two functions `void Put()` and `void Remove()`. These two functions recursively call themselves according to the rule.

3 Procedure

3.1 Bugs encountered

In the implementation of Lab02, the program ran into an infinite loop. The problem was that condition for a `while` loop was not set correctly:

```
bool Check4LastDigit(int16_t result){
    while(result) result = result - 10;
    return result == -2;
}
```

In this case the loop would continue after `result` reaches nagative.

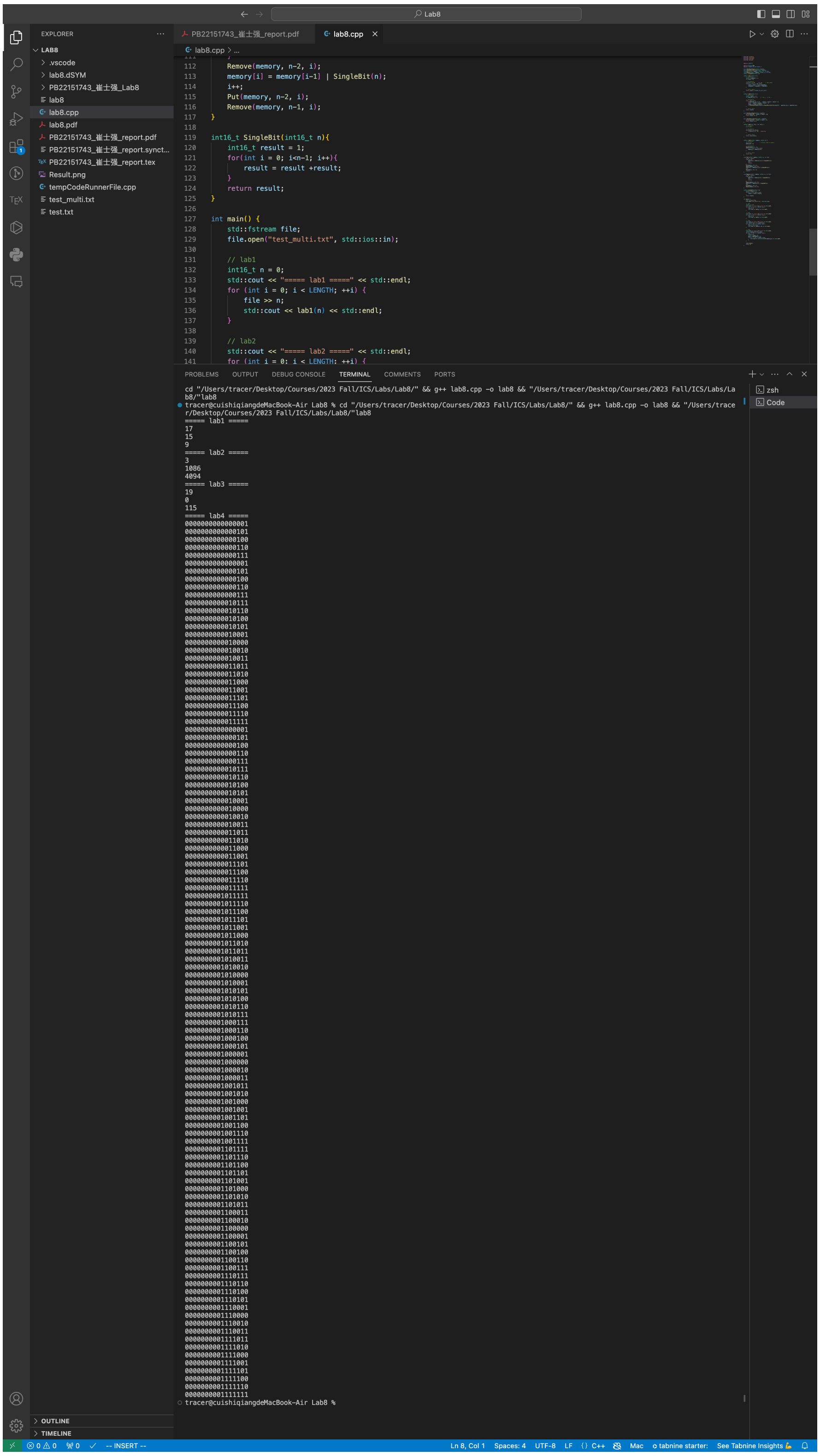
Correction: Change `result` to `result>0`.

4 Results

Testcases are as follows:

```
 5
 15
 6280
 1
 15
 24
 zfz gfg
 bfb bfb
 DsTAs DsTA
 3
 5
 7
```

Results:



```cpp
    Remove(memory, n-2, i);
    memory[i] = memory[i-1] | SingleBit(n);
    i++;
    Put(memory, n-2, i);
    Remove(memory, n-1, i);
}

int16_t SingleBit(int16_t n){
    int16_t result = 1;
    for(int i = 0; i<n-1; i++){
        result = result +result;
    }
    return result;
}

int main() {
    std::fstream file;
    file.open("test_multi.txt", std::ios::in);

    // lab1
    int16_t n = 0;
    std::cout << "===== lab1 =====" << std::endl;
    for (int i = 0; i < LENGTH; ++i) {
        file >> n;
        std::cout << lab1(n) << std::endl;
    }

    // lab2
    std::cout << "===== lab2 =====" << std::endl;
    for (int i = 0; i < LENGTH; ++i) {
```

```
cd "/Users/tracer/Desktop/Courses/2023 Fall/ICS/Labs/Lab8/" && g++ lab8.cpp -o lab8 && "/Users/tracer/Desktop/Courses/2023 Fall/ICS/Labs/La
b8/"lab8
tracer@cuishiqiangdeMacBook-Air Lab8 % cd "/Users/tracer/Desktop/Courses/2023 Fall/ICS/Labs/Lab8/" && g++ lab8.cpp -o lab8 && "/Users/trace
r/Desktop/Courses/2023 Fall/ICS/Labs/Lab8/"lab8
===== lab1 =====
17
15
9
===== lab2 =====
3
1086
4094
===== lab3 =====
19
0
115
===== lab4 =====
0000000000000001
0000000000000101
0000000000000100
0000000000000110
0000000000000111
0000000000000001
0000000000000101
0000000000000100
0000000000000110
0000000000000111
0000000000010111
0000000000010110
0000000000010100
0000000000010101
0000000000010001
0000000000010000
0000000000010010
0000000000010011
0000000000011011
0000000000011010
0000000000011000
0000000000011001
0000000000011101
0000000000011100
0000000000011110
0000000000011111
0000000000000001
0000000000000101
0000000000000100
0000000000000110
0000000000000111
0000000000010111
0000000000010110
0000000000010100
0000000000010101
0000000000010001
0000000000010000
0000000000010010
0000000000010011
0000000000011011
0000000000011010
0000000000011000
0000000000011001
0000000000011101
0000000000011100
0000000000011110
0000000000011111
0000000001011111
0000000001011110
0000000001011100
0000000001011101
0000000001011001
0000000001011000
0000000001011010
0000000001011011
0000000001010011
0000000001010010
0000000001010000
0000000001010001
0000000001010101
0000000001010100
0000000001010110
0000000001010111
0000000001000111
0000000001000110
0000000001000100
0000000001000101
0000000001000001
0000000001000000
0000000001000010
0000000001000011
0000000001001011
0000000001001010
0000000001001000
0000000001001001
0000000001001101
0000000001001100
0000000001001110
0000000001001111
0000000001101111
0000000001101110
0000000001101100
0000000001101101
0000000001101001
0000000001101000
0000000001101010
0000000001101011
0000000001100011
0000000001100010
0000000001100000
0000000001100001
0000000001100101
0000000001100100
0000000001100110
0000000001100111
0000000001110111
0000000001110110
0000000001110100
0000000001110101
0000000001110001
0000000001110000
0000000001110010
0000000001110011
0000000001111011
0000000001111010
0000000001111000
0000000001111001
0000000001111101
0000000001111100
0000000001111110
0000000001111111
tracer@cuishiqiangdeMacBook-Air Lab8 %
```

Figure 1: Result