lab 6 report

PB21081601 张芷苒

实验要求

- 使用 C/C++ 复现 lab1-4
- 考虑到 lc-3 不支持 *, /, %, >>, << 运算，所以只允许使用 +, −, = , ++, −−, ==, ! =, <, >, <=, >=, &, |
- 可以使用 for , while , do while , if , continue , break , switch case

实验原理

lab 1

计算一个 16 位二进制数 a 的低 b 位的个数，依次将 a 的低 b 位和 1 进行与操作，来判断该位是否为 1。

伪代码：

```
weight = 0;
judge = 0;
for (i = 0; i < b; ++i) {
    temp = a & judge;
    judge += judge;
    if (temp != 0)
        ++weight;
}
return weight;
```

实现：

```
int16_t lab1(int16_t a, int16_t b)
{
    // initialize
    int16_t ans = 0;
    int16_t judge = 1;
    // calculation
    for (int i = 0; i < b; ++i)
    {
        int16_t temp = a & judge;
        judge += judge;
```

```
        if (temp != 0)
        {
            ++ans;
        }
    }
    // return value
    return ans;
}
```

lab 2

计算

$$F(0) = F(1) = 1$$

$$F(N) = F(N-2)\%p + F(N-1)\%q \ \ (2 \leq N \leq 1024)$$

$$p = 2^k \ \ (2 \leq k \leq 10)$$

采用滑动窗口的方法来求解该问题。

伪代码：

```
num1 = 1;
num2 = 1;
for (n = n - 1; n > 0; --n) {
    temp1 = (p - 1) & num1;
    temp2 = num2;
    while (temp2 > 0) {
        temp2 -= q;
    }
    temp2 += q;
    F = temp1 + temp2;
    num1 = num2;
    num2 = F;
}
renturn F;
```

实现：

```
int16_t lab2(int16_t p, int16_t q, int16_t n)
{
    // initialize
    int16_t num1 = 1;
    int16_t num2 = 1;
    int16_t ans;
    // calculation
    for(int16_t N = n - 1; N > 0; --N)
```

```
    {
        int16_t temp1 = (p - 1) & num1;
        int16_t temp2 = num2;
        while(temp2 >= 0)
        {
            temp2 -= q;
        }
        temp2 += q;
        ans = temp1 + temp2;
        num1 = num2;
        num2 = ans;
    }
    // return value
    return ans;
}
```

lab 3

实现求字符串的最大重复子串。 通过一次遍历即可求解该问题，记录此时最大重复子串的长度，当出现新的重复子串时，更新最大值。

伪代码：

```
right = str[0];
left = right;
max_len = 1;
temp = 1;
for (i = 1; i < N; ++i) {
    right = str[i];
    i += 1;
    if (left == right)
        temp += 1;
    else {
        if (max_len < temp)
            max_len = temp;
        temp = 1;
    }
    left = right;
}
if (max_len < temp) {
    max_len = temp;
}
return max_len;
```

实现：

```
int16_t lab3(int16_t n, char s[])
```

```
{
    // initialize
    int16_t right = s[0];
    int16_t left = s[0];
    int16_t ans = 1;
    int16_t temp = 1;
    // calculation
    for(int16_t i = 1; i < n; ++i)
    {
        right = s[i];
        if(left == right)   ++temp;
        else
        {
            if(ans < temp)  ans = temp;
            temp = 1;
        }
        left = right;
    }
    if(ans < temp)  ans = temp;
    // return value
    return ans;
}
```

lab 4

对 16 个人的成绩的升序排列，并求出这 16 个人中获得评级 A，B 的数量。为了提高代码效率，在排序的同时计算评级 A，B 的数量，从而需要获得降序的排名，而题目要求将成绩升序排列，故如果采用逆序的降序排列，既可以得到正序的升序排列，又可以在排序是计算评级 A，B 的数量。

伪代码：

```
numa = 0;
numb = 0;
for (i = 15; i >= 0; --i) {
    max_index = i;
    for (j = i - 1; j >= 0; --j) {
        if (scores[max_index] < scores[j])
            max_index = j;
    }
    if (i != max_index)
        swap (scores[max_index], scores[i]);
    if (score[i] >= 85 && numa < 4)
        ++numa;
    else if (score[i] >= 85 && numa + numb < 8)
        ++numb;
    else if (score[i] >= 75 && numa + numb < 8)
        ++numb;
```

```
}
return numa, numb;
```

实现：

```
void lab4(int16_t score[], int16_t *a, int16_t *b)
{
    // initialize
    int16_t numa = 0;
    int16_t numb = 0;
    // calculation
    for(int i = 15; i >= 0; --i)
    {
        int16_t max_index = i;
        for(int j = i - 1; j >= 0; --j)
        {
            if(score[max_index] < score[j]) max_index = j;
        }
        if(i != max_index)
        {
            int16_t temp = score[max_index];
            score[max_index] = score[i];
            score[i] = temp;
        }
        if(score[i] >= 85 && numa < 4)  ++numa;
        else if(score[i] >= 85 && numa + numb < 8)  ++numb;
        else if(score[i] >= 75 && numa + numb < 8)  ++numb;
    }
    *a = numa;
    *b = numb;
    // return value
    return;
}
```

实验结果

经检验，符合预期。测试文件的名字需改为 `test.txt` .

实验思考

- 高级程序设计语言与 lc-3 汇编语言编程的区别
    - 高级语言可读性、可维护性较佳、代码简洁，lc-3 汇编语言的可 读性较差、代码繁琐。
    - lc-3 汇编语言程序的占用空间小，执行速度快，执行效率高，高级语言占用的空间大，执行效率较低。

- 你认为 lc-3 汇编语言需要添加哪些指令
    - 可以添加取余相关的指令
- 对于使用的高级语言，是否需要从 lc-3 中学到什么
    - 通过使用 lc-3 进行面向寄存器的编程练习，提升了对程序执行时底层原理的理解，可以使我们在后续使用高级语言编程时，更好的提成程序执行效率。