中国科学技术大学计算机学院
《计算机系统概论》实验报告

LAB 06: C Programming

姓名:王晨 学号:PE20060014

完成日期:2021 年 1 月 13 日

一、实验要求：

1. Implement lab02 to lab05 again but in C language.

2. Think about and write down the difference between low-level programming language and high-level programming language.

二、实验环境：

Mac OS Big Sur

Clion 2020.3 version

注意：本实验代码中由于Mac OS没有<conio.h>以及一些sleep/delay库函数，lab2～lab4由C编写，可以在windows环境下正常运行，lab5函数如果编译失败，请在unix/linux C++下编译运行。

三、How to Use my program：

A. Lab02

Lab02采用辗转相除法递归实现求两个正整数的最大公因数。代码非常简单，依次输入两个正整数即可得到结果：

```
/***--------------Lab02 Greatest Common Divisor--------------***/
请输入两个待求GCD的正整数:
27 6318
6318和27的最大公因数为: 27
```

B. Lab03

```
/***--------------Lab03 The Linked-List Sort--------------***/
请输入链表的长度:
8
请依次该链表8个结点的值:
-2 8 0 9 17 -30 22 30
升序排列后的链表为:
节点-(*addr)=value :  (*7FCABE8040F0)=-30
节点-(*addr)=value :  (*7FCABE8040E0)=-2
节点-(*addr)=value :  (*7FCABE8040D0)=0
节点-(*addr)=value :  (*7FCABE8040C0)=8
节点-(*addr)=value :  (*7FCABE8040B0)=9
节点-(*addr)=value :  (*7FCABE8040A0)=17
节点-(*addr)=value :  (*7FCABE804090)=22
节点-(*addr)=value :  (*7FCABE804080)=30
```

Lab03定义了一些链表静态结构和ADT，包括（头插法）插入、遍历打印、（冒泡）排序。程序运行时首先要求用户输入链表长度和各个结点的data值，用空格或回车隔开，输入完毕后会得到升序排列后的结果，见上图。

C. Lab04

Lab04是NIM游戏，用几个全局变量保存当前游戏状态，ADT只有两个：1.printBoard，根据当前的状态打印棋盘。2.checkInput，判断玩家输入的行号和数字是否合法，如果合法会更新棋盘状态。Player的切换通过 Player%2+1实现。

```
/***--------------Lab04 The Game of Nim--------------***/
int RowA=3,RowB=5,RowC=8;                          //全局变量 保存当前某一行还剩下的rock数量
int Player=1;                                      //当前的Player, 1=Player1, 2=Player2
```

两层do-while循环完成游戏功能，循环出口是ABC行的rock数量均为0。

```
void lab04(){
    printf("\n/***--------------Lab04 The Game of Nim--------------***/\n");
    printf("/***Please press <ENTER> when you finish input Row and Num***/ \n");
    do{
        char input_Row = '#';          //每次更换玩家必须重置上一玩家输入的行号和数字
        int  input_Num = 0;
        printBoard(RowA,RowB,RowC);
        printPlayer();
        do {
            input_Row = getchar();                  //use getchar() to obtain only one single character
            scanf("%d", &input_Num);
            fflush(stdin);                          //清空缓冲区
        }while(1==checkInput(input_Row,input_Num));  //输入非法值，则返回重新输入
    } while (0!=RowA || 0!=RowB || 0!=RowC);
    if(Player==1)
        printf("\nPlayer 1 Wins.\n");
    else
        printf("\nPlayer 2 Wins.\n");
}
```

游戏运行时的输入和输出同example.txt的测试集完全相同:

```
ROW A: oo
ROW B:
ROW C:
Player 2, choose a row and number of rocks:A1

ROW A: o
ROW B:
ROW C:
Player 1, choose a row and number of rocks:A*
Invalid move. Try again.
Player 1, choose a row and number of rocks:$4
Invalid move. Try again.
Player 1, choose a row and number of rocks:A1

Player 2 Wins.
```

D. Lab05

Lab05是基于键盘响应函数kbhit（）实现的（伪）中断，由于Mac OS没有相应的库，只能自己用C++实现kbhit（）。

```
//键盘响应子函数，MacOS不支持conio.h，需要自己实现
int _kbhit(){
    static const int STDIN = 0;
    static bool initialized = false;
    if (! initialized){
        // Use termios to turn off line buffering
        termios term;
        tcgetattr(STDIN, &term);
        term.c_lflag &= ~ICANON;
        tcsetattr(STDIN, TCSANOW, &term);
        setbuf(stdin, NULL);
        initialized = true;
    }
    int bytesWaiting;
    ioctl(STDIN, FIONREAD, &bytesWaiting);
    return bytesWaiting;
}
```

实现kbhit（）后只需要接收键盘输入作相应输出即可。本程序在输入字符后需要输入回车，输入q退出循环：

```
/***--------------Lab05 Interrupt a Running Program--------------***/
/***-------------Please press <ENTER> when you finish input------------***/
/***-------------Please press <q> to quit loop-------------***/
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
2

2 is a decimal number.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
w

w is not a decimal number.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
q

-------------quit loop-------------
```

四、实验心得：

由于实验代码比较多，就不详细阐述每个lab的c语言实现细节了，具体可以看.c文件。

从机器语言——汇编——C/C++可以看出低级语言到高级语言有以下差异：

1. 可读性：使用高级语言编写的代码一目了然，因为其更贴近我们的语言风格。而低级语言往往比较抽象，尤其是机器语言，如果不写注释对照指令集说明，则几乎无法看懂。
2. 高效性：低级语言使用指令集直接操作CPU，可以直接控制硬件操作，因此效率更高，速度更快。而高级语言在编译转换时，基本都会产生冗余机器码，降低指令效率。而且在实际编写中，使用高级语言也会自然而然的编写一些实际无用的代码。不过C语言已经几乎非常贴近硬件底层了，尤其是在嵌入式系统设计中，对于硬件的控制开发很多时候会采用汇编/C混合编程以平衡运行效率和开发效率。
3. 封装性：高级语言编写的代码对大部分复杂功能的开发相对低级语言更加简单简洁，这是由于高级语言提供的封装特性，比如在lab02～lab05中，用c语言编写的代码可以直接调用库函数，对于自己实现的代码也只需要分割成几个ADT，然后在主程序中组装起来即可，这种特性虽然低级语言也可以实现，但是会复杂的多，比如寄存器的调用保存，需要开发者去关心这些细节。高级语言的开发者并不需要了解底层的实现细节，只需要了解接口方式，因为这些事编译器都会帮我们完成，这种层层封装的模块化编程更加符合开发习惯。
4. 移植性：高级语言有相应的标准和规范，且与硬件结构和指令系统无关，因此同一个程序可以在不同的机器（平台）上编译运行。而低级语言机器语言是对特定的机器编制的，所以较难移植。

总而言之，对于一项任务来说，用高级语言和低级语言，只要指令集支持，都可以实现。机器做的少了，那么人就做得多了。低级语言到高级语言的过程就是层层封装的过程，高级语言最终也要翻译成机器语言执行。这种封装性使得高级语言多了看似低级语言所没有的很多语法、数据类型等等，例如for , do-while，int ， double ， char数据类型，这些虽然低级语言没有，但是本质上都是可以通过低级语言来实现的。