LAB 06

梁峻滔 PB19051175

1. 算法思想

基本都是根据之前各个实验所使用的算法，用 C 的形式实现出来。实验过程中，高级语言(C)与低级语言(LC-3 汇编语言和机器码)的主要区别在于：

(1)高级语言(C)的循环、条件选择等控制结构的编写、复杂算术运算、输出字符串、编写和调用子程序等都比低级语言方便，且编写高级语言时不需要考虑寄存器(有限)的分配使用和保存，。

(2)在数据类型方面，高级语言的数据类型更加丰富，在 LC-3 的汇编语言和机器码中的数据类型只有一种——16 bit 的二进制补码。一个长为 16 的 0-1 串，在 C 中可以用一个 short 型变量来表示，也可以用 char、int 型的数组来存，用数组来存时就不能直接进行算术运算，要先进行转换。

(3)低级语言的最小执行单元是指令，高级语言的是一条语句(可能相当于多条指令的功能)。高级语言的运算符和能直接进行的操作更加丰富，同时语法限制也更加严格。

(4)高级语言相比于低级语言更接近自然语言，可读性更好。

2. 代码实现和测试

(1) lab01

```
Status lab01(char word[word_len]){
    char temp;
    int rotamount;
    printf("\nInput rotate amount: ");
    scanf("%d",&rotamount);
    //rotation
    for(int j=0;j<rotamount;++j){
        temp = word[0];
        for(int i=0;i<word_len-2;++i) word[i] = word[i+1];
        word[word_len-2] = temp;
    }
    printf("\nAfter rotation: ");
    printf("%s",word);
    printf("\n");
    return OK;
}
```

main 函数设置：

```
//test lab01
char word[word_len] = "1101000100001000";
lab01(word);
```

终端运行结果：正确运行

```
Input rotate amount: 2

After rotation: 0100010000100011
```

(2) lab02

```
Status lab02(char word1[word_len],char word2[word_len]){
    //输入两个16 bit二进制补码表示的正整数，求出它们的最大公约数并打印显示
    char gcd[word_len];   //最大公约数的16 bit二进制补码表示
    int factor1=0,factor2=0;
    //将char表示的二进制数转化成int十进制数运算
    for(int i=0;i<word_len-1;++i){
        if(word1[i]!='0') factor1+=pow(2,15-i);
        if(word2[i]!='0') factor2+=pow(2,15-i);
    }
    int temp1=factor1, temp2=factor2;
    //辗转相减
    while(factor2!=factor1){
        if(factor1 > factor2) factor1 = factor1 - factor2;
        else factor2 = factor2 - factor1;
    }
    //将最大公约数转化从16 bit二进制补码表示
    for(int i=0;i<word_len-1;++i)
       if(factor1 >= pow(2,15-i)) {gcd[i] = '1'; factor1 = factor1 - pow(2,15-i);}
       else gcd[i] = '0';
    gcd[word_len-1] = '\0';
    int result = (factor1==0) ? factor2 : factor1;
    printf("The GCD of %s(%d) and %s(%d) is %s(%d)\n",word1,temp1,word2,temp2,gcd,result);
    return OK;
}
```

main 函数设置：

```
char word1[word_len] = "0000000001100010";
char word2[word_len] = "0000000000111111";
lab02(word1,word2);
```

终端运行结果：正确运行

```
The GCD of 0000000001100010(98) and 0000000000111111(63) is 0000000000000111(7)
```

(3) lab03

```
//Lab03需要用到的结点结构
typedef struct CPMNode{
    char CPM_int[word_len];
    struct CPMNode *NEXT;
}CPMNode;
```

```
63  Status lab03(CPMNode *Head){
64      srand(time(NULL));
65      printf("\nAssign the length of the linked list: ");
66      int list_len;
67      scanf("%d",&list_len);
68      CPMNode *p = NULL, *q = NULL, *r = NULL;
69      int temp;
70      //指定链表长度后，每个结点用随机生成的整数作为数据域的值来构造链表
71      for(int i=0;i<list_len;++i){
72          p = (CPMNode*)malloc(sizeof(CPMNode));
73          if(!p) exit(0);
74          p->NEXT = NULL;
75          temp = rand() % 16383;
76          for(int j=0;j<word_len-1;++j)
77              if(temp > pow(2,15-j)) {p->CPM_int[j]='1';temp = temp -pow(2,15-j);}
78              else p->CPM_int[j] = '0';
79          p->CPM_int[word_len-1] = '\0';
80          p->NEXT = Head;
81          Head = p;
82      }
```

```
    //选择排序
    p = Head;
    while(p){
        q=r=p;
        int min;
        temp=0;
        //将0-1串转换成int型
        for(int j=0;j<word_len-1;++j)
            if(p->CPM_int[j]=='1') temp += pow(2,15-j);
        min = temp;
        //选出未排序的最小值，用r指向该结点
        while(q){
            temp=0;
            for(int j=0;j<word_len-1;++j)
                if(q->CPM_int[j]=='1') temp += pow(2,15-j);
            if(temp < min) {min = temp;r=q;}
            q = q->NEXT;
        }
        //交换当前p和r所指的结点的数据域
        char exchange[word_len];
        for(int k=0;k<word_len-1;++k) exchange[k] = p->CPM_int[k];
        for(int k=0;k<word_len-1;++k) p->CPM_int[k] = r->CPM_int[k];
        for(int k=0;k<word_len-1;++k) r->CPM_int[k] = exchange[k];
        p = p->NEXT;
    }
```

```
    printf("The address and value of the list is as follows:\n");
    p = Head;
    while(p){
        printf("address:%d,value:%s,points to:%d\n",p,p->CPM_int,p->NEXT);
        p = p->NEXT;
    }
    return OK;
}
```

main 函数设置：

```
struct CPMNode *Head=NULL;
lab03(Head);
```

终端运行结果：正确运行

```
Assign the length of the linked list: 6
The address and value of the list is as follows:
address:1783032,value:0000000011101110,points to:1783000
address:1783000,value:0000011101010000,points to:1782968
address:1782968,value:0000110001100110,points to:1775352
address:1775352,value:0000110001111001,points to:1775320
address:1775320,value:0011000000110110,points to:1775272
address:1775272,value:0011111000011011,points to:0
```

(4) lab04

两个子函数：

```
117  Status PrintRows(int *Row){
118      int i,j;
119      for(i=0;i<Row_Num;++i){
120          switch(i){
121              case 0: {printf("ROW A: ");break;}
122              case 1: {printf("ROW B: ");break;}
123              case 2: {printf("ROW C: ");break;}
124          }
125          for(j=0;j<Row[i];++j) printf("o ");
126          printf("\n");
127      }
128      return OK;
129  }
```

```
131  int RemoveRocks(int Player, int *Row){
132      int winner=0;
133      if(Player>2 || Player<1) return ERROR;
134      int i;
135      for(int i=0;i<Row_Num;i++)
136          if(Row[i]!=0) break;
137      if(i==Row_Num) return ERROR;
138      printf("Player%d choose a row and number of rocks: ",Player);   //提示玩家输入
139      char row[2];
140      scanf("%s",row);
141      //判断输入是否有效
142      while(row[0]<'A' || row[0]>'C' || row[1]<'1' || row[1]-48>Row[row[0]-65]){
143          printf("Invalid. Try again.\n");
144          scanf("%s",row);
145      }
146      Row[row[0]-65]=Row[row[0]-65]-(row[1]-48);   //修改石头数
147      //判断游戏是否结束和结束时的赢家
148      for(i=0;i<Row_Num;i++)
149          if(Row[i]!=0) break;
150      if(i==Row_Num) {winner=Player==1? 2 : 1; printf("Player%d wins.",winner);}
151      return winner;
152  }
```

主程序函数：

```
154  Status lab04(int *Row){
155      PrintRows(Row);
156      int winner,player=1;
157      winner = RemoveRocks(player,Row);
158      while(!winner){
159          printf("winner=%d\n",winner);
160          PrintRows(Row);
161          player = (player==1)? 2:1;
162          printf("player=%d\n",player);
163          winner=RemoveRocks(player,Row);
164      }
165  }
```

main 函数设置：

```
//test lab04
int Row[Row_Num];
Row[0]=3; Row[1]=5;Row[2]=8;
lab04(Row);
```

终端运行结果：与 Example 一致，正确运行

```
ROW A: o o o
ROW B: o o o o o
ROW C: o o o o o o o o
Player1 choose a row and number of rocks: B2
winner=0
ROW A: o o o
ROW B: o o o
ROW C: o o o o o o o o
player=2
Player2 choose a row and number of rocks: A1
winner=0
ROW A: o o
ROW B: o o o
ROW C: o o o o o o o o
player=1
Player1 choose a row and number of rocks: C6
winner=0
ROW A: o o
ROW B: o o o
ROW C: o o
player=2
Player2 choose a row and number of rocks: G1
Invalid. Try again.
B3
winner=0
ROW A: o o
ROW B:
ROW C: o o
player=1
Player1 choose a row and number of rocks: A3
Invalid. Try again.
C2
winner=0
ROW A: o o
ROW B:
ROW C:
player=2
Player2 choose a row and number of rocks: A1
winner=0
```

```
ROW A: o
ROW B:
ROW C:
player=1
Player1 choose a row and number of rocks: A*
Invalid. Try again.
&4
Invalid. Try again.
A1
Player2 wins.
```

(5) lab05

```
167  Status lab05(char *str){
168      char ch;
169      while(1){
170          printf("%s\n",str);
171          for(int i=0;i<500000000;++i);
172          //当从键盘输入字符时，中断当前循环输出程序，判断输入的字符是否十进制数字
173          if(kbhit()){
174              ch = getch();
175              if(ch<'0' || ch>'9') printf("%c is not a decimal digit.\n",ch);
176              else printf("%c is a decimal digit.\n",ch);
177          }
178      }
179      return OK;
180  }
```

main 函数设置：

```
//test lab05
char *str = "ICS2020 ";
lab05(str);
```

终端运行结果：正确运行

```
ICS2020
ICS2020
ICS2020
ICS2020
a is not a decimal digit.
ICS2020
ICS2020
1 is a decimal digit.
ICS2020
ICS2020
ICS2020
0 is a decimal digit.
ICS2020
ICS2020
% is not a decimal digit.
ICS2020
ICS2020
b is not a decimal digit.
ICS2020
ICS2020
$ is not a decimal digit.
ICS2020
ICS2020
ICS2020
7 is a decimal digit.
ICS2020
ICS2020
ICS2020
m is not a decimal digit.
ICS2020
```