ICS-LAB6 REPORT

PB20000096 潘廷岳

Part1.代码及结果展示

lab0l:

```
int main()
{
    short int r0,r1;
    scanf("%hd%hd",&r0,&r1);
    bool flag=r0<0? 1:0;
    if(r0<0) r0=-r0;
    short int _r1=r1;
    while(--r0) r1+=_r1;
    r1 = (flag) ? -r1:r1;
    printf("%hd",r1);
    return 0;
}
```

设计思路：将r1加r0遍。

lab0p:

```
int main()
{
    short int r0,r1,r7,flag;
    cin>>r0>>r1;
    if(r0==0) {
        cout<<0;return 0;
    }
    r7=0;
    flag=(r0>=0)?0:1;
    if(flag) r0=-r0;
    while(r0){
        if(r0%2) r7+=r1;
        if(r0!=1) r1<<=1;
        r0>>=1;
    }
    if(flag) r7=-r7;
    cout<<r7;
    return 0;
}
```

设计思路：模仿快速幂求n次幂。

二者运行结果：

```
PS C:\Users\pty\Desktop\lab6> ./lab0l.exe
43 43
1849
PS C:\Users\pty\Desktop\lab6> ./lab0p.exe
43 43
1849
PS C:\Users\pty\Desktop\lab6>
```

fib:

```
int main()
{
    short int Mod=1024;
    short int r0=1,r1=1,r7=2,n;
    cin>>n;
    if(n==0||n==1||n==2) {
        if(n==2) cout<<"r7=2"<<endl;
        else cout<<"r7=1"<<endl;
        return 0;
    }
    for(int i=3;i<=n;i++){
        short int r6=r7;
        r7=(r7+2*r0)%Mod;
        r0=r1;r1=r6;
    }
    cout<<"r7="<<r7<<endl;
}
```

设计思路：模仿递推过程。

fib-opt:

```
short int F[16385]={1,1,2,4,6,10,18,30,50,86,...};
int main()
{
    short int n;
    cin>>n;
    cout<<"r7="<<F[n]<<endl;
    return 0;
}
```

设计思路：打表。

二者运行结果：

```
PS C:\Users\pty\Desktop\lab6> ./fib.exe
96
r7=258
PS C:\Users\pty\Desktop\lab6> ./fib-opt.exe
96
r7=258
PS C:\Users\pty\Desktop\lab6>
```

rec:

```
void count(short int &r0,short int &r1){
    if(!r1) return;
    r0+=1;r1-=1;
    count(r0,r1);
    return;
}//Recursive to solve

short int r0=0,r1;
int main()
{
    cin>>r1;
    count(r0,r1);
    cout<<"r0="<<r0<<" r1="<<r1<<endl;
    return 0;
}
```

设计思路：利用函数栈实现函数递归调用。

rec运行结果：

```
PS C:\Users\pty\Desktop\lab6> ./rec.exe
7
r0=7 r1=0
PS C:\Users\pty\Desktop\lab6> ./rec.exe
5
r0=5 r1=0
PS C:\Users\pty\Desktop\lab6>
```

mod:

```
short int r0,r1;
void Solve_Mod(short int &r1)
{
    if(r1<8) return;
    short int r0=r1>>3;//r0=k
    short int r2=r1 & 0x7;//r2=b
    r1=r0+r2;
    Solve_Mod(r1);
}

int main()
{
    cin>>r0;
    r1=r0;
    Solve_Mod(r1);
    cout<<"r1="<<r1<<endl;
    return 0;
}
```

设计思路：利用迭代求解一个数的模7余数，并存入R1中。 这里模仿迭代过程即可。

证明：

```
设x=8k+b,0<b<8，则x=7k+(k+b)
则x≡k+b (mod 7)
迭代求解，最终必得到mod 7结果。
```

mod运行结果：

```
PS C:\Users\pty\Desktop\lab6> ./mod.exe
0
r1=0
PS C:\Users\pty\Desktop\lab6> ./mod.exe
147
r1=7
```

prime:

```
int judge(int r0) {
    int i = 2,r1 = 1;
    while (i * i <= r0) {
        if (r0 % i == 0) {
            r1 = 0;break;
        }
        i++;
    }
    return r1;
}

int main()
{
    int r1,r0;
    cin>>r0;
    r1=judge(r0);
    cout<<"r1="<<r1<<endl;
    return 0;
}
```

设计思路：枚举2~根号n，若存在能整除的数则表示n为合数;否则为质数。

prime运行结果：

```
PS C:\Users\pty\Desktop\lab6> ./prime.exe
97
r1=1
PS C:\Users\pty\Desktop\lab6> ./prime.exe
69
r1=0
```

问题思考

一、. 可以从时间复杂度和空间复杂度两方面去评判，同时也需要从程序运行的稳定性和程序本身的可读性（是否简洁明了）进行评判。

二、本质上说，高级编程语言存在更多的指令与功能，并且程序的可读性更高，能够更好地支持使用者编写复杂的代码。

三、异或运算和移位运算。 拥有异或运算可以简化判断两个元素相等所需的指令数，而拥有移位运算便可极大简化乘除法以及取模的运算指令数。

四、本质上，高级语言可以看作能够直接调用的机器码指令;在编写代码的时候，应当注重程序算法的优化，以及程序可读性：类似LC3一样，以模块搭建的方式构建高级语言程序，能够令程序看起来更简洁，并大大简化工作量。 同时，模拟高级语言中函数栈的工作原理，可以得到LC3实现递归的方法。