

第1章：绪论

1、三数交换

```
/******  
【题目】试写一算法，如果三个整数a, b和c的值  
不是依次非递增的，则通过交换，令其为非递增。  
*****/  
void Descend(int &a, int &b, int &c) //这里的引用改变a的值也会改变n的值  
/* 通过交换，令 a >= b >= c */  
{  
    /*这里的&a是说给一个变量起一个别名，a的值改变，  
    原变量即实参的值也改变，但是在dev c++编译不通过*/  
    int temp;  
    if(a < b){  
        temp = a;  
        a = b;  
        b = temp;  
    }  
  
    if(a < c)  
    {  
        temp = a;  
        a = c;  
        c = temp;  
    }  
  
    if(b < c)  
    {  
        temp = b;  
        b = c;  
        c = temp;  
    }  
}
```

2、一元多项式求值

```
/******  
【题目】试编写算法求一元多项式  
 $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$   
的值P(x0)，并确定算法中每一语句的执行次数和整个算法  
的时间复杂度。  
*****/  
float Polynomial(int n, int a[], float x)  
/* 求一元多项式的值P(x)。 */  
/* 数组a的元素a[i]为i次项的系数，i=0,...,n */  
{  
    int i;  
    float sum;  
    if(n == 0){  
        sum = a[0];  
        return sum; //处理只有a0的情况  
    }
```

```

    }
    sum = a[n] * x + a[n-1];
    for(i = n-2; i >= 0 ; i --)
    {
        sum = sum * x + a[i] ;//秦九韶算法
    }
    return sum;
}

```

3、k阶斐波那契序列

```

/*****
【题目】已知k阶斐波那契序列的定义为
    f(0)=0, f(1)=0, ..., f(k-2)=0, f(k-1)=1;
    f(n)=f(n-1)+f(n-2)+...+f(n-k), n=k,k+1,...
试编写求k阶斐波那契序列的第m项值的函数算法，
k和m均以值调用的形式在函数参数表中出现。
*****/
Status Fibonacci(int k, int m, int &f)
/* 求k阶斐波那契序列的第m项的值f，如果能求得，返回OK */
/*否则（如参数k与m不合理），返回ERROR*/
{
    //这里的序列定义跟斐波那契数列不太一样
    //k不可以为1，因为f(k-2)=0，至少大于等于2
    //k-1项总为1，第k项往后等于前面k项之和
    //如K = 3，则f(k)=f(2)+f(1)+f(0);
    if(k < 2 || m < 0){
        return ERROR;
    }
    //前k-2项都为0
    int i,j,t[100] = {0,0},sum = 0;
    if(m < k-1) f = 0;
    else if(m == k-1) f = 1;
    else {
        t[k-1] = 1;//k-1项为1
        for(i = k; i <= m; i ++){//给第k项~m项赋值
            for(j = i - k; j < i; j ++){
                sum += t[j];//前k项之和
            }
            t[i] = sum;//赋值给第k项~m项
            sum = 0;//记得归零
        }
        f = t[m];
    }
    return OK;
}

```

4、计算 $i! \times 2^i$ 的值

```

/*****
【题目】试编写算法，计算 $i! \times 2^i$ 的值并存入数组
a[0..n-1]的第i-1个分量中（ $i=1,2,...,n$ ）。假设计
算机中允许的整数最大值为MAXINT，则当对某个k
( $1 \leq k \leq n$ )使 $k! \times 2^k > \text{MAXINT}$ 时，应按出错处理。注意
选择你认为较好的出错处理方法。
*****/

```

```

Status Series(int a[], int n)
/* 求 $i! \cdot 2^i$ 序列的值并依次存入长度为n的数组a; */
/* 若所有值均不超过MAXINT, 则返回OK, 否则OVERFLOW */
{
    int i, j = 1;
    Status sum = 1; //记得设sum为1而不是0
    for(i = 0; i < n; i++){
        for(j = i + 1; j > 0; j--){
            sum *= j * 2;
        }
        if(sum > MAXINT){
            return OVERFLOW;
        } else{
            a[i] = sum;
        }
        sum = 1;
    }
    return OK;
}

```

5、统计男女总分及团体总分

/******
【题目】 假设有A、B、C、D、E五个高等院校进行田径对抗赛，各院校的单项成绩均以存入计算机并构成一张表，表中每一行的形式为：

项目名称 性别 校名 成绩 得分

编写算法，处理上述表格，以统计各院校的男、女总分和团体总分，并输出。

相关数据类型定义如下：

```

typedef enum {female,male} Sex;
typedef struct{
    char *sport;      // 项目名称
    Sex gender;       // 性别（女: female; 男: male）
    char schoolname;  // 校名为'A','B','C','D'或'E'
    char *result;     // 成绩
    int score;        // 得分（7,5,4,3,2或1）
} ResultType;

```

```

typedef struct{
    int malescore;    // 男子总分
    int femalescore;  // 女子总分
    int totalscore;   // 男女团体总分
} ScoreType;

```

实现以下函数：

```

*****/
void Scores(ResultType *result, ScoreType *score)
/* 求各校的男、女总分和团体总分，并依次存入数组score */
/* 假设比赛结果已经储存在result[ ]数组中， */
/* 并以特殊记录 { "", male, ' ', "", 0 }（域score=0） */
/* 表示结束 */
{
    int i = 0;
    for(i = 0; i < 5; i++){
        score[i].malescore = 0;
    }
}

```

```

        score[i].femalescore = 0;
        score[i].totalscore = 0;
    }
    for(i = 0;;i++){
        if(strcmp(result[i].sport,"") == 0
        && result[i].gender == male
        && result[i].schoolname == ' '
        && strcmp(result[i].result,"") == 0
        && result[i].score == 0){
            break;
        }
        switch(result[i].schoolname){
            case 'A': {
                if(result[i].gender == male){
                    score[0].malescore += result[i].score;
                }else{
                    score[0].femalescore += result[i].score;
                }
                score[0].totalscore += result[i].score;
                break;
            }
            case 'B': {
                if(result[i].gender == male){
                    score[1].malescore += result[i].score;
                }else{
                    score[1].femalescore += result[i].score;
                }
                score[1].totalscore += result[i].score;
                break;
            }
            case 'C': {
                if(result[i].gender == male){
                    score[2].malescore += result[i].score;
                }else{
                    score[2].femalescore += result[i].score;
                }
                score[2].totalscore += result[i].score;
                break;
            }
            case 'D': {
                if(result[i].gender == male){
                    score[3].malescore += result[i].score;
                }else{
                    score[3].femalescore += result[i].score;
                }
                score[3].totalscore += result[i].score;
                break;
            }
            case 'E': {
                if(result[i].gender == male){
                    score[4].malescore += result[i].score;
                }else{
                    score[4].femalescore += result[i].score;
                }
                score[4].totalscore += result[i].score;
                break;
            }
            default:break;
        }
    }
}

```

```

    }
}
}

```

6、对序列S的第i个元素赋值e

```

/*****
【题目】试写一算法，对序列S的第i个元素赋以值e。
序列的类型定义为：
typedef struct {
    ElemType *elem;
    int length;
} Sequence;
*****/
Status Assign(Sequence &S, int i, ElemType e)
/* 对序列S的第i个元素赋以值e，并返回OK。 */
/* 若S或i不合法，则赋值失败，返回ERROR */
{
    if(&S == null){
        return ERROR ;
    }
    if(S.elem == null){
        return ERROR;
    }
    if(i < 0 || i > S.length){
        return ERROR;
    }
    S.elem[i] = e;
    return OK;
}

```

7、由长度为n的一维数组a构建一个序列S

```

/*****
【题目】试写一算法，由长度为n的一维数组a构建一个序列S。
序列的类型定义为：
typedef struct {
    ElemType *elem;
    int length;
} Sequence;
*****/
Status CreateSequence(Sequence &S, int n, ElemType *a)
/* 由长度为n的一维数组a构建一个序列S，并返回OK。 */
/* 若构建失败，则返回ERROR */
{
    if(n <= 0 || a == null){ //这里不要忽略n等于0的情况
        return ERROR;
    }
    S.length = n;
    S.elem = a;
    return OK;
}

```

8、构建一个值为x的结点

```

/*****
【题目】链表的结点和指针类型定义如下
typedef struct LNode {
    ElemType data;
    struct LNode *next;
} LNode, *LinkList;
试写一函数，构建一个值为x的结点。
*****/
LinkList MakeNode(ElemType x)
/* 构建一个值为x的结点，并返回其指针。*/
/* 若构建失败，则返回NULL。 */
{
    //代码一
    /*LinkList link ;
    link = (LinkList)malloc(sizeof(LNode));
    if(link == NULL) return NULL;
    link->data = x;
    return link;*/

    //代码二
    LNode link;
    link.data = x;
    return &link;
}

```

9、构建长度为2且两个结点的值依次为x和y的链表

```

/*****
【题目】链表的结点和指针类型定义如下
typedef struct LNode {
    ElemType data;
    struct LNode *next;
} LNode, *LinkList;
试写一函数，构建长度为2且两个结点的值依次为x和y的链表。
*****/
LinkList CreateLinkList(ElemType x, ElemType y)
/* 构建其两个结点的值依次为x和y的链表。*/
/* 若构建失败，则返回NULL。 */
{
    LNode link;
    link.data = x;
    LinkList p;
    p = (LinkList)malloc(sizeof(LNode));
    if(p == NULL) return NULL;
    link.next = p;
    p->data = y;
    return &link;
}

```

10、构建长度为2的升序链表

```

/*****
【题目】链表的结点和指针类型定义如下
typedef struct LNode {
    ElemType data;
    struct LNode *next;
}

```

```
} LNode, *LinkedList;
```

试写一函数，构建长度为2的升序链表，两个结点的值分别为x和y，但应小的在前，大的在后。

```
*****/
```

```
LinkedList CreateOrdLList(ElemType x, ElemType y)
```

```
/* 构建长度为2的升序链表。 */
```

```
/* 若构建失败，则返回NULL。 */
```

```
{
```

```
    if(x > y){
```

```
        //不开辟新内存交换两个数
```

```
        x = x+y;
```

```
        y = x-y;
```

```
        x = x-y;
```

```
    }
```

```
    LNode link;
```

```
    link.data = x;
```

```
    LinkedList p;
```

```
    p = (LinkedList)malloc(sizeof(LNode));
```

```
    if(p == NULL) return NULL;
```

```
    link.next = p;
```

```
    p->data = y;
```

```
    return &link;
```

```
}
```