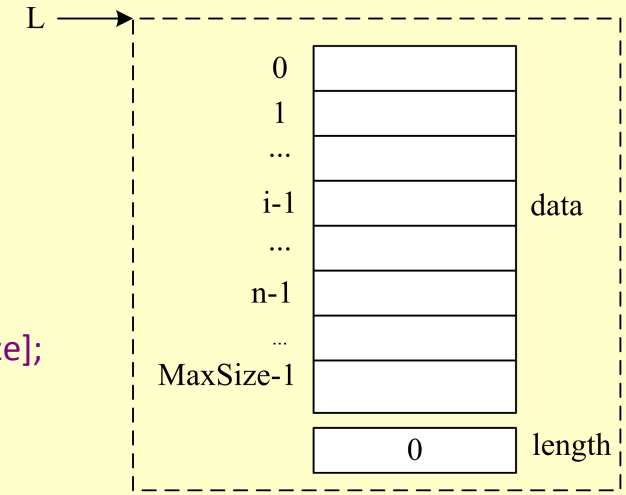
本节主题：

创建线性表的实现(暨参数类型的讨论)

一些约定

- 元素类型

```
typedef int ElemType;
typedef struct
{
    ElemType data[MaxSize];
    int length;
} SqList;
```

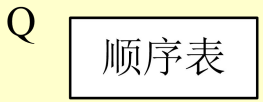


- 数据域的下标
  - 物理序号：从0开始（按C语言）
  - 逻辑序号：从1开始

- 对顺序表的引用
  - 指针法（间接法）


```
SqList *L;
L= malloc(...);
printf("%d\n", L->length);
```

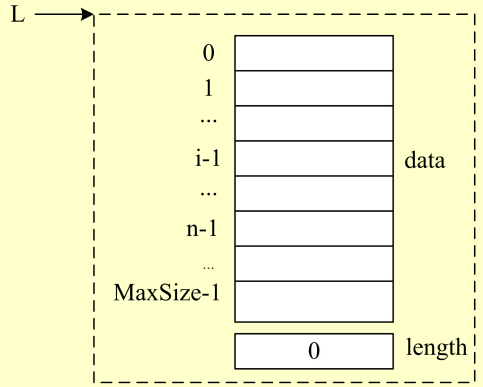
  - 变量法（直接法）




```
SqList Q;
printf("%d\n", Q.length);
```

建立顺序表

- 方法
  - 将给定的含有n个元素的数组的每个元素依次放入到顺序表中，并将n赋给顺序表的长度成员
- 算法




```
void CreateList(SqList *&L,ElemType a[],int n)
{
    int i;
    L=(SqList *)malloc(sizeof(SqList));
    for (i=0; i<n; i++)
        L->data[i]=a[i];
    L->length=n;
}
```

指针的引用？
SqList *&L;

```
int main()
{
   SqList *sq;
   ElemType x[6]={5,8,7,2,4,9};
   CreateList(sq, x, 6);
   DispList(sq);
   return 0;
}
```

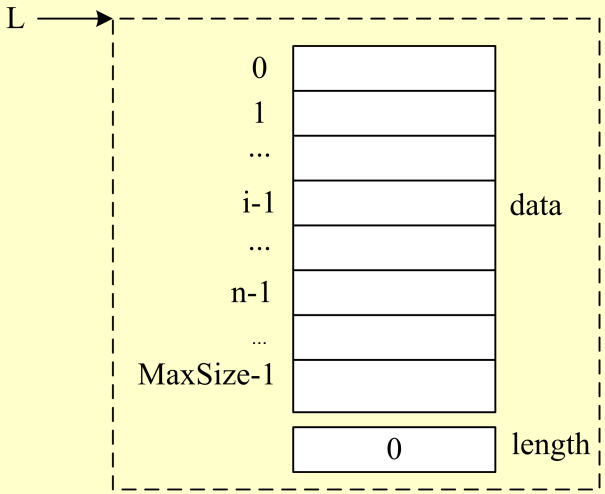
基本运算-初始化线性表InitList(L)

- 功能
  - 构造一个空的线性表L
- 方法
  - 分配空间，并将length成员设置为0
- 算法

```
void InitList(SqList *&L) //指针的引用
{
    L=(SqList *)malloc(sizeof(SqList));
    L->length=0;
}
```

- 时间复杂度
  - O(1)

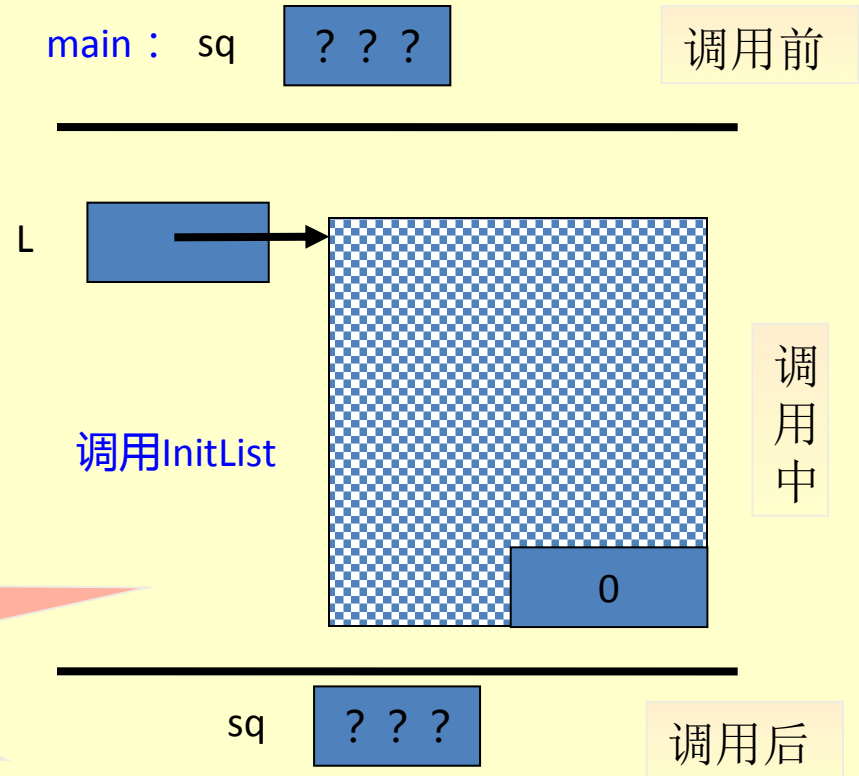
```
#define MaxSize 50
typedef struct
{
    ElemType data[MaxSize];
    int length;
} SqList;
```



讨论参数类型：用指针会怎么样

```
int main()
{
    SqList *sq;
    InitList(sq);
    DispList(sq);
}
void InitList(SqList *L) //不用引用
{
    L=(SqList *)malloc(sizeof(SqList));
    L->length=0;
}
```

用传指针方式，
返回到main()
后实参sq没有变化！



讨论参数类型：用引用会怎么样

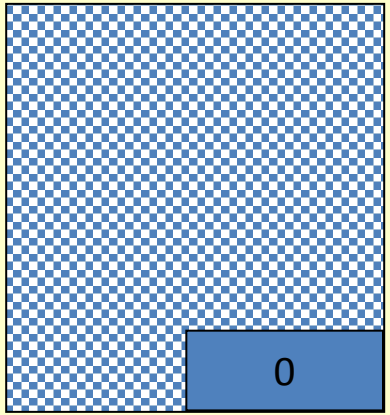
```
int main()
{
    SqList sq;
    InitList(sq);
    DispList(&sq);
}
void InitList(SqList &L) //用引用
{
    L.length=0;
}
```

仅靠引用，
死板的机制！

main： sq

L

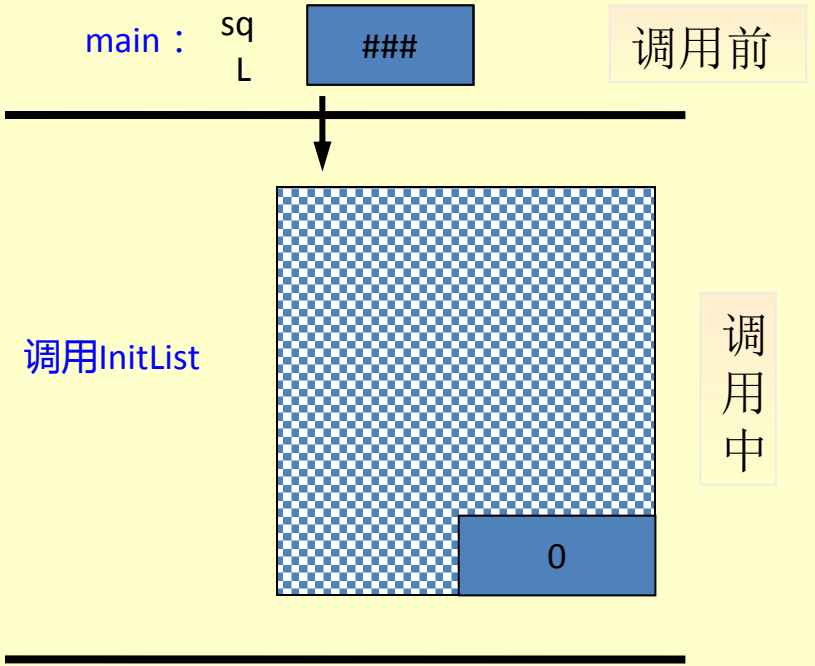
调用InitList



参数类型：用对指针的引用是这样的

```
int main()
{
    SqList *sq;
    InitList(sq);
    DispList(sq);
}
void InitList(SqList *&L) //用指针的引用
{
    L=(SqList *)malloc(sizeof(SqList));
    L->length=0;
}
```

```
int b=0;
int &a=b;
SqList *&L的“拆解”？
```



重要提示——针对引用

- 引用，是 C++中提供的类型，故本课中所有程序需要用 C++ 的编译环境完成
  - VC++6.0：直接建项目
  - CodeBlocks：建立项目时选C++
- 本课算法中还使用了某些C++中的成份，例bool类型