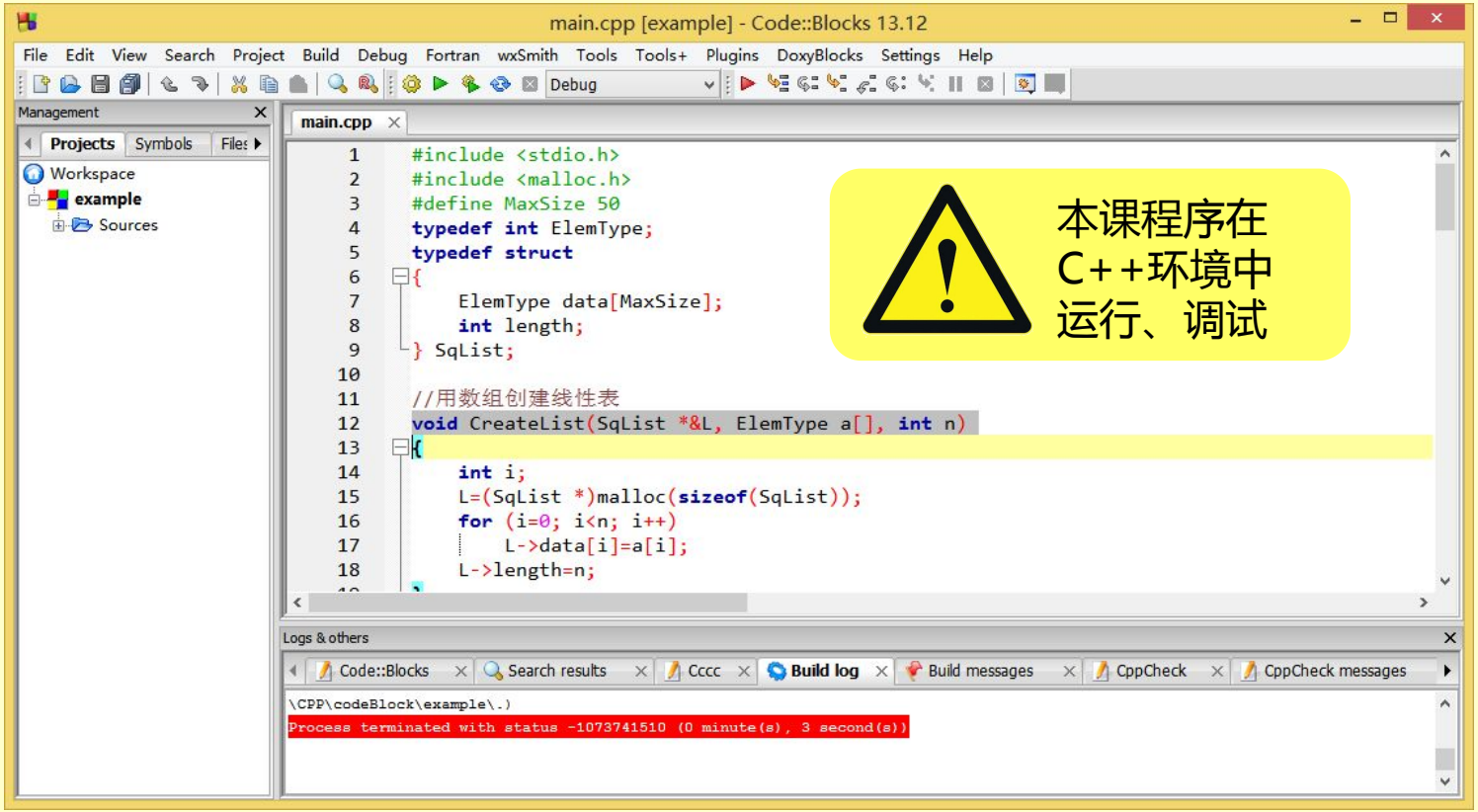
本节主题：

实践指导：用程序实践算法

实践方式1：将算法变程序，在验证中学习



将算法变程序的步骤

（1）定义数据的存储结构

```
#include <stdio.h>
#include <malloc.h>
#define MaxSize 50
typedef int ElemType;
typedef struct
{
    ElemType data[MaxSize];
    int length;
} SqList;
```

（3）分别定义main函数，进行测试

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

（2）实现各基本操作

```
//用数组创建线性表
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;
}
//初始化线性表InitList(L)
void InitList(SqList *&L)   //引用型指针
{
    L=(SqList *)malloc(sizeof(SqList));
    //分配存放线性表的空间
    L->length=0;
}
……
```

实践方式2(最佳实践)：多文件组织自己的算法库

头文件定义存储结构，声明函数
list.h

```
#ifndef LIST_H_INCLUDED
#define LIST_H_INCLUDED
#define MaxSize 50
typedef int ElemType;
typedef struct
{
   ElemType data[MaxSize];
   int length;
} SqList;
void InitList(SqList *&L);
void DestroyList(SqList *&L);
bool ListEmpty(SqList *L);
int ListLength(SqList *L);
……
#endif
```

实现各基本操作对应的函数
list.cpp

```
#include <stdio.h>
#include <malloc.h>

void InitList(SqList *&L)
{
   L=(SqList *)malloc(sizeof(SqList));
   L->length=0;
}

void DestroyList(SqList *&L)
{
   free(L);
}
……
```

写测试函数
main.cpp

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

实践方式3：基于算法库，解决问题

- 例如：设顺序表有10个元素，其元素类型为整型。设计一个算法，以第一个元素为分界线，将所有小于它的元素移到该元素的前面，将所有大于它的元素移到该元素的后面。


原 数据结构课程的实践方法指导

分类： 数据结构

实践方法 算法

目录(?) [+]

有不少人说数据结构课程抽象，学习起来感到困难。有些同学放弃了，有些少，效果不佳，反倒得到很多枯燥的感受。

其实，数据结构课中有不少理论性分析的内容，但对于本科生学习的内容，课程就可以展示出生动的实践性味道来。这就要求在学习过程中，将实践学习有

学习数据结构和算法的过程中做出的这些实践，也便成了程序设计能力提高

```
main.cpp
#include "list.h"
void move1(SqList *&L)  //定义解决问题的算法
{
   int i=0,j=L->length-1;
   ElemType pivot=L->data[0];
   ElemType tmp;
   while (i<j)
   {
          ……
   }
   ……
}
```

```
int main()   //测试函数
{
   SqList *sq;
   ElemType x[10]= {3, 8, 2, 7, 1, 5, 3, 4, 6, 0};
   CreateList(sq, x, 10);
   DispList(sq);
   move1(sq);
   DispList(sq);
   return 0;
}
```

细微观察方法——单步跟踪

main.cpp [DS] - Code::Blocks 13.12

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Debug

Watches (new)

Function a	
L	@0x28fefc: 0
Locals	
i	3
j	6
pivot	3
tmp	8
*L	SqList
data	
[0]	3
[1]	0
[2]	2
[3]	7
[4]	1
[5]	5
[6]	3
[7]	4
[8]	6
[9]	8
[10]	-1163005939
length	10

main.cpp | list.h | list.cpp

```
1  #include "list.h"
2
3  void move1(SqList *&L)  //定义解决问题的算法
4  {
5      int i=0,j=L->length-1;
6      ElemType pivot=L->data[0];
7      ElemType tmp;
8      while (i<j)
9      {
10         while (i<j && L->data[j]>pivot)
11             j--;
12         while (i<j && L->data[i]<=pivot)
13             i++;
14         if (i<j)
15         {
16             tmp=L->data[i];
17             L->data[i]=L->data[j];
18             L->data[j]=tmp;
19         }
20     }
21     tmp=L->data[0];
22     L->data[0]=L->data[j];
23     L->data[j]=tmp;
24 }
25
26 int main()   //在main函数中调用，保证程序能运行，解决问题
27 {
28     SqList *sq;
29     ElemType x[10]= {3, 8, 2, 7, 1, 5, 3, 4, 6, 0};
30     CreateList(sq, x, 10);
31     DispList(sq);
```