实践11小结

- 字符数组，末尾莫忘加串结束符'\0'。
- 更多的越界：- Stack around the variable ‘s’ was corrupted.
- strcpy,strcat函数中，被赋值的字符数组，莫忘预留足够的空间。
- 通过字符指针操作字符串是上学期的最重点
  - 熟练掌握C风格字符串的输入、输出、拷贝、拼接、比较、求长度、逆序、大小写转换等基本算法和相关库函数。

实践11小结

- 指针定义初始化三大坑：
  - int *p=NULL; *p=5; ×
  - int *p; *p=5; ×
  - int *p="OK"; *p=5; ×
- ☆正确用法int a; p=&a; *p=5;
- 指向数组的指针
  - int *p; int Arr[N]; p=Arr;
  - 第i个元素： Arr[i] *(p+i) p[i]
  - 第i个元素地址：&Arr[i] p+i &p[i]
- 实验中一级指针*p的错误用法 *p[i] ×

实践11小结

- strrev函数可以这么调用吗？为什么？

  char s1=“白雪”;

  cout<< myStrRev(s1);

  cout<< myStrRev(“MadamI‘mAdam”);

- 又见差一误差：
  - myStrRev时，将最后的串结束符'\0'拷贝到反转后字符数组的首个位置。

实践11小结

- 形参类型const char*为常量指针，表明指针指向的内容不可写，用于在函数调用时保证不改变实参内容。凡是函数内只读不写的参数，习惯加上const修饰。
- 关于返回

```
cout<<myStrCat(s,ct)<<endl;
cout<<s<<endl;
```

两句输出结果一样

```
cout<<myStrRev(s)<<endl;
cout<<s<<endl;
```

两句输出结果一样

实践11 小结

- 返回指针类型，与返回引用一样，实质是返回地址。
- 无论是返回引用或指针，都要返回有效的、合法的地址，所以不能返回函数当中定义的变量地址，因此返回指针形参本身的值比较常用(与返回引用时可以返回从形参传入的引用一样原理)。
- 正确使用的前提——看透内存存储的本质！

实践11小结

各种错误的存在，正是初学实验时应有的状态！

- 字符数组传入函数后的处理问题;
  - 滥用全局变量，函数内使用全局变量，使函数相互耦合，功能不明晰，可移植性差，除了学习全局变量的访问，现阶段所有作业都可以不使用全局变量。
  - 滥用动态内存申请，动态内存申请非灵丹妙药，可能是个坑，本学期禁入，现有作业都可以不使用。
- 课后习题5.8，“聪明地”各种改函数原型。
  - 逆序输出数组函数，形参增加了一个指针作为返回。
  - 返回指针类型改为**void**，破坏了函数功能完整性。

实践11小结

循环的三要素：循环的起始条件、维持条件和条件变量的改变。

注意：判断条件当中出现赋值、自增符号等情况。

【例 5.2】字符数组与字符数组相连接。

```
#include <iostream>
using namespace std;
void strcat(char s[],char ct[]){
    int i=0,j=0;
    while (s[i]!=0) i++;                    //找到s数组
    while (ct[j]!=0)   s[i++]=ct[j++];      //把ct数组的
    s[i]='\0';                              //加串结束符
}
```

```
char *strcpy(char *s, char *ct){
    char *t = s;
    while(*s++ = *ct++);
    return t;
}
```

√

```
char *strcpy(char *s, char *ct){
    while( *s++=*ct++);
    return s;
}
```

×

思考：1.为什么用=可以，2.为什么不加'\0'，3.为什么可以返回t，同样是返回地址，若函数内定义char t,为什么返回&t错误，4.为什么返回s错误。

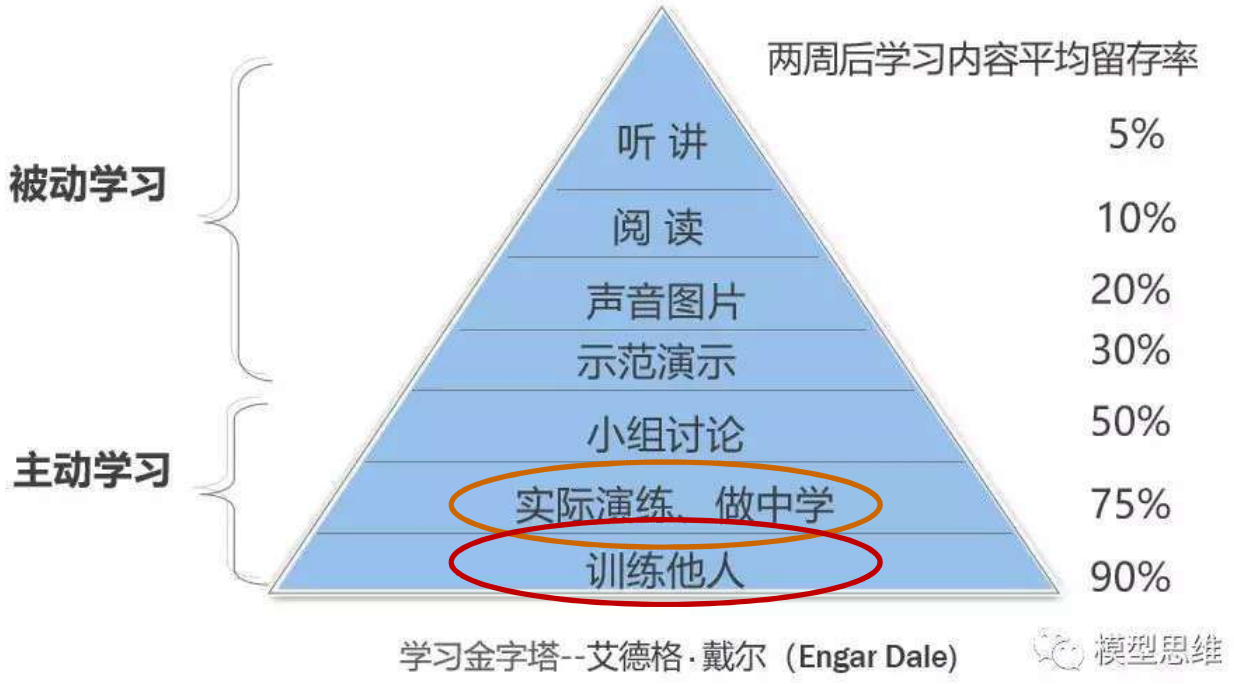
实践12 指针提高

- 实验内容：
  - 二级指针
  - 查找算法
  - 文本文件输入输出

在程序顶部用注释，列举出测试或运行结果。

实践12：

- 线性表的元素查找
  - 顺序
  - 折半/二分



实践12：提交程序清单1

顺序查找，指针作为函数参数的练习。

函数原型:（注意不得修改函数原型）

```
char *myStrChr(const char* s, char c);   //查找字符c在字符串s中第一次出现的位置，有则返回该字符指针，没有则返回NULL.

char *myStrStr(const char* s1, const char* s2); //查找字符子串s2在字符串s1中第一次出现的位置，有则返回该字符指针，没有则返回NULL，

int   myStrCmp(const char* s1, const char* s2); //比较字符串s1和s2，功能同 strcmp()，若str1=str2，则返回零；若str1<str2，则返回负数；若str1>str2，则返回正数。
```

- 注意区别：实验十三题2返回的是数组下标，本题返回的是指针

结果验证：与CString库函数调用结果比对

自定义函数	CString库函数
myStrChr	strchr
myStrStr	strstr
myStrCmp	strcmp

实践12：提交程序清单2

自定义初始化一个有重复数字的有序数组。

如：int arr[7]={1,2,2,4,5,5,6};

实现函数：

int BinarySearchFirst(int x, const int *data, int size);

使用二分查找法，返回数组中第一个大于等于查找值x的位置(下标+1)，

如：查找2，返回2，查找3，返回4，查找5，返回5.

若数组中不存在这样的数，函数返回数组长度加一。

如：查找8，返回8.

若数组为逆序，则上述大于改为小于。

实践12：提交程序清单3

数组为函数参数的再实践。

假设要完成学生成绩统计表如下：

	学号1	学号2	学号3	学号4	学号5	平均分
高数	5	5	5	5	5	5
英语	5	5	5	5	5	5
C++	5	5	5	5	5	5
总分	15	15	15	15	15	15

- 定义函数原型(不得更改函数原型)：

```
void createMatrix(double data[], int row, int column, int min, int max);
void avrgRow (double data[], int row, int column);
void sumColumn(double *data, int row, int column);
void saveData(const double data[], int row, int column, char file[] );
void outputData(const double *data, int row, int column);
```

- 1. 假设有一个4行6列的二维矩阵，矩阵定义为：

```
const unsigned int ROW=4;
const unsigned int COLUMN=6;
double martix[ROW][COLUMN];
```

(注意：不得更改数组声明)

用createMatrix函数随机生成前3行5列各元素，使得元素值为[-10,10]区间内的随机整数，且同时每次调用该函数，生成的数值不同。

- 2. 用avrgRow函数计算前3行元素的平均值，四舍五入保留到小数点后1位精度，存入矩阵第6列中。
- 3. 用sumColumn函数计算每列元素的和，存入矩阵第4行中。
- 4. 用outputData函数输出矩阵内容到console窗口

一次顺序执行了createMatrix、avrgRow、sumColumn、outputData四个函数的调用之后，console窗口输出范例：

-10	9	8	-7	5	1
3	1	8	7	0	3.8
1	-2	-8	10	5	1.2
-6	8	8	10	10	6

- 4. 用函数saveData()将所得矩阵各元素保存到文本文件中，

  文件名：Exp12_学号_Matrix.txt
- 各元素占据8个字符宽度，遇到新的一行数据换行，即存储到文件时按二维矩阵格式存储。
- 5. 提交文件：

  Exp12_学号_Matrix.cpp

  Exp12_学号_Matrix.txt
- 提示：尽管题目要求是二维矩阵，但函数实现时是以一维数组实现的，思考函数调用时该如何传入参数?

重要！！！

提交的文件名格式：

Exp12_学号_实验名.cpp

注意:1）下划线

2）学号别漏掉

3）各种名字(变量名，文件名)的可读性!

4）每个实验只要提交一个程序源文件，即.cpp文件。

附加题也用*Exp12_学号_实验名.cpp*格式提交！

实践12小结

- 函数的参数传递。
  - 传值(pass by value)，声明、定义、调用的格式。
  - 传地址(pass by address)，声明、定义、调用的格式。
- 传递地址，可对参数指向的内存进行读写。
  - 引用，比指针安全，优先使用。
  - 字符数组名，在函数内判断数组内容，获取字符串长度(以实参有终止符'\0'为前提)。
  - 其他类型数组名，一定要把数组size当作形参传递给函数。
  - 指针变量，一定传递有效地址，正确使用的前提——看透内存存储的本质！
  - 保护实参，若实参相关信息不想被修改，定义函数时相应形参可用const修饰引用和指针。

实践12小结

- 函数返回
  - 值返回，常规用法。
  - 地址返回，无论是返回引用或指针，都要返回有效地址，所以不能返回函数当中定义的变量地址，因此返回指针形参本身的值比较常用。正确使用的前提——看透内存存储的本质！(返回引用时，更易出错)

实践12小结

- 二维数组在调用时给一维指针形参传递的两种方式：

```
double martix[ROW][COLUMN];

avrgRow(martix[0], ROW, COLUMN);
outputData(&martix[0][0], ROW, COLUMN);
```

实践12小结

- 课后习题5.8，“聪明地”修改函数原型。
  - 子串判断函数，形参const *的问题。

```
char *myStrChr(const char *s, char c)
{……
    return (char*)s;
……}
```

```
char *myStrStr(const char *s1, const char *s2)
{… char *ps=NULL;
 …    ps=(char*)&s1[i];   //取地址方法一
 …    ps=(char*)(s1+i);   //取地址方法二
 … return ps;
…}
```

实践12小结（机试动员）

- 必须学会自己完成错误排查。
  - 语法错误，连接错误
  - 运行错误，逻辑错误(似是而非)
- 试题要求不可改动的源程序或函数原型
  - 不要擅自改动。
  - 自定义函数的声明——送分题。
- 本学期实验中调用过的所有库函数：
  - 熟背头文件名和常用函数名——送分题。
- **main()**函数中通常比较简单
  - 调用自定义函数——送分题。

实践12小结（机试动员）

- **VS2010蜜汁操作大赏**
  - **‘’中是字符串，过！**
  - **case’a’: 没有空格，过！**
  - **函数定义时没有形参名，过！**

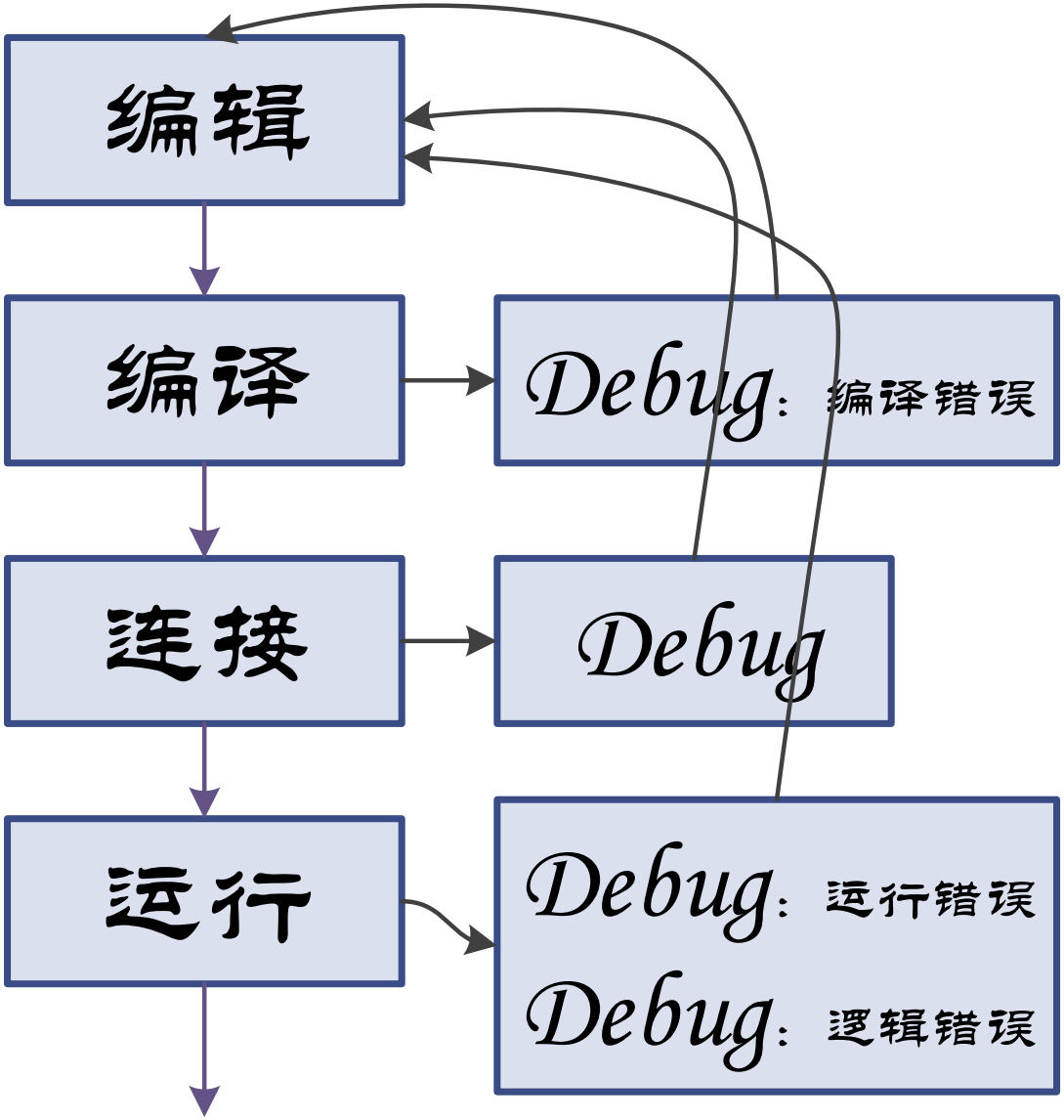
以上三种语法错误要靠自己去避免！！！

实践12小结（机试动员）

按要求！按时间！ 答题与提交

算法不会实现时，即使函数定义时只写一个空函数，相应的函数声明和调用也是得分点。

程序设计过程



实践12小结(笔试动员)

1.选择题

~~2.程序改错~~

侧重基础语法知识

3.读程序写结果

4.完善程序

侧重应用。
主要考察知识点：循环结构，分支结构，函数，静态局部变量，字符数组。引用、数组、指针作为函数形参等。

You deserve better!

- Nothing that has meaning is easy.
  - Easy doesn't enter into grown-up life.
- Why do we work hard?
  - to see a bigger world!
  - to have the opportunity to choose our life!