实践10小结

- 引用和数组作为函数参数时，声明，定义，调用时也遵循：参数类型、个数、顺序必须一致。
- 引用作为函数参数时
  - 函数头相应参数名前加&(直接取地址运算符)。
  - 调用函数时与传参调用一样的形式。
- 数组为函数参数时的注意事项：
  - 调用函数时传入数组名。
  - 数组名是只读的指针常量，编译器不检查也无法检查数组边界，对数组边界的控制由程序员掌握。

实践10小结

- 函数的功能——主调函数与被调函数之间的规约。
- 函数的参数——多个功能块协作时，相互之间的接口(interface)。

```
int dicing(){return 2;}
```

- 函数返回值的类型由函数返回类型决定。

```
int digit(char num[], int k) {
    ……
    return num[place];
}
```

vs.

```
int digit(char num[], int k) {
    ……
    return num[place]-'0';
}
```

实践10小结

- 函数作为左值问题：
  - *P15*,左值必须是在内存中可以访问并且可合法修改值的存储单元。（右值需要可以取到确定的值）
  - 传值返回时，函数不可以作为左值，返回前申请的临时变量在返回后生命期终止(*P123*,图4.6)。
  - 引用返回时，函数可以作为左值，实际是函数返回值可以作为左值，返回值必须是在生命期的地址，如，返回全局变量的引用(*P123*,图4.7)或者主调函数通过引用参数传递到被调函数的变量(*P123*,例4.5)。

实践10小结

- 程序员眼中的表象，计算机眼中的本质：

```
for (i=0; i<SIZE; i++) {
    srand(time(0));
    arr[i]=rand()%100;
}
```

vs.

```
int GetNum () {
    srand(time(0));
    return rand()%100;
}
```

同：{}内语句执行顺序一样；异：函数调用时在栈内有一系列动态：建立栈空间，保护现场，传递参数，控制程序执行的转移，恢复现场，释放栈空间。

```
int matr1[ROW*COL];
```

vs.

```
int matr2[ROW][COL];
```

同：数据在内存中的存储一样；异：matr1是一级指针，matr2是“二级”指针

实践10小结

- 查漏补缺：
  - 条件语句：if(表达式) {语句块}
  - 循环结构：for (式1；式2；式3）{语句块}
  - 函数：定义，声明，调用。
  - 错误：语法错误→连接错误→运行错误→逻辑错误(似是而非)
- 数组使用注意事项：
  - 使用循环访问数组时，切记检查循环的第一次和最后一次迭代，查看下标是否在允许范围内。有意识控制避免差一误差。
  - 多维数组更需要反复检查，测试阶段，每轮循环都要排查是否存在越界访问风险。

二维数组A2D等效一维数组A访问方式

数组下标访问法

A2D [m][n]

.

A [m*n]

*(A2D+i)+j

或 &A2D[0][0]+(i*n+j)

.

A+(i*n+j)

或 &A[0]+(i*n+j)

指针访问法

实践11 指针初步

- 实验内容：
  - 指针变量的定义与初始化
  - 指针作为函数参数

实践11：

- 指针的定义和使用方法。
- 指针与数组的相互关系。
- 数组或指针作为函数参数时，函数的声明、定义及调用。
  - 三种函数参数调用方式的理解

    (pass by Value， Address(Reference,pointer))
- 字符数组的进一步理解和使用。
- 实践内容：实验十三、十四，课本习题5.6~5.8。

实践11：提交程序清单1

- 1. 实验十四-3，自定义一个密钥，按题目规则对输入的字符串进行加密输出，再以相同的密钥进行解密输出。（即分别实现加密、解密函数）
- 注意：考虑是否会溢出的问题。题目要求%128实际已经是一种防溢出的操作了。密钥选择int型的话，也要保证 key+128<2^31-1
- 在C++中，定义了一些表示基本数据类型范围的常量，如：INT_MAX = 2^31-1, INT_MIN= -2^31

实践11：提交程序清单2

2、实验十四-4，以指针为函数参数，重新实现实验十三范例2。

目的1：理解在函数的参数列表中，“字符数组”与“字符指针”等价。

如：void trim(char s[]);　等价于　void trim(char *s);

目的2：熟悉C风格字符串cstring库函数的功能和使用。

目的3：进一步熟悉数组，尤其是字符数组的使用。

实践11：

*重要！自行完成，不需提交程序。

习题5.6，5.7，理解指针的定义、赋值、属性。

实践11：提交程序清单3

3，课本习题5.8，以指针为函数参数实现字符串处理。

函数原型及函数需要完成的功能：

```
char *myStrCat(char* s, const char* ct); //将串ct接到串s后面，形成一个长串，功能同 strcat()
int  myStrLen(const char*);   //返回字符串长度，功能同 strlen()
char *myStrCpy(char* s, const char* ct); //将串ct拷贝至串s，功能同 strcpy()
char *myReverse(char* s);     //反置字符串s，
```

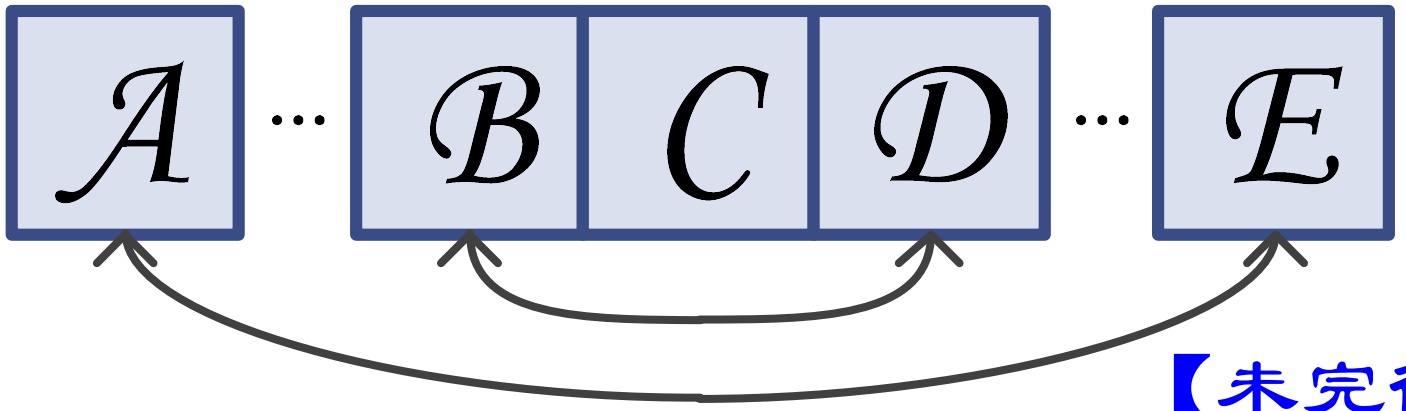
Tips:本实验函数形参都没有设置形参传递数组长度，因此都需要程序员自己保证实参字符数组有结束符'\0'。

【未完待续】☞

- 关于字符串反置函数myReverse ()，不设置字符串长度的形参，需要保证实参字符数组有结束符'\0'。
  - 实现方案1/2：经典算法。首先取到传入的实参数组长度，然后以中间位置为对称轴，依次交换对称轴前后字符。

方案一:
left=0, right=len-1;
left++,right--; left<right

方案二：
[i]<-->[len-i-1]
i++; i<len/2



【未完待续】 ☞

– 实现方案3：更简单。

- ①在函数内定义一个长度大于测试字符串长度的字符数组（本学期不使用动态内存申请，以大开小用原则定义这个临时变量），作为反置字符串的过渡(类似交换两数时的temp变量)，
- ②将实参字符数组元素逐个逆向拷贝给过渡字符数组，
- ③ 将过渡数组元素再复制(strcpy或自定义拷贝函数)给实参数组。

三种方案，总有一款适合你！

实践11：

4*、较简单，自选完成。习题5.5，编写函数将字符串s转换为整型数返回。

函数原型： int my_atoi(char *s);

与库文件<cstdlib>中包含的atoi()函数转换结果相比较来验证你的函数功能。

经测试，atoi函数也没有做防溢出保护，故而自定义的转换函数也不实现防溢出包含，仅实现数据类型在int取值范围内的字符串的转换。

即使不实现，也自行理解windows库函数中相应的函数。

实践11：提交程序清单4

5、实验十三.3,

要求1：使用递归和非递归方法分别编写myItoa函数，将整数n转换为以radix为基的数字字符数组。

函数原型为：void myItoa(int n, char s[], int radix);

其中，radix为转换后的进制数(2,8,10,16)。

注意：当radix为16时的处理与其他进制的差别。

要求2：调用myItoa函数，编写函数判断回文数函数。

函数原型为：bool isPalindrome(int n, int radix);

判断一个数在基radix=2,8,10,16进制下是否为回文数。

实践11：

6*、自选完成，用递归法实现判断回文数函数。函数原型仍为：bool isPalindrome(int n, int radix);

注：实现算法多样，不局限是否调用5中的myItoa函数，但肯定需要数字转换为字符的处理。

实践11：

6*、自选完成，课本第5.7节的范例。

目的：了解二级甚至多级指针的使用。

目的：理解下列两种数组名的差异

```
int Arr2D[ROW][COLUMN];          //“二级”指针
int *ArrPointer[SIZE];           //真二级指针
```