※凯撒密码

for(i = 0; i < (int)(sizeof(s)/sizeof(char)) ; i++) //!

{

if(s[i] >= 'A' && s[i] <= 'Z')

s[i] = (s[i] - 'A' + n) % 26 +'A';

if(s[i] >= 'a' && s[i] <= 'z')

s[i] = (s[i] - 'a' + n) % 26 +'a';

}

用模运算达到圆桌效应。

一、sizeof

sizeof(...)是运算符，在头文件中typedef为unsigned int，其值在编译时即计算好了，参数可以是数组、指针、类型、对象、函数等。

它的功能是：获得保证能容纳实现所建立的最大对象的字节大小。

由于在编译时计算，因此sizeof不能用来返回动态分配的内存空间的大小

sizeof返回的值表示的含义如下：

数组——编译时分配的数组空间大小；

指针——存储该指针所用的空间大小（存储该指针的地址的长度，是长整型，应该为4）；

类型——该类型所占的空间大小；

对象——对象的实际占用空间大小；

函数——函数的返回类型所占的空间大小。函数的返回类型不能是void。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

二、strlen

strlen(...)是函数，要在运行时才能计算。参数必须是字符型指针（char\*）。

当数组名作为参数传入时，实际上数组就退化成指针了。

它的功能是：返回字符串的有效长度。

该字符串可能是自己定义的，也可能是内存中随机的，该函数实际完成的功能是从代表该字符串的第一个地址开始遍历，直到遇到结束符NULL（'\0’）。返回的长度大小不包括NULL(字符串的有效长度)。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

区别：①对象 ②功能上的区别

三、举例（②）：

eg1、char arr[10] = "What?";

int len\_one = strlen(arr);

int len\_two = sizeof(arr);

cout << len\_one << " and " << len\_two << endl;

输出结果为：5 and 10

点评：sizeof返回定义arr数组时，编译器为其分配的数组空间大小，不关心里面存了多少数据。strlen只关心存储的数据内容（有效长度），不关心空间的大小和类型。

eg2、char \* parr = new char[10];

int len\_one = strlen(parr);

int len\_two = sizeof(parr);

int len\_three = sizeof(\*parr);

cout << len\_one << " and " << len\_two << " and " << len\_three << endl;

输出结果：23 and 4 and 1

点评：第一个输出结果23实际上每次运行可能不一样，这取决于parr里面存了什么（从parr[0]开始知道遇到第一个NULL结束）；第二个结果实际上本意是想计算parr所指向的动态内存空间的大小，但是事与愿违，sizeof认为parr是个字符指针，因此返回的是该指针所占的空间（指针的存储用的是长整型，所以为4）;第三个结果，由于\*parr所代表的是parr所指的地址空间存放的字符，所以长度为1。

* missing braces around initializer

警告的意思是初始化缺少了大括号，也就是没有用大括号明确的区分出初始化数据的归类。

一般情况下初始化使用到大括号的情况是初始化结构体struct或者数组时

结构体时一种将不同类型的数据组合成一个有机整体的聚合数据类型。那么如果结构体中的成员的类型是数组或者结构体时呢？那么这个时候你再仅仅只是用一个大括号来对结构体进行初始化时，就会出现定义不明确的问题了，此外如果对一个多维数组你也只使用一个大括号来进行初始化时，同样也会出现定义不明确的问题。

有时候我们想要的赋值并不是依次赋值的，例如我想array1[1][0]不赋值而array[1][1]=14,如果只使用大括号的情况是：array1[2][2] = {11, 12, 14}，结果是array1[1][0]=14，而array1[1][1]=0，这样的结果并不是编程者需要的，所以编译器需要提示你检查对应的地方是否是你需要的或者你需要规范一下代码，来避免这种隐藏的问题或者错误。

toupper（char ）

to upper

将ch变成大写字母，该函数没有副作用，返回值才是变为大写字母 <ctpye.h>

Isupper(char)

is upper

判断是否为大写字母 <ctype.h>

int f( char ch) //返回字母对应的数字。

{

const char a[27] = {'0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',

'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'}; //添0 使数字对应

int i = 1;

ch = toupper(ch);

if(isupper(ch))

{

while(ch != a[i++]);

//empty loop

return i - 1;

}

else

{

return -1;

}

}

※bool play\_game(void);

game = play\_game();//一定要加括号！！！

否则warning：assignment makes pointer from integer without a cast

因为函数名代表一个指针，将指针赋值给整数变量会导致warning