**作 业 报 告**

|  |  |
| --- | --- |
| **课程名称** | C程序设计语言 |
| **作业名称** | 函数 |

|  |  |
| --- | --- |
| **学院** | 生命科学学院 |
| **专业** | 生物信息学 |
| **班级** | 生信1801 |
| **学号** | 8305180710 |
| **姓名** | 张彦哲 |
| **实验日期** | 2019年10月20日 |
| **指导教师** | 彭小清 |

1. **编译环境：**

*//!encoding: UTF-8  
//IDE: Clion 2019.2.2*

*//CMake: Version 3.14.5  
//Architecture: Windows 10 SDK version 1903  
//Author: Sean Peldom Zhang  
//Create Time: 2019/10/10-2019/10/20*

1. **作业要求**：

任选教材第7章课后4道题，完成并打印或手抄。

1. **第7章第4题**

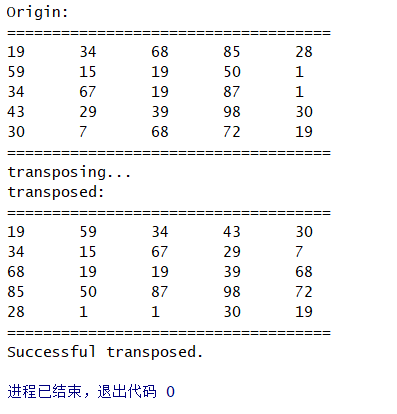
题目：写一个函数，对给定的3\*3的二维整形数组转置(行列互换)

思路：随机生成5\*5二维数组，然后行列交换赋值给一个新的外部数组，打印出来

源代码：

#include **<stdio.h>**#include **<stdlib.h>**#include **<time.h>  
int** b[5][5];  
**void** transpose(**int** a[5][5]){  
 **for** (**int** i = 0; i < 5; ++i) {  
 **for** (**int** j = 0; j < 5; ++j) {  
 b[i][j]=a[j][i];  
 }  
 }  
}  
**int** main(**void**){  
 **int** a[5][5];  
 srand((**unsigned**)time(**NULL**));  
 **for** (**int** i = 0; i < 5; ++i) {  
 **for** (**int** j = 0; j < 5; ++j) {  
 a[i][j]=rand()%100+1;  
 }  
 }  
 puts(**"Origin:\n===================================="**);  
 **for**(**int** i=0;i<5;i++){  
 **for** (**int** j = 0; j < 5; ++j) {  
 printf\_s(**"%d\t"**,a[i][j]);  
 }  
 puts(**""**);  
 }  
 transpose(a);  
 puts(**"====================================\ntransposing..."**);  
 puts(**"transposed:\n===================================="**);  
 **for**(**int** i=0;i<5;i++){  
 **for** (**int** j = 0; j < 5; ++j) {  
 printf\_s(**"%d\t"**,b[i][j]);  
 }  
 puts(**""**);  
 }  
 puts(**"====================================\nSuccessful transposed."**);  
 **return** 0;  
}

运行结果：



1. **第7章第5题**

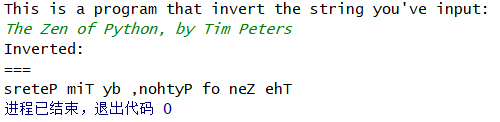
题目：写1个函数，使输入的一个字符串反向存放，然后输出转换后的字符串

程序思路：用一个char指针储存原字符串，然后移动到末尾，逐个赋值给新的指针，再写一个函数打印该字符串指针

源代码：

#include **<stdio.h>**#include **<stdlib.h>**#include **<string.h>  
char** \*invertstring(**char** \*a){  
 **char** \*inputstr=a; **int** strori=strlen(a);  
 **while**(\*inputstr)inputstr++;*//移到末尾地址* **char** \*ivtdstr=inputstr;  
 \*(ivtdstr+1)=**'\0'**;  
 **while** (\*--inputstr && strori--){  
 **char** temp=\*inputstr;  
 \*ivtdstr++=temp;  
 }  
 *// \*ivtdstr='\0';* **return** ivtdstr;  
}  
**void** printivtstr(**char** \*b){  
 **char** \*inputstr=b;  
 **int** strori=strlen(b);  
 **while**(\*inputstr){*//移到末尾地址  
 // printf\_s("%c,%x\n",\*inputstr,inputstr);* inputstr++;  
 }  
 puts(**"==="**);  
 **while**(\*--inputstr && strori--){  
 printf\_s(**"%c"**,\*inputstr);  
 }  
}  
**int** main(**void**){  
 puts(**"This is a program that invert the string you've input:"**);  
 **char** inputstr[100];gets\_s(inputstr,100);  
 puts(**"Inverted:"**);  
 *// printf\_s("\_strrev=%s\n",\_strrev(inputstr));//这个函数会打乱inputstr指针，彻底反转  
 //printf\_s("invertstring=%s\n",invertstring(inputstr));* printivtstr(inputstr);  
 **return** 0;  
}

运行结果：（文字来源pythonV3.7.4 >>>import this）



1. **第7章第6题**

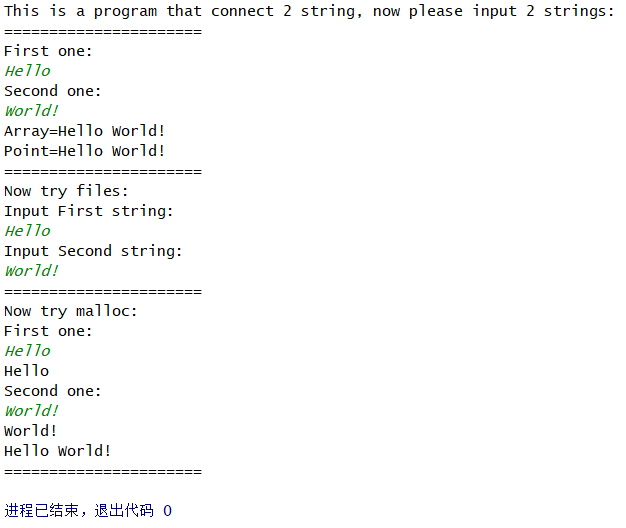
题目：写1个函数，连接2个字符串

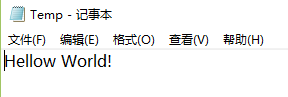
思路：4种方法：1. 数组，很容易，逐字赋值； 2.指针，随后逐字定义；3.malloc，分配2个空间储存原字符串，然后重新检测字符串长度，再赋值；4.对一个txt文件写入字符串2次，然后用dos命令打开；

源代码：

#include **<stdio.h>**#include **<string.h>**#include **<assert.h>**#include **<stdlib.h>  
char** \*strcat\_array(**char** first[160],**char** second[160]){*//数组处理方式* **assert**(first!= NULL && second!= NULL);  
 **unsigned int** i=strlen(first),k=strlen(second);*//i为first字符长度，k为second字符长度* **for** (**unsigned int** j = i+1; j < i+k; j++) {  
 first[j]=second[j-i];  
 }  
 **return** first;*//但是这样改变了原始数据*}  
**void** strcat\_malloc(){  
 **char** \*s1,\*s2;  
 s1=(**char**\*)malloc(50\* **sizeof**(**char**));  
 s2=(**char**\*)malloc(50\* **sizeof**(**char**));  
 puts(**"First one:"**);gets\_s(s1,50);*//初始化s1* puts(**"Second one:"**);gets\_s(s2,50);*//初始化s2* **int** str1st=strlen(s1),str2nd=strlen(s2);  
 **int** strall = str1st+str2nd;  
 s1=(**char**\*)realloc(s1,strall \* **sizeof**(**char**));*//改变s1大小* **for**(**int** i=str1st;i<strall;i++)*//此处和字符处理相同* s1[i]=s2[i-str1st];  
 s1[strall]=**'\0'**;*//终止字符* puts(s1);  
 puts(**"======================"**);  
*//****todo:bugs free(s1);free(s2);***}  
**void** strcat\_file(){*//文件处理方法* **int** ch;  
 FILE \*fp;errno\_t err;  
 **if** ((err = fopen\_s(&fp, **"Temp.txt"**, **"w+"**)) !=0){  
 printf(**"Unable to write the file.\n"**); *//如果打不开，直接结束* exit(0); *//end program* }  
 puts(**"Input First string:"**);  
 ch=getchar();*//initiate ch* **while**(ch!=**'\n'**){ *//end when input '\0'* fputc(ch,fp); *//向磁盘文件输出一个字符* ch=getchar();  
 }  
 puts(**"Input Second string:"**);  
 ch=getchar();  
 **while**(ch!=**'\n'**){ *//end when input '\0'* fputc(ch,fp); *//向磁盘文件输出一个字符* ch=getchar();  
 }  
 fclose(fp); *//关闭文件* system(**"Temp.txt"**);*//发出dos指令，打开该文件*}  
**char** \*strcat\_str(**char** \*a,**char** \*b){*//字符串处理方法* **char** \*p=a,\*q=b;  
 **while** (\*p)p++;*//找到p的末尾* **while** (\*q)\*p++=\*q++;*//从p的末尾开始吧q逐一赋给p* \*p=**'\0'**;*//p的末尾为\0  
 //上面2句可压缩成：while ((\*p++ = \*q++) != '\0');* **return** a;*//p和a是一样的，返回a；*}  
*//char \*strcat\_str(char \*a,char \*b)\_\_attribute\_\_((\_\_nonnull\_\_(1, 2)));//1,2个参数自动判断非空***int** main(**void**) {  
 **char** first[160],second[160];  
 puts(**"This is a program that connect 2 string, now please input 2 strings:\n"  
 "======================\n"  
 "First one:"**);  
 gets\_s(first,159);  
 puts(**"Second one:"**);  
 gets\_s(second,159);  
 printf\_s(**"Array=%s\nPoint=%s\n======================\n"**,strcat\_str(first,second),strcat\_array(first,second));  
 puts(**"Now try files:"**);*//第4种* strcat\_file();  
 puts(**"======================\nNow try malloc:"**);*//第3种* strcat\_malloc();  
 **return** 0;  
}

运行结果：





1. **第7章第10题**

题目：写1个函数，输入1行英文字符串，将其中最长的单词输出

思路：输入一个字符串，用预先定义的数组储存，前面加1个空格（方便后续处理）后传递给1个指针函数，然后从第二项开始逐一判断其每一个字符，用一结构体记录单词长度和单词首字母起始内存位置

是否为英文及其缩写符号“’”？

是：是否前一项也为英文？

是：说明为单词，延申单词长度

否：说明为单词开始，执行长度初始化和记录地址

否：是否前一项不为英文？

是：说明很可能是句子或段落中间的部分，不理会，继续扫描下一个

否：说明是单词末尾，断掉计数，从下一项开始

统计结构体中单词长度的最大值，打印最大长度，用指针打印结果

**注：出于精力，难免没考虑用户不符合语法规则故意输错的情况，如果出现，可能程序错误**

源代码：

#include **<stdio.h>**#include **<assert.h>**

**struct** Word{  
 **char** \*start\_position;  
 **int** length;  
};  
**struct** Word Wordeach[1000];**int** Wordeach\_unit=0;  
**int** inalphabet(**char** check\_char){*//检查是否为字母和缩写字符“’”,下次改进时要考虑单词缩写情况，此处把缩写算在整个单词内* **if**((check\_char>=**'A'**&& check\_char<=**'Z'**)||(check\_char>=**'a'**&&check\_char<=**'z'**)||check\_char==**'\''**)**return** 1;  
 **else return** 0;  
}  
**void** get\_word\_length(**char** \*a) {  
 **assert**(a != NULL);  
 **char** \*scanword = a;scanword++;*//从" "中的“ ”开始* **int** scancount = 0;  
 **int** acRAM = 0;  
 **while** (\*scanword != **'\0'**) {  
 **if**(inalphabet(\*scanword)==1){*//是字母的情况* **if**(inalphabet(\*(scanword-1))==0){*//上一个不是字母，说明是单词开始* Wordeach[scancount].length = 1;*//初始化* Wordeach[scancount].start\_position = a+acRAM+1;*//记录此时单词的起始点,注意不能用函数内消亡的变量  
//debug printf\_s("acRAM=%d,Wordeach[%d].start\_position=%x\n",acRAM,scancount);* } **else** Wordeach[scancount].length++;*//上一个是字母，单词延续* } **else**{*//不是字母的情况* **if**(inalphabet(\*(scanword-1))==0){*//上一个不是字母，可能是句子结束如“! ”* ;  
 } **else**{*//上一个是字母，说明是两单词间隔或者句子结尾如“e!”* scancount++;  
 }  
 }  
*//****todo:bug, 逻辑错误****//* ***if ((\*scanword == ' ' || \*scanword == '\t' || \*scanword == '\n')) {//句子,段落开始计数,判断规则：此处为空格，前一处不为字母****//* ***//这是两个单词间隔1个空格的情况****//* ***Wordeach[scancount].length = 0;//初始化****//* ***Wordeach[scancount].start\_position = a+acRAM;//记录此时单词的起始点,注意不能用函数内消亡的变量****//* ***printf\_s("acRAM=%d,Wordeach[%d].start\_position=%x\n",acRAM,scancount,Wordeach[scancount].start\_position);****//* ***scancount++;****//* ***}else if (inalphabet(\*scanword)){//为字母，开始计数****//* ***Wordeach[scancount].length++;****//* ***}else if (inalphabet(\*scanword = 0) && \*scanword!='\''){//非字母外的字符,终止计数,且不为缩写或‘s的情况****//* ***scancount++;//终止上一个，开始下一个****//* ***}*** scanword++;  
 acRAM++;  
 }  
 Wordeach\_unit = scancount + 1;*//scancount在函数内会释放，用外部变量储存*}  
**void** print\_longest\_word(**void**){  
 **int** longest\_word\_length = Wordeach[0].length, longest\_i = 0;  
 **for** (**int** i = 1; i < Wordeach\_unit; ++i) {*//得到最大的单词在数组中的位置* **if** (longest\_word\_length < Wordeach[i].length) {  
 longest\_word\_length = Wordeach[i].length;  
 longest\_i = i;  
 }  
 }  
 puts(**"Longest word:"**);  
 printf\_s(**"length=%d,"**, longest\_word\_length);  
  
 **for** (**int** j = 0; j < longest\_word\_length; ++j) {  
 printf\_s(**"%c"**,\*Wordeach[longest\_i].start\_position);  
 Wordeach[longest\_i].start\_position++;  
 }  
puts(**""**);  
}  
**int** main(**void**){  
 puts(**"This is a program that print the longest word from string you've input:"**);  
 **char** getstr[999];gets\_s(getstr,998);  
 **char** bkstr[1000];bkstr[0]=**' '**;*//在getstr前加上一个空格，方便处理* **for** (**int** i = 0; i < 999; ++i) {  
 **if**(getstr[i]!=**'\0'**)bkstr[i+1]=getstr[i];  
 **else if**(getstr[i]==**'\0'**){  
 bkstr[i+1]=getstr[i];  
 **break**;*//结束，赋值结尾后跳出循环* }  
 }  
 puts(**"Counting...\n================================================"**);  
 get\_word\_length(bkstr);  
 **for** (**int** j = 0; j < Wordeach\_unit; ++j) {  
 printf\_s(**"Word%d,length=%d,position=%x\n"**,j+1,Wordeach[j].length,Wordeach[j].start\_position);  
 }  
 puts(**"================================================"**);  
 printf\_s(**"Words number=%d\n"**,Wordeach\_unit);  
 print\_longest\_word();  
 **return** 0;  
}

运行结果：

