**作 业 报 告**

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

|  |  |
| --- | --- |
| **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11月1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 1809  
//Author: Sean Peldom Zhang  
//Create Time: 2019/11/10-2019/10/20*

1. **作业要求**：

任选教材第8章课后4道题，完成并打印或手抄，要求每道题使用指针。

1. **第8章第16题**

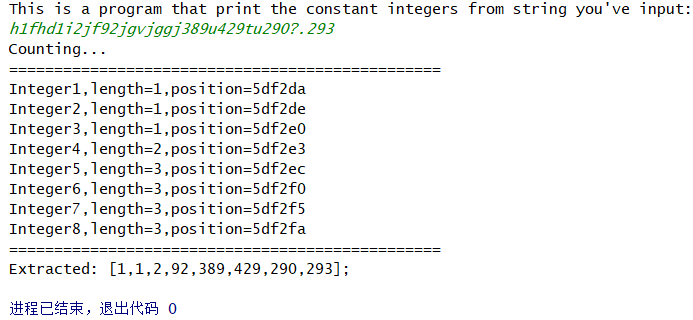
题目：输入一个含有各种字符的字符串，把其中的数字放进1个整型输出，输出其中的连续整数

思路1：典型的字符串的阅读题，模仿第7章第10题，稍加更改，加上转换字符到整型的函数，即可

源代码1(72行)：

#include **<stdio.h>**#include **<math.h>**#include **<assert.h>  
struct** Int{  
 **char** \*start\_position;  
 **int** length;  
};  
**struct** Int Inteach[1000];**int** Inteach\_unit=0;  
**int** int\_counts[1000]={0};  
**int** is\_int(**char** check\_char){*//检查是否为数字* **if**(check\_char>=**'0'**&& check\_char<=**'9'**)**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**(is\_int(\*scanword)==1){*//是数字的情况* **if**(is\_int(\*(scanword-1))==0){*//上一个不是数字，说明是连续数字开始* Inteach[scancount].length = 1;*//初始化* Inteach[scancount].start\_position = a+acRAM+1;*//记录此时连续的起始点* } **else** Inteach[scancount].length++;*//上一个是数字，数字延续* } **else**{*//不是字母的情况* **if**(is\_int(\*(scanword-1))==0){*//上一个不是数字* ;  
 } **else**{*//上一个是数字，说明是两数字间隔* scancount++;  
 }  
 }  
 scanword++;  
 acRAM++;  
 }  
 Inteach\_unit = scancount + 1;*//scancount在函数内会释放，用外部变量Inteach\_unit储存数字数量*}  
**void** fill\_int\_counts(){*//填充提取出的数字字符，并转化成int型储存在int\_counts[]中* **int** fill\_int\_countor=0;  
 **for** (**int** i = 0; i < Inteach\_unit; ++i) {  
 **for** (**int** j = 0; j < Inteach[i].length; ++j) {  
 int\_counts[fill\_int\_countor]+=(\*Inteach[i].start\_position-**'0'**)\*(**int**)pow(10,Inteach[i].length-j-1);  
 Inteach[i].start\_position++;  
 }  
 fill\_int\_countor++;*//计算下一个元素* }  
 printf\_s(**"Extracted: ["**);*//print the initiated origin\_array* **for**(**int** i=0;i<Inteach\_unit;i++){  
 printf\_s(**"%d"**,int\_counts[i]);  
 **if**(i==Inteach\_unit-1)printf\_s(**"];\n"**);  
 **else** printf\_s(**","**);  
 }  
}  
**int** main(**void**){  
 puts(**"This is a program that print the constant integers from string you've input:"**);  
 **char** getstr[998];gets\_s(getstr,997);  
 **char** bkstr[1000];bkstr[0]=**' '**;*//在getstr前加上一个空格，方便处理* **for** (**int** i = 0; i < 998; ++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 < Inteach\_unit; ++j) {  
printf\_s(**"Integer%d,length=%d,position=%x\n"**,j+1,Inteach[j].length,Inteach[j].start\_position);  
 }  
 puts(**"================================================"**);  
 fill\_int\_counts();  
 **return** 0;  
}

运行结果：

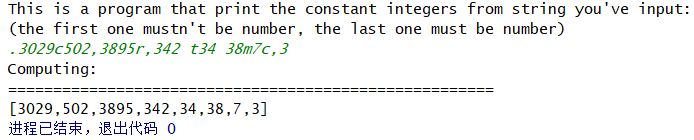


思路2：狂暴法，只要稍稍分析字符串，不用赋值直接输出，哈哈哈；

源代码2（19行）：

#include **<stdio.h>**#include **<string.h>  
int** isnum(**char** n){  
 **if**(n>=**'0'**&&n<=**'9'**)**return** 1;  
 **else return** 0;  
}  
**int** main(**void**){  
 puts(**"This is a program that print the constant integers from string you've input:\n"  
 "(the first one mustn't be number, the last one must be number)"**);  
 **char** store\_str[100];gets\_s(store\_str,100);*//忽略第1个*puts(**"Computing:\n======================================================"**);  
 printf\_s(**"["**);  
 **for** (**unsigned int** i = 1; i <strlen(store\_str) ; ++i) {  
 **if**(isnum(store\_str[i]))putchar(store\_str[i]);  
 **else if**(isnum(store\_str[i-1]))putchar(**','**);  
 }  
 printf\_s(**"]"**);  
 **return** 0;  
}

运行结果：



1. **第8章第4题**

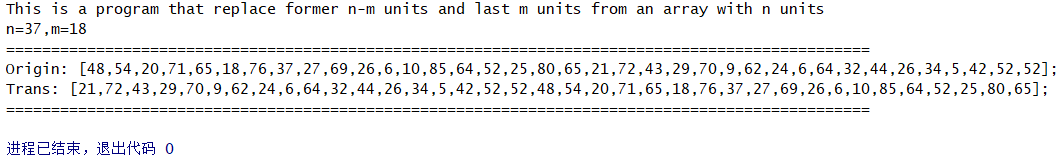
题目：有n个整数（n未知）把后面m个数移到前面，前面m-n个数移到后面；

程序思路：准备一个大空间，随机生成m，n（1<=m<n<=50），然后用指针帮助赋值到新的数组即可

源代码(59行)：

#include **<stdio.h>**#include **<stdlib.h>**#include **<time.h>**#include **<assert.h>  
int** origin[51]={0},trans[51]={0};*//more than 1 '0' inside***int** \*origin\_point=origin;**int** \*trans\_point=trans;  
**int** get\_less\_random\_m(**int** random\_n){  
 srand((**unsigned**)(time(**NULL**)));  
 **int** random\_m;  
 **do**{  
 random\_m=rand()%100+1;  
 **if**(random\_m<random\_n)**break**;  
 }**while** (random\_m>=random\_n);  
 **return** random\_m;  
}  
**void** transplant\_array(**int** random\_n,**int** random\_m){  
 **assert**(random\_m<random\_n);*//theoretically m<n* **for** (**int** i = 0; i < random\_n-random\_m; ++i) {  
 origin\_point++;  
 }*//read from m* **while** (\*origin\_point)\*trans\_point++=\*origin\_point++;*//assign m one by one* **for** (**int** j = 0; j < random\_n; ++j) {*//return origin\_point to the beginning* origin\_point--;  
 }  
 **for** (**int** k = 0; k < random\_n-random\_m; ++k) {*//assign n one by one* \*trans\_point++=\*origin\_point++;  
 }  
}  
**int** main(**void**){  
 puts(**"This is a program that replace former n-m units and last m units from an array with n units"**);  
  
 srand((**unsigned**)(time(**NULL**)));  
 **int** random\_n=(rand()%100+1)/4+25;*//25<=n<=50* **int** random\_m=get\_less\_random\_m(random\_n);*//1<=m<n* printf\_s(**"n=%d,m=%d\n"**,random\_n,random\_m);  
puts(**"================================================================================================"**);  
 **for** (**int** i = 0; i < random\_n; ++i) {  
 origin[i]=rand()%100+1;  
 }*//initiate origin\_array* printf\_s(**"Origin: ["**);*//print the initiated origin\_array* **for**(**int** i=0;i<random\_n;i++){  
 **if**(origin[i]!=0)printf\_s(**"%d"**,origin[i]);  
 **if**(i==random\_n-1)printf\_s(**"];\n"**);  
 **else** printf\_s(**","**);  
 }  
 transplant\_array(random\_n,random\_m);  
 printf\_s(**"Trans: ["**);*//print the transplanted origin\_array* **for**(**int** i=0;i<random\_n;i++){  
 **if**(origin[i]!=0)printf\_s(**"%d"**,trans[i]);  
 **if**(i==random\_n-1)printf\_s(**"];\n"**);  
 **else** printf\_s(**","**);  
 }  
puts(**"================================================================================================"**);  
 **return** 0;  
}

运行结果：



1. **第8章第15题**

题目：生信1802班(void)有10个学生，经历了5门期末考试，现在得到成绩（51~100），分析成绩：

1. 输出每门课的平均分；
2. 输出全班最高分，和该科目，与该同学的姓名；
3. 找到挂科多的人(>=2门)，输出其姓名；
4. 找到学霸（没挂科，平均分>80分）；

思路：问题很简单，就是数据结构定义写起来比较多，所以这里用指针字符串的数组来简化一点点；

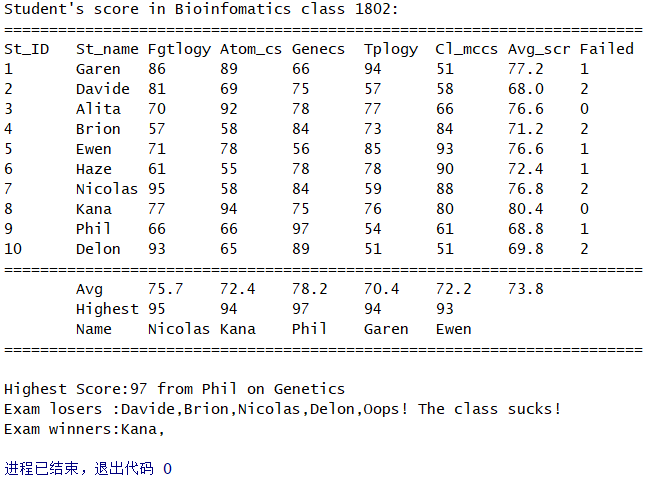
源代码（183行）：

#include **<stdio.h>**#include **<stdlib.h>**#include **<time.h>  
struct** student\_subjuct{  
 **int** Forgettenlogy;  
 **int** Atomic\_physics;  
 **int** Genetics;  
 **int** Topology;  
 **int** Celestial\_mechanics;  
};  
**struct** student\_score{  
 **int** St\_ID;  
 **char** \*pname;  
 **struct** student\_subjuct subjuct;  
 **double** avg\_score;  
 **unsigned int** failed;  
}myclass[10];  
**struct** myclass\_subject{  
 **char** \*subjuct\_name;  
 **double** subject\_avg;  
 **char** \*highest\_st\_name;  
 **int** highest;  
}mysubjects[5];  
**char** \*point\_name[10]={**"Garen"**,**"Davide"**,**"Alita"**,**"Brion"**,**"Ewen"**,**"Haze"**,**"Nicolas"**,**"Kana"**,**"Phil"**,**"Delon"**};  
**char** \*point\_subject[5]={**"Forgettenlogy"**,**"Atomic\_physics"**,**"Genetics"**,**"Topology"**,**"Celestial\_mechanics"**};  
**int** failed\_exam(**int** n){  
 **if**(n<60)**return** 1;  
 **else return** 0;  
}  
**void** initiate\_myclass(**void**){*//随机生成其分数值，并计算挂科数* srand((**unsigned**)(time(**NULL**)));  
 printf\_s(**"St\_ID\t"  
 "St\_name\t"  
 "Fgtlogy\t"  
 "Atom\_cs\t"  
 "Genecs\t"  
 "Tplogy\t"  
 "Cl\_mchcs\t"  
 "Avg\_scr\t"  
 "Failed\n"**);  
 **for** (**int** i = 0; i < 10; ++i) {  
 myclass[i].St\_ID=i+1;  
 myclass[i].failed=0;  
 myclass[i].pname=point\_name[i];  
myclass[i].subjuct.Forgettenlogy=rand()%50+51;**if**(failed\_exam(myclass[i].subjuct.Forgettenlogy))myclass[i].failed++;  
myclass[i].subjuct.Atomic\_physics=rand()%50+51;**if**(failed\_exam(myclass[i].subjuct.Atomic\_physics))myclass[i].failed++;  
myclass[i].subjuct.Genetics=rand()%50+51;**if**(failed\_exam(myclass[i].subjuct.Genetics))myclass[i].failed++;  
myclass[i].subjuct.Topology=rand()%50+51;**if**(failed\_exam(myclass[i].subjuct.Topology))myclass[i].failed++;  
myclass[i].subjuct.Celestial\_mechanics=rand()%50+51;**if**(failed\_exam(myclass[i].subjuct.Celestial\_mechanics))myclass[i].failed++;  
 myclass[i].avg\_score=(myclass[i].subjuct.Forgettenlogy+  
 myclass[i].subjuct.Atomic\_physics+  
 myclass[i].subjuct.Genetics+  
 myclass[i].subjuct.Topology+  
 myclass[i].subjuct.Celestial\_mechanics)/5.0;  
 printf\_s(**"%d\t%s\t%d\t%d\t%d\t%d\t%d\t%.1lf\t%d\n"**,  
 myclass[i].St\_ID,  
 myclass[i].pname,  
 myclass[i].subjuct.Forgettenlogy,  
 myclass[i].subjuct.Atomic\_physics,  
 myclass[i].subjuct.Genetics,  
 myclass[i].subjuct.Topology,  
 myclass[i].subjuct.Celestial\_mechanics,  
 myclass[i].avg\_score,  
 myclass[i].failed);  
 }  
}  
**void** initiate\_mysubject(**void**){ *//计算班级平均分和最高分*  
 **for** (**int** i = 0; i < 5; ++i) {  
 mysubjects[i].subjuct\_name=point\_subject[i];  
 mysubjects[i].subject\_avg =0;*//初始化avg* }  
 **for** (**int** j = 0; j < 10; ++j) {  
 mysubjects[0].subject\_avg +=((myclass[j].subjuct.Forgettenlogy)/10.0);  
 mysubjects[1].subject\_avg +=((myclass[j].subjuct.Atomic\_physics)/10.0);  
 mysubjects[2].subject\_avg +=((myclass[j].subjuct.Genetics)/10.0);  
 mysubjects[3].subject\_avg +=((myclass[j].subjuct.Topology)/10.0);  
 mysubjects[4].subject\_avg +=((myclass[j].subjuct.Celestial\_mechanics)/10.0);  
 **if**(j==0){*//initiate 5 highests* **for** (**int** i = 0; i < 5; ++i) {  
 mysubjects[i].highest =10\*mysubjects[i].subject\_avg;  
 mysubjects[i].highest\_st\_name=point\_name[0];*//supposing Garen* }  
 }  
 **if**(mysubjects[0].highest<myclass[j].subjuct.Forgettenlogy){  
 mysubjects[0].highest =myclass[j].subjuct.Forgettenlogy;  
 mysubjects[0].highest\_st\_name=point\_name[j];  
 }  
 **if**(mysubjects[1].highest<myclass[j].subjuct.Atomic\_physics){  
 mysubjects[1].highest =myclass[j].subjuct.Atomic\_physics;  
 mysubjects[1].highest\_st\_name=point\_name[j];  
 }  
 **if**(mysubjects[2].highest<myclass[j].subjuct.Genetics){  
 mysubjects[2].highest =myclass[j].subjuct.Genetics;  
 mysubjects[2].highest\_st\_name=point\_name[j];  
 }  
 **if**(mysubjects[3].highest<myclass[j].subjuct.Topology){  
 mysubjects[3].highest =myclass[j].subjuct.Topology;  
 mysubjects[3].highest\_st\_name=point\_name[j];  
 }  
 **if**(mysubjects[4].highest<myclass[j].subjuct.Celestial\_mechanics){  
 mysubjects[4].highest =myclass[j].subjuct.Celestial\_mechanics;  
 mysubjects[4].highest\_st\_name=point\_name[j];  
 }  
 }  
}  
**void** analysissb(**void**){*//这里用循环可能可以精简代码*

本来想先随机生成一个.csv，但发现C语言居然没有专门处理csv的库函数，外部库也没有很好的支持，使得读起来非常麻烦，还不如直接在内存里生成

printf\_s(**"\tAvg\t%.1lf\t%.1lf\t%.1lf\t%.1lf\t%.1lf\t%.1lf\n"**,  
 mysubjects[0].subject\_avg,  
 mysubjects[1].subject\_avg,  
 mysubjects[2].subject\_avg,  
 mysubjects[3].subject\_avg,  
 mysubjects[4].subject\_avg,  
 (mysubjects[0].subject\_avg+  
 mysubjects[1].subject\_avg+  
 mysubjects[2].subject\_avg+  
 mysubjects[3].subject\_avg+  
 mysubjects[4].subject\_avg)/5.0);  
 printf\_s(**"\tHighest\t%d\t%d\t%d\t%d\t%d\n"**,  
 mysubjects[0].highest,  
 mysubjects[1].highest,  
 mysubjects[2].highest,  
 mysubjects[3].highest,  
 mysubjects[4].highest);  
 printf\_s(**"\tName\t%s\t%s\t%s\t%s\t%s\n"**,  
 mysubjects[0].highest\_st\_name,  
 mysubjects[1].highest\_st\_name,  
 mysubjects[2].highest\_st\_name,  
 mysubjects[3].highest\_st\_name,  
 mysubjects[4].highest\_st\_name);  
}  
**void** findhighest(**void**){  
 **char** \*highest\_st\_name=mysubjects[0].highest\_st\_name;  
 **int** highest\_score=mysubjects[0].highest;  
 **char** \*highest\_subject=point\_subject[0];  
 **for** (**int** i = 1; i < 5; ++i) {  
 **if**(highest\_score<mysubjects[i].highest){  
 highest\_score=mysubjects[i].highest;  
 highest\_st\_name=mysubjects[i].highest\_st\_name;  
 highest\_subject=mysubjects[i].subjuct\_name;  
 }  
 }  
 printf\_s(**"\nHighest Score:%d from %s on %s\n"**,highest\_score,highest\_st\_name,highest\_subject);  
}  
**void** find\_exam\_losers(**void**){  
 printf\_s(**"Exam losers :"**);  
 **int** losers=0;  
 **for** (**int** i = 0; i < 10; ++i) {  
 **if**(myclass[i].failed>=2){  
 printf\_s(**"%s,"**,myclass[i].pname);  
 losers++;  
 }  
 }  
 **if**(losers>=4){puts(**"Oops! The class sucks!"**);**return**;}  
 puts(**""**);  
}  
**void** fina\_exam\_winners(**void**){  
 printf\_s(**"Exam winners:"**);  
 **int** winners=0;  
 **for** (**int** i = 0; i < 10; ++i) {  
 **if**(myclass[i].avg\_score>80 && myclass[i].failed==0){  
 printf\_s(**"%s,"**,myclass[i].pname);  
 winners++;  
 }  
 }  
 **if**(winners==0){puts(**"Oops! There isn't any exam winners!"**);**return**;}  
 puts(**""**);  
}  
**void** putpartingline(){  
 puts(**"======================================================================="**);  
}  
**int** main(**void**) {  
 puts(**"Student's score in Bioinfomatics class 1802:"**);  
 putpartingline();  
 initiate\_myclass();  
 initiate\_mysubject();  
 putpartingline();  
 analysissb();  
 putpartingline();  
 findhighest();  
 find\_exam\_losers();  
 fina\_exam\_winners();  
 **return** 0;  
}

运行结果：



1. **第8章第5题**

题目：n个人围成1圈，开始报数；凡报到1个特定数的人退出，求最后的1位

思路：使用链表(第1次用，多写点注释)，循环扫描，每次遇到清除该人的节点，然后重置上1节点的指针域，指向原来的下个节点；

源代码：

#include **<stdio.h>**#include **<stdlib.h>**#include **<assert.h>  
typedef struct** node{  
 **unsigned int** number;*//people's ID* **struct** node \*next;  
}person;  
person \*initLink(**unsigned int** n){*//初始化链表* **assert**(n>=2);  
 person \*head=(person\*)malloc(**sizeof**(person));*//头节点指针* head->number=1;  
 head->next=**NULL**;  
 person \*cyclic=head;*//用于循环的指针* **for** (**unsigned int** i=2; i<=n; i++) {*//定义中间节点* person \*body=(person\*)malloc(**sizeof**(person));  
 body->number=i;  
 body->next=**NULL**;  
 cyclic->next=body;  
 cyclic=cyclic->next;  
 }  
 cyclic->next=head;*//首尾相连* **return** head;  
}  
**void** find\_and\_kill\_K(person \*head,**unsigned int** k,**unsigned int** m){  
 person \*tail=head;*//找到链表第一个结点的上一个结点，为删除操作做准备* **while** (tail->next!=head) {*//循环到尾节点* tail=tail->next;  
 }  
 person \*p=head;*//从头开始找到编号为k的人* **while** (p->number!=k) {  
 tail=p;  
 p=p->next;  
 }*//从编号为k的人开始，只有符合p->next==p时，说明链表中除了p结点，所有编号都出列了* **while** (p->next!=p) {*//找到从p报数1开始，报m的人，并且还要知道数m-1的人的位置tail，方便做删除操作* **for** (**unsigned int** i=1; i<m; i++) {  
 tail=p;  
 p=p->next;  
 }  
 tail->next=p->next;*//从链表上将p结点摘下来* printf\_s(**"Removing player:%d\n"**,p->number);  
 free(p);*//把该人节点kill掉* p=tail->next;*//继续使用p指针指向出列编号的下一个编号，循环继续* }  
 printf\_s(**"Winner:%d\n"**,p->number);  
 free(p);*//最后1个*}  
**void** puts\_line(){  
 puts(**"======================================="**);  
}  
**int** main(**void**) {  
 puts(**"-This is a program that plays a game:\n-Several players surround a table, players number off one by one.\n-Who numbered a predetermined number will be removed out.\n-And it turns over and over again to the last player(winner)."**);  
 printf\_s(**"Input players number:\n"**);  
 **unsigned int** n;scanf\_s(**"%d"**,&n);  
 person \*head=initLink(n);  
 printf\_s(**"Input a integer, will start from the ranked player(1<x<%d):\n"**,n);  
 **unsigned int** k;scanf\_s(**"%d"**,&k);**assert**(k>=1);  
 printf\_s(**"Input a integer, player who counts it will be removed:\n"**);  
 **unsigned int** m;scanf\_s(**"%d"**,&m);**assert**(m<n);  
 puts\_line();  
 find\_and\_kill\_K(head, k, m);  
 puts\_line();  
 **return** 0;  
}

运行结果：

