**2015年复试C语言（伊诺吐血回忆版）**

一、简答题

1. 读程序写结果（参考Dr. 刘）

void main()

{

char p[10]="abc";

char q[]="xyz";

int i,j;

i=0;

while(\*(p+i)!='\0') i++;

j=0;i++;

while(\*(q+j)!='\0')

{

\*(p+i)=\*(q+j);

j++;

}

printf("%s", p);

}

1. 给出1-x+x^2/2! -x^3/3! + ... + x^n/n!的暴力解的代码，问该代码的功能，并分析其效率有啥改进之处

注：到底负号在哪我记不清楚了，反正考完了，这也对后面的学弟学妹们没有影响

所谓的改进方法就是用动态规划的思想啦。那么大家肯定要问啥是动态规划听起来好厉害的样子啊快来围观这只可耻卖萌的小兔子啊啊啊啊blabla。。其实大家不用怕，动态规划其实是一个偷懒的概念，其核心思想就是算过的就不再算了，哈哈，是不是一个偷懒的想法呀~为了说明白动态规划，我先说说这道题的暴力解法。

暴力解法：你看，题目里面要求x的n次方和n的阶乘，那么根据模块化的思想，我们可以专门写出两个函数分别来计算x的n次方（如函数pow(x,n)）和n的阶乘（如函数jc(n)），然后遍历求出每一项的和，再相加，便是答案。

伪代码如下,sum为结果：

for(i=1; i<=n; i++)

{

sum += flag\* pow(x,i)/jc(i);

flag \*= -1;

}

动态规划优化解：

看了上面的暴力解之后，不知道有没有发现，求n!之前(n-1)!的结果已经求过了，求x^n之前x^(n-1)已经求过了，又根据n!=(n-1)! \* n和x^n= x^(n-1) \* x两个基本的数学公式，动态规划的思路就出来了。我们要定义一个变量，将之前计算过的值保存下来，防止重复计算。伪代码如下（初始化sum和item都为1）：

for(i=1; i<=n; i++)

{

item \*= -1\*x/i;

sum += item;

}

1. 请用for(i=1; i<n; i++)S;显式结构语言表示该程序段，用伪代码写出来(注明条件跳转和强制跳转)。

其实这类题我原本不会做的，但是在考前那天下午我在水贴，然后陈奕迅发了一张黄迪明书上这类题的解法，我就手贱点开看了一下，第二天就考到了。可怜认真学习一天没水贴的老人233333333333333

i=1

S\_for:

if i<n goto S\_do

goto S\_exit

S\_do:

S

i=i+1

goto S\_for

S\_exit:

4. 根据下面的代码，填写表格。整数算2字节，字符1字节，指针4字节。每个区域的起始地址都是0，内存按2字节编址。

int num=2;

void main()

{

char str1[10]={"UESTC"};

char \*str2="CHENGDU";

char p;

}

void func(int m)

{

int n=10;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 内存区域 | 常量或变量名 | 占用内存大小 | 内存相对地址 |
| 常量区 | 2 | 2 | 0 |
| 10 | 2 | 1 |
| "UESTC" | 6 | 2 |
| "CHENGDU" | 8 | 5 |
| 10 | 2 | 9 |
| 全局 | num | 2 | 0 |
| main函数 | str1 | 10 | 0 |
| str2 | 4 | 5 |
| p | 1 | 7 |
| func函数 | m | 2 | 0 |
| n | 2 | 1 |

这道题难点有二：

1. char str1[10]={"UESTC"}; "UESTC"算字符串常量，但是str1对应的内存要占用10 字节。
2. void func(int m) 形参也是要占用内存的，在函数被调用时申请内存，在函数结束时撤销。

二、填空题

有四道题，每空3分，共72分，共有四道题，分别是求质因数、删除字符串中的英文和数字并将重复字符减为一个、约瑟夫环、链表逆序。

这里建议大家将上面的问题自己编程求解，做到这一步就可以了，因为就算你会写了也不一定上考场会做，因为题目给出的代码不是一个正常人能想到的代码。。。比如链表逆序大家马上想到头插法，但是题目给出代码确是一个算法极其糟糕、时间空间复杂度极其低下的还有一些错误的算法——所以，做这种题的思路就是先把会的做了，不会的到时候想办法也要写上一个不太离谱的答案。

三、编程题（3\*20=60分）

编程一：编写一个函数，从字符串中寻找整个连续的数字字符，将其转化成整数并保存在arr整型数组里，溢出以-1作为标志。字符串以‘#’结束（如"uestc2015jsj123#"）

void int\_find(int arr[], char \*pc)

int str2int(char \*str, int len)

{

int i, num=0;

for(i=len-1; i>=0; i--)

num += (str[len-i-1]-48)\*pow(10.0, i);

return num;

}

void int\_find(int arr[], char \*pc)

{

char \*p=pc, \*q;

int i=0;

while(\*p!='#')

{

if('0'<=\*p && \*p <= '9')

{

q=p+1;

while('0'<=\*q && \*q <= '9')q++;

arr[i++] = str2int(p, q-p);

p=q;

}

else

{

p++;

}

}

arr[i]=-1;

}

编程二：

随机输入若干整数和若干浮点数（顺序也是随机的，最多100个），要求编写完整的程序，将整数按从大到小排列，浮点数按从小到大排列（浮点数排序可省略），并输出。如：输入10 12.3 12 5 52.1 3.65 88.6 输出：12 10 5 3.65 12.3 52.1 88.6

这道题我采用的是一种简单的判断输入的数是整数还是浮点数的方法，但是该算法有个缺陷就是会将10.0认作整数10。但是考虑在考试中时间有限无法使用其它耗时的方法（比如读取成字符串然后判断有无字符’.’从而判断是否为浮点数）。并且，在原题中给出的数字并未出现10.0这样的情况，故综上，我选择这样的方法（具体见代码）。

void sort\_int (int num[], int n)

{

int i, j;

for(i=0; i<n-1; i++)

{

for(j=i+1; j<=n-1; j++)

{

if(num[j]<num[i])

{

int temp = num[i];

num[i] = num[j];

num[j] = temp;

}

}

}

}

void main()

{

float temp;

int num\_int[100], count\_int=0, count\_foat=0;

float num\_float[100];

int i, n;

scanf("%d", &n);

for(i=0; i<n; i++)

{

scanf("%f", &temp);

if(temp == (int)temp)

num\_int[count\_int++]=(int)temp;

else

num\_float[count\_foat++]=temp;

}

sort\_int(num\_int, count\_int);

sort\_float(num\_float, count\_foat);

for(i=0; i<count\_int; i++)

printf("%d ", num\_int[i]);

for(i=0; i<count\_foat; i++)

printf("%f", num\_float[i]);

}

编程三：

编写完整的程序，构造整数集合(其实就是一个整数链表)，并实现对该集合操作的若干功能：加入一个新元素，判断某元素是否在集合内，输出两个集合并集，输出两个集合交集，删除集合中某一元素。

struct set{int numb; struct set \*next;}

注意：链表里是没有重复数字的，因为题目中说了是集合，集合是不会有重复元素出现的。此外，本题有时间空间复杂度更优的算法，但是在考试时写出这样的算法不太现实，故我这里提供的还是普通版的。

//加入一个新元素

void add(struct set \*L, int num)

{

struct set \*p=(struct set \*)malloc(sizeof(struct set));

p->numb=num;

p->next = L->next;

L->next = p;

}

//判断某元素是否在集合内

int isexist(struct set \*L, int num)

{

struct set \*p=L->next;

while(p)

{

if(p->numb == num)return 1;

p=p->next;

}

return 0;

}

//输出两个集合并集

void intersection(struct set \*a, struct set \*b)

{

struct set \*p=a->next;

int flag=0;

while(p)

{

printf("%d ", p->numb);

p=p->next;

}

struct set \*q=b->next;

while(q)

{

p=a->next;

flag=0;

while(p)

{

if(q->numb == p->numb)

{

flag=1;

break;

}

p=p->next;

}

if(!flag)printf("%d ", q->numb);

q=q->next;

}

}

//输出两个集合交集

void Union(struct set \*a, struct set \*b)

{

struct set \*p=a->next, \*q=b->next;

while(p)

{

q=b->next;

while(q)

{

if(p->numb == q->numb)

{

printf("%d ", q->numb);

break;

}

q=q->next;

}

p=p->next;

}

}

//删除集合中某一元素

void del(struct set \*a, int num)

{

struct set \*p=a->next, \*pre=a;

while(p)

{

if(p->numb == num)

{

pre->next=p->next;

free(p);

break;

}

pre=p;

p=p->next;

}

}

void main()

{

struct set \*L = (struct set \*)malloc(sizeof(struct set));

struct set \*S = (struct set \*)malloc(sizeof(struct set));

L->next=0;

S->next=0;

int temp;

scanf("%d", &temp);

while(temp!=-1)

{

add(L, temp);

scanf("%d", &temp);

}

scanf("%d", &temp);

while(temp!=-1)

{

add(S, temp);

scanf("%d", &temp);

}

printf("\n并集：");

intersection(L, S);

printf("\n交集：");

Union(L, S);

}