程序设计重点总结（我以为的重点）

1. 先是一些低级错误
2. scanf后面的取地址符“&”别漏（对于指针等本身就是地址的别忘省略）
3. 各种字符要用英文形式（考试是手写无所谓了）
4. 说完一句话别忘了“；”
5. if后面的条件不要把“==”写成了“=”
6. 一些基本语法逻辑
7. **一些表达的合法性**

1）**标识符**：合法的要求是由字母，数字，下划线组成，只能以字母或下划线开头。包含有其它元素或第一个为数字就是错误的。另外关键字不可以作为用户标识符号。

2）**实型数据**的合法形式：2.3e-1 就是合法的，且数据是2.3×10-1。（e 前e 后必有数，e 后必为整数）。

3）**字符数据**的合法形式:

'a' 是字符占一个字节，"a"是字符串占两个字节(含有一个串尾’\0’符号)。

'0' 的ASCII 数值表示为48，'a' 的ASCII 数值是97，'A'的ASCII 数值是65。

'65' "1" 都是表示单个字符的错误形式。

字符可以进行算术运算： '0' – 0 = 48， 即字符的ascii码参加算术运算。

大写字母和小写字母转换的方法： 'A'+32='a' 相互之间一般是相差32。

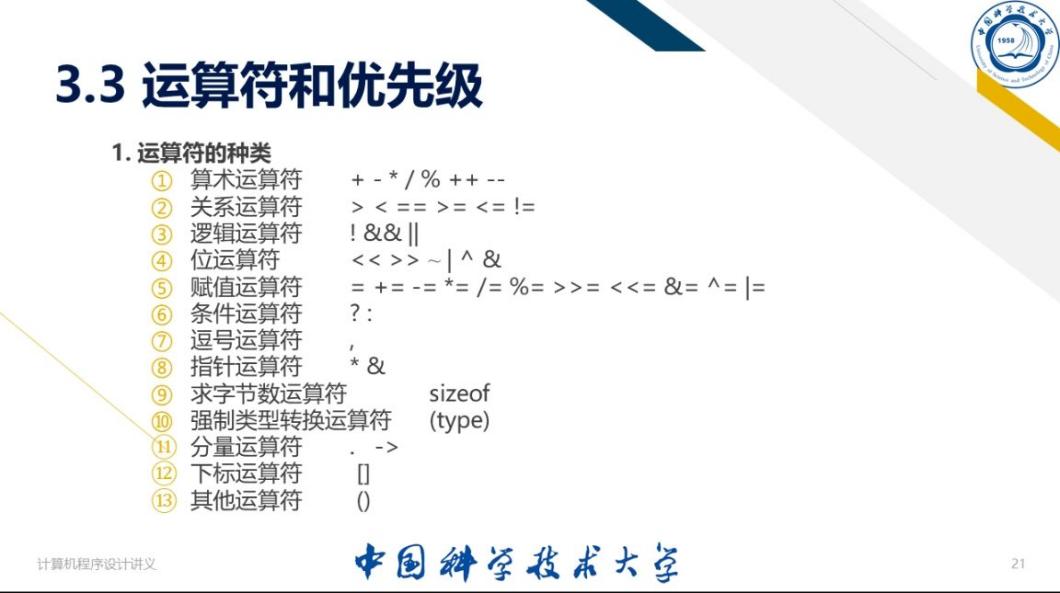
**2.转义字符**：

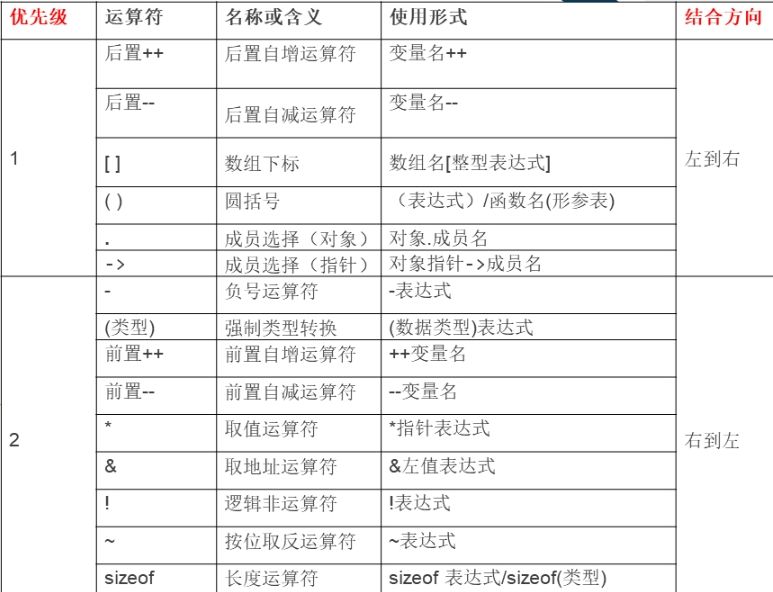
在程序中 int a = 0x6d，是把一个十六进制的常量给变量a 注意这里的0x 是前缀，必须存在。在程序中 int a = 077, 077是一个八进制的形式，0是前缀。

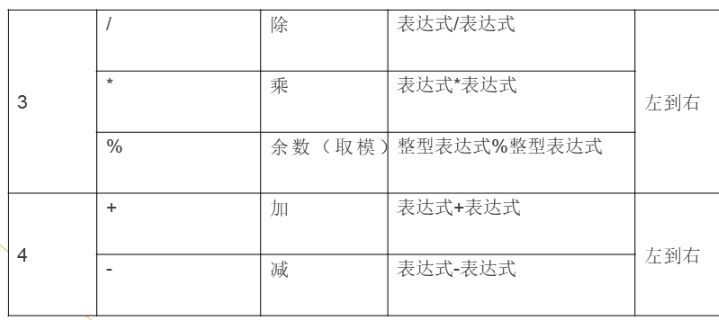
如果使用转义字符：在转义字符中(\ddd 8进制, \xhh 16进制，x 是小写 ), ‘\x6d’ 才是合法的，不能写’\0x6d’ 。‘\108’是非法的，因为8进制不可以出现8。

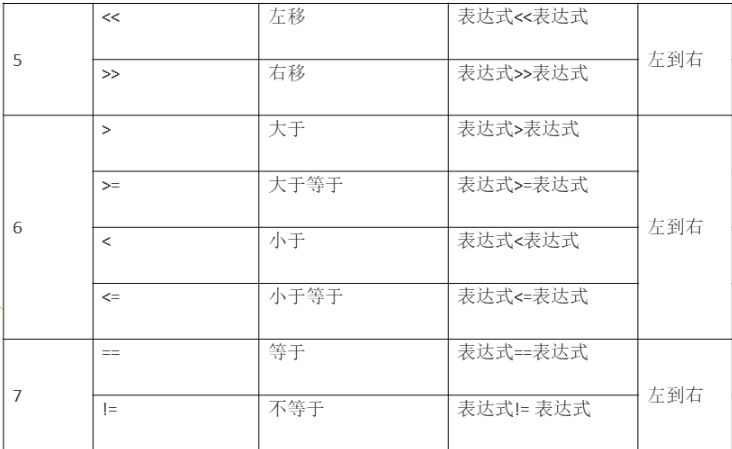
**3.运算符的优先级和结合方向：**

直接上PPT：

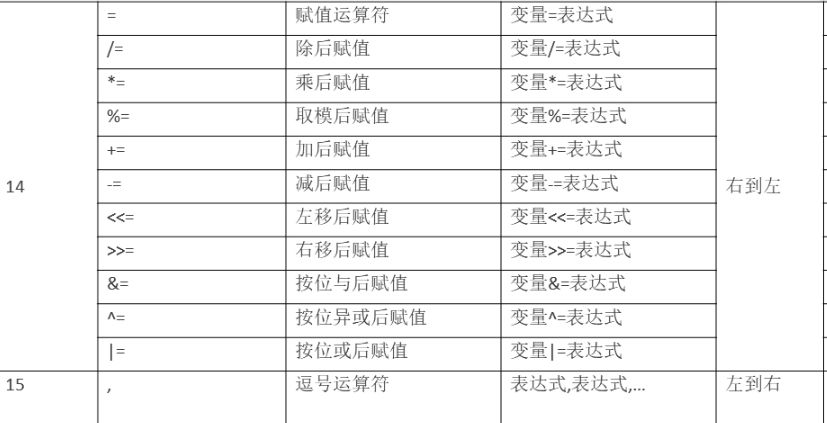












运算符的结合方向：

举例：a=b=1：从右向左，先计算b=1，再计算a=1；

规律：

①单目运算符（！，~，前置++，前置--，+，-，\*<指针运算符>,&<取地址运算符>，sizeof（type））的结合规律都是从右向左；

②赋值运算符的结合规律都是从右向左；（举例里的“=”就是赋值运算符，以及+=，-=......）

③三目运算符（“（表达式1）？（表达式2）：（表达式3）”，表达式1执行结果是真的，则执行表达式2，否则执行表达式3）的结合规律都是从右向左；

其他运算符结合方向都是从左向右。

可以用圆括号限定运算次序，（）的优先级最高。

**4.强制类型转换：**

一定是 （int）a 不是 int（a），注意是类型关键字在括号内。

注意（int）（a+b） 和（int）a+b 的区别。 前是把a+b 转型，后是把a 转型再加b。

**5.** **表达式的考查：☆**

是表达式就一定可求值！

**1）赋值表达式**：表达式数值是最左边的数值，a=b=5 ;该表达式为5，常量不可以被赋值。

**2）自加、自减表达式**：假设a=5，++a（表达式值为6）， a++（表达式值为5）；

运行的机理：++a 是先把变量的数值加上1，然后把得到的数值放到变量a 中，然后再用这个++a 表达式的数值为6，而a++是先用该表达式的数值为5，然后再把a 的数值加上1 为6，再放到变量a 中。 进行了++a 和a++后 在下面的程序中再用到a 的话都是变量a 中的6 了。

**3）逗号表达式**：优先级别最低 ； 表达式的数值逗号最右边的那个表达式的数值。（2，3，4）的表达式的数值就是4。

**6.%（取余）符号两边要求是整数。不是整数就错了。**

**7. 三种取整丢小数的情况**：

1）int a =1.6；

2）(int)a；

3）1/2； 3/2；

**8.字符型和整数基本通用：**

char a = 65 ;

printf(“%c”, a); 得到的输出结果：A

printf(“%d”, a); 得到的输出结果：65

**9.指针数组和数组指针**

指针数组：例子：char\* p[4]; char \*p[4]; p首先和[4]结合，说明是个数组，然后类型是char\*，说明数组的元素是指向字符串的指针。  
数组指针：例子：char (\*p) [4]; p与\*先结合，说明p是一个指针，然后指向的空间长度是4, 最后是char型，说明p是一个指向一维长度为4的字符数组的指针，p+1代表的是和p连续的下一个指向一维长度为4的字符数组的指针，所以可以代表二维数组。

指针数组是数组，数组元素的类型是指针；数组指针是指针，指向的是数组。不要弄混了。

指针数组，字符数组，整形数组，浮点数组这些是一类的，都是数组，数组前面的两个字是他们元素的类型

**10.结构体的使用**

1. 以下两种定义两个结构体变量方法等价

struct Student

{.....

.....}stu\_1,stu\_2;

和

struct Student

{.....

.....}

int main()

{

struct Student stu\_1,stu\_2;

}

1. 结构体指针的三种等价用法

假设：struct Student

{.....

.....};

struct Student stu;

struct Student \*p;

p=&stu\_1;

那么stu.成员名（如stu,num）等价于（\*p）.成员名（如（\*p）.num）等价于p->成员名（如p->num）.

1. 结构体中定义指针变量指向结构体变量，可以建立链表（链表这东西把数组指针结构体全都揉到一起）

如：struct Student

{.....

.....

struct Student\*next //next作为指针变量，指向结构体

};

**11.宏的使用**

求a,b最大值的宏举例：

#define max(a,b) ((a)>(b)? (a),(b))

以下讨论#define f（x）（x\*x） 和 #define f（x） x\*x 还有

#define f（x）（（x）\*（x））之间的差别

举个例子很容易看清：

int a,b,c,d,e;

a=1;b=2;c=4;d=6;

e=f(a+b)\*f(c+d)//我们想要的结果是（a+b）^2\*(c+d)^2，也就是900

对于1）#define f（x）（x\*x）

替换结果为：e=（a+b\*a+b)\*(c+d\*c+d) = 5\*34=170

2）#define f（x） x\*x

替换结果为：e=a+b\*a+b\*c+d\*c+d=1+2+12+24+4=43

3）#define f（x） ( (x)\*(x) )

替换结果为：e=( (a+b)\*(a+b)\*(c+d)\*(c+d) )=3\*3\*10\*10=900

最后这个才是我们想要的结果！

**12.函数指针的用法**（这东西好像不考，但是老师又让看了，那就随便举个例子）

找出整数a,b 中较大的一个。

#include<stdio.h>

int max(int a,int b)

{

return a>b ? a : b;

}

int main()

{

int a,b,(\*p)(int a,int b)=max;

printf("请输入两个整数：");

scanf("%d %d",&a,&b);

printf("较大者：%d\n",p(a,b));

return 0;

}

三．一些函数及其对应的头文件

1. stdio.h:最常见的
2. scanf,gets,getchar,以及对应的printf,puts,putchar(以下在字符操作中讨论）

scanf和getchar用法类似，一次只能读入一个字符；而gets可以读取一个字符串，但是要定义一个指针变量。

如：scanf：

{char ch1;

scanf("%c",&ch1);

printf("%c",ch1);}

getchar:

{char ch1;

ch1=getchar();

putchar(ch1);}

gets:

{char ch2[100];

gets(ch2);

puts(ch2);}

1. If,else,else if（略）
2. break,continue

break：是打破的意思，（破了整个循环）所以看见break 就退出整个一层循环。

continue： 是继续的意思，（继续循环运算），但是要结束本次循环，就是循环体内剩下的语句不再执行，跳到循环开始，然后判断循环条件，进行新一轮的循环。

1. switch,case,defult语句

a) 一定要注意 有break 和没有break 的差别，没有break时候，只要有一个case 匹配了，后续剩下的case都要执行，有break 则是直接跳出了switch 语句。

b) switch 只可以和break 一起用，不可以和continue 用。

c) switch(x) // x：是整型常量，字符型常量，枚举型数据。

{case 1: …. //表达式不可以是变量。

(break;)

case 2: ….

(break;)

......

defult: ......

}

1. for循环（略）
2. do,while语句

注：do

{

......

}

while(x)

表示只有当x=0时才会结束do的指令

(7~12与文件有关）

1. fopen

在使用文件之前,需打开文件.在C里使用fopen函数打开文件.格式为:

fopen(文件名,文件使用方式);

此函数返回一个指向FILE类型的指针.如:

FILE \*fp;

fp=fopen("file\_1","r");

如果调用成功,fp就指向file\_1,否则返回为NULL,所以为了保证文件的正确使用,要进行测试.采用如下语句:

If((fp=fopen("file\_1","r"))==NULL)

{

printf("Cannot open this file\n");

exit(0);

}

1. fclose

当文件的读写操作完成之后,使用fclose函数关闭文件.格式如下:  
fclose(文件指针)  
如:fclose(fp);

1. fread和fwrite

fread：从文件中读一个数据块；fwrite:向文件写一个数据块。格式如下：

fread(buffer, size, count, fp);

fwrite(buffer, size, count, fp);

1. buffer：是一个指针，对fwrite来说，是要获取数据的地址；  
   （2）size：要写入内容的单字节数；  
   （3）count:要进行写入size字节的数据项的个数；  
   （4）fp:目标文件指针；

例：fread(f,10,4,fp);//从fp中读10个4个字节的数据，存入数组f中

struct student

{ ......

}stu[40];

for(i=0;i<40;i++)

{

fwrite(&stu[i],sizeof(struct student),1,fp);//执行40次循环，每次向fp文件中写入结构体的一个元素

}

1. fgetc和fputc

1,调用putc(或fputc)函数输出一个字符  
调用形式为:  
fputc(ch,fp);  
功能是:将字符ch写到文件指针fp所指的文件中去.当输出成功,putc函数返回所输出的字符;否则,返回一个EOF值.EOF是在stdio.h库函数文件中定义的符号常量,其值等于-1.

2,调用getc(或fgetc)函数输入一个字符  
调用形式为:  
ch=fgetc(pf);  
功能是:从pf指定的文件中读如一个字符,并把它作为函数值返回.

1. fgets和fputs

fgets函数

fgets函数用来从文件中读入字符串.调用形式如下:

fgets(str,n,fp);

函数功能是:从fp所指文件中读入n-1个字符放入str为起始地址的空间内;如果在未读满n-1个字符时,则遇到换行符或一个EOF结束本次读操作,并已str作为函数值返回.

fputs函数

fput函数把字符串输出到文件中.函数调用形式如下:

fputs(str,fp);

注意:为了便于读入,在输出字符串时,应当人为的加诸如"\n"这样的字符串.

1. fscanf和fprintf

1.fscanf函数

fscanf只能从文本文件中按格式输入,和scanf函数相似,只不过输入的对象是磁盘上文本文件中的数据.调用形式为:

fscanf(文件指针,格式控制字符串,输入项表)

例如:fscanf(fp,"%d%d",&a,&b);

等价于scanf("%d%d",&a,&b);

2.fprintf函数

fprintf函数按格式将内存中的数据转换成对应的字符,并以ASCII代码形式输出到文本文件中.fprintf函数和printf函数相似,只是将输出的内容按格式存放到磁盘的文本文件中.调用形式如下:

fprintf(文件指针,格式控制字符串,输出项表)

如:fprintf(fp,"%d %d",x,y);

1. string.h(字符串操作)
2. strlen函数

计算字符串的长度（第一个‘\0’前面的字符的个数，不包括\0）

#include <stdio.h>

#include <string.h>

int main() {

char str[10];

gets(str);

int len = strlen(str);

printf("%d\n", len);

return 0;

}

1. strcmp函数

strcmp 函数返回两个字符串大小的比较结果，比较原则是按****字典序****

一般形式：strcmp（字符串1，字符串2）

如果字符串1和字符串2相同，返回值为0；

如果字符串1大于字符串2，范围值是一个正整数；

如果字符串1小于字符串2，范围值是一个负整数

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**char str1[50], str2[50];**

**gets(str1);**

**gets(str2);**

**int cmp = strcmp(str1, str2);**

**if(cmp < 0) printf("str1 < str2\n");**

**else if(cmp > 0) printf("str1 > str2\n");**

**else printf("str1 == str2\n");**

**return 0;**

**}**

1. strcpy函数

strcpy 函数可以把一个字符串复制给另一个字符串

strcpy(字符数组 1， 字符数组 2)

把字符数组 2 复制给字符数组 1，这里的“复制”包括了结束符 \0

实际上是字符数组2取代字符数组1的前n个元素（n=strlen(字符数组2)+1）

#include <stdio.h>

#include <string.h>

int main() {

char str1[50], str2[50];

gets(str1);

gets(str2);

strcpy(str1, str2);

puts(str1);

return 0;

}

注：可以用strncpy函数将字符串2的前n个字符复制到字符串1中，例如：strncpy（str1,str2,2），即把字符串2的前两个字符复制到字符串1中

1. strcat函数

strcat()可以把一个字符串接到另一个字符串后面

strcat(字符数组1， 字符数组2)

把字符数组 2 接到字符数组 1 后面

#include <stdio.h>

#include <string.h>

int main() {

char str1[50], str2[50];

gets(str1);

gets(str2);

strcat(str1, str2);

puts(str1);

return 0;

}

1. Stdlib.h(开辟空间)

1)函数名称: calloc

函数原型: void \* calloc(unsigned n,unsign size);

函数功能: 分配n个数据项的内存连续空间,每个数据项的大小为size

函数返回: 分配内存单元的起始地址,如果不成功,返回0

例：int \*p;

p=calloc(50,4);//开辟50x4字节的临时分配域，把首地址赋给指针变量p

2)函数名称: free

函数原型: void free(void\* p);

函数功能: 释放p所指的内存区

参数说明: p-被释放的指针

例：free（p）;//释放p所指向的动态分配空间

3)函数名称: malloc

函数原型: void \* malloc(unsigned size);

函数功能: 分配size字节的存储区

函数返回: 所分配的内存区地址,如果内存不够,返回0

例：malloc(100)//开辟100字节的临时分配域

4)函数名称: realloc

函数原型: void \* realloc(void \* p,unsigned size);

函数功能: 将p所指出的已分配内存区的大小改为size,size可以比原来分配的空间大或小

函数返回: 返回指向该内存区的指针.NULL-分配失败

例：realloc(p,50)；//将p所指向的已分配的动态存储空间改为50字节

5)exit(0)

1. math.h
2. pow:求次方

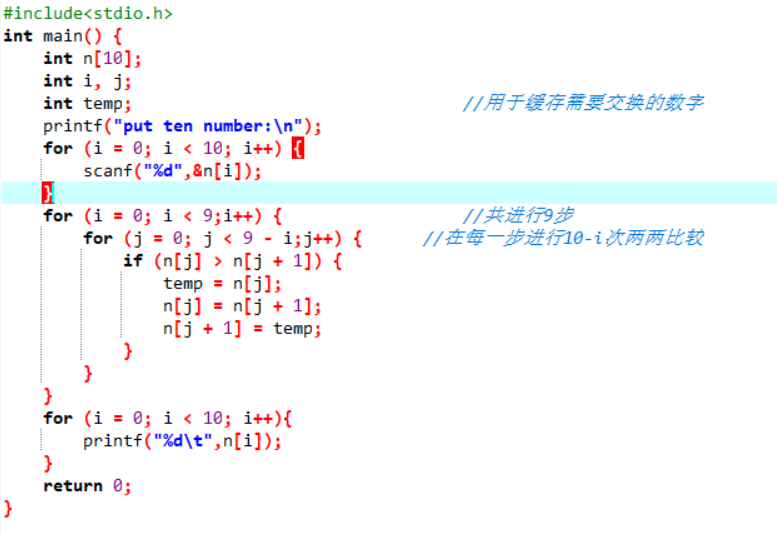
pow(x,y):即x的y次方

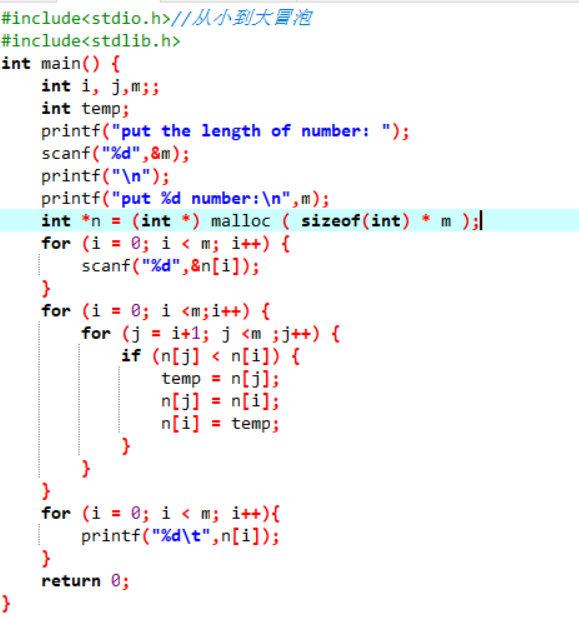
1. exp,sin,cos....

四．一些必考内容的代码举例

（一）. 数据排序

1. 冒泡法：两种方法





1. 选择法

第一次从要排序的数据元素中找到最小或最大的放到最前面，然后再剩下的元素中找到最小的或最大的放到后面，以此类推（以下例子是从小到大排序，如果需要从大到小，把min换成max，把标记位置的大于换成小于即可）

#include<stdio.h>

#include<stdlib.h>

int main() {

int i, j,m,min;

int temp;

printf("put the length of number: ");

scanf("%d",&m);

printf("\n");

printf("put %d number:\n",m);

int \*a = (int \*) malloc ( sizeof(int) \* m );

for (i = 0; i < m; i++) {

scanf("%d",&a[i]);

}//输入数组

for(i=0;i<m-1;i++)//进行m-1次循环，因为最后剩的那个不需要比较

{

for(min=i,j=i+1;j<m;j++)//min=i，j=i+1，即先从第i个和第i+1个比较，直到比较完i后所有的元素

{

if(a[min]>a[j])//如果第min个元素小于第j个，那就继续往下比较

min=j;//直到出现某个元素比min小时，让min表示这个元素

}

if(min!=i)//当i后所有元素比较完，发现还有比i小的元素时（这个元素已经用min暂存）

{

temp=a[i];

a[i]=a[min];

a[min]=temp;//让这个比i小的元素与i互换，这样a[i]就可以按顺序排列

}

}

for (i = 0; i < m; i++){

printf("%d\t",a[i]);

}

return 0;

}

1. 插入法

开始时将第一个元素看成一个有序表，将其他元素依次与表中的数据比较，找到合适位置插入。

#include<stdio.h>

#include<stdlib.h>

int main() {

int i, j,m,key;

int temp;

printf("put the length of number: ");

scanf("%d",&m);

printf("\n");

printf("put %d number:\n",m);

int \*a = (int \*) malloc ( sizeof(int) \* m );

for (i = 0; i < m; i++) {

scanf("%d",&a[i]);

}//输入数组

for(i=1;i<m;i++)//从第二个开始，和前面的数比较

{

for(key=a[i],j=i-1;j>=0&&a[j]>key;j--)//如果前面一个数大于后面那个数则进行循环，这里看起来要进行很多次循环，实际上只成功进行一次之后就不会再进行了，因为前面已经是有序表了，但j依然会--

{

a[j+1]=a[j];//前面那个数赋值给后面那个数

}

a[j+1]=key;//因为在循环里面j最后还会减1，那么这里的j+1和标记处的j+1其实是不相等的，比标记处的少了1，所以对应的不是第i个元素，而是i前面那个元素，这里把key,也就是a[i]赋值给前面那个元素，实现了两个元素的互换（如果没进行上面的循环，那么j+1代表的还是i，也就是没有变化）

}

for (i = 0; i < m; i++){

printf("%d\t",a[i]);

}

return 0;

}

（二）.二分查找

搜索过程中从数组的中间元素开始，如果中间元素正好是要查找的元素，结束。如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且和开始一样从中间元素开始比较。如果在某一步数组为空，则代表找不到，这种搜索算法每一次比较都使搜索范围减小一半。

#include<stdio.h>

#include<stdlib.h>

int main() {

int i, j,m,mid,x,key;

int temp;

printf("put the length of number: ");

scanf("%d",&m);

printf("\n");

printf("put %d number:\n",m);

int \*a = (int \*) malloc ( sizeof(int) \* m );

for (i = 0; i < m; i++) {

scanf("%d",&a[i]);

}//输入数组

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

{

for(key=a[i],j=i-1;j>=0&&a[j]>key;j--)

{

a[j+1]=a[j];

}

a[j+1]=key;

}

printf("now,%d number is:\n",m);

for (i = 0; i < m; i++){

printf("%d\t",a[i]);

}//先用插入法从小到大排序，二分查找是在有序数列里查找

printf("\n");

printf("put the number to find:");

scanf("%d",&x);//输入要找的元素

printf("\n");

i=0;

j=m-1;

while(i<=j)

{

mid=(i+j)/2;

if(x==a[mid]) //中间那个元素刚好是要找的元素

break;

else if(x>a[mid])//如果中间的元素比要找的元素小

i=mid+1;//把范围缩小到后一半

else //如果中间的元素比要找的元素大

j=mid-1;//把范围缩小到前一半

}

if(i>j)

printf("not found!\n");

else

printf("the location of %d is %d.\n",x,mid+1);//加一别忘了，因为第一个元素是a[0]

return 0;

}

（三）函数

1. 函数嵌套：找出四个整数的最大值

#include<stdio.h>

int max2(int a,int b)

{

if(a>=b)

return a;

else

return b;

}//也可写为return (a > b ? a : b);

int max4(int a, int b,int c,int d)

{

int max2(int a, int b);

int m;

m=max2(a,b);

m=max2(m,c);

m=max2(m,d);

return m;

}//也可写为 return max2(max2(max2(a,b),c),d);

int main()

{

int a, b, c,d,max;

printf("please enter four integer numbers:");

scanf("%d,%d,%d,%d", &a, &b,&c,&d);

max = max4(a, b,c,d);

printf("max is %d\n", max);

return 0;

}//没啥好解释的（手动狗头<\*\_\_<\*）

1. 函数递归

斐波那切数列第n项的值：

#include<stdio.h>

int fb(int n); //斐波那契求和函数原型申明

int main()

{

int n,s;

scanf("%d",&n);

s=fb(n);//第n项的值

printf("%d\n",s);

}

int fb(int n)

{

int r;

if( n==1 || n==2)

r= 1;

else

r= fb(n-1)+fb(n-2);//递归调用

return r;

}

或者更难的汉诺塔：//这玩意我看不太懂

#include <stdio.h>

#include <string.h>

/\*

算法思路：1将 n-1个盘子先放到B座位上

2.将A座上地剩下的一个盘移动到C盘上

3、将n-1个盘从B座移动到C座上

\*/

void move(char x, char y);

void hannuo(int n,char one ,char two,char three)

{

if(n==1)move(one, three); //递归截止条件

else

{

hannuo(n-1,one ,three,two);//将 n-1个盘子先放到B座位上

move(one,three);//将A座上地剩下的一个盘移动到C盘上

hannuo(n-1,two,one,three);//将n-1个盘从B座移动到C座上

}

}

void move(char x,char y)

{

printf("%c--->%c",x,y);

}

int main()

{

int n;

printf("input your number");

scanf("%d",&n);

hannuo(n,'A','B','C');

return 0;

}

（四）字符串操作

//具体函数上面（string.h包含的函数）已经讨论

（五）指针

国家名索引：（设 char countries[10][40], 并存储十个国家的名称。保持counties 内容不变，针对该数组制作两张索引表，索引表用指针数组实现，每个指针指向一个字符串（国家名称），要求将一张索引表按照串长（从小到大）排序，另一张索引表根据首字母（ASCII 码顺序）排序。利用两张索引表输出两种排好序的国家名称。）

#include<stdio.h>

#include<string.h>

int main()

{

void slength(char \*p[],int n);//按长度排序函数

void sfirst(char \*q[],int n);//按首字母排序函数

char country[10][40];

char\*p[10];

char\*q[10];//分别用于两个索引

int i;

for(i=0;i<10;i++)

{

gets(country[i]);//输入

p[i] = country[i];

q[i] = country[i];

}

slength(p,10);

sfirst(q,10);

printf("长度索引表\n");

for(i=0;i<10;i++)

printf("%s\n",p[i]);

printf("首字母索引表\n");

for(i=0;i<10;i++)

printf("%s\n",q[i]);

}

void slength(char \*p[],int n)

{

int i,j;

char \*temp;

for(i=0;i<n;i++)//冒泡排序法

{

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

{

if(strlen(p[i])>=strlen(p[j]))//长度比较

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

}

void sfirst(char \*q[],int n)

{

int i,j;

char\*temp;

for(i=0;i<n;i++)//同样是冒泡排序

{

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

{

if(\*q[i]>=\*q[j])//首字母大小比较

{

temp=q[i];

q[i]=q[j];

q[j]=temp;

}

}

}

}

（六）文件

1.文件的打开与读写（凯撒密码为例）

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<windows.h>

int main()

{

FILE \*fp;

char file[1000]="\0";

int i=0;

if((fp=fopen("secret.txt","r"))==NULL)

{

printf("error!\n");

exit(0);

}

else

{

fp=fopen("secret.txt","r");//只读方式打开文件

while(!feof(fp))

{

fgets(file,1000,fp);//从文件中拿出999个字符放入数组file中

}

rewind(fp);//fp的指针返回最开始

fclose(fp);//关掉文件

while(file[i]!='\0')//以下是对凯撒密码的加密

{

if(file[i]<=90&&file[i]>=65)

{

if(file[i]+7<=90)

file[i]=file[i]+7;

else

file[i]=file[i]-19;

}

else if(file[i]<=122&&file[i]>=97)

{

if(file[i]+7<=122)

file[i]=file[i]+7;

else

file[i]=file[i]-19;

}

i++;

}

}

i=0;

fp=fopen("secret.txt","w");//只写方式打开文件

while(file[i]!='\0')

{

fputc(file[i],fp);//把数组file中的已经加密的密码放回文件中并覆盖原码

i++;

}

fclose(fp);//关掉文件fp

return(0);

}

（七）链表

按顺序建立，插入，删除链表

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define LEN sizeof(struct student)//一个宏

struct student{

int stunum; //学号

char name[20]; //姓名

float examscore; //考试成绩

float labscore; //实验成绩

float totalmark; //总评成绩

struct student \* next; //下一个结点

};

struct student \*create(int n)

{

struct student \*head=NULL,\*p;

int i;

for(i=0;i<n;i++)//循环运行插入所有的数据

{

p=(struct student\*)malloc(LEN);//给p一个内存

scanf("%d%s%f%f",&p->stunum,p->name,&p->examscore,&p->labscore);//输入p

p->totalmark=0.6\*(p->examscore)+0.4\*(p->labscore);//计算p的总评成绩

if(head==NULL)//如果头是空的

{

p->next=NULL;

head=p;//把p的数据放到头里面

}

else//如果头里面已经有数据了

{

struct student \*q1,\*q2;//新定义q1,q2

q1=q2=head;//先让q1,q2=head

while(q1&&p->totalmark<q1->totalmark)//让q1遍历链表寻找p的总评成绩大于q1的总评成绩（即head的总评成绩） 跳出while循环

{//如果满足循环条件

q2=q1;//让q2预存q1的数据

q1=q1->next;//q1等于q1下一个结点的数据

}//即让q2代表q1后q1继续往后寻找，直到找到 p的总评成绩大于q1的总评成绩的位置

p->next=q1;//找到后让p的下一个结点指向现在的q1，即刚好比p小的那个数据

//接下来就是让p与前面的结点链接起来了

if(q1==head)//如果新来的p比链表里任何的数都要大，那么q1不需要后移寻找了，即q1此时就是head

head=p;//那么新来的p作为最大的直接担任头的地位

else//如果q1往后寻找了

q2->next=p;//那就让q2的下一个结点成为p，这样就形成一个链表片段：q2,p,q1，是按顺序排列的

}

}

return(head);//返回head为头的链表

}

struct student \* del(struct student \* head, int stunum)//删除结点

{

struct student \*q1,\*q2,\*m,\*p;

int i;

m=head;//先储存head的值，因为之后head有变动但又要回到原来的head

while(head)//循环寻找想删除的结点在哪

{

if(head->stunum==stunum)//如果是要删除的结点

{

i=1;//让变量 i=1

break;//结束循环

}

else i=0;//如果不是想删除的结点，那么让 i=0

head=head->next;//让head被检查之后继续往下检查

}

head=m;//因为while中head发生了变动，这里让head回到和原来一样

if(i==0)//如果i=0，说明一直没找到

{

printf("未找到\n");//不存在时的情况

}

if(i==1)//如果i=1，说明找到了

{

q1=q2=head;//让两个指针同时和head相等

while(q1&&q1->stunum!=stunum)//当q1的学号和输入学号不相等时

{

q2=q1;//让q2预存q1的数据

q1=q1->next;//q1等于q1下一个结点的数据

}

if(head->stunum==stunum)//如果第一个结点就是要删除的结点

{

p=head;//让p成为head

head=head->next;//新的head成为之前head连接的下一个结点

free(p);//把p释放了（即把head删除了）

}

else//如果第一个结点不是要删除的结点

{

q2->next=q1->next;//让q2链接q1的下一个结点

free(q1);//把q1释放了

}

}

return(head);//返回head为头的链表

}

struct student \*insert(struct student \*head,struct student \*p)//插入函数

{

struct student \*q1,\*q2;

q1=q2=head;//先让q1,q2=head

while(q1&&p->totalmark<q1->totalmark)//让q1遍历链表寻找p的总评成绩大于q1的总评成绩（即head的总评成绩） 跳出while循环

{

q2=q1;//让q2预存q1的数据

q1=q1->next;//q1等于q1下一个结点的数据

}

p->next=q1;//找到后让p的下一个结点指向现在的q1，即刚好比p小的那个数据

//接下来就是让p与前面的结点链接起来了

if(q1==head)//如果新来的p比链表里任何的数都要大，那么q1不需要后移寻找了，即q1此时就是head

head=p;//那么新来的p作为最大的直接担任头的地位

else//如果q1往后寻找了

q2->next=p;//那就让q2的下一个结点成为p，这样就形成一个链表片段：q2,p,q1，是按顺序排列的

return(head);//返回head为头的链表

}

void print(struct student \*head)

{

while(head)//使head为头的链表全部输出

{

printf("%d %s %.2f %.2f %.2f \n",head->stunum,head->name,head->examscore,head->labscore,head->totalmark);//输出head

head=head->next;//head变为下一个结点的数据

}

printf("\n");

}

int main()

{

struct student \*order,\*into,d={16589,"猪才怪", 100,100,100};//order是顺序链表；into是插入的结点，d是插入的数据

into=&d;

int stunum;

order=create(10);//顺序创建这十个数据

printf("\n");

printf("建立的顺序链表为：\n");

print(order);//用函数输出这十个数据

printf("输入删除学号:");

scanf("%d",&stunum);

order=del(order,stunum);//删除

printf("\n删除后数据为：\n");

print(order);

order=insert(order,into);//插入

printf("\n插入后数据为：\n");

print(order);

return(0);

}