**第一部分 易错易误点**

1. **位运算注意事项**

1）~按位取反用于求反码

2）进行右移运算的注意事项：

无符号数左侧一律补0——逻辑右移

有符号数取决于符号位——算术右移

左移乘以2，右移除以2，注意右移一位不一定和除以二得到的结果相同，要考虑两种右移方式

3）位运算，求余运算的操作数均为整型。求余运算中余数的正负取决于被除数，即符号的左操作数

1. **位运算表达式示例**
2. 写一个表达式判断int型变量x和y的第i位（最低位是0位）是否相同，相同为1，否则为0 x>>i&1|y>>i&1
3. 写一个表达式取整数x的最右边m位 ~(~0<<m)&x
4. 将int变量x向左循环移动n位，注意机器字长的不统一 x<<n|(unsigned)x>>(sizeof(int)\*8-n)
5. 对于unsigned short 类型的x和y，将y的低字节换为x的高字节

x>>8&0xff |y&0xff00

1. 写一个值是short变量x的原码的表达式 x>=0?x: -x|0x8000
2. 判断int变量a和b的符号是否相同，相同为1（均为0时符号视为相同），否则0
3. 请写一个C表达式，将unsigned short类型变量x的高字节送入unsigned char类型变量ch中。 【答案】ch=(x&0xff00)>>8

8）写一个截取short int型变量x的第n位（从右至左编号为0,1,2,…,15，0≤n≤15），

并将该位信息装配到short int型变量new的第15-k位(0≤k≤15)的表达式。

【答案】new = new &~(1<<(15-k))|((x >> n) & 1) << (15 – k)

9）请写一个C表达式，判断短整型数a的最高位是否为1，是，表达式的值为1，否则为零。 【答案】x&0x8000?1:0 或 (x&0x8000) == 0x8000 或 a>>15 & 1

10) 写一个表达式，取整数x最右边的m位。如0x123456fa，取最右边5位等于0x0000001a; 【答案】~(0xffffffff<<m)&x

11) 写一个C表达式，将取出int型变量a的最高字节作为表达式的值

【答案】a>>(sizeof(int)-1)\*8 && 0xff

12) 写一个C表达式，判断两个int变量a与b之和值c（c为int变量,c=a+b;）是否溢出：如果是溢出，则表达式的值为非0，否则为0。

【答案】a>0 && b>0 && (c<a || c<b) || a<0 && b<0 && (c>a || c>b)

13）设3个unsigned 变量a、b和c，分别代表3个线段的长度，写一个C表达式，判定这个3个线段是否能够成一个三角形：如果可以构成，则表达式值为非0，否则为0。

【答案】a+b>c && a+c>b && b+c>a && abs(a-b)<c && abs(a-c)<b && abs(b-c)<a

1. **位运算计算实例**
2. char a=4, b=6, c; short x=0x80ff, y=10;

计算c=x>>8（注意如果是负数而且是signed类型，要先求其原码得到真实的数，而要以%d输出时char类型提升为int类型符号位扩展是一样的做法）

y & x<<4 ~ b | x

2) unsigned i =0, j=1; int a;

计算a=~ i > j a=~ i ^j -i >>15 & & -j <<15

3) char c; short x=0x80ff; 计算 c=x>>8; -128

4) char c1=16, c2=0x61; short h1=0x10 ,h2=0xFF01; unsigned long lu=0x12345678;

计算 c1= lu c2 & h2 >>4 120 96

5）short x[ ]={ -1,0,1}, \*p=x; unsigned short y=8;

计算y =~x[1] y >> x[2] ~(\*p<<3) 65535 4 7

6) unsigned i = 0, j = 1; int a;

计算a = ~i ^ j 【答案】-2

7) char c1='\x41',c2; short int m=3,n=6,i=0x35F7,j=0xF315； float x=12.0,y=20.0;

【-13510或0XCB3A】m=~(i&j>>2)

8）unsigned short i=65535,j=2; short int m=4, n=1;

计算m=I 【答案】-1

1. **标识符与关键字**

1）主函数名和标准库函数名均不是关键字，而是标识符

2）typedef sizeof const enum goto volatile struct union以及四种变量声明对应单词均为关键字,include不是关键字

1. **各种常量注意事项**

1）双引号可以作为图形符号也可用转义序列表示，故‘”’和‘\”’均合法

而反斜杠\和单引号仅可用转义序列表示，即‘\’’和‘\\’

字符串中的双引号字符必须用转义序列表示。

2）“”是合法的字符串常量，‘’则不是合法的字符常量。

3）整型常量后缀有u l以及其大写和组合，浮点型常量总是有符号的，只有l和f以及大写字母的后缀，故浮点型常量不可用unsigned和signed修饰。

4）0开头的是八进制整型常量，后面不可跟有非八进制数字，0X或0x开头的是十六进制整型常量，后面不可跟有非十六进制数字或字母，但跟有u和l是要注意，可能是合法的。

5）在转义序列中，‘\hhh’为八进制转义序列，‘\xhh’（必须为小写x）为十六进制转义序列，不可出现不合法的进制字符，当在字符串常量中读取时，八进制表示方法读取到非八进制字符时结束，十六进制表示方法若在转义序列内有非十六进制字符时则会直接报错。

6）不常见的转义序列有\r \f \a \b

7）浮点型常量中有指数时（即科学计数法），小数点可缺省，指数部分由字母e或E、一个可选的正负号（默认为+）和一个整数n组成，如45e-3 **.**15e5

当没有指数时，带小数点且不可省略，可以以小数点开头无整数部分，也可以小数点结尾无小数部分。

注意float类型的存储范围float a=1.45e350是不合法的

无后缀就是double，后缀f是float，后缀lf是long double

1. short和char类型整数提升的注意点：

任何表达式中的char、unsigned char、short、unsigned short均需先转换为int或unsigned，若原始类型的所有值均可用int表示的转换为int，否则转换为unsigned

例如unsigned short x=-1;如果short类型范围小于int则转换为int，则（-x）的值为-1（先转换为int再进行单目符号运算）；若类型范围相等（short int与int的长度相等的可能，这种情形下，unsigned short就无法提升为int表示，只能提升为unsigned int），则（-x）类型为unsigned，结果为一个大正数。例如在32位字长机器中，short x=0x80ff；在运算时，short提升为int，符号位扩展（整数提升时扩展位为符号位），转换为0xffff80ff

用整数提升的知识解释表达式-1<1U的值为0

1. 枚举名和左花括号间没有等号 enum color {red, blue, green} c;
2. 枚举类型进行赋值时要在值域之内；例如设有声明: enum {A, B=0, C, D, E=1} a; 则不能赋给a的值是 【D】

A、E+1 B、2 C、A D、D+1（超过了值域）

1. char在表示整数时范围为-128到127，unsigned char的表示范围则是0到255.
2. char的长度≤ short int的长度≤ int的长度
3. 自动/隐式类型转换：short/char→int→unsigned →long→unsigned long→float→ double→long double
4. 逻辑与与逻辑或中当左式已经可以确定该逻辑表达式的值时将不再运算右式
5. ~~x/y/z等同于x/y\*z?~~
6. 逗号表达式中依次对表达式进行运算，最后一个表达式的值是整个逗号表达式的值。
7. char类型均为1字节，int与unsigned均为2或4字节，short为2字节，long为4字节，float为4字节，double为8字节
8. x=1/2\*x和x/=2并不相同，前者结果为0，而x=1.0/2\*x则和后式相同
9. 在没有特殊说明时常量50就是整型数据，要赋给float就要强制转换或者写为50.0
10. **输入输出注意事项**

1）%#o,%#x可输出前缀

%.nd可对浮点数规定小数点后位数，并四舍五入输出，对于整数则规定至少输出的位数，不足时则补上前导0，对于字符串则规定至多输出的字符个数

%-d右对齐

%0md可补前导0，%md中当实际长度大于m时按实际输出，否则左补空格

%2c表示输入c的值时必须是两位，但是仅把第一个字符赋给c

1. scanf可规定输入数据所占有的列数，但是不可规定精度
2. 输出float类型用%f，输出double类型可用%f可用%lf，输出long double必须用%lf

3）%\*表示本输入项输入后不传送给任何量，而在输出中可作为可变域宽使用，在打印时，要有max作为\*的值。

4）整型int或unsigned的长度与机器字长相同，可用sizeof（int）获取机器字长

5）(c=getchar( ))!=EOF注意，赋值运算符优先级要低于!=运算符

1. **函数与编译预处理**

1）sqrt函数返回值和power函数返回值均为double类型，要注意是否使用了库函数却忘记引用头文件

2）b的平方必须用b\*b表示，或者用power函数实现，b^2是按位异或运算，不是平方运算

3）宏名展开时要注意有无括号，只是简单地替换，要注意运算符优先级，而不可默认加上括号。注意宏定义时最后有分号是隐式错误，可能会导致错误，但是不一定。

1. ~~可变参数宏与防止重复包含头文件（P98）~~

5）函数无类型名时，默认为int类型，此时如果返回值实际不是int类型，会自动转换为int类型，这属于隐式错误，会导致误差。

6）注意函数声明是否有，是否正确，声明函数原型可出现在任一程序块，甚至是main函数中

7）strlen函数不包含‘\0’，strcat函数第一个参数只可是字符数组名，第二个参数可为字符数组可为字符串常量，strcpy函数也复制了‘\0’，注意无语句循环结构的分号不可省略。

8）注意typedef、const和define的区别

#define定义的是宏，不是C语句，在编译前处理，且不需要为M分配内存单元。

const说明的在编译时处理，且需要根据类型为M分配内存单元。

9）C语言函数只可嵌套定义，不可递归定义

10)已知int fun(int) ; 则(\*fun)(6)的调用方式也正确

11）定义带参数的宏swap(a,b)，使两个参数的值交换。

答：#define swap（a，b） { int t；t=a；a=b；b=t；}

12）以下定义的带参的宏实现计算大于或等于整数x的最小偶数。

#define EVEN(x) (( (x)+2)/2\*2) 【答案】(( (x)+1)/2\*2)

1. **switch与if结构说明:**
2. switch语句中表达式的值不是整数则自动取整
3. 一旦一个case语句匹配成功，将不再匹配其他表达式，若有break则退出，否则继续执行后面的语句，包括case语句块
4. 当default语句置于中间时，则将执行default语句，若有break退出，否则执行之后的所有语句
5. if()

语句1；语句2；

else()

以上声明是错误的，没有配对的if语句块

1. else与最近的未配对的if配对
2. **下面程序段可判断整数x是否为素数，当x是素数时，标志变量flag为1，否则，flag为0。请改写下面程序段中的for语句，要求功能等效，但不能使用break，goto等非结构化语句。**

for (i=2; i<x/2; i++)

if (!(x%i)) break;

if (i >= x/2) flag = 1;

else flag = 0;

for (i=2,flag=1; i<x/2&&flag; i++) flag = (x % i != 0);

1. **用完全等效的新代码段替代以下代码，要求新代码段中不得使用转移语句。**

for (i=2; i\*i<=n; i++) if (!(m = n % i)) break;

for (i=2; i\*i<=n && (m=n%i); i++)

1. **二维数组初始化注意事项：**
2. 一维可缺省，二维不可
3. int a[‘0’]是正确的，会进行整数提升
4. 变长数组声明时不可初始化
5. 仅允许右端值可不初始化
6. 可顺次初始化，数组a[2][3]中的a[0][4]代表第四个数，是合法的取值方式，注意二维数组的存储结构
7. 数组名是常量地址，不能给数组名赋值，也不能进行自增操作，但可以进行加减操作
8. 修改代码——char s1[20]; s1="china";

【答案】char s1[20] ="china" 或者strcpy(s1, "china")

1. 给首列元素赋值和给每行元素赋值的区别
2. **四种变量声明的区别**
3. auto是局部变量的默认存储类型，可省略，只可作用于变量。
4. extern类型是在函数外部定义的全局变量，定义时不可使用extern当引用时才可使用，使用extern可以引用在外部文件中的变量，有永久生命周期，仅第一次赋值时有效。在外部定义无初始化时相当于默认初始化为0。既可作用于变量，也可作用于函数。当局部变量与全局变量重名时，局部变量将屏蔽全局变量。当作用于函数时，该函数可作用于已经声明引用该函数的其他文件中。
5. static是静态变量，有永久生命周期，只是与extern作用域不同，分为静态局部变量和静态外部变量，不可被其他文件引用，仅第一次赋值时有效。在外部定义无初始化时相当于默认初始化为0，如果已经定义为static，即使再加extern声明也不可在其他文件中使用。函数定义时，返回值类型前的static表明函数返回值的生命周期不一定为程序运行的整个期间，这需要根据两种类型进行判定。函数形参被当作局部变量使用，且存储类型不可能是static。既可作用于变量，也可作用于函数。静态外部变量会屏蔽同名的外部变量。当作用于函数时，静态函数只作用于所在文件。
6. register变量只可定义局部变量，且直接分配至寄存器中，适用于访问频次高的变量，不一定能够成功使用，当CPU寄存器使用情况紧缺时无效，相当于auto类型变量，直接在栈上分配存储。该类型变量成功使用时无内存地址，不可用&求寄存器变量地址。
7. 函数的形参的类型不可能为static和extern，而可以声明为auto（默认）或register
8. 应用例如：文件file1.c和file2.c共享变量x。file2.c和file3.c共享变量y并且不允许file1.c共享该变量。file1.c的各个函数共享变量y并且不允许file2.c和file3.c共享该变量。请在file1.c、file2.c、file3.c文件中写出相关的声明语句。

file1.c file2.c file3.c

int x ; extern int x; int y;

static int y ; extern int y ;

... ... ...

1. **指针注意事项**
2. 定义指针数组时也会有指针悬挂的情况，要注意初始化
3. 用指针变量读取字节应用
4. 字段结构中字段没有地址，不可用&进行取地址操作
5. 指针的地址不能直接赋给一个一级指针，要定义一个二级指针
6. 无值型指针（void）

任何类型的指针都可以将其值赋给无值型指针，但是对其进行\*（间访）是不合法的

6）说明二维字符数组与字符指针数组处理多个字符串时的区别。

答：二维字符数组一旦定义，那么每个字符串的最大长度、首地址都不能改变了。

字符指针数组是存放字符指针的数组。由于它仅用来存放指针，

所以它指向的每个字符串的首地址可以改变，字符串最大长度也可以改变。

1. 结构体中指针p->数据域的类型是一个值，当对其进行赋值操作时需要用&符号
2. 字符指针数组的输出，对于char\*argv[5];输出方式为printf(“%s”,argv[i]);
3. **复杂的指针之指向数组的指针**
4. 初识：int x[1][2],(\*p)[2]=x;定义了一个指向一个有两个元素的一维数组的指针，而该指针所指向的是x，也就是x[0][0]的地址，而p自增后地址增量是2\*sizeof（int），指向了第二行，也即第二个一维数组。如输出2维数组元素m[1][2]的代码片段如下：

int m[3][4], (\*p)[4]=m; p=p+1; printf("%d", \*p[2]);

1. int v[2][3][4],(\*p2)[3][4]=v;定义了一个指向二维数组的指针，而该指针指向的是v[0][0][0],自增后地址增量是3\*4\*sizeof(int)，指向了第二个二维数组。
2. 依此类推，n维数组可以逐级分解为以n-1维数组为元素的一维数组，而n维数组的数组名字则是一个指向n-1维数组的常量指针，其值为首元素的地址，而n-1维数组亦可仿此类推
3. 例如对于int a[2][3]而言，数组名a是指向有三个整形元素的一维数组的指针，类型为int（\*）[3]; a[0]的类型则为int \*，a[1]指向第二行首元素的指针，类型也为int \*
4. **复杂的指针之多维数组的指针与引用**
5. 对于二维数组u,访问其u[i][j]的方式有\*(u[i]+j) \*(\*(u+i)+j)
6. 对于三维数组u,访问其u[i][j][k]的方式有\*(u[i][j]+k) \*(\*(u[i]+j)+k) \*(\*(\*(u+i)+j)+k)

(\*(u+i)[j][k])

**接下来，将用此方法表示多维数组：**

1. int A[I][J][K][L][M][N],B[I][J],C[I][J][K];

int (\*p5)[J][K][L][M][N]=A; int (\*p1)[J]=B;int(\*p2)[J][K]=C;

注意，p5和A均为指向五维数组的指针，因此可以赋值。

1. 用typedef定义类型表达式（对于一个声明语句，去掉对象标识符即为一个类型表达式）
2. 例如typedef unsigned short unit, SIZE；则用unit和 SIZE说明的对象都将是无符号短整型的。typedef char\*(\*p)(char\*,char\*);则用p说明的对象都将是这样形式的函数指针。再如typedef int\*(\*Array[10])(int\*);
3. **复杂的指针之指针的说明与解释**
4. 根据优先级与结合性进行拆分一步步地说明复杂指针。
5. **多级指针与指针数组**

#include<stdio.h>

void main()

{

char \*str[]={"happywuhan","beijingok","chinayes","hongkong"};

char \*\*p[]={str+3,str+2,str+1,str};

char \*\*\*pp=p; //这时pp指向str+3

++pp; //这时pp指向str+2

printf("%s",\*\*pp); //\*\*后即指向字符串chianyes的首字符，输出得到chinayes

printf("%s",\*--\*pp+3); //\*pp是str+2自减后是str+1，\*操作后指向beijinggok的首字符，+3后指针移动到字符j,这时只输出jinggok,注意此时是str+2变成了str+1

++pp; //pp再次自增指向str+1

printf("%s",\*(\*(pp-1)-1)+1); //pp-1指向str+1（这个str+1是原先的str+2），\*后\*（pp-1）的内容是str，-1后得到str，\*后再加一指向happywuhan的第二个字符a，这时自此输出得到appywuhan

}

chinayesjingokappywuhan

1. **二分查找函数（需要元素有序）**

int BinarySearch(int a[],int x,int n)

{

int front=0,back=n-1,middle;

while(front<=back) {

middle=(front+back)/2;

if(x<a[middle])

back=middle-1;

else if(x>a[middle])

front=middle+1;

else

return middle;

}

return -1;

}

1. **数字串与数值之间的转换的函数**
   1. 将十进制数字串转换为数字

int atoi(char \*s)

{

int j=0,num=0;

for(;\*(s+j)!='\0';j++)

num=num\*10+\*（s+j）-‘0’;

return num;

}

1. **有关文件的知识**
   1. r表示只读（指定文件一定要存在，否则会出错）
   2. a表示添加（即追加，在尾部进行写）
   3. w表示只写（已经存在的文件要清除内容）
   4. +表示更新 b表示二进制文件
   5. 标识路径的斜线必须使用双反斜线“d:\\one\\file.c”
   6. fopen函数如果成功，则在内存中开辟一个存储空间存放FILE类型数据并返回起始地址，否则返回NULL
   7. exit()函数关闭所有文件终止程序，fclose函数先释放文件缓冲区成功关闭返回0，否则返回EOF（定义在stdio.h中）
   8. 读写文件时访问的是文件缓冲区而非直接访问文件，如果没有清除和保存缓冲区内容就终止程序则其中数据会丢失，且同时打开的文件数目有限
   9. int fgetc(FILE \*p) 返回字符或EOF
   10. int fputc(int c,FILE\*p)返回字符或EOF
   11. char \*fgets(char\*str,int n,FILE\*p)当读到回车换行符、文件末尾或者读满n-1个字符时函数返回该字符串的首地址，同时文件指针偏移至当前位置，若出错或已到结尾则返回NULL（空指针）
   12. fputs形如fgets，若写入错误返回EOF
   13. fread函数参数为待读入数据块的起始地址(void\*)，每个数据块的大小（字节数，size-t），最多允许读取的数据块个数（size-t）以及文件指针（FILE\*），返回实际读到的数据块个数
   14. fread函数参数为待读入数据块的起始地址(void\*)，每个数据块的大小（字节数，size-t），最多允许写入的数据块个数（size-t）以及文件指针（FILE\*），返回实际写入的数据块个数
   15. feof函数：文件结束返回1，否则返回0
2. **希尔排序**
   1. 先取一个小于n的整数d1作为第一个增量，把文件的全部记录分成d1个组。所有距离为d1的倍数的记录放在同一个组中。先在各组内进行直接插入排序；然后，取第二个增量d2<d1重复上述的分组和排序，直至所取的增量dt=1(dt<dt-l<；…<d2<d1），即所有记录放在同一组中进行直接插入排序为止。
   2. 该方法实质上是一种分组插入方法。
   3. 代码为：void shellsort(int a[],int n)

{ int gap,i,j,t;

for(gap=n/2;gap>0;gap/=2）

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

for(j=i-gap;j>=0&&a[j]>a[j+gap];j-=gap)

{ t=a[j]; a[j]=a[j+gap]; a[j+gap]=t; } } }

1. **冒泡排序（另外还有空间换时间的排序法）**

for (i=0;i<s-1;i++){

for(j=0;j<s-i-1;j++)

if(b[j]>b[j+1])

t=b[j],b[j]=b[j+1],b[j+1]=t; }

1. **选择排序**

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

{

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

{

if(a[i]>a[j])

{

temp=a[i]; a[i]=a[j]; a[j]=temp;

} } }

1. **插入排序**

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

{ t=a[i];

for(j=i-1;j>-1&&a[j]>t;j--) a[j+1]=a[j];

a[j+1]=t; }

1. **指向常量的指针与指向常量的常指针以及const语句讲解**
   1. const 类型名 标识符=常量；
   2. 系统会分配只读区域存储将常量值放入其中，且不可再被修改，每次再次出现该标识符，即对该存储单元进行访问，而且定义是必须进行初始化；
   3. char \*const sp[2]={“abcd”,”1234”};sp是有两个元素的字符型常量指针数组，即每个元素都是常指针，分别固定指向两个字符串，但只可以修改指针所指向对象的值，不可修改地址的值。
   4. const char \*const sp[2]={“abcd”,”1234”};sp是一个指向常量字符串的常指针数组，地址值和对象值都不可修改。
   5. const char \* sp[2]={“abcd”,”1234”};则可以修改地址值，不可修改该常量字符串。
2. **链表的各个操作：**
   1. 构建链表的两种声明方式

函数原型1： struct stu \* creat(void); 调用语句：pstu = creat();

或：struct stu creat(); 调用语句：pstu = &creat();

函数原型2： void creat(struct stu \*\*); 调用语句：creat(&pstu);

* 1. **构建先进先出链表：构建第一个节点loc\_head，tail指向第一个节点，tail所指结点的指针域指向动态创建的新节点，tail指向新创建节点，赋值，再次动态创建下一个节点该步骤循环创建完毕后，最后tail指针域为NULL**

代码如下：

void create\_list\_v1(struct s\_list \*\*headp,int \*p)

{ struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) ;/\* 相当于\*p==0 \*/

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

\*headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

* 1. **构建后进先出链表：节点p后动态创建下一个节点，赋值，p的指针域指向值为NULL的head，紧接着p成为新的链头，进而循环时，新输入的数据始终是链头，从而达到后进先出的目的**

代码如下：

void create\_list\_v1(struct s\_list \*\*headp,int \*s)

{ struct s\_list \* loc\_head=NULL,\*p;

if(s[0]==0) ;

else {

while(\*s){

p=(struct s\_list \*)malloc(sizeof(struct s\_list));

p->data=\*s++;

p->next=loc\_head;

head=p;

} }

\*headp=loc\_head;

}

* 1. 递归方法统计链表节点数

（注意，可以通过添加判断条件来统计满足条件的节点数）

int count (struct s\_list \* head)

{ struct s\_list \* p=head;

if(p) return (1+count (p->next));

else return 0; }

* 1. 删除所有满足条件的节点

void del( Node \*\*headp,int x)

{ Node \*p,\*last,\*t;

p=\*headp;

while(p!=NULL) {

while(p->score!=x && p->next != NULL ) { /\* 查找值x的节点 \*/

last=p;

p=p->next;

}

if(p->score==x) { /\* 找到 \*/

if(p==\*headp) \*headp=p->next;

else last->next= p->next ;

t=p;

p=p->next;

free(t);

}

else break ;

} }

* 1. 递归查找第一个满足条件的节点

struct s\_list \* find\_nodes\_recursive(struct s\_list \* head,int n)

{ struct s\_list \* p=head;

if(p){ /\* 链表非空，查找 \*/

if(p->data==n) return p; /\* 找到，返回该结点的地址 \*/

else

find\_nodes\_recursive(p->next,n);/\* 递归查找 \*/

}

else return NULL;

}

* 1. 插入节点

struct s\_list \* insert\_nodes(struct s\_list \* head,int n)

{ struct s\_list \* current=head,\*after,\*other;

while(current->data != n && current != NULL)

current=current->next;

if(!current) return NULL; /\*如果没有找到符合条件的插入点，返回NULL\*/

after=current->next; /\*找到符合条件的插入点，建立after\*/

/\* other指向新建立结点 \*/

other = (struct s\_list \*)malloc(sizeof(struct s\_list));

scanf("%d",&other->data); /\*对新结点数据成员赋值\*/

if(after){ /\* 在链头或链中执行插入 \*/

other->next=after; /\*新结点指向后继结点\*/

current->next=other; /\*新结点成为插入点的后继结点\*/

}

else{ /\* 在链尾执行插入 \*/

other->next=NULL; /\*置新结点指针域为空\*/

current->next=other; /\*原尾结点指向新结点，新结点成为新的尾结点\*/

}

return current; /\*返回插入点的地址\*/

}

* 1. 删除节点

struct s\_list \* delete\_nodes(struct s\_list \*\*headp,int n)

{ struct s\_list \* current=\*headp,\*prior=\*headp;

while(current->data!=n&&current!=NULL){ /\* 查找成员值与n相等的结点 \*/

prior=current; /\* prior指向当前结点 \*/

current=current->next; /\* current指向下一结点 \*/

}

if(!current) /\* 没有符合条件的结点 \*/

return NULL;

if(current==\*headp) /\* 被删结点是链头\*/

\*headp=current->next;

else /\* 被删结点不是链头 \*/

prior->next= current->next;

free(current); /\* 释放被删结点的存储 \*/

return current;

}

* 1. 创建无限输入的链表

**基本思路：将输入的数存入节点p中，寻找链表的尾节点（注意，也有可能就是链头，要加判断条件），让其指向新建的节点p，之后循环创建，直到输入结束**

Node \* create\_list(Node \*\*pphead){

int number;

do {

printf("请输入一个umber:\n");

scanf("%d",&number);

if (number != -1) {

//add to linked-list

Node \*p = (Node \*)malloc(sizeof(Node));

p->value = number;

p->next = NULL;

//find the tail Node

Node \*tail = \*pphead;

if(tail){

while (tail->next) {

tail = tail->next;

}

//attach

tail->next = p;

}else{

\*pphead = p;

}

}

} while (number != -1);

return \*pphead;

}

1. **其他注意事项：**
2. 时刻注意变量初始化对之后程序的影响
3. 注意计算题中是否标注要说明类型名
4. %f输出时默认小数点后六位
5. 做题时先看循环变量初始值是多少，循环了几次，对于较为复杂的循环结构先改写
6. 改错题一题2分一般有2个错误
7. 一行一行的读代码，当填空时判断是否与先前代码重复
8. 观察是否有初始化，是否有定义，typedef定义的类型名是否重复，是否有指针悬浮，如果悬浮可以再定义一个数组或者动态分配存储空间（注意要有头文件）
9. 只有指针变量可以用->，而结构变量要用‘.’，当定义一个结构数组时，数组名便是一个指针，而单独的结构变量不是指针变量。
10. 在输出时时刻注意指针的指向是不是字符串首，
11. 注意循环变量的左自增还是右自增
12. 注意if判断语句的关联性（是多个 if还是else if的配对）

因为else总是和最近的if配对

#include<stdio.h>

int main()

{

float x;

scanf("%f",&x);

if (x>0) printf("1");

else if (x<0) printf("-1");

else printf("0");

}

1. 注意continue和break在循环结构中的正确选用
2. 注意数组名是个地址常量不可以自增
3. 特别注意switch语句中case后有没有跟有break语句
4. 交换两个字符串和交换两个数类似，要用地址传递实现，这是需要取指针的地址作为形参——

void fun(char \*\*pa, char \*\*pb)

{

char \*tmp=\*pa;

\*pa=\*pb,\*pb=tmp;

}

**第二部分 错题集**

1. 有关复杂的指针
   1. 请解释声明语句char\*(\*(\*pa)[8])(int(\*)(int));所声明标识符 pa 的数据类型。

答案：pa是一个指向8个元素的函数指针数组的指针，得2分；

函数指针数组中每个元素所指向函数返回值的类型为char\*，得1分；

函数指针数组中每个元素所指向函数的形参为指向一个返回值类型为int、含有一个int参数的函数指针，得1分。

* 1. 设a是有2个元素的函数指针数组，数组中每个元素所指向的函数带有一个数组指针类型的形参，该形参指向有3个元素的整型数组，而函数返回值的类型与该形参类型相同；请写出相应的声明语句。

答案：int (\*(\*a[2])(int (\*)[3]))[3];

3）设p是长度为3的函数指针数组，该数组中元素的类型为指向无参字符指针函数的指针。请写出相应的声明语句。 【答案】char \*(\*p[3])(void);

4）

1. 有关结构体的计算题

**1）设有说明：**

struct T{

int n[2];

char \*s;

} t[2]={{{1,0},"xy"},{{0, 1},"yx"}},\*p=t;

求(\*++p).s[\*(\*p).n]-1的值 答案：x

**2）设有说明：**

typedef unsigned short int UINT16;

UINT16 u=1,v=2,w=4,x=8,y=16,z=32;

计算x+y+z>-1ul（先计算原码再提升为unsigned类型） 答案：0

**4)** int a[3]={6,-20,8}; char c; unsigned short h=0x1234;

【1】a[1]>0U ? 1: 0

**5)** struct T{

char c; char \*s; int x;

} a[2]={{'C',"CHINA", 1},{'U',"US", 2}},\*pa=a;

【C】\*pa++ ->s++