三、顺序结构

顺序结构地程序设计框架

printf()

scanf()

putchar()、getchar()

习题

四、选择结构

1、两类表达式

关系运算符及关系表达式

逻辑运算符于逻辑表达式

2、if()……else……

if语句

if……else语句

if语句的嵌套

if……else if……else

3、switch(){……}

4、例题分析

优先级

五、循环结构

循环控制语句（四类）

方式1：

方式2：

方式3：

例题分析

- 指向指针的指针变量
- 空指针

构造类型

- 结构体类型
  - 构造结构体类型
  - 定义结构体变量
  - 使用结构体变量
  - 指向结构体数据类型的指针
- 链表
  - 动态存储分配函数<stdlib.h>
  - 链表操作
  - 访问链表
  - 链表操作
- 共用体类型
  - 公用体变量的定义：
  - 共用体变量的引用
- typedef

位运算（按位计算）

- 按位与运算（&）
- 按位或运算（|）
- 按位异或运算（^）
- 按位取反运算（~）
- 按位左移运算（<<）
- 按位右移运算（>>）

一、基础知识

1、数制及转换

1、四种数制：二进制、八进制、十进制、十六进制

二进制应用

内存与地址

十进制的定义

0 1 2 3 .....9十种编码

逢十进一

二进制的定义

0 1 两种编码符号

逢二进一

八进制的定义

0 1 2 3 .....7八种编码符号

逢八进一

在八进制前会加数字0以示其为八进制

十六进制的定义

0 1 2...9 A B C D E F 十六种编码符号（大小写均可）

逢十六进一

在十六进制前会加数字0x以示其为十六进制

2、进制的转换

十进制转二进制：除二取余倒排

65=1000001　255=11111111

二进制转十进制：按位乘权相加

11001=1x2^4+1x2^3+0x2^2+0x2^1+0x2^0=25

10111=　　11111111=

二进制转十进制：三位压成一位（从后向前组合，不够给前面加0）

（11001）2=（31）8

000->0

001->1

010->2

011->3

100->4

101->5

110->6

111->7

例:

（101111）2=（57）8

（11111111）2=（377）8

八进制转二进制：一位展三位

（363）8=（011110011）2

二进制转十六进制：四位压成一位

0000->0　　**1000->8**

0001->1　　**1001->9**

0010->2　　**1010->A**

0011->3　　**1011->B**

0100->4　　**1100->C**

0101->5　　**1101->D**

0110->6　　**1110->E**

0111->7　　**1111->F**

(3f9)16=(001111111001)2

十六进制转二进制：一位展成四位

数制应用

1、ASCII码

定义：唯一的二进制编码

A->1000001=65

B->66

a->97

b->98

2、内存

性质：

1、内存是由若干个存储单元构成的，每个存储单元内都可以存储一个内容值

2、内存的地址是一维的、线性的

3、内存中包括地址值和内容值

4、每个内存所存放的内容值大小是固定的（例：位、字节）

5、对于内存的值操作是先根据地址值找到内容值进行读写操作

6、内存关机则信息消失

内存单位：

位（bit） 字节（Byte） 8位=1字节 1024字节=1KB 1K=1024=2^10

1M=1024*1024=2^20

内存大小由地址位数决定

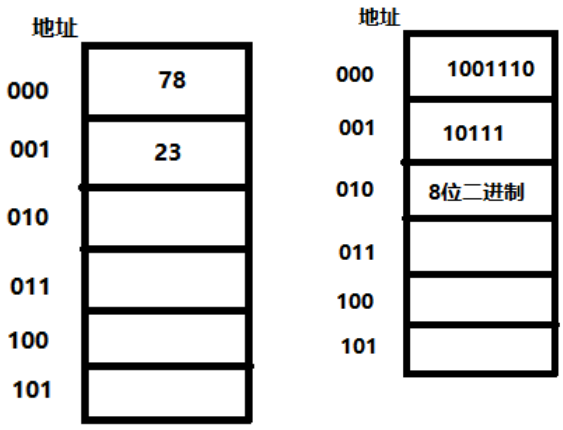
例：

1M的内存地址应为20位

3、机器码

真值->机器码（三种）

十进制->二进制



4、地址映射

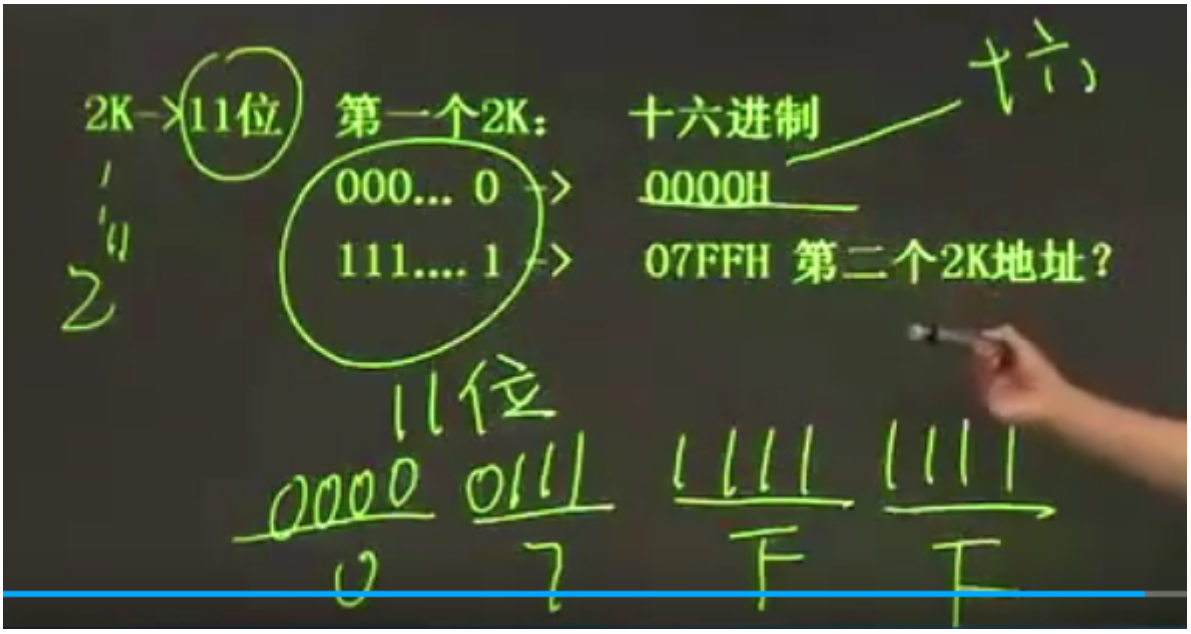
1024字节=1KB

1024KB=1MB

1024MB=1GB

1024GB=1TB

```
题1：16位的逻辑地址，逻辑地址空间是多大？物理内存32K，地址多少位？
答：1K = 2^10 逻辑地址空间大小=2^16=64k 地址：32k=2^5x2^10=2^15即15位
题2：假设逻辑地址32位，页面大小4k,逻辑地址分成多少个页面？
答：一页为4K=2^12 2^32/2^12=2^20=220 即分成2^20个页面，页号是20位
题3：4个2K存储芯片，每个2K存储空间起始地址是多少？
答：2k=2^11 11位
即起始地址为0000H-07FFH
第二个起始位置为0800H-0FFFH
第三个起始位置为1000H-17FFH
```



5、IP地址

```
题1：某主机IP地址为180.80.77.55子网掩码为255.255.252.0，问主机号部分多少位
```

```
提示：IPV4地址是32位，对应二进制1的是网络号，对应0的倍数是主机号倍数
255->11111111 252->11111100
255.255.252.0->11111111.11111111.11111100.00000000
即主机号部分是10位

题2：求IP分组到达时，经过多少个路由器
提示：两个十六进制的差
40H-31H=0FH=15个
```

二、C语言组成及数据

1、C语言组成

组成：由若干个文件组成

文件：由若干个函数体组成

函数：由函数头+函数尾组成

函数头：四部分->函数名+()

函数体：{}+语句

语句：三类语句

1、注释语句 /* */

2、定义语句；

3、执行语句；

2、标识符（三类）

1、**保留字（关键字）**：共32个（均为小写）。如：int float if else for

2、**预定义标识符**：预先定义并具有一定特定含义的标识符 scanf printf include

3、**用户定义标识符**：由用户根据需要定义的标识符。如：变量名、数组名、函数名等

【注】1、用户自定义标识符：第一个字符必须是字母或下划线，后面由字母、数字、下划线组成

2、有大小写区分

3、数据与数据类型

四类数据：常量、变量、表达式、函数

常量：值不发生变化的量

变量：值随时随地改变的量

表达式：用运算符将数据连接起来的合法式子

函数：调用具有一定功能的函数作为运算量

数据与数据类型

数据四大类型：基本类型、构造类型、指针类型、空类型

1、基本类型：整型int、字符类型char、浮点型【实型】(单精度float、双精度double）、枚举enum

2、构造类型：数组型、结构体类型struct、共用体类型union

3、指针类型

4、空类型 void

1、常量（四类）

：整型常量、实型常量、字符型常量、字符串常量

1、整型常量：三种形式（十进制、八进制、十六进制）

2、实型常量：两种形式（小数、指数）

小数形式：由数字和小数点组成 如：123.23　　123. .456 0.456

指数形式： 用E或e后紧跟一个整数表示以10为底幂数 如：123E-5

【注】：字母E或e之前之后都必须有数字，且后是整数

字母E或e前后及各数字之间不能有空格

3、字符型常量：四种形式（常规、转义、八进制、十六进制）

一堆单靠引号括一个字符

转移字符常量：必须以一个反斜杠\开头

常用转义符：

\n　表示换行符

\t　表示制表符

\b　表示退格符

\r　表示回车符

不能存在'\'这样的写法，单斜杠以 '\\'表示，不i能存在''

4、字符串常量：使用一堆双引号括起来的若干个字符

【注】每一个字符串尾都有一个**字符串结束符号**'\0'

C语言中没有字符串变量。一定要区分字符常量和字符串常量

2、变量

使用规则：先定义后使用

定义格式：数据类型名　　变量名列表;

例：int i;int j;

int i.j.k;

int i = 2,j = 4,k = 6; /*赋初值*/

【注】：1、变量必须先定义后使用（int long float double char(与常量类型一直))

2、变量名树与用户标识符，遵守用户标识符命名规则

3、在同一“函数体”{}中不能定义同名变量

4、同时定义多变量，必须用逗号分开

5、变量可赋初值（初始化），无初值是随机值

3、函数

函数(标准函数、自定义函数)

#include<math.h>

4、表达式

定义：用运算符将数据连接起来的合法式子（具有唯一确定的值）

运算符的三个属性：**功能、优先级、结合性**（运算方向）

不同级运算符先考虑优先级，同级情况下运算方向：从左向右

1、算数运算符及算数表达式

+　-　优先级为4，结合性（从左向右）

*　/　%　优先级为3（从左向右）

求余（%）：两边运算对象必须式整型

余数的符号与被除数的符号保持一致（例17%3=2 -17%3=-2）

【注】：单独的常量、变量或函数调用都是C语言合法表达式

但凡表达式都有确定的值

2、赋值运算符及赋值表达式

赋值运算符：=　（优先级为14）

格式：变量名=表达式

赋值运算符的功能：将表达式的值赋给“=”左边的变量

括号（）的优先级为1级

【注】1、赋值运算符左边只能是一个变量（a+b=3 不合法　a+(b=3)合法表达式）

2、赋值运算符右边可以是合法表达式

3、赋值表达式的值就是赋值运算符左边变量值

4、实型数据赋给整形变量时，实型小数舍去；整形数据赋给实型变量时系统自动将整形数据转成实型数据

3、符合赋值运算符

+=、-=、*=、/=、%=、&=、|=、>>=、<<=　（优先级为14级，运算方向为从右向左）

注：复合赋值运算符的两个运算符中间不能有空格

例　A+=3 -> A=A+3

a/=7+3　-> a=a/(7+3)

```
例1：若int a=5,b=8则下列表达式的值是多少？用完表达式后变量a和b的值分别是多少？
a+=b/=a
答：
b/=a -> b=b/a b=1
```

```
a+=1 -> a=a+1 a=6
表达式值为:6
a的值：6
b的值：1

 例2:若有int a=5,b=9;float c ;则表达式c=b/a+1.2的值是多少
 答：

```

4、自增自减运算符

自增运算符：++　　i++　++i

自减运算符：--　　i--　--i

注：只能用于变量不能用于常量或表达式

隐式转换——自动完成（向高类型靠）

char(1)->int(2)->unsigned(2)->long(float(4))->double(8)

e的前面要有数，e的后面要有整数

逗号，的优先级为15

5、逗号运算符及其表达式

格式：表达式1，表达式2，表达式3，表达式4……表达式n

功能：逗号表达式的值就是表达式n（最后一个表达式）的值，求职顺序是从左到右一次求解

```
如int a,b;
则表达式a=3,b=1的值为?
答：表达式值为1
运算结束后a为？b为?
答：a=3,b=1
```

6、强制类型转换（显式转换）

格式：

（类型名）表达式 或 （类型名）　（表达式）

利用强制类型转换运算符可将一个表达式的值转换成指定的类型

如：float x=123.456,y;

1. (int)x　(int)(x)　　//将x强制转换为int

2. (int)x+y　　//仅强制转换x为int

3. (int)(x+y)　　//将x+y的值强制转换为int

进行强制类型转换，得到的是一个中间值，而原来表达式或变量的类型未发生改变

如：若变量x为int型，则表达式(float)x的结果是一个单精度的值，单是x还是int型

三、顺序结构

1、定义：从main()考试，由上往下一条一条地执行

2、三大类语句

执行语句（五类）：

控制语句（9条）

函数调用语句；

表达式语句；

空语句；

复合语句{}　//{}算一条语句

定义语句

注释语句

顺序结构地程序设计框架

```
#include<头文件名>
main()
{
 输入
 计算
 输出
}
```

printf()

格式：printf("普通/占位符",输出列表的值)

功能：按格式将值输出

注意：“占位符”以%号开始的使用,分隔

“普通字符”将直接输出

输出值，先计算后输出

常用占位符

占位符	含义
%d	带符号的十进制整数
%c	输出一个字符，不输出单引号
%s	输出一个字符串，输出时不输出引号
%f	输出一个实型数，隐含输出六位小数，可使用%.nf修改n的值设置输出位数

	用%.nf修改n的值设置输出位数
%u	无符号十进制整数
%o	无符号的进制整数
%X,%x	无符号十六进制整数
%E，%e	输出直输形式的实型数

scanf()

格式：scanf("普通/占位符"，地址列表)

如;

```
int x;
scanf("%d",&x);
```

如果输入时为普通字符必须原样输入

```
scanf("a=%d.b=%d",&a,&b);
即在控制台应写入 a=2,b=3;//示例
```

“占位符”，输入时四种输入

在没有普通字符的情况下（即没有表明分割符） 则可以使用空格、回车、空格+回车、回车+空格

当占位符为%c时，只有一种输入 ab回车

当占位符为混合输入时

在进行输入时不能忘记&取地址符号

```
//求取某同学三门功课的平均分
#include<stdio.h>
int main{
 int a,b,c; //定义变量
 scanf("%.2f,%.2f,%.2f",*&a,&b,&c); //输入成绩并赋值
 float aver; //定义平均分变量
 aver=(a+b+c)/3; //计算平均分
 printf("三门成绩的平均分为：%.2f",aver); //输出平均分
}
```

putchar()、getchar()

字符数据的输入输出

1、putchar()

格式：putchar(字符常量或字符变量)

功能：输出一个字符但不输出单引号

如:

```
putchar('a');
putchar('\n'); //回车
putchar(100); //根据asci码输出对应值 d
```

2、getchar()

格式：getchar()

功能：接收**一个**从键盘输入的字符

注：getchar()没有任何参数，函数的返回值就是输入的字符

```
char a,b;
a=getchar();
b=getchar();
若输入为：1<回车>
则变量a的值为1 变量b的值为<回车>
```

习题

已知两个两位数a和b,要求按照如下规则合并成一个四位数，其中a的个位和十位分别作四位数的千位和个位，而b的个位数和十位分别作四位数的百位和十位，求这个四位数?

```
#include<stdio.h>
int main(){
 int a,b;
 int d,e,f,g;
 scanf("%d,%d",&a,&b);
 d = (a / 10)*1;
 e = (a % 10)*1000;
 f = (b / 10)*10;
 g = (b % 10)*100;
  printf("%d",d+e+f+g);
}

```

四、选择结构

定义：从main（）开始，由上往下有些语句执行有些不执行

两类语句——执行语句：控制语句

1、两类表达式

关系运算符及关系表达式

关系运算符的结果是逻辑值，即0或1

<　<=　>　>=　　//结合性为6级

== !=　　//结合性为7级

三要素：1、每个运算符有自己的**功能**

2、每个运算符有自己的**优先级**

3、每个运算符的**结合性**

逻辑运算符于逻辑表达式

2级为单目运算符（运算方向为从右向左）

3-12级位双目运算符（运算方向为从右向左）

&&（逻辑与）　　//结合性为11级　　全真为1，有0为0

||（逻辑或）　　//结合性为12级　　全0为0，有真为1

!(单目，逻辑非)　　//结合性为2级　　真变假。假变真

在C语言中逻辑表达式的值只有1或0两种值。其中1表示“真”，0表示“假”

对于&&操作，第一个表达式为假（0）则短路，对于||操作，第一个表达式为真（1）则短路

判断表达式方法：1、判断是否合法　2、判断是否短路 3、判断优先级，同级考虑结合性

2、if()……else……

if语句

格式1;

if（表达式）

　　语句序列1；if子句

后继语句

if……else语句

格式：

if(表达式)

　　语句序列1；

else

　　语句序列2；

```
#include<stdio.h>
int main(){
 int m = 5;
 if(m++>5){
 printf("%d",m);
 }else{
 printf("%d",m++); //6
 printf("%d",++m); //8
 }
}
```

```c
#include<stdio.h>
int main(){
 int m = 5;
 if(m++>5){
 printf("%d",m--);
 printf("%d",m);
 }else{
 printf("%d",m); //6
 }
}
```

if语句的嵌套

if(表达式1)

    if(表达式2)

        语句序列11;

    else

        语句序列12;

else

    语句序列2;

if……else if……else

```c
#include<stdio.h>
int main(){
 int x;
 scanf("%d",&x);
 if(x<0){
 printf("-1\n");
 }else if(x>0){
 printf("1\n");
 }else{
  break;
  }
}
```

3、switch(){……}

格式:

switch(表达式){

case常量表达式1：子句1

case常量表达式2：子句2

……

case常量表达式n：子句n

default:　　子句n+1

}

【注意事项】

1、switch后必须用小括号将表达式括起

2、case后常量不能出现相同的值

3、case后常量整型或字符型，不能有变量和逗号运算符

4、case和default只是一个入口标记，不起终端作用，他们的顺序可以任意颠倒

5、case和default必须出现在switch语句中

4、例题分析

```
if (a==b){
 if(b==c){
 printf("****");
 }
}else{
 ptintf("aaa");
}
```

```
int a=5,b=2;
int c=0,d;
d = a/b+(b&&c||a)
d=2+1=3;

int a,b=1,c=3;
a=(b<c)+!b;
a=1+0=1;
```

优先级

优先级	运算符	结合律
1	后缀运算符：[]　()　·　->　++　--(类型名称){列表}	从左到右
2	一元运算符：++　--　!　~　+　-　*　&　sizeof_Alignof	从右到左
3	类型转换运算符：(类型名称)	从右到左
4	乘除法运算符：*　/　%	从左到右

5	加减法运算符：+　-	从左到右
6	移位运算符：<<　>>	从左到右
7	关系运算符：<<=　>>=	从左到右
8	相等运算符：==　!=	从左到右
9	位运算符 AND：&	从左到右
10	位运算符 XOR：^	从左到右
11	位运算符 OR：|	从左到右
12	逻辑运算符 AND：&&	从左到右
13	逻辑运算符 OR：||	从左到右
14	条件运算符：?:	从右到左
15	赋值运算符： =　+=　-=　*=　/=　%=　&= ^=　|= <<=　>>=	从右到左
16	逗号运算符：，	从左到右

1、（）、2、单目 3-12、双目（算数高于关系，关系高于逻辑） 13、三目 14、条件运算符 15、赋值运算符 16、逗号

五、循环结构

1、定义：从main()开始，从上向下，是的某些语句重复执行

2、循环结构的程序：在顺序结构中加入循环控制语句

循环控制语句（四类）

方式1：

```
while(表达式){
循环体语句序列;
}
后继语句;
```

说明：

遇到while会进行表达式的求解，表达式只有真假两类，若为真，执行循环体，然后继续进行表达式的判断，直到表达式结果为假，才会跳出循环语句去执行后继语句。

while和它控制的语句（整个算一条）

复合语句算一条，若没有花括号，则仅控制随后的一条语句，值执行一次。

例题：

```
#include<stdio.h>
int main(){
 int n = 6;
 while(n < 9){
 printf("%d\n",n);
 n++;
 }
 printf("%d\n",n--); //9
 printf("%d\n",n); //8
}
```

方式2：

```
do{
循环体语句序列;
}while(表达式);
后继语句;
```

说明：

do循环内的循环体语句至少执行一次（首先会执行do语句，之后再进行while)

若while表达式的值为真，则执行do后的语句，否则执行后继语句

do只拿就近控制一条语句，若需要控制多条需要使用{}

do和while之间只能有一条语句

例题：

```
while(表达式);
 语句1;
 语句2
 语句3;
一共有四条语句
```

```
do{
 语句1;
 语句2;
 语句3;}
while(表达式);

```

方式3：

```
for(表达式1;表达式2;表达式3){
循环体语句序列;s
}
```

```
后继语句;
```

说明：

若表达式2永远为真则为死循环

若没有表达式2，系统会默认填充为1，即永为真

表达式3可以缺少

例题：

```
#include<stdio.h>
for (i = 1;i++< 4; )
循环体执行多次？结束时变量i的值为？
循环体执行三次，i为5
#include<stdio.h>
for (i = 1;++i< 4;i++);
循环体执行多次？结束时变量i的值为？
循环体执行一次，i为4
```

例题分析

1+2+3+.....+100使用三种循环

```
//使用while循环计算100以内整数和
#include<stdio.h>
int main(){
 int i,sum = 0;//初始化变量
 while(i <= 100){//当i<100时进入while循环
 sum += i;
 i++;
 }
 printf("%d\n",sum);//打印输出运算结果
}
```

```
//使用do...while循环计算100以内整数和
#include<stdio.h>
int main(){
 int i = 0,sum = 0;//初始化变量
 do{
 sum += i;
 i++;
 }while(i <= 100);
 printf("%d\n",sum);//打印输出运算结果
}
```

```c
//使用for循环计算100以内整数和
#include<stdio.h>
int main(){
 int sum = 0;//初始化变量
 for(int i = 0;i <= 100;i++){
 sum += i;
 }
 printf("%d\n",sum);//打印输出运算结果
}
```

```c
//使用goto计算100以内整数和
#include<stdio.h>
main(){
 int i,sum = 0;
 i = 1;
 LP:sum = sum + i;
 i++;
 if(i <= 100) goto LP;//条件为真则执行LP
 printf("sum = %d",sum);
}
```

方式4：

if(){}else与goto配合使用

两个控制语句——break continue

break

功能：中止退出。范围：循环体中和switch体中

```c
#include<stdio.h>
main(){
 int i;
 for(i = 1;i < 4;i++){
 i++;
 break;
 }
 printf("%d",i);//i = 2
}

```

循环结束的两种情况：表达式为假；break

continue

功能：结束一次循环继续

范围：循环体中

```
#include<stdio.h>
main(){
 int i;
 for(i = 1;i < 4;i++){
 i++;
 continue;//continue后的语句不执行
 }
 printf("%d",i);//i= 5
}
```

1、for循环跳回表达式

2、while/do...while跳回表达式

练习题

1、下面程序的运行结果是;

```
#include<stdio.h>
main(){
 int num = 0;
 while(num <= 2){
 num++;
 printf("%d\n",num);
 }
}
输出结果为:
 1
 2
 3
```

2、设整形变量x数为当前值为3，执行一下魂环语句后，输出的结果是

```
do
 printf("%d\n",x-=2);
while(!(--x));//判断真假

输出结果为:
1 -2
```

3、执行循环语句

```
for(x = 0;y = 0;y != 250||x < 4;x++)
 y += 50;
其循环体共执行多少次?
4次
```

4、下面程序的输出是

```
#include<stdio.h>
main(){
 int x = 3,y = 6,a = 0;
 while(x++ !+ (y -= 1)){
 a += 1;
 if(y < x){
 break;
 }
 }
  printf("x = %d,y = %d,a = %d\n",x,y,a);
}
输出结果为:
5 4 1
```

5、什么程序结构

双循环

```
#include<stdio.h>
main(){
 int i,j;
 for(i = 1; i < 3;i++){
 for(j = 1;j < 4;j++){
 printf("i = %d,j = %d\n",i,j);
 }
 }
 printf("i = %d,j = %d\n",i,j);
}
输出:
i = 1,j = 1
i = 1,j = 2
i = 1,j = 3
i = 2,j = 1
i = 2,j = 2
i = 2,j = 3

i = 3,j = 4
```

双循环：1、先执行外循环再执行内循环。 2、当内循环的循环体执行完后回到内循环的表达式三上，只有当内循环的表达式二为假时才回到外循环的表达式三上面，然后内循环从表达式一开始重新执行。　　3、外循环每取一个值，内循环要完整的走一圈。

6、双循环

```c
#include<stdio.h>
main(){
 int m,n;
 printf("enter m,n);
 scanf("%d,%d",&m,&n);//36,24; 24,17;
 while(m != n){
 while(m > n) m -= n;
 while(n > m) n -= m;
 }
  printf("m = %d\n",m); //12 ; 1
}
```

7、以下程序的运行结果是

```c
#include<stdio.h>
main(){
 int i,j,m;
 for(i = 5; i >= 1;i--){
 m = 0;
 for(j = i;j <= 5;j++)
 m = m+i*j;
 }
 printf("%d\n",m); //15
}
```

8、执行以下程序后，a的值为：

```c
#include<stdio.h>
main(){
 int a,b;
 for(a = 1,b = 1;a <= 100;a++){
 if(b >= 20) break;
 if(b % 3 == 1){
 b +=3;
 continue;
 }
  b -= 5;
  }
}
运行结果：
a = 8 b = 22
```

9、以下程序的运行结果是

```c
#include<stdio.h>
```

```
main(){
 int n = 4;
 while(n--){
 printf("%d",--n);
 }
}
运行结果;
 20
```

10、以下程序的运行结果是

```
#include<stdio.h>
main(){
 int n = 4;
 while(n--);
 printf("%d",--n);
 }
}
运行结果:2
```

11、以下程序的运行结果为;

```
#include<stdio.h>
main(){
 int i = 5;
 do{
 switch(i % 2){
 case 4:
 i--;
 break;
 case 6:
  i--;
  continue;
 }
 i--;
 i--;
 printf(%d",i);
 }while(i>0);
}
输出结果
31-1
```

六、数组

数组的特点;

同一数组中的所有元素都属于同一种数据类型（int char float）数组元素用数组名和相应的下标来确定。

方法;

一个数组元素其实就是一个变量（可以称为带下表的变量）

一维数组

一维数组的定义格式;

类型名　　数组名[常量/常量表达式];

如：　float s[25];　　//定义一个长度为25的float型数组

定义数组的注意事项

1、数组名树与用户定义标识符。

2、定义一维数组时数组名后必须用一个方括号[]将常量表达书括起来，常量表达式的值表示所定义数组共有多少个元素（数组长度或数组大小）。

3、定义数组时方括号中的表达式不能含有变量。且表达式的值必须要**大于零的正整数**。

4、C语言每个数组再内存中分配空间时是连续分配的。

【注】数组必须先定义后使用

数组不能整体用数组名，只能使用数组元素（格式 数组名[下标]）

一维数组的初始化;

格式:

类型名　　数组名[常量表达式]={初始值1，初始值2，初始值3……};

一维数组的使用

1、以下程序的功能是什么，结果是多少?

```
#include <stdio.h>
main(){
 int a[]={1,2,3,4},i,j,s=0;
 j = 1;
 for(i = 3;i >= 0; i--){
 s = s+a[i]*j;
 j = j* 10;
 }
 printf("s=%d\m",s);
  }
 }
输出结果:1234
功能:将数组的元素值组成一位十进制数
```

二维数组

二维数组的格式定义

类型名　　数组名　　[常量表达式1][常量表达式2];

示例;

int a[10][10];//10行10列

　　行　列

【注】

1、数组名要使用用户自定义标识符的规则

2、定义二维数组第一个方括号常量表示行数，第二个方括号表示列数

3、定义数组时方括号中的表达式不能含有变量，且表达式的值必须时大于0的正整数

4、在内存中二维数组元素的存放也是连续的，先行后列原则

二维数组的初始化

方法1：将初始值依序放在一对{}中，与一维数组初始化格式相同

例：　int a[2][3]={1,2,3,4,5,6};

方法2;定义二维数组同时，按行初始化每一行初值均用一堆{}括起，采用嵌套的{}格式。

例：int a[2][3]={{1,2,3},{4,5,6}};

【注意】

　　定义二维数组的同时给数组初始化，则可以省略行数，但列数不能省略。仅定义数组时行列都不能少。

二维数组的使用

1、程序输出结果?

```
#include<stdio.h>
int main(){
 int a[3][3]={{1,2},{3,4},{5,6}},i,j,s = 0;
 for(i = 0;i < 3;i++){
 for(j = 1;j <= i;j++){
 s += a[i][j];
 }
 }
 printf("%d\n",s);
}
输出结果为：10
```

2、设有3x3矩阵，编写程序求解矩阵B=A+A^T,即矩阵B为矩阵A及其转置矩阵A^T之和

```
#include<stdio.h>
int main(){
 int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};//存放原矩阵内容
 int b[3][3]={0};//内容为0的矩阵，存放转置后的内容
```

```
int c[3][3]={0};//内容为0的矩阵，存放原矩阵与转置后的矩阵和
int i,j,s;
printf("原矩阵\n");
for(i = 0;i < 3;i++){
for(j = 0;j < 3;j++){
 printf("%3d",a[i][j]);
 }
 printf("\n");
 }
 printf("\n转置矩阵\n");
 for(i = 0;i < 3;i++){
 for(j = 0;j < 3;j++){
 b[i][j] = a[j][i];
 printf("%3d",b[i][j]);
 }
 printf("\n");
 }
 printf("\n原矩阵与转置矩阵和\n");
 for(i = 0;i < 3;i++){
 for(j = 0;j < 3;j++){
 c[i][j] = a[i][j]+b[i][j];
 printf("%3d",c[i][j]);
 }
 printf("\n");
 }
 //可以不要b矩阵直接求原矩阵与转置矩阵之和
 printf("\n原矩阵与转置矩阵和\n");
 for(i = 0;i < 3;i++){
 for(j = 0;j < 3;j++){
 c[j][i] = a[i][j]+a[j][i];
 printf("%3d",c[i][j]);
 }
 printf("\n");
 }
}
```

字符串

C语言中没有字符串变量，因此存放方法分为两种：1、字符数组来存放　　2、字符型指针变量

char a[5];

字符数组初始化：

char s[]={'s','s','d','g'};

与其他类型数组的初始化方式一样，只是其初始值是字符。

字符串

因为字符串最后都有一个字符串结束符（‘ \0'），所以用字符数组来存放字符串时一定要有一个元素存放结束符'\0'

1、字符串常量最后又一个'\0'

如：“abcd"由**五个**字符组成，” “等价于"\0"，是一个空字符串

2、用字符串给一个字符数组进行初始化有三种情况：

a、chat a[]="abcd";

b、char a[]={"abcd"};

c、char a[]={'a',,'b','c','d';'\0'};

字符串的使用

1、字符串的存储

方法1:scanf()；键盘输入 //在此变量不需要添加取地址&，数组中直接显示的时地址

printf();输出

注：用%s格式输入时，遇到空格符或回车符则自动结束输入

输出时则从当前地址开始知道遇到结束符

例：char a[3];

scanf("%s",a);

printf("%s",a);

方法2：puts()输出

格式；puts(字符数组或字符串常量)

功能：在屏幕上输出字符数组中的字符

注：该函数输出与用%s格式输出一样，只是 ‘\0'转换成'\n'输出

例：puts("abc");

方法3：字符串初始化

例：char a[]={"abcd"};

方法4：strcpy()赋值拷贝

2、字符串函数（处理）

strcat()

格式：strcat(字符数组1，字符数组2)；

使用时应引入 #inculude<string.h>

功能：连接字符串，其连接后的结果发返回的是第一字符数组的地址

示例 char a[18]="jack";

```
char b[10]="jason";
strcat(a,b);
printf(a);//jackjason
printf(b);//jsaon
```

strcpy()

格式：strcpy(字符数组1，字符数组2)；

使用时应引入 #inculude<string.h>

功能：将字符数组2中的字符串替换到字符数组1中。函数值为字符数组1的首地址。

例；

```
char str1[10]={"123456"} ,str2[10]={"abc"};
strcpy(str1,str2);
printf("%s\n",str1);//str1=>abc
printf("%s\n",str2);//str2=>abc
```

strcmp()

格式：strcmp(字符数组1，字符数组2)；

使用时应引入 #inculude<string.h>

功能：函数返回值相同位置不同字符的ASCII码差值

```
strcmp("abc","abc"); // =>0
strcmp("abc","abc"); //=>-3
strcmp("abc","abc");//=>99
```

注：到不同字符时不再比较，直接计算ACSII差值

strlen()

格式：strlen(字符数组)；

使用时应引入 #inculude<string.h>

功能：求出字符数组的实际长度（不包括结束符）

```
char w[10]={"45628345"};
printf("%d",strlen(w));//8
char p[] = {"abc\018\0"};
printf("%d",strlen(p));//5
```

3、例题分析

```
#include<stdio.h>
void main(){
 char pa[15] = {"12345"};
 char *pb="EFG";
 puts(pa);//12345
 puts(pb);//EFG
}
```

数组名是常量，常量不能放在赋值号的左边

```
1
2 #include<stdio.h>
3 void main(){
4  char pa[15] = {"12345"};
5  char *pb="EFG";
6  pb = pa;
7  strcpy(pa,"abcd")
8  puts(pa);//sbcd
9  puts(pb);//abcd
10 }
```

七、函数

函数的定义

C语言的框架有两种：

1、一个main()　　单框架

2、一个main()+多个子函数　　复合框架

【注】

1、一个源程序文件可由一个或多个函数组成

2、一个C语言程序可以由一个或多个源程序文件组成

3、C程序执行总是从main()开始，结束于main()，可调用其他函数

4、函数不能嵌套定义(即在自定一函数内再次定义一个新的函数），但可以相互调用，不能调用main()函数的分类

a、无参函数和有参函数

b、库函数和用户自定义函数+main()

函数的定义

函数返回值类型　　函数名（形式参数列表）{

函数体;

}

说明:

1、函数日可以没有语句，但不能没有花括号，函数名后必须有一堆小括号

2、定义有参函数时，形参的定义可以采用传统方式或现代方式两种

传统方式：

int max(x,y)

int x,y;//不能定义形参意外的其他变量

{}

现代方式：

int max(int x,int y){ }

3、含相互返回值类型

int类型（即非void型）;在函数体内存在return 表达式;

void型（即空类型）： 在函数体内不能存在return 语句;

函数调用

方式1：非void型效用

变量名 = 函数名（实参列表）;

示例：

```
#include<stdio.h>
int fun(int x,int y){ //定义一个返回值类型为int函数名为fun的函数，形参列表为x,
 return x>y?x:y; //当x>y为真时返回x的值，否则返回y值
}
main(){ //主函数
 int a = 9,b = 7,max; //定义变量
 max = a; //将a的值传递给max
 max = fun(max,b); //调用funh函数，参数列表为max,b将结果赋值给max
 printf("max=%d",max); //9
 }
```

方式2：void型调用

函数名（实参列表）;

```
#include<stdio.h>
void fun(int x,int y){
 int z;
 z = x > y?x:y;
 printf("z = %d",z);//9
}
main(){
 int a = 7,b = 9,max;
 max = a;
  fun(max,b);
  printf("max = %d",max);//7
 }
 最终的输出结果：
 z = 9max = 7
```

函数使用例题

原型声明

方式1：#include<头文件>　　//库函数

方式2：声明格式　　//用户自定义函数

函数类型　函数名（形参类型1 形参1，形参类型2 形参2……）；

函数类型　函数名（形参类型1，形参类型2……）；

【注】子涵数必须定义在主函数上方（若子函数在主函数下方则必须使用函数原型声明语句）

```
#include<stdio.h>
void sum(int x,int y);//函数原型声明
main(){
 sum(3,4);//函数的调用
}
void sum(int x,int y){ //定义sum函数
 printf("%d",x+y);
}
输出结果;7
```

函数调用过程

在函数调用时系统将实参值对应地（按位置次序对应）传给形参，是一种值的单向传递。实参与形参之间由一个关系。

```
#include<stdio.h>
int fun(int x.int y){
 return x>y?x:y;
}
main(){
 int a[3] = {9,2,5},i,max;
 max = a[0];
 for(i = 1;i < 3;i++){
 max = fun(max,a[i]);
  }
  printf("max=%d",max);//9
 }
```

变量三属性

类型

类型名 变量名列表;

例：　int s,d;　//这是一个定义语句，定义了两个整形变量s,d

作用范围（空间）

局部变量（ 内部变量）

定义：在**一个函数内部**定义的变量为局部变量

1、局部变量只能在它所在的函数内有效

2、在不同的函数中可以出现同名的变量，他们分别属于不同的变量（作用范围不同）

3、复合语句中定义的变量只能在此复合语句中有效

4、所有形式参数都是局部变量

```c
#include<stdio.h>
main(){
 int a = 3,b = 9;{ //以下为复合语句
 int a = 1,c = 7;
 a = a + b;
 b = c + a;
 printf('a=%d,b = %d',a,b); //a=10,b=17
 }
 printf('a=%d,b = %d',a,b);//a=3,b=17
}
```

```c
#include<stdio.h>
voif fun(int x,int y){ //局部变量
 int a,b;//局部变量
}
main(){
 fun();
}

```

全局变量（外部变量）

定义：在函数外部定义的变量为全局变量

1、全局变量的有效范围是在文本文件内定义该变量的位置开始到本文件的结束

2、全局变量可以在它的有效范围内被每个函数使用

3、在同一文件中若全局变量与局部变量同名，局部变量“屏蔽”全局变量

```c
#include<stdio.h>
void fun1(){
 a = 100;//局部变量
 printf("%d\n",a);
}
void fun2(){
 a = 1000;//局部变量
```

```
 printf("%d\n",a);
}
main(){
 a = 10000;//全局变量
 fun1(); //100
 fun2(); //1000
 printf("%d\n",a);//1000
```

```
#include<stdio.h>
int a,b;//全局变量
void fun(){
 a = 100;
 b = 200;
}
main(){
 int a = 5,b = 7;//局部变量
 fun();
 printf("%d%d\n",a,b);//57
}
```

存储类别（时间）

auto类别变量

1、auto类别变量用完后释放所占空间

2、局部变量默认为auto类别，无初始化时，初值为随机值

3、使用时间短，一般都为auto类别变量

static

1、static类别从定义到程序运行结束均占用存储空间

2、全局变量默认为static类别，无初始化时，初值为0

3、**static类别变量只进行一次初始化**

```
#include<stdio.h>
int fun(int x,int y){
 static int m = 0,i = 2;
 i += m+1; //3 12
 m = i+x+y;//8 17
 return m;
}
main(){
 int j = 4,m = 1,k;
 k = fun(j,m);
 printf("%d",k);//8
```

```
 k = fun(j,m);
 printf("%d",k);//17
}
```

```
#include<stdio.h>
int d = 1;
fun(int p){
 static int d = 5; //静态变量只执行一次初始化
 d += p; //6 15
 printdf("%d",d);
 return(d)
}
main(){
 int a = 3;
 printf("%d",fun(a+fun(d)))//15
}
```

register（寄存器）

1、register类别只能是局部变量才能被说明

2、一般不能直接使用

extern

1、extern类别变量可以加大变量的作用范围

2、格式说明

格式1：定义是说明类别

存储类别 类型名 变量名；

格式2：分别定义、说明

类型名 变量名；

存储类型 变量名；

```
#include<stdio.h>
main(){
 ectern x,y;
 printf("%d,%d\n",x,y);
}
int x,y;
```

预编译命令

预编译

预编译命令标志#

文件包含（include）

#include<>

#include" "

多个文件

#include<文件名.c>

宏

第一种：无参宏定义

格式：

#define 宏名 宏内容

功能：用一个指定的标识符（宏名）来代表一串字符（宏内容）

例;

#define PI 3.1415926

#define N 19

【注】

1、宏名一般用大写字母表示，遵守用户自定义标识符明明规则

2、#define可以在函数外定义也可以在函数内定义，但该命令应该在单独的一行上

#undef命令可以提前种植宏名的作用域

3、在进行宏定义时，可以引用已经定义的宏名进行层层置换

4、在进行宏提换时，必须先替换完所有的宏后再运行，同时替换过程不能乱加括号

第二种：带参宏定义

#define 宏名（参数列表） 宏内容

功能：提供了一种更加灵活的替换方式

如：#define s(x,y) x*y+2

【注】

1、在定义有参宏时，参数列表必须用一对小括号括起且小括号和宏名之间不能有空格

2、对有参宏名进行替换时，需要将形参改成相应的实参，并且注意分清形参和实参的对应关系

```c
#include<stdio.h>
#define N 5+4
int main(){
 a = N * N + 30;//先替换再计算
 //a = 5+4*5+4+30=59
 printf("d",a)//59
}
```

```c
#include<stdio.h>
#define s(a,b) a*b
```

```
main(){
 printf("s=%d",s(3,5));//3*5 = 15
 printf("s=%d",s(3+2,5+7));//3+2*5+7=20
 printf("s=%d",s((3+2), (5+7));// (3+2) * (5+7) =60
}
```

```
#include<stdio.h>
#define SQR(x) x*x
main(){
 int a = 18,k = 2,m = 1;
 a /= SQR(k + m)/SQR(k + m); //a=/2+1*2+1/2+1*2+1 a/= 7 a=2
 printf("%d",a);
}
```

复习题

```
#include<stdio.h>
int a = 1;
main(){
 int s = 3;
 {
 int a = 5;s += ++a;
 }
 s += ++a;
 printf("%d",s);//11
}
```

```
#include<stdio.h>
int w = 3;
fun(int k){
 if(k ==0) return w;
 return(fun(k -1) *k);
}
main(){
 int w = 10;
 printf("%d\n",fun(5)*w);//3600
}
```

指针

指针变量定义

指针变量的定义：

C语言有两种变量：其中变量（普通变量）存贮内容值；地址变量（指针变量）存贮地址值。

指针变量定义格式

类型名　　*指针变量名

例：

int　　*p1,*p2;

float　　*p3,*p4;

char　　*p5,*p6;

【注】

1、定义变量（普通变量、指针变量）都必须在前面有类型名

2、在定义指针变量时，指针变量名前面的*表示现定义的是一个指针类型的变量。*并不是指针变量名的一部分，只是一个标志

3、指针变量专门用来存储地址，禁止将一个整型值直接赋给一个指针变量

指针变量的引用

&取地址运算符，通过&运算符可以取出普通变量的地址

*指针运算符，*可以取出指针变量所指向的普通变量的值（间接引用普通两）

例：

```
int a,b=20,c = 30,d = 40,*p;
p = &d;//将d的地址赋值给p
a = *p;//将p的内容值赋值给a
a = d;//将d的值赋值给a
```

【注】

1、可以通过赋值使一个指针变量指向某一铍铜变量（指针变量 =&普通变量）

例;

int a = 10;int *p;

p = &a;

可以写作：int a = 10; int *p=&a;

错误写法：int a = 10;int *p; *p = &a;

2、C语言中正确的做法是先让指针变量指向一个确定的存储单元后，再通过该指针变量引用它所指向的存储单元。

不能写作：int *p;　　*p = 200;

3、变量名（普通变量、指针变量）都表示其存储单元内的值

p1 = p2　　//*p1指向了p2所指向的单元地址

*p1= *p2　　//*p1指向了p2所指向的单元内容值

4、若指针变量p指向变量a,即将变量a的地址赋给了指针变量p

例： int a = 20,*p;　*p = &a; 则有下列结果(<=> 等价于)

*p<=> a　//*p等价于a

p <=> &a

&(*p)<=>&a<=>p

*&a　<=>*p<=>a

(*p)++ <=>a++

++(*p)<=>++a

5、所有的指针变量在内存总分配的字节数相同。sizeof()　//都占两个字节，随着指针变量的类型不一样，占用字节数不一样

```
下列程序的运行结果为
#include<stdio.h>
void fun(int *x,int *y);
main(){
 int x = 1,y = 2;
 fun(&y,&X);
 printf("%d %d",x,y);//4 3
}
void fun(int *x,int *y){
  printf("%d %d",*x,*y);// 2 1
  *x = 3;
  *y = 4;
 }
```

```
#include<stdio.h>
void swap(int *p1,int *p2);
main(){
 int a,b;
 int *p1 = &a,*p2 =&b;
 scanf("%d%d",p1,p2); //23
 swap(p1,p2)
 printf("%d%d",*p1,*p2);//32
}
 void swap(int *p1,int *p2){
  int temp; //若按以下写法，则结果不会交换，因其仅交换了地址
  temp = *p1; //temp = p1;
  *p1 = *p2; //p1 = p2;
```

```
14   *p2 = temp; //temp = p1;
15  }
```

说明：

1、int fun(int a[10] <=> int fun(int *a) <=> int fun(int a[ ])若数组作为形参，则将数组名做指针变量来处理

与[]是等价的　&与　[]互逆

指向数组的指针变量

指向数组元素的指针变量

由于数组元素与普通一样，所以定义指向数组元素的指针变量与定义指向普通变量的指针变量完全一样

例;

int s[10],a,b;

int b[2][3];

int *p;

p = &a;//将a的地址赋给p

p = &s[2];//将数组第二个元素的地址赋给p

p = &b[1][2];//讲数组第一行第二列的元素地址赋给p

一维数组与指针变量

注：

1、在C语言中规定，数组名代表数组的首地址，而且是一个地址常量

例：

int a[10];

int *[;

p = a;　<=> p = &a[0];

2、当指针变量指向数组中的某一个元素时，指针变量加1后指向数组的下一个元素，指针变量减1时指向数组中前一个元素

例：

float a[10];　　float *p;

p = &a[4];　　则p-3指向a[1];

3、当制作真变量指向数组时，下标运算([])用域数组也可用域指针变量

int a[N],*p = a;

p + i　　a + i　　&a[i]

*(p+i)　　*(a+i)　　a[i] <=>p[i]

即有以下等价关系

*(p+i)　　*(a+i)　　a[i]　　p[i]

p++　　++p　　p+=1　　p = p+1

p--　　--p　　p-=1　　p = p -1

*p++　　*(p++)

(*)p++　　++(*p)　　++*p

(*p)--　　--(*p)　　--*p

```c
#include<stdio.h>
main(){
 intp *p,a[3],i;
 p = a;
 for (i = 0;i < 3;i++)
 scanf("%d",p++);//123
 printf("\n\n");
 for (p = &a[0];p < a+3;)
 printf("%d",*p++); //123
}
```

4、若两个指针变量指向同一个数组，则这两个指针变量可以进行大小比较

例;

char s[10];　　char *p1 = s + 3,*p2 = &s[7];

则：　p1>p2　　p1-p4 = 4(两个地址减是中间相隔元素的个数，加法无意义)

5、在形参中的数组实际上是一个指针变量，并不是真正的数组，因为该“数组名”的值是可以改变的，而真正的数组名的值是不能改变的

6、若形参是数组或指针变量，则在函数中可以通过该形参改变实参的值

多维数组与指针变量

例：int a[3][4];

a+1是跳过一行。因为二维数组名是行指针，**加1是跳过一行而不是一个元素**

【※】

1、只有列指针f才是真正指向元素，即指向某一个元素的存储单元。

2、一维数组名标鼠的是列指针：二维数组名表示的是行指针

注：若a是一个二维数组，则有

1、a+1是行指针，即指向的是一整行，若对它加1则指向下一行（整行的所有元素）

2、*(a+i)和a[i]一样，都是一**列指针**即指向的是一个元素

3、*(a+i)+j和a[i]+jyi行，都表示元素a[i][j]的地址，即与&a[i][j]等价

地址三等价：*(a+i)+j　a[i]+j　&a[i][j]

4、*(*(a+i)+j)、*(a[i]+j)、(*(a+i))[j]和a[i][j]一样，都表示元素a[i][j]　//元素四等价

例:

若有以下定义: int w[2][3]则对w数组元素非法引用的是; **DB(B越界了、D不符合规范)**

A、*(w[0]+2)　B、*(w+1)[2]　C、w[0][0]　　D、w[1]+2　　E、*(w[1]+2)

指向多维数组元素的指针变量

列指针

例I；int a[3][4];　　int *p= &a[0][3];

则：p+1 指向元素a[1][0]　　p+4指向元素a[1][3]　　p-2指向元素a[0][1]

常用于取二维数组a元素地址的方式：&a[i][j]、a[i]+i、*（a+i)+j

```c
#include<stdio.h>
//遍历a中的内容
main(){
 int a[3][3] = {1,2,3,4,5,6,7,8,9},*p;//p为列指针
 for(p=a[0];p<a[0]+9;p++)
 printf("%d",*p);
}
```

行指针

指向由m个元素组成的一维数组的指针变量

定义向由m个元素组成的一维数组的指针变量的格式:

数据类型　(*指针变量名)[m]（m为大于0的正整数）

例:

int a[5][7];

int (*p)[7];//行指针变量

p = a;

char b[10][80];

char(*r)[80;

r = b + 5;

```c
#include<stdio.h>
//遍历所有元素
main(){
 int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
 int (*p)[4];//所指向的每一行有四个元素
 p = a;//a原本指向起始行，此时令p指向初始行
 for(i = 0;i < 3;i++){
 for(j = 0;j < 4;j++)
 printf("%d",p[i][j]);
  printf("\n");
  }
}
```

```c
#include<stdio.h>
main(){
 int a[][3] = {{1,2,3},{4,5,0}},(*pa)[3];
 int i;
 pa = a;
 for(i = 0;i < 3;i++){
 if(i < 2){
 pa[1][i] = pa[1][i] - 1
 }else{
  pa[1][i] = 1;
  }
  }
  printf("%d\n",a[0][1]+a[1][1]+a[1][2]);//2+4+1=7
}
```

指向字符串的指针变量

字符串常量：C语言对字符串常量是**按首地址处理字符串常量**

""就代表地址

1、char str[] = “china";

2、char *p = "chine";

p = "china"

char str[]={"china"}

指向函数的指针变量

函数名与数组名一样，是起始地址，而且是一个地址常量

定义指向函数的指针变量的方式

类型名　　(*指针变量名）();

```c
#inclide<stdio.h>
int min(int a,int b){
 return a>b?b:a;
}
int max(int a,int b){
 rertuen a>b?a:b;
}
main(){
 int x = 6, y = 10;
  int (*p)();
  p = max;//p指向函数max
  printf("%d",max(x,y));//10
```

```
 printf("%d",p(x,y));//10
 p = min;
 printf("%d",min(x,y));//6
 printf("%d",p(x,y));//6
}
```

【注】

1、在定义指向函数的指针变量时，要注意有两个小括号必须要有，不需要定义形参

2、单独的函数名代表该函数的首地址（函数的入口地址）

3、函数的指针变量只增指向函数的入口处（函数的首地址），不能指向函数中的某条指令。（对于指向函数的指针变量加1是没有意义的）

4、给指向函数的指针变量赋值时，致谢函数名即可，不必写参数。

返回指针的函数

类型名　　函数名（形参）{}　//定义一个函数

返回指针的函数的定义方式：

类型名　　*函数名（形参列表）{

　　return 地址值;

}

```
#inclide<stdio.h>
int *fun(int *x,int *y){ //定义一个返回指针的函数fun
 if(*x < *y)
 return x;
 else
 retuen y;
}
main(){
 int a = 7,b = 8, *p, *q,*r;
 p = &a; // p指向a 7
 q = &b; // p指向b 8
 r = fun(p,q); //7
 printf("%d,%d,%d\n",*p,*q,*r);//7,8,7
}
```

指针数组和指向指针的指针变量

指针数组

1、若一个数组的所有元素均为指针类型（地址），则成为指针数组

格式

类型名　　*数组名[常量表达式];

int *s[10];

【注】

1、要注意它与定义指向m个元素组成的一维数组的指针变量之间的区别

2、当它的各个元素都是一个指针类型（地址），即它的每个元素都相当于一个指针变量

```
#inclide<stdio.h>
main(){
 char ch[3][4] = {"123","456","78"},*p[3];
 int i;
 for(i = 0;i < 3;i++){
 p[i] = chi[i];
 }
 for(i= 0;i < 3;i++){
 printf("%d",p[i]);
  }
 }
 输出结果为：12345678
```

指向指针的指针变量

用来存放指针变量地址的指针变量称为指向指针的指针变量

定义格式：

数据类型 **指针变量名；

例：

int a= 3;

int *p = &a;

int **k =&p;

则*k得到变量p（变量a的地址),**k得到变量a的值（a的数据3)

空指针

指针变量可以有空值，即指针变量不指向任何变量，不知想任何有用的存储单元

在系统中以将null定义为0，即null的值为0

int a,b,c,*p = NULL;

此时p的值为空指针，即p不知想任何有用的存储单元，尽管NULL的值为0，但我们不能认为p指向 了地址为0的存储单元

【注】

1、当一个指针变量的值为空指针时，我们不能引用它所指向的存储单元

2、若某指针（地址）的基类型为void，则有引用时应进行相应的强制类型转换

```
#inlude<stdio.h>
main(){
```

```
  int a[] = {1,2,3,4,5,6,7,8,9,10,11,12};
  int *p = a+5,*q = NULL;
  *q = *(p+5);//此语句不能正确运行
  printf("%d%d\n",*p,*q);//6
}
```

构造类型

结构体类型

构造结构体类型

struct 结构体类型名称{

成员1的定义;

成员2的定义

.....

成员n的定义;

};

示例:

```
struct student{
 int age;
 int num;
 char sex;
 int s[3];
};
```

【注】:

1、定义成员的方式与定义普通变量的方式相同

2、成员列表必须用一对花括号括起来

3、结构体名可以省略

定义结构体变量

1、先定义结构体类型名再定义结构体变量

```
#include<stdio.h>
main(){
 struct student{
 int age;
 int nu;
 char sex;
 int s[3];
 };//类型定义
```

```
9
10   struct student stu1,stu2,st[25]; //变量定义
11  }
```

2、再定义结构体类型的同时可以定义结构体变量

```
1  #include<stdio.h>
2  int main(){
3   strut student{
4   int sn;
5   int age;
6   char sex;
7   inr s[4];
8   }stu1,stu2,s1[24];
9  }
```

3、类型、变量名同时定义，类型名student省略(不推荐使用)

```
1  #include<stdio.h>
2  int main(){
3   strut {
4   int sn;
5   int age;
6   char sex;
7   inr s[4];
8   int *p;
9   }stu1,stu2,s1[24];
10  }
```

结构体变量再内存中占用字节数为各成员占用字节数总和

```
1  #include<stdio,h>
2  main(){
3   struct aa{
4   int num;//2
5   char namec[10]; //10
6   };
7   struct bb{
8   int a; //2
9   float b;//4
10   struct aa c;//12
11   };
12   struct bb a;
13
14  }
```

```
15 sizeof(a)的值为多少？18
16
```

double a;

double *b;//无论什么类型的指针都占用两个字节

double c[5];

则sizeof(a)=8;sizeof(a)=2;sizeof(a)=40;

若有定义语句char *pt[3]，则sizeof（pt）=6

若有定义语句char pt[3]，则sizeof（pt）=3

使用结构体变量

在定义结构体变量的同时可以将各成员的初始化顺序放在一对花括号中，来进行对结构体变量的初始化。若初值个数多余成员个数则出错，若初值个数少于成员个数，则多余成员自动赋0

```
#iinclude<stdio.h>
int main(){
 struct aa{
 int a;
 char b[10];
 float c;
 }a1 = {30,"china",40.5},a2 = {60,"kunming"},a3;
}
```

结构体变量不能整体引用，只能引用它的成员（同数组相似）

引用结构体变量名.成员名

例：stu1.sex

stu1.age

指向结构体数据类型的指针

指向结构体变量的指针

可以使用指针变量指向结构变量，也可能用指针变量指向结构体变量中的成员。要注意指针变量的类型必须与它所指向变量的类型相同，当指针变量指向结构体变量时，对指针变量加1则跳过整个结构体而不是跳过一个成员。

```
#include<stdio.h>
main{
 struct student{
 int num;
 char namc[20];
 char sex;
 float score;
 };
```

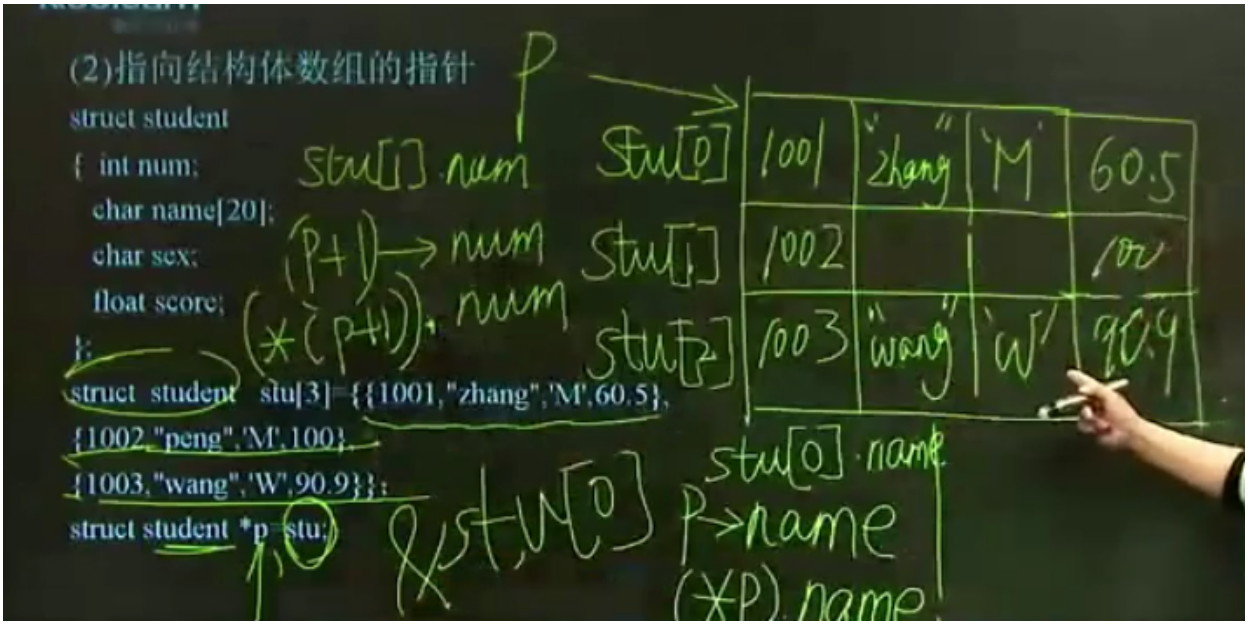
```
 struct student aa={1001,"zhangsan",'M',80.4};
 struct student *p=&aa;//将aa的地址赋给p
 char *q = aa.name;//zhangsan
 int *r = &aa.num;//r指向num的地址值
}
```

```
#include<stdio.h>
main{
 struct student aa={1001,"zhangsan",'M',80.4};
 struct student *p=&aa;//p指向aa
 char *q = aa.name;//zhangsan
 int *r = &aa.num;//r指向num的地址值
 //当指针变量p指向结构体变量aa时，引用aa中成员的方式有三种：
 aa.num
 (*p).num //p指向num
 p->num //p指向Num
 aa.score
 (*p).score //p指向score
 p->num
}
```

指向结构体数组的指针

```
#include<stdio.h>
main{

}
```



【注】

1、可以用结构体变量的成员作为实参，它与普通变量作为实参的方法是一样的
2、用结构体变量作为实参时，要求形参必须是同一结构体类型的变量，传递后形参与实参各对应成员值是一样的
3、也可以用结构体类型的地址（指针变量或数组）作为实参，要求形参必须是同一结构体类型的指针变量或数组。只要是地址传递，则可以通过形参来改变实参的值。

链式存储是顺序访问，而顺序存储结构是随机访问的
将数据集定义为数组时，所占用的是连续的空间
将数据集定义为链表时，结点必须被定义(节点不必连续），结点是由结构体构造而来

链表

链表概述:

链表是一种数据结构，它采用**动态**分配存储单元方式。它能有效地节省存储空间（同数组相比）

链表都有一个“头指针”变量（Head)，它用于指向链表中的第一个元素（即用于存放链表中第一个元素的地址）。链表中的元素称为“结点”，链表中的所有介结点都是结构体类型。且同一链表中的结点都是同一结构体类型。每个结点都应该包括数据部分和下一个结点的地址两部分内容。链表的最后一个元素（结点）称为链尾，它不再指向其他结点（即该结点的指针成员值尾Null)

链表的访问都是通过指针变量从头结点开始。

用于俩表中的结点时一个结构体类型，并且结点中有一个成员用域指向下一个结点。所以定义作为结点的格式:

struct 结构体名{
    定义数据成员;
    struct　　结构体名　*指针变量名;
};

结构体中出现指向自身的指针，则一定为结点的类型。

```
struct student{
 int num;
 float score;
 struct student *next;
};
struct student a,*p;
```

动态存储分配函数<stdlib.h>

1、malloc()函数

格式：malloc(size)

作用：在内存的动态存储区中分配一个长度为size个字节的连续空间，函数返回值为一个指向分配域起始地址的指针,若分配失败则返回Null

示例：开辟一个用于存放struct stdent 数据的内存空间，并让p指向内存空间。

struct student *p = (struct student *)malloc(sizeof(strct student)) //(struct student*)意为强制类型转换。

2、free()函数

格式：free(p)

作用：释放用malloc()分配的内存

链表操作

1、建立动态链表（假定若输入的成员为0则表示结束）

```
#include<stdlib.h>
main(){
 struct node{
 int data;
 struct node *next;
 };
 struct node *head,*p,*q;//定义结点
 p = (struct node*)malloc(sizeof(struct node));
 p -> data = 10; //
  head = p;
  q = (struct node*)malloc(sizeof (struct node));
  q -> data = 20;
  q -> next = null;
  p -> next = q;
 }
 //将两个链进行连接
```

如果是链表必须要有结点（结点的标志是有指向自己的指针）

访问链表

```
struct node{
 int data;
 struct node *next;
};
struct node *p;
int sum = 0;
p = head;
while (p != null){ //遍历链表内的内容
```

```
9   sum += p-> data;
10   p = p->next;
11  }
12  sum = sum / 3;
13
```

链表操作

链表结点的删除

例:

1、请将结点d从链表中删除(结点位于中间)

s d t

s -> next = d ->next;

s -> next = s ->next -> next;

2、请将结点d从链表中删除（结点位于头）

head = head->next;

3、请将结点a从链表中删除（结点位于尾部)

p = q->next;

q -> next = null;

增加结点

例:

1、请在链表中插入结点c（结点位于中间）

q (s) t

s - > data = 20;//所插入点的数据

s ->next = q -> next;

q -> next = s;

2、请在链表中插入结点c（结点位于头）

s -> data = 5;

s -> next = head;

head = s;

3、请在链表中插入结点c（结点位于尾部)

假设最后一个结点为q

s -> data = 15;

s -> next = null;

q -> next = s;

共用体类型

共用体类型中的所有成员共用同一段内存（所有成员的起始地址都是一样的）

格式：

union 共用体名{

成员列表;

};

【注】：

1、成员列表为定义该共用体的成员，成员定义的方式与普通变量的方式一样

2、成员列表必须用一对花括号括起来

3、共用体名可以省略

例:

```
union data{//定义了一个名为data的共用体类型，该类型有三个成员：i  ch s
    int i;
    char hc[10];
    float s;
};
```

公用体变量的定义：

1、先定义类型，在定义变量

2、定义类型的同时定义变量

3、直接定义变量

```
union data{
    int i;
    char ch[10];
    float s;
}a1;
```

注：

由于共用体类型变量的所有成员都共用同一段内存，所以共用体类型变量所占的字节数等于该共用体类型中**占用字节数最多**的成员所占的字节数；

sizeof(a1) => 10;

共用体变量的引用

注：

1、不能整体引用共用体变量，只能引用其成员

引用格式： 公用体变量名.成员名

2、同类型成员共享值

3、在内存中整形数据的二进制数低8位占用前面一个字节，高八位占用后面一个字节

如：

整数：255在内存中的存储形式位：

11111111 00000000

一个字符型数据占用一个字节，对于数组来说前面元素占用前面的字节

4、共用体变量之间可以相互赋值，赋值后两个变量应使用同一成员

5、共用体变量的地址与各成员的地址都相同

6、在定义共用体时，可以对其进行初始化，但只能有一个初值且必须用花括号将初值括起来。相当于给第一个成员赋值

7、共用体、结构体的成员均可以是共用体或结构体类型

8、不能用共用体类型变量作为函数参数

9、共用体占用字节数为共用体类型中**占用字节数最多**的成员所占的字节数；

```
struct date{
 int day;
 int month;
 int year;
 union {
 int share1;
 float share2;
 }share;
}a;
 sizeof(a)和sizeof(a.share)的值是多少
 10 4
```

```
union myun{
 struct {
 int x,y,z;
 }u;
 int k;
}a;
main(){
 a.u.x = 4;
 a.u.y = 5;
  a.u.z = 6;
  a.k = 0;
  printf("%d\n",a.u.x) ; //0
 }
```

```
#include <stdio.h>
main(){
```

```
union {
char i[2];
int k;
}r;
r.i[0] = 2;
r.i[1] = 0;
printf("%d\n",r.k);//2
}
```

typedef

用typedef定义新类型名

在编程中可以用typedef来定义新的类型名来代替已有的类型名

格式：

typedef int INTEGER;

以后再定义变量时，int 和INTEGER是等价的。

INTEFGER a[10],b;

int a[10],b;

typedef　已有类型名　　新的类型名；

1、typedef可用于定义各种类型名，但不能定义变量。即只要见到typedef则该语句最后的标识符必定是一个类型名而不是变量名

2、typedef只能对已经存在的类型新增一个别名，而不是创造新类型，即在typedef后必须是一个已有的类型

例：

typedef it ARR[10];//10为公共的10

ARR a[10],b[2][10];　=> ARRa,b[2];

typedef char *POINT;

POINT *p1,**p2;　=> POINT p1,*p2;

位运算（按位计算）

位运算的操作对象只能是整型或字符型数据

先将非二进制转换为二进制，再将二进制的结果还原为非二进制

位运算不产生进位、借位

C语言提供6位运算符：

&（按位与）　|（按位或）　^（按位异或）　~（按位取反）　<<（按位左移）　>>（按位右移）

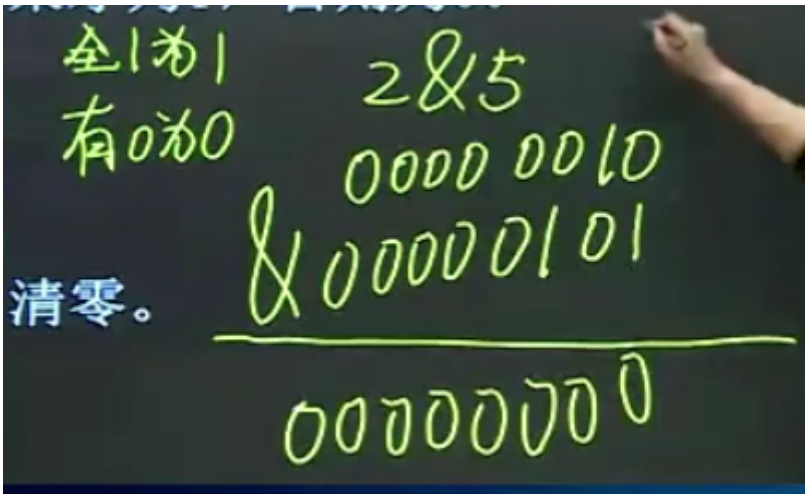
复合赋值运算符：

&=（）　|=（）　^=（）　<<=（）　>>=（）

按位与运算（&）

两个相应的二进制位都是1时，他们按位运算后的结果才为1，否则为0

全1为1，有0为0



1 & 1 = 1

1 & 0 = 0

0 & 1 = 0

0 & 0 = 0

按位或运算（|）

两个相应的二进制位中只要有一个为1，则他们按位或运算后结果为1

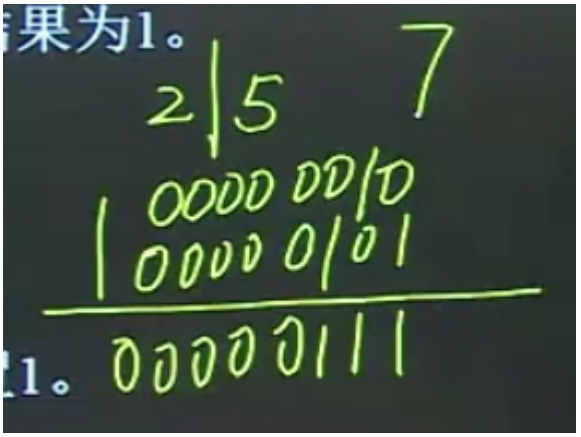
有1为1，全0为0

1 | 1 = 1

1 | 0 = 1

0 | 1 = 1

0 | 0 = 0



按位异或运算（^）

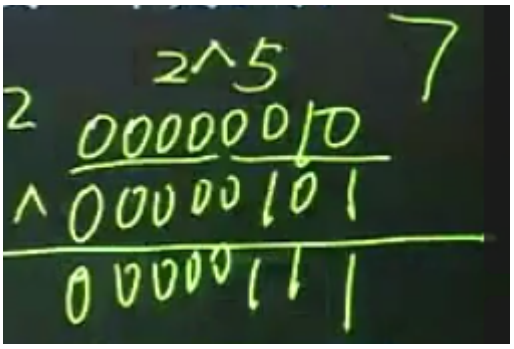
当两个相应位同为1或同为0时，按位异或运算结果为0，两个相应位一个位1另一个为0时，按位异或运算结果为1

1 ^ 1 = 0

1 ^ 0 = 1

0 ^ 1 = 1

0 ^ 0 = 0



按位取反运算（~）

按位取反运算符时一个单目运算符。按位取反或0变1，1变0

注：

对一个数按位取反得到的值为该数+1后再乘以-1

~24 -> -25

~0 -> -1

按位左移运算（<<）

格式：数<< n

功能：将二进制位按位依序左移n位

对一个十进制数左移n位后得到的值为该数乘以2^n的积

即：数*2^n

2 << 4 -> 32

按位右移运算（>>）

格式：数>>n

功能：将二进制位按位依序右移位

若该数位一个符数并且不能被2^n整除则得到的数为商加-1

即：数/2^n 数(负且不能整除）/2^n+(-1)

例：

int b = 2;表达式(b >> 2)/(b >> 1)的值是什么?

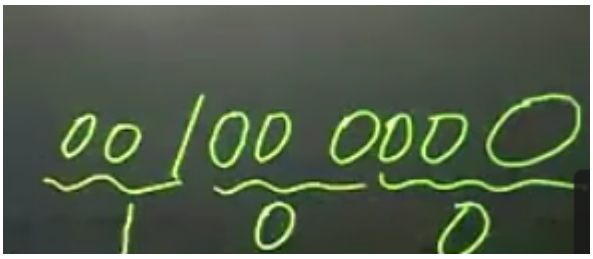
解：表达式的值为0

例:

main(){

```
    char x = 040;
    printf("%O\n",x << 1);  //%O以八进制的形式进行输出
}
```

结果为100



文件

文件概念

1、文件：记录在外部存储介质（外存）上的数据的集合

2、文件的分类

文本文件（ASCII码文件）：每个字符存储到文件中

二进制文件：以二进制存储

C语言中对文件的存取是以字符（字节）为单位

3、文件类型指针

FILE就是文件类型名，它是一个结构体类型。对一个文件进行操作，通过文件指针进行的，定义如下：

```
FILE *fp,*in,*out;
```

文件的打开与关闭

文件的打开（fopen()函数）

格式：fopen（文件名，文件使用方式）

功能：按指定的“使用方式”打开文件，函数返回所打开文件的指针，该指针的基类型为文件类型，文件名和文件使用方式均为字符串

如：以只读方式打开文件data.txt，并用指针变量fp指向它

```
FILE *fp;
fp = fopen("data.txt","r");
```

打开文件的使用方式：

"r":打开已存在的文件

"w":若文件存在则刷新写，若文件不存在则创建写

"a":若文件存在则追加写，若文件不存在则创建写

"+":增强

b:二进制文件

"r="　　"w+"　　"a+"　　"rb"　　"wb"　　"ab"　　"rb+"　　"wb+"　　"ab+"

注：

1、文件使用方式只能用小写字母，文件名用大写获小写均可

2、在文件使用方式中若含有字母b则打开的是一个二进制文件（bit)

3、当fopen打开失败时，函数返回NULL

```
if (fp=fopen(文件名，文件使用方式）==NULL){
    printf("can not open this file\n");
    exit(0);
}
```

文件的关闭（fclose函数）

文件使用完成后应该关闭该文件，保证文件数据不丢失.将内存中的数据回写到外村中

格式：fclose(文件指针)

如:fclose(fp);

fputc()

格式：fputc(字符，文件指针）

功能：把一个字符写带文件指针所指的文件中，其中字符可以是字符常量也可以是字符变量。若输出成功则函数返回输出的字符，失败则返回EOF（stdio.h中定义为-1）

fgetc()

格式：fgetc（文件指针）

功能：从文件指针所值文件中读取以一个字符。若读取成功则函数返回读取的字符，失败（遇到文件结束）则返回eof

fgets()

格式：fgets(str,n,fp)

功能：其中str表示一个字符指针，可以是字符数组名也可以是字符指针变量名。从fp所致文件中读取n-1个字符，并在这些字符的最后加一个字符串结束符 ‘\0'后赋值给str

函数返回str的首地址

fputs()

格式：fputs(str,fp)

功能：向fp所指文件中写（输出）str中的字符串，str可以是字符串常量、字符数组获字符指针变。在输出时字符串的结束符’ \0 ‘不输出。若输出成功则返回0,失败返回EOF

fread()、fwrite()

格式：

```
fread(buffer,size,count,fp);
fwrite(buffer,size,count,fp);
```

其中：
bufffer是数据的地址
size是每次读写的字节数
count表示让函数进行多少次的读写
fp是要进行读写的文件指针变量
功能：用来读写一个连续的数据块
注：
1、这两个函数按二进制方式进行读写

fprintf()、fscanf()

格式：
fprintf(文件指针，格式说明符，输出列表);
fscanf(文件指针，格式说明符，输入列表);
功能：按格式说明符所指定的格式向文件中读写（输入输出）数据，其中格式说明符和输入（输出）列表的用法与scanf和printf函数相同。

补充：feof（文件指针）

作用是测试文件的当前读写位置是否在文件末尾，若是则返回非0值（真），否则返回0(假)

文件当前读写位置函数

1、重新定位指针
格式：rewind(文件指针)
作用：使用当前的读写位置重新指向文件的开头。函数无返回值
2、fseek（）
格式：fseek(文件指针，位移量，起始点)
位移量：要在数值后加字母l或L
如：fseek(fp,100L，SEEK_SET)
功能：将当前的读写位置从”起始点“开始按”位移量“所指定的地洞字节数向后移动
起始点有：
SEEK_SET 或0 （表示”文件的开始“）
SEEK_CUR或1 （表示”当前位置“）
SEEK_END 或2（表示文件末尾）

例题：
将位置指针定位到离文件头59个字节的地方
fseek(fp,59L,SEEK_SET）；
3、ftell()
格式：ftell（文件指针）

功能：返回当前文件的位置，用相对于文件头的位移量表示。若返回-1L表示出错

例题分析 ： 求文件长度

C语言的发展及其特点

C语言是国际上广泛流川的计算机高级语言

C语言的祖先为BCPL(Basic Combined Programming Language),C语言的新特性主要表现在具有多种数据类型（如字符、数值、数组、结构体和指针），开发C语言的目的在于尽可能降低它所写的软件对硬件平台的依赖程度，使之具有可移植性。

C语言的主要特点

1、语言简洁、紧凑、使用方便、灵活

2、运算符丰富

3、数据类型丰富

4、具有结构化的空值语句

5、语法限制不太严格，程序设计自由度大。

6、C语言允许直接访问物理地址，能进行位(bit)操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作

7、C语言编写的程序可移植性好

8、生成目标代码质量高，程序执行效率高

程序示例

每一个C语言都必须有一个main函数。函数体由花括号{}括起来

在使用函数库中的输入输出函数时，编译系统要求成簇提供有关此函数的信息，#include <stdio.h>的作用就是来提供这些信息的。stdio.h是系统提供的一个文件名，是"standard input & output "的缩写，文件后缀.h的意思是头文件，因为这些文件都是放在程序各文件模块的开头。#include指令是将这些信息调入供使用。

```c
#include <stdio.h>    //编译预处理指令
int main()    //定义主函数
        {       //函数开始的标志
        printf("this is a C program\n"); //输出指定的一行信息
        return 0; //函数执行完毕时返回函数值0
}
```

求两个整数之和

```c
#include <stdio.h> //预编译指令
int main(){ //定义主函数
 int a = 3; //定义a变量，并赋值
 int b = 2; //定义b变量，并赋值
 int c = a + b; //定义c变量为a，bz值相加
 printf("the reault of a + b is :%d",c); //输出加法运算的结果
 return 0;  //使函数返回值为0
}  //结束函数
```

判断两个数的大小

```c
#include <stdio.h>
int main(){ //定义主函数
 int a,b,max; //定义变量a,b,c
 scanf("%d %d",&a,&b); //从键盘输入a,b
 if (a > b){ //判断a,b的大小
 max = a; //将a的值赋给max
 printf("最大值为: %d",max); //输出max
 }else{
 max = b;
  printf("最大值为:%d",max);
  }
  return 0;
}
```

同一个数区分进制

17	十进制
017	八进制
0X17	十六进制

C语言程序的结构特点

1、一个程序由一个或多个源程序文件组成（1、预处理指令2、全局声明3、函数定义）

2、函数是C程序的主要组成部分

3、一个函数包括两个部分（1、函数首部2、函数体（声明部分、执行部分））

4、程序总是从main函数开始执行

5、程序中对计算机的操作是由函数中的C语句完成的

6、在每个数据声明和语句的最后必须有一个分号

7、C语言本身不提供输入输出语句

8、程序应当包含注释