```
/*
*现在开始基本的C语言内容和复杂的数据结构内容相互配合的过程
*数据结构和C语言都是前期极为关键的内容
*
//sizeof是关键字，而不是函数
//常亮的两种写法
//const 常量
//#define
//不可以对一个算数表达式进行加一和减一操作处理
#include<stdio.h>
#ifndef maxSize
#define maxSize 10
#endif

  int main()
  {
  int k= 0;

        printf("%d\n",sizeof(k));

          }
          *
          #include <stdio.h>

                int main( )
                {
                char ch;
                printf("please put\n");
                ch= getchar();//getchar没有参数，只需要有一个接受的返回值即可。
                ch = ch -32;
                putchar(ch);
                putchar('\n');  //putchar需要要求有一定的输出的值
                /* code *
                return 0;
                }
```

```
#include <stdio.h>
//#define pi 3.1415926
int main( )
{
const double pi = 3.14;
printf("%5f\n",pi);
printf("%-5f\n",pi);
printf("%-05f\n",pi);
printf("%.2f\n",pi);
printf("%5.7f\n",pi);
}
/*
```

```
#include<stdio.h>

  int main()
  {
  int a,b;
  scanf("%d %d",&a,&b);//分隔符为空格，tab，回车，域宽，非法输入

  scanf("%d,%d",&a,&b);//逗号输入
  scanf("%2d,%2d",&a,&b);//输入的长度达到

  scanf("%d%*c%d",&a,&b);//输入任何一个字符均可以作为字符的分界

  printf("a = %d,b = %d\n",a,b);
  return 0;
  }
```

//两种错误的形式，编译器会报错的是编译错误，
//运行时报错的称为运行时错误

```
//单分支控制:交换两个数据的过程“戈”形结构
#include<stdio.h>

int main(){
        int a = 5;
        int b = 4;
        int temp;
        if(b>a)
        {
                temp = a;
                a = b;
                b = temp;
        }
        else{
                temp = 1>3?a:b; //此处是一个三元运算符号。当前面的条件成立的时候执行前，不成功的时候输出后半部分
                printf("%d\n",temp);
                return 0;
        }
        printf("a = %d,b = %d\n",a,b);
        return 0;
}

//exit函数的作用
//其的作用是终止整个程序的执行，强制返回操作系统
//当值为0或者EXIT_SUCCESS是正常的出!=0或者为EXIT_FAILURE表示程序出错后退出

//此题仍然需要再议
#include<stdio.h>

int main(){
        float a = 0.0000001 ;
        int b = 100000000;
        float temp;
```

```
        temp = a*b;
        if(temp == 10)
        {
        printf("OJBK\n");
        }
        else{
        printf("NOT OJBK\n");
        printf("%f\n",temp);
        }
        return 0;
}*/
```

//浮点数并非真正意义上的浮点数，其在实际上只是一个在浮点数在转换为二进制存储下的一种范围的近似处理

//所以，只能使用近似的方法将实数与0进行比较处理。

//所以在处理浮点数时不能使用dis == 0来进行比较

/*

对于switch语句的使用

switch语句的内容只能是“char” 和int类型的内容

char类型的可以是一些符号或者指定的字符的结构，而不能是其他的一些结构类型

常量和case中间至少有一个空格，常量后面的是冒号，常量的类型应该与switch后括号内表达式的类型相一致

break才会跳出，default作为默认的输入和输出结构即可。其不需要输入任何的待定结果项目

```
#include<stdio.h>
int main(){
        int b = 0;
        switch(b)
        {
        case 2:
                printf("h2\n");
                break;
        case 3:
```

```
                printf("h3\n");
                break;
        case 4:
                printf("h4\n");
                break;
        case 5:
                printf("h5\n");
                break;
        default:
                printf("。。。。。。。。\n");
        }
        return 0;
}
```

//在数学上成立的表达式在C语言中不一定是正确的
//逻辑运算的内容是:

//短路与和短路或
/*
1. 或逻辑短路

```
#include <stdio.h>
int main()
{
        int a=5,b=6,c=7,d=8,m=2,n=2;
        (m=a<b)||(n=c>d);
        printf("%d\t%d",m,n);
        return 0;
}
```

输出的结果为1,2.为什么呢，因为a<b，m=1,这个“或”逻辑就被“短路”掉了，后面的语句就没被执行，所以n还是等于原先的2

2. 与逻辑短路

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

```
{
        int a=5,b=6,c=7,d=8,m=2,n=2;
        (m=a>b)&&(n=c>d);
        printf("%d\t%d",m,n);
        return 0;
}
```

输出的结果为0,2。因为a>b为0，m=0，整个“与”逻辑判断就为“假”，所以后面的“c>d”就被短路掉了，所以n还是等于原先的2

总结：

>>>>>“或”逻辑前面为1(true)就会发生短路。短路的时候后面的内容就不再进行判断了

>>>>>“与”逻辑前面为0(true)就会发生短路
>>>>> !的高于&&高于|

```
//计算某一年是否是闰年
#include <stdio.h>
int main()
{
        int a=5;
                int year;
                printf("please put need year:");
                scanf("%d",&year);
                if((year%4 == 0)&&(year %100 != 0)||(year %400 == 0))
                {
                        printf("%d是闰年\n",year);
                }
                else
                {
                        printf("%d不是闰年\n",year);
                }
                return 0;
}
```

程序测试只能证明程序有错，而不能证明程序无错

程序测试的目的就是为了尽可能多的发现程序中的错误

测试的内容称为测试用例

黑盒测试时功能测试

白盒测试是结构测试

进行按位运算：

~非

//除二或者称二都建议以这种方法进行

>>右移　再向外或者向右移动都会损失一定的数据

<<左移

&与

^异或

|或