1、输入与输出

所有函数使用时都必须添加相应的头文件

函数调用：函数名(实际参数); 实际参数与形式参数要求个数、顺序、数据类型要一致

返回值不一定需要，若需要，则需要定义和返回值类型相同的变量接收返回值

（1）字符输入与输出

getchar()函数

头文件：#include <stdio.h>

函数原型：int getchar(void);

功能：从标准输入 stdin 获取一个字符（一个无符号字符）

参数：无

返回值：成功 返回获取的字符的ASCII值

到达文件末尾或发生读错误 返回EOF

```
int ret = getchar();//将从键盘输入的字符对应的ASCII值保存到ret中
  printf("ret= %d\n",ret);
```

putchar()函数

头文件：#include <stdio.h>

函数原型：int putchar(int val);

功能：把参数 val指定的字符（一个无符号字符）写入到标准输出 stdout 中

参数：val -- 这是要被写入的字符。该字符以其对应的 int 值进行传递。

返回值：成功 返回显示字符的ASCII值

发生错误 返回 EOF

```
putchar(100); //将100对应的字符显示到屏幕
  //int a = putchar('r'); //将'r'显示到屏幕，并且获取putchar函数的返回值
  //printf("a = %d\n",a);
```

（2）字符串输入与输出

gets()函数

头文件：#include <stdio.h>

函数原型：char *gets(char *str);

功能：从标准输入 stdin 读取一行，并把它存储在 str 所指向的字符串中。当读取到换行符时，或者到达文件末尾时，它会停止，具体视情况而定。

参数：str -- 这是指向一个字符数组的指针，该数组存储了 C 字符串。

返回值：成功 返回 str

发生错误或者到达文件末尾时还未读取任何字符 返回 NULL。

注：获取的字符串中字符个数要小于保存字符串的空间大小

```
char tmp[12] = "";
  printf("---tmp= %s\n",tmp);
  gets(tmp);//将从键盘输入的一串数据存放到tmp中
  printf("tmp= %s\n",tmp);
```

puts()函数

头文件：#include <stdio.h>

函数原型：int puts(const char *str);

功能：把一个字符串写入到标准输出 stdout，直到空字符，但不包括空字符。换行符会被追加到输出中。

参数：str -- 这是要被写入的 C 字符串。

返回值：成功　　返回一个非负值为字符串长度（包括末尾的 \0）

　　发生错误　　返回 EOF。

```
char tmp[64] = "hello world!";
  puts(tmp);//将tmp中保存的字符串内容显示到屏幕
```

（3）格式化输入与输出

①scanf()函数

头文件：#include <stdio.h>

函数原型：int scanf(const char *format, ...);

功能：从标准输入 stdin 读取格式化输入

参数：format -- 这是 C 字符串，包含了以下各项中的一个或多个：空格字符、非空格字符和 format 说明符。

类型	合格的输入	参数的类型
%c	单个字符：读取下一个字符。	char
%d	十进制整数：数字前面的 + 或 - 号是可选的。	int
%hd	短整型	short
%ld	长整型	long
%f、%F	单精度浮点数：包含了一个小数点	float
%lf	双精度浮点型	double
%e、%E	浮点数：科学计数法	float
%o	八进制整数。	int
%s	字符串。这将读取连续字符，直到遇到一个空格字符（空格字符可以是空白、换行和制表符）。	char *
%u	无符号的十进制整数。	unsigned int
%x、%X	十六进制整数。	int
%p	读入一个指针 。	

....：表示变量地址表

返回值：成功　　返回成功匹配和赋值的个数

到达文件末尾或发生读错误　　返回 EOF

注：①变量地址表中的地址数量、顺序、数据类型与format中的格式控制符要求一一对应

② 变量地址表中的多个地址之间使用 “ , ” 分隔

③ 按照什么格式从终端输入数据，需要和format中一致

④ 若字符串中有空格，可以使用gets()函数或scanf(“%[^\n]” ,buff);获取字符串

⑤ 当出现垃圾字符(一般是\n)，可以使用getchar()吸收垃圾字符保证正常输入

```
    char tmp[32] = "";
    //scanf("%s",tmp);//字符串输入时，直接使用数组名
    //gets(tmp);//获取带空格的字符串
    scanf("%[^\n]",tmp);
    printf("+++tmp =%s\n",tmp);
  int a,b;char c;
  scanf("%d",&a);
    scanf("%d",&b);
    getchar();
    scanf("%c",&c);
    printf("a = %d,b= %d,c = %c\n",a,b,c);
```

②printf()函数

头文件：#include <stdio.h>

函数原型：int printf(const char *format, ...);

功能：向标准输出 stdout 按照格式化输出

参数：format -- 这是 C 字符串，包含了以下各项中的一个或多个：空格字符、非空格字符和 format 说明符(与scanf说明符相同)

....：表示待输出的变量表

返回值：成功　　返回成功匹配和赋值的个数

注：①变量表中的变量数量、顺序、数据类型与format中的格式控制符要求一一对应

② 变量表中的多个变量之间使用 “ , ” 分隔

③ 输出格式由format决定

④ 特殊格式：

%md：m表示显示数据的宽度，若数据宽度大于m，则正常输出；若数据宽度小于m，m为正，则显示数据右对齐；否则左对齐

%.nf：n表示小数点后输出的位数，默认四舍五入

%#o,%#x：带标记的八进制数和十六进制数

%e：科学计数法显示数据

2、类型转换

将已有的数据类型按照需求转化为其他数据类型

例：使用公式计算温度：C = 5 / 9（F - 32），若F = 100时，求C的值

```
float F;
  floatC;
  scanf("%f",&F);
C=(float)5/9*(F-32);
  printf("C=%f\n",C);
```

（1）强制类型转换

也被称为显式类型转换，即转换过程可见

格式：

(期望被转换的数据类型) 被转换的对象

注：①期望被转换的数据类型 可以是基本数据类型，也可以是构造数据类型

② 被转换的对象可以是常量、变量、表达式等

③ 类型转换只在本行生效

（2）隐式类型转换

即转换过程不可见

① 符号转换

有符号数和无符号数一起运算会将有符号数转化为无符号数

笔试原题：

```
int main()
{
    int a = -20;//-20 + 2^32
    unsigned int b = 6;
    if(a + b > b)
    {
        puts("a + b >b!");
    }
```

```
        else
        {
                puts("a + b <=b!");
        }
        return 0;
}
```

请问输出什么？为什么？

输出puts("a + b > b!"); 无符号数与有符号数相加，会将有符号的数转换为无符号数；a=-20，转换为2^32-20;在于b相加计算。

② 精度转换

高精度转化为低精度过程中会出现精度丢失，低精度转化为高精度的过程没有影响

```
//高精度->低精度
  floata = 123.87654;
  intb = a;
  printf("b= %d\n",b);//123直接去掉小数点后
  //低精度->高精度
  intv = 12345;
  floatc = v;
  printf("c= %f\n",c);// 12345.000000
```

③ 字节转换

高字节转化为低字节过程中会出现数据错误，低字节转化为高字节的过程没有影响

```
//高字节->低字节
  inta = 255;
  charb = a;
  printf("b= %d\n",b);      //a-256=-1 或者%256
  //低字节->高字节
  charc = 23;
  intd = c;
  printf("d= %d\n",d);     //23 不影响
```

注：在隐式类型转换中会出现向高字节、高精度转化的趋势

3、运算符