北京航空航天大學
BEIHANG UNIVERSITY

程序设计基础

Fundamentals of Programming

北京航空航天大学 程序设计课程组

软件学院 谭火彬

2022年

北京航空航天大學
BEIHANG UNIVERSITY

第二讲 编程基础框架

Framework

- 变量定义、简单数组与字面量
- 表达式：算术、关系与逻辑表达式
- 标准库函数与输入输出函数

再看a/b

```c
#include <stdio.h>
int main()
{
    int a, b, div;

    scanf("%d%d", &a, &b);
    if( 0 == b ) {
        printf("ERROR. The divisor is ZERO.");
        return 0;
    }
    div = a / b;
    printf("%d/%d = %d\n", a, b, div);
    return 0;
}
```

2.1 变量 & 2.2 数组

- 名称：标识符
- 值：在程序中可随时获取和改变取值
- 类型：需提前声明特定的数据类型
- 关键字：表示特定功能的标识符
- 数组：多个同类型变量

2.3 字面量（常量）

- 一次使用
- 有不同数据类型

2.4 & 2.5 逻辑表达式

- 各类算术、逻辑和关系运算
- 不同类型有不同运算逻辑、不同优先级

2.6 标准库函数&2.7 输入输出函数

- 一组预先提供的功能
- 标准输入输出函数

提纲：数据表示、存储和基本运算

- 2.1 **变量**：标识符、类型和初始化
- 2.2 一维数组简介
- 2.3 字面量（常量）
- 2.4 **算术表达式**
- 2.5 **逻辑表达式：逻辑与关系运算符**
- 2.6 标准库函数基础
- 2.7 **标准输入输出函数**：scanf和printf

2.1 变量

```
int a, b, div;
```

- ◆变量定义：声明a, b, div三个整数类型的变量
- ◆变量类型：int
- ◆变量名称：a, b, div
  - ✓多个变量用逗号分隔，名称必须是合法的标识符
- ◆标识符
  - ✓标识符是由字母（或下划线_）开头，由字母、数字和下划线组成的字符串
  - ✓系统保留的关键字不能作为标识符，如：main, int, return 等
  - ✓标识符在可作为变量名、常量名、函数名、参数名、类型名、枚举名和标号等

下面哪些是非法的标识符?

Beijing	beijing	C_Programming		
a8673	~~8a673~~	_1024		
~~sin(25)~~	num	x6	~~b.3~~	~~sum*co~~
~~#stdudent~~	~~a[1]~~			
xMin	topLeft	numOfStdudent		
x_min	top_left	num_of_stdudent		

- C语言区分大小写，因此变量名 Beijing 和 beijing 不一样。
- 为了增加程序的可读性，变量名通常由表示特定含义的英语单词组成，几个单词（或其缩写）由下划线或单词首字母大写连接在一起

（变量）命名法

- 遵循一致的命名约定会对提高代码的可用性起到重大作用
- 使得许多开发人员使用同一框架成为可能
- 统一的命名规范，帮助你在阅读代码的时候快速跟踪代码逻辑

2018年9月：据外媒报道，本周三上午，来自美国的一名程序员因同事不写注释，不遵循驼峰命名，括号换行，最主要还天天git push -f 等因素枪击了4名同事，导致一人情况危急。

代码千万条，规则第一条。
命名不规范，生命有不测。

（变量）命名法

- 常用的变量命法：利用体现其含义的英文单词命名变量，
  - ✓匈牙利法、骆驼法、下划线法、帕斯卡（pascal）法
- 骆驼法（驼峰法）
  - ✓首个单词小写，后面每个单词首字母大写
  - ✓利用大写字母区分多个单词
- 下划线法
  - ✓利用下划线区分多个单词
  - ✓Linux下传统用法

```
double stuScore;

int printEmployeePaychecks();
```

```
double stu_Score;

int print_employee_paychecks();
```

编程提示：变量命名时：

- 风格同一性
- 最小化长度 && 最大化信息量原则
- 避免过于相似
- 避免在不同级别的作用域中重名

补充：C语言保留的关键字

关键字(*keyword*)：已经被系统使用的标识符，具有预先定义好的特殊意义，不能作为变量名、函数名及标号等使用

int	struct	auto	goto	if
float	union	static	return	else
char	void	extern	break	while
short	enum	register	continue	for
long	typedef			do
unsigned	sizeof			switch
double	const			case
signed	volatile			default

define, undef, include, ifdef, ifndef, endif, 及line，虽不是关键字，但是最好把它们看作关键字，它们主要用于C预处理程序中。

变量类型：4种基本数据类型

数据类型		长度（二进制位数）（32/64位操作系统）	长度（二进制位数）（16位操作系统）	格式控制符
int	整型	32	16	%d
float	单精度浮点型	32	32	%f
double	双精度浮点型	64	64	%lf（输出仍采用%f）
char	字符型	8	8	%c

- ◆数据长度是目前常用编译器的长度，C语言并没有严格规定，不同平台、不同编译器可能有所不同
- ◆数据是有范围的
  - ✓32位int，其数值范围为-2,147,483,648 ~ 2,147,483,647（-2^{32-1} ~ $2^{32-1}-1$）
  - ✓浮点数使用IEEE-754标准， float有效数字大约十进制的7位，表示范围大约为 -$3.4*10^{38}$ ~ $3.4*10^{38}$，表示的绝对值最小数约为$10^{-44.85}$，double范围和精度更大
  - ✓浮点数表示是近似值，如显示1.0，计算机中实际可能是0.99999999...，也可能是1.0000001...

类型修饰符

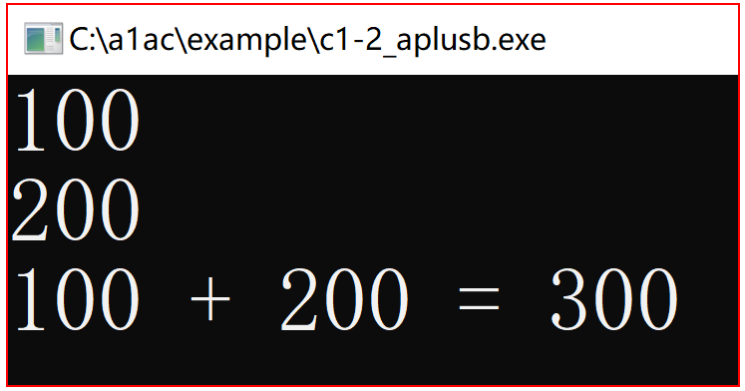
类型修饰符		长度（32/64位）	长度（16位）	格式控制符
short (int)	短整型	16	16	%hd
long (int)	长整型	32	32	%ld
unsigned (int)	无符号整型	32	16	%u
unsigned short (int)	无符号短整型	16	16	%hu
unsigned long (int)	无符号长整型	32	32	%lu
unsigned char	无符号字符型	8	8	%hhu

- ◆对于int数据类型，可以在前面加上长度修饰符
  - ✓short 短整型、long长整型（声明long时，int可以省略，直接用long）
- ◆对于int和char类型，可以加上无符号修饰符（默认有符号），即只表示是0和正数部分，不表示负数部分
  - ✓unsigned int （$0\sim2^{32}-1$）、unsinged char

再看C01-03的加法计算器

```c
/*
    This is a program for a+b;
*/
//Copyright 2022: thbin

#include <stdio.h>

int main()
{
    int a, b, sum;

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

    sum = a + b;

    printf("%d + %d = %d\n", a, b, sum);

    return 0;
}
```

运行程序



再运行，分别输入十亿和二十亿
1000000000
2000000000

看看结果怎么样。为什么？

编程提示 时刻注意变量的数据范围是否满足题目或实际需求

思考：哪种数据类型可以解决，为什么？

C99中引入更长的整数和双精度数类型

- long long (int)
  - ✓64位（8个字节）
  - ✓取值范围：-2^{63}到$2^{63}-1$
  - ✓格式控制符：%lld
- unsigned long long (int)
  - ✓64位（8个字节）
  - ✓取值范围：0 到$2^{64}-1$
  - ✓格式控制符：%llu
- long double
  - ✓96位（12个字节）
  - ✓取值范围：-/+1.19 * 10^{4932}（精确到18位小数）
  - ✓对应的格式串：%llf, %Lf

编程提示

OJ上涉及到数据范围特别大的时候，请考虑使用这两种超长的数据类型（需要支持C99版本的编译器）

变量的赋值：C02-01

```c
#include <stdio.h>
int main()
{
    int a, b; // 定义变量
    double d;

    a = 55;
    b = a; // b = a = 55;
    d = 123.456;

    printf("a = %d, b = %d\n", a, b);
    printf("d = %f\n", d);

    return 0;
}
```

- ◆赋值运算符：=
  - ✓把=右边的值，赋值给左边的变量
  - ✓该表达式返回赋值的结果
- ◆支持连运算，从右向左结合
  - ✓b = a = 55;

输出?

```
a = 55, b = 55
d = 123.456000
```

赋值过程中的类型转换：如果=左右的类型不同?

```
#include <stdio.h>
int main(){
    int a, b; // 定义变量
    double d;
    b = a = 55;
    d = 123.456;

    a = d;
    d = a;

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

赋值时左右两边类型不一致?

◆赋值类型不一致

- ✓编译器会自动将表达式右边值转换为左边类型
- ✓如果无法转换，则编译出错
- ✓转换过程中可能出现数据精度损失

输出?

a = 123, b = 55
d = 123.000000

.456　去哪儿了?

变量的初始化：C02-02

```c
#include <stdio.h>
int main( )
{
    int a, b;
    double d;

    short s = 0;
    int a1 = 55, b1 = a1;
    int day_seconds = 24*60*60;

    const double pi = 3.141592653589;

    printf("a = %d\na1 = %d\n", a, a1);
    return 0;
}
```

- ◆变量必须在赋值后才可以使用
  - ✓局部变量定义时没有赋值，其取值是不确定的
- ◆可以在定义时初始化变量的值

输出?

a = 4200656
a1 = 55

此处a的值是确定的，不同编译器、不同机器、不同时间结果可能不同
P.S. dev-c下可能为0

类型限定符const

```
#include <stdio.h>
int main( )
{
    int a, b;
    double d;

    short s = 0;
    int a1 = 55, b1 = a1;
    int day_seconds = 24*60*60;
    const double pi = 3.141592653589;

    pi = 0.618; // 程序中此处能否增加这样一行?

    printf("a = %d\na1 = %d\n", a, a1);
    return 0;
}
```

- ◆const可以变量类型签名，说明变量是只读的（不能修改的）
  - ✓即只能初始化时赋值
  - ✓后续不能进行任何赋值操作
- ◆const变量(常量变量)
  - ✓仍然是变量，有变量的属性

2.2 一维数组简介：多个同类型变量

◆C02-03：成绩统计

✓输入多人（少于200）成绩（成绩为0~100的整数，输入-1结束）

✓输出：平均分（保留小数点后两位），以及所有成绩大于平均分的输入序号和分数

◆问题分析

✓与C01-08相比，除了输出平均分，还需要把之前输入成绩中大于平均分的成绩的序号和分数输出？

✓如何保存最多200个输入的成绩和序号？

➢定义200个整数变量？

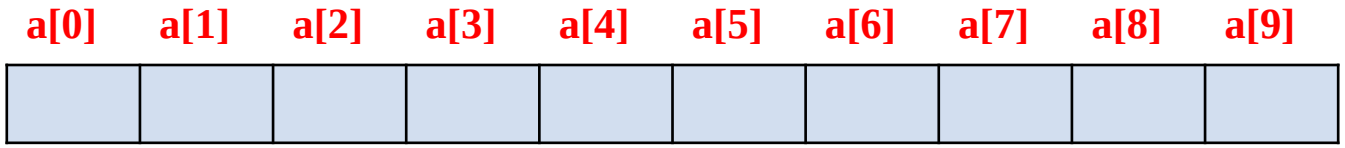
➢采用一维整数数组存储！！

一维数组简介

◆数组：定义同时存储多个同类型变量

```
int a[10]; //定义最多能够存储10个整数的数组变量a
```

✓a是整数类型的数组变量，最多存储10个整数（10为数组的**长度**）



✓数组中的每个元素通过下标访问，分别为a[0]、a[1]、…、a[9]

✓第一个元素的下标为**0**

一维数组的初始化和访问

◆数组作为一个整体，定义时，可以使用大括号初始化

```
int a[10] = {1, 3, 5, -2, -4, 6, 5, 2, 3, 1};  // 定义数组a并初始化
int b[10] = {1, 3};    // 定义数组b，并部分初始化（剩余变量为0）
int c[10];             // 定义数组e，未做初始化，初始不确定
```

◆使用数组时，每次只能访问单个元素（如：a[下标]），单个元素的访问规则与普通类型完全一样

- ✓a[0]、a[1]就是普通的整数变量
- ✓注意：**下标从0开始**，对于长度为10的数组最多只能访问到a[9]

```
int i=0, a[10];

while ( i < 10 ){
    a[i] = (i+1)*(i+1);
    i++;
}
i = 0;
while ( i < 10 ){
    printf("%d\n", a[i]);  i++;
}
```

回到C02-03：打印平均分和高出平均分的成绩

◆C02-03：成绩统计

✓输入多人（少于200）成绩（成绩为0~100的整数，输入-1结束）

✓输出：平均分（保留小数点后两位），以及所有成绩大于平均分的序号和分数

◆要点分析

✓由于要输出大于平均分的序号和分数，因此必须在输入时按照顺序保存原始成绩，可以用数组一个个保存输入的分数，数组的下标就是输入的顺序，数组元素的值就是成绩

score[0]	score[1]	score[2]	score[3]	score[4]	score[5]	…
76	82	61	92	50	-1	…

C02-03：打印平均分和高出平均分的成绩

```c
#include <stdio.h>
int main(){
    int score[201], sum=0;//成绩数组和总分数，数组的长度为201(200即可，但可适当长一些）
    int n = 0, i;         //学生数量n（初始化为0）和临时变量i
    double average;       //平均分，double类型
    scanf("%d", &score[n]);//第一个成绩输入到数据的第一个元素，n的初值为0
    while(score[n] != -1){  //判断成绩是否-1
        sum += score[n];    //累加成绩
        n++;                //人数+1
        scanf("%d", &score[n]);//输入下一个成绩
    }
    if (n>0){          //输入的有效成绩不为0
        average = (double)sum/n; //计算平均分，注意理解如何求double类型的平均分（后面讲解）
        printf("%d/%d = %.2f\n", sum, n, average);//打印平均分，double打印格式（后面讲解）
        i = 0;          //初始化i为0，从第0个开始依次访问数组中的每个元素
        while(i<n){  //依次访问成绩数组，打印出高于平均分的序号和成绩
            if(score[i]>average)//如果分数高于平均分，打印输出
                printf("%d : %d\n", i+1, score[i]);//注意输出i+1，数组下标从0开始
            i++;
        }    }
    return 0;
}
```

2.3 字面量（常量）

◆字面量（常量）：直接写在代码中的数据

✓整数（integer）类型

➢直接写在代码中的整数（1）

✓实数（小数, double）类型

➢直接写在代码里面的小数（3.14）

✓字符（character）类型

➢用单引号（'A'）括起来的一个字符

✓字符串（string）类型

➢用双引号（"%d\n"）括起来的一串字符

```c
#include <stdio.h>
int main(){
    int  a = 1;
    double pi = 3.14;
    char  ch = 'A';
    char s[] = "Hello";
    printf("%d\n", a);
    printf("%f\n", pi);
    printf("%c\n", ch);
    printf("%s\n", s);
    return 0;
}
```

整型字面量

- 默认写在代码中的整数数字为十进制的int类型
- 可以用不同的后缀表示不同的整数类型

```
int  a = 1; //基本int类型
unsigned int ua = 1u;//后缀u，表示无符号整数
long long int la = 1ll; //后缀ll，表示long long
unsigned long long int la = 1ull;   //后缀ull，表示无符号long long
```

- 除了十进制，还可以表示八进制和十六进制整数
  - ✓有关不同进制的转换，第三讲详细讲解

```
int ha = 0xFF11; //以0x开头，表示16进制整数
int oa = 012;    //以0开头，表示8进制整数
```

实数（小数、浮点数）字面量

- 默认写在代码中的小数（带小数点的）为double类型
- 两种表示方式
  - ✓一般小数表示法，合法的double示例（必须包含小数点）
    - -0.016　1233.6　0.28　.28　8000.
  - ✓科学计数法（小数部分+指数部分，中间用E（或e）来分隔）
    - -1.6E-2　　// 表示-1.6*10-2
    - 2.8E-1　　// 表示0.28
    - 1.2345e3　// 表示1234.5
    - 8e3　　// 表示小数8000.0
- 可以通过后缀f表示float类型
  - float f = 0.16f

```
int main()
{
    int   a = 1;
    double PI = 3.14;
    double eps = 1e-8;
    float pi = 3.14f;

    char  ch = 'A';
    ……
```

字符字面量

- 用一对单引号括起来的单个字符， 如：'A', 'b', '?', …
  - ✓字符是最基础的输入符号，键盘上各种键就是基本的字符
- 字符与整数存在映射关系，每个字符对应一个整数值
  - ✓实际存储时，以整数值存储，访问时转换为对应的字符量
- 编码规则：将字符映射到整数

```
int main()
{
    int  a = 1;
    double PI = 3.14;
    double eps = 1e-8;
    float pi = 3.14f;

    char  ch = 'A';
    ……
```

ASCII编码表（英文字符到整数的映射）

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL	32	(space)	64	@	96	、
1		33	!	65	A	97	a
2		34	”	66	B	98	b
3		35	#	67	C	99	c
4		36	$	68	D	100	d
5		37	%	69	E	101	e
6		38	&	70	F	102	f
7		39	,	71	G	103	g
8		40	(	72	H	104	h
9		41	)	73	I	105	i
10	\n	42	*	74	J	106	j
11		43	+	75	K	107	k
12		44	,	76	L	108	l
13	\r	45	-	77	M	109	m
14		46	.	78	N	110	n
15		47	/	79	O	111	o
16		48	0	80	P	112	p
17		49	1	81	Q	113	q
18		50	2	82	R	114	r
19		51	3	83	X	115	s
20		52	4	84	T	116	t
21		53	5	85	U	117	u
22		54	6	86	V	118	v
23		55	7	87	W	119	w
24		56	8	88	X	120	x
25		57	9	89	Y	121	y
26		58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28		60	<	92	/	124	|
29		61	=	93	]	125	}
30		62	>	94	^	126	~
31		63	?	95	_	127	DEL

ASCII字符和整数的关系

- ASCII字符对应到0~127这128个整数
- 表示数字的字符与数字值是不同
  - ✓字符0（'0'）的ASCII编码48（十六进制：0x30）
  - ✓数字0，就是0（两者差值为48）
- 字符可以直接参与整数运算，按照规则转换为对应的整数
- 数字字符 0~9（即'0' ~ '9'），字母a~z，A~Z之间的所有字符都是连续编码
  - ✓'5'编码等于 '0'+5，即 0x35
  - ✓'a'+2 等于 'c'
  - ✓'8' - '0' 等于 8

```
int ch = 'a';
printf("%c\n", ch); //输出字符 a
ch = 97;            //97是'a'的ASCII码值
printf("%c\n", ch);
ch = '\x61';
printf("%c\n", ch);
ch = '\141';
printf("%c\n", ch);

printf("%d\n", ch); // 输出什么
```

转义字符

◆部分特殊的字符，无法从键盘有效输入或被用作其他用途，需要进行特殊的处理

转义字符及其含义

转义字符	含义	转义字符	含义
\n	换行	\t	水平制表
\v	垂直制表	\b	退格
\r	回车	\f	换页
\a	响铃	\\	反斜线
\'	单引号	\"	双引号
\ddd	3位8进制数代表的字符	\xhh	2位16进制数代表的字符

C02-04：字符图案输出

```
#include <stdio.h>
int main( )
{
    printf("    * * * * *\n"
           "  *         *\n"
           " *   OO  \117\117  *\n"
           "*             *\n"
           "*             *\n"
           "*             *\n"
           " *    ___      *\n"
           "  *  (___\x29 >>>>>>>>>>\"Hi All\"\n"
           "   *         *\n"
           "    * * * * *\n");
    return 0;
}
```

```
    * * * * *
  *         *
 *   OO  OO  *
*             *
*             *
*             *
 *   (___) >>>>>>>>>>"Hi All"
  *         *
    * * * * *
```

- '\117'代表字符'O'，117是八进制数，不能写前导0
- '\x29'代表字符')'，x29是十六进制数，不能写前导0且x为小写
- 转义字符中只能使用小写字母
- 每个转义字符只能看作一个字符
- 表示单引号、反斜杠和双引号时，须使用转义字符表示

字符串字面量

- 字符串字面量：用一对双引号括起来的一串字符："hello"
  - ✓双引号中可以包括若干个字符、转义字符等
  - ✓所有字符串字面量均以不可见的‘\0’ 结束（ASCII为 0 的字符，表示字符串的结束，该字符存在但不可见），末尾 '\0' 由编译器自动添加
    - ➢"x" 和 'x' 不同， "x"其实包括两个字符'x'和'\0' 。
  - ✓字符串变量类型将在第6讲介绍

```
printf("Welcome \n  to \nBeijing!\n");
printf("\"This is \x61 string\"\n");
```

输出

```
Welcome
  to
Beijing!
```

输出

```
"This is a string"
```

2.4 算术表达式

- 算术表达式：由算术运算符、括号和操作数连接的表达式
  - ✓基础算术运算符：+ - * / %
  - ✓操作数：数字类型的变量或字面量
- 使用5个基础算术运算符
  - ✓均为二元运算符，即每个算术运算符取两个操作数进行运算
    - ➢如：a * (b+c)
  - ✓分数形式的除法
    - ➢如：$\frac{a+b+c}{3}$，对应合法的C语言表示式为(a+b+c)/3
  - ✓运算符优先级：z = p * r % q + w / x – y;
  - ✓括号的使用（“多余”的括号有时并不多余！）
    - ➢如：d = (a * (b + c)) + (c * (d + e));

基础算术运算符

操作	算术运算符	数学表达式	C表达式
加	+	b + 8	b + 8
减	-	f - h	f - h
乘	*	ab	a * b
除	/	a/b 或 a÷b	a / b
求模	%	r mod s	r % s

C02-05：求某天是星期几

◆已知某月day日是星期weekday，该月有n天，求下一个月的第k_th日是星期几newday?

◆问题分析

✓用0~6表示周日至周六，则下一个月的第k_th日是星期几newday，可表示为

✓newday = (n - day + k_th + weekday) % 7

```c
#include <stdio.h>
int main(){
    int n, day, weekday, k_th, newday;
    //假设选择2020年2月20日，  星期四
    n = 29;     //2020年2月有29天
    day =20;    //20日
    weekday = 4;//星期四

    printf("day of next month: ");
    scanf("%d", &k_th);

    newday = (n - day +k_th +weekday) % 7;

    printf("the newday is: %d\n", newday);
    return 0;
}
```

C02-06：一元二次方程求根

- 输入a、b、c三个实数，求一元二次方程$ax^2+bx+c=0$的实根，按照从小到大的顺序输出；如果没有实根，则输出字符串NULL
- 程序分析
  - ✓求根公式：$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
  - ✓如何开根号？
    - ➢**math.h**头文件中，涉及各类数学函数，其中sqrt求平方根
  - ✓将该公式写成C语言表达式
    - ➢`r1 = (-b + sqrt(b * b - 4 * a * c)) / (2 * a);`
    - ➢`r2 = (-b - sqrt(b * b - 4 * a * c)) / (2 * a);`

C02-06：一元二次方程求根

```c
#include <stdio.h>
#include <math.h>
int main()
{
    double a, b, c, delta, r1, r2;
    scanf("%lf%lf%lf", &a, &b, &c);

    delta = b * b - 4 * a * c;  //求delta = b^2 - 4ac
    if (delta < 0)//如果delta小于0，则没有解
        printf("NULL");
    else {
        r1 = (-b + sqrt(delta)) / (2 * a);
        r2 = (-b - sqrt(delta)) / (2 * a);
        if (r1 <= r2)   //判断两个根的大小，决定输出顺序
            printf("%8.2f,%8.2f", r1, r2);//输出8位，小数点后2位
        else
            printf("%8.2f,%8.2f", r2, r1);//输出8位，小数点后2位
    }
    return 0;
}
```

算术运算符：自增和自减运算符

◆自增和自减运算符

- ✓自增++（变量递增1）运算，用于取代赋值运算，如：a = a + 1
- ✓自减--（变量递减1）运算
- ✓自增++和自减--运算符可放在变量的前面也可放在变量后面

运算符	名称	示例表达式	说明
++	前置自增	++a	将a加1，然后在a出现的表达式中使用新值
++	后置自增	a++	在a出现的表达式中使用当前值，然后将a加1
--	前置自减	--a	将a减1，然后在a出现的表达式中使用新值
--	后置自减	a--	在a出现的表达式中使用当前值，然后将a减1

自增和自减运算符

- 前置和后置自增/减运算符
  - ✓相同点：变量加/减1
  - ✓不同点
    - ➢前置：先加/减后用
    - ➢后置：先用后加/减

在单独一条语句中，前置自增（减）与后置自增（减）是相同的；

a++; ⇔ ++a; ⇔ a = a + 1;

但在表达式中含义不同：

c = a++;	c = ++a;
⇔	⇔
c = a;	a = a + 1;
a = a + 1;	c = a;

```
int i = 1, j = 2;
int m = 5 , n = 6;
int u, v, x, y;
u = i++;
v = ++j;
x = m--;
y = --n;
```

思考：u、v、x、y值?

自增和自减运算通常在循环语句中**控制条件变量的改变**

```
while(i <= n){
    ……
    i++;
}
for(i=0;i<n;i++){
    ……
}
```

复合赋值运算符

- 对一个变量的值在其原有值基础上修改时，赋值运算符可以缩写成赋值表达式
  - ✓a += 5 等价于 a = a + 5，　x *= y + 5 等价于 x = x * (y + 5)
- 其它二元运算符均可写成类似的形式，即
  - ✓(v) = (v) op (expression) 可写为 (v) op= (expression)
- 建议使用复合赋值运算符
  - ✓程序更清晰简洁
  - ✓提高编译执行效率
- 注意细节，如：y *= n+1;
  - ✓等价于 y = y*(n+1); 而不是y = y*n+1;

赋值运算符	示例表达式	等价含义
+=	c += 7	c = c + 7
-=	d -= 4	d = d - 4
*=	e *= 5	e = e * 5
/=	f /=3	f = f / 3
%=	g %=9	g = g % 9

小结：算术表达式与运算符

基础算术运算符

操作	算术运算符	数学表达式	C表达式
加	+	b + 8	b + 8
减	-	f - h	f - h
乘	*	ab	a * b
除	/	a/b 或 a÷b	a / b
求模	%	r mod s	r % s

复合赋值运算符

赋值运算符	示例表达式	等价含义
+=	c += 7	c = c + 7
-=	d -= 4	d = d - 4
*=	e *= 5	e = e * 5
/=	f /=3	f = f / 3
%=	g %=9	g = g % 9

自增、自减运算符

运算符	名称	示例表达式	说明
++	前置自增	++a	将a加1，然后在a出现的表达式中使用新值
++	后置自增	a++	在a出现的表达式中使用当前值，然后将a加1
--	前置自减	--a	将a减1，然后在a出现的表达式中使用新值
--	后置自减	a--	在a出现的表达式中使用当前值，然后将a减1

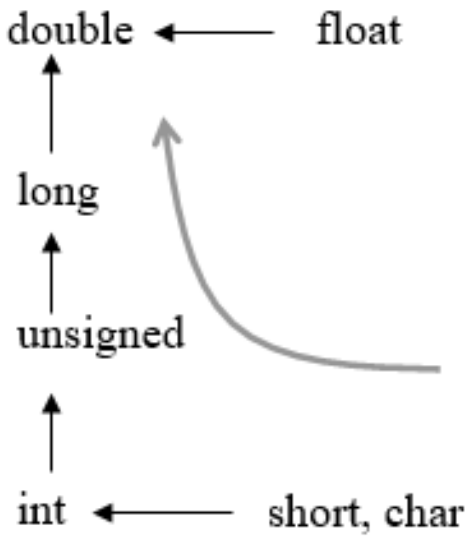
运算中的数据类型和类型转换

- C语言为强类型语言，只有同类型数据才可以一起运算
- 不同类型数据混合运算时，需要转换为同类型（类型转换）
  - ✓隐式类型转换：C语言根据运算类型的关系，将参与运算的不同类型进行**自动转换**为统一类型
  - ✓强制类型转换：程序员可以根据需要，强制将一个类型转换为另一类型

隐式类型转换

◆隐式类型转换规则：在混合类型表达式中的每个值的类型会升级为此表达式中的“最高”类型

- ✓字符与整数：字符即8位整数，可以用整数的地方就可以用字符；整数转换成字符时，超出8位就将高位丢掉
- ✓浮点数与整数：浮点数变为整数时直接去掉小数点后面的部分
- ✓整数与无符号整数：普通整数int 和无符号整数 unsigned 混合使用，则普通整数转换成无符号整数
- ✓算术转换：如果一个运算符，有不同类型的运算对象，那么“较低”类型会自动转换成“较高”类型
- ✓赋值：赋值右边表达式的类型会自动转换为赋值号左边变量类型



◆注意：高精度转换低精度数据时，可能会丢失数据精度

强制类型转换

◆程序员可以显示将数据类型转换为另一类型

✓将需要转换的目标类型放在括号里，将后面的变量或表达式的值转换为目标类型（注意：变量的类型本身不会发生变化，只是返回新类型的结果，可以定义新类型的变量来存储这个结果）

(目标类型) 变量

```
double d = 2.5; //定义double类型的变量，并初始化为2.5
int i = (int) d;//将double类型转换为int类型，保存在i中
```

C02-07：理解类型转换

```
char c = '1';          //字符类型变量
int a = 1, b = 8;      //整数类型变量
double d = 1;          //double类型变量
a = c;                 //隐式类型转换，char转化为int，按ASCII转换
printf("%d\n", a);
//乘法(*)按整数运算，加法(+)按整数运算；除法(/)按double运算，减法(-)按double运算
 a = 60 + 4*50 - 3/2.0;       //赋值(=)时将double隐式转换为int
printf("%d\n", a);
//左边都为整数，按整数运算；运算结束后，执行赋值时转换为double
d = (3+2)*(7-2)/4;
printf("%f\n", d);
c = a;
printf("%d\n", c);
//b/5按照整数运算，运算结果再转double
d = (double) (b/5);
printf("%f\n", d);
//将b转换为double，再除以5（按double运算）
d = (double) (b)/5;
printf("%f\n", d);
```

```
49
258
6.000000
2
1.000000
1.600000
```

类型转换的应用

◆利用类型转换规则，可以获得浮点数的整数和小数部分

```
double x = 2020.321, decimal_part;
int int_part;

int_part = (int) x;                  // 获取x的整数部分
decimal_part = x - (int) x;    // 获取x的小数部分
```

2.5 逻辑表达式：逻辑运算符和关系运算符

- 逻辑表达式：由逻辑运算符或关系运算符构成的表达式，表达式的结果为逻辑值（真1、假0）
- 逻辑运算符
  - ✓与 &&（两个都为真，结果为真；其他为假）
  - ✓或 ||（两个都为假，结果为假；其他为真）
  - ✓非!（真假互换，真变假，假变真）
- 关系运算符
  - ✓大于>，小于<，大于等于>=，小于等于<=、等于==，不等于!=
- 除逻辑非（!）为一元运算符，其他均为二元运算符

&&		A	
		T	F
B	T	T	F
	F	F	F

||		A	
		T	F
B	T	T	T
	F	T	F

A	T	F
!A	F	T

C语言中的逻辑值

- 早期C语言没有逻辑数据类型，整数值可直接转化为逻辑值
  - ✓整数0表示逻辑假；逻辑假对应整数0
  - ✓非0整数表示逻辑真；逻辑真对应整数1
- C99中新增了逻辑类型：bool
  - ✓使用头文件：<stdbool.h>
  - ✓两个逻辑值：true（真）、false（假）

```
#include <stdio.h>
#include <stdbool.h>
int main() {
    bool a = true, b = false;
    printf("a = %d,  b = %d\n", a, b);
    return 0;
}
```

使用逻辑和关系运算符

◆示例1：判断数n是否大于1并小于等于10：1<n≤10

```
(1 < n <= 10) //错误?
```

```
( (n > 1) && (n <= 10) )
```

◆示例2：判断字符变量ch是否是否是字母

```
( ((ch >= 'a') && (ch <= 'z')) || ((ch >= 'A') && (ch <= 'Z')) )
```

◆示例3：判断gcd不是a和b的公约数

```
(!((a % gcd == 0) && (b % gcd == 0)) )
```

◆示例4：判断year年是否是闰年（即能被4整除，并且不能被100整除；或者能被400整除）

```
( (( year%4 == 0) && (year%100 != 0 )) || (year%400 == 0) )
```

逻辑表达式的短路求值

A && B	若A为0（假），则B不必求值，结果一定为0（假）
A || B	若A为非0（真），则B不必求值，结果一定为非0（真）

◆如上逻辑表达式的这一重要性质称为“短路求值”，在程序设计中经常使用

```
//读入字符，如果正确读入，并且不是回车，则循环继续
while( ((ch=getchar()) != EOF) && (ch != '\n') )
{
    ...
}
```

提示：getchar()函数的使用
从键盘读入一个字符，返回读入的值；
如果返回EOF，则表示输入结束，没有字符

C02-08：小写转大写

◆C02-08：输入若干字符（可以多行），将输入中的小写字母转换为大写字母，其他字符保持不变

```c
#include <stdio.h>
int main(){
    char ch;
    while( ( ch = getchar() ) != EOF )
    {
        if((ch>='a') && (ch<='z'))
            printf("%c", ch - 32);   //根据ASCII码规则实现小写转大写
            //printf("%c", ch - 'a' + 'A'));//作用相同，仔细理解
        else
            printf("%c", ch);
    }
    return 0;
}
```

理解：大小写字母的转换规则

思考：如何实现大写转小写？

提示：EOF表示输入（文件）结束，没有新的数据；
本地输入时，先回车，再按Ctrl + Z模拟输入结束

C02-09：水仙花数

- 输入一个三位整数x，判断x是否为水仙花数，如果是，则输出该数是水仙花数
  - ✓水仙花数：一个三位数，其各位数字的立方之和等于该数
  - ✓如：153是水仙花数，因为：$153 = 1^3 + 5^3 + 3^3$
- 要点分析：如何获取3位整数x的每一位值?
  - ✓个位：`x % 10`
  - ✓百位：`x / 100`
  - ✓十位：`(x %100) / 10`

C02-09：水仙花数

```c
#include <stdio.h>
int main()
{
    int x, a, b, c;
    scanf("%d", &x);

    a = x % 10;           //个位
    b = (x % 100) / 10; //十位
    c = x / 100;          //百位
    if (x == a * a * a + b * b * b + c * c * c)
        printf("%d is a daffodil number\n", x);
    else
        printf("%d is a not daffodil number\n", x);
    return 0;
}
```

C02-10：名次预测

◆C02-10：有人在赛前预测A~F六名选手在比赛中会按顺序分获第1名到第6名，如果他在这一预测中刚好猜对了三人的名次，则输出他是一个正常的人（不是小笨，也不是天才）

◆如何计算正确个数

✓穷举：逐一枚举上述预测中只有3项正确的所有可能组合，则逻辑表达式应为：

✓if(((A == 1) && (B == 2) && (C == 3)
&& (D != 4) && (E != 5) && (F != 6))
|| ((A == 1) && (B == 2) && (C != 3)
&& (D == 4) && (E != 5) && (F != 6))
|| ...
)

✓由逻辑或连接起来“多个逻辑与表达式”有20项 C_6^3。每个逻辑与表达式中有6个关系表达式，有 20*6=120 个关系表达式，复杂!

选手	预测名次	实际名次（输入）	猜对情况
A	1	1	1
B	2	2	1
C	3	5	
D	4	4	1
E	5	6	
F	6	3	

C02-10：名次预测

◆C02-10：有人在赛前预测A~F六名选手在比赛中会按顺序分获第1名到第6名，如果他在这一预测中刚好猜对了三人的名次，则输出他是一个正常的人（不是小笨，也不是天才）

```
#include <stdio.h>
int main(){
    int a, b, c, d, e, f;
    printf("The actual ranking of sporters :\n");
    printf("A: ");    scanf("%d", &a);
    printf("B: ");    scanf("%d", &b);
    printf("C: ");    scanf("%d", &c);
    printf("D: ");    scanf("%d", &d);
    printf("E: ");    scanf("%d", &e);
    printf("F: ");    scanf("%d", &f);
    if( (a == 1) + (b == 2) + (c == 3) +
              (d == 4) + (e == 5) + (f == 6) == 3 )
        printf("Got it! You are a normal person.\n");
    return 0;
}
```

理解：充分利用真为0、假为1的性质

思考：输入每个选手实际名次，根据猜对个数，计算幸运指数。输入：

1 3 4 5 6 2

1 2 3 4 5 6

灵活书写逻辑表达式

◆自然语言描述的条件转换为逻辑表达式通常不唯一，如：

✓a不大于b：　(a <= b)，也可以为 !(a >b)

✓a不等于b：　(a != b)，也可以为 !(a == b)

✓x不大于a且不大于b：

x <= a && x <= b

➢也可以为 !(x > a) && !(x > b)

➢还可以为 ! ((x > a) || (x > b))

德.摩根定理（反演律）

运算符的优先级和结合律

- 基本优先级规则：
  - ✓先乘除（模），后加减
  - ✓关系运算优于逻辑运算
  - ✓括号优先
  - ✓……
- 提示：遇事不决，加括号！

```
x = a - b * c;
```

```
if( score >= 80 && score < 90 )
{ …… }
```

```
char c;
if( ((c = getchar( ))!= EOF) && (c >= 'a') && (c <= 'z') )
{…}
```

运算符的优先级和结合律

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	后置++	后置自增运算符	变量名++	左到右	
	后置--	后置自减运算符	变量名--		
	[ ]	数组下标	数组名[整型表达式]		
	( )	圆括号	（表达式）/函数名(形参表)		
	.	成员选择（对象）	对象.成员名		
	->	成员选择（指针）	对象指针->成员名		
2	-	负号运算符	-表达式	右到左	单目运算符
	(类型)	强制类型转换	(数据类型)表达式		
	前置++	前置自增运算符	++变量名		单目运算符
	前置--	前置自减运算符	--变量名		单目运算符
	*	取值运算符	*指针表达式		单目运算符
	&	取地址运算符	&左值表达式		单目运算符
	!	逻辑非运算符	!表达式		单目运算符
	~	按位取反运算符	~表达式		单目运算符
	sizeof	长度运算符	sizeof 表达式/sizeof(类型)		
3	/	除	表达式/表达式	左到右	双目运算符
	*	乘	表达式*表达式		双目运算符
	%	余数（取模）	整型表达式%整型表达式		双目运算符

运算符的优先级和结合律（续）

优先级	运算符	名称或含义	使用形式	结合方向	说明
4	+	加	表达式+表达式	左到右	双目运算符
	-	减	表达式-表达式		双目运算符
5	<<	左移	表达式<<表达式	左到右	双目运算符
	>>	右移	表达式>>表达式		双目运算符
6	>	大于	表达式>表达式	左到右	双目运算符
	>=	大于等于	表达式>=表达式		双目运算符
	<	小于	表达式<表达式		双目运算符
	<=	小于等于	表达式<=表达式		双目运算符
7	==	等于	表达式==表达式	左到右	双目运算符
	!=	不等于	表达式!= 表达式		双目运算符
8	&	按位与	整型表达式&整型表达式	左到右	双目运算符
9	^	按位异或	整型表达式^整型表达式	左到右	双目运算符
10	|	按位或	整型表达式|整型表达式	左到右	双目运算符
11	&&	逻辑与	表达式&&表达式	左到右	双目运算符
12	||	逻辑或	表达式||表达式	左到右	双目运算符
13	?:	条件运算符	表达式1? 表达式2: 表达式3	右到左	三目运算符

运算符的优先级和结合律（续）

优先级	运算符	名称或含义	使用形式	结合方向	说明
14	=	赋值运算符	变量=表达式	右到左	
	/=	除后赋值	变量/=表达式		
	*=	乘后赋值	变量*=表达式		
	%=	取模后赋值	变量%=表达式		
	+=	加后赋值	变量+=表达式		
	-=	减后赋值	变量-=表达式		
	<<=	左移后赋值	变量<<=表达式		
	>>=	右移后赋值	变量>>=表达式		
	&=	按位与后赋值	变量&=表达式		
	^=	按位异或后赋值	变量^=表达式		
	|=	按位或后赋值	变量|=表达式		
15	,	逗号运算符	表达式,表达式,...	左到右	从左向右顺序运算

这么多！怎么记忆?

1. 读一遍，能记住多少就记多少。
2. 使用的时候，按常识和感觉，相信自己的直觉。
3. 加括号。
4. 不理它，天天坚持编程（模仿书上的例程），很快，不知为何，就都会了。

小结：运算符的优先级和结合律

- ◆按前述表格，优先级从上到下依次递减，最上面具有最高的优先级，逗号操作符具有最低的优先级
- ◆相同优先级中，按结合顺序（结合律）计算
  - ✓大多数运算是从左至右计算，
  - ✓三个优先级是从右至左结合的：单目运算符、条件运算符、赋值运算符
- ◆基本的优先级需要记住：
  - ✓指针最优
  - ✓单目运算优于双目运算
    - ➢如正负号，如 -5+6
  - ✓先乘除（模），后加减
  - ✓逻辑运算最后计算
  - ✓不清楚的地方，善用小括号！
- ◆很多优先级是比较显然的。记住一些常用运算的优先级。记不住怎么办？

2.6 标准库函数

- C语言标准提供了很多可直接使用的基础功能函数
  - ✓这些函数被封装在不同的头文件中，通过“函数名(...)”的方式调用
  - ✓标准库函数的函数原型在相应的标准库头文件（*.h）中说明
  - ✓函数的定义以目标码的形式保存在函数库中（用户使用时，用#include 引用相应的 .h 文件，编译时以编译选项的方式指明需要链接的库函数）

```c
#include <stdio.h>
#include <math.h>
int main(){
    double a=3, b=4, c;

    c = sqrt(a+b);  //math.h库中的开方函数
    printf("%.2f\n", c);//stdio.h的打印函数
    return 0;
}
```

提示：善于使用标准库函数，有标准库函数实现的功能优先使用标准库
包含头文件，使用库函数！

使用标准库函数

- 理解函数的参数和返回值的含义
  - ✓查参考书
  - ✓查网站
    - http://www.cplusplus.com/
    - https://zh.cppreference.com/
    - 百度搜索、……

```
double pow( double base, double exponent );
参数
    base：作为底的浮点值
    exponent：作为指数的浮点值
返回值
    返回 base 的 exponent 次幂（ base^exponent）
```

- 使用时必须严格按照函数的规格说明传递参数，接收返回值
  - ✓返回值 = 函数名(参数表)

```
int base, exp;
//调用pow函数，计算base^exp，结果保存在ans中
double ans = pow (base, exp);
```

C89中的标准库函数

头文件	宏及函数的功能类别
assert.h	运行时的断言检查
ctype.h	字符类型和映射
errno.h	错误信息及处理
float.h	对浮点数的限制
limits.h	编译系统实现时的限制
locale.h	建立与修改本地环境
math.h	数学函数库
setjmp.h	非局部跳转

头文件	宏及函数的功能类别
signal.h	事件信号处理
stdarg.h	变长参数表处理
stddef.h	公用的宏和类型
stdio.h	数据输入输出
stdlib.h	不属于其它类别的常用函数
string.h	字符串处理
time.h	日期和时间

标准库函数：stdio.h

常用的I/O函数（主要文本的 I/O）（更多含义，参阅教材附录）

函数原型	功能说明	备注
int getchar();	返回从标准输入文件中读入的一个字符	读到文件尾时返回EOF
int putchar(int c);	向标准输出文件中写入一个字符	函数出错时返回EOF
char *gets(char *buf);	从标准输入文件中读入的一行字符，并将其保存在buf所指向的存储区	当读到文件尾部或函数出错时返回NULL，否则返回存储区buf
char *fgets(char *buf, int n,FILE *fp);	从指定文件中读入的一行不超过n-1个字符的字符串，并将其保存在buf所指向的存储区	当读到文件尾部或函数出错时返回NULL，否则返回存储区buf
int puts(const char *string);	向标准输出文件中写入的一行字符，并将字符末尾处的结束符'\0'替换成换行符'\n'	函数出错时返回EOF
int scanf(const char *format [,argument]...);	根据format中的格式规定从标准输入文件中读入数据，并保存到由argument指定的存储单元中	当读到文件尾部或函数出错时返回EOF，否则返回读入数据的个数
int printf(const char *format [,argument]...);	根据format中的格式规定将argument指定的数据写入标准输出文件中	返回输出的字符数，函数出错时返回负值

标准库函数：math.h

常用数学计算（include <math.h>）（参数和返回类型都是double）

函数原型	函数功能	函数原型	函数功能
double sqrt(double x);	x的平方根	double asin(double x);	反正弦函数，x以弧度为单位
double sin(double x);	正弦函数，x以弧度为单位	double acos(double x);	反余弦函数，x以弧度为单位
double cos(double x);	余弦函数，x以弧度为单位	double log(double x);	x的自然对数
double tan(double x);	正切函数，x以弧度为单位	double log10(double x);	x的常用对数
double atan(double x);	反正弦函数，返回值以弧度为单位	double exp(double x);	指数函数ex
double atan2(double y,double x);	反正弦函数，返回值以弧度为单位，根据x和y的符号确定象限	double fabs(double x);	x的绝对值

标准库函数：stdlib.h

通用数据处理（include <stdlib.h>），求整数绝对值、随机数、快速排序等

函数原型	函数功能	函数原型	函数功能
int abs(int n);	n的绝对值	int rand();	生成伪随机数
double atof(char *s);	将s转换为double类型的数	void srand(unsigned int s);	设置随机数的种子
int atoi(char *s);	将s转换为int类型的数	void exit(int status);	终止程序运行
void *bsearch(void *key,void *base,size_t num,size_t width,int (*f)(void *e1, void *e2));	以二分查找的方式在排序数组base中查找元素key	void qsort(void *base,size_t num,size_t width,int (*f)(void *e1, void *e2));	以快速排序方式对数组base进行排序

标准库函数：ctype.h

字符类型判断和处理函数

函数原型	函数功能
int isalnum(int c);	c是否是字母或数字
int isalpha(int c);	c是否是字母(a~z,A~Z)
int iscntrl(int c);	c是否是控制符(0x00~0xlf,0x7f)
int isdigital(int c);	c是否是数字
int isgraph(int c);	c是否是可打印字符(空格除外)
int islower(int c);	c是否是小写字母(a~z)
int tolower(int c);	将c转换为小写字母
int toupper(int c);	将c转换为大写字母

函数原型	函数功能
int isprint(int c);	c是否是可打印字符(0x20~0x7e)
int ispunct(int c);	c是否是符号，可打印但非字母数字
int isspace(int c);	c是否是空白符(0x09~0x0D,0x20)
int isupper(int c);	c是否是大写字母(A~Z)
int isxdigit(int c);	c是否是16进制数字(0~9，a~f，A~F)

<table>
<tr><td rowspan="6">isprint()</td><td rowspan="5">isgraph()</td><td rowspan="2">isxdigit()</td><td>isdigit()</td><td>0~9</td><td></td></tr>
<tr><td colspan="3">A~F,a~f</td></tr>
<tr><td rowspan="3">isalnum()</td><td rowspan="2">isalpha()</td><td>isupper()</td><td>A~Z</td></tr>
<tr><td>islowwe()</td><td>a~z</td></tr>
<tr><td>isdigit()</td><td colspan="2">0~9</td></tr>
<tr><td>ispunct()</td><td colspan="3">!@#$%^&*()_-+=|\{}:;"'<>?/~`</td></tr>
<tr><td colspan="4">空格符(' ',0x20)</td></tr>
<tr><td>isspace()</td><td colspan="4">空格符(0x20)，换页符('\f'),换行符('\n'),回车符('\r'),水平制表符('\t'),垂直制表符('\v')</td></tr>
<tr><td>iscntrl()</td><td colspan="4">控制字符(0x00~0x1f,0x7f)</td></tr>
</table>

标准库函数：string.h

字符串数处理函数（学完字符串之后再学使用）

函数原型	函数功能
char *strcat(char *dst, char *src);	将src追加到dst之后
char *strncat(char *dst, char *src,size_t n);	将src中的前n个字符追加到dst之后
char *strcpy(char *dst, char *src);	将src复制到dst中
char *strncpy(char *dst, char *src,size_t n);	将src中的前n个字符复制到dst中
size_t strlen(char *str);	返回字符串str的长度

函数原型	函数功能
int strcmp(char *s1, char *s2);	比较字符串s1和s2
int strncmp(char *s1,char *s2,size_t n);	比较字符串s1和s2的前n个字符
char *strchr(char *str,int c);	在str中查找c首次出现的位置
char *strrchr(char *str, int c);	在str中查找c最后一次出现的位置
char *strstr(char *str, char *s1);	在str中查找s1首次出现的位置

2.7 标准输入输出函数：scanf和printf

- OJ评测原理就是给定输入，判断输出是否符合要求
  - ✓必须严格按照题目要求的输入格式读取数据
  - ✓必须严格按照题目要求的输出格式显示数据
- 用好scanf和printf，严格按照要求读取和显示数据

```
//单个字符输入
int getchar();
//单个字符输出
int putchar(int c);
//格式化输入，根据format格式，读入各种类型的变量
int scanf(const char *format [,argument]...);
//格式化输出，根据format格式，输出各种类型的变量
int printf(const char *format [,argument]...);
```

标准输入函数scanf

```
int scanf(const char *format [,argument]...);
```

- ◆format格式控制串：占位符字符串
  - ✓转换控制字符（占位符，以%开始）决定输入数据的个数和格式
  - ✓其它字符必须原样输入
  - ✓空格可以接收输入的空格符、制表符、换行符等空白字符
- ◆argument参数
  - ✓个数和类型必须与format中的转换控制符严格一致
  - ✓除指针变量外，一般变量前面一定加上&，表示保存变量的地址
- ◆返回值
  - ✓成功赋值的数据个数（全部读入成功即为后面参数的个数）
  - ✓读到文件末尾而没有数据时返回 EOF (符号宏，实际值为：-1)

scanf中的格式控制符

转换控制字符(输入占位符)	输入形式	对应参数类型	输入实例
%d	32位有符号整数	int	123、-123
%u	32位无符号整数	unsigned int	123
%lld	64位有符号整数	long long int	10000000000ll
%llu	64位无符号整数	unsigned long long	10000000000llu
%x	十六进制整数形式	int	8C
%c	一个字符	char	x
%s	一个字符串（不含空白符）	char *或char[]	hello123
%f	小数形式（单精度浮点数）	float	-1.23
%lf	小数形式（双精度浮点数）	double	-1.23

- ◆转换控制字符和后面的变量列表严格一一对应
- ◆读入时，除%c外，其他控制字符（如：%d, %f, %lf），都会跳过输入数据前面的空格符、制表符、换行符等空白符，从第一个有效字符开始读入数据直到遇到非有效字符，读入完成后尝试按规定格式转换；转换成功，则数据读入内存，转换失败，读入停止

C02-11：按照格式控制符输入

◆从键盘读入圆的信息，包括圆心和半径（double类型），输出圆的周长和面积（PI值取3.14），输入格式为：

- ✓第一行半径：r
- ✓第二行圆心：(x, y)
- ✓样例数据：
  - ➢2
  - ➢(3.5, 4.8)

提示：除%c外，其它格式控制符自动忽略掉输入前面的空格；但其他的普通字符则必须原样输入，否则出错！
例中3.5、4.3数字前面空格不需要处理，但‘(’、‘,’这种格式串前面空白字符则必须单独处理

```
#include <stdio.h>
int main()
{
    double x, y, r;
    double PI= 3.14;

    scanf("%lf", &r); //读入半径信息
    //(前面一定要有空格，以便处理半径输入后剩余的回车
    scanf(" (%lf,%lf)", &x, &y);

    printf("%6.2f\n%6.2f", 2*PI*r, PI*r*r);
```

OJ中常见输入方式的处理

◆（1）单组数据输入

✓定义相应的变量，按照指定的格式输入

```
//C02-12
输入两个数a和b，求a+b的值。
输入：一行，分别为整数 a 和 b（空格分开）
输出：一个数，表示a+b的值。

输入样例：
1 2
输出样例：
3
```

```c
#include <stdio.h>

int main()
{
    int a, b;
    int sum;

    scanf("%d%d", &a, &b);
    sum = a + b;
    printf("%d\n", sum);

    return 0;
}
```

OJ中常见输入方式的处理

◆（2）多组数据输入：给定数据组数为n，计算n组a+b

//C02-13

给定数据组数n，每组两个数a和b，求a+b的值

输入：第一行为n，后面n行，分别为整数a 和 b

输出：n行，表示a+b的值。

输入样例：

3

1 2

3 4

5 6

输出样例：

3

7

11

```c
#include <stdio.h>

int main()
{
    int n, a, b;
    int sum;
    scanf("%d", &n);//输入n
    while(n--){ //循环n次
        scanf("%d%d", &a, &b);
        sum = a + b;
        printf("%d\n", sum);
    }

    return 0;
}
```

OJ中常见输入方式的处理

- （3）多组数据输入：数据组数不确定，直到输入结束
  - ✓本地模拟输入结束方式：输入完成后回车，再输入ctrl+Z

```
//C02-14
输入若干行，每行两个数a和b，求a+b的值
输入：若干行，每个两个整数a 和 b
输出：每行a+b的值
输入样例:
1 2
3 4
输出样例:
3
7
```

提示：scanf( )函数有返回值，若输入成功，则返回输入的数据项数，成功读到文件结束末尾时返回 EOF (-1)，输入错误返回0

```c
#include <stdio.h>

int main()
{
    int a, b;
    int sum;
    //循环读入数据，直到没有数据可读入(EOF)
    while(scanf("%d%d",&a, &b)!=EOF){
        sum = a + b;
        printf("%d\n", sum);
    }

    return 0;
}
```

标准输出函数printf

```
int printf(const char *format [,argument]...);
```

转换控制字符(输入占位符)	输入形式	对应参数类型	输出实例
%d	32位有符号整数	int	123、-123
%u	32位无符号整数	unsigned int	123
%lld	64位有符号整数	long long int	10000000000
%llu	64位无符号整数	unsigned long long	10000000000
%x	十六进制整数形式	int	8C
%c	一个字符	char	x
%s	一个字符串（不含空白符）	char *或char[]	hello123
%f	小数形式（单精度浮点数）	float	-1.23
%e	小数形式（科学计数表示）	double、float	-1.23e-3
%%	转义符：输出%号		

◆格式控制符与scanf基本相同，但有些细节不同

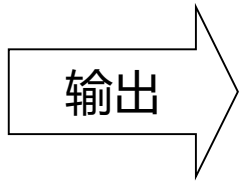
✓不区分double和float，都用%f；%e可以输出科学计数法；%%输出%

输出的高级格式控制

- 输出时，可以在转换控制符的%和字母之间添加符号，进行复杂的格式控制
- 对于整数输出，常见的格式控制包括：
  - ✓**%4d**：输出最少占4个位置（超出则按实际长度），不足的在左边（前面）补空格（右对齐）
  - ✓**%04d**：输出最少占4个位置（超出则按实际长度），不足的在左边（前面）补0（右对齐）
  - ✓%-4d：输出最少占4个位置（超出则按实际长度），不足的在右边（后面）补空格（左对齐）
- 对于浮点数输出，与整数类似，但可以指定小数点后面的位数
  - ✓%6.2f：输出最少占6个位置（含小数点，超出则按实际长度），不足的在右边（后面）补空格（左对齐）
- 更多的输出格式控制，请自行查阅资料

C02-15：控制输出格式

```c
#include <stdio.h>
int main()
{
    int a=5;
    double pi=3.14159265;
    //输出6位整数，不足签名补0
    printf("%06d\n", a);
    //输出double，不控制格式
    printf("%f\n", pi);
    //输出double，6位，小数点后2位
    printf("%6.2f\n", pi);
}
```




```
000005
3.141593
  3.14
```

C02-16：求圆的面积和周长（复杂的控制格式）

```c
#include <stdio.h>
#define PI 3.14
int main() {
    double radius, area, perimeter;
    scanf("%lf", &radius);
    area = PI * radius * radius;
    perimeter = 2 * radius * PI;

    printf("Radius = %6.2f\n", radius);
    printf("Area = %6.2f\n", area);
    printf("Perimeter = %6.2f\n\n", perimeter);

    printf("Radius    = %6.2f\n", radius);
    printf("Area      = %6.2f\n", area);
    printf("Perimeter = %6.2f\n\n", perimeter);

    printf("   Radius = %6.2f\n", radius);
    printf("     Area = %6.2f\n", area);
    printf("Perimeter = %6.2f\n\n", perimeter);

    printf("%12s = %6.2f\n", "Radius", radius);
    printf("%12s = %6.2f\n", "Area", area);
    printf("%12s = %6.2f\n\n", "Perimeter",perimeter);

    printf("%-12s = %6.2f\n", "Radius", radius);
    printf("%-12s = %6.2f\n", "Area", area);
    printf("%-12s = %6.2f\n\n", "Perimeter",perimeter);
    return 0;
}
```

输出

```
1
Radius =   1.00
Area =   3.14
Perimeter =   6.28

Radius    =   1.00
Area      =   3.14
Perimeter =   6.28

   Radius =   1.00
     Area =   3.14
Perimeter =   6.28

      Radius =   1.00
        Area =   3.14
   Perimeter =   6.28

Radius       =   1.00
Area         =   3.14
Perimeter    =   6.28
```

常用的输出格式：

- 左对齐，
- 右对齐，
- 占指定宽度，
- 等等

小结

- 变量的定义和使用
  - ✓标识符；类型（int、float、double、char）；初始化和赋值
- 一维数组的基本使用规则：长度和下标
- 字面量：直接写在代码中的数据值
- 表达式
  - ✓算术运算符、关系运算符和逻辑运算法
  - ✓运算符的优先级
- 标准库函数：stdio.h、math.h、ctype.h、string.h
- 标准输入输出函数：scanf、printf

补充：define定义符号常量和符号宏

◆#define　是编译预处理中的宏定义，

✓定义一个符号和它所代表的字符串

✓编译预处理时这一符号被替换为该字符串。

✓例：EOF 就是 stdio.h 中定义的常量

妙用define能“加快”编程速度（看大佬）

看大佬如何能能在7分钟内做出4道题！（注：题目并不是太简单）

C	D	E	F
188/200	171/187	169/177	71/106
0:01:40	0:00:55	0:05:14	0:07:12

兵马未动

粮草先行

```
/*
 Author: [redacted]
 Result: AC      Submission_id: 1906614
 Created at: Wed Oct 16 2019 19:00:55 GMT+0800 (CST)
 Problem: 2533  Time: 2 Memory: 3052
*/

#include <cstdio>
#include <iostream>
#include <string>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <climits>
#include <cstdint>
#include <algorithm>
#include <vector>
#include <map>
#include <set>

#define IL inline
#define LL long long
#define ULL unsigned LL
#define _BS 1048576
char _bf[_BS];
char *__p1=_bf,*__p2=_bf;
#define _IO char *_p1=__p1,*_p2=__p2;
#define _OI __p1=_p1,__p2=_p2;

#define GC getchar()
#define PC putchar
#define _Q_(x) {register char _c=GC,_v=1;for(x=0;_c<48||_c>57;_c=GC)if(_c==45)_v=-1;
#define _H_(x) for(;_c>=48&&_c<=57;x=(x<<1)+(x<<3)+_c-48,_c=GC);if(_v==-1)x=-x;}
#define sc(x) _Q_(x)_H_(x)
#define se(x) _Q_(x)else if(_c==EOF)return 0;_H_(x)
#define _G1(_1) int _1;sc(_1)
#define _G2(_1,_2) int _1,_2;sc(_1)sc(_2)
#define _G3(_1,_2,_3) int _1,_2,_3;sc(_1)sc(_2)sc(_3)
#define _gG(_1,_2,_3,_get, ...) _get
#define get(...) _gG(__VA_ARGS__,_G3,_G2,_G1, ...)(__VA_ARGS__)
#define F(i,l,r) for(int i=(l);i<(r);++i)
#define FD(i,l,r) for(int i=(l);i<=(r);++i)

template<class T>
void UPRT(const T _){if(_>=10)UPRT(_/10);PC(_%10+48);}
#define UPL(_) UPRT(_),PC(10)
template<class T>
void PRT(const T _){if(_<0){PC(45),UPRT<ULL>(-(ULL)_);return;}if(_>=10)PRT(_/10);PC(_
#define PL(_) PRT(_),PC(10)

#define MIN(a, b) ((a)<(b)?(a):(b))

constexpr int MN(1e3+7);

LL dp[2][MN], t[2][MN], p1[MN], p2[MN];

int main()
{
        int n;
```

字面量的符号表示方法

- #define 编译预处理中宏定义
- 用一个标识符来表示（定义）一个字符串，称为“宏”
  - ✓在编译预处理时，对程序中所有出现的“宏”，都用宏定义中的字符串去代换，这称为“宏代换”或“宏展开”
  - ✓宏代换是由预处理程序在编译前完成
- 宏代换是纯粹的文字替换，不进行类型检查
- 使用常量定义的好处：
  - ✓可提高程序的可读性
  - ✓程序的可移植性更好
  - ✓可维护性更好

```c
#include <stdio.h>
#define SEC_HOUR (60 * 60)
#define SEC_DAY (SEC_HOUR * 24)
#define SUCCEED "test, Succeed!\n"

int main()
{
    int sec_oneweek = 7 * SEC_DAY;
    printf("seconds of oneweek: %d\n", sec_oneweek);

    printf(SUCCEED);
    return 0;
}
```

输出：

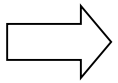
seconds of oneweek: 604800

test, Succeed!

问题：如上代码定义了多少个宏?

字面量的符号表示方法

```
1   #include <stdio.h>
2   #define SEC_HOUR (60 * 60)
3   #define SEC_DAY (SEC_HOUR * 24)
4   #define SUCCEED "test, Succeed!\n"
5
6   int main()
7   {
8       int sec_oneweek = 7 * SEC_DAY;
9       printf("seconds of oneweek: %d\n", sec_oneweek);
10
11      printf(SUCCEED);
12      return 0;
13  }
```



```
#include <stdio.h>
#define SEC_HOUR (60 * 60)
#define SEC_DAY (SEC_HOUR * 24)
#define SUCCEED "test, Succeed!\n"

int main()
{
    int sec_oneweek = 7 * ((60 * 60) * 24);
    printf("seconds of oneweek: %d\n", sec_oneweek);

    printf("test, Succeed!\n");
    return 0;
}
```

宏替换示例：

第8行： SEC_DAY ⇨ (SEC_HOUR * 24) ⇨ ((60 * 60) * 24)

第11行： SUCCEED ⇨ "test, Succeed!\n"

#define 的更多知识

- ◆宏可以带参数，替换时，相应参数也进行替换，如
  - ✓#define _DIV(a,b) a/b
  - ✓(1) 程序中出现 _DIV(1,2) 时就替换为 1/2
  - ✓(2) 程序中出现 _DIV(3,4) 时就替换为 3/4
  - ✓(3) 程序中出现 _DIV(1+3,3+5) 时就替换为 1+3/3+5
  - ✓【此处，题意是希望计算(1+3)/(3+5)，但替换后没有实现此功能，如何修改？】
- ◆宏定义中的常见错误
  - ✓宏定义的最后加上多余的分号，如 #define N 10;
  - ✓带参数的宏中漏写了必要的括号，如
    #define _DIV(a,b) a/b 应写成 #define _DIV(a,b) (a)/(b)
    则(3)中的错误就可以避免

```c
#include <stdio.h>
#define N 200
#define _F(i, Lf, Rt) for (i = Lf; i < Rt; i++)
#define _CUBE(a) a *a *a
//#define _CUBE(a) (a) *(a) *(a)

int main()
{
    int sum = 0, n = 0, i, score[N] = {0};
    int cubesum = 0;
    while (scanf("%d", &score[n]) != EOF)
        n++;
    _F(i, 0, n) //for (i = 0; i < n; i++)
    {
        sum += score[i];
        cubesum += _CUBE(score[i]);
        //cubesum += _CUBE(score[i]+1);
    }
    printf("%d\n", sum);
    printf("%d\n", cubesum);
    return 0;
}
```

#define与const

- ◆const定义“常数”（常量变量），该常量带类型，存在于程序的数据段，在内存中分配空间
  - ✓const常量是一个运行时(Run-Time)概念，在程序中真实存在着，并可以被使用；const常量有数据类型，编译器可以对const常量进行类型的安全检查
  - ✓C语言中，const常量不宜用作数组定义的大小。
- ◆#define 定义的是一个符号表示，不带类型
  - ✓用来做文本的替换，当程序进行编译预处理时，所有宏都被替换成相应的字符串，再进行编译
  - ✓#define常量是一个编译时(Compile-Time)概念，#define常量的生命周期停止于编译期，存在于程序的代码段，在实际程序中只是一个常数

```
…
#define arraySize 10

// const int arraySize = 10;

…
int main()
{
    int i=0, j=0, s[arraySize];

    while(i<arraySize)
    { …… }

    while(i<arraySize)
    { …… }

    return 0;
}
```