计算机导论与程序设计 [CS006001]

段江涛

机电工程学院

西安电子科技大学 XIDIAN UNIVERSITY

2022 年 10 月

Outlines

Part I

课程介绍, C 语言简介, 数据输入输出, 基本数据类型与表达式

Outlines

课程内容

- 计算机导论：了解计算机的基本知识；掌握计算机操作基本技能。
- 程序设计：掌握结构化程序设计方法，训练程序逻辑思维能力。会读、会编、会调试 C 语言程序。
- 学习方法：线上、线下相结合。认真书写课堂笔记，按时完成上机练习作业，鼓励大量编程练习。
- 教材
  - 大学计算机，龚尚福，贾澎涛，西安电子科技大学出版社
  - C 程序设计第五版，谭浩强，清华大学出版社
- 线上参考课程资源链接：online resource.pdf

线上导论部分学习内容

1. 计算机历史、现状、发展趋势与前沿技术概述
2. 计算机体系结构及其编码方式
3. 计算机组成与软件系统
4. 计算机应用实践

考核

1. 平时成绩: 10% 由上机练习, 课堂讨论等部分组成。
2. 导论部分: 20% 结合线上资源, 自学字处理软件。总结知识点、课堂笔记, 撰写课程学习报告。
3. 期中考试: 30% 根据机试系统给出的题目编写程序, 通过调试得到正确结果并通过**机试系统提交**。
4. 期末考试: 40% 根据机试系统给出的题目编写程序, 通过调试得到正确结果并通过**机试系统提交**。

计算机导论主要内容

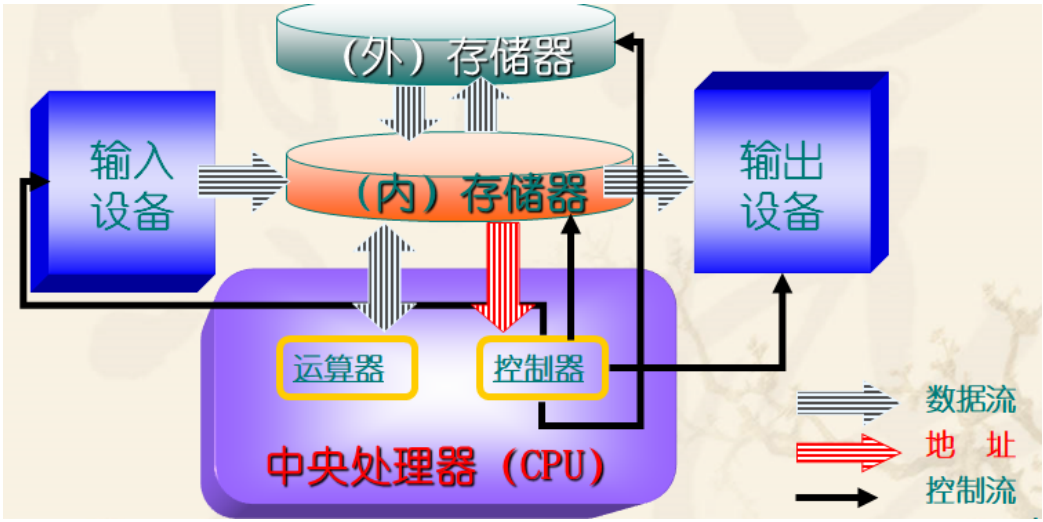
总体要求：了解计算机的基本知识；掌握计算机操作基本技能。

- 计算机系统组成
- 计算机工作原理
- 操作系统
- 字处理：Microsoft Word
- 电子表格：Microsoft Excel
- 演示文稿：Microsoft PowerPoint

计算机工作原理

工作原理: “存储程序” + “程序控制”

1. 以**二进制**形式表示数据和指令
2. 将程序存入存储器中, 由控制器自动读取并执行
3. 外部存储器存储的程序和所需数据 $\Longrightarrow$ 计算机内存 $\Longrightarrow$ 在程序控制下由 CPU 周而复始地取出指令、分析指令、执行指令 $\Longrightarrow$ 操作完成。



计算机程序

指令

可以被计算机理解并执行的基本操作命令。

程序

一组计算机能识别和执行的指令。
一个特定的指令序列用来完成一定的功能。

软件

与计算机系统操作有关的计算机程序、规程、规则，以及可能有的文件、文档及数据。

计算机语言

机器语言

计算机能直接识别和接受的二进制代码称为**机器指令**。机器指令的集合就是该计算机的**机器语言**。

特点：难学，难记，难检查，难修改，难以推广使用。依赖具体机器难以移植。

```
B8  7F 01
BB  21 02
03   D8
B8  1F 04
2B  C3
```

汇编语言

机器语言的符号化。用英文字母和数字表示指令的**符号语言**。

特点：相比机器语言简单好记，但仍然难以普及。

汇编指令需通过**汇编程序**转换为机器指令才能被计算机执行。依赖具体机器难以移植。

```
MOV  AX  383
MOV  BX  545
ADD   BX  AX
MOV  AX  1055
SUB    AX  BX
```

高级语言

高级语言更接近于人们习惯使用的自然语言和数学语言。

特点：功能强大，不依赖于具体机器。用高级语言编写的**源程序**需要通过**编译程序**转换为机器指令的**目标程序**。

```
int  x =1055, y = 383, z = 545
int S;
S = x-(y+z);
S=1055-(383+545)
```

高级语言的发展

结构化语言

规定：

程序必须由具有良好特性的基本结构(**顺序结构、选择结构、循环结构**)构成，程序中的流程不允许随意跳转，程序总是由上而下顺序执行各个基本结构。

特点：

程序结构清晰，易于编写、阅读和维护。

C 语言的特点

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. C 语言是完全模块化和结构化的语言

   具有结构化的控制语句 (顺序、选择、循环结构)

   用函数作为程序的模块单位,便于实现程序的模块化
5. 兼具高级语言和低级语言的功能

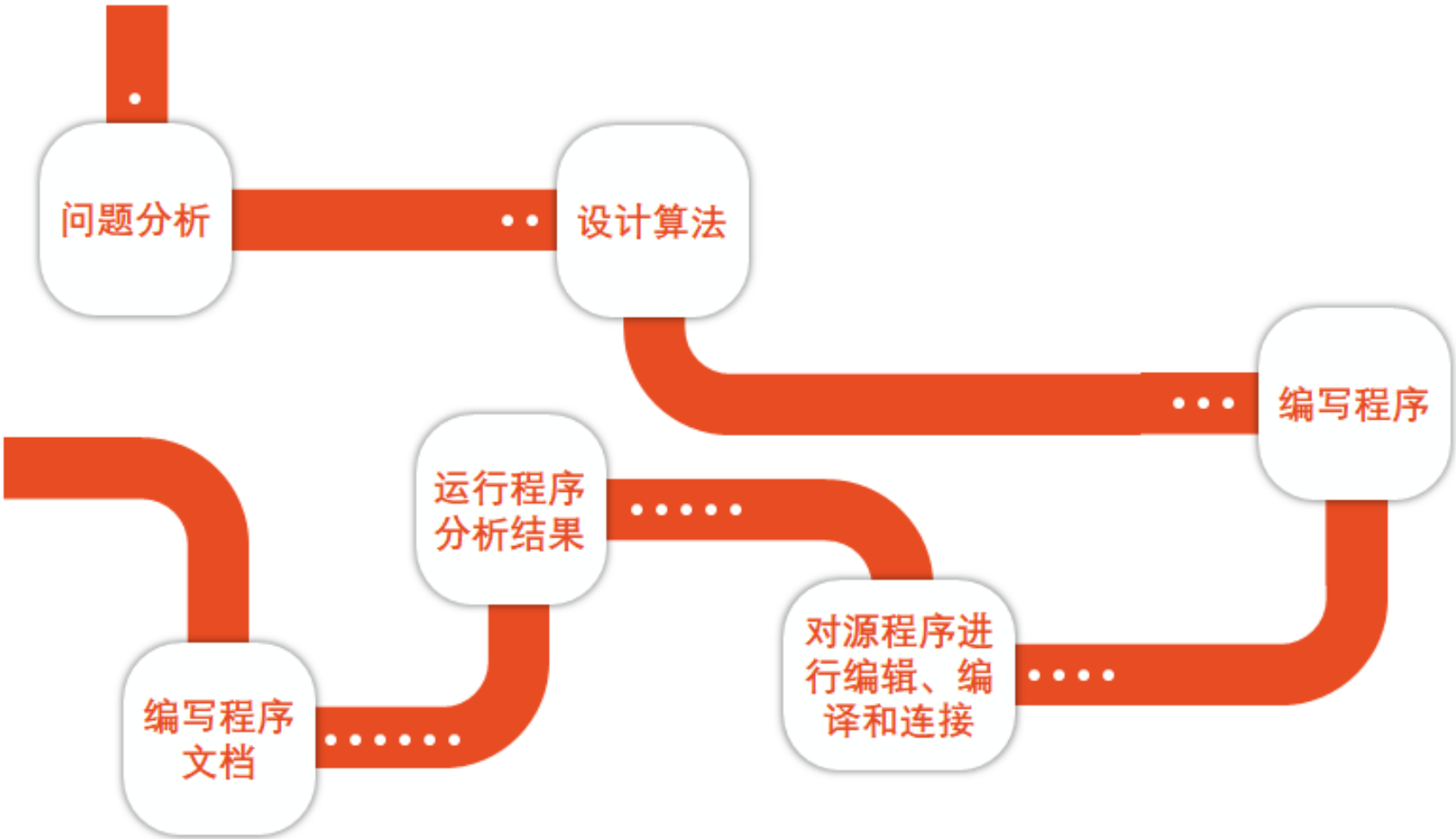
   允许直接访问物理地址

   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

程序设计的任务



第一个 C 语言程序

```c
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
   printf("Hello World!"); // printf函数，输出一行信息
   return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

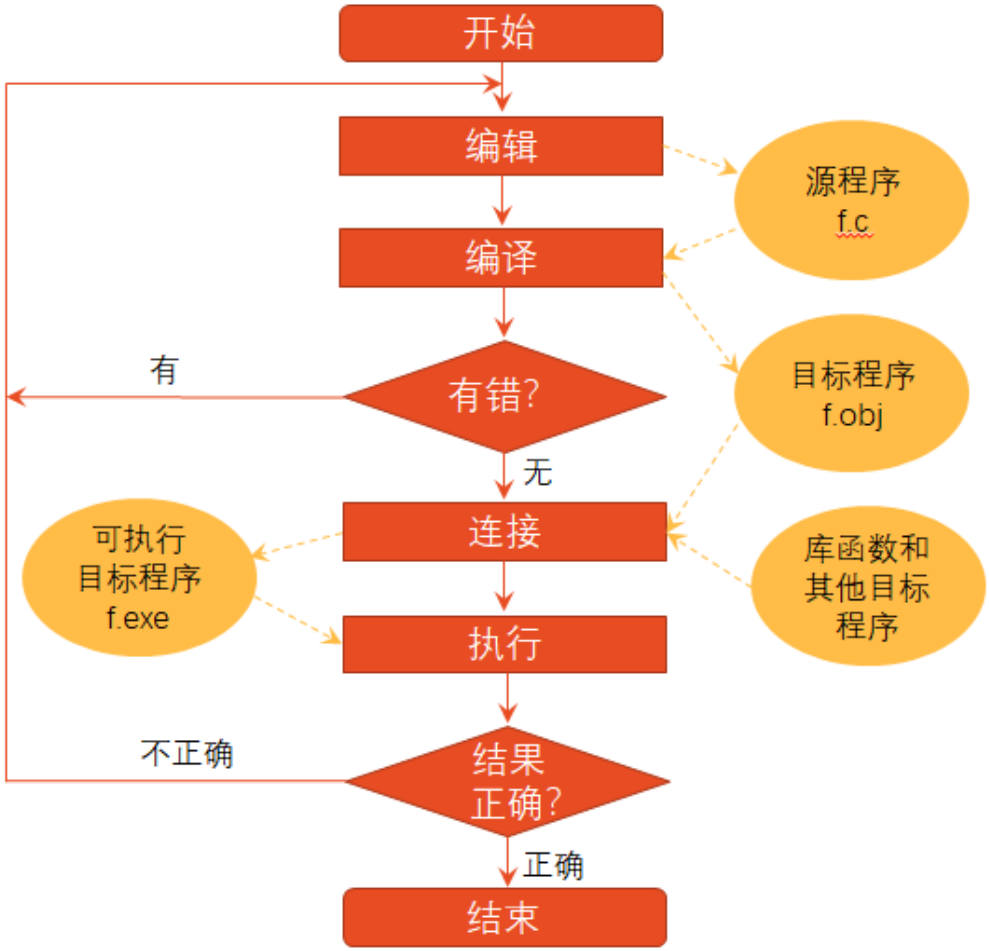
求两个整数之和

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
   int a,b,sum; // 定义a,b,sum为整型变量
   a=123; // 对变量a赋值
   b=456;     // 对变量b赋值
   sum=a+b; // 计算a+b, 并把结果存放在变量sum中
   printf("sum is %d\n",sum);// printf函数, 输出结果
   return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

求 5! 的 C 语言程序

```c
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
    int i,p; // p表示被乘数, i表示乘数
    p=1; // 对变量p赋值
    i=2; // 对循环变量i赋值
    while(i<=5) // 循环结构
    {
        p=p*i; // =右端p记录本语句执行前的值, 左端p记录执行后p的值。
        i++; // i = i + 1
    }
    printf("%d\n",p); // printf函数, 输出p的计算结果
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

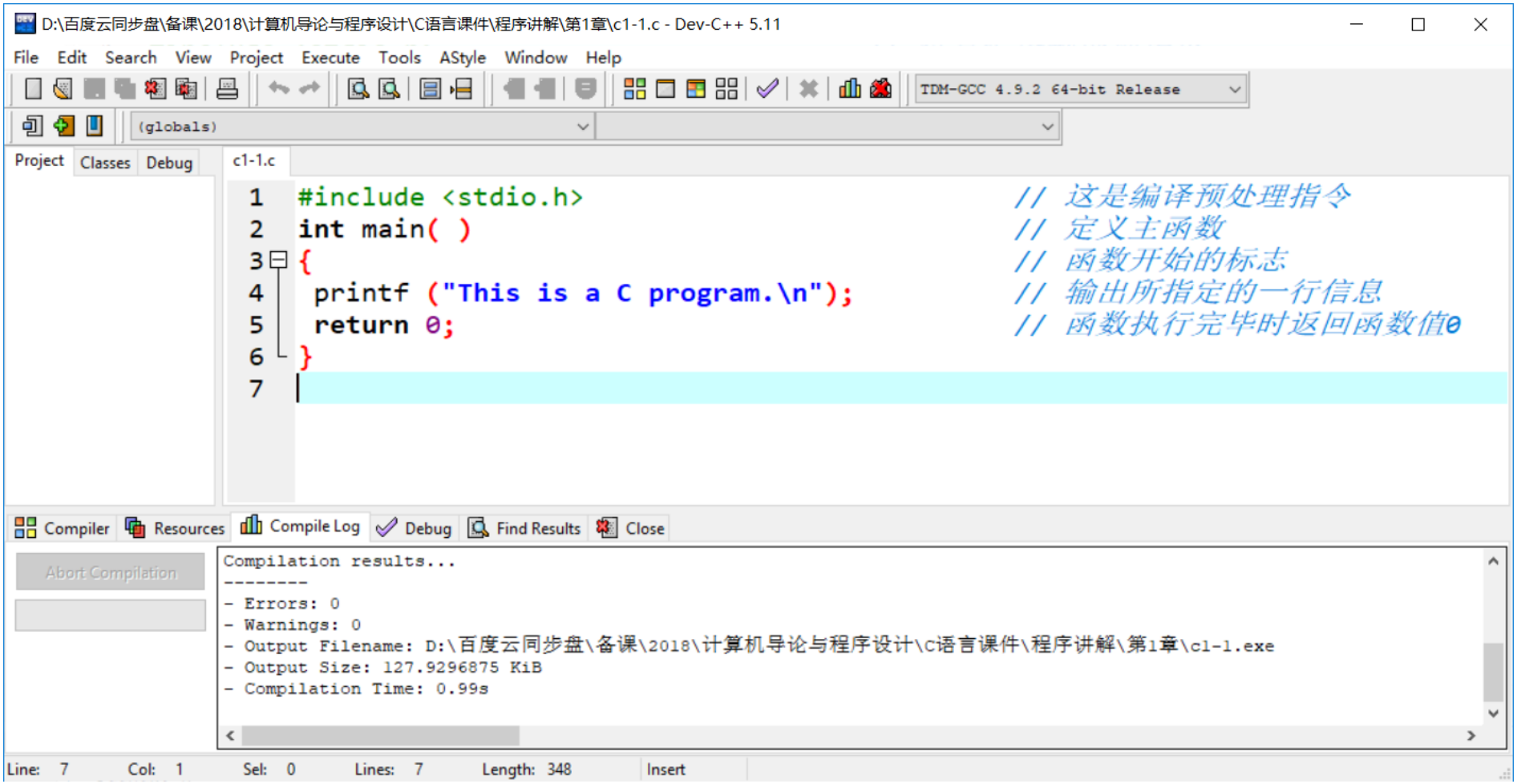
运行 C 程序的步骤与方法



集成开发环境—编译系统

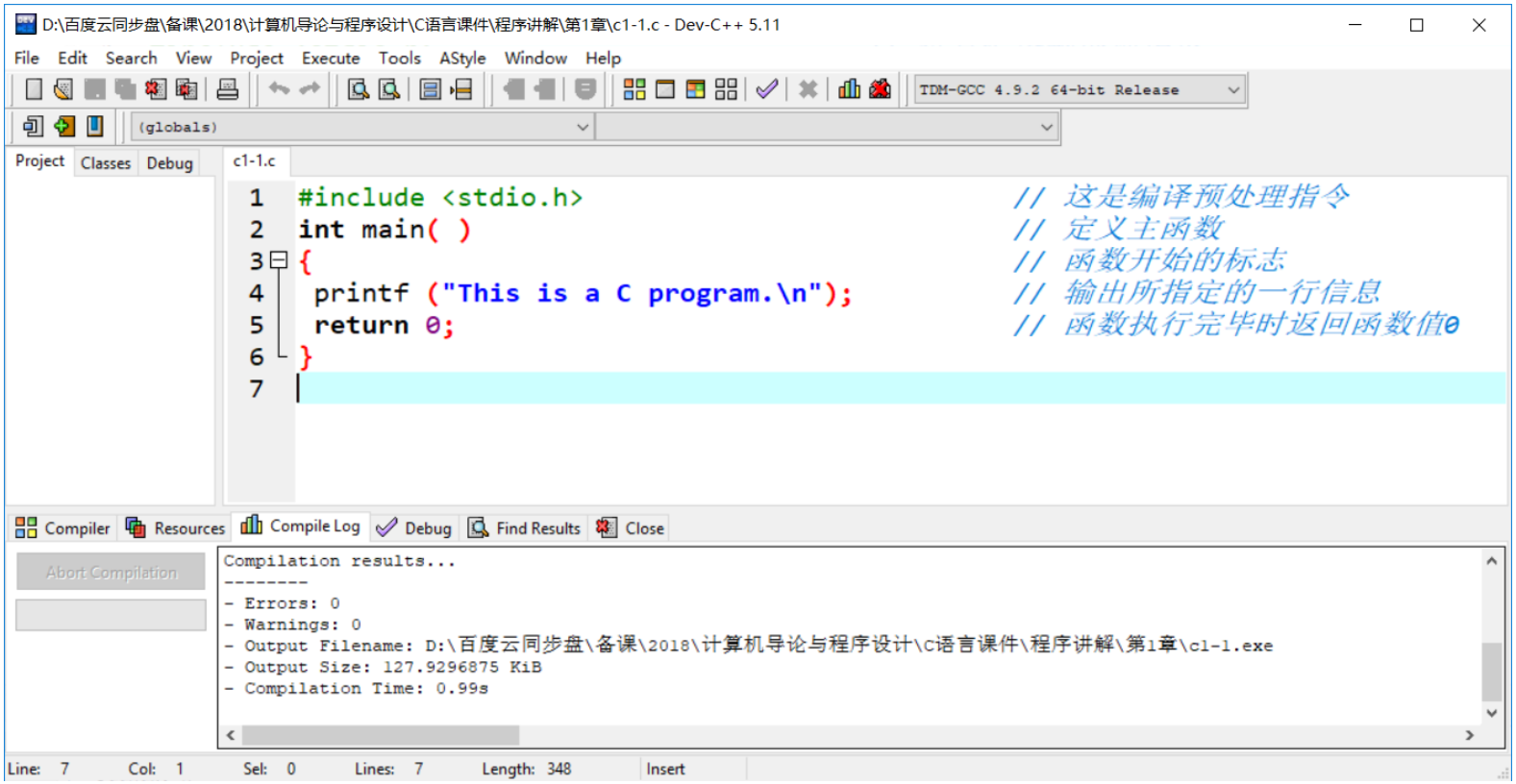
- Bloodshed Dev-C++
- Turbo C
- Visual C++6.0
- Visual Studio(VS2015, VS Community 2019 等)

Bloodshed Dev-C++ 集成开发环境



Bloodshed Dev-C++ 集成开发环境

- 选择“文件”菜单，选择“源文件”，编辑程序。
- 保存时，保存为.cpp 或.c 文件。
- 选择“编译和运行”菜单，生成.exe 文件，运行程序。



基本数据类型

- 整数`int`
- 单精度浮点数`float`
- 双精度浮点数`double`
- 字符`char`

机内用二进制表示, 不同数据类型占用存储空间大小不同。

变量在使用之前首先要定义它的数据类型

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
    int a,b; // 定义变量a, b为整型数值, 同类型变量可以在一条语句中定义。
    float f; // 定义变量f为单精度浮点数
    double d; // 定义变量d为双精度浮点数
    char c; // 定义变量c为单个英文字母
    a=10; // 变量赋值
    b=20;
    f=10.2;
    d=20.3;
    c='A'; // 字符用单引号括起来
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

标识符

标识符就是一个对象的名字。用于标识变量、符号常量、函数、数组、类型等。

以字母或下划线开始; 区分大小写; 不能使用关键字; 最好有含义。

```
#include<stdio.h>
int main()
{
    int r = 123; // 合法整型变量名
    int 3a; // 不合法的变量名
    int break; // 不合法的变量名, 因为break是关键字, 被系统使用。
    int Radius; // 变量名最好有含义
    int radius; // 与Radius是不同的变量, C语言是大小写敏感的语言
    return 0;
}
```

C 语言关键字

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

输出语句 printf(“原样输出, % 格式符”, 对应变量值);

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
    int a=10,b; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
    float f=10.2; // 定义变量f为单精度浮点数
    double d; // 定义变量d为双精度浮点数
    char c; // 定义变量c为单个英文字母
    f=10.2;
    d=20.356;
    c='A';
    printf("a=%d,b=%d,c=%c,f=%f,d=%.2lf\n",a,b,c,f,d); // %.2f, %.2lf保留
        两位小数, \n为换行符
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

变量 b 没有被赋值, 将是一个随机值。

常用格式描述符与数据类型的对应关系

格式符	对应的数据类型	备注
%d	int	
%f	float	
%c	char	
%lf	double	
%.2f	float	保留两位小数, 四舍五入。不适用于 scanf()。
%.2lf	double	保留两位小数, 四舍五入。不适用于 scanf()。
%x(%X)	int,char	十六进制显示 (大写)
%ld	long int	

详见 p73, 表 3.6

十进制与二进制

十进制: 以 10 为底的幂展开式:

$(123)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$;

自低到高各位数 (除 10 取余至商为 0): $3 = 123\%10$, $2 = 123/10\%10 = 12\%10$,

$1 = 123/10/10\%10 = 1\%10$

二进制: 以 2 为底的幂展开式:

$$(77)_{10} = (0100 \quad 1101)_2 = 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4$$
$$+ 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

自低到高各位数 (除 2 取余至商为 0): $1 = 77\%2$, $0 = 77/2\%2 = 38\%2$,

$1 = 77/2/2\%2 = 19\%2$, $1 = 77/2/2/2\%2 = 9\%2$,

$0 = 77/2/2/2/2/\%2 = 4\%2$, $0 = 77/2/2/2/2/2/\%2 = 2\%2, \ldots$

10 进制、2 进制、16 进制的幂展开式

$$(D)_{10} = D_{n-1} \times 10^{n-1} + D_{n-2} \times 10^{n-2} + \cdots + D_1 \times 10^1 + D_0 \times 10^0 + D_{-1} \times 10^{-1} + D_{-2} \times 10^{-2} + \cdots + D_{-m+1} \times 10^{-m+1} + D_{-m} \times 10^{-m}$$

$$(B)_2 = B_{n-1} \times 2^{n-1} + B_{n-2} \times 2^{n-2} + \cdots + B_1 \times 2^1 + B_0 \times 2^0 + B_{-1} \times 2^{-1} + B_{-2} \times 2^{-2} + \cdots + B_{-m+1} \times 2^{-m+1} + B_{-m} \times 2^{-m}$$

$$(H)_{16} = H_{n-1} \times 16^{n-1} + H_{n-2} \times 16^{n-2} + \cdots + H_1 \times 16^1 + H_0 \times 16^0 + H_{-1} \times 16^{-1} + H_{-2} \times 16^{-2} + \cdots + 16_{-m+1} \times 16^{-m+1} + H_{-m} \times 16^{-m}$$

进制对照表 $2^3 2^2 2^1 2^0 = 8 + 4 + 2 + 1$

十进制	二进制	十六进制	十进制	二进制	十六进制
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

十进制、二进制与十六进制数值分解举例

$$(123)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0;$$

自低到高各位数: $3 = 123\%10,\ 2 = 123/10\%10,\ 1 = 123/10/10\%10$

$$(77)_{10} = (0100 \quad 1101)_2 = 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4$$
$$+ 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$(77)_{10} = (4D)_{16} = 4 \times 16^1 + 13 \times 16^0$$

数值在计算机中的表示 (以 8bit 编码为例)

- 原码:正数的符号为 0,负数的符号为 1,其它位按一般的方法表示数的绝对值。

$$x = (+103)_{10} \qquad [x]_{原} = (01100111)_2$$

$$x = (-103)_{10} \qquad [x]_{原} = (11100111)_2$$

- 反码: 正数的反码与原码相同;负数的反码是符号位不变,其他位按位取反
- 补码: 正数的补码与其原码相同;负数的补码为其反码最末位加 1. 即,

负数补码 = 反码 $+1 = 2^n -$ 该数的绝对值, n 是编码二进制位数.

$$(77)_{10} = (0100\quad 1101)_2, \qquad (-77)_{10} = (1100\quad 1101)_2$$

$$(-77)_{补} = 2^8 - 77 = 1111\quad 1111 + 0000\quad 0001 - 0100\quad 1101$$

$$= \underbrace{1111\quad 1111 - 0100\quad 1101}_{(-77)_{反}} + 0000\quad 0001$$

$$= \underbrace{1011\quad 0010}_{(-77)_{反}} + 0000\quad 0001 = 1011\quad 0011$$

数值表示示例


+77
原码
0 1 0 0 1 1 0 1
反码
0 1 0 0 1 1 0 1
补码
0 1 0 0 1 1 0 1
-77
原码
1 1 0 0 1 1 0 1
反码
1 0 1 1 0 0 1 0
补码
1 0 1 1 0 0 1 1

机内以补码形式存储有符号数

1. 对于正数，原码 = 反码 = 补码

2. 对于负数，补码 = 反码 + 1

   反码 = 符号位不变，其他位按位取反

3. 补码是可逆的，即再对补码求补得到原码。

4. 引入补码后，使减法统一为加法。

   $(+77)_{补} + (-77)_{补} = 0100\ 1101 + 1011\ 0011 = 0000\ 0000$

   注意：由于采用 8bit 编码，从右到左的第 9 位的 1 被舍弃。

补码运算实例 (以 8bit 编码为例)

补码可逆:

$$[-25]_{原} = (1001\ 1001)_2 \quad [-25]_{反} = (1110\ 0110)_2$$

$$[-25]_{补} = [-25]_{反} + 1 = (1110\ 0110 + 1)_2 = (1110\ 0111)_2$$

$$[-25]_{原} = ([-25]_{补})_{补} = (1001\ 1000 + 1)_2 = (1001\ 1001)_2$$

减法统一为加法: $[a - b]_{补} = a_{补} + [-b]_{补}$

$$[102 - 25]_{补} = [77]_{补} = (0100\ 1101)_2 = 77$$

$$[102]_{补} + [-25]_{补} = (0110\ 0110)_2 + (1110\ 0111)_2 = (0100\ 1101)_2 = 77$$

所以, $[102 - 25]_{补} = [102]_{补} + [-25]_{补}$

同样有, $[25 - 102]_{补} = [25]_{补} + [-102]_{补}$

计算机数据存储单位

位 (bit) 是最小的存储单位，每一位存储 1 位二进制码，一个字节 (Byte) 由 8 位组成。

- 1B = 8b
- 1KB = 1024B
- 1MB = 1024KB
- 1GB = 1024MB
- 1TB = 1024GB

整型数据输出 printf("%d,%x", -25, -25);

```
#include<stdio.h> // standard input/output编译预处理指令
int main()
{
    int a=-25,b=102; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
    printf("a=%d,b=%d,a=%x,b=%x\n",a,b,a,b);
    printf("%d,%d,%x,%x\n",-25,102,-25,102);
    printf("%d,%d,%x,%x,%X\n", 77,-77,77,-77,-77);
    return 0;
}
```

```
// 编译运行, 解释输出结果。
a=-25,b=102,a=ffffffe7,b=66
-25,102,ffffffe7,66
77,-77,4d,ffffffb3,FFFFFFB3
```

$[-25]_{补} = (1110\ 0111)_2 = \text{E7H (0XE7)}$

$[-77]_{补} = (1011\ 0011)_2 = \text{B3H (0XB3)}$

$66\text{H} = 6 \times 16^1 + 6 \times 16^0 = 102$

$4\text{DH} = 4 \times 16^1 + 13 \times 16^0 = 77$

ASCII 编码表 $B_6B_5B_4B_3B_2B_1B_0$

$B_6B_5B_4$ / $B_3B_2B_1B_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	空格	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	•	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- ASCII 码连续排列

  ‘0’～‘9’, ‘A’～‘Z’, ‘a’～‘z’

- 数字 = 编码值 - ‘0’

  9=‘9’-‘0’

- 大小字符间隔:

  ‘a’ - ‘A’ = 32

  ‘a’=0110 0001=61H=0X61=97

  ’A’=0100 0001=41H=0X41=65

字符类型

```c
#include<stdio.h>
int main()
{
   char c1 = 'A', c2 = 'a', c3 = '\n'; // 字符型变量,
   // '\n': 表示特殊字符 --- 换行符
   printf("%c,%c,%d\n",c1,c2,c3); // A,a,10
   // 整型变量的整数值就是ASCII编码值
   printf("%c,0X%x,%d\n",c1,c1,c1); // A,0X41,65
   c1 = c1 + 1; // 在表达式中, char类型当作int处理
   printf("%c,0X%x,%d\n",c1,c1,c1); // B,0X42,66
   c1 = c1 + 32; // 转换为小写字母
   printf("%c,0X%x,%d\n",c1,c1,c1); // b,0X62,98
   printf("%d\n",'9'-'0'); // 数字 = 编码值- '0'
   printf("%c,%d,%c,%d\n",'A','A','a','a'); // 输出字符和相应的ASIII编码
   return 0;
}
```

常量

```
#include<stdio.h>
#define PI 3.14 // 符号常量, 注意没有分号
int main()
{
    int a = 123; // 整型常量
    float f = 12.2, f1=123E-1; // 实型常量, 123 × 10^-1
    char c1 = 'A', c2='\n'; // 字符常量, 特殊字符 --- 换行符
    char s[50] = "boy"; // 字符串常量
    printf("半径为%d的圆周长是%f\n",a,2*PI*a);
    printf("回车换行\n");
    printf("单引号\',双引号\"转义字符前缀\,\n");
    return 0;
}
```

转义字符 (特殊字符), 见 p40, 表 3.1

常量与常变量

```c
#include<stdio.h>
#define PI 3.14 // 符号常量，注意没有分号
int main()
{
    int r = 123; // 整型变量
    const int a = 425; // 常变量
    r = 100; // 合法，因为r是变量，可以随时更改它的值
    a = 100; // 不合法，因为a是常变量，不能更改
    printf("半径为%d的圆周长是%f\n",r,2*PI*r);
    return 0;
}
```

长整型、无符号整型, 浮点型数据类型

```
#include<stdio.h>
int main()
{
    int a = 123; // 整型变量
    long int b = 1E+8; // 长整型变量
    unsigned int u = 0XFF; // 无符号整型, 最高位不作为符号位处理
    float f = 10.2; // 单精度浮点数
    double d = 1E-8; // 双精度浮点数
    printf("%x,%d\n",u,u); // ff, 255
    printf("%d,%ld,%x,%f,%lf\n",a,b,u,f,d);
    // sizeof函数返回类型分配的字节数, 整型数据存储空间和值的范围见p45, 表3.2
    printf("%d,%d,%d,%d\n",sizeof(int),sizeof(float),sizeof(double),
        sizeof(long int),sizeof(long long int));
    return 0;
}
```

Recall: 常用格式描述符与数据类型的对应关系

格式符	对应的数据类型	备注
%d	int	
%f	float	
%c	char	
%lf	double	
%.2f	float	保留两位小数, 四舍五入。不适用于 scanf()。
%.2lf	double	保留两位小数, 四舍五入。不适用于 scanf()。
%x(%X)	int,char	十六进制显示 (大写)
%ld	long int	

详见 p73, 表 3.6

Recall: 输出语句 printf(“原样输出, % 格式符”, 对应变量值);

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
  int a=10,b; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
  float f=10.2; // 定义变量f为单精度浮点数
  double d; // 定义变量d为双精度浮点数
  char c; // 定义变量c为单个英文字母
  f=10.2;
  d=20.356;
  c='A';
  printf("a=%d,b=%d,c=%c,f=%f,d=%.2lf\n",a,b,c,f,d); // %.2f, %.2lf保留
      两位小数
  return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

变量 b 没有被赋值, 将是一个随机值。

输入语句 scanf(“% 变量格式符”, & 变量名);

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
  int a=10,b; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
  float f=10.2; // 定义变量f为单精度浮点数
  double d; // 定义变量d为双精度浮点数
  char c='A'; // 定义变量c为单个英文字母, 字符输入以后讲
  printf("请输入整数和浮点数, 空格隔开:\n"); // 提示语句[可选]
  scanf("%d%f",&a,&f); // 尽量简单, 不要有其它字符和'\n'
  printf("请输入两个浮点数, 空格隔开:\n"); // 提示语句[可选]
  scanf("%f%lf",&f,&d);
  printf("a=%d,b=%d,c=%c,f=%f,d=%lf\n",a,b,c,f,d); // \n为换行符
  return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

字符输出函数 putchar

```
#include<stdio.h>
int main()
{
   char a = 'B',b = 'O',c = 'Y'; //定义3个字符变量并初始化
   putchar(a); //向显示器输出字符B
   putchar(b); //向显示器输出字符O
   putchar(c); //向显示器输出字符Y
   putchar ('\n'); //向显示器输出一个换行符
   return 0;
}
```

字符输入函数 getchar, 遇到回车, 开始从缓冲区中接收字符。

```
#include<stdio.h>
int main()
{
   char a,b,c; //定义字符变量a,b,c
   a = getchar(); //从键盘输入一个字符，送给字符变量a
   b = getchar(); //从键盘输入一个字符，送给字符变量b
   c = getchar(); //从键盘输入一个字符，送给字符变量c
   putchar(a); //将变量a的值输出
   putchar(b); //将变量b的值输出
   putchar(c); //将变量c的值输出
   printf("\na=%d,b=%d,c=%d,a=%c,b=%c,c=%c\n",a,b,c,a,b,c);
   return 0;
}
```

字符输入函数 getchar, 遇到回车, 开始从缓冲区中接收字符。

```
char a,b,c; //定义字符变量a,b,c
a = getchar(); //从键盘输入一个字符，送给字符变量a
b = getchar(); //从键盘输入一个字符，送给字符变量b
c = getchar(); //从键盘输入一个字符，送给字符变量c
putchar(a); //将变量a的值输出
putchar(b); //将变量b的值输出
putchar(c); //将变量c的值输出
printf("\na=%d,b=%d,c=%d,a=%c,b=%c,c=%c\n",a,b,c,a,b,c);
```

从键盘输入 abc 回车, 观察结果, 应该是正确的结果。遇到回车, 开始从缓冲区中接收字符。

字符输入函数 getchar, 遇到回车, 开始从缓冲区中接收字符。

```
a = getchar(); //从键盘输入一个字符，送给字符变量a='a'
b = getchar(); //从键盘输入一个字符，送给字符变量b='\n'
c = getchar(); //从键盘输入一个字符，送给字符变量c='b'
putchar(a); //将变量a的值输出a
putchar(b); //将变量b的值输出\n
putchar(c); //将变量c的值输出b
printf("\na=%d,b=%d,c=%d,a=%c,b=%c,c=%c\n",a,b,c,a,b,c);
```

再运行一次程序, 输入 a 回车, 输入 b 回车, 输入 c 回车, 观察结果。

```
a
b
a
b
a=97, b=10, c=98, a=a, b=
, c=b
```

开发平台上演示讲解

在开发平台，以具体的示例，详细讲解以下内容:

- int, float, double, char 数据类型, sizeof( ) 函数
- %d, %f, %c, %lf, %x 格式符的使用 (见 ppt 中的表格)
- if( ){ }, while( ){ } 简单语句
- `char c; scanf("%c", &c);` 接收输入的字符

  `char op; int x,y; scanf("%d%c%d",&x,&op,&y);`
- `char c; c=getchar();` 接收输入的字符，`putchar(c)`输出一个字符
- 编程理解数字 ASCII 码与整数的对应关系以及大小写字符之间的关系。
- 重点理解字符缓冲区的概念，以及消费无用字符的技巧。

算术运算符 +, −, *, /, %, ++, −−

整数 = 整数/整数, 结果不会四舍五入。

```
#include<stdio.h>
int main()
{
   int a=5, b=2; float c=5,d=2,f;
   f = a/b; printf("%f\n",f); // 2.000000
   f = c/d; printf("%f\n",f); // 2.500000
   f = (float)a/b; printf("%f\n",f); // 2.500000
   printf("%f\n",5.0/2); // 2.500000
   printf("%d\n",2a); // 错误
   printf("%d\n",2*a); // 正确
   return 0;
}
```

余数 r=a%b, a,b 必须是整数。

```
#include<stdio.h>
int main()
{
    int a = 123;
    while(a != 0) // 当a不等于0时, 执行循环体。
    {
      printf("%d ",a%10); // 3 2 1
      a = a/10; // 或 a /= 10;
    }
    printf("\n");
    return 0;
}
```

算术运算符 ++, −−

++i, - -i: 先加 (减)1, 再使用。i++, i- -: 先使用, 再加 (减)1

```
#include<stdio.h>
int main()
{
    int a,b=10;
    a = ++b;
    printf("a=%d,b=%d\n",a,b); // a=11,b=11
    a = b++;
    printf("a=%d,b=%d\n",a,b); // a=11,b=12
    a--;
    b--;
    printf("a=%d,b=%d\n",a,b); // a=10,b=11
    return 0;
}
```

数学库函数, 详见附录 E p365.

```
#include<math.h>
```

函数原型	功能	调用举例
`int abs(int x);`	求整数 x 的绝对值	`int x=-10,y; y=abs(x);`
`double fabs(double x);`	求浮点数 x 的绝对值	`double x=-0.5,y; y=fabs(x);`
`double sqrt(double x);`	计算 $\sqrt{x}$	`double x=0.5,y; y=sqrt(x);`
`double pow(double x, double y);`	计算 x^y	`double x=0.5,y=2,z; z=pow(x,y);`
`int rand(void);`	产生-90~32767 的随机整数	`int y; y=rand();`
`double log(double x);`	求 $\log_e x$, 即 $\ln x$	`double x=2.0; y=log(x);`
`double log10(double x);`	求 $\log_{10} x$	`double x=2; y=log10(x);`

因为函数内是用二进制计算浮点数的指数值, 注意, 调用 pow 函数使用整型参数有精度问题, 可能得不到正确的值。以后学习函数章节后, 最好编写自己的函数计算整数的指数值。

Example (例 3.5 p64)

求 $ax^2 + bx + c = 0$ 方程的根。a, b, c 由键盘输入，设 $b^2 - 4ac > 0$。

```
#include<stdio.h>
#include<math.h> // 数学库函数
int main()
{
  double a,b,c,x1,x2,delta;
  scanf("%lf%lf%lf",&a,&b,&c);
  if(b*b-4*a*c <= 0) { printf("输入错误!"); return 0; } // 程序结束
  delta = sqrt(b*b-4*a*c);
  x1 = -b + delta/(2*a);
  x2 = -b - delta/(2*a);
  printf("x1=%lf,x2=%lf\n",x1,x2);
  return 0;
}
```

Example (例 3.1 p37)

有人用温度计测量出用华氏法表示的温度 (如 64°F), 今要求把它转换为以摄氏法表示的温度 (如 17.8°C)。

$$c = \frac{5}{9}(f - 32)$$

其中, f 代表华氏温度, c 代表摄氏温度.

特别注意: 整数/整数 = 整数, 不会四舍五入。

Example (例 3.2 p38)

计算存款利息。有 1000 元, 想存一年。有 3 种方法可选:

(1) 活期, 年利率为 r_1;

(2) 一年期定期, 年利率为 r_2;

(3) 存两次半年定期, 年利率为 r_3。

请分别计算出一年后按 3 种方法所得到的本息和。

$$p_1 = p_0(1 + r_1), p_2 = p_0(1 + r_2), p_3 = p_0(1 + \frac{r_3}{2})(1 + \frac{r_3}{2})$$

Example

1. 自定义各变量类型和值, 求 $y = |x^3 + \log_{10} x|$
2. 自定义各变量类型和值, 求 $y = \frac{3ae}{cd}$
3. 自定义各变量类型和值, 求 $y = \frac{ax + \frac{a+x}{4a}}{2}$
4. m 是一个已知 3 位整数, 从左到右用 a, b, c 表示各位数字。
   (1) 求数 bac 的值;
   (2) 计算 m 的最后一个字节;
   (3) 思考, 如果 m 是多位数, 如何计算获取每位数字?

开发平台上演示讲解

- 回顾基本数据类型`int`,`float`,`double`,

  输出输入语句`printf( );scanf();putchar(),getchar()`,

  格式转换符`%d,%f,%lf,%c,%x`, ASCII 编码与整数之间的对应关系。
- 无符号整型

```
unsigned int a=0x85; int b=-5; printf("%X,%X,%d",a,b,b); //补码存储
```

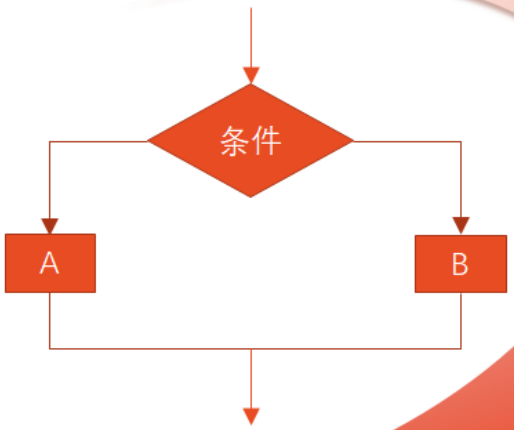
- 定义常数`#define PI 3.14`
- 标识符, 以字母或下划线开始。区分大小写, 不能使用关键字。
- 算术运算符`+,-,*,/,%,++,--`。特别注意: 整数 = 整数/整数, 不会四射五入。
- `++i, --i`: 先加 (减)1, 再使用。`i++, i--`: 先使用, 再加 (减)1
- 数学库函数`int abs(int x); double fabs(double x); double sqrt(double x); doubel pow(double x,double y); double log10(double x)`
- 作业: 练习编程: 上页 Example, 下节课检查。

Part II

选择结构程序设计

Outlines

选择结构和条件判断



C 语言有两种选择语句

- if 语句，用来实现两个分支的选择结构
- switch 语句，用来实现多分支的选择结构

if(条件表达式){表达式为真(非0)时执行语句;}

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
  int a=10; // 定义变量a为整型数值, 定义变量时, 可以指定变量的初值
  if(a>=10)
  {
    printf("a>=10\n"); // \n为换行符
  }
  else
  {
    printf("a<10\n"); // \n为换行符
  }
  return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

[例 4.1 p84] 求 $ax^2 + bx + c = 0$ 方程的根。a, b, c 由键盘输入。

```
#include<stdio.h>
#include<math.h> // 数学库函数
int main()
{
  double a,b,c,x1,x2,delta;
  scanf("%lf%lf%lf",&a,&b,&c);
  if(b*b-4*a*c < 0)
  { printf("This equation hasn\'t real roots!\n"); }
  else
  {
    delta = sqrt(b*b-4*a*c);
    x1 = (-b + delta)/(2*a); x2 = (-b - delta)/(2*a);
    printf("x1=%.2lf,x2=%.2lf\n",x1,x2);
  }
  return 0;
}
```

[例 4.2 p85] 输入两个实数, 按由小到大的顺序输出这两个数。

```
#include<stdio.h>
int main()
{
   float a,b,t;
   scanf("%f%f",&a,&b);
   //不好: scanf("%f,%f",&a,&b);
   // 假定a<=b, 否则交换a,b的值
   if(a>b)
   { // 将a和b的值互换
      t=a; a=b; b=t;
   }
   printf("%.2f,%.2f\n",a,b);
   return 0;
}
```

两变量值互换基本技巧

//把变量 b 的值赋给变量 a, a 的值等于 b 的值

a=b;

//再把变量 a 的值赋给变量 b, 变量 b 值没有改变

b=a;

因此, 为了实现互换, 必须借助于第三个变量。

[例 4.3 p86] 输入 3 个数 a, b, c, 要求按由小到大的顺序输出。

```
#include<stdio.h>
int main()
{
   float a,b,c,t;
   scanf("%f%f%f",&a,&b,&c); //不好: scanf("%f,%f,%f",&a,&b,&c);
   if(a>b) // 假定a<=b<=c, 否则交换。
   {
     t=a; a=b; b=t; //借助变量t, 实现变量a和变量b互换值
   } //互换后, a小于或等于b
   if(a>c)
   {
     t=a; a=c; c=t; //借助变量t, 实现变量a和变量c互换值
   } //互换后, a小于或等于c
   if(b>c) //还要
   {
     t=b; b=c; c=t; //借助变量t, 实现变量b和变量c互换值
   } //互换后, b小于或等于c
   printf("%.2f,%.2f,%.2f\n",a,b,c); //顺序输出a,b,c的值
   return 0;
}
```

if 语句的一般形式

条件表达式 ⇒ 关系表达式; 逻辑表达式; 数值表达式。

形式 1(无 else)

```
// 形式1(无else)
if(条件表达式)
{
   多条语句(复合语句);
}
```

形式 2

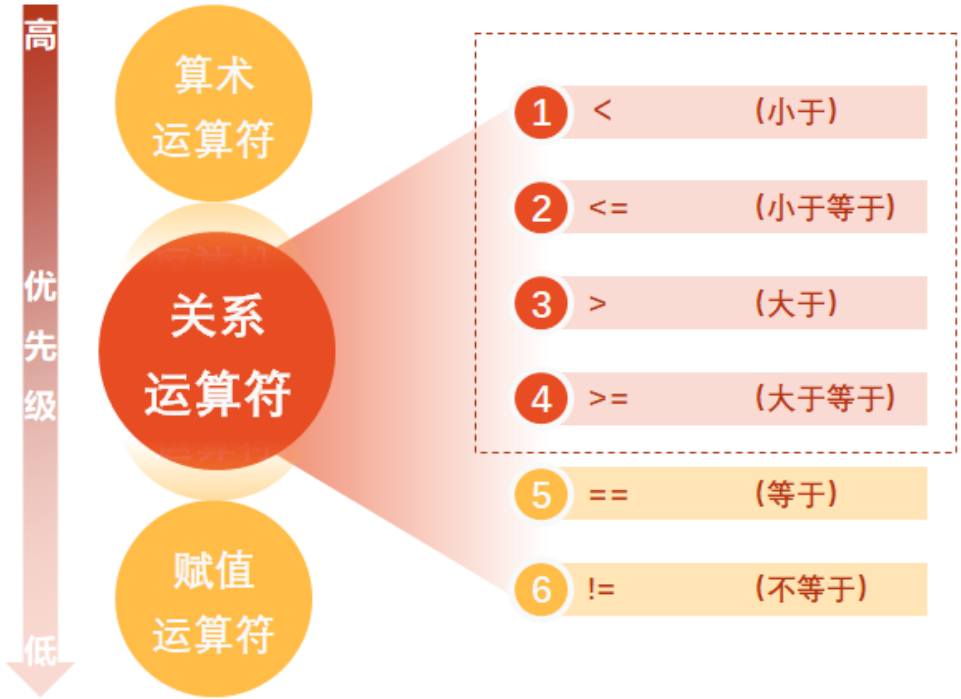
```
// 形式2
if(条件表达式)
{
   多条语句(复合语句);
}
else
{
   多条语句(复合语句);
}
```

形式 3(排除式)

```
// 形式3(排除式)
if(条件表达式1)
{
   多条语句(复合语句);
}
else if(条件表达式2)//可多个
{
   多条语句(复合语句);
}
else
{
   多条语句(复合语句);
}
```

关系运算符及其优先次序

```
int a=5,b=10,c=20; //以int为例
if(a<b+c) // 相当于a<(b+c)
    { a小于(b+c)时执行 }
if(a<=b+c)
    { a小于等于(b+c)时执行 }
if(a>b+c)
    { a大于(b+c)时执行 }
if(a>=b+c)
    { a大于等于(b+c)时执行 }
if(a==b+c) // 与a=(b+c)不同
    { a等于(b+c)时执行 }
if(a!=b+c)
    { a不等于(b+c)时执行 }
```



分析:

```
if(a>b==c) // 相当于(a>b)==c
if(a=b>c) // 相当于a=(b>c),避免==写为=
```

关系表达式的值, 非 0 即真

关系表达式

- 用关系运算符将两个数值或数值表达式连接起来的式子, 称为关系表达式。
- 关系表达式的值是一个逻辑值, 即 “真” 或 “假”。
- 在 C 的逻辑运算中, 以 “1” 表示 “非 0”, 代表 “真”; 以 “0” 代表 “假”。

```
int a=3, b=2, c=1, d1, d2;
d1 = a>b; // d1=(a>b), d1=1
d2 = a>b>c;//d2=(a>b>c),d2=0
if(d1)// d1=1,非0即真
   { printf("执行此语句"); }
if(d2) // d2=0,条件为假
   { printf("不执行此语句"); }
```

```
if(d1 = a>b) //d1=(a>b),赋值后d1=1,非0即真
   { printf("执行此语句"); }
if(d2 = a>b>c)//d2=(a>b>c),赋值后d2=0,即假
   { printf("不执行此语句"); }
if(a>b) //表达式=1,非0即真
   { printf("执行此语句"); }
if(a>b>c),表达式值=0,即假
   { printf("不执行此语句"); }
```

逻辑运算符 (与 &&, 或 ||, 非!)

```
int a=5,b=10,c=0; //以int为例
if(!a) // 逻辑非(NOT), a是非0, 所以!a的值是0
   { ... }
if(a && b) // 逻辑与(AND), a,b均为非0, 所以(a && b)的值为1
   { ... }
if(a || c) // 逻辑或(OR), a,c之一是非0, 即为真
   { ... }
```

逻辑运算符真值表, 非 0 即真

a	b	!a	!b	a&&b	a||b
真 (非 0)	真 (非 0)	假 (0)	假 (0)	真 (1)	真 (1)
真 (非 0)	假 (0)	假 (0)	真 (1)	假 (0)	真 (1)
假 (0)	真 (非 0)	真 (1)	假 (0)	假 (0)	真 (1)
假 (0)	假 (0)	真 (1)	真 (1)	假 (0)	假 (0)

- “&&” 和 “||” 是双目运算符, 要求有两个运算对象 (操作数);
  “!” 是单目运算符, 只要有一个运算对象
- 由高到低优先次序: !(非)→&&(与)→||(或);
  逻辑运算符中的 “&&” 和 “||” 低于关系运算符, “!” 高于算术运算符
- 逻辑运算结果不是 0 就是 1, 不可能是其他数值。
  而运算对象可以是 0(假) 或任何非 0 的数值 (按 “真” 对待)

逻辑运算示例 (1)

判别用 year 表示的某一年是否闰年。闰年的条件是符合下面二者之一: (1) 能被 4 整除,但不能被 100 整除。(2)能被 100 整除,又能被 400 整除。

```
int year;
scanf("%d",&year);
// 闰年
if(year%4 == 0 && year%100 != 0)
{ printf("%d是闰年\n", year); }
else if(year%100 == 0 && year%400 == 0)
{ printf("%d是闰年\n", year); }
else
{ printf("%d不是闰年\n", year); }
```

逻辑运算示例 (2), 使用标志变量是基本技巧

判别用 year 表示的某一年是否闰年。闰年的条件是符合下面二者之一: (1) 能被 4 整除,但不能被 100 整除。(2)能被 100 整除,又能被 400 整除。

```
int year, flag = 'N'; // flag称为标志变量, 等效设置 char flag = 'N';
scanf("%d",&year);
// 闰年
if(year%4 == 0 && year%100 != 0)
   { flag = 'Y'; }
else
{
  if(year%100 == 0 && year%400 == 0)
      { flag = 'Y'; }
}
if(flag == 'Y')
   { printf("%d是闰年\n", year); }
else
   { printf("%d不是闰年\n", year); }
```

逻辑运算示例 (3), 善用 &&,||,! 组成逻辑表达式, 简化程序逻辑表达

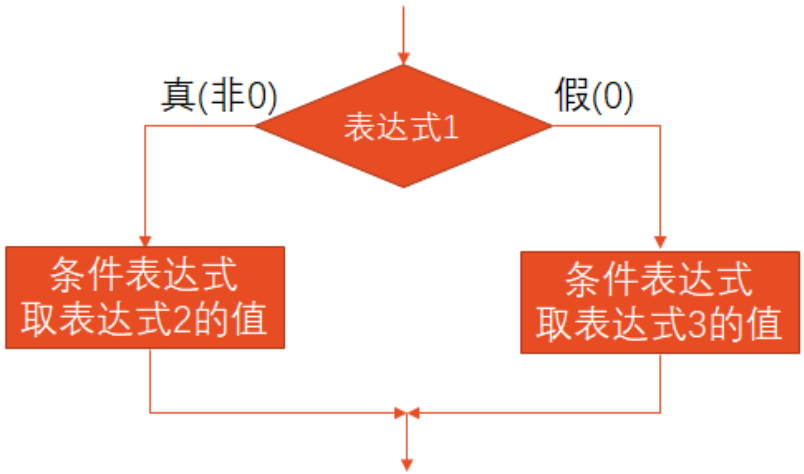
判别用 year 表示的某一年是否闰年, 可以用一个逻辑表达式来表示。闰年的条件是符合下面二者之一: (1) 能被 4 整除, 但不能被 100 整除。(2)能被 100 整除, 又能被 400 整除。

```
int year;
scanf("%d",&year);
// 闰年
if((year%4 == 0 && year%100 != 0)||(year%100 == 0 && year%400 == 0))
{ printf("%d是闰年\n", year); }
else
{ printf("%d不是闰年\n", year); }
```

条件运算符和条件表达式

```
int a,b,max;
scanf("%d%d",&a,&b);
if(a>b)
{ max = a; }
else
{ max = b; }
// 等效于条件表达式
max = (a>b) ? a : b;
// 或
a>b ? (max=a) : (max=b);
// 甚至可用在语句中
printf("%d\n",a>b ? a : b);
```

表达式1？表达式2：表达式3



例: 大写转小写字母

[例 4.4, p96] 输入一个字符, 判别它是否为大写字母, 如果是, 将它转换成小写字母; 如果不是, 不转换。然后输出最后得到的字符。

```
char ch;
scanf("%c",&ch);
ch = (ch>='A' && ch<='Z') ? (ch+32) : ch;
// 等效于
if(ch>='A' && ch<='Z')
{
   ch = ch+32; // 可简写为 ch += 32;
}
printf("ch=%c\n",ch);
```

特别注意: 数学表达式与C语言表达式的不同

```
int a = 100;
if(20 <= a && a <= 30)//表达式的值为假(0), 条件表达式与数学含义(20 ≤ a ≤ 30)相同
// 或 if(a>=20 && a<=30)
{ ... }
if(20 <= a <= 30)//(20<=a)<=30, 表达式为真(1), 与数学含义(20 ≤ a ≤ 30)不同
{ ... }
// 类似的
// if(a==20) 与 if(a=20)意义不同
if(a==20) //表达式的值是假(0), a的值没有变化
{ ... }
if(a=20) //表达式的值是20, 非0, 表示为真, 并且a被赋值为20(赋值语句)
{
   printf("%d\n",a); // 20
}
printf("%d\n",a); // 20
```

用 switch(int 或 char 型表达式), 实现多分支选择结构

```
int a;
scanf("%d",&a)
switch(a) // a必须是int或char类型
{
    case 10: 多条语句1;
        //break表示该条件执行完毕
        break;
    case 20: 多条语句2;
        break;
    case 30: 多条语句3;
        break;
    default:
        // 以上不满足时执行
        多条语句4;
}
```

```
// 等效于
int a;
scanf("%d",&a)
if(a == 10)
    { 多条语句1; }
else if(a == 20 )
    { 多条语句2; }
else if(a == 30)
    { 多条语句3; }
else
    { 多条语句4; }
```

无 break, 直接往下执行

```
char a;
scanf("%c",&a)
swach(a) // a必须是int或char类型
{
    case 'A':
    case 'a': 多条语句1;
        break;
    case 'B':
    case 'b': 多条语句2;
        break;
    case 'C':
    case 'c': 多条语句3;
        break;
    default: 多条语句4;
}
```

```
// 等效于
char a;
scanf("%d",&a)
if(a == 'A' || a == 'a')
    { 多条语句1; }
else if(a == 'B' || a == 'b')
    { 多条语句2; }
else if(a == 'C' || a == 'c')
    { 多条语句3; }
else
    { 多条语句4; }
```

[例 4.10,p99] 运输公司对用户计算运输费用。路程越远,运费越低。标准如下:

$s < 250$	没有折扣
$250 \leq s < 500$	2% 折扣
$500 \leq s < 1000$	5% 折扣
$1000 \leq s < 2000$	8% 折扣
$2000 \leq s < 3000$	10% 折扣
$3000 \leq s$	15% 折扣

```
int c,s; //c是分类整数, s是距离
float p,w,d,f; //单价,重量,折扣,运费
// 运费 f = p*w*s*(1-d%)
scanf("%f%f%d",&p,&w,&s);
if(s>=3000) { c = 12; }
else {c = s/250; }
switch(c) {
  case 0: d=0; break;
  case 1: d=2; break;
  case 2: case 3: d=5; break;
  case 4: case 5: case 6: case 7:
      d=8; break;
  case 8: case 9: case 10: case 11:
      d=10 break;
  case 12: d=15; break;
}
f = p*w*s*(1-d/100);
printf("%.2f\n",f);
```

给出一个百分制的成绩,要求输出成绩等级'A','B','C','D','E'。90分以上为'A',80～89分为'B',70～79分为'C',60～69分为'D',60分以下为'E'。

```
int grade;
scanf("%d",&grade);
grade /= 10; // 等效于 grade=grade/10;
switch(grade)
{
   case 0: case 1: case 2: case 3: case 4:
   case 5: printf("E"); break;
   case 6: printf("D"); break;
   case 7: printf("C"); break;
   case 8: printf("B"); break;
   case 9:
   case 10: printf("A"); break;
}
```

思考:如果输入成绩等级,输出分数段,如何修改程序?

给出一个百分制的成绩,要求输出成绩等级'A','B','C','D','E'。90分以上为'A',80～89分为'B',70～79分为'C',60～69分为'D',60分以下为'E'。

```
int grade;
scanf("%d",&grade);
grade /= 10; // 等效于 grade=grade/10;
switch(grade)
{
   case 0: case 1: case 2: case 3: case 4:
   case 5: printf("E"); break;
   case 6: printf("D"); break;
   case 7: printf("C"); break;
   case 8: printf("B"); break;
   case 9:
   case 10: printf("A"); break;
}
```

思考: 如果输入成绩等级,输出分数段,如何修改程序?

Part III

循环结构程序设计

Outlines

为什么需要循环控制

- 要向计算机输入全班 50 个学生的成绩;(重复 50 次相同的输入操作)
- 分别统计全班 50 个学生的平均成绩; (重复 50 次相同的计算操作)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
// 输入第1个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第1个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver); // 输出第1个学生平均成绩
// 输入第2个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第2个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver);// 输出第2个学生平均成绩
...
```

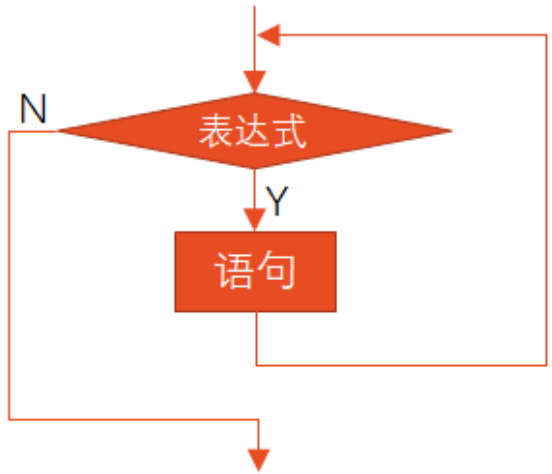
用循环控制处理重复操作

- 要向计算机输入全班 50 个学生的成绩;(重复 50 次相同的输入操作)
- 分别统计全班 50 个学生的平均成绩; (重复 50 次相同的计算操作)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
int i=1; // 设整型变量i初值为1
while( i<=50 ) // 当i的值小于或等于50时执行花括号内的语句
{
   scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
   aver=(score1+score2+score3+score4+score5)/5;
   printf("aver=%7.2f",aver);
   i++; // 每执行完一次循环使i的值加1
}
```

while(表达式){···}

```
while(表达式)
{
    // 循环体
    执行多条语句;
}
```



while 循环特点

每轮循环: 首先判断表达式的值, 若“真” (表达式值为非 0) 时, 就执行循环体语句; 为“假” (表达式值为 0) 时, 就不执行循环体语句。

常见错误

```
while(表达式);
{
    // 循环体
    执行多条语句;
}
```

变体

```
while(1)
{
  if(表达式) break; // 退出循环
  执行多条语句;
}
```

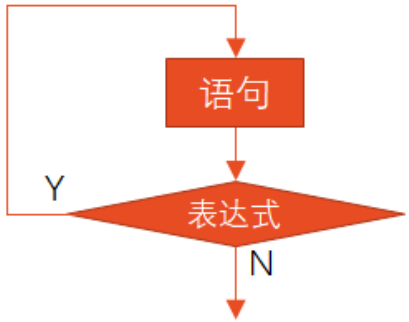
[例 5.1] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$

```
int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
while(i <= 100) //当i>100, 条件表达式的值为假(0), 不执行循环体
{ //循环体开始
   sum=sum+i; //第1次累加后, sum的值为1
   i++; //加完后, i的值加1, 为下轮累加做准备
} //循环体结束
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 循环体如果包含一个以上的语句,必须用花括号括起来,作为复合语句出现。
2. 不要忘记给 i 和 sum 赋初值,否则它们的值是不可预测的,结果显然不正确。
3. 在循环体中应有使循环趋向于结束的语句。如本例中的 $i++;$ 语句。如果无此语句,则 i 的值始终不改变,循环永远不结束。

do{···}while(表达式);

```
do
{
  // 循环体
  执行多条语句;
} while(表达式);
```



循环特点

首先无条件地执行循环体,然后判断循环条件是否成立。

易犯错误

```
do
{
  // 循环体
  执行多条语句;
} while(表达式)
```

变体

```
do
{
  执行多条语句;
  if(!表达式) break; // 退出循环
} while(1);
```

[例 5.2] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$

```
int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
do
{ //循环体开始
   sum=sum+i; //第1次累加后，sum的值为1
   i++; //加完后，i的值加1，为下次累加做准备
}while(i <= 100); //当i>100，条件表达式i<=100的值为假，不执行循环体
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 在一般情况下,用 while(){···} 语句和用 do{···}while(); 语句处理同一问题时,若二者的循环体部分是一样的,那么结果也一样。
2. 但是如果 while 后面的表达式一开始就为假 (0 值) 时,两种循环的结果是不同的。

while(表达式){···} 与 do{···}while(表达式);

[例 5.3] 求 $\sum_{i=n}^{100} i$。考虑输入 $n > 100$ 时的情况，以下程序的不同。

```
int i,sum=0;
scanf("%d",&i);
while(i <= 100)
{
  sum=sum+i;
  i++;
}
printf("sum=%d\n",sum);
```

```
int i,sum=0;
scanf("%d",&i);
do
{
  sum=sum+i;
  i++;
}while(i <= 100);
printf("sum=%d\n",sum);
```

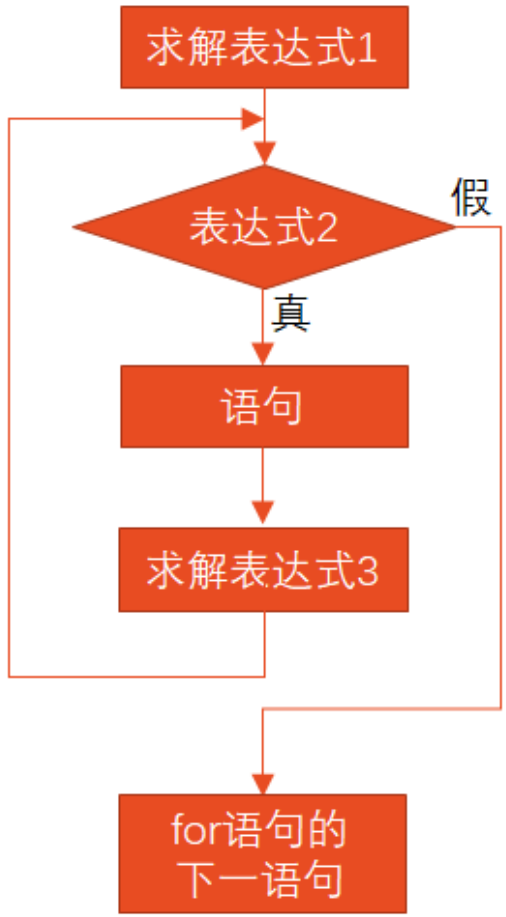
如果输入 101，左边程序输出：sum=0；右边输出：sum=101.

for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2;表达式3)
{
   // 循环体
   执行多条语句;
}
```

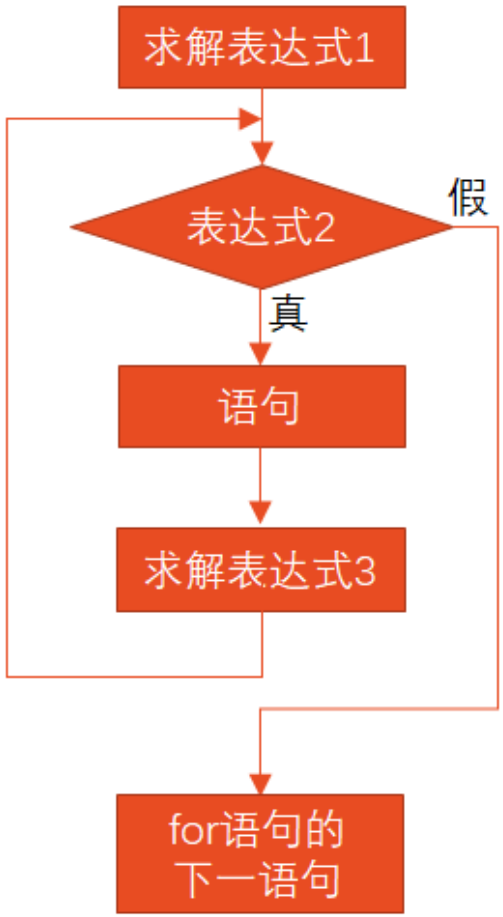
≡

```
表达式1;
while(表达式2)
{
   // 循环体
   执行多条语句;
   表达式3;
}
```



for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2
      ;表达式3)
{
   // 循环体
   执行多条语句;
}
```



- 表达式 1: 设置初始条件,只执行一次。可以为零个、一个或多个变量 (逗号隔开) 设置初值。
- 表达式 2: 是循环条件表达式,用来判定是否继续循环。在每次执行循环体前先执行此表达式 (包括第 1 次循环),决定是否继续执行循环。
- 表达式 3: 作为循环的调整,例如使循环变量增值,它是在执行完循环体后才进行的。

for(表达式 1; 表达式 2; 表达式 3){···}, 省略表达式

```
int i;
for(i=0;i<=100;i++)
{
  printf("%d\n",i);
}
```

```
int i=0;
for(;i<=100;i++)
{
  printf("%d\n",i);
}
```

```
int i=0;
for(i=0;;i++)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
}
```

```
int i;
for(i=0;i<=100;)
{
  printf("%d\n",i);
  i++;
}
```

```
int i=0;
for(;i<=100;)
{
  printf("%d\n",i);
  i++;
}
```

```
int i=0;
for(;;)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
  i++;
}
```

循环的嵌套

01

```
while()
{    ⋮
     while()
     {···}
}
```

内层循环

02

```
do
{    ⋮
     do
     {···}
     while();
}while();
```

内层循环

03

```
for( ; ; )
{
     for( ; ; )
     {···}
}
```

内层循环

04

```
while()
{    ⋮
     do
     {···}
     while();
     ⋮
}
```

内层循环

05

```
for( ; ; )
{    ⋮
     while()
     {···}
     ⋮
}
```

内层循环

06

```
do
{    ⋮
     for( ; ; )
     {···}
}while();
```

内层循环

几种循环的比较

1. 3 种循环都可以用来处理同一问题，一般情况下它们可以互相代替。
2. 在 while 循环和 do…while 循环中，只在 while 后面的括号内指定循环条件，因此为了使循环能正常结束，应在循环体中包含使循环趋于结束的语句(如 i++ 等)。
3. for 循环可以在表达式 3 中包含使循环趋于结束的操作，甚至可以将循环体中的操作全部放到表达式 3 中 (逗号隔开)。因此 for 语句的功能更强，凡用 while 循环能完成的，用 for 循环都能实现。
4. 用 while 和 do…while 循环时，循环变量初始化的操作应在 while 和 do…while 语句之前完成。而 for 语句可以在表达式 1 中实现循环变量的初始化。
5. while 循环、do…while 循环和 for 循环都可以用 break 语句跳出循环，用 continue 语句结束本次循环。

while 循环中，用 break,continue 改变循环执行的状态


```c
while(表达式)
{
    printf("语句1\n");
    //提前终止循环
    if(条件表达式) break;
    printf("语句2\n");
}
```

```c
while(表达式)
{
    printf("语句1\n");
    //结束本轮循环，进入下轮循环
    if(条件表达式) continue;
    printf("语句2\n");
}
```

while 循环中, 体会 break 与 continue 的不同

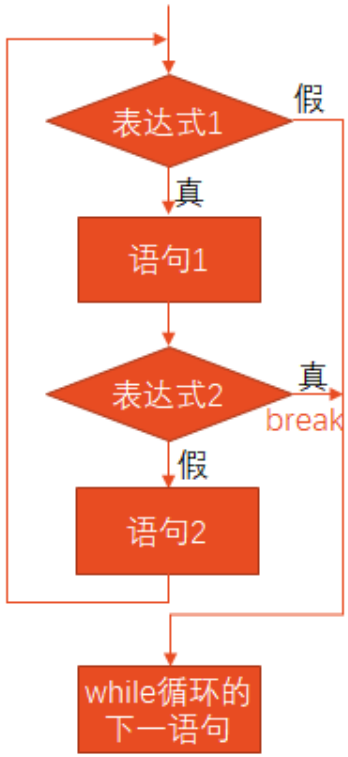
```
int i=0;
while(i<4)
{
    if(i==2)
    {
        i++;
        //终止本轮循环,开始下轮循环
        continue;
    }
    printf("%d,",i);//0,1,3,
    i++;
}
printf("\n%d\n",i);//4
```

```
int i=0;
while(i<4)
{
    if(i==2)
    {
        i++;
        // 终止整个循环
        break;
    }
    printf("%d,",i);//0,1,
    i++;
}
printf("\n%d\n",i);//3
```

while 循环中,break 语句和 continue 语句的区别

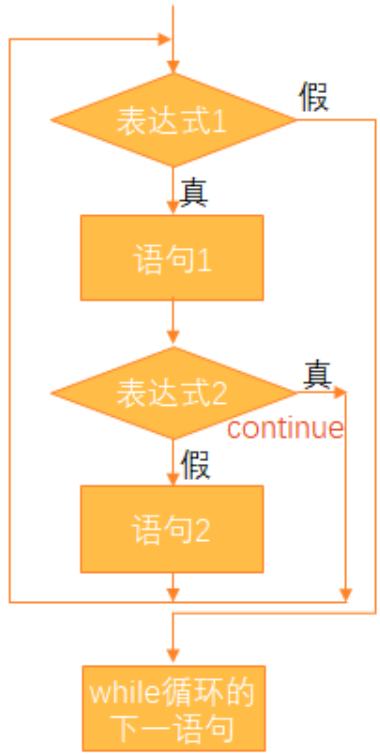
break;

```
while(表达式1)
{
        语句1;
        if(表达式2) break;
        语句2;
}
```



continue;

```
while(表达式1)
{
        语句1;
        if(表达式2) continue;
        语句2;
}
```

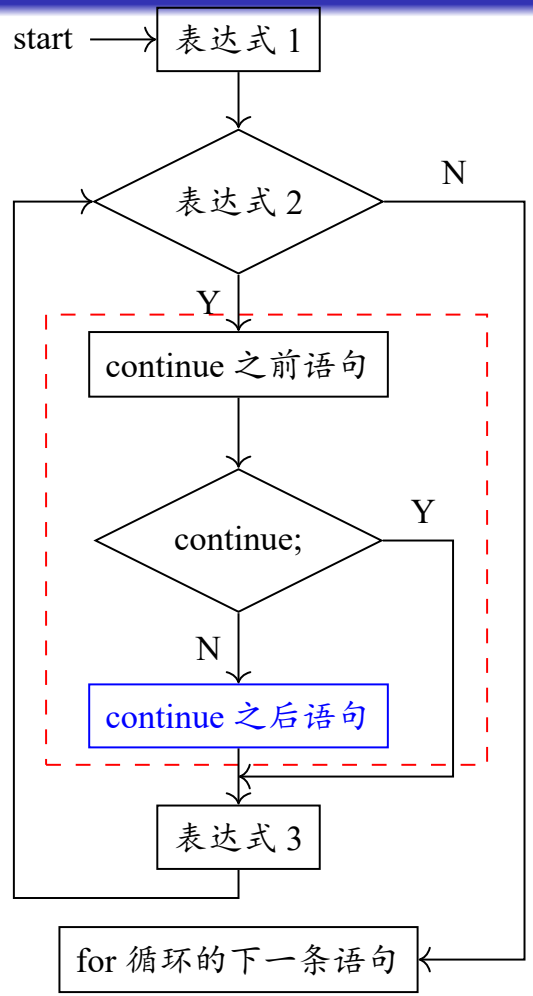


注意: continue 语句只结束本次循环,而非终止整个循环。break 语句结束整个循环,不再判断执行循环的条件是否成立。

注意: for 循环的表达式 3 被视为下轮循环的首语句

[例 5.4] 要求输出 100～200 之间的不能被 3 整除的数。

```
#include <stdio.h>
int main()
{
    int n;
    for (n=100;n<=200;n++)//(表达式1;表达式2;表达式3)
    {
        //终止本轮循环,但会执行for语句的表达式3(n++),开始下
            轮循环。表达式3被视为下轮循环的首语句。
        if (n%3==0) continue;
        printf("%d ",n);
    }
    printf("\n");
    return 0;
}
```



for 循环中, break 语句和 continue 语句的区别

[例 5.5] 输出以下 4 × 5 的矩阵。

```
int i,j,n=0; //n: 累计输出数据个数
for(i=1;i<=4;i++) // 行
{
  for(j=1;j<=5;j++,n++) //列
  {
    if(n%5==0) printf("\n");//5数换行
    if (i==3 && j==1) break;
    printf("%d\t",i*j);
  }
}
printf("\n");
```

```
int i,j,n=0; //n: 累计输出数据个数
for(i=1;i<=4;i++) // 行
{
   for(j=1;j<=5;j++,n++) // 列
   {
     if(n%5==0) printf("\n");//5数换行
     if (i==3 && j==1) continue;
     printf("%d\t",i*j); // \t就是Tab
          键，是特殊字符，表示多个空格
   }
}
printf("\n");
```

for 循环中, 体会 break 与 continue 的不同

```
int i=0;
for(i=0;i<4;i++)
{
  if(i==2) continue; //终止本轮循环,但会
       执行for语句的表达式3 (i++), 开始下
       轮循环
  printf("i=%d,",i); //i=0,i=1,i=3,
}
printf("\nend i=%d\n",i);
//end i=4
```

```
int i=0;
for(i=0;i<4;i++)
{
  if(i==2) break; //终止整个循环,也不
       会执行for语句的表达式3 (i++)
  printf("i=%d,",i); // i=0,i=1,
}
printf("\nend i=%d\n",i);
//end i=2, 此处根据循环变量i的值可以判
     断上述循环是否正常结束, if(i==4)
     正常结束。
```

for 循环中，用 break 语句提前终止循环示例

[例 5.4] 在全系 1000 名学生中举行慈善募捐，当总数达到 10 万元时就结束，统计此时捐款的人数以及平均每人捐款的数目。

```
#define SUM 100000 //指定符号常量SUM代表10万
float amount,aver,total; // 单人捐款数，平均捐款，捐款总量
int i; // 捐款人数
for (i=1,total=0; i<=1000; i++)// 表达式1给多个变量赋初值，用逗号隔开。
{
   printf("please enter amount:");
   scanf("%f",&amount);
   total = total + amount;
   if(total >= SUM) break; // 终止整个循环，也不会执行for语句的表达式3 (i++)
}
i=(i<=1000)?i:(i-1); // 技巧：判断循环是否提前终止
aver=total/i;
printf("num=%d\naver=%10.2f\n",i,aver);
```

注意：break 语句只能用于循环语句和 switch 语句之中，而不能单独使用。

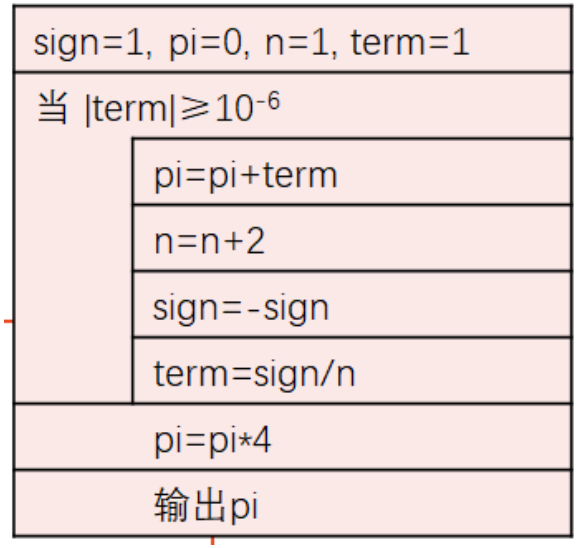
注意事项小结

1. while( ){ }; do { } while( ); for(;;){ } 执行顺序;
2. 循环变量的开始和结束条件;
3. 循环体是复合语句时,必须用 { } 扩起来;
4. 必要时,用 break 结束整个循环,用 continue 结束本轮循环;
5. 关键是找出循环规律,必要时设计流程图,指导代码实现。

[例 5.7] 用公式 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$ 求 π 的近似值, 直到发现某一项的绝对值小于 10^{-6} 为止 (该项不累加)。

解题思路: 找规律

1. 每项的分子都是 1。
2. 后一项的分母是前一项的分母加 2。
3. 第 1 项的符号为正, 从第 2 项起, 每一项的符号与前一项的符号相反。在每求出一项后, 检查它的绝对值是否大于或等于 10^{-6}。



```c
#include <stdio.h>
#include <math.h> //程序中用到数学函数fabs，应包含头文件math.h
int main()
{
  int sign=1; //sign用来表示数值的符号
  double pi=0.0,n=1.0,term=1.0; //pi开始代表多项式的值，最后代表π的值，n代表分
      母，term代表当前项的值
  while(fabs(term)>=1e-6) //检查当前项term的绝对值是否大于或等于10^-6
  {
    pi=pi+term; //把当前项term累加到pi中
    n=n+2; //n+2是下一项的分母
    sign=-sign; //sign代表符号，下一项的符号与上一项符号相反
    term=sign/n; //求出下一项的值term
  }
  pi=pi*4; //多项式的和pi乘以4，才是π的近似值
  printf("pi=%10.8f\n",pi); //输出π的近似值
  return 0;
}
```

[例 5.8] 求 Fibonacci(斐波那契) 数列的前 40 个数。这个数列有如下特点：第 1, 2 两个数为 1, 1。从第 3 个数开始，该数是其前面两个数之和。即该数列为 1,1,2,3,5,8,13,…，用数学方式表示为：

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n \geq 3) \end{cases}$$

这是一个有趣的古典数学问题：有一对兔子，从出生后第 3 个月起每个月都生一对兔子。小兔子长到第 3 个月后每个月又生一对兔子。假设所有兔子都不死，问每个月的兔子总数为多少？

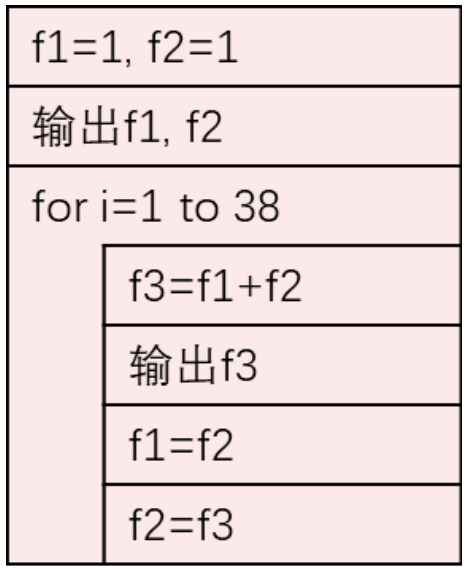
兔子繁殖的规律

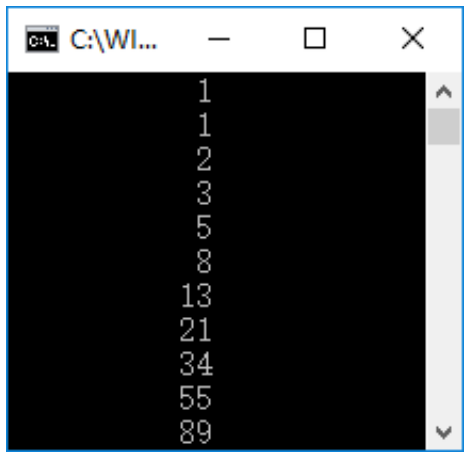
月数	小兔子对数	中兔子对数	老兔子对数	兔子总对数
1	1	0	0	1
2	0	1	0	1
3	1	0	1	2
4	1	1	1	3
5	2	1	2	5
6	3	2	3	8
7	5	3	5	13
⋮	⋮	⋮	⋮	⋮

不满 1 个月的为小兔子，满 1 个月不满 2 个月的为中兔子，满 2 个月以上的为老兔子。

解法一: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_3 = F_1 + F_2; F_1 = F_2; F_2 = F_3;$

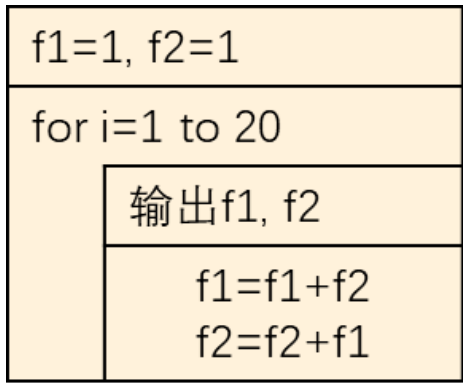
```
#include <stdio.h>
int main()
{
    int f1=1,f2=1,f3;
    int i;
    printf("%12d\n%12d\n",f1,f2);
    for(i=1; i<=38; i++)
    {
        f3=f1+f2;
        printf("%12d\n",f3);
        f1=f2;
        f2=f3;
    }
    return 0;
}
```

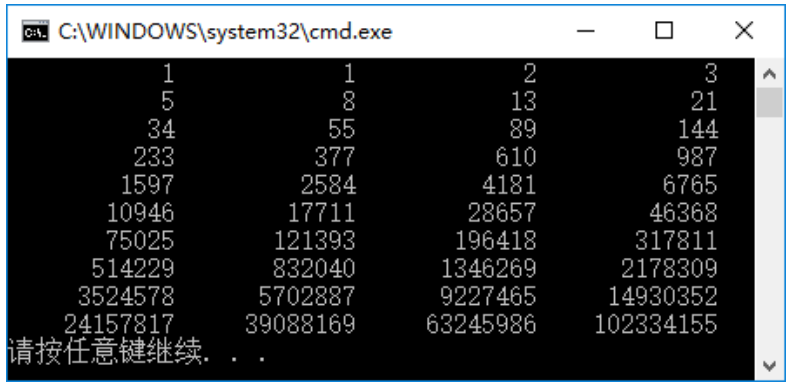




解法二: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_1 = F_1 + F_2; F_2 = F_1 + F_2;$

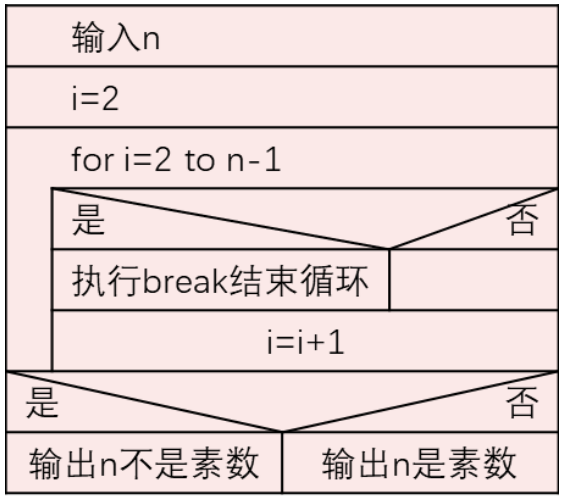
```c
#include <stdio.h>
int main()
{
  int f1=1,f2=1;
  int i;
  for(i=1; i<=20; i++)
  {
    printf("%12d%12d",f1,f2);
    if(i%2==0) // 等效 if(!(i%2))
       printf("\n");
    f1=f1+f2;
    f2=f2+f1;
  }
  return 0;
}
```





[例 5.9] 输入一个大于 3 的整数 n,判定它是否为素数 (prime, 又称质数)。

```
#include <stdio.h>
int main()
{
  int n,i;
  printf("please enter a integer number,n=?");
  scanf("%d",&n);
  for (i=2;i<n;i++)
    if(n%i==0) break;
  if(i<n) // for提前结束
    printf("%d is not a prime number.\n",n);
  else // for正常结束
    printf("%d is a prime number.\n",n);
  return 0;
}
```



只要在循环结束后检查循环变量 i 的值,就能判定循环是提前结束还是正常结束的。从而判定 n 是否为素数。这种判断循环结束的方法以后会常用到。

优化: n 不必被 $2 \sim (n-1)$ 内的各整数去除,只须将 n 被 $2 \sim \sqrt{n}$ 之间的整数除即可。因为 n 的每一对因子, 必然有一个小于 n, 另一个大于 n。

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  printf("please enter a integer number,n=?");
  scanf("%d",&n);
  k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
  for (i=2;i<=k;i++)
    if(n%i==0) break;
  if(i<=k)
    printf("%d is not a prime number.\n",n);
  else
    printf("%d is a prime number.\n",n);
  return 0;
}
```

使用标志变量,判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++) // 比较与上面for的不同
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

使用标志变量,判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++) // 比较与上面for的不同
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

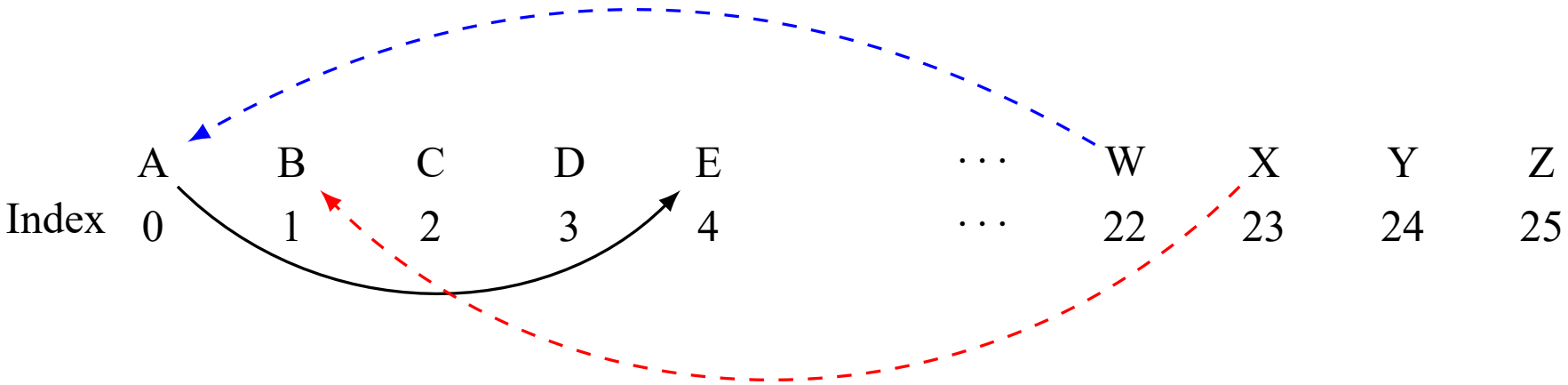
[例 5.10] 求 100 ～ 200 间的全部素数。

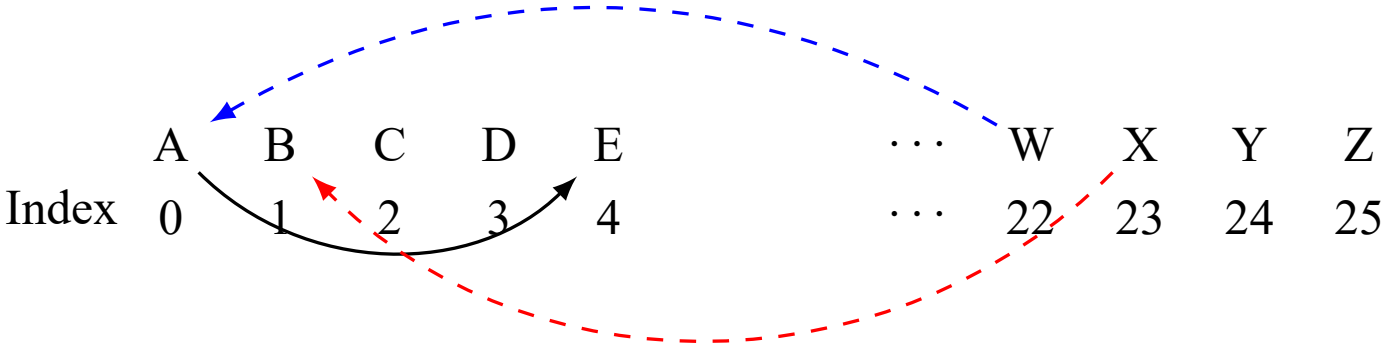
```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;n<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
    k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
    for(i=2;i<=k;i++)
      if(n%i==0) break;
    if(i>k)
      printf("%d ",n);
  }
  printf("\n");
  return 0;
}
```

[例 5.10] 求 100 ~ 200 间的全部素数。

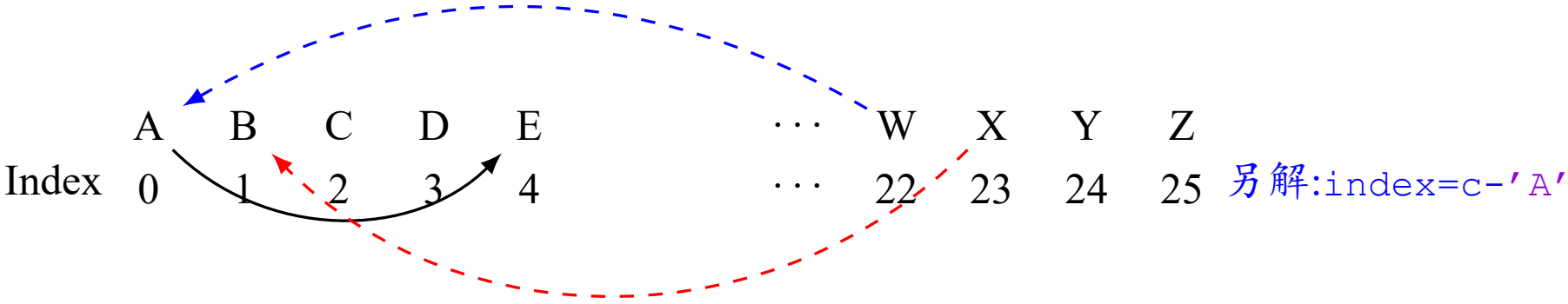
```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;n<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
     k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
     for(i=2;i<=k;i++)
        if(n%i==0) break;
     if(i>k)
        printf("%d ",n);
  }
  printf("\n");
  return 0;
}
```

[例 5.11] 译密码。为使电文保密,往往按一定规律将其转换成密码,收报人再按约定的规律将其译回原文。例如,可以按以下规律将电文变成密码: 将字母 A 变成字母 E,a 变成 e,即变成其后的第 4 个字母,W 变成 A,X 变成 B,Y 变成 C,Z 变成 D。






```
char c;
c=getchar(); //输入一个字符给字符变量c
while(c!='\n') //检查c的值是否为换行符'\n'
{
    if((c>='a' && c<='z') || (c>='A' && c<='Z')) //c如果是字母
    {
        if((c>='W' && c<='Z') || (c>='w' && c<='z')) c = c-22; //如
            果是26个字母中最后4个字母之一就使c-22
        else c =c + 4; //如果是前面22个字母之一，就使c + 4
    }
    printf("%c",c); //输出已改变的字符
    c=getchar(); //再输入下一个字符给字符变量c
}
printf("\n");
```




```
char c;
c=getchar(); //输入一个字符给字符变量c
while(c!='\n') //检查c的值是否为换行符'\n'
{
    if((c>='A' && c<='Z')) //c如果是大写字母
        c='A'+(c-'A'+4)%26;
    else if((c>='a' && c<='z')) //c如果是小写字母
        c='a'+(c-'a'+4)%26;
    printf("%c",c); //输出已改变的字符
    c=getchar(); //再输入下一个字符给字符变量c
}
printf("\n");
```

在循环条件中接收输入的字符是一种常见技巧

```
char c;
while((c=getchar())!='\n') //首先从键盘接收一个字符c，再检查c的值是否为换行
    符'\n'
{
   if((c>='A' && c<='Z')) //c如果是大写字母
      c='A'+(c-'A'+4)%26;
   else if((c>='a' && c<='z')) //c如果是小写字母
      c='a'+(c-'a'+4)%26;
   printf("%c",c); //输出已改变的字符
}
printf("\n");
```

附加题 1: 求 $s = a + aa + aaa + \cdots + a\cdots a$, 其中 a 是一个 1 ~ 9 的数字。例如 $a = 2, n = 4$ 时, $s = 2 + 22 + 222 + 2222$, a 和 n 由键盘输入。

```
int i,s,n,term = 0;
for(i=1,s=0; i<=n; i++) // 初始化循环变量用逗号隔开
{
  term = term*10 + a;
  s += term;
}
```

附加题 1: 求 $s = a + aa + aaa + \cdots + a\cdots a$, 其中 a 是一个 1 ~ 9 的数字。例如 $a = 2, n = 4$ 时, $s = 2 + 22 + 222 + 2222$, a 和 n 由键盘输入。

```
int i,s,n,term = 0;
for(i=1,s=0; i<=n; i++) // 初始化循环变量用逗号隔开
{
   term = term*10 + a;
   s += term;
}
```

附加题 2: 韩信点兵。韩信有一队兵, 他想知道有多少人, 便让士兵排队报数:

按从 1 至 5 报数, 最末一个士兵报的数为 1;

按从 1 至 6 报数, 最末一个士兵报的数为 5;

按从 1 至 7 报数, 最末一个士兵报的数为 4;

按从 1 至 11 报数, 最末一个士兵报的数为 10;

计算韩信至少有多少兵。

```
int x=1;
for(;;x++) // 循环体仅含if()结构, 看作一条语句, '{}'可省略
   if(x%5==1 && x%6==5 && x%7==4 && x%11==10)
   {
      printf("%d\n",x);
      break;
   }
```

附加题 2: 韩信点兵。韩信有一队兵, 他想知道有多少人, 便让士兵排队报数:

按从 1 至 5 报数, 最末一个士兵报的数为 1;

按从 1 至 6 报数, 最末一个士兵报的数为 5;

按从 1 至 7 报数, 最末一个士兵报的数为 4;

按从 1 至 11 报数, 最末一个士兵报的数为 10;

计算韩信至少有多少兵。

```
int x=1;
for(;;x++) // 循环体仅含if()结构, 看作一条语句, '{}'可省略
   if(x%5==1 && x%6==5 && x%7==4 && x%11==10)
   {
      printf("%d\n",x);
      break;
   }
```

附加题 3-1: 求水仙花数。如果一个三位数的个位数、十位数和百位数的立方和等于该数自身,则称该数为水仙花数。

编程求出所有的水仙花数。

解法一: 采用三重循环

```
int i,j,k; // 百、十、个位
for(i=1;i<=9;i++) // 百位
  for(j=0;j<=9;j++) // 十位
    for(k=0;k<=9;k++) // 个位
      if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
        printf("%d\n",i*100+j*10+k);
```

附加题 3-1: 求水仙花数。如果一个三位数的个位数、十位数和百位数的立方和等于该数自身,则称该数为水仙花数。

编程求出所有的水仙花数。

解法一: 采用三重循环

```
int i,j,k; // 百、十、个位
for(i=1;i<=9;i++) // 百位
  for(j=0;j<=9;j++) // 十位
    for(k=0;k<=9;k++) // 个位
      if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
        printf("%d\n",i*100+j*10+k);
```

附加题 3-2: 求水仙花数。如果一个三位数的个位数、十位数和百位数的立方和等于该数自身,则称该数为水仙花数。

编程求出所有的水仙花数。

解法二: 采用一重循环

```
int m,i,j,k;
for(m=100;m<=999;m++)
{
   i=m/100; j=m/10%10; k=m%10;
   if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
     printf("%d\n",i*100+j*10+k);
}
```

思考: 输出共有多少个水仙数?

附加题 3-2: 求水仙花数。如果一个三位数的个位数、十位数和百位数的立方和等于该数自身,则称该数为水仙花数。

编程求出所有的水仙花数。

解法二: 采用一重循环

```
int m,i,j,k;
for(m=100;m<=999;m++)
{
   i=m/100; j=m/10%10; k=m%10;
   if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
      printf("%d\n",i*100+j*10+k);
}
```

思考: 输出共有多少个水仙数?

附加题 3-3: 求整数区间 $[a,b]$ 中水仙花数的个数。

```
int n=0; //计数
int a,b; // a,b 区间
int i,t; // 循环变量, 代表a,b区间的每个数
int sum; // i的各位立方和
scanf("%d%d",&a,&b);
for(i=a;i<=b;i++) // 考察i是否水仙数
{
  sum = 0; t=i; // 临时变量记住i; 易遗漏每次内层循环前sum要归0
  while(t!=0) // 累加各位立方
  { sum=sum+(t%10)*(t%10)*(t%10); // 不好: sum+=pow(t%10,3);
    t=t/10;
  }
  if(sum==i) n++; // i是水仙数
}
printf("%d\n",n);
```

附加题 3-3: 求整数区间 $[a,b]$ 中水仙花数的个数。

```
int n=0; //计数
int a,b; // a,b 区间
int i,t; // 循环变量，代表a,b区间的每个数
int sum; // i的各位立方和
scanf("%d%d",&a,&b);
for(i=a;i<=b;i++) // 考察i是否水仙数
{
  sum = 0; t=i; // 临时变量记住i; 易遗漏每次内层循环前sum要归0
  while(t!=0) // 累加各位立方
  { sum=sum+(t%10)*(t%10)*(t%10); // 不好: sum+=pow(t%10,3);
    t=t/10;
  }
  if(sum==i) n++; // i是水仙数
}
printf("%d\n",n);
```

附加题 4: 百钱百鸡, 已知公鸡 5 个钱 1 只, 母鸡 3 个钱 1 只, 小鸡 1 个钱 3 只, 用 100 个钱买了 100 只鸡。问公鸡、母鸡、小鸡各几只?

```
int x,y,z; // 公鸡、母鸡、小鸡个数
for(x=0;x<=100;x++)
  for(y=0;y<=100;y++)
    for(z=0;z<=100;z++)
      if(5*x+3*y+z/3 == 100 && x+y+z == 100 && z%3 == 0) //全部条件
        printf("%d,%d,%d\n",x,y,z);
```

如何考虑无解的情况?

附加题 4: 百钱百鸡, 已知公鸡 5 个钱 1 只, 母鸡 3 个钱 1 只, 小鸡 1 个钱 3 只, 用 100 个钱买了 100 只鸡。问公鸡、母鸡、小鸡各几只?

```
int x,y,z; // 公鸡、母鸡、小鸡个数
for(x=0;x<=100;x++)
   for(y=0;y<=100;y++)
     for(z=0;z<=100;z++)
       if(5*x+3*y+z/3 == 100 && x+y+z == 100 && z%3 == 0) //全部条件
         printf("%d,%d,%d\n",x,y,z);
```

如何考虑无解的情况?

附加题 5-1: 求整数 a,b 的最大公约数,当两个数中有一个为 0 时,公约数是不为 0 的那个整数; 当两个整数互质时最大公约数为 1。输入两个整数 a 和 b,求最大公约数。

```
int main() // 暴力循环求解, 效率低
{
   int a,b,t=-1,i; // t给初值是好习惯, 否则下面程序逻辑有可能使t得到随机值。
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if(b==0) t=a; // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   else
   {
      for(i=b;i>0;i--)
      {
         if(a%i==0 && b%i==0)
         { t=i; break; } // 求得最大公约数, a,b互质, 必然t=1
      }
   }
   printf("%d\n",t);
   return 0;
}
```

附加题 5-1: 求整数 a,b 的最大公约数，当两个数中有一个为 0 时，公约数是不为 0 的那个整数；当两个整数互质时最大公约数为 1。输入两个整数 a 和 b，求最大公约数。

```
int main() // 暴力循环求解, 效率低
{
    int a,b,t=-1,i; // t给初值是好习惯, 否则下面程序逻辑有可能使t得到随机值。
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
    if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
    if(b==0) t=a; // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
    else
    {
        for(i=b;i>0;i--)
        {
            if(a%i==0 && b%i==0)
            { t=i; break; } // 求得最大公约数, a,b互质, 必然t=1
        }
    }
    printf("%d\n",t);
    return 0;
}
```

求整数 a, b 的最大公约数, 欧几里得算法

古希腊数学家欧几里德在其著作 «The Elements» 中最早描述了这种算法。

定理: 两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数。

令 $a = bq + r, q = \lfloor \frac{a}{b} \rfloor, r = a\%b$, 设 a, b 的公约数 $u, (a = su, b = tu)$, 我们有

$r = a - bq = su - qtu = (s - qt)u$, 表明 a, b 的公约数 u 也整除 r.

类似的, 设 b 和 r 的公约数 $v, (b = s'v, r = t'v)$, 则

$a = bq + r = s'vq + t'v = (s'q + t')v$, 表明 b 和 r 的公约数 v 也整除 a.

(1) 如果 $r = a\%b = 0$, 最大公约数为 b. (2) 如果 $r \neq 0$, 令 $a = b, b = r$, 返回到 (1).

重复迭代 (1),(2) 步, 直至余数为 0, 即求得最大公约数.

注: $\lfloor \frac{a}{b} \rfloor$ 表示 $\frac{a}{b}$ 的结果向下取整, 在 C 语言中, $a/b = \lfloor \frac{a}{b} \rfloor$

欧几里得算法 — 求整数 a, b 最大公约数: 迭代过程

$$
\begin{array}{lll}
q_1 = \lfloor \frac{a}{b} \rfloor & a = bq_1 + r_1 & r_1 = a - bq_1 \\
q_2 = \lfloor \frac{b}{r_1} \rfloor & b = q_2 r_1 + r_2 & r_2 = b - q_2 r_1 \\
q_3 = \lfloor \frac{r_1}{r_2} \rfloor & r_1 = q_3 r_2 + r_3 & r_3 = r_1 - q_3 r_2 \\
q_4 = \lfloor \frac{r_2}{r_3} \rfloor & r_2 = q_4 r_3 + r_4 & r_4 = r_2 - q_4 r_3 \\
\cdots & & \\
q_n = \lfloor \frac{r_{n-2}}{r_{n-1}} \rfloor & r_{n-2} = q_4 r_{n-1} + r_n & r_n = r_{n-2} - q_n r_{n-1} \\
q_{n+1} = \lfloor \frac{r_{n-1}}{r_n} \rfloor & r_{n-1} = q_{n+1} r_n + 0 & r_n = r_{n-1} / q_{n+1}
\end{array}
$$

迭代至 $q_{n+1}, r_{n-1} \% r_n = 0$, 得到 a, b 的最大公约数: $r_n = r_{n-1}/q_{n+1}$. 就是余数为 0 时的分母值.

欧几里得算法 — 求整数 a, b 最大公约数: 迭代过程 (例)

$a = 42, b = 30$

$$q_1 = \lfloor \frac{a}{b} \rfloor = \lfloor \frac{42}{30} \rfloor = 1 \qquad r_1 = 12$$

$$q_2 = \lfloor \frac{b}{r_1} \rfloor = \lfloor \frac{30}{12} \rfloor = 1 \qquad r_2 = 6$$

$$q_3 = \lfloor \frac{r_1}{r_2} \rfloor \lfloor \frac{12}{6} \rfloor = 2 \qquad r_3 = 0$$

Greatest Common Divisor of $(42, 30)$ is

$r_2 = 6$

$a = 144, b = 55$

$$r_1 = a\%b = 34 \qquad r_2 = b\%r_1 = 21$$

$$r_3 = r_1\%r_2 = 13 \qquad r_4 = r_2\%r_3 = 8$$

$$r_5 = r_3\%r_4 = 5 \qquad r_6 = r_4\%r_5 = 3$$

$$r_7 = r_5\%r_6 = 2 \qquad r_8 = r_6\%r_7 = 1$$

$$r_9 = r_7\%r_8 = 0$$

Greatest Common Divisor of $(144, 55)$ is

$r_8 = 1$

欧几里得算法 — 求整数 a, b 的最大公约数: 伪代码

定理: 两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数。

a(大),b(小)的最大公约数: 因为: a=qb+r, q=a/b; r=a%b, ⇒a,b的公约数能整除b和r. r=a%b,r为0, 则b就是最大公约数。否则迭代循环, a=b, b=r, 直到余数为零, 则分母就是最大公约数。

```
while(1)
{
   if(b==0) { gcd=a; break; } // 分母(上轮计算的余数)为0时, a就是最大公约数
   r = a%b; // 注意b为0时, 不能计算余数, a就是最大公约数
   if(r==0) { gcd=a; break; } // 本轮循环的a(上轮循环的b)就是最大公约数
   a=b; b=r; // 准备下一轮迭代
}
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (1)

```
int a,b,r,t;
scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
while(1)
{
   if(b==0) { t=a; break; } // 分母(上轮计算的余数)为0时, a就是最大公约数
   r = a%b;
   if(r==0) {t=b; break;} // b就是最大公约数
   a=b; b=r; // 准备下一轮迭代
}
printf("%d\n",t);// 输出最大公约数
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (2)

```
int main()
{
   int a,b,r,t;
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if (b==0) // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   {
      printf("%d\n",a);
      return 0; // 主函数结束
   }
   while((r=a%b)!=0) // 去除了分母为0的情况
   {
      a=b; b=r; // 准备下一轮迭代
   }
   printf("%d\n",b);
   return 0; // 主函数结束
}
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (3)

```
int main()
{
   int a,b,r,t;
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if (b==0) // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   {
      printf("%d\n",a);
   }
   else
   {
      // 排除了分母为0时不能求余数的情况
      while((r=a%b)!=0) // a/b的余数赋值给r,r不等于0时执行循环体
      { a=b; b=r; }
      printf("%d\n",b);
   }
   return 0; // 主函数结束
}
```

注意事项小结

1. while( ){ }; do { } while( ); for(;;){ } 执行顺序;
2. 循环变量的开始和结束条件;
3. 循环体是复合语句时,必须用 { } 扩起来;
4. 必要时,用 break 结束整个循环,用 continue 结束本次循环;
5. 关键是找出循环规律,必要时设计流程图,指导代码实现。