

知识点 1 【hello world 程序】

```
//include 头文件包含 stdio 标准输入输出头文件

#include<stdio.h>

int num=100;

//系统自动调用 main 函数（他是程序的入口）

//每一个 c 语言项目 有且仅有一个 main

int main(int argc,char *argv[])

{ //复合语句（下方就是函数体）

    //分号;作为语句的结束

    //printf 将" "这种的内容当成字符串输出

    //\n 是换行符 行刷新

    printf("hello world\n");


    //1、返回函数的值 2、结束当前函数

    return 0;

}
```

知识点 2 【关键字的概述】

1、数据类型相关的关键字

用来存储数据。

采用这么多数据类型，本质合理的利用空间。

char 字符 1 字节、short 短整型 2 字节、int 整型 4 字节、long 长整型 4/8 字节、

float 单精度 4 字节、double 双精度 8 字节、struct 结构体、union 共用体、enum 枚举、signed 有符号、unsigned 无符号、void

2、存储相关关键字

register 寄存器变量、static 静态变量、const 只读变量、auto 自动变量、extern 声明外部变量或函数可用

3、控制语句相关的关键字

if、else、break、continue、for、while、do、switch case goto、default

4、其他关键字

sizeof 测量类型的大小、typedef 为已有的类型取个别名、volatile 强制访问内存

知识点 3 【常量和变量】

1、常量 立即数 既见既所得

常量的值 不能被修改 它存放在 文字常量区

比如：10 3.14 'a' "hello world"

```
#include<stdio.h>

void test01()

{

//左值 可以放在=的左值（允许被赋值）

//右值 必须放在=的右边（只能取值或读操作）

10=100;

}

int main(int argc,char *argv[])
```

```
{  
  
test01();  
  
return 0;  
  
}
```

2、变量

1、变量的定义方式：类型名 变量名。

编译器 会更具变量的类型的大小 为变量名 开辟空间

比如：int num; char ch; float f; double d;

int num; 变量名的本质 代表的是所属空间的内容



定义变量是否成功 就看 编译器 能否为变量开辟空间。

```
void test02 ()  
{  
    //由于num的类型为void 编译器不知道void为多大  
    //无法为num开辟空间 所以定义失败  
    void num;  
}
```

变量的命名规则：由字母、数值、下划线组成，不能是数值开头，不能是关键字。（背）

案例：以下命名方式正确的是_CD__

A: int 2_num; B: int auto; C:int _2 D:int num;

知识点 4 【整型】

整型：分为 short int long

1、整型常量 10 20 30

```
void test03()

{

// %d 输出有符号整数 100 默认为 int

printf("%d\n", 100);

// %u 输出无符号整数 100U 才是 unsigned int

printf("%u\n", 100U);


// %hd 输出有符号短整型 short

printf("%hd\n", (short)100);

// %hu 输出无符号短整型 unsigned short

printf("%hu\n", (unsigned short)100);


// %ld 输出有符号长整型 long

printf("%ld\n", 100L);

// %lu 输出无符号长整型 unsigned long

printf("%lu\n", 100LU);

}
```

2、整型变量

2.1 整型变量的定义

```
void test04()  
{  
    //定义变量  
    int num;  
  
    //一行上定义多个变量  
    int a,b;//等价 int a; int b;  
}
```

```
int data;//全局变量  
  
void test04()  
{  
    //定义局部变量  
  
    //局部变量 不初始化 内容为不确定的值  
  
    int num;  
  
    printf("num=%d\n",num);//num=不确定的值  
  
}
```

案例 1：以下代码循环多少次__**不确定**__

```
void test05()  
{  
  
    int i;
```

```
for(i<100;i++)  
  
{  
  
;  
  
}  
  
}
```

2.2 整型变量的初始化

初始化：在**定义变量**的时候给 变量**赋值** 叫做变量的初始化。（背）

```
//定义局部变量的时候给变量赋值 叫初始化  
  
int num = 100;  
  
  
  
//建议将变量初始化为 0  
  
int data = 0;
```

2.3 整型变量的声明（不是必须的）

先**使用** 后**定义**变量 必须先**声明**变量。

```
//变量的声明：告知编译器该变量 将在后方定义 请通过编译  
  
//声明变量的时候 不要给变量赋值 因为变量还没有开辟空间  
  
extern int data;  
  
  
  
  
void test07()  
  
{  
  
  
//使用  
  
printf("data=%d\n",data);
```

```
}
```

```
//定义
```

```
int data = 100;
```

2.4 变量的读写操作

读：取变量的值 写：给变量赋值

```
void test08()
```

```
{
```

```
    int data = 0;
```

```
    //写
```

```
    data = 100;
```

```
    printf("data=%d\n", data);
```

```
    //将 data 的值 初始化 num
```

```
    int num = data;//num 是写 data 是读
```

```
    num=num+1;//读写
```

```
    num+1;//读操作 只是将 num 的值 取值 +1 而已 并没有给 num 赋值
```

```
}
```

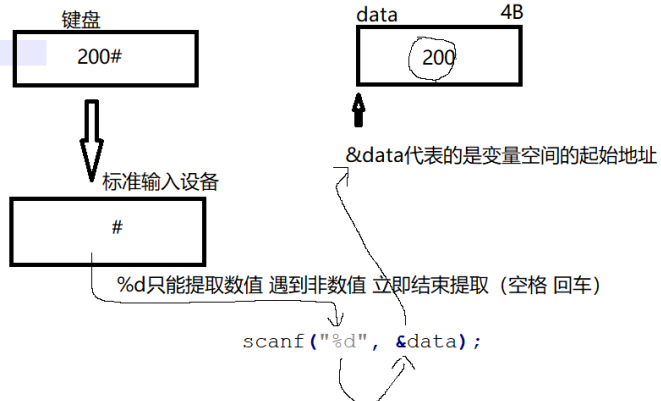
2.5 键盘给变量赋值

```
int data = 0;
```

```
printf("请输入一个int数值:");
```

```
scanf("%d", &data);
```

```
printf("data=%d\n", data);
```



```
void test09()
```

```
{
```

```
int data = 0;
```

```
printf("请输入一个 int 数值:");
```

```
scanf("%d", &data);
```

```
printf("data=%d\n", data);
```

```
}
```

2.7 键盘给两个 int 变量获取输入

```
void test10()
```

```
{
```

```
int data1=0,data2=0;
```

```
printf("请输入两个 int 数值:");
```

```
scanf("%d %d",&data1,&data2);//推荐
```

```
//%d 如果一开始遇到空格或回车跳过
```



```
//scanf("%d%d",&data1,&data2);

//scanf("%d,%d",&data1,&data2);//输入的时候 请按 100,200

printf("data1=%d, data2=%d\n", data1,data2);

}
```

案例 1：键盘输入两个数值 求最大值

知识点 5 【实型】

实型：浮点型（小数）

1、常量

```
//sizeof 是测量类型的大小

//不以 f 结尾的小数为 double

printf("%ld\n",sizeof(3.14));//8B

//以 f 结尾的小数为 float

printf("%ld\n",sizeof(3.14f));//4B


//%f 输出的是 float 类型

printf("%f\n", 3.14f);

//%lf 输出的是 double 类型

printf("%lf\n", 3.14);

123e3 代表 123*10 的三次方
```

2、变量

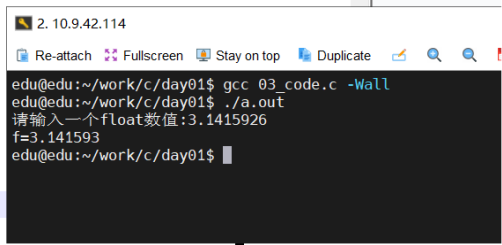
单精度(float)和双精度(double)3.1415926753456

float 型: 占 4 字节, 7 位有效数字,指数-37 到 38

3333.333 33

double 型: 占 8 字节, 16 位有效数字, 指数 -307 到 308

```
16 void test02()  
17 {  
18     float f=0.0f;  
19  
20     printf("请输入一个float数值:");  
21     scanf("%f", &f);  
22  
23     printf("f=%f\n", f);  
24 }
```



```
2. 10.9.42.114  
Re-attach Fullscreen Stay on top Duplicate  
edu@edu:~/work/c/day01$ gcc 03_code.c -Wall  
edu@edu:~/work/c/day01$ ./a.out  
请输入一个float数值:3.1415926  
f=3.141593  
edu@edu:~/work/c/day01$
```

知识点 6【字符 char】

1、常量

字符常量: 'a' '1' ':' '\123'

单引号的作用:

- 1、描述表达式为**字符**。
- 2、取字符的 **ASCII**

0	NUL	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[123	{
28	FS	60	<	92	/	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

https://blog.csdn.net/vqq_44520665

4.1 字符常量



```
void test03()
```

```
{
```

```
    // %d 输出字符的 ASCII 值
```

```

printf("%d\n",'a');

//%c 输出字符

printf("%c\n",'a');

printf("%c\n", 97);

//在内存中'a'和 97 是等价的

printf("%ld\n",sizeof(char));//1B

//'a'取的的 ASCII 值 97 sizeof 是对 97 测量 所以 4 字节

printf("%ld\n",sizeof('a'));//4B
}

```

注意：单引号 只能作用一个字符（转义字符除外）

比如：'a' ok '1' ok '12' error '\123'

2、变量

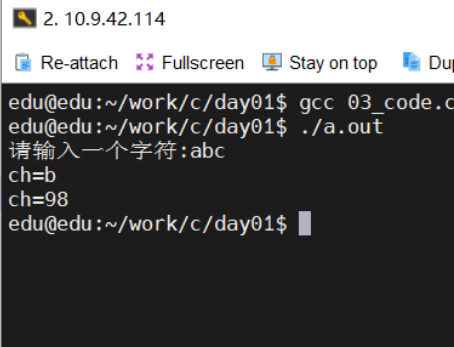
```

void test04 ()
{
    char ch='\0';//'\0'的ASCII值数值0

    printf("请输入一个字符:");
    //%c提取一个字符
    //scanf("%c", &ch);
    //ch = getchar();
    getchar();
    ch = getchar();

    printf("ch=%c\n",ch);
    printf("ch=%d\n",ch);
}

```



```

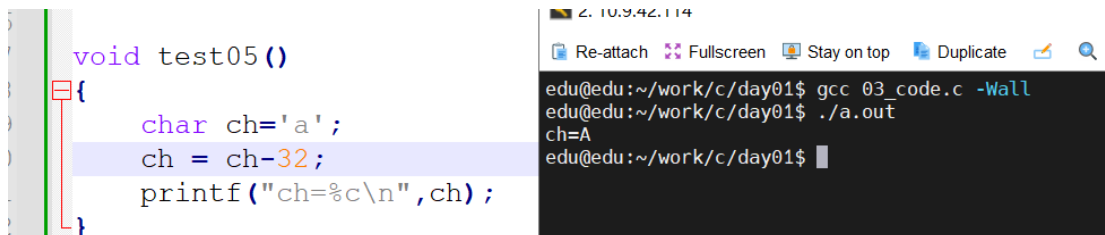
2. 10.9.42.114
Re-attach Fullscreen Stay on top Dup
edu@edu:~/work/c/day01$ gcc 03_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入一个字符:abc
ch=b
ch=98
edu@edu:~/work/c/day01$

```

'a' 97 'b' 98 'c' 99 'z' 122

'A' 65 'B' 66 'C' 67 'Z' 90

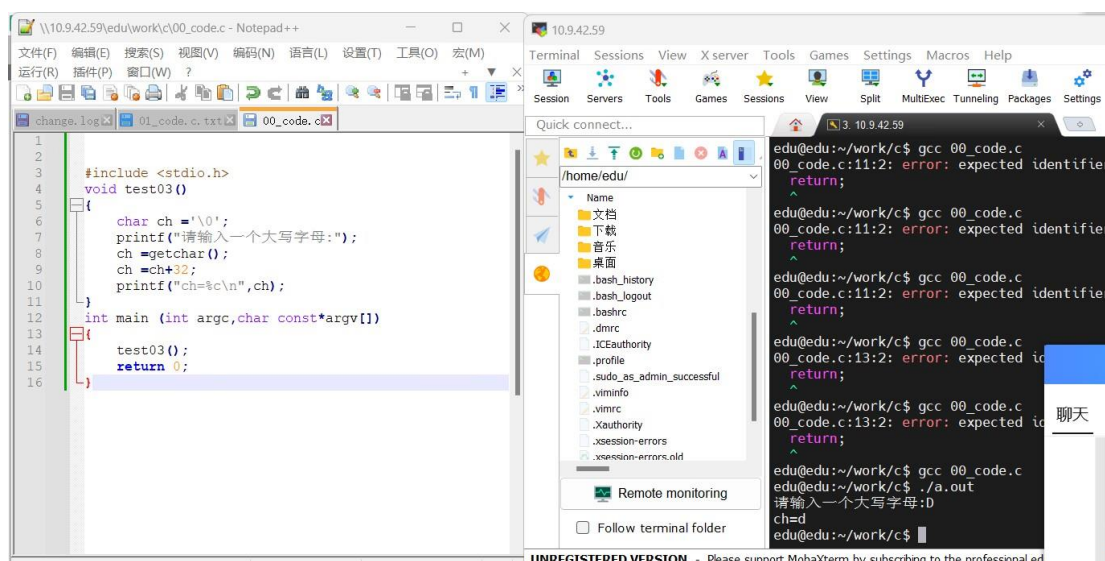
从上分析得到：小写与大写字母之间的 ASCII 差值为 32



```
void test05 ()
{
    char ch='a';
    ch = ch-32;
    printf("ch=%c\n",ch);
}
```

```
edu@edu:~/work/c/day01$ gcc 03_code.c -Wall
edu@edu:~/work/c/day01$ ./a.out
ch=A
edu@edu:~/work/c/day01$
```

案例 1：键盘输入一个大写字母，将其转成小写。



```
#include <stdio.h>
void test03()
{
    char ch = '\0';
    printf("请输入一个大写字母:");
    ch = getchar();
    ch = ch+32;
    printf("ch=%c\n",ch);
}

int main (int argc,char const*argv[])
{
    test03();
    return 0;
}
```

```
edu@edu:~/work/c$ gcc 00_code.c
00_code.c:11:2: error: expected identifier
    return;
    ^
edu@edu:~/work/c$ gcc 00_code.c
00_code.c:11:2: error: expected identifier
    return;
    ^
edu@edu:~/work/c$ gcc 00_code.c
00_code.c:11:2: error: expected identifier
    return;
    ^
edu@edu:~/work/c$ gcc 00_code.c
00_code.c:13:2: error: expected identifier
    return;
    ^
edu@edu:~/work/c$ gcc 00_code.c
00_code.c:13:2: error: expected identifier
    return;
    ^
edu@edu:~/work/c$ gcc 00_code.c
edu@edu:~/work/c$ ./a.out
请输入一个大写字母:D
ch=d
edu@edu:~/work/c$
```

知识点 7 【输出格式】

%d %hd %ld 十进制有符号整数

%u %hu %lu 十进制无符号整数

%x, 以十六进制表示的整数

%o 以八进制表示的整数

%f float 型浮点数 %lf double 型浮点数

%e 指数形式的浮点数

%s 字符串 %c 单个字符

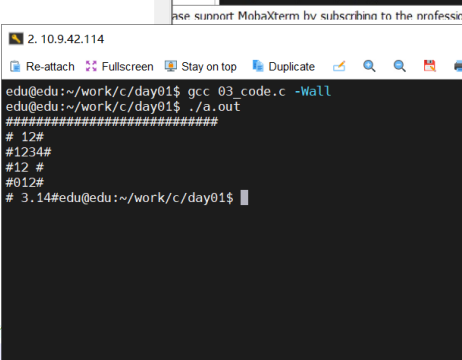
%p 指针的值

特殊应用：

%3d 占 3 个位宽 如果实际输出位数超过 3 格式无效

%03d %-3d %5.2f

```
65 printf("#####\n");
66 // %3d 默认右对齐
67 printf("#%3d#\n", 12);
68 printf("#%-3d#\n", 1234);
69
70 // %3d 左对齐
71 printf("#%-3d#\n", 12);
72
73 // %03d 占 3 位宽 不足 补 0
74 printf("#%03d#\n", 12);
75
76 // %-03d 无效
77 // printf("#%-03d#\n", 12);
78
79 // %5.2f 5 表示整个数的总位宽为 5 2 表示小数部分的
80 printf("#%5.2f#", 3.1445926f);
```



知识点 8 【进制的转换】

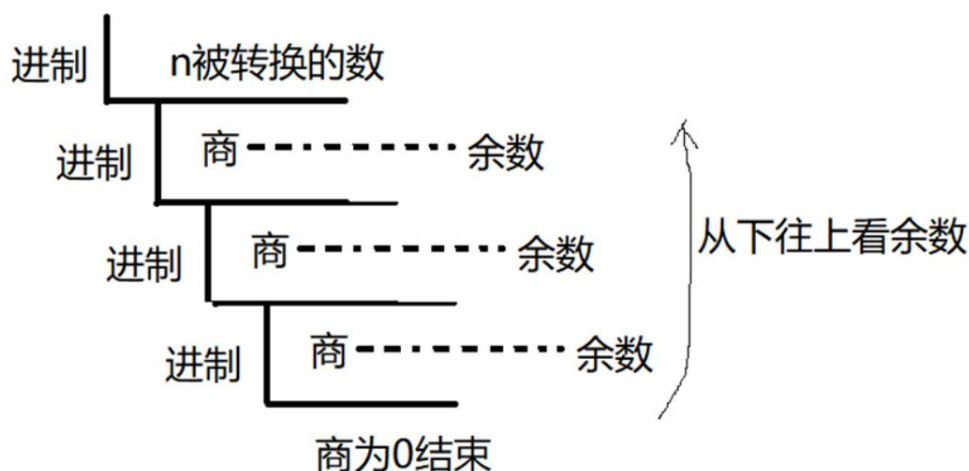
二进制，每一位只能存放 0 或 1 以 0b 开头 c 语言不支持直接输出二进制

八进制，每一位存放范围：0~7 以数值 0 开头 比如：0123 输出格式 %o

十进制，每一位存放范围：0~9 比如：123 输出格式 %d %u %hd %hu %ld %lu

十六进制，每一位存放范围：0~9 a~f 以 0x 开头 比如：0x123 输出形式 %x

1、十进制转二进制、八进制、十六进制 （短除法）



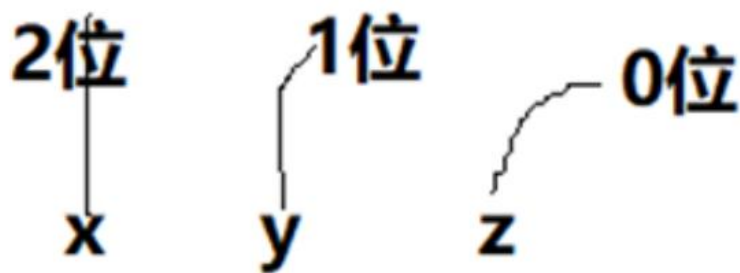
案例 1: 83 转二进制 0b0101 0011

案例 2: 83 转八进制 0123

案例 3: 83 转十六进制 0x53

2、二进制、八进制、十六进制转十进制 (位次幂)

将n进制数xyz转换成 十进制



$$x \cdot n^2 + y \cdot n^1 + z \cdot n^0$$

案例 1: 二进制 0101 0011 转成十进制 83

```
0000 0001 ----->1
0000 0010 ----->2
0000 0100 ----->4
0000 1000 ----->8
0001 0000 ----->16
0010 0000 ----->32
0100 0000 ----->64
1000 0000 ----->128
```

案例 2: 八进制 0123 转十进制 83

案例 3：十六进制 0x123 转十进制 291

3、二进制转八进制

从右往左 将二进制数 每 3 位二进制 对应 1 位八进制

01 011 010 -->0132

11 000 011 --->0303

4、八进制 转 二进制

1 位八进制 转换成 3 位二进制

0123---->0b0101 0011

5、二进制 转 十六进制

从右往左 每 4 位二进制 对应 1 位 十六进制

0101 0110 ----->0x56

6、十六进制转二进制

每 1 位十六进制 对应 4 位二进制

0x123 转二进制 0001 0010 0011

7、八进制 转 十六进制

八进制----->二进制 ----->十六进制

0123----> 0101 0011 ---> 0x53

8、十六进制 转 八进制

十六进制---->二进制----》八进制

0x123---->100 100 011--->0443


```
void test07()
{
    //不管多少进制 在内存中都是二进制存储
    //不同的进制 只是数值的表现形式
    int data = 100;
    printf("十进制data=%d\n",data);
    printf("八进制data=%#o\n",data);
    printf("十六进制data=%#x\n",data);
}
```

```
2. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 03_code.c -Wall
edu@edu:~/work/c/day01$ ./a.out
十进制data=100
八进制data=0144
十六进制data=0x64
edu@edu:~/work/c/day01$
```

知识点 9 【有符号和无符号数】

1、有符号和无符号的区别

有符号：二进制的最高位为符号位，1 表示负数 0 表示正数，其他位为数据位（背）

以 1 字节为例：xddd dddd

-10 二进制位：1000 1010

1111 1111~1000 0000~0000 0000~0111 1111

-127 ~-0 ~+0 ~127

计算机直接将-0 看成-128 所以 1 字节有符号表示范围：-128~127

无符号：二进制中没有符号位 全是数据位（背）

以 1 字节为例：dddd dddd

0000 0000 ~ 1111 1111

0 ~ 255

2、有符号和无符号的定义

//定义有符号变量(不推荐)

```
signed int data=0;
```

//定义有符号变量(推荐)

```
int data=0;
```

```
//定义无符号变量 必须加 unsigned
```

```
unsigned int data=0;
```

3、有符号和无符号输出格式

%d %hd %ld 有符号输出

%u %lu %hu %o %x 都是无符号输出

知识点 10【原码、反码、补码】

1、原码：就是数值的二进制 直接表现形式

无符号数：10 原码 0000 1010

有符号数：

正数：+10 原码 0000 1010

负数：-10 原码 1000 1010

2、反码：

无符号数：10 原码 0000 1010 == 反码 0000 1010

有符号数：

正数：+10 原码 0000 1010 == 反码 0000 1010

负数：反码 等于 原码的符号位不变 其他位按位取反

-10 原码 1000 1010 == 反码 1111 0101

3、补码：

无符号数：10 原码 0000 1010 == 反码 0000 1010 == 补码 0000 1010

有符号数：

正数：+10 原码 0000 1010 == 反码 0000 1010 == 补码 0000 1010

负数：补码 = 反码+1 不在乎符号位

-10 原码 1000 1010 反码 1111 0101 补码：1111 0110

4、总结

无符号数、有符号正数：原码=反码=补码

负数：反码 为符号位不变 其他位按位取反 补码：反码+1

任何数据在计算机中以补码存储。

无符号数、有符号正数 在计算机中按原码存储。

负数 在计算机中 按补码存储。

知识点 11 【计算机存数据】

无符号数、有符号正数 在计算机中按原码存储。

负数 在计算机中 按补码存储。

八进制、十六进制赋值都是原码存储。

十六进制 输出的时候 会将内存数据原样输出。

```
void test01()
{
    unsigned short a = 100;

    printf("a=%#x\n",a);//0x64==0000 0000 0110 0100

    short b = 100;

    printf("b=%#x\n",b);//0x64==0000 0000 0110 0100
```

```

b = -100;

//原码 1000 0000 0110 0100

//反码 1111 1111 1001 1011

//补码 1111 1111 1001 1100

printf("b=%#x\n",b);//0xffff9c

b=0123;//原码 0000 0000 0101 0011

printf("b=%#x\n",b);//0x00000053 == 0x53

b=0x8001;//原码 1000 0000 0000 0001

printf("b=%#x\n",b);//0xffff8001
}

```

知识点 12 【计算机取数据】

%d %hd %ld 有符号提取

%u %lu %hu 无符号提取

%o %x 内存数据原样输出

1、对变量取值

无符号 (%u %hu %lu %o %x) 提取,都是内存原样输出 (正常)

有符号 (%d %hd %ld) 提取 (非正常)

需要查看内存的最高位, 如果为 0 原样输出 如果为 1 求补码 当成有符号输出。

```
void test02()
```

```
{
```

```

unsigned int data=0;

data=10;//内存:0000 0000 0000 0000 0000 0000 0000 1010

//对于无符号变量 %u 提取 内存原样输出

printf("%u\n",data);//10

//%d 提取 先看内存的最高位 如果为 0 原样输出

printf("有符号提取%d\n",data);//10


//将-10 赋值给无符号数 先将-10 转成无符号数 (-10 的补码)

data=-10;//-10 的补码 1111 1111 1111 1111 1111 1111 1111 0110

printf("%u\n",data);//4294967286=1111 1111 1111 1111 1111 1111 1111 0110

//%d 提取 先看内存的最高位 如果为 1 表明为某个负数的补码 需要再次求补码 得出原
码

printf("有符号提取%d\n",data);//-10


data=0x80000001;//内存:1000 0000 0000 0000 0000 0000 0000 0001

printf("%u\n",data);//原样输出 2147483649=1000 0000 0000 0000 0000 0000
0000 0001

//1111 1111 1111 1111 1111 1111 1111 1111 ==-2147483647

printf("有符号提取%d\n",data);//-2147483647
}

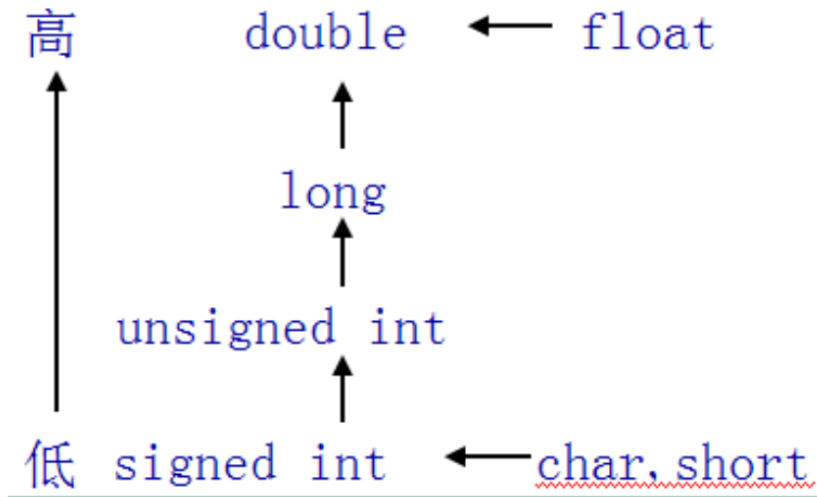
```

知识点 13 【类型转换】

不同的数据类型参加运算 必然涉及类型转换问题。

1、自动类型转换：

保证精度不丢失（小类型往大类型转）



1、有符号和无符号参加运算 先将符号转换无符号（记）

```
void test04()
{
    int a=-10;
    unsigned int b=4;
    //有符号和无符号参加运算 先将符号转换无符号
    //先将-10转成无符号数（-10的补码）+4 是一个很大的数 所以大于0
    if(a+b>0)
    {
        printf("大于0\n");//结果为>0
    }
    else
    {
        printf("小于0\n");
    }

    //%x 内存原样数据
    printf("%x\n",a+b);//0xfffffffffa
    printf("%d\n",a+b);//-6
}
```

```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
大于0
0xfffffffffa
-6
edu@edu:~/work/c/day01$
```

2、char short 只要参加运算 都会先将 char short 转换成 int（留意）

```
void test05()
{
    char a=1;
    short b=2;

    //char short不管是否是混合运算 只要是运算 都将char short先转换成int
    printf("%ld\n",sizeof(a+a));//4
    printf("%ld\n",sizeof(b+b));//4
    printf("%ld\n",sizeof(a+b));//4

    //a没有运算 此处 只是测量a变量的类型大小 为1字节
    printf("%ld\n",sizeof(a));//1
}
```

2、强制类型转换

(强制转换的类型) (被转换的表达式)

强制类型转换 只是临时改变类型。

```
void test06()
{
    float f=3.94f;

    int num = 0;

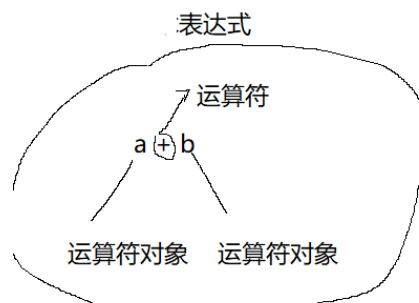
    //在该语句 f 临时转换成 int 一旦该语句结束 f 立马恢复 float

    num = (int)(f);

    printf("f=%f,num=%d\n", f, num);//f=3.940000,num=3
}
```

知识点 14 【运算符】

1、表达式的概述



运算符 如果最少需要n个运算符对象 那就是n目运算符

+就是双目运算符

++就是单目运算符

?:就是三目运算符

2、算数运算符

+ - * / % += -= *= /= %=

1、/ 如果**运算对象都为 char short int long 表示取整。**

```
5/3 == 1
```

```
2/3 == 0
```

2、/ 如果**运算对象有一个是 float 或 double 表示除法。**

```
5/2 == 2
```

```
5/2.0 == 2.5
```

3、%取余 运算对象 不能是实型 (float,double)

```
3%2 == 1
```

```
5%2 == 1
```

```
6%3 == 0
```

$m \% n == 0$ 表明 m 能被 n 整除 或则说 m 是 n 的倍数

$m \% 2$ 取余的范围：0~1

$m \% 3$ 取余的范围：0~2

$m \% 4$ 取余的范围：0~3

$m \% n$ 取余的范围：0~n-1

如果 rand()产生的是一个**随机数** 请使用 rand()产生 60~100 的随机数。

```
rand()%41 +60;
```

如果 rand()产生的是一个**随机数** 请使用 rand()产生'a'~'z'的随机字符。

```
'a' +0~ 'a' +25
```

```
rand()%26+'a';
```

案例 1（晚上）：如果 data=1234 请将千位、百位、十位、各位上的数分别取出


```
int a = data/1000;//千位

int b = data/100%10;//百位

int c = data/10%10;//十位

int d = data%10;//个位
```

4、复合运算符

`+= -= *= /= %=`

```
int data = 10;

data+=20;//data=data+20 ---->data=30
```

注意：将=号右边看成一个**整体**。

```
int data=10;

data *=5+3;//data=data*(5+3) --->data=80
```

案例 1：加入 `int a=12`,请为执行以下语句后 a 的值为__0__

```
a+=a-=a*=a;
```

3、关系运算符

(`>`、`<`、`=`、`>=`、`<=`、`!=`)

`!=`为不等于

4、逻辑运算符

1、&& 逻辑与

A && B 如果 A 和 B 同时成立 整个表达式为结果为真， 只要有一个为假 整个表达式为结果为假

逻辑与的短路特性（背）

如果 A 为假 整个表达式为假 B 的结果不起任何作用 所以编译器不会判断 B （逻辑与的短路特性）

```
113 void test08()  
114 {  
115     int a=5;  
116     int b=6;  
117     (a!=5) && (b=200);  
118     printf("b=%d\n", b);  
119 }
```

```
3. 10.9.42.114  
Re-attach Fullscreen Stay on top Duplicate  
edu@edu:~/work/c/day01$ gcc 04_code.c  
edu@edu:~/work/c/day01$ ./a.out  
b=6  
edu@edu:~/work/c/day01$
```

2、|| 逻辑或

A || B 只要有一个为真 整个表达式为真 只有 A 和 B 同时为假 整个表达式为假

逻辑或的短路特性（背）

如果 A 为真 整个表达式为真 B 的结果不起任何作用 所以编译器不会判断 B （逻辑或的短路特性）

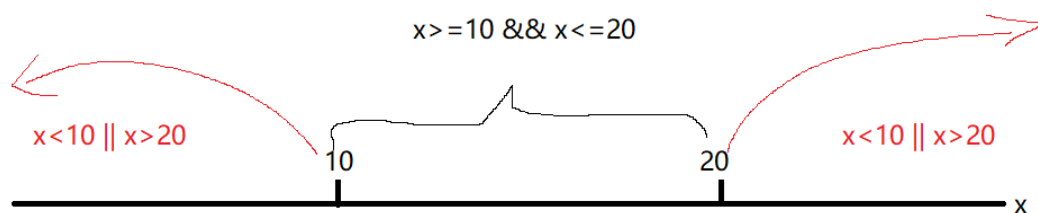
3、!逻辑非

!真 == 假 ! 假 == 真

注意：在 C 语言中 0 表示假 其他都为真

! 10 == 0 ! -10 == 0 ! 0 == 1

4、注意综合案例



5、位运算符

二进制位操作。

1、按位与 &

基本语法：有 0 为 0 全 1 为 1

语法特点：和 1 按位与 保持不变 和 0 按位与 清 0

应用场景：将指定位 清 0

将某位清 0 需要这位与 0 与 其他位 与 1 相与保持不变。

```
1010 0011
& 0000 1111
-----
0000 0011
```

案例 1：将 1 字节的 data 的第 3,5 位清 0 其他为保持不变。

```
data = data & 1101 0111
```

```
data = data & 0xd7;
```

2、按位或 |

基本语法：有 1 为 1 全 0 为 0

语法特点：和 0 按位或 保持不变 和 1 按位或 置 1

应用场景：将指定位 置 1

```
1010 0011
| 0000 1111
-----
1010 1111
```

案例 1：将 1 字节的 data 的第 2,6 位置 1 其他为保持不变

```
data = data | 0100 0100
```

```
data = data | 0x44;
```

3、按位取反~

按位取反：0 变 1 1 变 0

```
~1010 0011 == 0101 1100
```

4、按位异或 ^

基本语法:相同为 0 不同为 1

语法特点：和 0 异或 保持不变 和 1 异或 取反

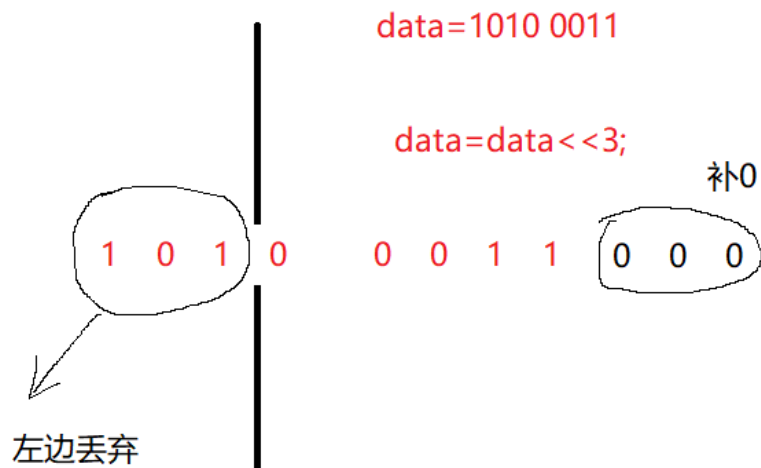
应用场景：将指定位 翻转（取反）

```
1010 0011
^ 0000 1111
-----
1010 1100
^ 0000 1111
-----
1010 0011
```

5、左移 <<

左边**丢弃** 右边**补 0**.

移动的位数 不能操作数据自身的位宽。



假如 data=0000 0001 data=data<<0 data=0000 0001 == 1 == $1*2^0$

假如 data=0000 0001 data=data<<1 data=0000 0010 == 2 == $1*2^1$

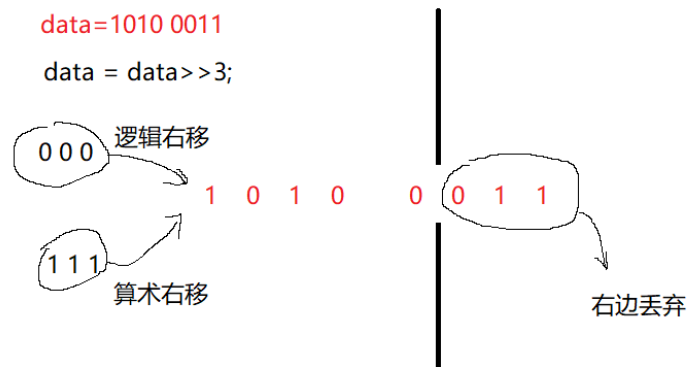
假如 data=0000 0001 data=data<<2 data=0000 0100 == 4 == $1*2^2$

.....

假如 data=0000 0001 data=data<<7 data=1000 0000 == 128 == $1*2^7$

左移 可以简单看成 乘法运算

6、右移>>



无符号、有符号正数：右边丢弃 左边补0

有符号负数：

右边丢弃 左边补0 逻辑右移

右边丢弃 左边补1 算术右移

逻辑右移、算术右移是编译器决定的。

```
121 void test09()  
122 {  
123     int data = -10; //内存数据:1111 1111 1111 1111 1111 1111 1111 0110  
124     data = data >> 16;  
125     printf("%#x\n", data); //逻辑右移0000 0000 0000 0000 1111 1111 1111 1111 == 0x0000ffff == 0xffff  
126     //算术右移1111 1111 1111 1111 1111 1111 1111 1111 == 0xffffffff  
127     if(data == 0xffffffff)  
128     {  
129         printf("算术右移\n");  
130     }  
131     else if(data == 0xffff)  
132     {  
133         printf("逻辑右移\n");  
134     }  
135 }
```

3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01\$ gcc 04_code.c
edu@edu:~/work/c/day01\$./a.out
0xffffffff
算术右移
edu@edu:~/work/c/day01\$

逻辑右移：

假如:data=128==1000 0000 data=data>>0 data=1000 0000 == 128 ==

128/2^0

假如:data=128==1000 0000 data=data>>1 data=0100 0000 == 64 == 128/2^1

假如:data=128==1000 0000 data=data>>2 data=0010 0000 == 32 == 128/2^2

....

假如:data=128==1000 0000 data=data>>7 data=0000 0001 == 1 == 128/2^7

简单认为：右移 是除法运算

7、综合案例（重要）

案例 1：将 1 字节的 data 的第 3,5 位清 0 其他为保持不变。

```
data = data & 1101 0111

1101 0111 == ~(0010 1000) == ~(0010 0000 | 0000 1000)

           == ~(0000 00001<<5 | 0000 0001<<3)

           == ~(0x01<<5 | 0x01<<3)

data = data & ~(0x01<<5 | 0x01<<3)

data &= ~(0x01<<5 | 0x01<<3)
```

案例 2：将 1 字节的 data 的第 2,6 位置 1 其他为保持不变

```
data = data | 0100 0100

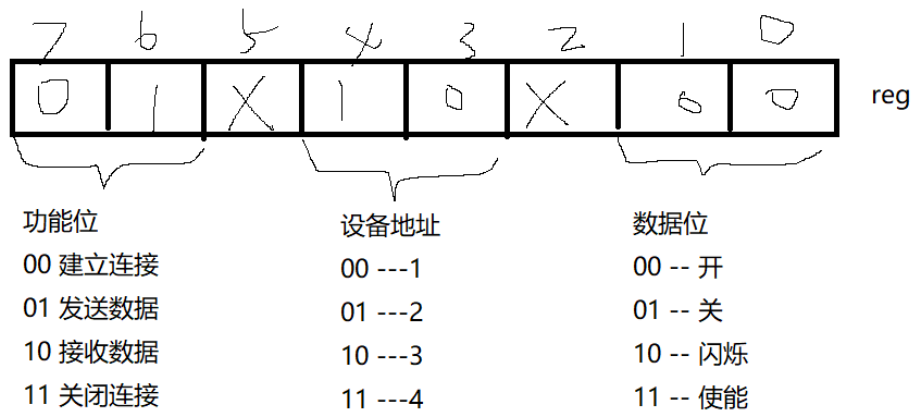
0100 0100 == 0100 0000 | 0000 0100

           == 0000 0001<<6 | 0000 0001<<2

           == 0x01<<6 | 0x01<<2

data = data | (0x01<<6 | 0x01<<2)

data |= (0x01<<6 | 0x01<<2)
```



功能：给3号设备 发送开指令

```
reg &=~(0x01<<0 | 0x01<<1 | 0x01<<3 | 0x01<<7);
reg |= (0x01<<4 | 0x01<<6)
```

8、三目运算符？：

表达式 1 ? 值 1:值 2;

如果表达式 1 为真 整个语句的结果为值 1 否则为值 2

```
137 void test10()
138 {
139     int a = 0, b=0;
140     printf("请输入两个int数值:");
141     scanf("%d %d", &a, &b);
142
143     printf("最大值:%d\n", (a>b?a:b) );
144     printf("最小值:%d\n", (a<b?a:b) );
145
146 }
```


```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入两个int数值:10 20
最大值:20
最小值:10
edu@edu:~/work/c/day01$
```

```
136
137 void test10()
138 {
139     int a = 10;
140     int b = 5;
141
142     a>5?(a=100):(b=200);
143
144     printf("a=%d,b=%d\n", a, b);
145
146 }
```

```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
a=100,b=5
edu@edu:~/work/c/day01$
```


9、逗号运算符

```
148 void test11()  
149 {  
150     int num = 0;  
151     num = 1,2,3,4;  
152     printf("num=%d\n",num); //1  
153  
154     int data = 0;  
155     data = (1,2,3,4);  
156     printf("data=%d\n",data); //4  
157 }
```



```
3.10.9.42.114  
Re-attach Fullscreen Stay on top Duplicate  
edu@edu:~/work/c/day01$ gcc 04_code.c  
edu@edu:~/work/c/day01$ ./a.out  
num=1  
data=4  
edu@edu:~/work/c/day01$
```

知识点 15【优先级】

数值越小 优先级越高 先执行。

优先级 一致 看结合性。

优先级别	运算符	运算形式	结合方向	名称或含义
1	() [] . ->	(e) a[e] x.y p->x	自左至右	圆括号 数组下标 成员运算符 用指针访问成员的指向运算符
2	- + ++ -- ! ~ (t) * & sizeof	-e ++x 或 x++ ! e ~e (t)e * p &x sizeof(t)	自右至左	负号和正号 自增运算和自减运算 逻辑非 按位取反 类型转换 指针运算, 由地址求内容 求变量的地址 求某类型变量的长度

3	* / %	e1 * e2	自左至右	乘、除和求余
4	+ -	e1 + e2	自左至右	加和减
5	<< >>	e1 << e2	自左至右	左移和右移
6	< <= > >=	e1 < e2	自左至右	关系运算(比较)
7	== !=	e1 == e2	自左至右	等于和不等比较
8	&	e1 & e2	自左至右	按位与
9	^	e1 ^ e2	自左至右	按位异或
10		e1 e2	自左至右	按位或
11	&&	e1 && e2	自左至右	逻辑与(并且)
12		e1 e2	自左至右	逻辑或(或者)
13	? :	e1 ? e2 : e3	自右至左	条件运算
14	= += -= *= /= %= >>= <<= &= ^= =	x = e x += e	自右至左	赋值运算 复合赋值运算
15	,	e1, e2	自左至右	顺序求值运算

自增自减运算符

知识点 16 【控制语句】

顺序执行、选择、循环

1、if 语句 选择语句

1.1 如果只在乎一个结果 需要使用 if 语句

//if 去判断条件 1 如果成立执行语句 1 否则 跳出 if

```
if(条件 1)

{

    语句 1;

}
```

1.2 如果项目有**两个**结果 不会同时出现 请选择 if。。。else。。。

如果条件 1 为真 执行语句 1 否则执行语句 2

```
if(条件 1)

{

    语句 1;

}

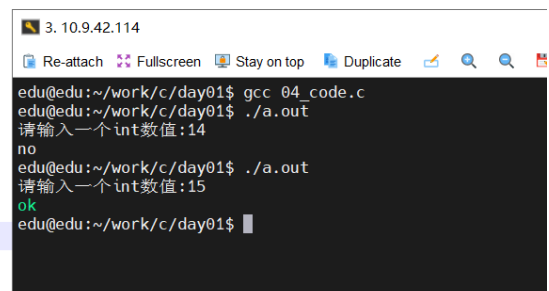
else

{

    语句 2;

}
```

```
159 void test12()
160 {
161     int data = 0;
162     printf("请输入一个int数值:");
163     scanf("%d",&data);
164
165     if(data % 3 == 0)
166     {
167         printf("ok\n");
168     }
169     else
170     {
171         printf("no\n");
172     }
173 }
```



```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入一个int数值:14
no
edu@edu:~/work/c/day01$ ./a.out
请输入一个int数值:15
ok
edu@edu:~/work/c/day01$
```

1.3 项目有过个结果 但是同一时间 只会出现一个结果 if elseif

elseifelse....

```
if(条件 1)

{

    语句 1;

}

else if(条件 2)

{

    语句 2;

}

else if(条件 3)

{

    语句 3;

}

else//可以省略

{

    语句 4;

}
```

如果某个条件成立 后续的判断条件将不会执行。

```

159 void test12()
160 {
161     int data = 0;
162     printf("请输入一个int数值:");
163     scanf("%d",&data);
164
165     if(data % 3 == 0)
166     {
167         printf("0\n");
168     }
169     else if(data % 3 == 1)
170     {
171         printf("1\n");
172     }
173     else if(data % 3 == 2)
174     {
175         printf("2\n");
176     }
177 }

```

```

3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入一个int数值:15
0
edu@edu:~/work/c/day01$ ./a.out
请输入一个int数值:25
1
edu@edu:~/work/c/day01$

```

1.4 项目有多个结果 不确定 是否同时出现

if(条件 1)

```

{
    语句 1;
}

```

if(条件 2)

```

{
    语句 2;
}

```

if(条件 3)

```

{
    语句 3;
}

```

案例 1（晚上）：键盘输入一个字符，如果是大写 转换成小写，如果是小写 转换成大写

其他保持不变

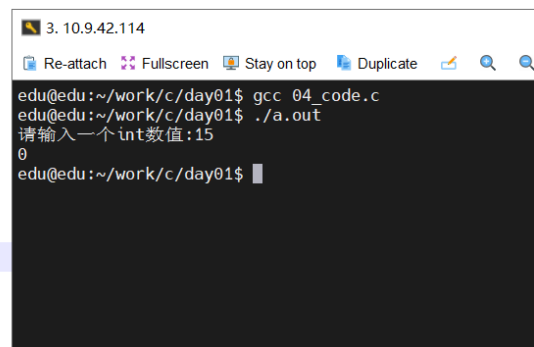
2、switch 语句

switch 会计算"表达式 1"的值 与 case 后面的值 比较

switch(表达式 1)//只能是 char short int long

```
{  
  
    case 值 1://case 后面必须是常量表达式  
  
        语句 1;  
  
        break;//跳出 switch 语句 一般每个 case 都有一个 break  
  
    case 值 2:  
  
        语句 2;  
  
        break;  
  
    default://可以省略  
  
        语句 3;  
  
        break;  
  
}
```

```
178 void test13()  
179 {  
180     int data = 0;  
181     printf("请输入一个int数值:");  
182     scanf("%d",&data);  
183  
184     switch(data%3)  
185     {  
186     case 0:  
187         printf("0\n");  
188         break;  
189     case 1:  
190         printf("1\n");  
191         break;  
192     case 2:  
193         printf("2\n");  
194         break;  
195     }  
196 }
```



```
3. 10.9.42.114  
Re-attach Fullscreen Stay on top Duplicate  
edu@edu:~/work/c/day01$ gcc 04_code.c  
edu@edu:~/work/c/day01$ ./a.out  
请输入一个int数值:15  
0  
edu@edu:~/work/c/day01$
```

```

178 void test13()
179 {
180     int data = 0;
181     printf("请输入一个int数值:");
182     scanf("%d",&data);
183
184     switch(data%3)
185     {
186     case 0:
187         printf("0\n");
188         //break;
189     case 1:
190         printf("1\n");
191         break;
192     case 2:
193         printf("2\n");
194         break;
195     }
196 }

```

```

3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入一个int数值:15
0
1
edu@edu:~/work/c/day01$

```

```

199 void test14()
200 {
201     char ch = '\0';
202     printf("请输入盲僧的技能:");
203     ch = getchar();
204
205     switch(ch)
206     {
207     case 'q':
208     case 'Q':
209         printf("天音波\n");
210         break;
211     case 'w':
212     case 'W':
213         printf("铁布衫\n");
214         break;
215     }
216 }

```

```

3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
请输入盲僧的技能:q
天音波
edu@edu:~/work/c/day01$ ./a.out
请输入盲僧的技能:Q
天音波
edu@edu:~/work/c/day01$

```

3、for 循环语句

for(初始化语句 ; 循环条件 ; 步进语句)

```

{
    循环语句;
}

```

初始化语句：只会在最开始的时候执行一次

循环条件：每次都会判断 只有会真 进入循环语句 为假 跳出循环

步进语句：每次循环结束的时候 执行

216

1 for(初始化语句 ; 循环条件 ; 步进语句)
2 {
3 循环语句;
4 }
5 初始化语句: 只会在最开始的时候执行一次

for(初始语句 1, 语句 2 ; 循环条件 ; 步进语句 1, 语句 2)

```
{
    循环语句;
}
```

```
218 void test15()  
219 {  
220     int i;  
221     int sum;  
222     for( i=0,sum=0; i<=100; i=i+1)  
223     {  
224         sum = sum+i;  
225     }  
226     printf("sum=%d\n",sum);  
227 }
```

3. 10.9.42.114

```
Re-attach Fullscreen Stay on top Duplicate
```

```
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum=5050
edu@edu:~/work/c/day01$
```


1、break 跳出循环

```
void test15()
{
    int i;
    int sum;
    for( i=1,sum=0; i<=100; i=i+1)
    {
        if(i==50)
            break;//跳出循环
        sum = sum+i;
    }
    printf("sum=%d\n",sum); //1~49的和
}
```

```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum=1225
edu@edu:~/work/c/day01$
```

2、continue 结束本次循环 直接进入下一次循环

```
void test15()
{
    int i;
    int sum;
    for( i=1,sum=0; i<=100; i=i+1)
    {
        if(i==50)
            continue;//continue后的代码在本次无法执行
        sum = sum+i;
    }
    printf("sum=%d\n",sum); //1~100除去50的和 == 5000
}
```

```
3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum=5000
edu@edu:~/work/c/day01$
```

3、循环嵌套循环

```
for(i=0;i<10;i=i+1)//外层循环一次
{
    for(j=0;j<100;j=j+1)//内层循环 100 次
    {
        语句 1;

        //break;//只能跳出内层循环
    }
}
```

案例 1：实现九九乘法表

1×1=1									
1×2=2	2×2=4								
1×3=3	2×3=6	3×3=9							
1×4=4	2×4=8	3×4=12	4×4=16						
1×5=5	2×5=10	3×5=15	4×5=20	5×5=25					
1×6=6	2×6=12	3×6=18	4×6=24	5×6=30	6×6=36				
1×7=7	2×7=14	3×7=21	4×7=28	5×7=35	6×7=42	7×7=49			
1×8=8	2×8=16	3×8=24	4×8=32	5×8=40	6×8=48	7×8=56	8×8=64		
1×9=9	2×9=18	3×9=27	4×9=36	5×9=45	6×9=54	7×9=63	8×9=72	9×9=81	

```

231 void test16()
232 {
233     int i=0;//第6行
234     for(i=1;i<=9;i=i+1)
235     {
236         int j=1;
237         for(j=1;j<=i;j=j+1)
238         {
239             printf("%2d*%-2d=%-2d ",j,i,j*i);
240         }
241         printf("\n");
242     }
243 }
244
245

```

```

3. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
1*1 =1
1*2 =2 2*2 =4
1*3 =3 2*3 =6 3*3 =9
1*4 =4 2*4 =8 3*4 =12 4*4 =16
1*5 =5 2*5 =10 3*5 =15 4*5 =20 5*5 =25
1*6 =6 2*6 =12 3*6 =18 4*6 =24 5*6 =30 6*6 =36
1*7 =7 2*7 =14 3*7 =21 4*7 =28 5*7 =35 6*7 =42 7*7 =49
1*8 =8 2*8 =16 3*8 =24 4*8 =32 5*8 =40 6*8 =48 7*8 =56 8*8 =64
1*9 =9 2*9 =18 3*9 =27 4*9 =36 5*9 =45 6*9 =54 7*9 =63 8*9 =72 9*9 =81
edu@edu:~/work/c/day01$

```

案例 1（晚上） 实现如下的九九乘法表

```

1 * 9 = 9 2 * 9 = 18 3 * 9 = 27 4 * 9 = 36 5 * 9 = 45 6 * 9 = 54 7 * 9 = 63 8 * 9 = 72 9 * 9 = 81
1 * 8 = 8 2 * 8 = 16 3 * 8 = 24 4 * 8 = 32 5 * 8 = 40 6 * 8 = 48 7 * 8 = 56 8 * 8 = 64
1 * 7 = 7 2 * 7 = 14 3 * 7 = 21 4 * 7 = 28 5 * 7 = 35 6 * 7 = 42 7 * 7 = 49
1 * 6 = 6 2 * 6 = 12 3 * 6 = 18 4 * 6 = 24 5 * 6 = 30 6 * 6 = 36
1 * 5 = 5 2 * 5 = 10 3 * 5 = 15 4 * 5 = 20 5 * 5 = 25
1 * 4 = 4 2 * 4 = 8 3 * 4 = 12 4 * 4 = 16
1 * 3 = 3 2 * 3 = 6 3 * 3 = 9
1 * 2 = 2 2 * 2 = 4
1 * 1 = 1

```

```

void test19()

{

    int i=5;

    for(i=9;i>0;i=i-1)

```

```

{

    int j=1;

    for(j=1;j<=i;j=j+1)

    {

        printf("%d*%d=%d ",j,i,j*i);

    }

    printf("\n");

}

}

```

4、while

//注意:实现初始语句

while(循环条件)

```

{

    循环语句;

    //注意: 步进语句

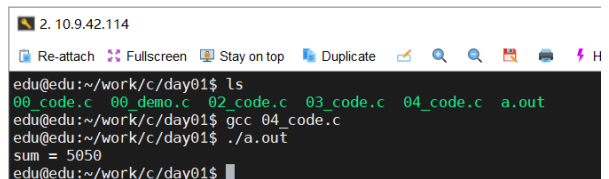
}

```

```

7 void test17 ()
8 {
9     int i=1;
10    int sum = 0;
11
12    while (i<=100)
13    {
14        sum+=i;
15        i=i+1;
16    }
17    printf("sum = %d\n",sum);
18
19 }

```



```

2. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ ls
00_code.c 00_demo.c 02_code.c 03_code.c 04_code.c a.out
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum = 5050
edu@edu:~/work/c/day01$

```

```

47 void test17()
48 {
49     int i=1;
50     int sum = 0;
51
52     while(i<=100)
53     {
54         if(i==50)
55             break; 当等于50跳出while
56         sum+=i;
57         i=i+1;
58     }
59     printf("sum = %d\n",sum) ;//1~49
60
61 }

```

```

2. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum = 1225
edu@edu:~/work/c/day01$

```

```

void test17()
{
    int i=1;
    int sum = 0;

    while (i<=100)
    {
        if(i==50)
            continue; 死循环
        sum+=i;
        i=i+1;
    }
    printf("sum = %d\n",sum) ;
}

```

```

2. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum = 5050
edu@edu:~/work/c/day01$

```

5、do...while

do

```
{
```

循环语句;

```
}while(循环条件);
```

先执行一次循环语句 然后判断循环条件 如果为真 继续循序 为假跳出循环

```

247 void test17()
248 {
249     int i=1;
250     int sum = 0;
251
252     do
253     {
254         sum+=i;
255         i=i+1;
256     }while(i<=100) ;
257     printf("sum = %d\n",sum) ;
258
259 }

```

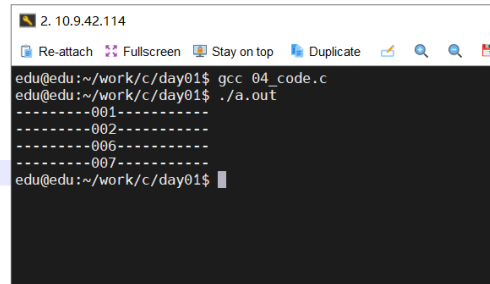
```

2. 10.9.42.114
Re-attach Fullscreen Stay on top Duplicate
edu@edu:~/work/c/day01$ gcc 04_code.c
edu@edu:~/work/c/day01$ ./a.out
sum = 5050
edu@edu:~/work/c/day01$

```

6、goto

```
261 void test18()  
262 {  
263     printf("-----001-----\n");  
264     printf("-----002-----\n");  
265     goto here;  
266     printf("-----003-----\n");  
267     printf("-----004-----\n");  
268     printf("-----005-----\n");  
269     here:  
270     printf("-----006-----\n");  
271     printf("-----007-----\n");  
272 }  
273
```



```
2.10.9.42.114  
edu@edu:~/work/c/day01$ gcc 04_code.c  
edu@edu:~/work/c/day01$ ./a.out  
-----001-----  
-----002-----  
-----006-----  
-----007-----  
edu@edu:~/work/c/day01$
```

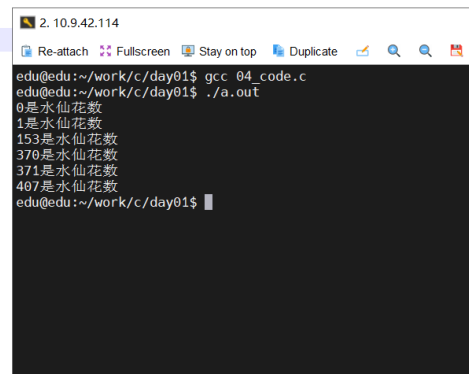
晚上练习:

1.输出 0-1000 以内的水仙花数

水仙花数算法：一个数=它各位的立方和，例如:153= 111 + 555 + 333

提示：for 循环,求余(%)、取整(/)运算符

```
void test20()  
{  
    int data = 0;  
    for(data=0;data<1000;data=data+1)  
    {  
        //data可能是水仙花数  
        //1、取出data每位上的额数值  
        int a = data/100;  
        int b = data/10%10;  
        int c = data%10;  
  
        //2、判断是否是水仙花数  
        if(data == (a*a*a + b*b*b + c*c*c))  
        {  
            printf("%d是水仙花数\n",data);  
        }  
    }  
}
```



```
2.10.9.42.114  
edu@edu:~/work/c/day01$ gcc 04_code.c  
edu@edu:~/work/c/day01$ ./a.out  
0是水仙花数  
1是水仙花数  
153是水仙花数  
370是水仙花数  
371是水仙花数  
407是水仙花数  
edu@edu:~/work/c/day01$
```

2、任意给出一个年、月、日，判断是这一年的第几天:

闰年算法：能被 4 整除且不能被 100 整除，或者能被 400 整除

如：2012 5 10 是这一年的第 131 天

提示：switch

```
void test21()
```

```
{
```

```
int year=0;

int month = 0;

int day=0;

int sum = 0;

printf("请输入年月日:");

scanf("%d %d %d",&year,&month,&day);


//加的整月的天数

switch(month-1)

{

case 11:

sum+=30;

case 10:

sum+=31;

case 9:

sum+=30;

case 8:

sum+=31;

case 7:

sum+=31;

case 6:

sum+=30;
```

```
case 5:

sum+=31;

case 4:

sum+=30;

case 3:

sum+=31;

case 2:

sum+=28;

case 1:

sum+=31;

break;

}


//加号数

sum+=day;


//加平闰年的那一天

if((month>=3) && ((year%4==0)&&(year%100!=0)) || (year%400==0))

sum+=1;

printf("%d %d %d 是这一年的第%d 天\n",year,month,day,sum);

}
```