目录

[1. 类型转换 1](#_Toc37005506)

[1.1 内存截断 1](#_Toc37005507)

[1.2 隐式类型转换 1](#_Toc37005508)

[1.3 显示类型转换 1](#_Toc37005509)

[2. 流程图 2](#_Toc37005510)

[3. 语句 3](#_Toc37005511)

[3.1 表达式语句 3](#_Toc37005512)

[3.2 复合语句 3](#_Toc37005513)

[3.3 return语句 3](#_Toc37005514)

[3.4 分支语句 4](#_Toc37005515)

[3.5 循环语句 8](#_Toc37005516)

[3.6 break语句 11](#_Toc37005517)

[3.7 continue语句 12](#_Toc37005518)

## 1. 类型转换

### 1.1 内存截断

* 将一个数据范围较广的类型赋值给一个数据范围较小类型时，**可能**会发生数据截断（内存截断）。

int a = 0x00020001; // int 4字节  
short b = a; // short 2字节 0x0001  
printf("%d\n", a); // 131073  
printf("%hd\n", b); // 1

### 1.2 隐式类型转换

int a = 10, b = 3;  
printf("%f\n", a/b); // 0.000000 格式转换说明符与实际输出类型不匹配 double <- int  
printf("%f\n", a/b\*1.0); // 3.000000 运算符优先级相同，结合性从左到右 3 \* 1.0  
printf("%f\n", a\*1.0/b); // 3.333333

* **算术运算符**在运算之前必须保证运算符左右两侧的数据类型是一致的。
* 当运算符左右两侧操作数不一致时，数据范围较小的类型会提升到较大类型，这个叫做类型提升。
* 类型提升，我们也可以称它为**隐式类型转换**。
* 隐式类型转换是由编译器自动完成，不需要程序员手动转换。

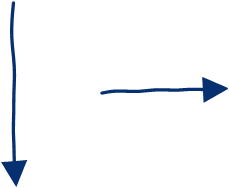
long double  
 |  
double  
 |  
float  
 |  
long long  
 |  
long   
 |  
int <= [char] 、 [short]

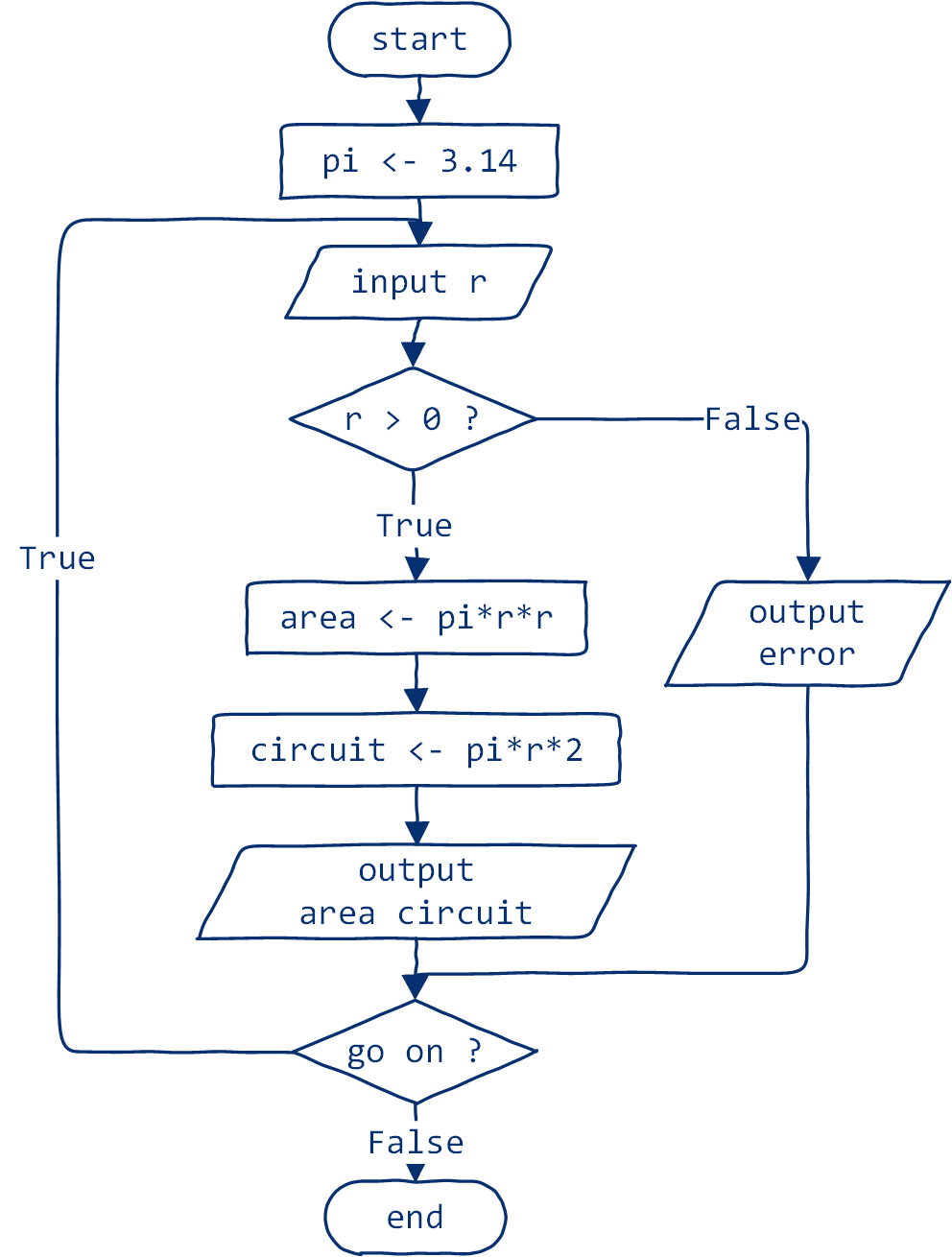
### 1.3 显示类型转换

* 显示类型转换 又称为 强制类型转换。
* 由程序员通过代码，显示的标注出类型转换的方向。
* 显示类型转化格式：**(将要转换成的类型)将要进行转换的数据**

int a = 10, b = 3;  
printf("%f\n", (double)a / b); // 3.333333

## 2. 流程图

* 开始/终止框：表示算法的开始和结束，**开始框**只有一个，**结束框**可以有多个。
* 
* 处理框：表示算法中关于数据的计算和处理。
* 
* 输入/输出框：表示数据的输入和输出。
* 
* 判断框：根据条件，选择算法执行的走向。
* 
* 流程线：表示算法执行的方向。
* 
* 流程图示例：



## 3. 语句

* 顺序结构：执行过程是**从上到下**依次执行每一条**语句**。
* C语言中的语句：表达式语句、复合语句、return语句、条件语句、switch语句、循环语句、break语句、continue语句、goto语句、空语句。

### 3.1 表达式语句

* 表达式语句就是由 **[表达式] + [分号]** 组成的语句。
* 表达式：就是由 **[运算符] 与 [操作数]** 组成的**符合C语言语法**要求的式子。
* 常见表达式语句：赋值语句、自增自减语句、函数调用语句。
  + - 赋值语句： c = a + b;
    - 自增自减语句：++a; b--;
    - 函数调用语句：printf("Hello World!\n");

### 3.2 复合语句

* 复合语句就是由**大括号**中的多条语句组成的一条语句。
* **复合语句可以认为是一条语句。**

{  
 int a = 10;  
 ++a;  
 printf("%d\n", a);  
}

### 3.3 return语句

* return;
* return 表达式;
* **return语句表示结束当前函数的执行。**

### 3.4 分支语句

**if语句最简结构**

if (条件)  
{  
 语句;  
}

* 会根据**条件**的结果，来选择性（条件结果为真）的执行分支内部的语句。
* **if语句会认为它后面的第一条语句为其分支，如果想在分支内书写多条语句，可以通过复合语句来实现。**
* 条件结果为真？C语言中 **非0** 代表 **真**。
* 如果分支内只有一条语句，可以省略大括号。但是不推荐。

**if...else 语句的结构**

（）逻辑中的条件判断真假是逻辑运算，逻辑运算在CPU中的逻辑运算器进行，完成后直接可以得到结果，不用走内存条。

if (条件)  
{  
 语句1;  
}  
else  
{  
 语句2;   
}

* 如果**条件**结果为真，执行语句1所属分支，如果**条件**结果为假，执行语句2所属分支。
* else 代表否则的意思，也就是if条件不成立的时候，所执行的分支。

**if...else if...else 结构**

if (条件1)  
{ 语句1; }  
else if (条件2)  
{ 语句2; }  
else if (条件3)  
{ 语句3; }  
...  
else  
{ 语句N; }

* else if 相当于 [否则]，是前面条件不满足时，再次判断其他条件。
* **if 在一个if语句中只允许出现1次。**
* else if 在一个if语句中 可以出现任意多次（>=0次）。
* else 在一个if语句中可以出现 0 或 1 次。

#include <stdio.h>  
  
int main(void)   
{  
 // 输入一个简单的算术表达式，输出其结果  
 // 1 + 2  
 int a, b, flag;  
 char opt;  
 flag = scanf("%d %c %d", &a, &opt, &b);  
 if(flag != 3)   
 { printf("输入非法！\n"); }  
 else if (opt == '+')   
 { printf("= %d\n", a + b); }  
 else if (opt == '-')   
 { printf("= %d\n", a - b); }  
 else if (opt == '\*')   
 { printf("= %d\n", a \* b); }  
 else if (opt == '/')  
 {   
 if (b != 0) // b  
 { printf("= %.2f\n", (double)a / b); }  
 else  
 { printf("输入非法！\n"); }  
 }  
 else  
 { printf("输入非法！\n"); }  
  
 return 0;  
}

* 示例：判断闰年

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

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

* 示例：成绩评级

#include <stdio.h>  
  
/\*  
 输入一个成绩（整数，[0,100]），要求输出成绩对应的评级  
 优 [90, 100]  
 良 [80, 89]  
 中 [60, 79]  
 差 [0 , 59]  
\*/  
  
int main(void)   
{  
 int n;  
 scanf("%d", &n);  
  
 if (n<0 || n>100)   
 {  
 printf("输入非法！\n");  
 return 0;  
 }  
  
 if (n >= 90)   
 { printf("优\n"); }  
 else if (n >= 80)   
 { printf("良\n"); }  
 else if (n >= 60)   
 { printf("中\n"); }  
 else   
 { printf("差\n"); }  
  
 return 0;  
}

#### 3.4.2 switch语句

* 多路分支语句， 又叫做 开关语句

**格式：**

switch (表达式)  
{  
 case 整型字面值1:  
 语句1;  
 break;  
 case 整型字面值2:  
 语句2;  
 break;  
 case 整型字面值3:  
 语句3;  
 break;  
 ...  
 default  
 语句 n;  
}

* 执行过程：
  + 如果表达式的值等于switch语句中某个case标号中的整型字面值（每个case标号中的字面值**不能相同**），则程序会控制跳转到相应case标号执行，并且逐行执行，一直到switch语句执行结束，或遇到break为止。
  + 如果表达式的值不等于任何case标号后面的字面值，则跳转到default分支执行。如果switch中没有default分支，则不执行任何分支，跳转到switch后面的语句执行。
* 注意：
  + 如果分支后面没有break语句，**则继续往下执行**
  + case标号后面字面值只能是整型字面值（**包括字符字面值**）

int n;  
scanf("%d", &n);  
  
if (n<0 || n>100)  
{  
 printf("输入非法\n");  
 return 0;  
}  
  
switch (n/10) {  
case 0:   
case 1:   
case 2:  
case 3:   
case 4:   
case 5:   
 printf("差\n"); break;  
case 6:   
case 7:   
 printf("中\n"); break;  
case 8:   
 printf("良\n"); break;  
case 9:   
case 10:   
 printf("优\n"); break;  
}

### 3.5 循环语句

#### 3.5.1 for循环

**格式：**

for (初始化表达式 ; 条件表达式 ; 循环表达式)  
{  
 循环体;   
}

* 执行过程：
  1. 执行**初始化表达式**
  2. 判断**条件表达式**的结果是否为真，如果结果为[假]-->[循环结束]，如果为[真]-->执行[**循环体**]
  3. **当循环体执行完成后，执行循环表达式**
  4. **循环表达式**执行完成后，跳转到**步骤2**执行
* C语言中for循环，认为括号后面的第一条语句是其**循环体**，通常会将循环体通过复合语句来实现。
* 初始表达式、条件表达式、循环表达式都可以为空，初始和循环表达式为空表示什么都不做，**条件表达式为空表示条件恒定成立。**
* C语言中，**不允许在初始化表达式中定义变量**，但是C++、Java时允许的。
* 示例：打印水仙花数

/\*  
打印所有的水仙花数。  
三位数中[100, 999]，每一位上的数的三次方之和  
等于这个数本身，我们称之为水仙花数。  
153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3   
\*/  
int i;  
for (i=100; i<1000; i++)  
{  
 int a = i%10;  
 int b = i/10%10;  
 int c = i/100;  
 if( a\*a\*a + b\*b\*b + c\*c\*c == i)   
 {  
 printf("%d\n", i);  
 }  
}

* 示例：**打印int的二进制（难点）**

/\*  
输入一个正整数，打印其二进制形式（32位二进制）。  
\*/  
int n, i;  
scanf("%d", &n);  
  
for (i=31; i>=0; i--)   
{  
 printf("%d", (n >> i) % 2);  
 if (i % 4 == 0)  
 printf(" ");  
}  
printf("\n");  
  
for (i=31; i>=0; i--)   
{  
 printf("%d", (n >> i) & 1);  
 if (i % 4 == 0)  
 printf(" ");  
}  
printf("\n");  
  
for (i=0; i<32; i++)   
{  
 printf("%d", (n << i) < 0);  
 if (i % 4 == 3)  
 printf(" ");  
}  
printf("\n");

* 示例：输入数的和

/\*  
 由用户逐个输入5个正整数，如果用户输入负整数，  
 要求用户重新输入，当用户5个正整数都输入完成后，  
 输出5个正整数的和。  
   
// 请输入第1个数：12  
// 请输入第2个数：-2  
// 输入非法，请重新输入！  
// 请输入第2个数：34  
// 请输入第3个数：1  
\*/   
  
int n, i;  
int sum = 0;  
for(i=1; i<=5; i++)   
{  
 printf("请输入第%d个数：", i);  
 scanf("%d", &n);  
 if (n < 0)   
 {  
 printf("输入非法，请重新输入！\n");  
 i--;  
 }  
 else   
 { sum += n; }  
}  
printf("和为：%d\n", sum);

* 示例：猜数字（**随机数**）

#inlcude <stdlib.h> // C语言标准库头文件，包含rand()函数  
#include <time.h> // C语言时间库头文件，包含time()函数  
  
int main(void)   
{  
 srand(time(0));// 把当前程序运行的时刻作为随机数种子  
 rand(); // 可以产生一个随机数  
 return 0;  
}

#include <stdio.h>  
#include <stdlib.h> // C语言标准库头文件，包含rand()函数  
#include <time.h> // C语言时间库头文件，包含time()函数  
int main(void)   
{  
 /\*  
 猜数字游戏，电脑随机取一个数[1, 100]，用户去猜，电脑  
 根据用户猜测的数字给出提示：大了、小了、猜对了，一共有  
 10次猜数机会。  
 \*/   
  
 int cpt, user, i;  
 srand(time(0));  
 cpt = rand() % 100 + 1;  
 for (i=0; i<10; i++)  
 {  
 printf("请输入一个数：");  
 scanf("%d", &user);  
 if (user > cpt)  
 printf("你猜大了!\n");  
 else if (user < cpt)  
 printf("你猜小了!\n");  
 else  
 break;  
 }  
  
 if (i < 10)  
 printf("你猜对了!\n");  
 else  
 printf("你真笨啊!\n");  
  
 return 0;  
}

#### 3.5.2 while循环

* 又叫做当型循环，表示当条件满足时执行循环。
* 格式：

while (执行循环的条件)  
{  
 // 循环体   
}

* 在使用上，可以认为while循环就是没有初始表达式和循环表达式的for循环。
* while循环中，条件不允许为空，死循环写法： while(1)。
* for和while两种循环的使用场景
  + for：在确定循环次数的时候，选择使用for循环
  + while：在不确定循环次数的时候，选择使用while循环
* 示例：求和

// 用户在命令行会输入很多正整数(>=0)，每个  
// 整数之间使用空格隔开，整数的个数不确定，  
// 所有整数输入完后，用户会输入 -1，表示输入  
// 完成，要求输出所有输入的正整数之和  
  
int num, sum;  
sum = 0;  
scanf("%d", &num);  
while (num != -1)   
{  
 sum += num;  
 scanf("%d", &num);  
}  
  
printf("和等于：%d\n", sum);

* 示例：买苹果
* 每个苹果0.8元，第一天买2个苹果，第二天开始买前一天的2倍，直至购买的苹果个数达到不超过100的最大值。求每天平均花多少钱。

#include <stdio.h>  
  
int main() {  
 int day = 0, apple = 2;  
 double money = 0.0;  
 while (apple <= 100)   
 {  
 money += apple \* 0.8;  
 day++;  
 apple \*= 2;  
 }  
 printf("%.2f\n", money / day);  
  
 return 0;  
}

#### 3.5.3 do...while循环

* 直到型循环，直到某个条件不满足时，才退出循环（会先执行循环体）
* 格式：

do  
{  
 // 循环体   
} while (循环条件) ;

* do...while循环以分号表示结束
* do...while循环无论条件结果如何，循环体都会至少执行一遍
* for/while循环，循环体最少可能一次都不会执行
* 示例：求解一元二次方程小程序

#include <stdio.h>  
#include <math.h>  
#include <windows.h>  
  
int main(void)  
{   
 // 写一个一元二次方程求解小程序，输入三个系数，输出解  
 // 当用户输入完成一次，程序给出解后，会再次询问用户是  
 // 否继续输入  
  
 int a, b, c;  
 double x1, x2, delta;  
 char flag;  
  
 do  
 {  
 printf("======欢迎来到一元二次方程求解小程序======\n");  
 printf(" 请输入三个系数(用空格隔开):");  
 scanf("%d %d %d", &a, &b, &c);  
 delta = b\*b - 4\*a\*c;  
 if (delta < 0)   
 {  
 printf(" 您输入的方程无解!\n");  
 }  
 else  
 {  
 x1 = (-b + sqrt(delta)) / (2\*a);  
 x2 = (-b - sqrt(delta)) / (2\*a);  
  
 if (delta == 0)  
 {   
 printf(" 您输入的方程的解为："  
 "X=%.2f !\n", x1);  
 }  
 else   
 {  
 printf(" 您输入的方程的解为："  
 "X1=%.2f X2=%.2f !\n", x1, x2);   
 }  
 }  
 printf(" 请问是否需要继续计算?(Y/N)：");  
 // fflush 表示清理指定缓存区  
 // stdin 表示标准输入缓存区  
 fflush(stdin);  
 scanf("%c", &flag);  
 system("cls"); // 表示在命令行输入什么指令  
 } while (flag == 'Y' || flag == 'y');  
  
 printf(" 谢谢您的使用，欢迎下次再用！\n");  
  
 return 0;  
}

3.5.4 多重循环

* 示例：打印三角形

#include <stdio.h>  
  
int main(void) {  
 // 输入n，n表示三角形的行数，要求根据n的取值  
 // 打印三角形图形，例如 n=5，需打印如下图形  
 // i 星星 空格  
 // ----\* 0 1 4  
 // ---\*\*\* 1 3 3  
 // --\*\*\*\*\* 2 5 2  
 // -\*\*\*\*\*\*\* 3 7 1  
 // \*\*\*\*\*\*\*\*\* 4 9 0  
 // 2\*i+1 (5-1)-i (5行数)  
  
 // ---\* 0 1 3  
 // --\*\*\* 1 3 2  
 // \_\*\*\*\*\* 2 5 1  
 // \*\*\*\*\*\*\* 3 7 0  
 // (4-1)-i  
  
 int i, j, n; // n表示行数  
 n = 5;  
 // 控制打印多少行  
 for (i=0; i<n; i++)   
 {   
 // 控制每一行打印多少空格  
 for (j=0; j<n-1-i; j++)  
 { printf(" "); }  
  
 // 控制每一行打印多少星星  
 for (j=0; j<2\*i+1; j++)   
 { printf("\*"); }  
  
 printf("\n");  
 }  
  
 return 0;  
}

* 示例：打印菱形

#include <stdio.h>  
  
int main(void)   
{  
 // 打印菱形，输入n，n表示行数 下图n=9   
  
 // n1 = n - n / 2 = 5  
 // i 星星 空格  
 // ----\* 0 1 4  
 // ---\*\*\* 1 3 3  
 // --\*\*\*\*\* 2 5 2  
 // -\*\*\*\*\*\*\* 3 7 1  
 // \*\*\*\*\*\*\*\*\* 4 9 0  
  
 // n2 = n / 2 = n - n1 = 4  
 // -\*\*\*\*\*\*\* 4 7 1  
 // --\*\*\*\*\* 3 5 2  
 // ---\*\*\* 2 3 3  
 // ----\* 1 1 4  
 // 2\*i-1 n2+1-i  
  
 int i, j, n, n1, n2; // n表示行数  
 scanf("%d", &n);  
  
 // 打印上半部分  
 n1 = n - n / 2;  
 for (i=0; i<n1; i++)   
 {   
 for (j=0; j<n1-1-i; j++)  
 { printf(" "); }  
 for (j=0; j<2\*i+1; j++)   
 { printf("\*"); }  
 printf("\n");  
 }  
  
 // 打印下半部分  
 n2 = n - n1;  
 for (i=n2; i>=1; i--)   
 {   
 int space = n2 - i + n%2;  
 for (j=0; j<space; j++)  
 { printf(" "); }  
 for (j=0; j<2\*i-1; j++)   
 { printf("\*"); }  
 printf("\n");  
 }  
  
 return 0;  
}

* 示例：十以内加减法自测小程序

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
#include <windows.h>  
  
int main(void)   
{  
 // 实现一个十以内加减法的测验小程序  
 // 运行程序后，会随机出10道十以内加减法，  
 // 让用户来答题，每道题答完之后给出是否正确  
 // 十道题答完之后给出成绩  
 // 并且询问用户是否继续答题  
  
 int a, b, c, i, res, sum;  
 char flag;  
 srand(time(0));  
  
 do {  
 printf("=====欢迎使用智商自测程序=====\n");  
 sum = 0;  
 for (i=0; i<10; i++) {  
 c = rand()%11; // 0-10  
 a = rand()%(c+1); // 0-c  
 b = c - a;  
 if (rand()%2 == 1) {  
 // 加法  
 printf("%d + %d = ", a, b);  
 }  
 else {  
 // 减法  
 c ^= a ^= c ^= a;  
 printf("%d - %d = %", a, b);  
 }  
 fflush(stdin);  
 scanf("%d", &res);  
 if (c == res) {  
 printf("[√]\n");  
 sum += 10;  
 } else {  
 printf("[X]\n");  
 }  
 }  
 printf("您本次测验的成绩为%d!\n", sum);  
 printf("是否继续测验？(Y/N):");  
 fflush(stdin);  
 scanf("%c", &flag);  
 system("cls");  
 } while (flag == 'Y' || flag == 'y');  
  
 printf("谢谢您的使用，再见！^-^\n");  
  
 return 0;  
}

### 3.6 break语句

* 在switch语句中
* 跳出switch语句，结束当前switch语句的执行。
* 在循环语句中
* 跳出当前循环，执行循环后的语句，表示结束循环执行的意思。
* **break会跳出离它最近的switch语句或循环语句。**

### 3.7 continue语句

* **只允许出现在循环语句中**
* 表示结束本次循环，执行下次循环

### 3.8 goto语句

* 一个功能特别强大的语句，强大到不允许出现在正式开发的代码当中。
* 建议不要在实际开发中使用goto语句，会使程序代码结构变得异常混乱。
* 示例：打印1-100

#include <stdio.h>  
  
int main(void)  
{   
 int i = 1;  
label1:  
 printf("%d\n", i);  
 i++;  
 if (i == 101)  
 goto label2;  
 goto label1;  
label2:  
  
 return 0;  
}