

《C语言程序设计》

丁盟
C语言课程组

```
#include <iostream.h>
#include "bignumb.h"

void main(void){
    big_number a(50);
    long five=5;
    double pi=3.1415926;
    cout << "\n\n";
    cin >> a;
    cout << "b=";
    cin >> b;
    cout
    if (a<b)
        cout << "\na<b";
    if (a>b)
        cout << "\na>b";
    if (a==b)
        cout << "\na=b";
    cout << "\na+b=" << a+b;

    f_in1.unsetf(rskipws);
    getline(f_in1,s);
    try
    {
        s.erase(0,s.find("]",1));
        s.erase(0,(s.find("]",1)+10));
        str= s.substr(0,s.find("]",1));
    }
    return 1;
}

size=str.compare(ip);
if (size==0)
{
    try{
        str=s.substr((s.find("]",1))
```

上一讲知识复习

- ◆理解左值及右值。
- ◆掌握运算符的种类、重点掌握运算符优先级。
- ◆熟悉各种运算符的功能及相关表达式的求值方法。
- ◆了解sizeof运算符。
- ◆了解表达式副作用。
- ◆掌握显式类型转换的方法，了解隐式转换。
- ◆掌握溢出的计算方法，了解在什么情况下可能会造成溢出。

本讲教学目标

- ◆掌握三种基本结构的控制流程。
- ◆熟练掌握C语言的语句：基本语句、分支语句（条件语句）、循环语句。
- ◆着重掌握分支、多重循环的执行过程。
- ◆能读懂程序，明白该程序功能。

本章授课内容



流程图

顺序结构

分支语句

循环语句

中断语句

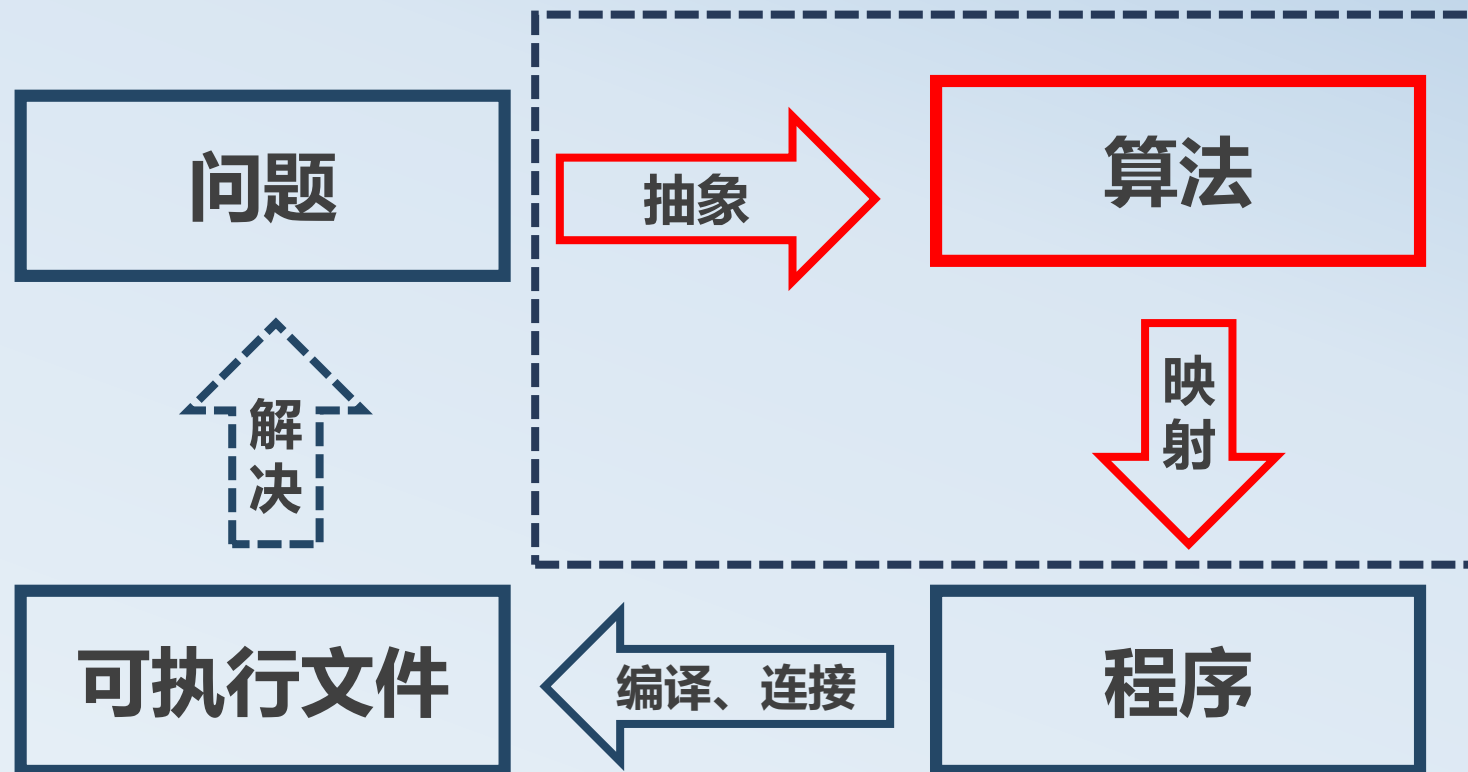
多重循环

再续算法



再续算法

❖ 从问题到求解的大致过程



再续算法

- ❖ 算法就是解决问题的方法（步骤）
- ❖ 目标：
 - ◆ 清晰规范的表示算法（设计一些不太困难的算法）。
 - ◆ 能将算法翻译成c语言程序。
 - ◆ 能够读懂c语言程序

再续算法

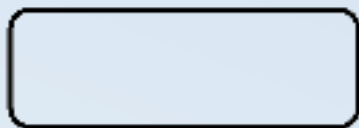
❖ 如何描述算法（流程）？

◆ 文字描述

◆ 流程图（Flow Chart）：使用图形表示算法的思路是一种极好的方法，因为千言万语不如一张图。

流程图

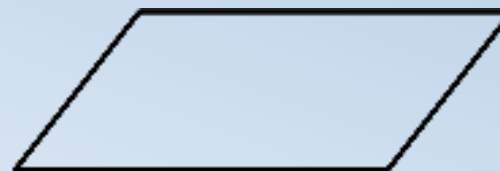
❖ ANSI发布的标准流程图符号



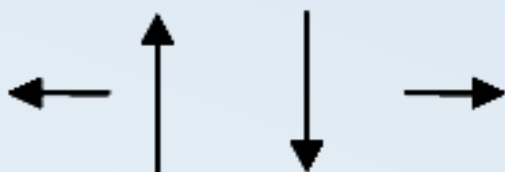
开始/终止框



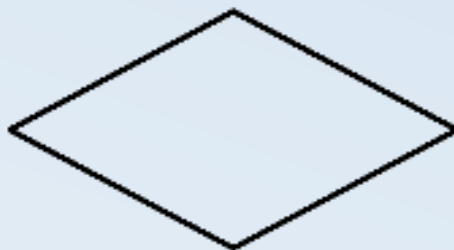
处理框



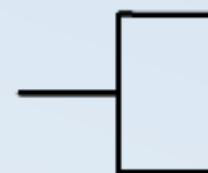
输入/输出框



流程线



判断框



注释框



连接点

流程图

❖ 常见算法的基本结构：

◆ 顺序结构

◆ 分支结构

◆ 循环结构

流程图

- ❖设计一个算法：先得到圆的半径，然后计算并显示圆的面积和周长。

算法的自然语言描述

算法输入：pi（浮点型字面值） 半径

算法输出：圆的面积、周长

处理过程：

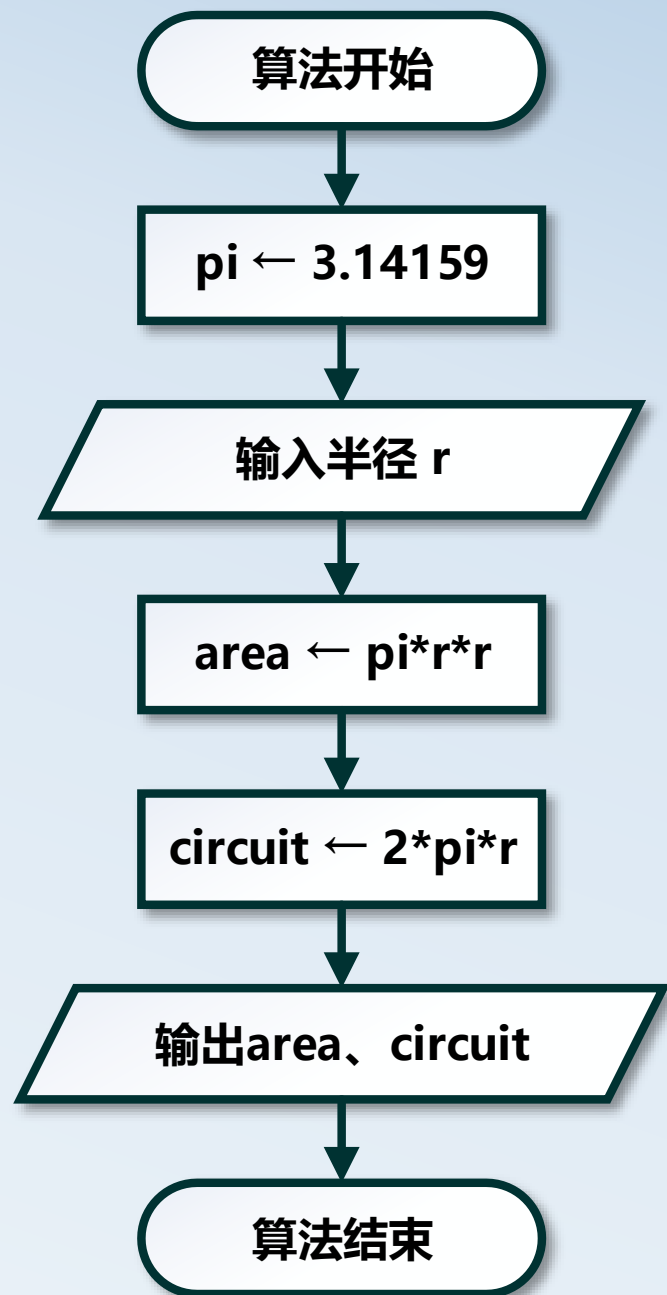
步骤1：输入半径

步骤2：计算面积，计算公式为： $\text{pi} * \text{半径}^2$

步骤3：计算周长，计算公式为： $2 * \text{pi} * \text{半径}$

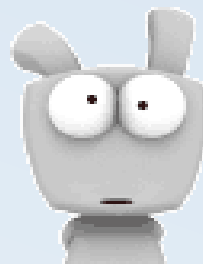
步骤4：输出面积和周长

流程图



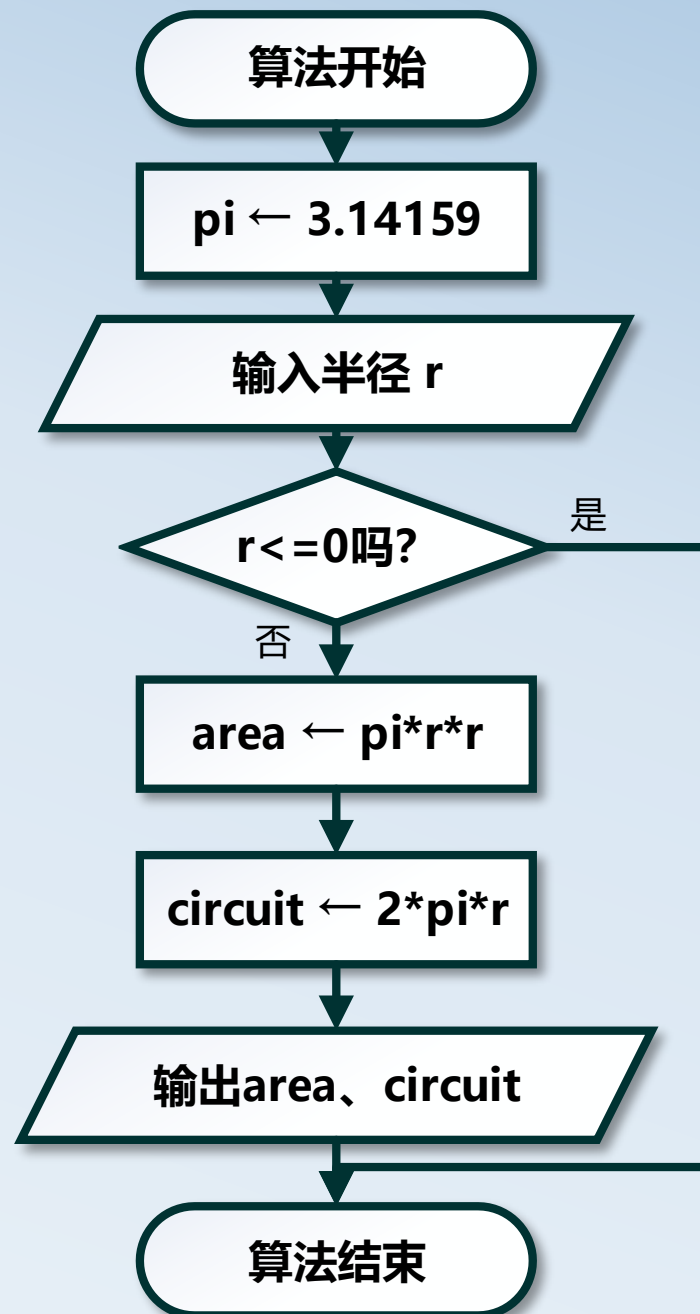
问题:

当 $r \leq 0$ 的时候, 我的计算有什么意义呢?

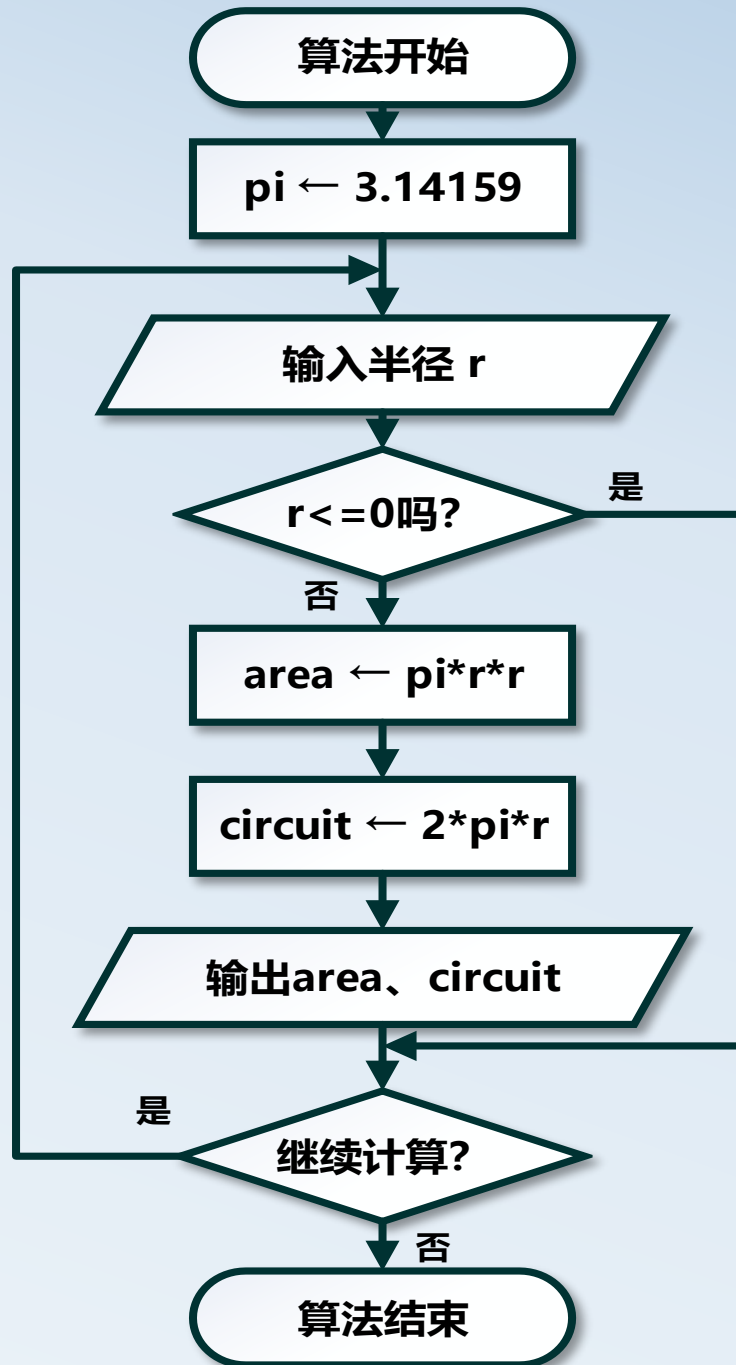


流程图

问题：
怎么做到一
直计算，直到
我想退出呢？



流程图



完美!



本章授课内容



流程图

顺序结构

分支语句

循环语句

中断语句

多重循环

顺序结构

❖ C语言中的顺序结构：执行过程就是**从上到下依次执行语句**。

❖ 常见C语言的语句：

◆ 表达式语句

◆ label语句

◆ 复合语句

◆ 条件语句、switch语句

◆ 循环语句

◆ break语句、continue语句

◆ return语句

◆ goto语句

◆ 空语句

表达式语句

- ❖ 表达式语句由表达式加分号构成。
- ❖ 常见的表达式语句有：赋值语句、自增语句、自减语句、函数调用语句。
- ❖ 例 观察下面的表达式语句。
 - ◆ 赋值语句： $x = y + 3;$
 - ◆ 自增语句： $++i;$
 - ◆ 自减语句： $--i;$
 - ◆ 函数调用语句： `printf("请输入半径:\n");`

复合语句

- ❖ 复合语句由**大括号**中0个或多个**声明**和**语句列表**共同构成.
- ❖ 复合语句是**一条语句**

```
{  
    int x = 3;           // 声明语句  
    ++x;                 // 自增表达式语句  
    printf("%d", x);    // 函数调用语句  
}
```


复合语句

```
#include <stdio.h>
int main(void)
{
    {
        static int y = 4;
        int x = 3;
        {
            int x = 5;
            printf("%d", x);
        }
        ++x;
        printf("%d", x);
    }
    printf("%d\n", x);
    printf("%d\n", y);
    return 0;
}
```

return语句

❖ return语句的基本形式为：

◆ return; 或 return 表达式;

❖ return语句的作用是结束当前函数

```
#include <stdio.h>
int main(void)
{
    ...
    printf("%d\n", x);
    printf("%d\n", y);
    return 0;
}
```

本章授课内容



流程图

顺序结构

分支语句

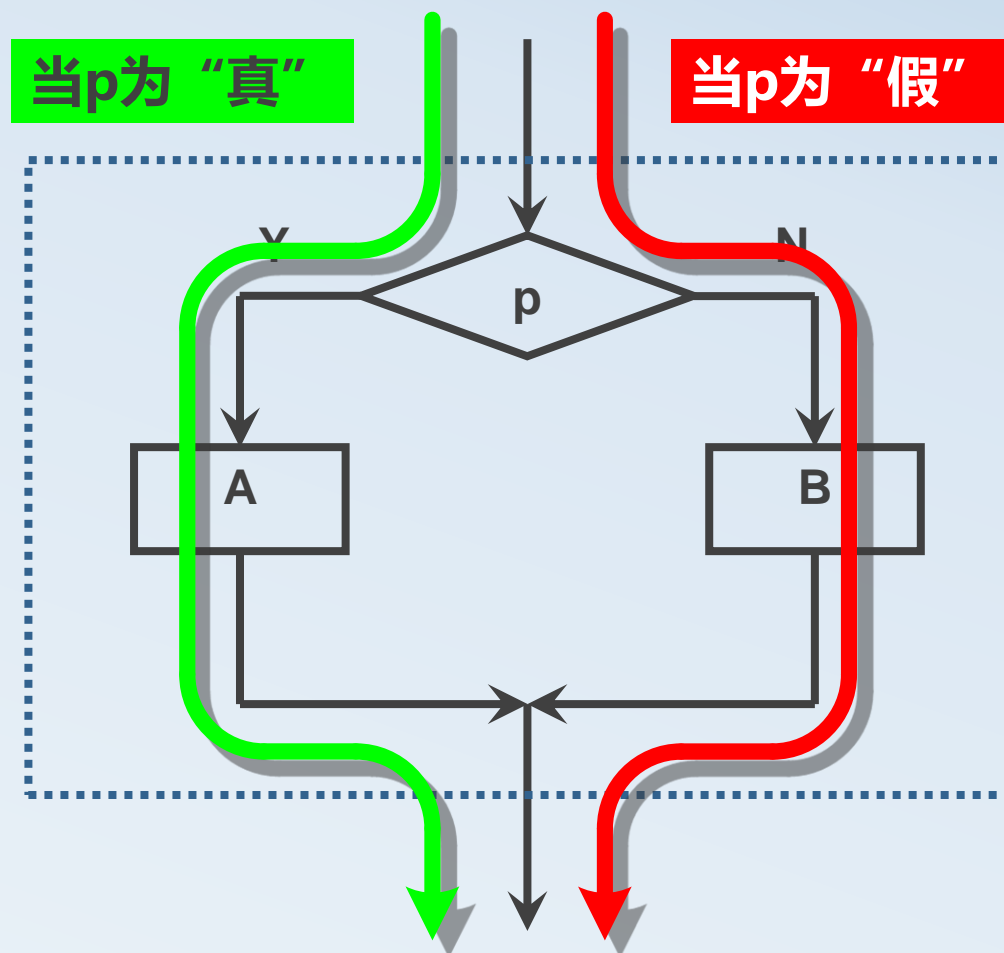
循环语句

中断语句

多重循环

分支结构

- ❖ C语言中的分支结构：执行过程就是满足条件执行A分支顺序语句，否则执行B分支顺序语句。



if语句

❖ 分支语句有两种基本形式：

❖ if语句的一般形式：

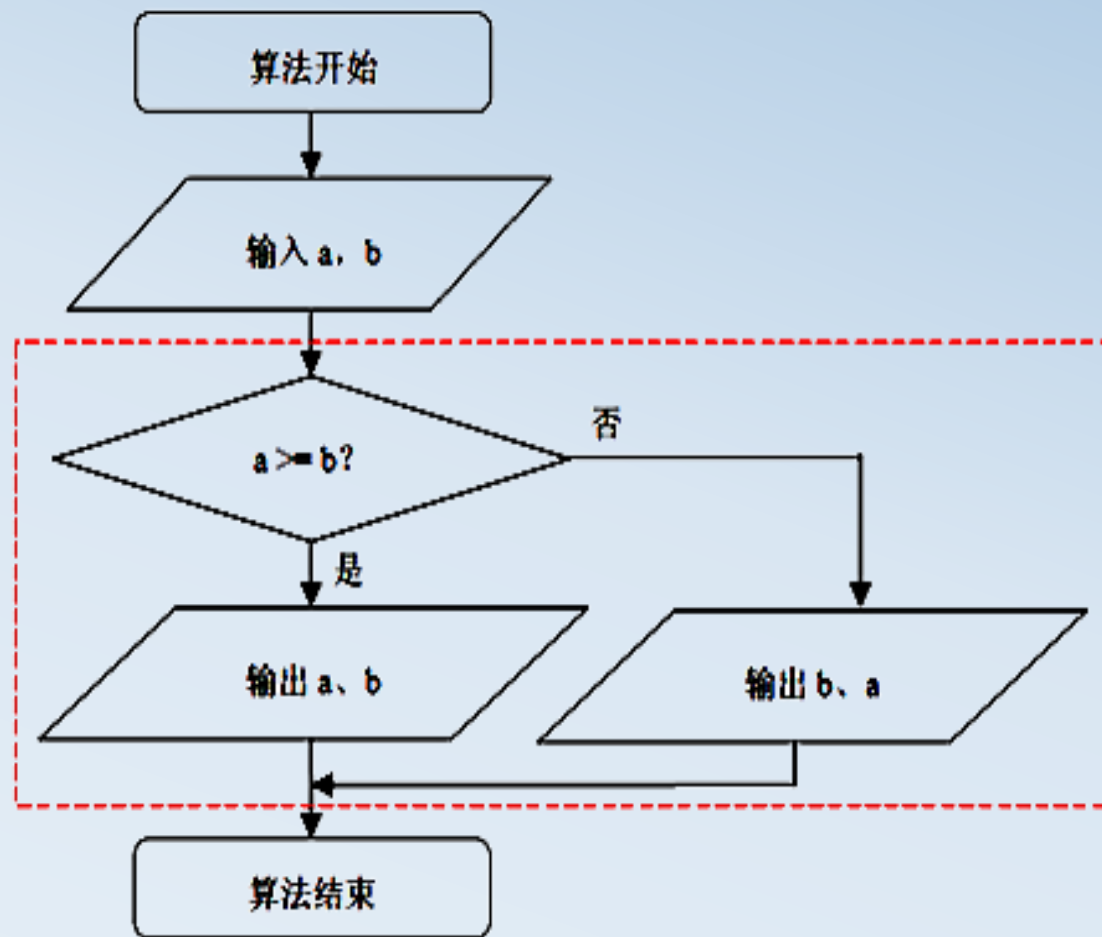
```
if(表达式)
{
    语句;
}
```

❖ if – else 语句的一般形式：

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


if语句

```
#include <stdio.h>
int main(void)
{
    int a, b;
    scanf("%d %d", &a, &b);
    if(a >= b)
    {
        printf("%d %d\n", a, b);
    }
    else
    {
        printf("%d %d\n", b, a);
    }
    return 0;
}
```



if语句

❖ C语言中的多路分支结构：执行过程就是根据判断条件的多个值来判断走哪一条分支，一般为多分支。

❖ 例 从键盘上接受一个百分制的成绩，要求输出成绩等级优、良、差。

[80, 100]之间的成绩等级为优；

[60, 80)之间的成绩等级为良；

[0, 60)之间的成绩等级为差。

if语句

```
#include <stdio.h>
int main(void)
{
    float score;
    printf("请输入成绩: ");
    scanf("%f", &score);
    if(score >=80 && score <= 100)
    {
        printf("优\n");
    }
    else if(score >=60 && score < 80)
    {
        printf("良\n");
    }
}
```

```
else if(score >=0 && score < 60)
{
    printf("差\n");
}
else
{
    printf("输入的成绩有错\n");
}
return 0;
```

}

switch语句

❖ switch语句被称为多路分支语句

❖ 一般形式：

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

表达式的值为整型或字符型

case 标号不能超过1023个

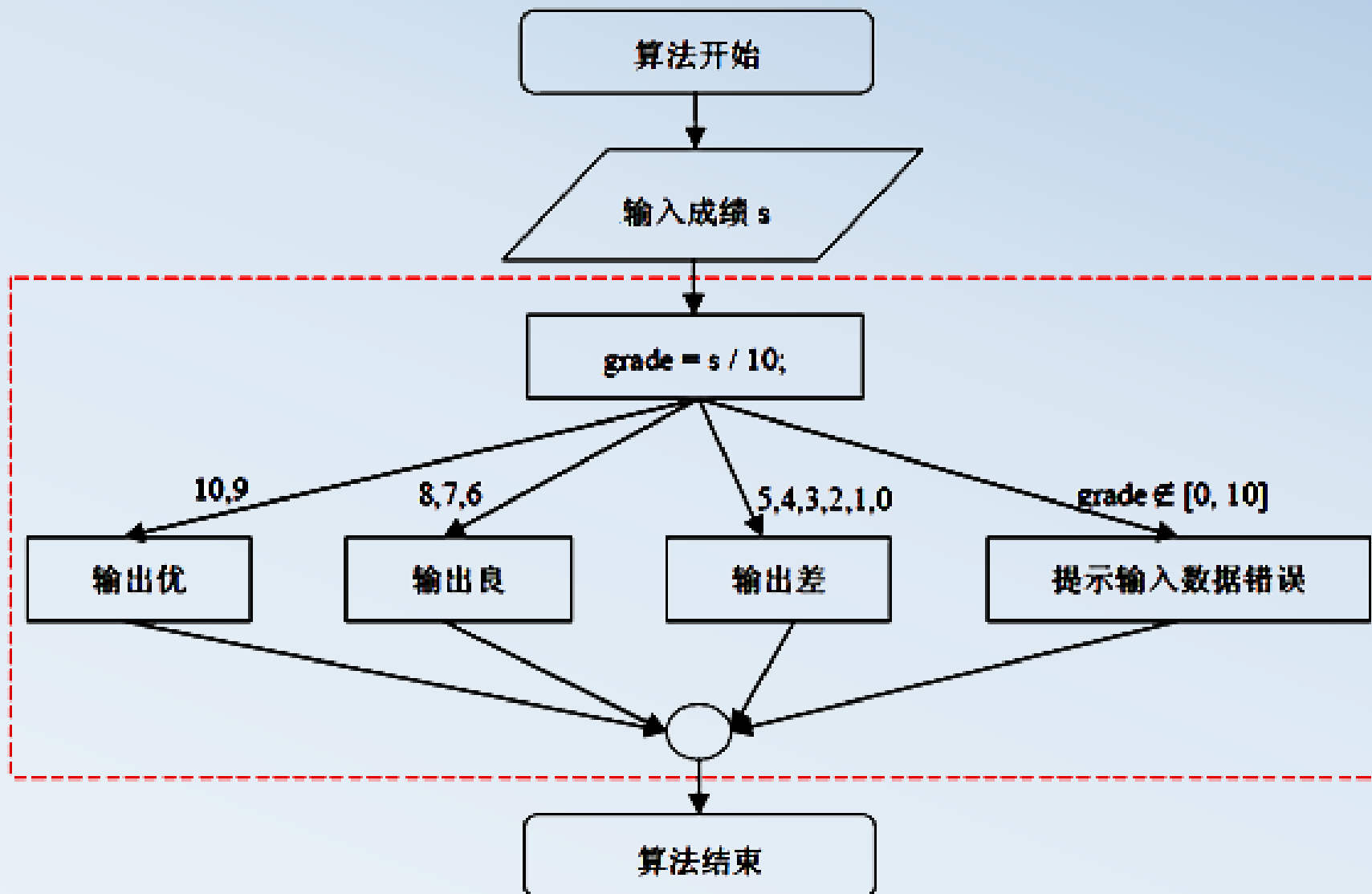
良好的编程习惯

switch语句

❖ switch 语句的执行方法如下：

- ◆ 若表达式的值等于 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)则程序控制**转移**到该case 标号表示的点，从此点开始执行,直到遇见 **break**跳出switch。
- ◆ 若表达式的值不等于任何 case 标号中的字面值，则程序控制转移到default 标号（如果有的话）表示的点，从此点开始执行。
- ◆ 若表达式的值不等于任何 case 标号中的字面值，又没有 default 标号，则不执行switch 语句体中的语句。

switch语句



switch语句

```
#include <stdio.h>
int main(void)
{
    double score;
    int grade;
    printf("请输入成绩: ");
    scanf("%lf", &score);
    grade = (int)score / 10;
    switch(grade)
    {
        case 10:
        case 9:
            printf("优\n");
            break;
        case 8:
        case 7:
```

```
        case 6:
            printf("良\n");
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            printf("差\n");
            break;
        default:
            printf("输入的成绩有错\n");
    }
    return 0;
}
```

本章授课内容



流程图

顺序结构

分支语句

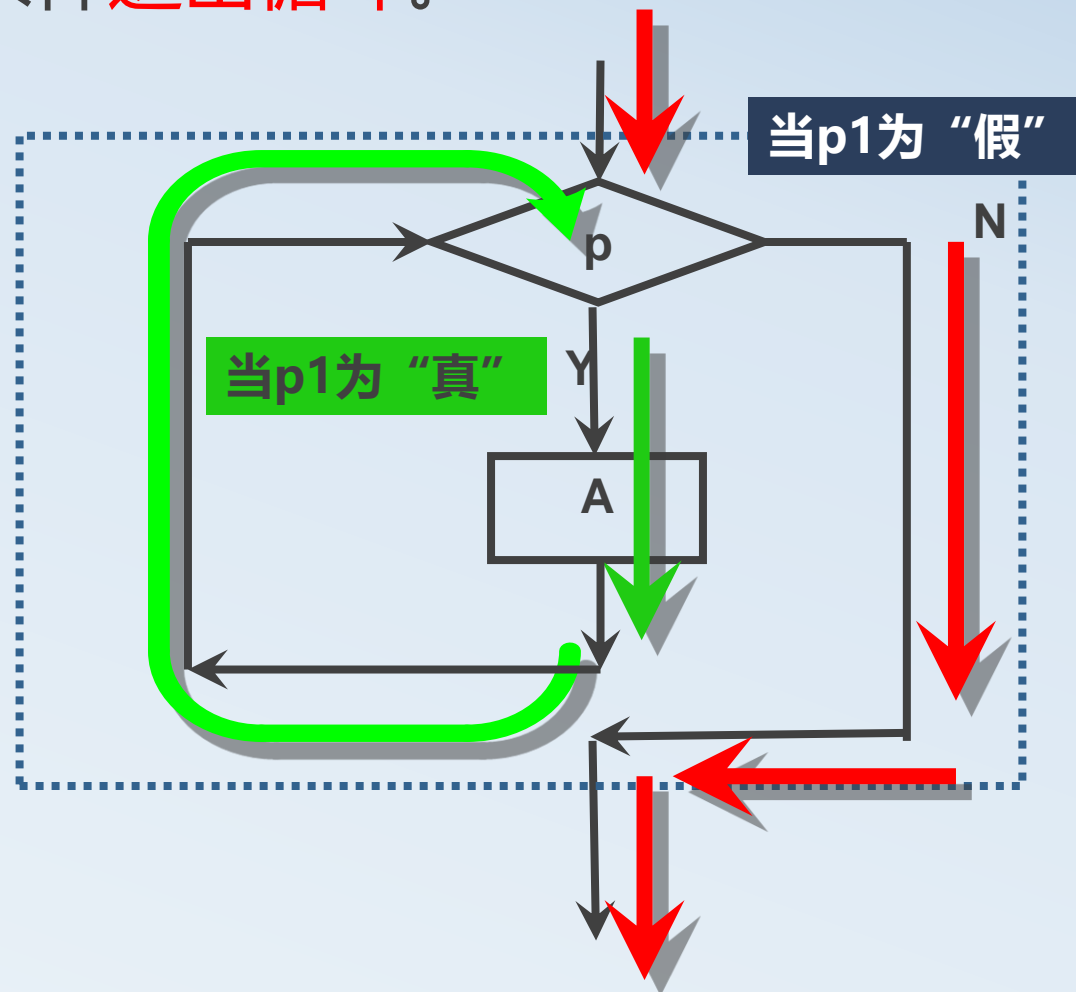
循环语句

中断语句

多重循环

循环结构

- ❖ C语言中的循环结构：执行过程就是**满足条件**进入循环，直到不满足条件**退出循环**。



循环语句

❖ C语言中的循环结构

- ◆ 循环起始结束条件
- ◆ 循环体
- ◆ 循环特殊约束

❖ C语言中的循环语句常见有三种：

- ◆ for语句
- ◆ while语句
- ◆ do...while语句

for循环语句

❖ for循环语句的基本形式为：

for(初始表达式1; 条件表达式2; 循环表达式3)

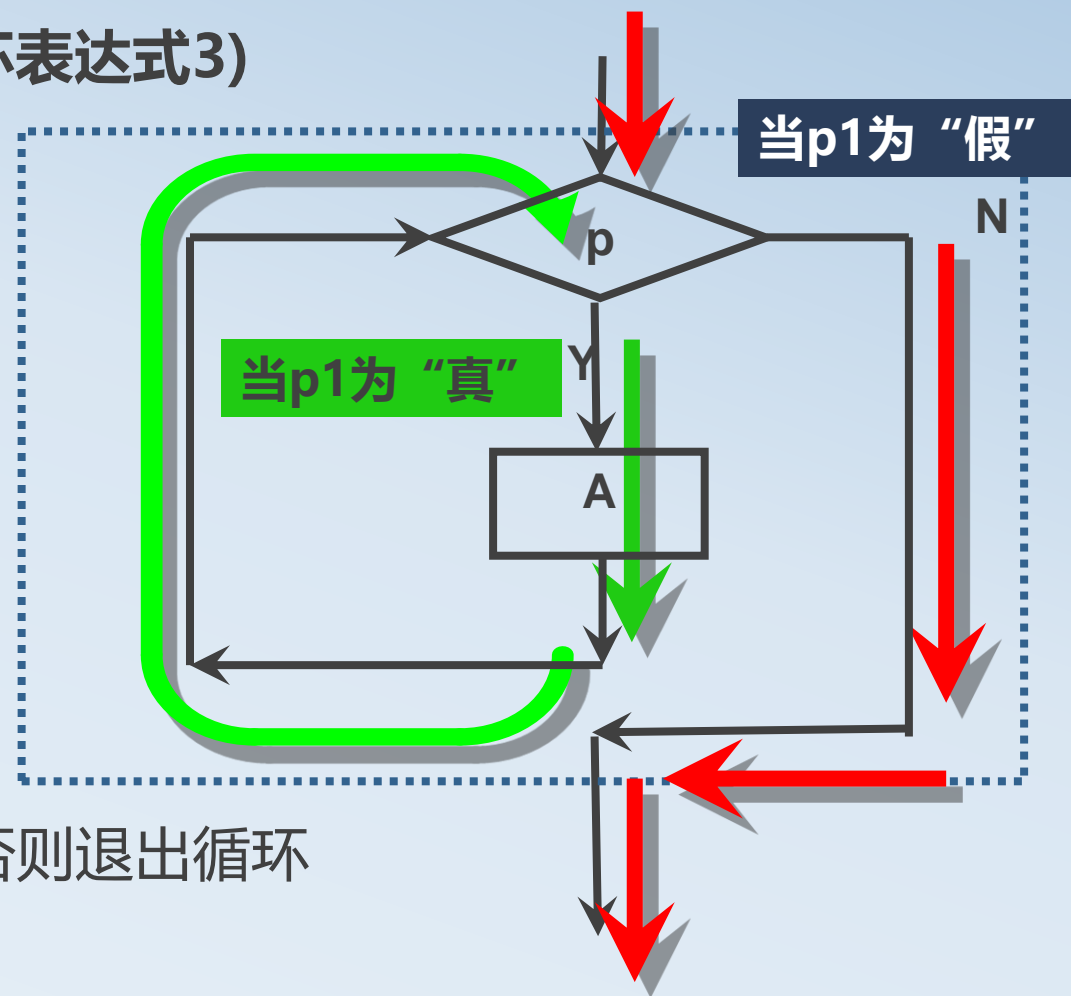
{

语句; // 被称为for 语句体

}

❖ 执行过程：

- 1) 执行表达式1
- 2) 判断是否满足表达式2，满足进入，否则退出循环
- 3) 顺序执行循环体内容
- 4) 执行表达式3，然后转入2)



for循环语句

- ❖ for循环中三个表达式都不是必须，不管是否有内容，执行过程不变
- ❖ 循环编写的时候，要特别注意边界
- ❖ 逻辑错误很难排查

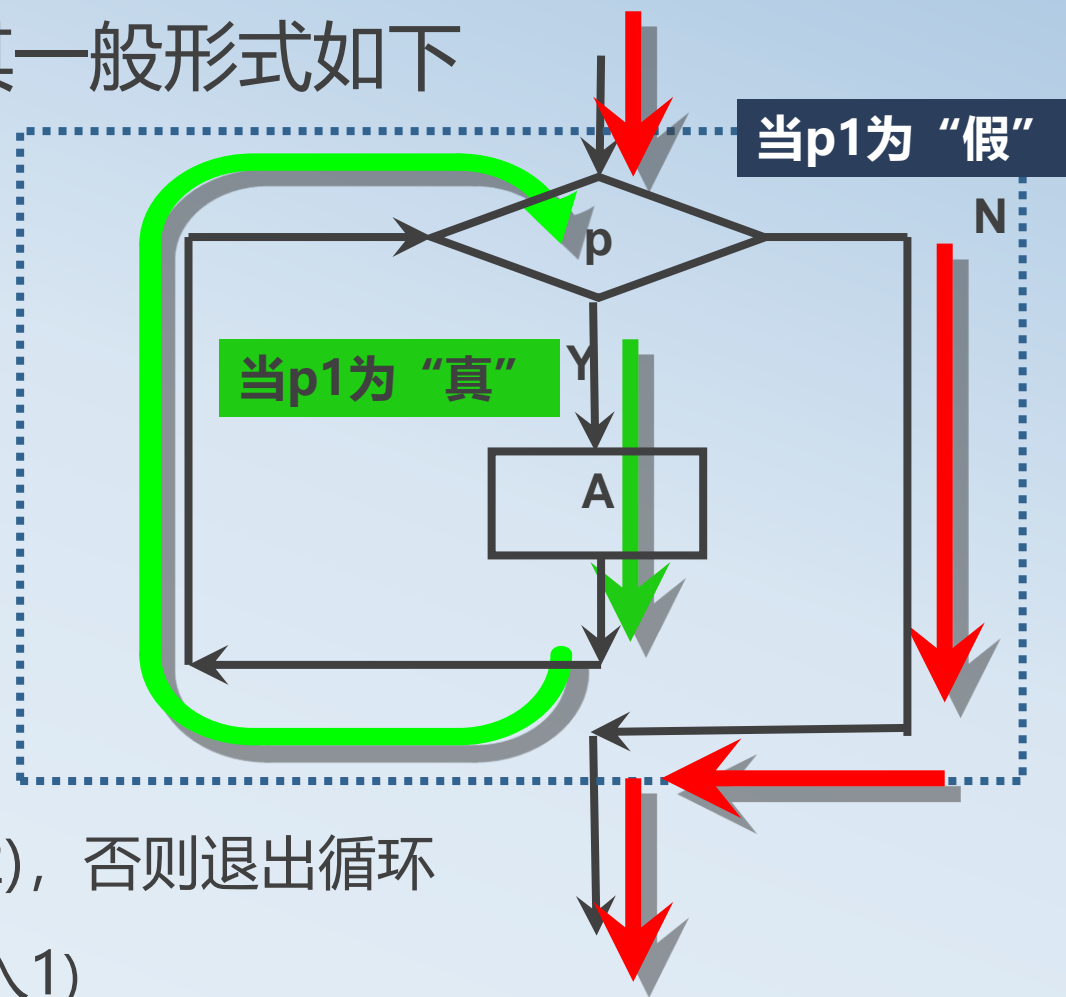
while循环语句

❖ while语句是另一种循环语句,其一般形式如下

```
while(表达式)
{
    语句;
}
```

❖ 执行过程:

- 1) 判断表达式是否为真, 满足进入2), 否则退出循环
- 2) 顺序执行循环体内容, 完毕后进入1)



do...while循环语句

❖ do...while语句是第三种循环语句，基本形式：

```
do
```

```
{
```

```
    语句; // 被称为do 语句体
```

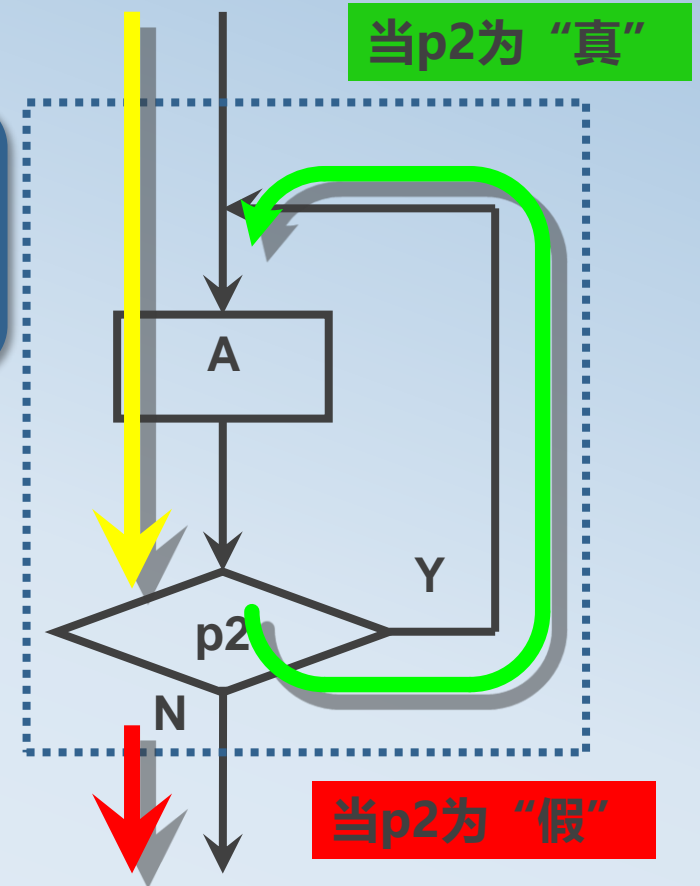
```
} while(表达式);
```

注意：do...while循环语句至少执行一次(与前面两种语句根本区别:结构不同)

❖ 执行过程：

- 1)先执行一次循环体内容
- 2)判断表达式是否为真，满足进入3)，否则退出循环
- 3)顺序执行循环体内容，完毕后进入2)

注意：do...while循环语句至少执行一次(与前面两种语句根本区别:结构不同)



循环语句

❖ 三种循环语句

- ◆ for (表达式1; 表达式2; 表达式3) {...}
- ◆ while (表达式) {...}
- ◆ do{...}while (表达式) ;

❖ for和while的选择

- ◆ for常用在知道循环次数，而且每次循环都要迭代变量
- ◆ while常用在不明确循环次数，只知道一个终止条件

注意：掌握循环的起点，范围，终点。

本章授课内容



流程图

顺序结构

分支语句

循环语句

中断语句

多重循环



改变循环的执行过程

- ❖ 循环语句的执行方式可以被改变：
 - ❖ break: 停止整个循环的执行, 执行循环后的内容
 - ❖ continue: 停止当次循环执行, 执行下一次循环
 - ❖ goto: 跳到指定标号执行

goto语句

❖ goto语句的一般形式为：

◆ *goto* 命名标号;

❖ 标号及紧随其后的语句形成了一个**标号语句**

◆ 标号语句的一般形式为：

◆ 标号：

– 语句

本章授课内容



流程图

顺序结构

分支语句

循环语句

中断语句

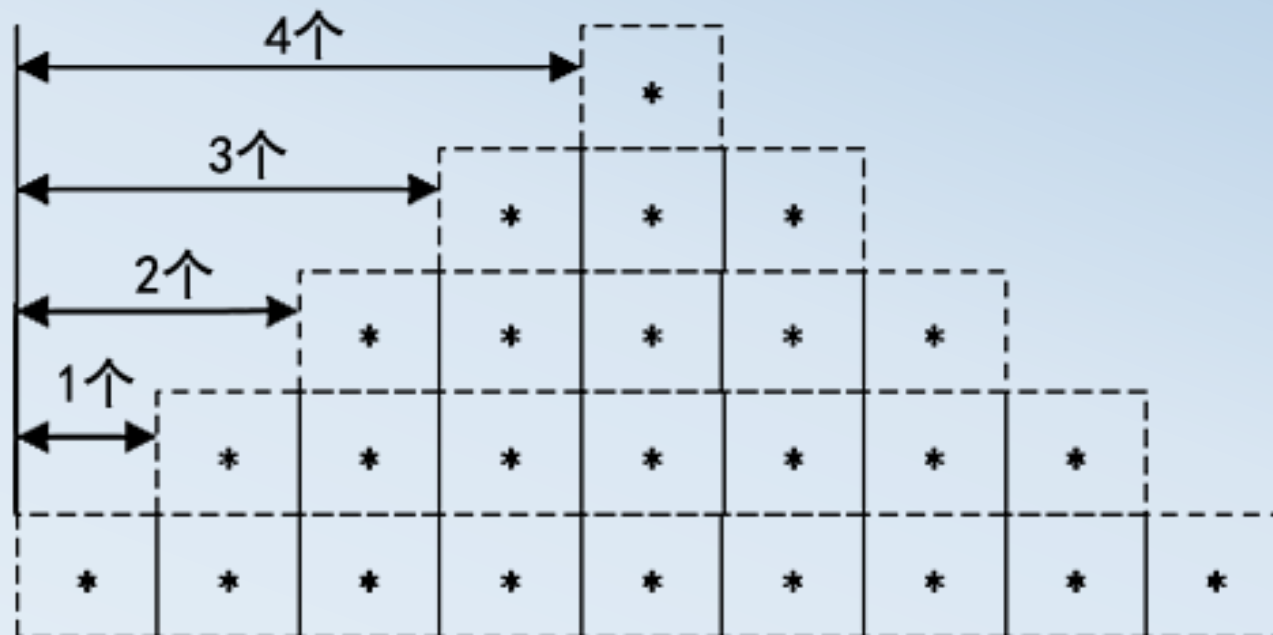
多重循环



多重循环

❖ 例 绘制倒置金字塔，请使用循环语句打印出如下图形。

```
      *
     ***
    *****
   *********
  ***********
```



多重循环

```
#include <stdio.h>

int main(void)
{
    int num = 10;
    int i;

    for(i=0; i<num; ++i)
    {
        int n, m;
        for(m=0; m<num-1-i; ++m)
        {
            printf(" ");
        }

        for(n=0; n<i*2+1; ++n)
        {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

Thank You !