



《计算概论A》课程 程序设计部分

C 语言的基本成分 — 控制成分

李 戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn

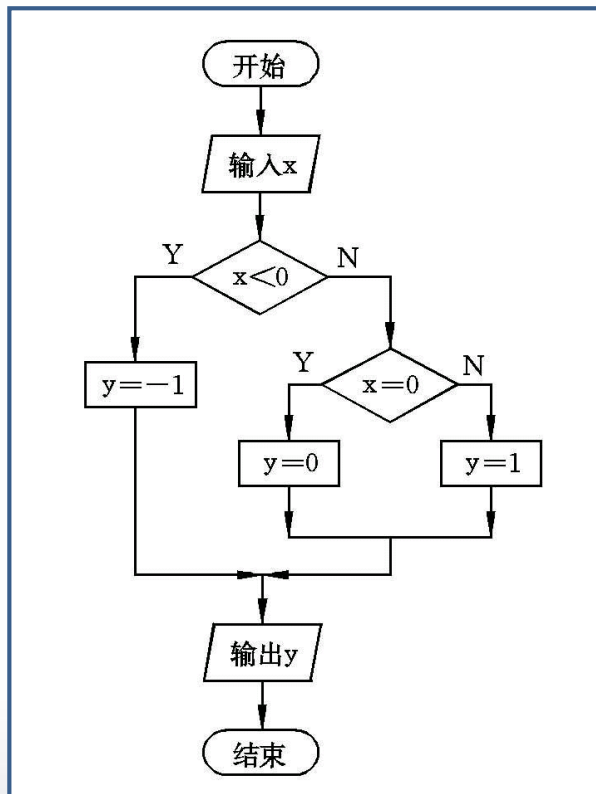
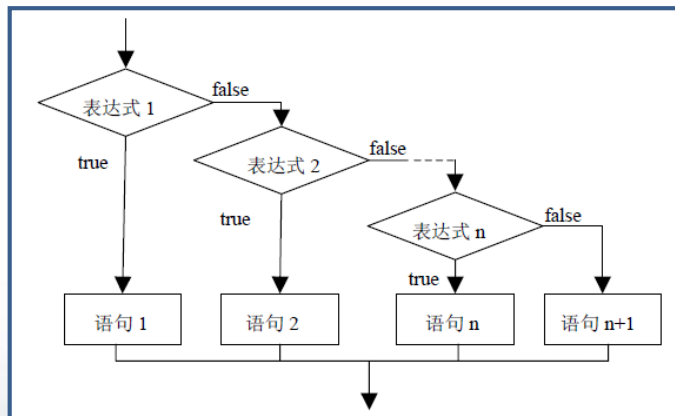
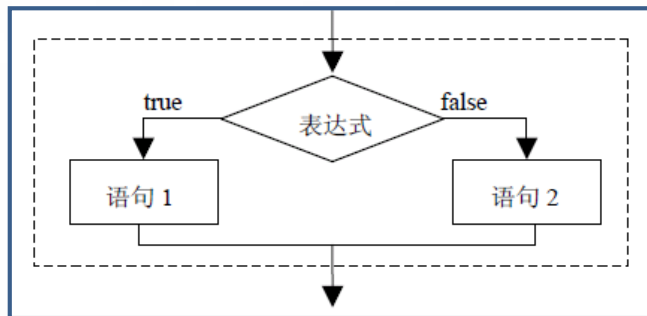




计算机程序的基本结构

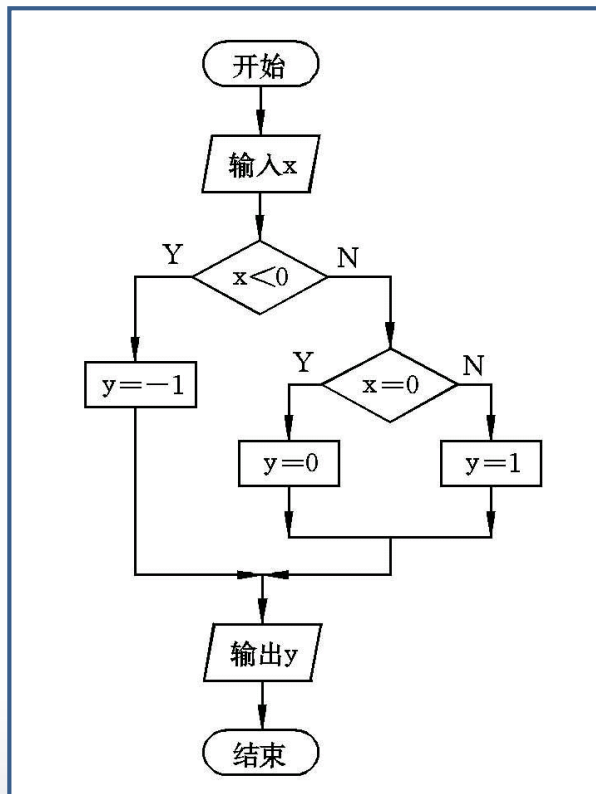
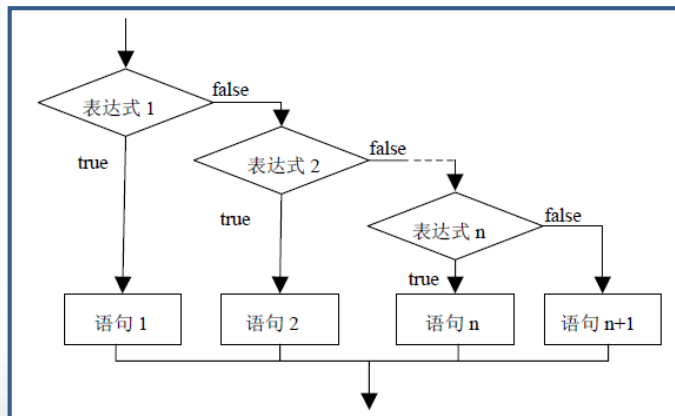
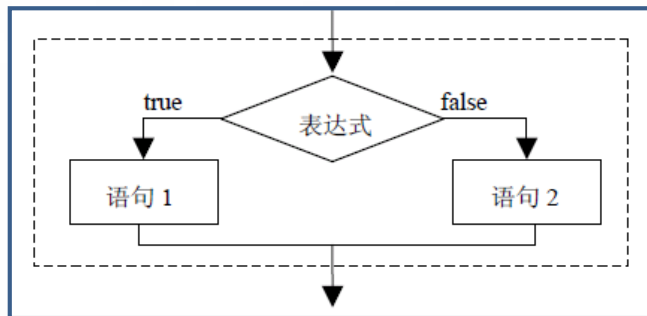
- 什么样的结构才能支持程序运行的逻辑？
- 1966年，G. Jacopini 和 C. Bohm在 “Communications of the ACM” 上发表论论文 “Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules” 。
- 从理论上证明了：任何具有单入口单出口的程序都可以用三种基本结构表达：
 - ◆ 顺序结构
 - ◆ 分支结构
 - ◆ 循环结构
- C. Bohm & G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Communications of the ACM, vol9(5) May 1966, pp 366-371.

分支语句



```
#include<iostream>
using namespace std;
int main()
{
    float weight = 0, height = 0, healthRate = 0;
    cin >> weight>>height;
    healthRate = weight / (height*height);
    if ((18 <= healthRate)&&(healthRate <= 25))
        cout << "体重适中！" << endl;
    else if ((25 < healthRate)&&(healthRate <= 30))
        cout << "超重！ 注意控制！" << endl;
    else if ((30 < healthRate)&&(healthRate <= 35))
        cout << "肥胖！ 减肥吧！" << endl;
    else if ((35 < healthRate)&&(healthRate <= 40))
        cout << "重度肥胖！ 别吃了！" << endl;
    else
        cout << "请直接拨打120！" << endl;
    return 0;
}
```

分支语句



```
#include <iostream>
using namespace std;
int main()
{
    int year = 0;
    cin >> year;
    if (year % 4 == 0)
    {
        if (year % 100 == 0)
        {
            if (year % 400 == 0)
                cout << "Y";

            else
                cout << "N";
        }
        else
            cout << "Y";
    }
    else
        cout << "N";

    return 0;
}
```



if 语句使用须知

■ 在执行 if 语句前先对表达式求解

◆ if ()内可以是任意的数值类型

(包括整型、实型、字符型、指针型数据)

● `if ('a') cout<<'a'<<endl ;`

● `if (3) cout<<"OK"<<endl ;`

◆ 若表达式的值为 0 , 按 “假” 处理

◆ 若表达式的值为 非0 , 按 “真” 处理

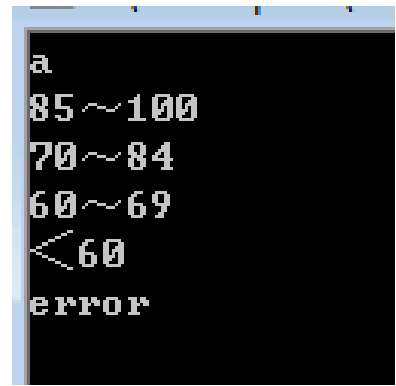
多分支语句

- Switch语句的一般形式如下：

```
switch(表达式)
{
    case 常量表达式1: 语句1;
    case 常量表达式2: 语句2;
    ... ;
    case 常量表达式n: 语句n;
    default: 语句n+1;
}
```

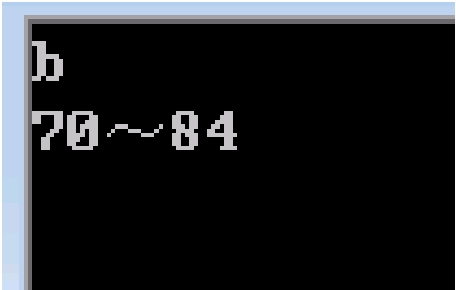
- 当表达式的值与某一个case后面的常量表达式的值相等时，就执行此case后面的语句，若所有的case中的常量表达式的值都没有与表达式的值匹配的，就执行default后面的语句。


```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin>>grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl;
        case 'b': cout << "70~84" << endl;
        case 'c': cout << "60~69" << endl;
        case 'd': cout << "<60" << endl;
        default: cout << "error" << endl;
    }
    return 0;
}
```



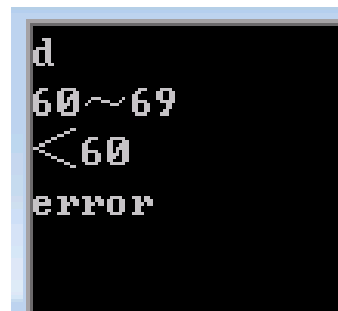
```
a
85~100
70~84
60~69
<60
error
```

```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin >> grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl; break;
        case 'b': cout << "70~84" << endl; break;
        case 'c': cout << "60~69" << endl; break;
        case 'd': cout << "<60" << endl; break;
        default: cout << "error" << endl;
    }
    return 0;
}
```



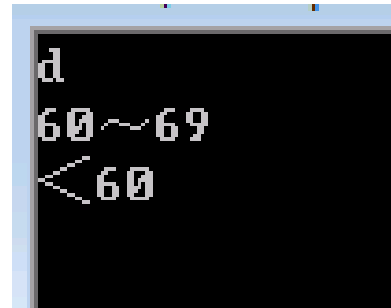
```
b
70~84
```

```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin >> grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl;
        case 'b': cout << "70~84" << endl;
        case 'c':
        case 'd':
        case 'e':
        case 'f': cout << "60~69" << endl;
        case 'g': cout << "<60" << endl;
        default: cout << "error" << endl;
    }
    return 0;
}
```

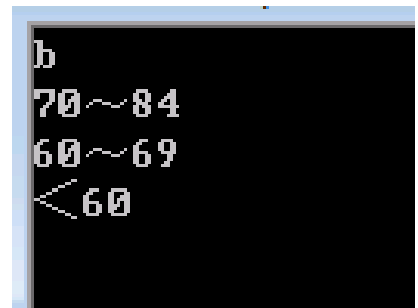


```
d
60~69
<60
error
```

```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin >> grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl;
        case 'b': cout << "70~84" << endl;
        case 'c':
        case 'd':
        case 'e':
        case 'f': cout << "60~69" << endl;
        case 'g': cout << "<60" << endl;
        break;
        default: cout << "error" << endl;
    }
    return 0;
}
```

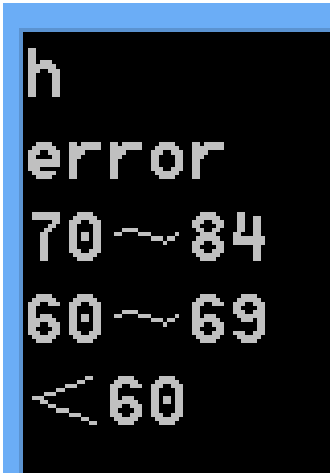


```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin >> grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl;
        default: cout << "error" << endl;
        case 'b': cout << "70~84" << endl;
        case 'c':
        case 'd':
        case 'e':
        case 'f': cout << "60~69" << endl;
        case 'g': cout << "<60" << endl;
    }
    return 0;
}
```



```
b
70~84
60~69
<60
```

```
#include<iostream>
using namespace std;
int main()
{
    char grade = 'a';
    cin >> grade;
    switch (grade)
    {
        case 'a': cout << "85~100" << endl;
        default: cout << "error" << endl;
        case 'b': cout << "70~84" << endl;
        case 'c':
        case 'd':
        case 'e':
        case 'f': cout << "60~69" << endl;
        case 'g': cout << "<60" << endl;
    }
    return 0;
}
```



h
error
70~84
60~69
<60

思考与练习

■ 问题：

- ◆ 学校要求实行成绩等级制度。现在已经有同学们的百分制成绩,要求按照百分制成绩输出相应的等级成绩. 90-100为' A', 80—90分为' B', 70—80分为' C', 60—70分为' D', 60分以下为' E'。
- ◆ 最直接的办法：
 - 输入成绩；
 - 判定成绩是否： $90 \leq \text{成绩} \leq 100$
 - 判定成绩是否： $80 \leq \text{成绩} < 90$
 -
- ◆ 如果用switch语句，如何解决？

```
#include<iostream>
using namespace std;
int main()
{
    int score, num;
    cout << "please give the score" << endl;
    cin >> score;
    num = score / 10;
    switch (num)
    {
        case 10: cout << "A" << endl;
        case 9: cout << "A" << endl;
        case 8: cout << "B" << endl;
        case 7: cout << "C" << endl;
        case 6: cout << "D" << endl;
        default: cout << "E" << endl;
    }
    return 0;
}
```



```
#include<iostream>
using namespace std;
int main()
{
    int score, num;
    cout << "please give the score" << endl;
    cin >> score;
    num = score / 10;
    switch (num)
    {
        case 10: cout << "A" << endl; break;
        case 9: cout << "A" << endl; break;
        case 8: cout << "B" << endl; break;
        case 7: cout << "C" << endl; break;
        case 6: cout << "D" << endl; break;
        default: cout << "E" << endl;
    }
    return 0;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int score, num;
    cout << "please give the score" << endl;
    cin >> score;
    num = score / 10;
    switch (num)
    {
        case 10: cout << "A" << endl;
        default: cout << "E" << endl;
        case 9: cout << "A" << endl;
        case 8: cout << "B" << endl;
        case 7: cout << "C" << endl;
        case 6: cout << "D" << endl;
    }
    return 0;
}
```



循环结构

■ C程序中的循环结构：

- ◆ 用for语句构成循环。
- ◆ 用while语句构成循环；
- ◆ 用do...while语句构成循环；
- ◆ 用goto语句和if语句构成循环；

■ 循环中止或跳出语句：

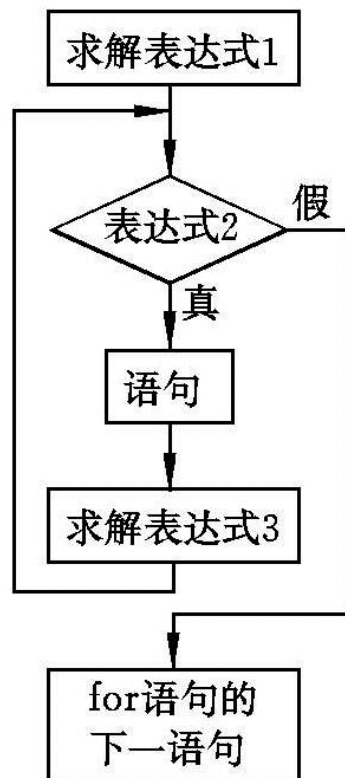
- ◆ 用continue语句结束本次循环；
- ◆ 用 break 语句 跳出 本层循环；

for 语句

■ for语句

```
for (表达式1; 表达式2; 表达式3)  
    语句;
```

```
for ( i = 0; i < 10; i++ )  
{  
    a = a*a ;  
}
```



While 语句

```
while( 条件语句 )  
{  
    执行语句;  
}
```

```
#include<iostream>  
using namespace std;  
int main()  
{  
    int i, sum = 0;  
    i = 1;  
    while (i <= 100)  
    {  
        sum = sum + i;  
        i++;  
    }  
    cout << sum << endl;  
    return 0;  
}
```

示 例

- 小红10岁,父亲33岁,问多少年之后,父亲的年龄是小红的二倍?

```
#include<iostream>
using namespace std;
int main()
{
    int ageOfHong = 10, ageOfFather = 33, count = 0;
    while (2 * ageOfHong != ageOfFather)
    {
        ageOfHong++;
        ageOfFather++;
        count++;
    }
    cout << count;
    return 0;
}
```

Do...While 语句

do

{

执行语句;

} while(条件语句);

```
#include<iostream>
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, sum = 0;
```

```
    i = 1;
```

```
    do
```

```
    {
```

```
        sum = sum + i;
```

```
        i++;
```

```
    } while (i <= 100);
```

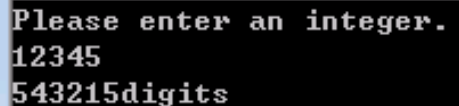
```
    cout << sum << endl;
```

```
    return 0;
```

```
}
```

示 例

```
#include<iostream>
using namespace std;
int main()
{
    int num; int count = 0;
    cout << "Please enter an integer." << endl;
    cin >> num;
    do {
        cout << num % 10;
        num = num / 10;
        count++;
    } while (num != 0);
    cout << count << "digits" << endl;
    return 0;
}
```



A screenshot of a terminal window with a black background and white text. The text shows the program's execution: a prompt 'Please enter an integer.', followed by the input '12345', and the output '543215digits'.

循环语句的嵌套

```
(1) while( )  
    {...  
    while( )  
    {...}  
}
```

```
(2) do  
    {...  
    do  
    {...}  
    while( );  
}  
while( );
```

```
(3) for(;;)  
    {  
    for(;;)  
    {...}  
}
```

```
(4) while( )  
    {...  
    do  
    {...}  
    while( );  
...  
}
```

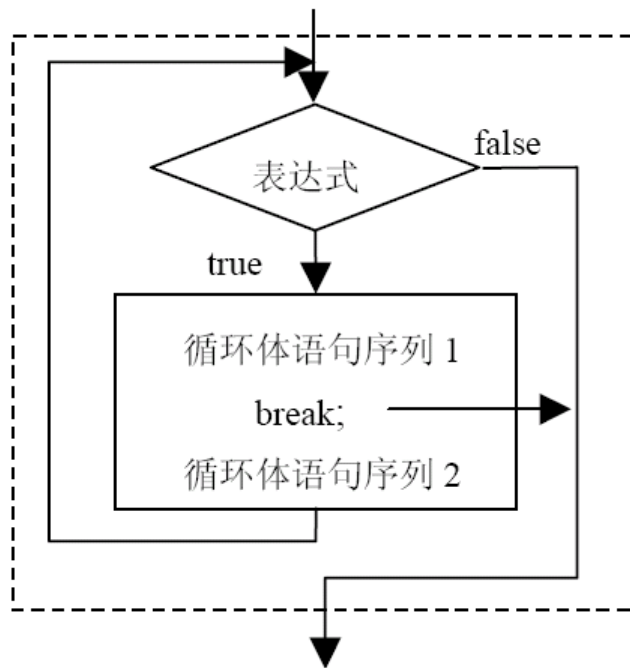
```
(5) for(;;)  
    {...  
    while( )  
    { }  
...  
}
```

```
(6) do  
    {  
...  
    for (;;)  
    { }  
}  
while( );
```

转向控制语句

■ break 语句

- ◆ 在switch 语句、while 语句、do-while 语句、for 语句中使用;
- ◆ 以跳出switch语句或内层循环，继续执行逻辑上的下一条语句。



示例

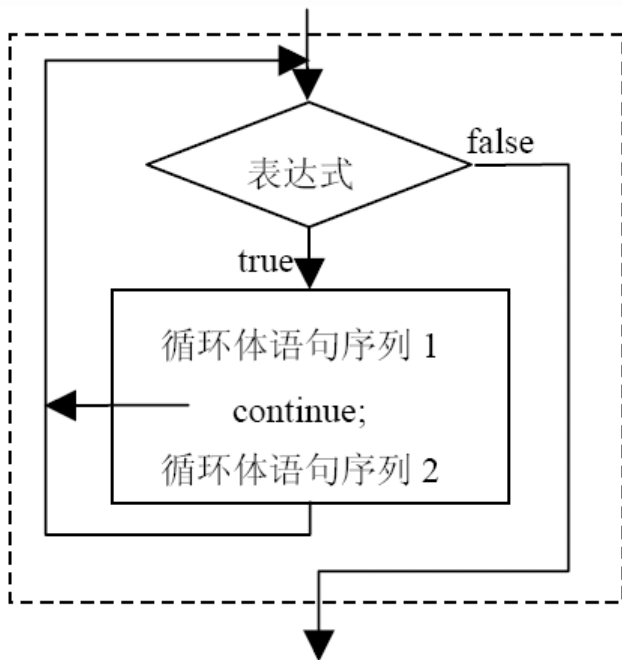
```
#include <iostream>
using namespace std;
int main()
{
    int n = 0;
    for ( ; ; )
    {
        cin >> n;
        if (n == 0)
            break;
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int n = 0;
    while (true)
    {
        cin >> n;
        if (n == 0)
            break;
    }
    return 0;
}
```

转向控制语句

■ continue 语句

- ◆ 用于循环语句中;
- ◆ 结束本次循环，接着立即测试循环控制表达式，判断是否继续执行下一次循环。



示 例

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, counter = 0;
```

```
    for (n = 1; n <= 100; n++)
```

```
    {
```

```
        if (n % 3 == 0 || n % 5 == 0 || n % 7 == 0)
```

```
            continue;
```

```
        cout << n << '\t';
```

```
        counter++;
```

```
        if (counter % 10 == 0)
```

```
            cout << endl;
```

```
    }
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

输出1 ~ 100 之间所有数，除非它能被3
或5 或7 整除的整数，每行显示10 个。

1	2	4	8	11	13	16	17	19	22
23	26	29	31	32	34	37	38	41	43
44	46	47	52	53	58	59	61	62	64
67	68	71	73	74	76	79	82	83	86
88	89	92	94	97					

直抒胸臆

**——选择能够直接表达你的逻辑思路
的控制语句！**

早期的程序控制方法

■ Goto 语句

◆ 无条件转向语句

◆ 它的一般形式为

goto 语句标号 ;

● 语句标号：标识符，定名规则与变量名相同；

◆ 例如：

● goto label-1;



● goto 123 ;



Goto 语句的例子

汇编语言

```
load 0 a  数据装入寄存器0
load 1 b  数据装入寄存器1
loop:
mult 0 1  寄存器0与1的数据相乘
load 1 c  数据装入寄存器1
add 0 1   寄存器0与1的数据加
goto loop
save 0 d  保存寄存器0里的数据
```

C++ 语言

```
#include <iostream>
using namespace std;
int main()
{
    int i = 0, sum = 0;
    i = 1;

loop:
    sum = sum + i;
    i++;
    goto loop;
    cout << sum << endl;
    return 0;
}
```


关于 Goto 语句的讨论

■ 著名的荷兰教授 E. W. Dijkstra

◆ 1965年，IFIP (International Federation for Information Processing) 会议上，Dijkstra提出“Go To语句可以从高级语言中取消”，“一个程序的质量与程序中所含的Go To语句的数量成反比”。

◆ 但是，Dijkstra讲话的影响很小，当时人们正广泛地使用FORTRAN，而Go To语句则是FORTRAN的支柱。



Algol60设计实现者之一
GoTo有害论提出者
信号量理论提出者
最短路径算法提出者
THE操作系统设计者
程序正确性证明推动者



关于 Goto 语句的讨论

- 60年代末至70年代，关于goto语句的争论非常激烈
- **正方**：从高级语言中去掉goto语句
 - ◆ 包含goto语句的程序难以阅读，难以查错；
 - ◆ 去掉goto语句后，可以直接从程序结构上反映程序的运行过程。使程序的结构清晰、便于阅读，便于查错，而且也有利于程序正确性的证明。
- **反方**：goto语句无害，应该保留
 - ◆ goto语句使用起来比较灵活，而且有些情形能够提高程序的效率。
 - ◆ 如果一味强调删除goto语句，有些情形反而会使程序过于复杂，增加一些不必要的计算量。

关于 Goto 语句的讨论

- Donald E. Knuth (高德纳)
- 1974年, D.E.Knuth对于goto语句的争论作了全面的公正的评述:
 - ◆ 不加限制地使用goto语句, 特别是使用往回跳的goto语句, 会使程序的结构难于理解, 这种情形应该尽量避免使用goto语句;
 - ◆ 为了提高程序的效率, 同时又不破坏程序的良好结构, 有控制地使用一些goto语句是有必要的。
- “有些情形, 主张废除转向语句, 有些情形我主张引进转向语句。”

