



《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

2010年11月4日



北京大学

程序的基本结构——顺序结构

■ 顺序结构的程序

- ◆ 程序严格按语句书写顺序依次逐条进行
- ◆ 低级语言的执行顺序:

机器语言

00000001000000001000	数据装入寄存器0
00000001000100001010	数据装入寄存器1
00000101000000000001	寄存器0与1的数据乘
00000001000100001100	数据装入寄存器1
00000100000000000001	寄存器0与1的数据加
00000010000000001110	保存寄存器0里的数据

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
mult 0 1	寄存器0与1的数据乘
load 1 c	数据装入寄存器1
add 0 1	寄存器0与1的数据加
save 0 d	保存寄存器0里的数据





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.





再谈 分支 与 循环



北京大学

分支语句——复杂的if 语句

■ if ... else ... 语句

if (条件表达式)

 单一语句;

else if (条件表达式)

 单一语句;

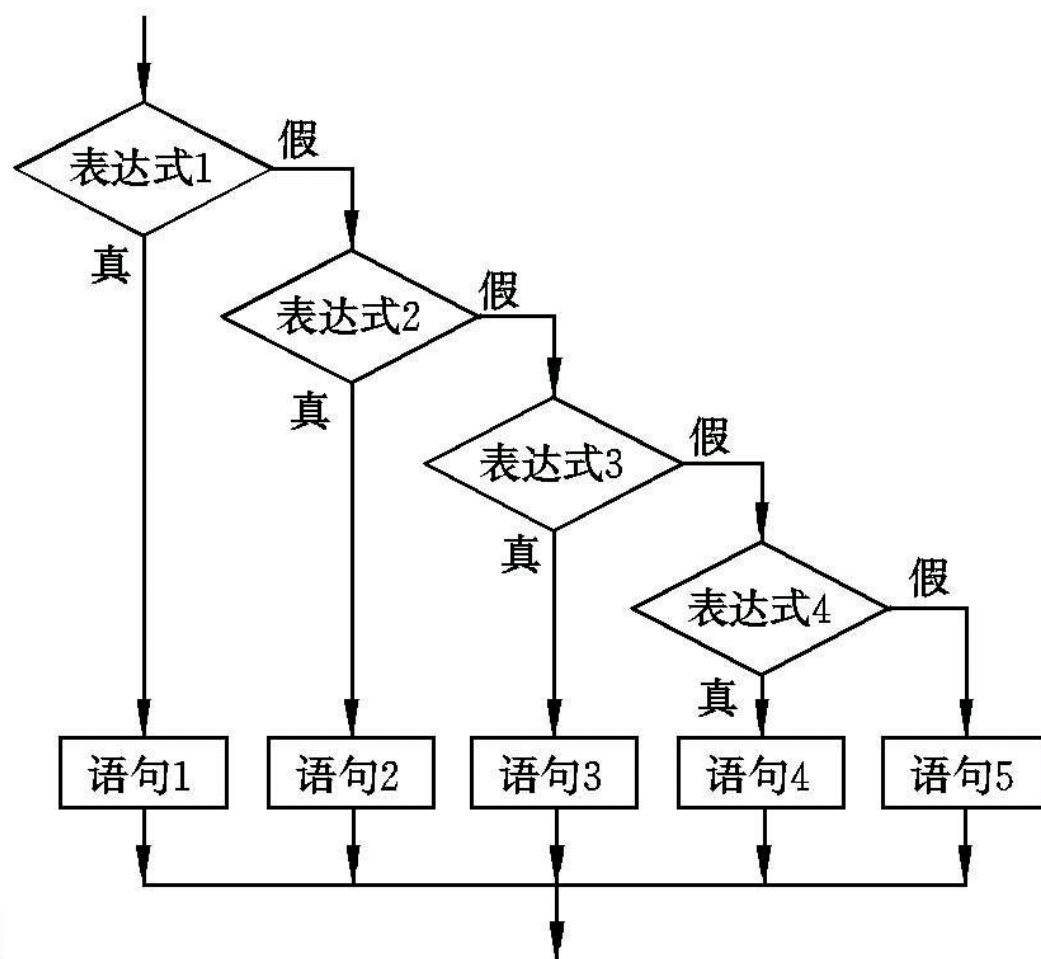
else if (条件表达式)

 单一语句;

...;

else

 单一语句;



北京大学

示 例

```
#include<iostream>
using namespace std;
int main( )
{
    float weight = 0, standard_weight = 0, rate = 0;
    cin>>weight;
    standard_weight = (weight - 150)*0.6+50;
    rate =(weight-standard_weight)/ standard_weight ;
    if((-0.1 <= rate)&&(rate <= 0.1))
        cout<<"体重适中！ "<<endl;
    else if((0.1 <= rate)&&(rate <= 0.2))
        cout<<"超重！ 注意控制！ "<<endl;
    else if((0.2 < rate)&&(rate <= 0.4))
        cout<<"肥胖！ 减肥吧！ "<<endl;
    else
        cout<<"请直接拨打120！ "<<endl;
    return 0;
}
```




关于 if 语句的说明 (1)

- 在执行 if 语句前先对表达式求解
 - ◆ 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
 - ◆ 表达式的类型不限于逻辑表达式，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;`
- 注意: “;”号的使用!!!



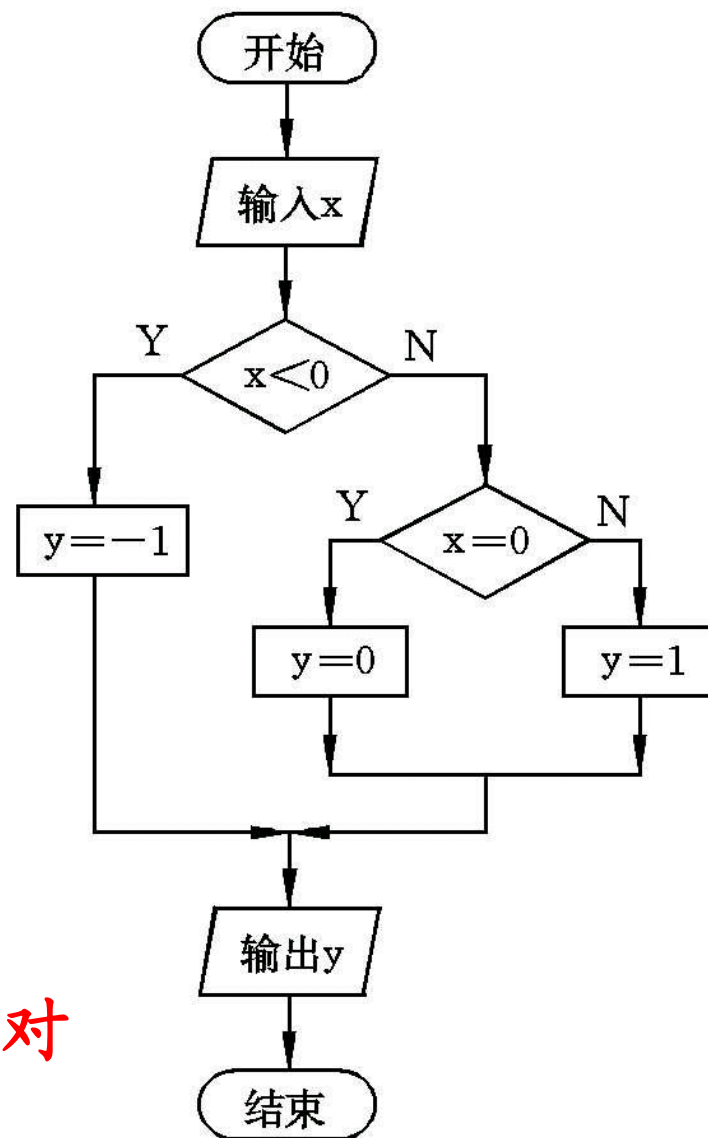
关于 if 语句的说明 (2)

■ if 语句可以嵌套

```
if (x < 0)    y=-1;  
else  
    if (x==0) y=0;  
    else     y=1;
```

```
if (x < 0)    y=-1;  
else  
    if (x==0) y=0;  
    else     y=1;
```

else总是与它上面的最近的if配对





多分支选择语句

- 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。





多分支选择语句的说明

(1) switch后面括弧内的“表达式”，ANSI标准允许它为任何类型。

(2) 执行完一个case后面的语句后，流程控制转移到下一个case**继续执行**。

“case常量表达式”只是起语句标号作用，并不是在该处进行条件判断。在执行Switch语句时，根据Switch后面表达式的值找到**匹配的入口标号**，从此标号开始执行下去，不再进行判断。

(3) 多个case可以共用一组执行语句。



例子

```
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;
}
```





思考与练习

■ 课堂练习

- ◆ 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: $90 \leq \text{成绩} \leq 100 \dots\dots$
 - 判定成绩是否: $80 \leq \text{成绩} < 90 \dots\dots$
 - $\dots\dots$
- ◆ 如果用switch语句, 如何解决?



思考与练习

```
int main( )
{
    int score,num;
    char grade;
    cout<<"please give the score"<<endl;
    cin>>score;
    num = score / 10;
    switch(num)
    {
        case 10: grade = 'A';  break;
        case 9 : grade = 'A';  break;
        case 8 : grade = 'B';  break;
        case 7 : grade = 'C';  break;
        case 6 : grade = 'D';  break;
        default : grade = 'E' ;
    }
    cout<<score<<" "<<grade<<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 (表达式1; 表达式2; 表达式3) 语句;

■ 执行过程:

(1) 先求解表达式1。

(2) 再求解表达式2,

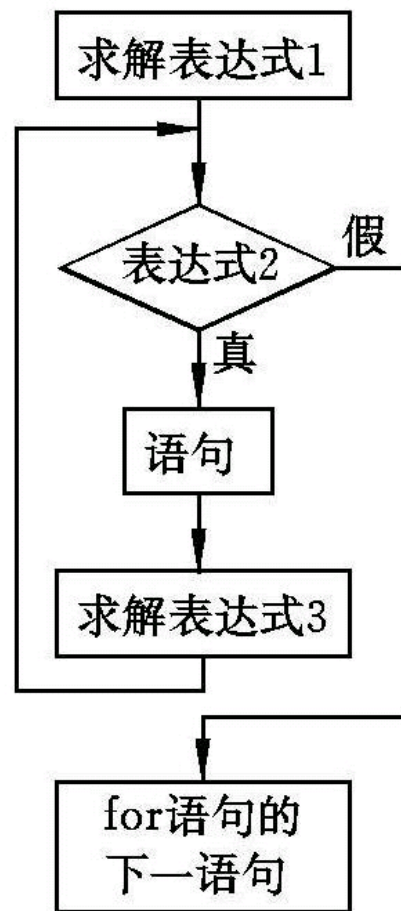
◆ 若其值为真(值为非0), 则执行for语句中指定的内嵌语句, 然后执行下面第(3)步。

◆ 若为假(值为0), 则结束循环, 转到第(5)步。

(3) 求解表达式3。

(4) 转回上面第(2)步骤继续执行。

(5) 循环结束, 执行for语句下面的一个语句。



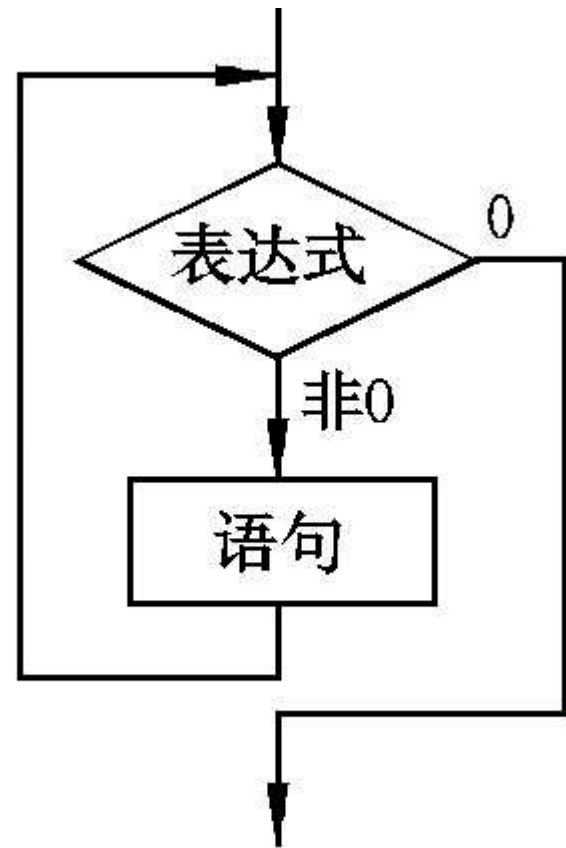
北京大学

While 语句

■ while语句一般形式如下：

while (表达式) 语句；

```
void main()
{
    int i, sum=0;
    i=1;
    while (i<=100)
    {
        sum = sum + i;
        i++;
    }
    cout<<sum<<endl;
}
```





While 语句说明

两点:

- (1) 循环体如果包含一个以上的语句，应该用**大括弧**括起来，以复合语句形式出现。如果不加花括弧，则while语句的范围只到while后面**第一个分号**处。
- (2) 在循环体中应有使循环**趋向于结束**的语句。若无此语句，则i的值始终不改变，循环**永不结束**。



北京大学

示 例

```
void main()
{
    const int N = 99;
    int counter, sum = 0;
    counter = 2;
    while(counter <= N)
    {
        sum += counter;
        counter += 2;
    }
    cout<<"The total is "<<sum<<endl;
}
```

示 例

小红今年12岁,父亲比她大20岁,问多少年之后,父亲的年龄是小红的二倍?

```
void main( )
```

```
{ int ageOfHong = 12, ageOfFather = 32, count=0;
```

```
while(2*ageOfHong != ageOfFather)
```

```
{
```

```
    ageOfHong++;
```

```
    ageOfFather++;
```

```
    count++;
```

```
}
```

```
cout<<count;
```

```
}
```



Do ... while语句

- do while语句的特点是**先执行**循环体，**然后判断**循环条件是否成立。其一般形式为

do

循环体语句

while (表达式);

- 执行方式:

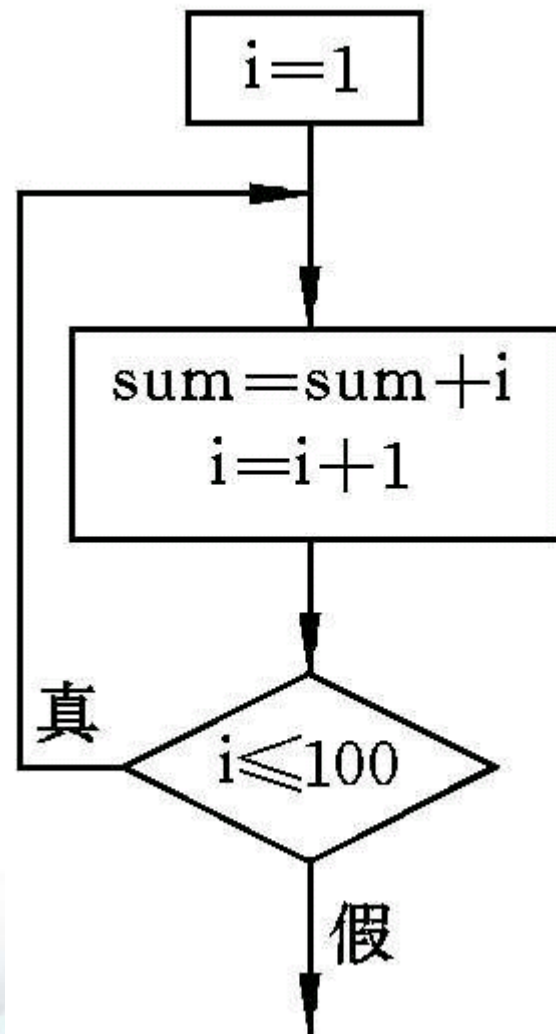
- ◆ 先执行一次指定的循环体语句，然后判别表达式，当**表达式**的值为**非0**时，返回重新执行**循环**体语句，如此反复，直到**表达式**的值**等于0**为止，此时**循环结束**。



北京大学

Do ... while语句

```
void main()
{
    int i, sum=0;
    i=1;
    do
    {
        sum = sum + 1;
        i++;
    }
    while ( i <= 100 );
    cout<<sum<<endl;
}
```



示 例

```
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;
}
```

循环语句的嵌套

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

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

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

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

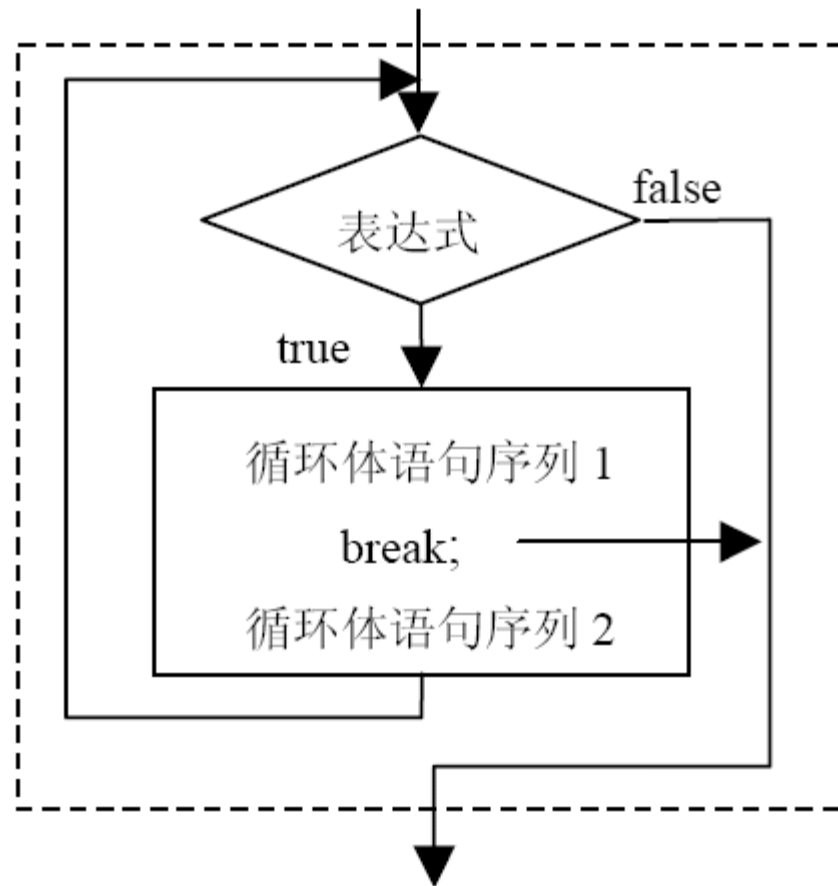
```
(5) for(;;)
    { ...
      while( )
        { }
    ...
    }
```

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

转向控制语句

■ break 语句

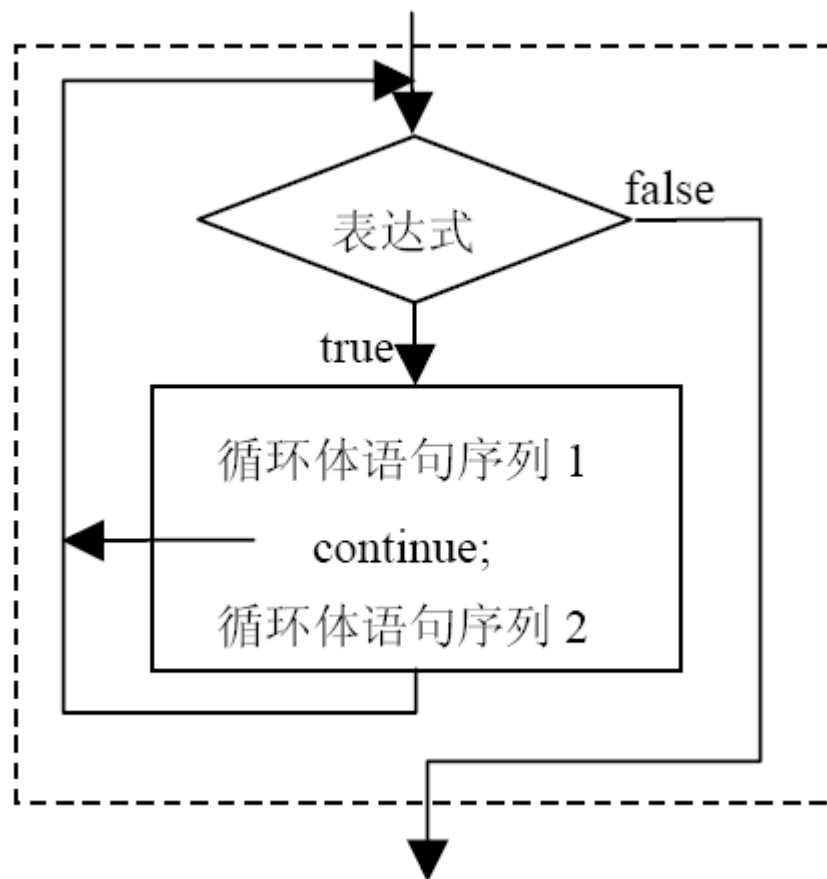
- ◆ 在switch 语句、while 语句、do-while 语句、for 语句中使用break 语句，以跳出switch 语句或内层循环，继续执行逻辑上的下一条语句。



转向控制语句

■ continue 语句

- ◆ 用于循环语句中，作用是结束本次循环，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>
void 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;
}
```




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; ✓



北京大学



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

```
int main( )
{
    int i, sum=0;
    i=1;
loop:  sum=sum+i;
        i++;
        goto loop;
        cout<<sum<<endl;
        return 0;
}
```





Goto 语句和 if 语句

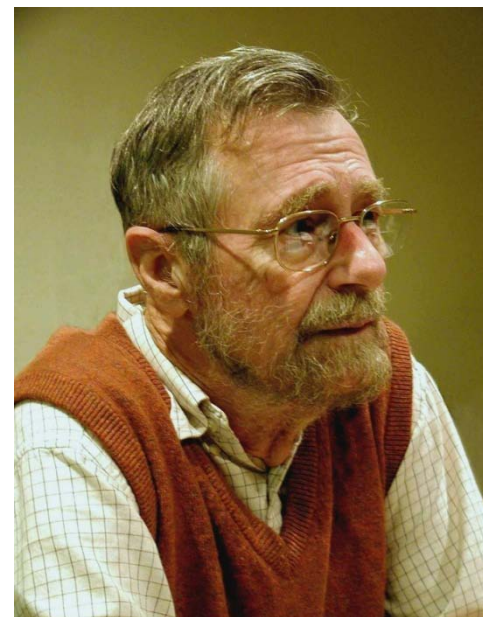
```
main( )
{
    int i,  sum=0;
    i=1;
loop: if (i<=100)
    {
        sum=sum+i;
        i++;
        goto loop;
    }
    printf("%d", sum);
}
```



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
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

- ◆ 1968年，Dijkstra给ACM通讯写了一篇短文“Go To Statement Considered Harmful”，该文后改成信件形式刊登，成为著名的“Go To Letter”。
- ◆ Dijkstra在信中建议：“Go To语句太容易把程序弄乱，应从一切高级语言中去掉；只用三种基本控制结构就可以写各种程序，而这样的程序可以由上而下阅读而不会返回”。
- ◆ 在整个计算科学的范围内，引发了关于Goto语句的讨论。





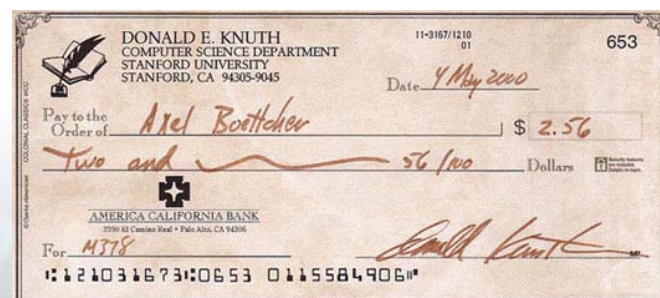
关于 Goto 语句的讨论

- 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。”





好好想想，有没有问题？

谢谢！



清华大学