

第二章 程序控制结构

第2章 程序控制结构

语句是程序的基本语法成分。程序设计语言的语句按功能可以分成三类：

> 声明语句

指示编译器分配内存，或者提供程序连接信息

> 操作语句

描述对数据的处理

表达式是按照求值规则，表达一个值的式子

> 控制语句

用于控制程序的执行流程。

所有程序都只能包含三种控制结构：

顺序结构、选择结构和循环结构

2.1 表达式

2.1.1 运算符

运算符是以简洁的方式表达对数据操作的符号

算术运算符	+ - * / % ++ --
关系运算符	> < == >= <= !=
逻辑运算符	! &&
位运算符	<< >> ~ ^ &
赋值运算符	= 及扩展的复合运算符
条件运算符	? :
逗号运算符	,
指针运算符	* &
求字节运算符	sizeof
强制类型转换符	类型
分量运算符	. ->
下标运算符	[]
其他	() :: new delete

优先级	运算符	结合性
1	() [] -> ::	LR
2	! ~ + - ++ -- & * 类型 sizeof new delete . -> *	RL
3	* / %	LR
4	+ -	LR
5	<< >>	LR
6	< <= >= >	LR
7		LR
8	== !=	LR
9	&	LR
10	^	LR
11		LR
12	&&	LR
13		LR
14	? :	RL
15	= *= /= += -= = <<= >>=	RL
16	,	LR

记忆方法：算、关、逻、条、赋、逗，先算括号

> 运算符是以简洁的方式表达对数据操作的符号

• 单目运算符

运算符 右操作数

例如：-123 +500

• 双目运算符

左操作数 运算符 右操作数

例如：a*3 x>y

• 三目运算符

条件运算： 操作数1 ? 操作数2 : 操作数3

例如：a ? b : c

2.1.2 算术表达式

基本算术运算符有：

+	加法，或单目求正
-	减法，或单目求负
*	乘法
/	除法
%	求模(求余)
sizeof	求存储字节

/	8 / 2 ;	< 4, int >
	6 / 4 ;	< 1, int >
	4 / 5 ;	< 0, int >
	-11 / 2 ;	< -5, int > 或 < -6 , int >
	6 / 0 ;	< undef, int >
%	10 % 3 ;	< 1, int >
	23 % 4 ;	< 3, int >
	5 % 0 ;	< undef, int >

注意整除和求模运算，%无法对浮点数操作

C++ 为其提供自增和自减算符

	算 符	前 缀 式	后 缀 式	等 价 语 句
自 增	++	++ i	i ++	i = i + 1
自 减	--	-- i	i --	i = i - 1

注：

• 自增、自减算符的运算对象只能是整型变量，不能为常量或表达式；

例：

5++

++(a++)

(x+y)--

错误

不是整型变量

自增式和自减式作为独立的表达式，前缀式和后缀式没有区别；
但作为表达式右值时：

(1) 前缀式 先增值后引用
例： x = ++ i 相当于 i = i + 1； x = i；

(2) 后缀式 先引用后增值
例： x = i ++ 相当于 x = i； i = i + 1；

```
#include <iostream>
using namespace std;
int main()
{
    int a1 = 1, a2=1, a3=1, a4=1;
    cout << ++a1 << endl<< a2++ << endl;
    int b = a3++;
    int c = ++a4;
    cout << a3 << " " << b << endl;
    cout << a4 << " " << c << endl;
}
```

Microsoft Visual Studio 调试

2
1
2 1
2 2

从这里可以看出来，前缀和后缀是有所不同的

```
cout << b++ + ++c<<" " << (b++) + (++c) << endl;
```

这是允许的

表达式求值之前，要对操作数进行必要的类型转换：

➤ 目的： (1) 将短数扩展为机器处理的长度
(2) 使运算符两端的操作数具有相同的类型

➤ 原则： (1) 开始运算前，扩展数据长度：

char、short ➔ int
unsigned char、unsigned short ➔ unsigned int
float ➔ double
 long
 unsigned long

参与算术运算的只有5种类型数据

(2) 算符两端运算量类型不同时：

· “向高看齐”，向表达能力强的类型转换

· 逐个算符转换

(3) 赋值转换具有强制性

强制类型转换

用类型符对表达式值转换成所需类型，一般形式为：

(类型) (表达式)

类型 (表达式)

(类型) 表达式

赋值时的类型转换和用类型符实现的类型转换是强制性

```
#include <iostream>
using namespace std;
int main()
{
    double a = 3.1415;
    void* p = &a;
    cout << &a<<" "<<(int*)p << endl;
    cout <<a<<" "<<* (int*)p << endl;
}
```

Microsoft Visual Studio 调试

0000000804AFF988 0000000804AFF988
3.1415 -1065151889

可以将指针从 void*类型转化为 int*等

这样不会把double类型强制转化为int类型

而是用int的读取方式来读取double类型的数据

是的

int是补码方式存储，double是ieee754方式

学长这个是因为什么呀

这样读取就会产生错误是吗？

2.1.3 逻辑表达式

- 逻辑表达式用于判断运算
- 逻辑表达式的值只有两个：
 - 表达式 成立 为 true (1)
 - 表达式 不成立 为 false (0)
- 表达式求值过程中，非0值都作为逻辑真
- 构成逻辑表达式的运算符有关系运算符和逻辑运算符

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double a = 1;
6      if (0.4&&-1)
7      {
8          cout << a << endl;
9      }
10 }
```

Microsoft Visual Studio 调试 1

逻辑表达式中非 0 就是 1

关系运算

关系运算符:

>	大于	>=	大于等于	<	小于
<=	小于等于	=	等于	!=	不等于

优先级: 低于算术运算类，高于赋值类

结合性: 从左向右结合

```
cout << 3 < 4 << endl;
//注意运算优先级
```

注意不存在 1<=x<=2 这样的写法

若有 **int x = 2, y = 3, z;**

则 **x == y** 值为 **0**

x != y 值为 **1**

z = 3 - 1 >= x + 1 <= y + 2 **z 的值为 1**

↓ ↓ ↓

z = 2 >= 3 <= 5

↓

z = 0 <= 5

↓

z = 1

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << (3 < 4) << endl;
6      cout << (3 < 4 > 4) << endl;
7      cout << (3 < 4 > 4 < 5) << endl;
8  }
```

Microsoft Visual Studio 调试 1 0 1

从左往右进行运算，但这样的表示并没有什么实际意义

(3) 关系表达式的结果为整数，所以也可以看作整型表达式

统计符合条件的事件数

例(1)

```
int score1, score2, score3, score4, score5 ;
int  excellent=0;
//.....
excellent=int(score1>=85)+int(score2>=85)+int(score3>=85)
+int(score4>=85)+int(score5>=85) ;
cout<<"excellent= "<<excellent<<endl;
```

应用：

(4) 字符数据按ASCII码值进行比较

例：

'a' > 0	值为 1 (true)
'A' > 100	值为 0 (false)

要注意计算机浮点数比较精度问题

1. 关系运算

注意关系运算的正确使用：

(5) 实数比较的误差

例：

```
1.2345678901234567897 == 1.2345678901234567898  值为 1 (true)
```

这是数据有效位数引起的误差。

讨论： 实数比较宜采用求误差值形式：

```
fabs ( x - y ) < ε
```

例如： fabs(x - y) < 1e-5

逻辑运算

逻辑运算符：		
&&	逻辑与	左结合
	逻辑或	左结合
!	逻辑非	右结合

逻辑运算规律：	
E1 && E2	当表达式 E1 的值为 0 时，不再对 E2 求值
E1 E2	当表达式 E1 值为非 0 时，不再对 E2 求值


```
int main()
{
    int i = 1, j = 1;
    i++ || ++j;
    cout << i << " " << j << endl;
    int m = 1, n = 2;
    m-- && (n = n + 1);
    cout << m << " " << n << endl;
}
```

Microsoft Visual Studio 调试 × + v

2 1
0 3

防止出现逻辑错误可以加上括号

2.1.4 赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：变量 = 表达式

应用：可以利用辅助变量交换两个变量的值

赋值运算带强制类型转换

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a;
    cout << a << endl;
}
```

Microsoft Visual Studio 调试 × + v

2.1
2

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int a=1,b=2,c=0;
6     c = a = b;
7     cout << a<<" "<<b<<" "<<c << endl;
8 }
9
10
11 2 2 2
12
```

Microsoft Visual Studio 调试 × + v

赋值表达式有限级低（高于逗号表达式）右结合

```
int a=1,b=2,c=0;
c +2= a = b;
```

错误，不知道向哪个量赋值

复合赋值运算

双目算符 op 的表达式：A = A op B

可以缩写成：A op = B

C++ 提供的10个复合赋值运算符：

+=

-=

*=

/=

%=

<<=

>>=

&=

^=

|=

用于位运算

2.1.5 条件表达式

条件运算表达式格式：

$$E1 \text{ ? } E2 \text{ : } E3$$

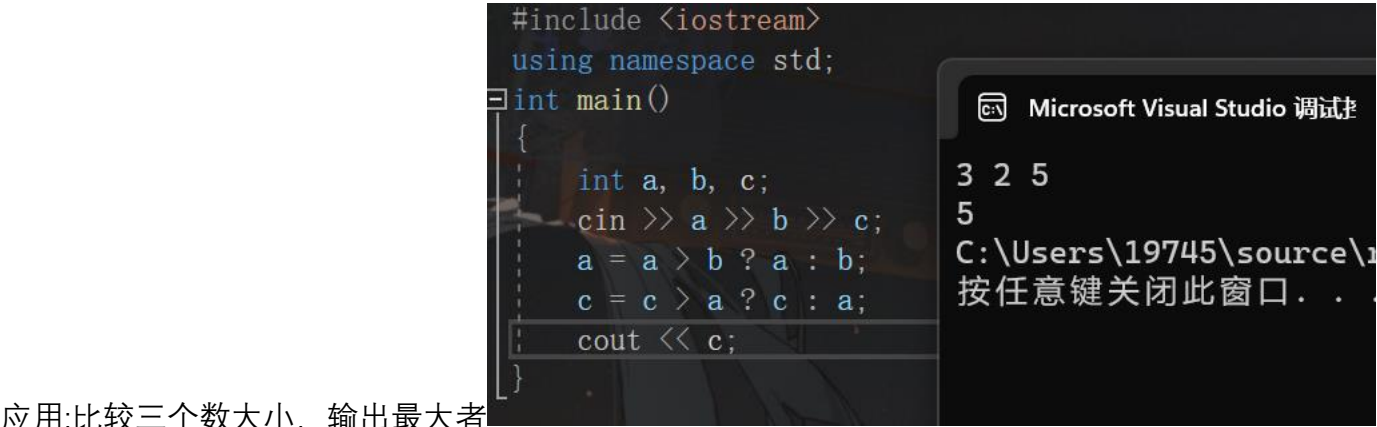
其中 ? : 称为条件运算符，是C++中唯一的三目运算符；

 E1、E2、E3 为表达式。

功能 根据 E1 的值，决定计算 E2 或 E3。

 若 E1 的值为true（非0），条件表达式的值为 E2 的值；

 若 E1 的值为false（0），条件表达式的值为 E3 的值。



应用:比较三个数大小，输出最大者

2.1.6 逗号表达式

逗号表达式格式：

$$E1 \text{ , } E2 \text{ , } E3 \text{ , } En$$

其中 , 称为逗号运算符，运算级别最低；

 E1、E2、E3、..... En 为表达式。

功能 顺序执行 E1、E2、E3 En，最后获得逗号表达式的值。

注意 , 也用于分隔符，如：

$$\text{int func (int a, int b , double x);}$$

要注意逗号表达式从左往右计算，输出最后一个表达式的值

2.2 选择控制（条件语句）

1、数据溢出问题

- INT_MAX 和 INT_MIN 等最大最小值被定义在<climits> <float>中
- 规范解决方案：判断 a+b 是否溢出，利用 INT_MAX-a>b &&INT_MIN-a<b

```
object4 (全局范围)
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << INT_MAX << " " << INT_MIN << endl;
6  }
```

Microsoft Visual Studio 调试 × + ▾

2147483647 -2147483648

2、if 语句

格式：if (a>0) (注意 if 后面不应该加；否则会有逻辑错误)

```
{
    cout << 1<<endl;
}
```

(添加的语句可以加在花括号中，若无花括号则只有第一句加在 if 语句中)

在C++中，执行以下语句将输出（ ）。
int grade = 90;
if (grade >= 60);
 cout << "Passed";

☒ A Passed
☐ B 编译错误
☐ C 链接错误
☐ D 无输出

If else 语句

```
if ( b>a )
    max = b ;
else max = a ;
cout << "max = " << max << endl;
```

注意：if else 的结尾不应该加 ；

```
if (2>1);
{
}
else
{
}
```

即： (语法错误)

```
#include <iostream>
using namespace std;
int main()
{
    if (2>1)
    {
        cout << "yes" << endl;
    }
    else;
    {
        cout << "no" << endl;
    }
}
```

Microsoft Visual Studio 调试 ×

yes
no

C:\Users\19745\source\repo
按任意键关闭此窗口...

(逻辑错误)

2.1.1 if 语句

// 例2-1 计算货物运费

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{ double t, p, w, s;
  cout << "Please input weight( ton ) : ";
  cin >> w;
  cout << " Please input distsnce(kilometre ) : ";
  cin >> s;
  if( s < 100 ) p = 30;
  else if ( s < 200 ) p = 27.5;
    else if ( s < 300 ) p = 25;
      else if ( s < 400 ) p = 22.5;
        else p = 20;
```

想一想：
几个 else 分支语句的次序可以改变

```
graph TD
    Start(( )) --> D1{s < 100}
    D1 -- true --> P30[P = 30]
    D1 -- false --> D2{s < 200}
    D2 -- true --> P275[P = 27.5]
    D2 -- false --> P25[P = 25]
    D2 -- false --> P225[P = 22.5]
    D2 -- false --> P20[P = 20]
    P30 --> T[P = 30]
    P275 --> T
    P25 --> T
    P225 --> T
    P20 --> T
    T --> End(( ))
    style T fill:#fff,stroke:#333,stroke-width:1px
```

也可以写六条单分支语句，但这样步骤多计算慢。

解决实际问题

2.1.1 if 语句

例 2-2 输入三个整数，按从小到大顺序输出。

数据排序。先找出最小值，放在a中，然后找次小值，放在b中：

- 对 a、b 进行比较，把小值放于 a ；
`if a > b a ⇔ b // a < b`
- 对 a、c 进行比较，把小值放于 a ；
`if a > c a ⇔ c // a < c, a < b`
- 对 b、c 进行比较，把小值放于 b ；
`if b > c b ⇔ c // a < b && b < c`
- 输出 a, b, c 的值。

第62页

2.1.1 if 语句

例 2-2 输入三个整数，按从小到大顺序输出。

```
#include <iostream>
using namespace std;
int main()
{ int a, b, c, t;
  cout << " Please input three integer numbers: ";
  cin >> a >> b >> c;
  if ( a > b )
    { t = a; a = b; b = t; }
  if ( a > c )
    { t = a; a = c; c = t; }
  if ( b > c )
    { t = b; b = c; c = t; }
  cout << a << " " << b << " " << c << endl;
}
```

注意语句块结构

第65页

3、if 语句的嵌套（摇摆）问题

else 总是和离他（逻辑意义上）最近的 if 结合

2.1.1 if 语句

2. if 语句的嵌套

分析 S1 和 S2 的执行条件：

```
if ( E1 )
  if ( E2 ) S1
  else S2

if ( E1 )
  { if ( E2 ) S1 }
else S2
```

注意括号的作用

E1 && E2 执行 S1
E1 && !E2 执行 S2
!E1 执行 S2

第66页

4、良好的编程习惯

if...else

第90页

```
if ( condition )
{
    statement;
    ....
    statement;
}
else
{
    statement;
    ....
    statement;
}
```

90

Most programmers write the preceding statement as

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
    cout << "A";
else if ( studentGrade >= 80 ) // 80-89 gets "B"
    cout << "B";
else if ( studentGrade >= 70 ) // 70-79 gets "C"
    cout << "C";
else if ( studentGrade >= 60 ) // 60-69 gets "D"
    cout << "D";
else // less than 60 gets "F"
    cout << "F";
```

嵌套格式最好都并排，如果嵌套数量少也可以一级一级来。

5、switch 语句

可以解决不同输入对应不同输出结果的问题（相当于代替了多个 if 语句）

switch 是一条一条按顺序执行的，不同于 if 直接跳走

2.1.2 switch语句

第107页

例2-5 根据考试成绩的等级打印出百分制分数段。

```
#include <iostream>
using namespace std;
int main ()
{ char grade;
  cout << "Input grade of score (a_d): " << endl;
  cin >> grade;
  switch ( grade )
  { case 'a': cout << "85_100\n";
    case 'b': cout << "70_84\n";
    case 'c': cout << "60_69\n";
    case 'd': cout << "<60\n";
    default : cout << "error\n";
  }
}
```

输入 a

输出 85_100
70_84
60_69
60

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a;
    switch (a)
    {
        case 1:cout << 1 << endl;
        default:cout << "no" << endl;
        case 2:cout << 2 << endl;
    }
}
```

Microsoft Visual S

3
no
2

C:\Users\19745
按任意键关闭此

但如果改为 case 语句；后面加 break；就可以做到跳出 switch 结构，起到控制流程的作用

```
#include <iostream>
using namespace std;
int main ()
{ char grade;
  cout << " Input grade of score (a_d or A__D): " << endl;
  cin >> grade;
  switch ( grade )
  { case 'a' :
    case 'A': cout << "85_100\n"; break;
    case 'b' :
    case 'B': cout << "70_84\n"; break;
    case 'c' :
    case 'C': cout << "60_69\n"; break;
    case 'd' :
    case 'D': cout << "<60\n"; break;
    default : cout << "error\n";
  }
}
```

'a' 或 'A' 共同执行一个语句

输入 a

输出 85_100

输入 B

输出 70_84

116

Switch 可以做到多个 if 嵌套结构的问题

2.1.2 switch语句

第113页

switch 结构嵌套

嵌套结构的 case 标号与外层无关

```
switch (i)
{
    case 1 : .....
    case 2 :
        switch (j)
        {
            case 1 : .....
            case 2 : .....
            .....
        }
    case 3 : .....
}
```

113

Switch 语句的嵌套方法，互不干扰

2.1.2 switch语句

第114页

if 与 switch 语句互换

逻辑表达式

```
if (E1)
    S1;
else S2;
```

↔

```
switch (int (E1))
{
    case 1 : S1; break;
    case 0 : S2;
}
```

114

If 语句和 switch 语句的相互交换，但要注意不是每次都是 int（E1）

```
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cin >> a>>b;
    switch (a>b)
    {
        case 1:cout << a << endl;
        case 0:cout << b << endl;
    }
}
```

Microsoft Visual S
2 1
2
1
C:\Users\19745
按任意键关闭此

逻辑表达式可以不用加 int，返回值都是 0 或 1

6、switch 语句的注意事项

单选题 20分

第118页

在C++中，已知char grade ='A' ;执行以下语句则输出（ ）。

```
switch ( grade )
{
    case 'A' : cout << "A";
    case 'B' : cout << "B"; break ;
}
```

A

 A

B

 逻辑错误

C

 语法错误

D

 AB

118

- A
- B
- C
- D

2.2 循环控制

第128页

2.2 循环控制

两种典型循环结构

当型循环

```
graph TD; A[ ] --> B{循环条件}; B -- true --> C[循环体]; C --> B; B -- false --> D[ ]
```

直到型循环

```
graph TD; A[ ] --> B[循环体]; B --> C{循环条件}; C -- true --> B; C -- false --> D[ ]
```

- 循环体的算法是什么？
- 循环的条件、循环结束条件是什么？
- 如何修改循环条件？

128

两种不同的循环思路

1、while 语句

第135页

2.2.1 while语句

一个简单的循环跟踪：求 $sum = \sum_{i=1}^{100} i$

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

想一想：

循环条件是什么？

循环结束条件是什么？

哪一个语句修改循环条件？

135

循环条件为 i<=100，当 i=101 时就跳出循环

第136页

2.2.1 while语句

一个简单的循环跟踪：求 $sum = \sum_{i=1}^{100} i$

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

可以写成：

`sum += i; i++;`

或：

`sum += i++;`

想一想

如果写成：

`sum += ++i;`

会有什么问题？

136

注意循环计数“1”的错误 正常执行为 5050 若 sum+=++i，相当于 2+3+...+101=5150；

```
#include <iostream>
using namespace std;
int main()
{
    int i = 1, sum = 0;
    while (i < 101);
    {
        sum += i;
        i++;
    }
    cout << sum<<endl;
}
```

注意这样会陷入死循环而不会 cout 结果


```
#include <iostream>
using namespace std;
int main()
{
    int i = 1, sum = 0;
    while (i > 101);
    {
        sum += i;
        i++;
    }
    cout << sum<<endl;
}
```

Microsoft Visual Studio 调试 ×

1

C:\Users\19745\source\repos
按任意键关闭此窗口. . .|

但是这样输出结果为 1，这是由于 $i=0 > 101$ 不成立因此会直接跳出 while 循环

以下是一个难题

```
#include <iostream>
using namespace std;
int main()
{
    int i=100;
    int sum = 0;

    while (i--);
    {
        sum+=++i;
    }

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

这样的输出结果是 0。这是由于一开始当 $i--$ 直满足 $i > 0$ 时，会一直执行循环直到某一次循环结束 $i=0$

然后此时会执行最后一次 while 语句判断（因为 $i--$ 是先判断 $i=0$ （假））然后减一再跳出循环，因此跳出循环时候 $i=-1$

则最后输出结果是 0

应用 1：求 n 个数的平均数 用 `ctrl+z`（不一定是这个）强制跳出循环（相当于百货商场的售货员）

第166页

```
// 例2-9 计算若干个浮点数的平均值，以Ctrl-Z作输入结束
#include <iostream>
using namespace std ;
int main()
{ cout << "Please enter list of numbers" << endl;
  int ValuesProcessed = 0;
  double ValueSum = 0;
  double Value;
  while (cin >> Value)
  { ValueSum += Value;
    ++ValuesProcessed;
  }
  if (ValuesProcessed > 0)
  { double Average = ValueSum / ValuesProcessed;
    cout << "\nAverage: " << Average << endl;
  }
  else cout << "No list to average" << endl;
}
```

修改循环条件
键入 `Ctrl-Z` 结束

166

第168页

```
// 例2-9 计算若干个浮点数的平均值，以Ctrl-Z作输入结束
#include <iostream>
using namespace std ;
int main()
{ cout << "Please enter list of numbers" << endl;
  int ValuesProcessed = 0;
  double ValueSum = 0;
  double Value;
  while ( 1 )
  { if ( ! (cin >> Value) ) break ;
    ValueSum += Value; ++ValuesProcessed;
  }
  if (ValuesProcessed > 0)
  { double Average = ValueSum / ValuesProcessed;
    cout << "\nAverage: " << Average << endl;
  }
  else cout << "No list to average" << endl;
}
```

循环条件永真

break 语句终止循环

168

补充：如何跳出循环体

多选题 20分 第193页

已知int a, b; 如何结束以下死循环（）
while (cin>>a>>b)
{ ...} //...代表若干条语句

A

CTRL+Z on Windows System

B

CTRL+D on Unix, Linux or Mac OS X System

C

数据流到达文件末尾时（EOF）

D

当语句块中含有break语句时

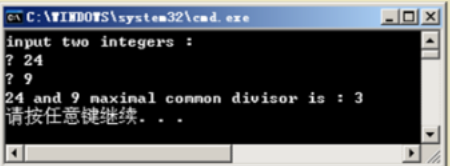
193

答案：ABCD

应用 2：辗转相除法求最大公因数

例2-10 求两个正整数m和n的最大公约数 第186页

#include <iostream>
using namespace std;
int main()
{ int m, n, a, b, r;
cout << "input two integers :\n"; // 提示并输入数据
cout << "? "; cin >> m;
cout << "? "; cin >> n;
if (m > n) { a = m; b = n; } // 把大值放在a 中
else { a = n; b = m; }
r = b;
while (r != 0)
{ r = a % b;
a = b;
b = r;
}
cout << m << " and " << n << " maximal common divisor is : " << a <<
endl;
}



186

例2-10 求两个正整数m和n的最大公约数 第169页

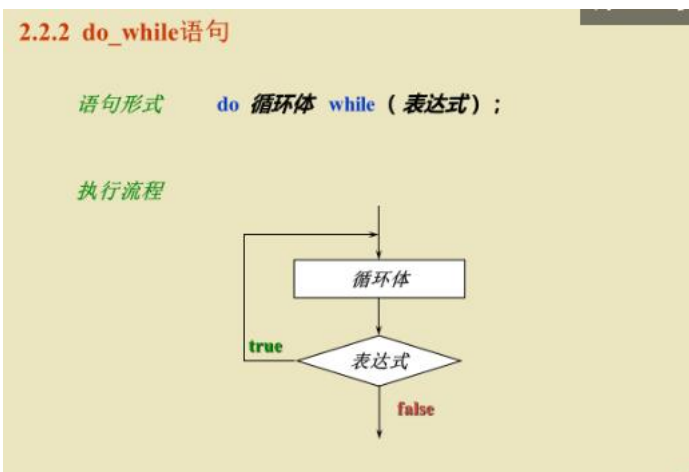
辗转相除法：
当 m > n，m 与 n 的最大公约数等于 n 和 m%n 的最大公约数；
当 n = 0，m 和 n 的最大公约数等于 m。

例如：
m = 24，n = 9
24 和 9 的最大公约数等于 (24 % 9) = 6 和 9 的最大公约数；
9 和 6 的最大公约数等于 (9 % 6) = 3 和 6 的最大公约数；
6 和 3 的最大公约数等于 (6 % 3) = 0 和 3 的最大公约数；
所以，24 和 9 的最大公约数等于 3。

169

2、do-while 语句

Do-while 语句可以理解为先执行循环体然后再判断表达式



这说明，至少会执行了一次循环

良好的编程习惯：while 跟在}后面，不宜在下一行，这样容易引起歧义

```
do
{
    sum+=i++;
}while(i<=100);
```

```
do
{
    sum+=i++;
}while(i<=100);
```

注意事项

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

    do;
    {
    }
    while(i<=100);

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

Error: 应输入“while”

Do 后面只能控制一个语句，这里有一个空语句和一个{ }里面的语句

应用 1 do-while 语句计算从一加到一百

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

    do
    {
        sum+=i++;
    }while(i<=100);

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

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

    while(i<=100)
    {
        sum+=i;
        i++;
    }

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

应用 2 用级数求 pi 的近似值

```

#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int sign=1,i=1; double s=0, x=1;long k=1;

    do
    {
        s+=x;
        k=k+1;
        sign=-sign;
        x=sign/double(2*k-1);
        i++;
    }while(fabs(x)>1e-8);
    s=s*4;

    cout<< s <<endl;
    cout<<i<<endl;
}

```

3、for 语句

第211页

2.2.3 for语句

语句形式

for (表达式1; 表达式2; 表达式3)

循环体;

注意

```

#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int i, sum=0;
    for(i=1;i<=100;i++);
        sum+=i;

    cout<<sum<<endl;
}

```

这样的输出结果是 101，因为循环体是空语句（缺省表达体）

表达式省略问题

2.2.3 for语句

$$sum = \sum_{i=1}^n i$$

不同形式的for 语句结构

(1) **i=1;** //缺省表达式1
for (**;** i <= n ; i++)
{ sum = sum + i ; }

(2) for (i=1; **;** i++)
{ sum = sum + i ;
 if (i >= n) break ; ; }
//缺省表达式2

(3) for (i=1; i <= n **;**)
{ sum = sum + i ;
 i++ ; ; }
//缺省表达式3

(4) for
(i=1; i <= n ; **sum += i++ ;**) ;
//缺省循环体

应用 1：斐波那契数列计算

第257页

例2-13 求菲波那契数列的前n 项

```
#include <iostream>
using namespace std ;
int main()
{ int n, i, a0, a1 ;
  cout << "n = " ;
  cin >> n ;                // 输入项数
  a0 = 0 ; a1 = 1 ;          // 置第1、2 项的值
  cout << a0 << " " << a1 << " " ;
  for ( i = 2; i <= n/2 ; i++ )
  { a0 = a0 + a1 ;            // 迭代新的两项
    a1 = a1 + a0 ;
    cout << a0 << " " << a1 << " " ; // 输出两项
    if ( i % 5 == 0 ) cout << endl ; // 每行输出10 个数据
  }
  if ( n > (i-1)*2 ) cout << a0+a1 << endl ; // i %2
}
```

应用 2：用 for 语句求从一加到一百

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int i, sum=0;
    for(i=1;i<=100;i++)
        sum+=i;

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

2.2.4 循环的嵌套

一个循环语句内包含循环语句，称为嵌套循环

任意循环之间可以相互嵌套

应用 1：判断 m 是不是素数

```

#include <iostream>
using namespace std;
int main()
{
    int i;
    long m;
    cin>>m;
    for(i=2;i<m;i++)
    {
        if(i%m==0)break;
    }
    if(i==m)
    {
        cout<<"是素数"<<endl;
    }
    else
    {
        cout<<"不是素数"<<endl;
    }
}

```

算法优化：

若任意小于等于根号 m 的数不能整除 m，则大于根号 m 的数也不能整除 m

例2-15 判定素数

判定整数m是否素数

整数 m 是素数的条件：除 1 和 m 外，没有其它因数。

算法优化：

若 m 不是素数，有： $m = i * j$ ， $i \leq \sqrt{m}$ ， $j > \sqrt{m}$

```

for (int i = 2; i <= sqrt( m ); i++)
    if ( m % i == 0 ) break;
if (sqrt( m ) < i)
    m 是素数；
else
    m 不是素数；

```

Sqrt()函数在头文件<cmath>中

进一步

例2-16 找出100~200之间的所有素数

分析：

用循环 for m : 100 ~ 200，判断每个 m 是否素数。

for (m = 101; m <= 200; m+=2)

if (m是素数)

输出 m；

除2外
所有偶数不是素数

2.3 判断表达式的使用

- C++没有逻辑数据类型？
- 表达式的值等于非0，是逻辑真；表达式的值等于0，是逻辑假
- 所有的表达式都可以作为判断（逻辑）表达式

表达式可以用于判断

2.3 判断表达式的使用

➤ 算术表达式用于判断

if (*expression* != 0)

→

if (*expression*)

if (*expression* == 0)

→

if (*!expression*)

判断*a*是否等于*b*

if (a - b) cout << a << " != " << b << endl ;

else cout << a << " == " << b << endl ;

2.3 判断表达式的使用

➤ 赋值表达式用于判断

if (*c = a - b*)

→

c = a - b ;

if (c != 0)

cout << " the difference of " << a << " and " << b << " is : " << c << endl ;

else cout << a << " is equal to " << b << endl ;

要区分 c==a-b 和 c=a-b

➤ 对输入作判断

while (*cin >> x*)

→

Ctrl-Z

向输入流插入结束符

n ++ , s += x ;

cout << "n = " << n << endl << "sum = " << s << endl ;

(注意这里用“，”可以起到串联表达式作用)

2.4 转向语句

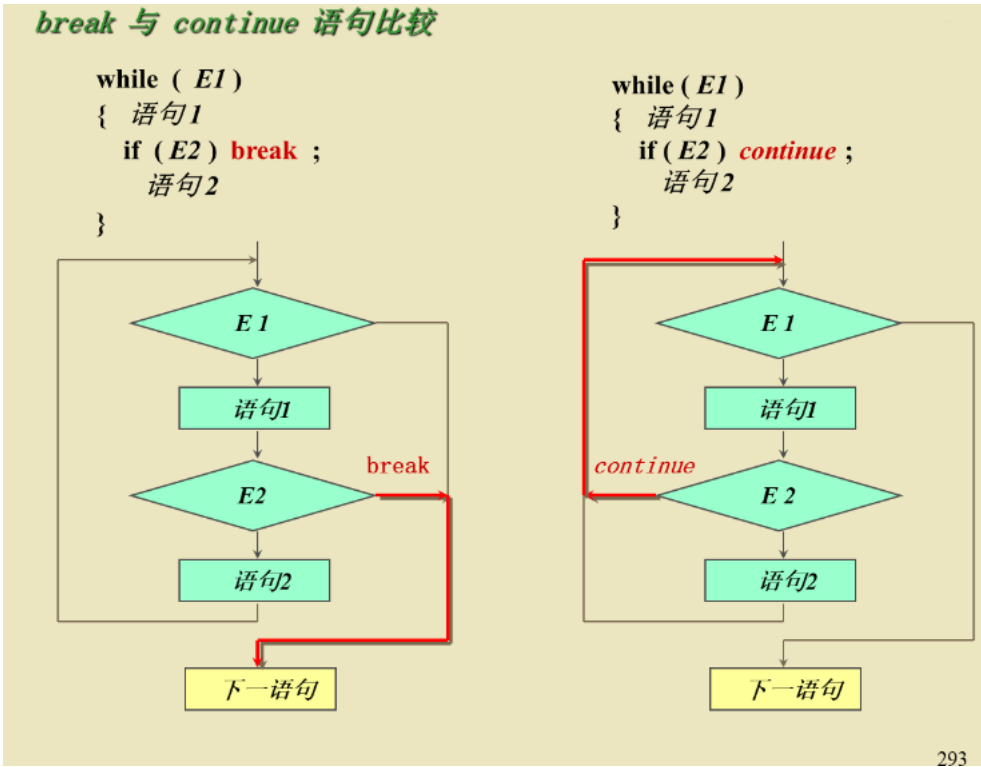
1、break 语句和 continue 语句

➤ *break*语句

无条件地结束switch语句，或循环语句，转向执行语句块的后续语句

➤ *continue*语句

用于循环体中，终止当前一次循环



案例一 对于break语句

假设一个场景，你有一个漂亮的老婆，然后你犯错了，你老婆罚你做5天家务，每天都是给老婆做菜，但是做菜到第三天老婆心软了，原谅你不用做了。

代码如下（示例）：

```
1 public class Test1{
2     public static void main(String[] args){
3         for (int i = 1; i <= 5; i++) {
4             System.out.println("快乐的做菜~~~~");
5             if(i == 3) { //给老婆做菜第3天
6                 break; // 跳出并结束当前循环的执行
7             }
8         }
9     }
10 }
```

结果是你给老婆只做了3天的菜

```
"C:\Program Files\Java\jdk17\jdk-17\
快乐的做菜~~~~
快乐的做菜~~~~
快乐的做菜~~~~

Process finished with exit code 0
```

案例二 对于continue语句

假设和上面案例同样的场景，你犯错了，你老婆罚你做5天家务，每天都是做菜，但是做菜到第三天老婆心软了，原谅你不用做了，但后面依然不解恨，让你继续做第4、5天。

代码如下（示例）：

```
1 public class Test2{
2     public static void main(String[] args){
3         for(int i = 1; i <= 5; i++ ){
4             if(i == 3){ //给老婆做菜第3天
5                 continue;//立即跳出当次执行，进入下一次循环
6             }
7             System.out.println("给老婆第"+ i +"天做菜");
8         }
9     }
10 }
```

结果是你给老婆只做了第1、2、4、5天的菜

```
"C:\Program Files\Java\jdk17\jdk-17\
给老婆第1天做菜
给老婆第2天做菜
给老婆第4天做菜
给老婆第5天做菜
```

下面通过实践证明 break continue 实际上是跳出最近的一个循环



在 goto 前标记一个 loop，当执行到 goto loop 时就跳转到该步骤

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

int main()
{
    loop: cout<<"1"<<endl;
    goto loop;
}
```

这样会一直输出 1.

