第 3 章 语句控制结构

目录

学习目标

- 掌握基本语句控制结构的语法和特点；
- 学会运用基本控制结构解决简单问题；
- 理解并能够运用递推法和穷举法解决实际应用问题。

3.1 语句—空语句

语句

表达式后面加上分号就变成了一个表达式语句（expression statement）。
如：

```
counter + 1;   //一条没有实际意义的表达式语句
counter += 1;  //一条有用的复合赋值语句
```

空语句

- 只有一个分号构成的语句，如：

```
;   //空语句
```

- 空语句不会执行任何操作，如：

```
counter += 1;;   //第二个分号不会影响该语句的执行
```

是否可以随意使用分号？

复合语句

- 复合语句（compound statement）指用花括号括起来的语句和声明序列，也被称作语句块。
- 在块内引入的名字只在块内可见，如：

```
{ //语句块开始
    int sum = 0; // 定义一个对象
    /*...*/
} //语句块结束
```

3.1 语句—控制结构语句作用域

C++ 控制结构语句包括：

- `if`语句
- `switch`语句
- `while`语句
- `for`语句

这些控制语句的作用域只包括紧跟其后的一条语句

下面 while 语句的作用域是什么?

```
while(counter < 10 )
    ++counter;
sum += counter;
```

可用花括号扩展其作用域，如:

```
while(counter < 10 ){ //while作用域从这里开始
    ++counter;
    sum += counter;
} //while作用域到这里结束
```

3.2 分支结构

C++ 提供了两种分支形式

- `if` 语句
- `switch` 语句

3.2 分支结构—if 语句

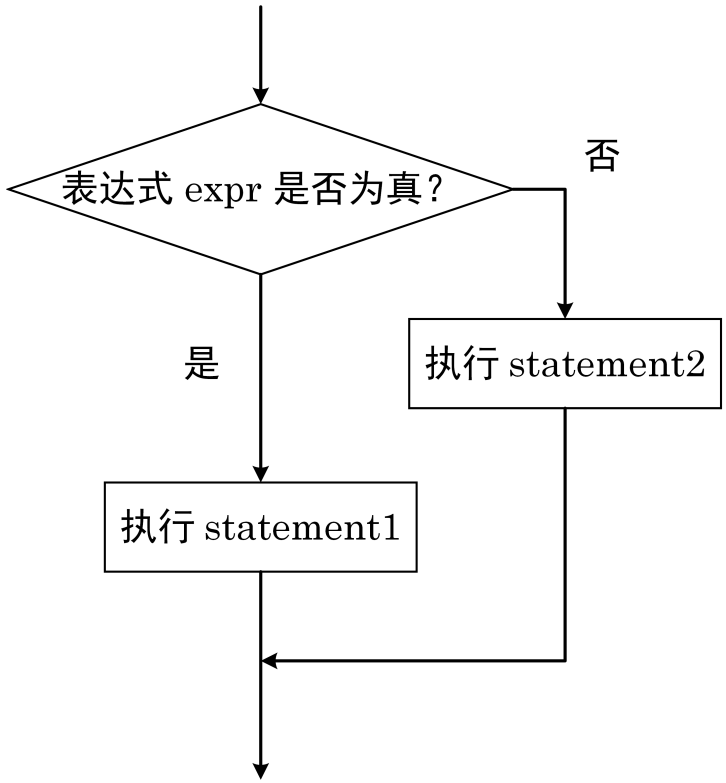
if 语句的语法格式:

```
if (expr) {  //条件表达式
    statement; //语句
}
```

else 分支结构格式:

```
if (expr) {
    statement1; //分支语句1
}
else {
    statement2; //分支语句2
}
```

建议：尽量使用花括号改善程序的可读性



练习：找出下面程序段中的错误

1.
```
if (val1 != val2)
    val1 = val2
else
    val1 = val2 = 0;
```

2.
```
if (val1 = 10)
  //如果val等于10
    value = 1;
```

3.
```
if (val1 < val2)
  //执行以下两个语句
    val1 = 1;
    val2 = 2;
```

4.
```
if val1 < val2
  cin >> val1 >> endl;
```

3.2 分支结构—if 语句

例 3.1:

判断一个整数是否大于 0 且是 3 的倍数。

3.2 分支结构—if 语句

代码清单 3.1，例 3.1:

```
1  #include<iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cout << "请输入一个整数n:";
6      cin >> n;
7      if (n > 0 && n % 3 == 0) { //n大于0且被3整除
8          cout << "Yes" << endl;
9      }
10     else {
11         cout << "No" << endl;
12     }
13     return 0;
14 }
```

示例：输入：7　输出：No　　输入：9　输出：Yes

3.2 分支结构—if 语句

嵌套的 if 语句

- 有两个以上分支时，选用嵌套的 if 语句结构
- 内嵌 if 语句既可以嵌套在 if 语句中，也可以嵌套在 else 语句中

例 3.2:

将百分制的成绩转换成五级制，如果成绩在 90 分到 100 分范围内（包括 90 分和 100 分），则转换成 A ，80 分到 90 分为 B （包括 80 分不包括 90 分），依次类推，60 分以下为 F。

3.2 分支结构—if 语句

代码清单 3.2，例 3.2:

```
#include<iostream>
using namespace std;
int main() {
    unsigned score;
    cout << "请输入一个分数:";
    cin >> score;
    if (score < 60) {
        cout << "F" << endl;
    }
    else if (score < 70) {
        cout << "D" << endl;
    }
    else if (score < 80) {
        cout << "C" << endl;
    }
    else if (score < 90) {
        cout << "B" << endl;
    }
    else {
        cout << "A" << endl;
    }
    return 0;
}
```

示例：输入 76　输出：C

避免悬垂 else

- 上例中 if 和 else 语句个数相同，若 if 语句数目多于 else 语句数目，就会出现 else 和 if 匹配的问题，也称悬垂 else（dangling else）。
- C++ 规定 else 和离它最近的尚未匹配的 if 匹配，如：

```
if (n % 2 == 0) //n 被 2 整除
    if (n % 3 == 0) //n 被 3 整除
        cout << "n 是 6 的倍数";
    else //n 被 2 整除但不能被 3 整除
        cout << "n 是 2 的倍数不是 3 的倍数";
```

思考：

为什么 else 语句数目不能多于 if 语句数目？

避免悬垂 else

- 上例中 if 和 else 语句个数相同，若 if 语句数目多于 else 语句数目，就会出现 else 和 if 匹配的问题，也称悬垂 else（dangling else）。
- C++ 规定 else 和离它最近的尚未匹配的 if 匹配，如：

```
if (n % 2 == 0) //n 被 2 整除
    if (n % 3 == 0) //n 被 3 整除
        cout << "n 是 6 的倍数";
    else //n 被 2 整除但不能被 3 整除
        cout << "n 是 2 的倍数不是 3 的倍数";
```

思考：

为什么 else 语句数目不能多于 if 语句数目？
因为 else 可以省略，而 if 不能省略

避免悬垂 else

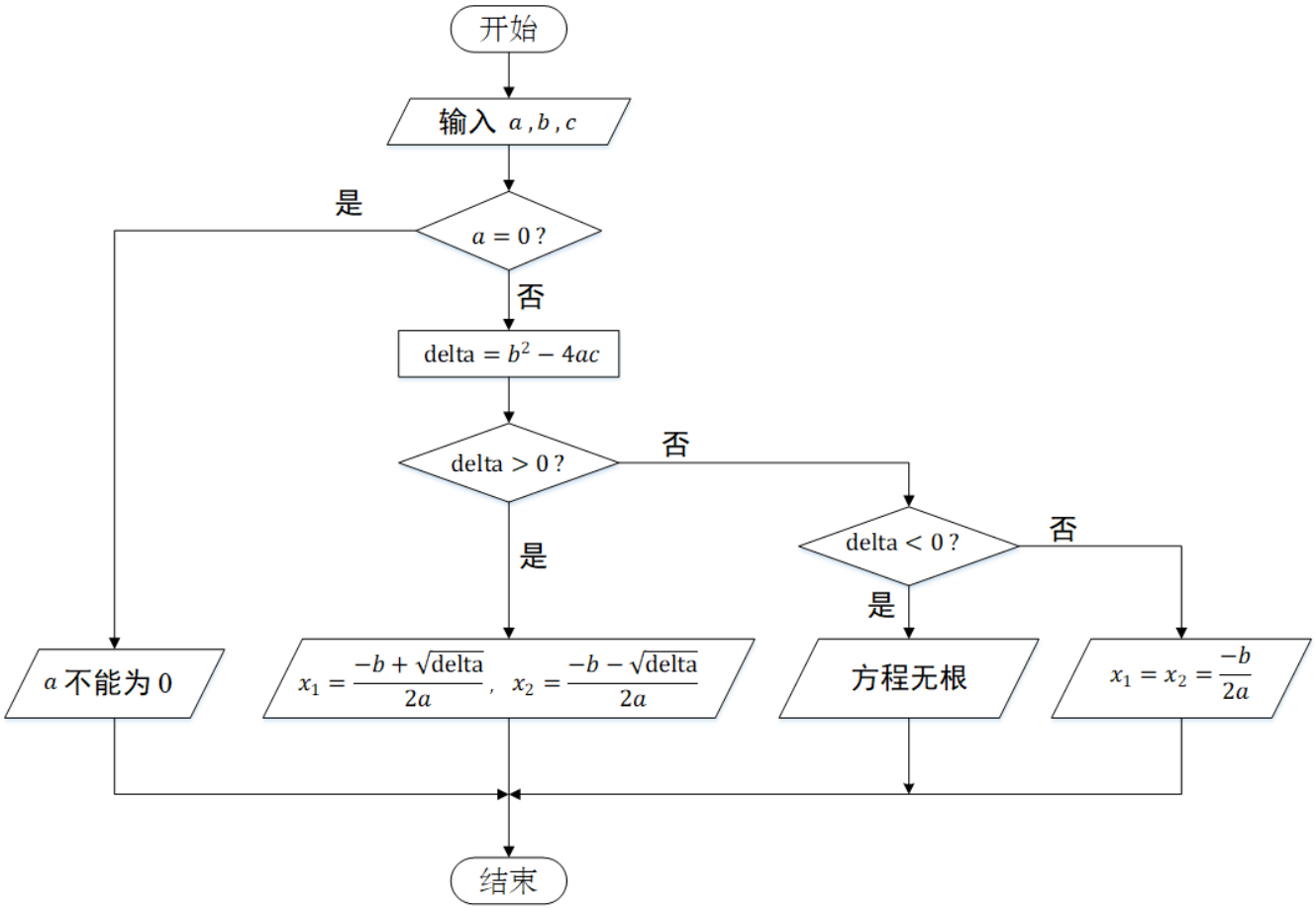
- 根据原则，上例中 else 会和第二个 if 匹配，若我们的本意是 else 和第一个 if 匹配，则相应代码如下：

```
if (n % 2 == 0) {
    if (n % 3 == 0)
        cout << "n 是 6 的倍数";
}
else //n 不能被 2 整除
    cout << "n 不是 2 的倍数";
```

例 3.3:

求一元二次方程 $ax^2 + bx + c = 0$ 的根。

提示：创建对象存放方程系数 → 计算判别式 → 根据判别式处理



代码清单 3.3，例 3.3:

```
 1  #include<iostream>
 2  #include<cmath> //用于求平方根函数sqrt，第12行代码
 3  using namespace std;
 4  int main() {
 5      double a,b,c; //创建3个double类型对象存放三个系数值
 6      cout << "请输入a,b,c:";
 7      cin >> a >> b >> c;
 8      if (a != 0) {
 9          double delta = b*b - 4 * a*c;
10          if (delta > 0) {
11              double x1, x2;  //需要时创建对象
12              delta = sqrt(delta); //求delta的平方根
13              x1 = (-b + delta) / (2 * a);
14              x2 = (-b - delta) / (2 * a);
15              cout << "方程有两个实根: " << x1 << ", " << x2 << endl;
16          }
```

代码清单 3.3，例 3.3:

```
17          else if (delta < 0) {
18              cout << "方程无实根" << endl;
19          }
20          else {
21              cout << "方程有两个相同的实根: " << -b / (2 * a) << endl;
22          }
23      }
24      else { //二次项系数不能为0
25          cout << "a不能为0" << endl;
26      }
27      return 0;
28  }
```

示例：输入 a = 1, b = -4, c = 4　　输出：方程有两个相同的实根：2

3.2 分支结构—switch 语句

switch 分支结构格式:

```
/*...*/
int score;
cin >> score;
switch (score/10){
case 9:
    cout << "A" << endl;
    break;
case 8:
    cout << "B" << endl;
    break;
...
default;
cout << "F" << endl;
}
/*...*/
```

3.2 分支结构—switch 语句

switch 语句语法规则：

- 每个开关入口可以对应多个标签值，执行相同的操作

```
/*...*/
switch(score/10){
case 9:case 10:
    cout << "A" << endl;
    break;
/*...*/
```

- case 标签的值必须为整型常量，且每个标签值必须不同，否则会引发语法错误：

```
case 9.0:case 10: //报错：表达式必须为整型常量表达式
    cout << "A" << endl;
    break;
case 10: //报错：标签值已经出现在次开关
    cout << "B" << endl;
    break;
```

switch 语句语法规则：

- break 语句需根据需要谨慎选择。如果因疏忽，忘记 break 语句，可能带来灾难性的逻辑错误：

```
/*...*/
case 8:
    cout << "B" << endl;
case 7:
    cout << "C" << endl;
    break;
/*...*/
```

当 score 在 B 分数段内时，比如 85 分，会得到如下错误的输出：

```
B
C
```

switch 语句语法规则：

- 开关语句里面定义对象时需要使用花括号。例如：

```
case 8:
    char c = 'B';
    break;
case 7:
    c = 'C'; //修改在前面标签处定义的对象
    break;
```

编译正确，但当进入开关 case 7 时，由于对象 c 未定义，出现错误

switch 语句语法规则：

- 开关语句里面定义对象时需要使用花括号。例如：

```
case 8:{
    char c = 'B'; //对象c只在case 8的作用域内可见
    break;
}
case 7:
    c = 'C'; //修改在前面标签处定义的对象，报错
    break;
```

编译错误，case 7 处对象 c 未定义

代码清单 3.4，使用 switch 语句解决例 3.2

```
#include<iostream>
using namespace std;
int main() {
    int score;
    cout << "请输入一个分数:";
    cin >> score;
    //整型值表达式
    switch (score/10){
    case 9:case 10:
        cout << "A" << endl;
        break;
    //常量标签值后面紧跟冒号
    case 8:
        cout << "B" << endl;
        break;
    case 7:
        cout << "C" << endl;
        break;
    case 6:
        cout << "D" << endl;
        break;
    default:
        cout << "F" << endl;
        break;
    }
    return 0;
}
```

示例：输入：76　　输出：C

3.3 循环结构

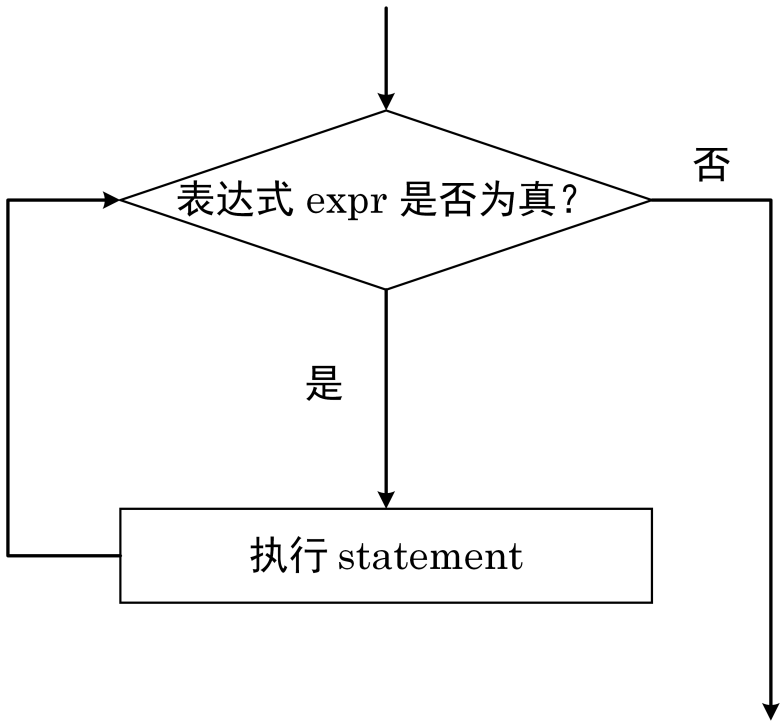
三种循环结构:

- `while`语句
- `do while`语句
- `for`语句

3.3 循环结构—while 语句

while 语句语法格式

```
while (expr) { //条件表达式
    statement; //循环体语句
}
```



3.3 循环结构—while 语句

练习：

下面程序段的运行结果是 ()

```
int x = 0, y = 0;
while (x < 15) {
    ++y;
    x += 1;
}
cout << y << endl;
```

3.3 循环结构—while 语句

练习：

下面程序段的运行结果是 ()

```
int x = 0, y = 0;
while (x < 15) {
    ++y;
    x += 1;
}
cout << y << endl;
```

答案：15

3.2 循环结构—while 语句

例 3.4:

根据以下公式利用迭代法求 π 的近似值，最后一项小于或等于 $1.0E-10$ 时停止。

$$\frac{\pi}{2} = 1 + \frac{1}{3} + \frac{1}{3} \times \frac{2}{5} + \frac{1}{3} \times \frac{2}{5} \times \frac{3}{7} + \ldots + x_i, \quad x_i = x_{i-1} \times \frac{i-1}{2i-1}$$

3.3 循环结构—while 语句

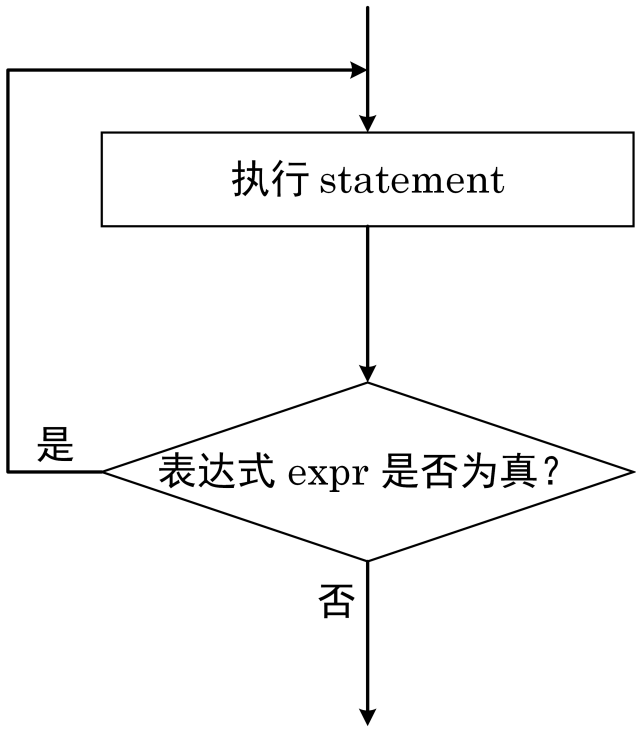
代码清单 3.5，例 3.4:

```
1  #include<iostream>
2  #include <iomanip> //库函数setprecision
3  using namespace std;
4  int main() {
5      //sum存放数列前i项的和，x存放当前项的值，注意初始值
6      double sum=0,x=1;
7      int i = 1; //求解第i项
8      while (x > 1.0E-10) {
9          sum += x;
10         ++i;          //在当前项基础上计算下一项
11         x *= (i - 1.) / (2 * i - 1);//注意1后面的小数点
12     } //fixed和setprecision用于控制输出精度
13     cout<<"pi="<<fixed<<setprecision(10)<<2*sum<<endl;
14                         //输出结果pi=3.1415926533
15     return 0;
16 }
```

3.3 循环结构—do while 语句

do while 语句语法格式

```
do{
    statement; //循环体语句
}while (expr); //条件表达式，注意以分
 号结束
```

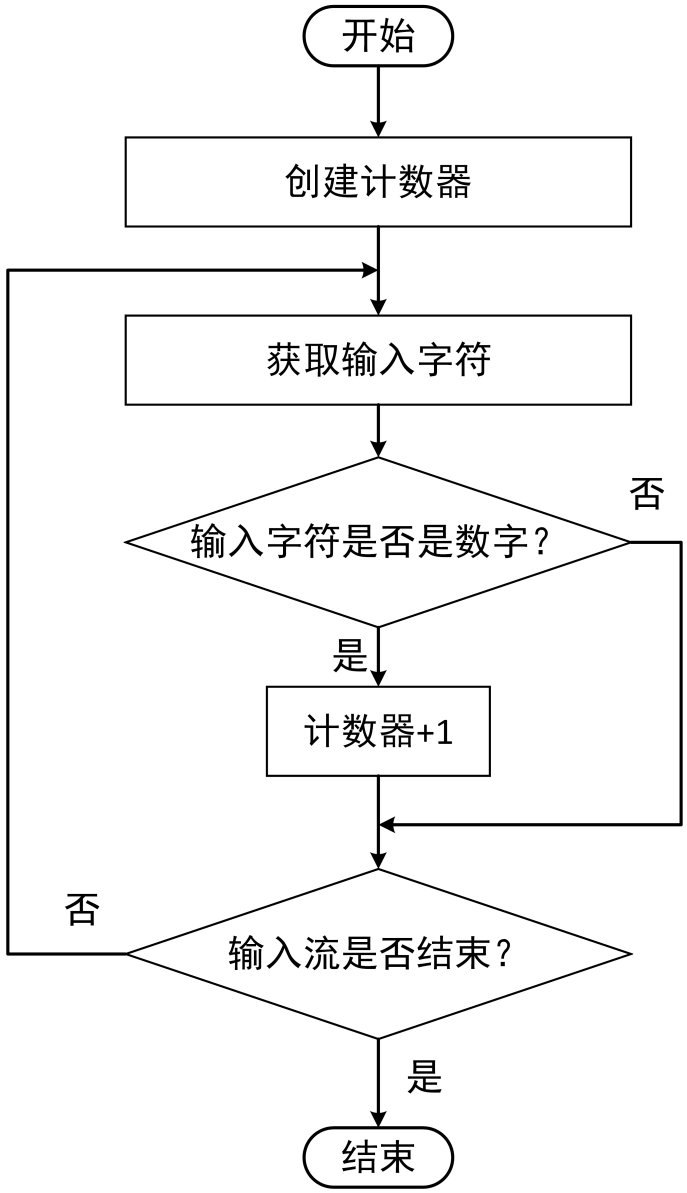


3.2 循环结构—do while 语句

例 3.5:

输入一段文本，统计数字字符个数。

提示：程序主要流程如右图所示



3.3 循环结构—do while 语句

代码清单 3.6，例 3.5：

```
 1  #include<iostream>
 2  using namespace std;
 3  int main() {
 4      int cnt = 0;
 5      char x;
 6      do {
 7          x = cin.get(); //获取终端输入的任意一个字符
 8          if (x >= '0'&&x <= '9') ++cnt;
 9      } while (x != EOF); //EOF为输入流结束标志，组合键Ctrl+z
10      cout << "数字字符个数为:  " << cnt << endl;
11      return 0;
12  }
```

输入：ab12345　　输出：数字字符个数为：5

3.2 循环结构—do while 语句

练习:

1. 下面程序段中的循环执行几次?

```
int x = -1;
do{
    x = x * x;
}while(!x);
```

A. 死循环　B. 执行 3 次　C. 执行一次　D. 有语法错误

2. 下面程序段的输出结果是?

```
int y = 10;
do{
    y--;
}while(--y);
cout << y-- << endl;
```

A.-1　B.1　C.8　D.0

3.2 循环结构—do while 语句

练习:

1. 下面程序段中的循环执行几次?

```
int x = -1;
do{
    x = x * x;
}while(!x);
```

A. 死循环　B. 执行 3 次　C. 执行一次　D. 有语法错误

2. 下面程序段的输出结果是?

```
int y = 10;
do{
    y--;
}while(--y);
cout << y-- << endl;
```

A.-1　B.1　C.8　D.0

答案: 1.C　2.D

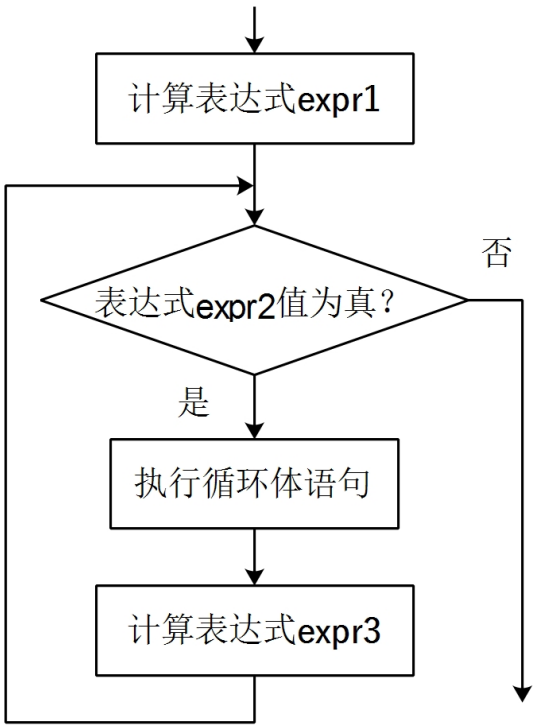
3.3 循环结构—for 语句

for 语句语法格式

```
for(expr1; expr2; expr3){
    statement;
}
```

例如，1 到 100 累加求和：

```
int sum = 0;
for(int i = 1; i <= 100; ++i){
    sum += i;
}
```



3.3 循环结构—for 语句

for 非常灵活，可以有多种形式

- 可以省略任意一个表达式，但分号不能省略：

```
int i(1),sum(0);
for(; i<=100; ++i){
    sum += i;
}
//省略表达式 expr1
```

```
int i(1),sum(0);
for(; i<=100; ){
    sum += i++;
}  //注意不是 ++i;
//省略 expr1 和 expr3
```

```
int i(1),sum(0);
for(; ; ){
    sum += i++;
    if(i>100) break;
} //三个全部省略
```

3.3 循环结构—for 语句

for 非常灵活，可以有多种形式

- 表达式 expr1 可以定义多个对象，表达式 expr3 可以是任意表达式：

```
for (int i = 1, j = 100; i < j; ++i, --j) {
    sum += i + j;
}
```

注意：

虽然上述表达式可以省略，但是需要在合适的位置添加相应功能的语句。

3.3 循环结构—for 语句

练习:

1. 下面程序段中的循环执行几次?

```
/*...*/
int a,b;
for(a = 0,b = 5;a <= b+1;a += 2,b--)
    cout << a << endl;
/*...*/
```

A.3 B.2 C.1 D.0

3.3 循环结构—for 语句

练习:

1. 下面程序段中的循环执行几次?

```
/*...*/
int a,b;
for(a = 0,b = 5;a <= b+1;a += 2,b--)
    cout << a << endl;
/*...*/
```

A.3 B.2 C.1 D.0

答案: A

3.3 循环结构—for 语句

练习:

2. 下面程序段运行结束后，K 的值是?

```
int main() {
    int i, j, k;
    for (i = 0, j = 10; i <= j; i++, j--)
        k = i + j;
    cout << k << endl;
}
```

A.0 B.9 C.8 D.10

3.3 循环结构—for 语句

练习:

2. 下面程序段运行结束后，K 的值是?

```
int main() {
    int i, j, k;
    for (i = 0, j = 10; i <= j; i++, j--)
        k = i + j;
    cout << k << endl;
}
```

A.0 B.9 C.8 D.10

答案: D

练习:

3. 下列语句中，哪一个不是无限循环？

A.
```
i=100;
while(1)
{ i=i%100; i++;
  if (i > 100) break;
}
```

B.
```
for(; ;)
```

C.
```
short k=32765;
do {
    k++; k++;
}while(k>0);
```

D.
```
short i=32765;
while((i++%2)||(i%2))
    i++;
```

练习:

3. 下列语句中，哪一个不是无限循环？

A.
```
i=100;
while(1)
{ i=i%100; i++;
  if (i > 100) break;
}
```

B.
```
for(; ;)
```

C.
```
short k=32765;
do {
    k++; k++;
}while(k>0);
```

D.
```
short i=32765;
while((i++%2)||(i%2))
    i++;
```

答案: C

3.3 循环结构

循环语句的选择原则

- 循环次数确定，选择`for` 语句
- 循环次数不确定，选择`while`或`do while` 语句
  - 循环体至少执行一次，选择`do while` 语句
  - 循环体一次也不执行，选择`while` 语句

例 3.6:

猜数字游戏。程序随机选择一个 0-100 之间的一个数，玩家来猜测程序选择的数，如果猜对了，游戏结束，否则玩家继续猜测，直到猜中为止。对于玩家的每一次猜测，需要给出相应的提示信息：猜对了、猜大了或猜小了。

提示：根据猜测次数，选择合适的循环语句

代码清单 3.7，例 3.6:

```
1  using namespace std;
2  int main(){
3      srand(time(0));//系统当前时间作为随机数发生器的种子
4      int target = rand() % 100;//获取一个[0-100)内的随机数
5      int guess;
6      cout << "`请猜0-100之内的数`" << endl;
7      do {
8          cin >> guess;
9          if (guess < target) {
10             cout << "猜小了" << endl;
11         }
12         else if(guess > target) {
13             cout << "猜大了" << endl;
14         }
15         else {
16             cout << "恭喜！猜对了！" << endl;
17         }
18     } while (guess != target); //猜中时游戏结束
19     return 0;}
```

3.4 跳转语句

跳转语句用于中断当前的执行顺序，包括：

- break语句
- return语句：

```
int main(){
    return 0; /*返回一个整型值*/
}
```

- continue语句
- goto 语句（不作介绍）

3.4 跳转语句—break 语句

break 语句

break 语句只能用于 switch 语句或循环语句中，用来跳出离它最近的 switch 语句或终止循环的执行，它的作用域仅限离它最近的 switch 语句或循环语句。

练习：

1. 下面程序段的运行结果是？

```
int a = 10, y = 0;
do {
    a += 2; y += a;
    cout<<"a="<<a<<","<<"y="<<y<< endl;
    if (y > 50) break;
} while (a = 14);
```

3.4 跳转语句—break 语句

break 语句

break 语句只能用于 switch 语句或循环语句中，用来跳出离它最近的 switch 语句或终止循环的执行，它的作用域仅限离它最近的 switch 语句或循环语句。

练习：

1. 下面程序段的运行结果是？

```
int a = 10, y = 0;
do {
    a += 2; y += a;
    cout<<"a="<<a<<","<<"y="<<y<< endl;
    if (y > 50) break;
} while (a = 14);
```

答案：

a=12,y=12
a=16,y=28
a=16,y=44
a=16,y=60

3.4 跳转语句—break 语句

示例:

将例 3.6 改造成由 `while` 内嵌一个 `switch` 结构来说明 `break` 语句的用法

提示: `switch` 结构需要整型值表达式，可根据玩家猜测的数字与电脑给出的数字的大小关系进行转换

3.4 跳转语句—break 语句

代码清单 3.8，例 3.6:

```
1  using namespace std;
2  int main(){
3      //系统当前时间作为随机数发生器的种子
4      srand(time(0));
5      //获取一个0-100内的随机数
6      int target = rand() % 100;
7      int guess;
8      cout << "`请猜0-100之内的数`" << endl;
9      while(1) {
10         cin >> guess;
11         int val = (guess > target) - (guess < target);
12         //将guess和target的大小关系转化为三个数
13         switch (val){
14         case -1:
15             cout << "猜小了" << endl;
16             break; //跳出switch
```

3.4 跳转语句—break 语句

代码清单 3.8，例 3.6:

```
17              case 1:
18                  cout << "猜大了" << endl;
19                  break; //跳出switch
20              default:
21                  cout << "恭喜！  猜对了！" << endl;
22                  //跳出switch
23                  break;
24              }//switch结束
25              if (val == 0) {
26                  break;//跳出while，游戏结束
27              }
28          }//while结束
29          return 0;
30      }
```

3.4 跳转语句—continue 语句

continue 语句

continue 语句只在循环结构中有作用，用来终止当前操作，进入下一次循环。continue 语句的作用域仅作用于离它最近的循环

例如：

```
//当i为奇数，输出*#，为偶数时无输出
for (int i = 0; i < 5; ++i) {
    if (i % 2) {
        cout << "*"; //打印符号*，然后再打印符号#
    }
    else {
        continue; //结束当前迭代，跳转到for语句，执行++i
    }
    cout << "#";
}
```

输出结果：*#*#

3.4 跳转语句—continue 语句

练习:

1. 下面程序段的输出结果是什么?

```
int main(){
    int x(0);
    for(int i=0; i<2; i++){
        x++;
        for(int j=0; j<=3; j++){
            if(j%2) continue;
            x++;
        }
        x++;
    }
    cout<<x<<endl;
    return 0;
}
```

A. x = 4 B. x = 8 C. x = 6 D. x = 12

3.4 跳转语句—continue 语句

练习:

1. 下面程序段的输出结果是什么?

```
int main(){
    int x(0);
    for(int i=0; i<2; i++){
        x++;
        for(int j=0; j<=3; j++){
            if(j%2) continue;
            x++;
        }
        x++;
    }
    cout<<x<<endl;
    return 0;
}
```

A. x = 4 B. x = 8 C. x = 6 D. x = 12

答案: B

3.4 跳转语句—continue 语句

练习:

2. 下面程序段的输出结果是什么?

```
int k=0; char c='A';
do{
    switch(c++){
        case 'A':k++; break;
        case 'B':k--;
        case 'C':k+=2; break;
        case 'D':k=k%2; continue;
        case 'E':k=k*10; break;
        default :k=k/3;
    }
    k++;
} while(c<'G');
cout<<k<<endl;
```

A.k = 3 B.k = 4 C.k = 2 D.k = 0

3.4 跳转语句—continue 语句

练习:

2. 下面程序段的输出结果是什么?

```
int k=0; char c='A';
do{
    switch(c++){
        case 'A':k++; break;
        case 'B':k--;
        case 'C':k+=2; break;
        case 'D':k=k%2; continue;
        case 'E':k=k*10; break;
        default :k=k/3;
    }
    k++;
} while(c<'G');
cout<<k<<endl;
```

A.k = 3 B.k = 4 C.k = 2 D.k = 0

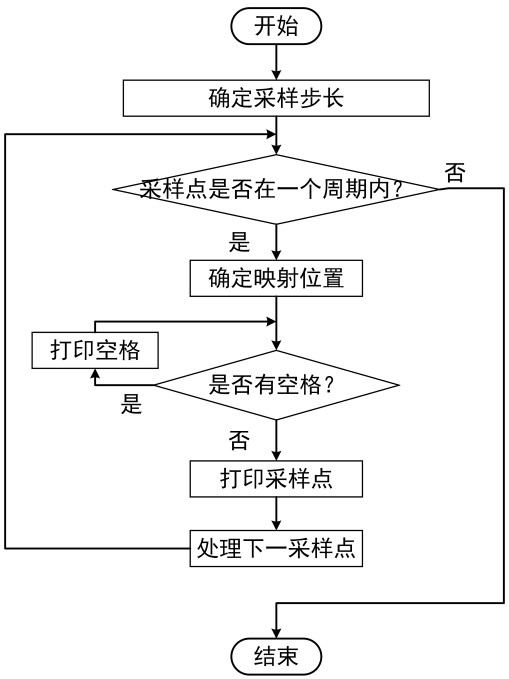
答案: B

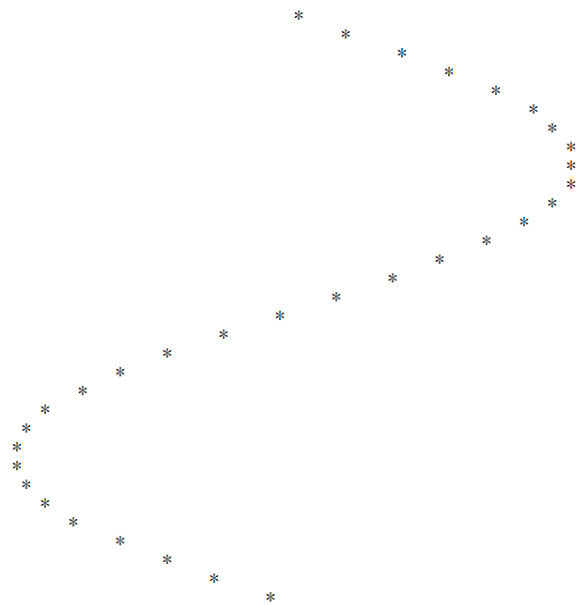
3.5 嵌套结构和应用实例

例 3.7:

将坐标系顺时针旋转 90 度，画出 $\sin(x)$ 在 $x \in [0, 2\pi]$ 之间的曲线，如右图所示。

提示：程序主要流程如左图所示





3.5 嵌套结构和应用实例

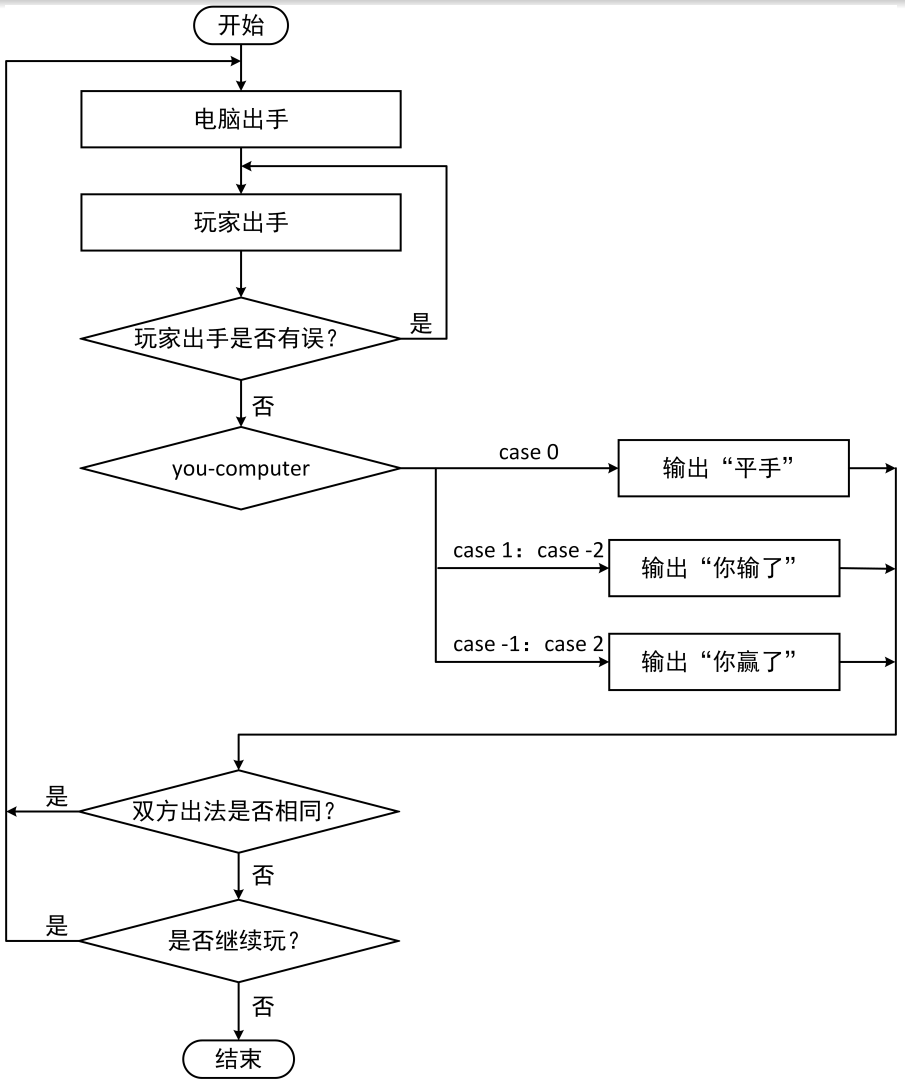
代码清单 3.9，例 3.7：

```
1  using namespace std;
2  int main() {
3      double step = 0.2;  //x增加的步长
4      double x = 0;       //x从0开始
5      while (x < 6.28) {  //画一个周期的曲线
6          int val = 30*(sin(x)+1);//计算sin(x)左侧的空格数
7          for (int i = 0; i < val; ++i) //画出所有空格
8              cout << " ";
9          }
10         cout << "*" << endl;//在相应的位置打印*
11         x += step;         //处理下一个x
12     }
13     return 0;
14 }
```

例 3.8：石头剪刀布游戏

玩家和电脑出法相减结果如右图所示。

玩家 \ 电脑	石头/0	剪刀/1	布/2
石头/0	0	−1	−2
剪刀/1	1	0	−1
布/2	2	1	0



代码清单 3.10，例 3.8:

```
1  int main() {
2      srand(time(0));
3      while (1) {
4          int computer, you;
5          do {
6              computer = rand() % 3;
7              do {
8                  cout << "请出手，0(石头)，1(剪刀)，2(布)：";
9                  cin >> you;
10             } while (you < 0 || you>2);//如果输入有误，从新输入
11             switch (you - computer) {
12             case 0:
13                 cout << "平手" << endl;
14                 break;
15             case 1: case -2:
16                 cout << "你输了!" << endl;^^I^^I^^I^^I^^I
17                 break;
```

代码清单 3.10，例 3.8:

```
18              case -1: case 2:
19                  cout << "你赢了!" << endl;
20              }
21          } while (computer == you);//双方出法相同，继续出
22          cout << "还要玩吗？Y/N:";
23          char play;
24          cin >> play;
25          if (play == 'N' || play == 'n') break;
26      }
27      return 0;
28  }
```

代码清单 3.10，例 3.8:

```
18              case -1: case 2:
19                  cout << "你赢了!" << endl;
20              }
21          } while (computer == you);//双方出法相同，继续出
22          cout << "还要玩吗？Y/N:";
23          char play;
24          cin >> play;
25          if (play == 'N' || play == 'n') break;
26      }
27      return 0;
28  }
```

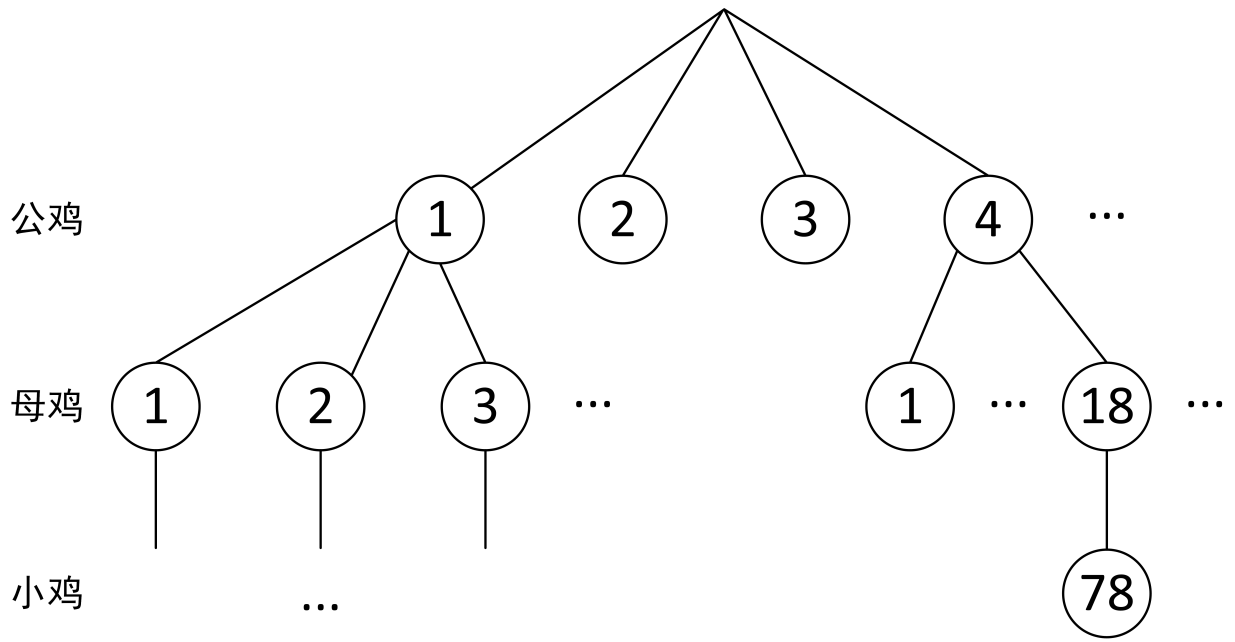
如何让电脑更聪明？提示：尝试人工智能-强化学习方法

3.5 嵌套结构和应用实例

例 3.9:

公元前五世纪，我国古代数学家张丘建在《算经》一书中提出了（百钱百鸡）：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

提示：穷举法：对可能是解的众多候选解按某种顺序进行逐一列举和检验，从中找出符合要求的解。本例穷举示意图如下图所示。



上图中公鸡、母鸡、小鸡数目分别为：4、18、78 是一种符合要求的结果

代码清单 3.11，例 3.9:

```
1  using namespace std;
2  int main() {
3      //公鸡、母鸡最大数目
4      int max_rst = 100 / 5, max_hen = 100 / 3;
5      for (int i = 0; i < max_rst; ++i) {
6          for (int j = 0; j < max_hen; ++j) {
7              int k = 100 - i - j; //小鸡数目
8              //跳过不能被3整除的数，执行流程跳转到++j
9              if (k % 3) continue;
10             if (5 * i + 3 * j + k / 3 == 100) {
11                 cout<<"公鸡: "<<i<<" 母鸡: "<<j<<" 小鸡: "<<k<<endl;
12             }
13         }
14     }
15     return 0;
16 }
```

3.5 嵌套结构和应用实例

```
输出:
公鸡: 0 母鸡: 25 小鸡: 75
公鸡: 4 母鸡: 18 小鸡: 78
公鸡: 8 母鸡: 11 小鸡: 81
公鸡: 12 母鸡: 4 小鸡: 84
```

课后作业

作业本

1. 习题 3.5、3.6 和 3.13

上机练习

1. 实验指导书：第三章

本章结束