CS 1501 程序设计思想与方法 (C++)
Principles and Methodology in Programming
Chapter 4. 循环程序设计-1

陈东尧
上海交通大学
2025年秋季

switch 语句 — 自动出四则运算计算题

- 算法设计：要求自动出0 - 9之间的四则运算题， 并批改结果。
- 关键在生成题目， 其算法伪代码为：

```
Switch(题目类型)
{ case 加法： 显示题目，输入和的值，判断正确与否
  case 减法： 显示题目，输入差的值，判断正确与否
  case 乘法： 显示题目，输入积的值，判断正确与否
  case 除法： 显示题目，输入商和余数的值，判断正确与否
}
```

- 如何让程序每次执行的时候都出不同的题目？包括不同的题目类型和不同的计算数据。
  - 解决方案：随机数生成器rand()

switch 语句 — 自动出四则运算计算题— rand()

- 随机数生成器rand()：能随机生成0到RAND_MAX之间的整型数
- 将生成的随机数映射到0 - 9之间：
  - rand() % 10
- 运算符的生成：用编码0 - 3表示四个运算符。因此题目的生成就是生成0 - 3之间的随机数。

【拓展阅读，C++ rand】https://www.cplusplus.com/reference/cstdlib/rand/

switch 语句 — 自动出四则运算计算题— rand()

- 随机数生成器rand()：能随机生成0到RAND_MAX之间的整型数
- 将生成的随机数映射到0 - 9之间：
  - rand() % 10

  注： to get a random integer in the range [0, x]：rand() % (x + 1)

- 运算符的生成：用编码0 - 3表示四个运算符。因此题目的生成就是生成0 - 3之间的随机数

【拓展阅读，C++ rand】https://www.cplusplus.com/reference/cstdlib/rand/

switch 语句 — 自动出四则运算计算题— rand()

- 设置种子的函数srand ： srand （种子）
- 如何让程序每次执行时选择的种子都不一样呢?
  - 选择系统时间为种子：time(NULL) 取当前的系统时间。

【拓展阅读，C++ rand】https://www.cplusplus.com/reference/cstdlib/rand/

switch 语句 — 自动出四则运算计算题例

```
#include <cstdlib>   //包含伪随机数生成函数
#include <ctime>     //包含取系统时间的函数
#include <iostream>
using namespace std;

int main()
{
  int num1, num2, op, result1, result2;
  //num1,num2:操作数，  op:运算符，  result1,result2: 结果

  srand(time(NULL));                    //随机数种子初始化

  num1=rand() % 10;// 生成运算数
  num2= rand() % 10 ;//生成运算数
  op= rand() % 4 ; // 生成运算符 0--+，  1-- -，  2--*，  3-- /
```

```
switch (op)
   {case 0: cout << num1 << "+" << num2 << "= ?" ; cin >> result1;
              if (num1 + num2 == result1)  cout << "you are right\n";
                    else  cout << "you are wrong\n";
                    break;
    case 1: cout << num1 << "-" << num2 << "= ?" ; cin >> result1;
              if (num1 - num2 == result1) cout << "you are right\n";
                    else  cout << "you are wrong\n";
              break;
    case 2: cout << num1 << "*" << num2 << "= ?" ; cin >> result1;
              if (num1 * num2 == result1) cout << "you are right\n";
             else  cout << "you are wrong\n";
                   break;
    case 3: cout << num1 << "/" << num2 << "= ?" ;  cin >> result1;
                    cout << “余数为 = ?";  cin >> result2;
              if  ((num1 / num2 == result1) && (num1 % num2 == result2))
                         cout << "you are right\n";
                     else  cout << "you are wrong\n";
               break; }
  return 0;
}
```

程序的不足

- 每次执行只能出一道题
- 减法可能出现负值
- 除法可能出现除0
- 结果太单调

回顾分支程序设计

- 分支程序设计的快速回顾
  - 逻辑判断
    - 关系表达式
    - 逻辑表达式
      - 注意它们的优先级问题
  - 根据逻辑判断执行不同的处理
    - if语句
    - switch语句

回顾分支程序设计

- 分支程序设计的快速回顾
  - 逻辑判断
    - 关系表达式
    - 逻辑表达式
      - 注意它们的优先级问题
  - 根据逻辑判断执行不同的处理
    - if语句
    - switch语句
- 四大湖问题

四大湖面积排序？

Credit: https://www.sohu.com/a/204550265_651611

逻辑表达式：四大湖问题

- 上地理课时，四位同学回答我国四大湖大小时分别说：

  A：洞庭最大，洪泽最小，鄱阳第三

  B：洪泽最大，洞庭最小，鄱阳第二，太湖第三

  C：洪泽最小，洞庭第三

  D：鄱阳最大，太湖最小，洪泽第二，洞庭第三

- 对于每个湖的大小，每个人仅答对一个，试判断四个湖的大小。

逻辑表达式：四大湖问题

- 上地理课时，四位同学回答我国四大湖大小时分别说：

  A：洞庭最大，洪泽最小，鄱阳第三　　a==1 && b==4 && c==3

  B：洪泽最大，洞庭最小，鄱阳第二，太湖第三　　a==4 && b==1 && c==2 && d==3

  C：洪泽最小，洞庭第三　　a==3 && b==4

  D：鄱阳最大，太湖最小，洪泽第二，洞庭第三　　a==3 && b==2 && c==1 && d==4

- 对于每个湖的大小，每个人仅答对一个，试判断四个湖的大小。
- 思路：用a,b,c,d分别表示四个湖的排序。a表示洞庭，b表示洪泽，c表示鄱阳，d表示太湖。则四位同学的回答可表示出来.

逻辑表达式：四大湖问题

- 上地理课时，四位同学回答我国四大湖大小时分别说：

  A：洞庭最大，洪泽最小，鄱阳第三　　a==1 && b==4 && c==3

  B：洪泽最大，洞庭最小，鄱阳第二，太湖第三　　a==4 && b==1 && c==2 && d==3

  C：洪泽最小，洞庭第三　　a==3 && b==4

  D：鄱阳最大，太湖最小，洪泽第二，洞庭第三　　a==3 && b==2 && c==1 && d==4

- 对于每个湖的大小，每个人仅答对一个，试判断四个湖的大小。
- 思路：用a,b,c,d分别表示四个湖的排序。a表示洞庭，b表示洪泽，c表示鄱阳，d表示太湖。则四位同学的回答可表示（抽象）如下：

A: ((a==1) + (b==4) + (c==3)) == 1　　B: ((a==4) + (b==1) + (c==2) + (d==3)) == 1

C: ((a==3) + (b==4)) == 1　　D: ((a==3) + (b==2) + (c==1) + (d==4)) == 1

逻辑表达式：四大湖问题

- 上地理课时，四位同学回答我国四大湖大小时分别说：

  A：洞庭最大，洪泽最小，鄱阳第三　　a==1 && b==4 && c==3

  B：洪泽最大，洞庭最小，鄱阳第二，太湖第三　　a==4 && b==1 && c==2 && d==3

  C：洪泽最小，洞庭第三　　a==3 && b==4

  D：鄱阳最大，太湖最小，洪泽第二，洞庭第三　　a==3 && b==2 && c==1 && d==4

- 对于每个湖的大小，每个人仅答对一个，试判断四个湖的大小。
- 思路：用a,b,c,d分别表示四个湖的排序。a表示洞庭，b表示洪泽，c表示鄱阳，d表示太湖。则四位同学的回答可表示（抽象）如下：
- 然后编程查找同时满足4个式子的取值。

```
A:  ((a==1) + ( b==4) + ( c==3)) == 1     B: ((a==4) + ( b==1) + ( c==2) + ( d==3)) == 1
C:  ((a==3) + ( b==4)) == 1               D:  ((a==3) + ( b==2) + ( c==1) + (d==4)) == 1
```

核心代码

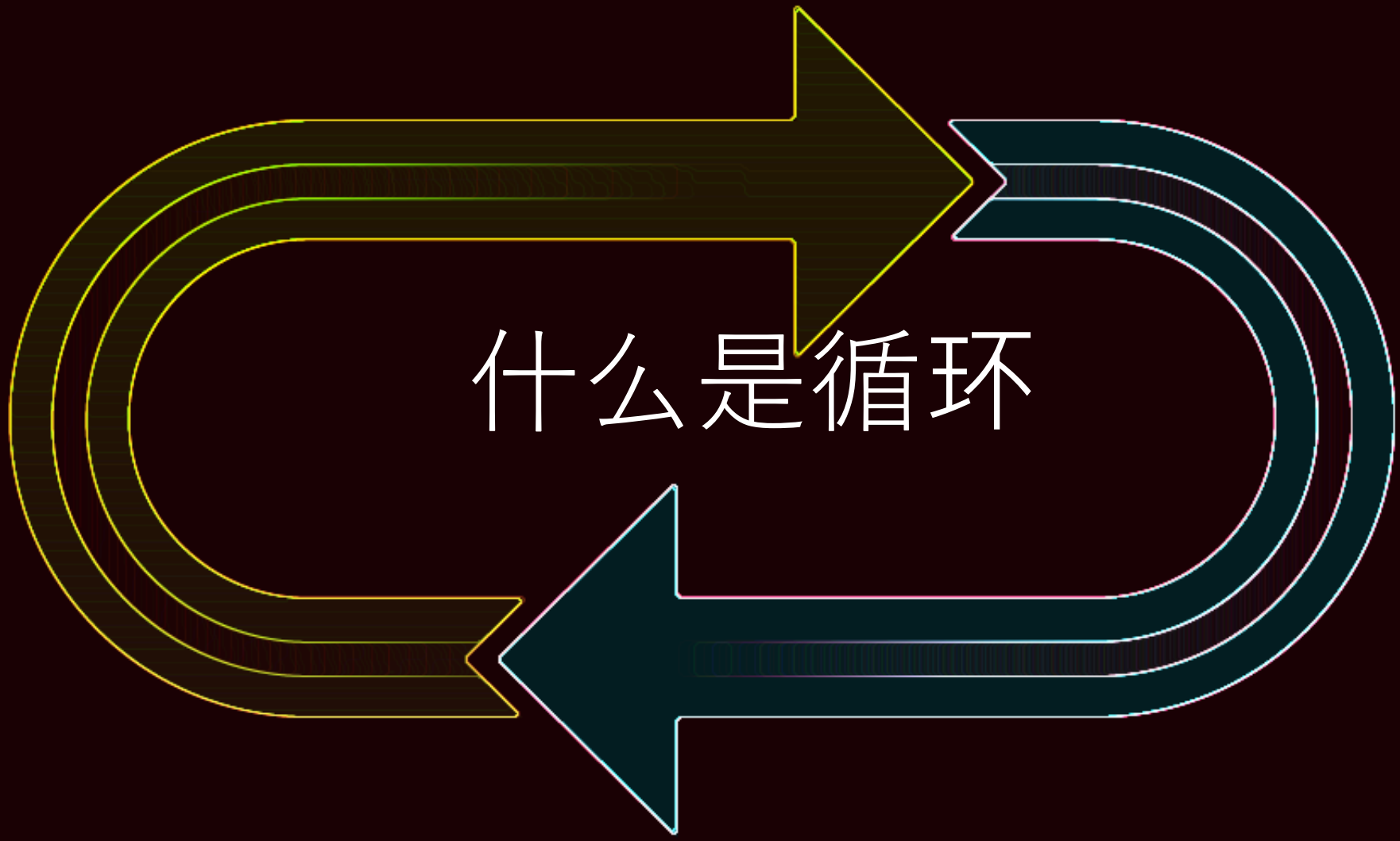
```
if( ((DT==1)+(HZ==4)+(PY==3))==1 && ((HZ==1)+(DT==4)+(PY==2)+(TH==3))==1 && ((HZ==4) + (DT==3)
== 1)
&& ((PY==1)+(TH==4)+(HZ==2)+(DT==3))==1)
{
	cout<<"洞庭湖名次为：  "<<DT<<endl<<"洪泽湖名次为：  "<<HZ<<endl<<
	"鄱阳湖名次为：  "<<PY<<endl<<"太湖名次为：  "<<TH<<endl;
	return 0;
}
```

核心代码

```
if( ((DT==1)+(HZ==4)+(PY==3))==1 && ((HZ==1)+(DT==4)+(PY==2)+(TH==3))==1 && ((HZ==4) + (DT==3)
== 1)
&& ((PY==1)+(TH==4)+(HZ==2)+(DT==3))==1)
{
	cout<<"洞庭湖名次为：  "<<DT<<endl<<"洪泽湖名次为：  "<<HZ<<endl<<
	"鄱阳湖名次为：  "<<PY<<endl<<"太湖名次为：  "<<TH<<endl;
	return 0;
}
```

如何保证尝试过各种可能性?

枚举！（iteration）


什么是循环

一个例子

- 变成输出

1+2+3+…+100的结果

方法1：顺序程序设计

```
int main( ) {
        int s = 1;
        s += 2;
        s += 3;
        …
        s += 100;
        std::cout << s << std::endl;
      return 0;
   }
```

缺点：

1.大量相似的重复代码

2.编写麻烦，维护麻烦

方法2：巧算

1+2+3+...+n=?

```
int main( ) {
    int s = (1+100)*100/2;
    std::cout << s << std::endl;
    return 0;
}
```

问题：

1. 巧算是有应用针对性的，而非**普适的**
2. 非现成的巧算设计需要大量的思考与正确性验证
3. 绝大部分计算并没有巧算

方法3: 循环控制

```
int main() {
    int s = 0;
    for (int i=1; i<=100; i++)
        s += i;
    std::cout << s << std::endl;
    return 0;
}
```

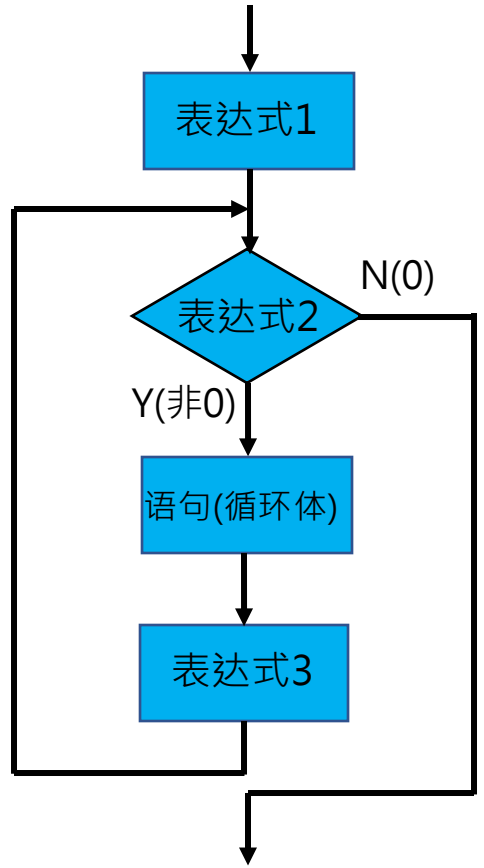
采用循环控制的逻辑：

1. 程序员来设计循环控制
2. 计算机完成繁重的、但有规律的计算

章节内容

1. 重复 N 次循环
2. while 循环
3. do ... while 循环
4. 循环的中途退出
5. 枚举法
6. 贪婪法

重复循环N次：for循环语句




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

重复循环N次：for循环语句

- 格式：

表达式1 初始化　**表达式2** 条件：常常反映了循环的终止条件　**表达式3** 循环步长

```
for (init; condition; step) {          //循环控制行
     Statement  //循环体中的执行语句
}
```

重复循环N次：for循环语句

- 格式：

表达式1 初始化　**表达式2** 条件：常常反映了循环的终止条件　**表达式3** 循环步长

```
for (init; condition; step) {        //循环控制行
    Statement  //循环体中的执行语句
}
```

- **执行过程：**
  1. **执行表达式1**
  2. **执行表达式2**
  3. **如果表达式2的结果为“true”，则执行循环体和表达式3，**然后回到2，否则for语句执行结束

- 循环体所有语句的一次完全执行称为一个循环周期
- 循环体可以是复合语句或空语句

具体几个问题

- C++中的具体写法是？
- for循环的**具体**工作流程
  - 表达式1执行几次？
  - 表达式2执行几次？
  - 表达式3执行几次？
- for内循环体执行了几次？

逗号表达式

- 有一连串基本的表达式组成，一个一个依次执行

如a的初值为0，则表达式

a += 1, a += 2, a += 3, a += 4, a += 5

的结果为

15

有了逗号表达式…

- 从1加到100的问题可以只用一个语句：

```
for (int i=1, int s=0; i<=100; i++) {
    s += i; // 注意s变量的生存周期仅在循环体内
}
```

将所有的初始化都放在循环外，即

```
int i=1 ； int s=0 ；
for ( ；i<=100; i++)
    s += i;
```

- 建议还是用

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

Much Better! 

for 循环的进一步讨论

```
for (init; condition; step)
```

- 表达式 2 不一定是关系表达式。它可以是逻辑表达式，可以是算术表达式
- 当表达式 2 是算术表达式时，只要表达式的值为非0，就执行循环体，表达式的值为0时退出循环

for 循环的进一步讨论

```
for (init; condition; step)
```

- 表达式 2 不一定是关系表达式。它可以是逻辑表达式，可以是算术表达式
- 当表达式 2 是算术表达式时，只要表达式的值为非0，就执行循环体，表达式的值为0时退出循环
- 如果表达式 2 省略，即不判断循环条件，循环将无终止地进行下去。无终止的循环称为“死循环”（infinite loop）
  - 最简单的死循环是 for (;;)
- 要结束一个无限循环，必须从键盘上输入特殊的命令以中断程序执行并强制退出。 VS Code: “control + C”

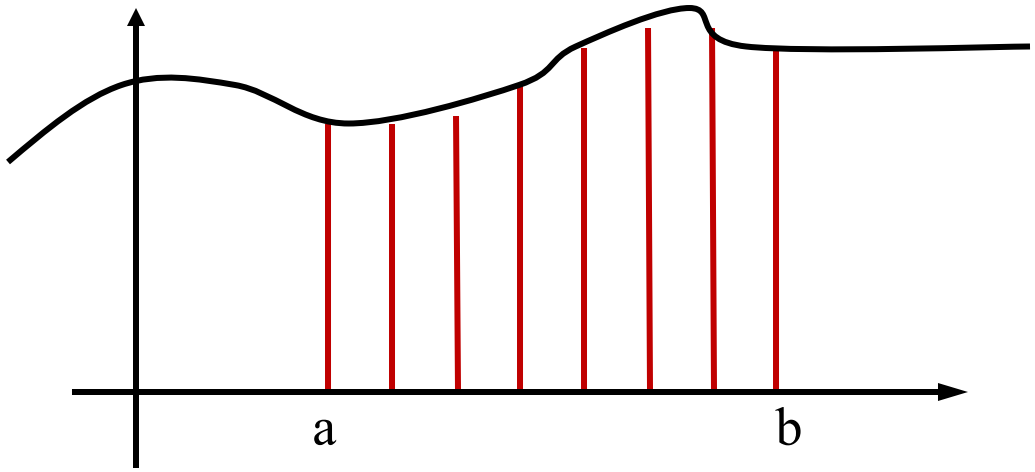
for 循环的进一步讨论

```
for (init; condition; step)
```

- 表达式 3 也可以是任何表达式，一般为赋值表达式或逗号表达式。
- 表达式 3 是在每个循环周期结束后对循环变量的修正。
- 表达式 3 可以省略，此时做完循环体后直接执行表达式 2。

For 循环实例

- 求函数 $f(x) = x^2 + 5x + 1$ 在区间[a，b]之间的定积分（definite integral）
- 实现思想：函数与x轴围成的区域的面积。定积分可以通过将这块面积分解成一连串的小矩形，计算各小矩形的面积的和而得到



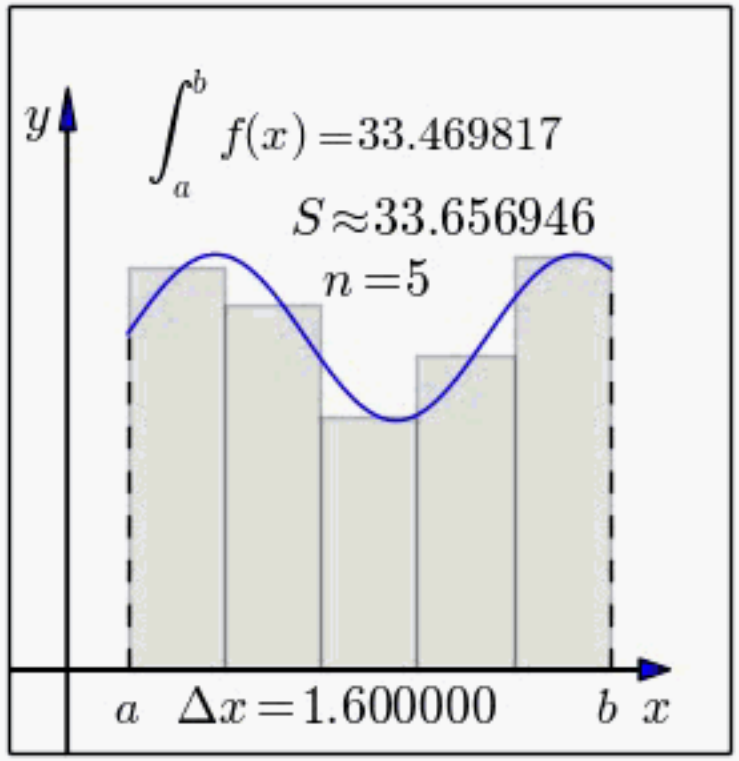
思考：积分的计算方法

黎曼积分 （Riemann integral）

对一个在区间[a, b]有定义的实值函数f，f关于取样分割$x_0,\cdots,x_n$、$t_0,\cdots,t_{n-1}$的黎曼和定义为如下和式：

$$\sum_{i=0}^{n-1} f(t_i)(x_{i+1} - x_i)$$

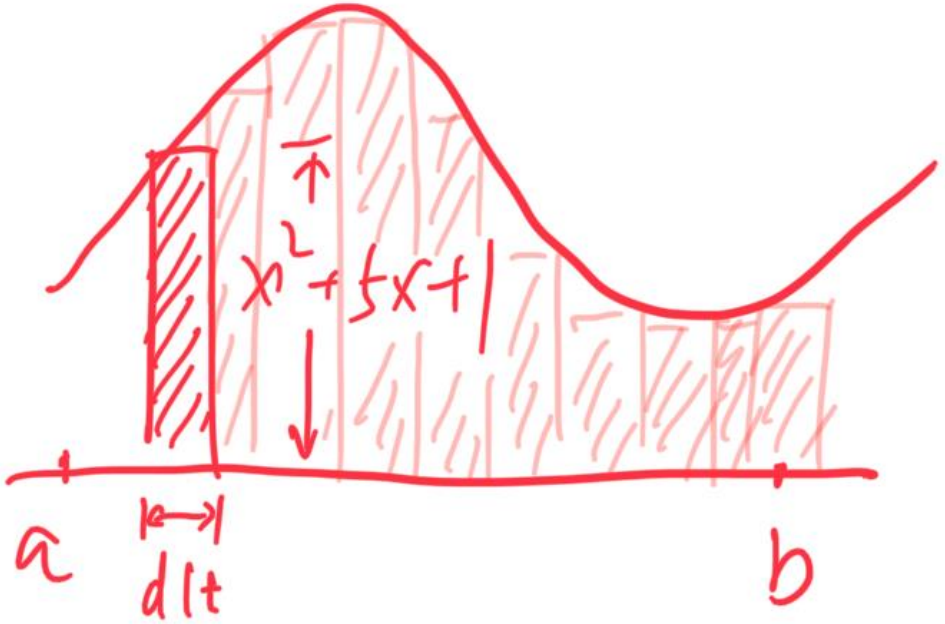
和式中的每一项是子区间长度x_{i+1}-x_i与在t_i处的函数值$f(t_i)$的乘积。其中$x_i <= t_i <= x_{i+1}$



【扩展阅读】梯形公式（Trapezoidal Rule）：https://www.geeksforgeeks.org/trapezoidal-rule-for-approximate-value-of-definite-integral/

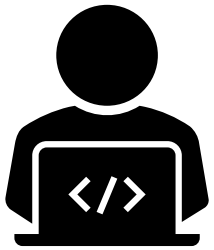
思考：积分的计算方法

直观地说：就是分割区间为长、函数值为高的矩形的面积之和



【扩展阅读】梯形公式（Trapezoidal Rule）：https://www.geeksforgeeks.org/trapezoidal-rule-for-approximate-value-of-definite-integral/

```cpp
#include <iostream>
using namespace std;
int main()
{
    double a, b, dlt, integral = 0;

    cout << "请输入定积分的区间：" ;
    cin >> a >> b;
    cout << "请输入小矩形的宽度：" ;
    cin >> dlt;

    for (double x = a; x < b; x += dlt)
    integral += (x * x + 5 * x + 1) * dlt;

    cout << "积分值为：" << integral << endl;
    return 0;
}
```



代码演示

chapter4-integral.cpp

for循环的用法探讨

- 一般形式：

  for (secs =1; secs <=100; secs++) {.....}

- 递减计数

  for (secs = 5; secs > 0; --secs) {.....}

- 循环变量

  for (ch = 'a'; ch <= 'z'; ch++) {.....}

- 表达式1的任意性

  for (n = 1, printf ("The n is %d.¥n", n); n <= 5 ; n++)
  {ans *= n; .....}

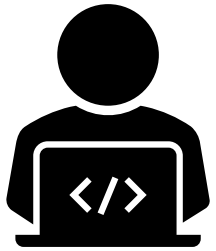
for循环的用法探讨

- 表达式2的变化
  for (secs = 1; secs * secs * secs <= 600; secs ++) {.....}
- 表达式3的改变
  for (x = 1; y <= 75; y = (++x * 5) + 50) {.....}
- 几何增长（exponential growth）方式
  for (secs = 10.0; secs < 60.0; secs *= 1.3) {.....}
- 表达式的省略（注：不能省略分号）
  for (n = 3; ans <= 25;) {ans *= n; .....}
- 循环体中对循环的改变（逻辑易出错，慎用）
  for (n = 1; n <= 25; n += delta) {delta += 1;}

课堂练习：从1加到100

假设变量 s, i 均已经定义好，横线均为一条语句：

- int s=0; for (int i=1; i<=100; i++) {_____}
  int s=0; for (int i=1; i<=100; i++) { **s+=i;** }
- int s=0; for (int i=1; i<=100;) { _____; _____}
  int s=0; for (int i=1; i<=100;) { **s+=i**; **i++**;}
- int s=0; for (int i=1; i<=100;) { _____}
  int s=0; for (int i=1; i<=100;) { **s+=i++**;}
- int s=0; for (int i=1; i<=100; _____) {};
  int s=0; for (int i=1; i<=100; **s+= i, i++**) {};



代码演示

chapter4-1to100.cpp

for 循环语句常用于确定循环次数的循环控制

- for (i=1; i<=100; ++i) s += i;

无法确定循环次数的循环如何控制？

如：未知长度的字符流中word计数…

那么回到我们的四大湖问题

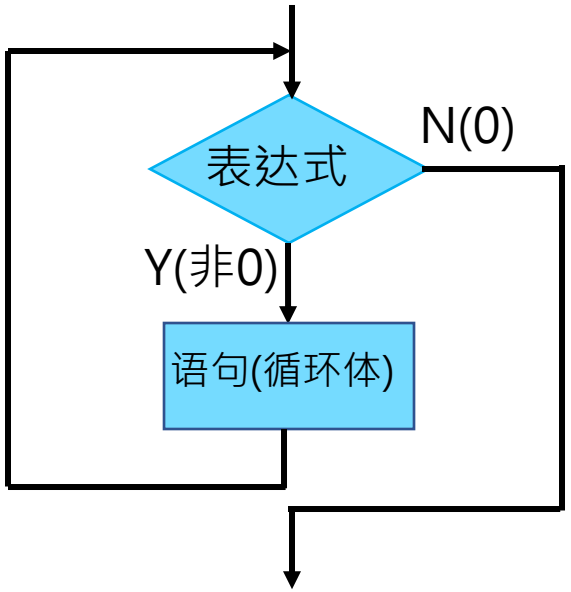

```
int a = 0, b = 0, c = 0, d = 0; // a:洞庭湖； b: 洪泽湖； c: 鄱阳湖； d: 太湖

for (a=1; a<=4; ++a){
    for (b=1; b<=4; ++b){
        for (c=1; c<=4; ++c){
            for (d=1; d<=4; ++d){
                if(((a==1)+(b==4)+(c==3))==1&&((b==1)+(a==4)+(c==2)+(d==3))==1&&((b==4)+(a==3))==1&&((c==1)
        +(d==4)+(b==2)+(a==3))==1){
                    cout << a << b << c << d << endl;
                    }
                }
        }
    }
}
```

还可以继续优化！
TBD…

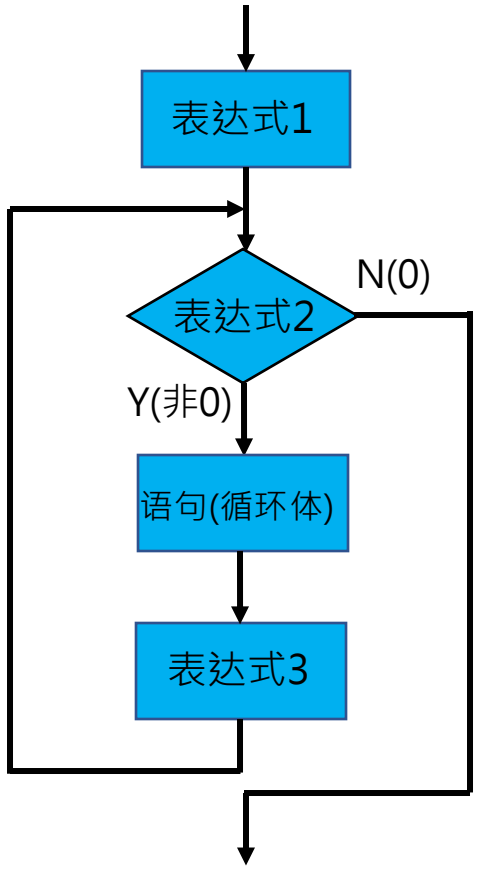
章节内容

1. 重复 N 次循环
2. while 循环
3. do ... while 循环
4. 循环的中途退出
5. 枚举法
6. 贪婪法

While 循环语句

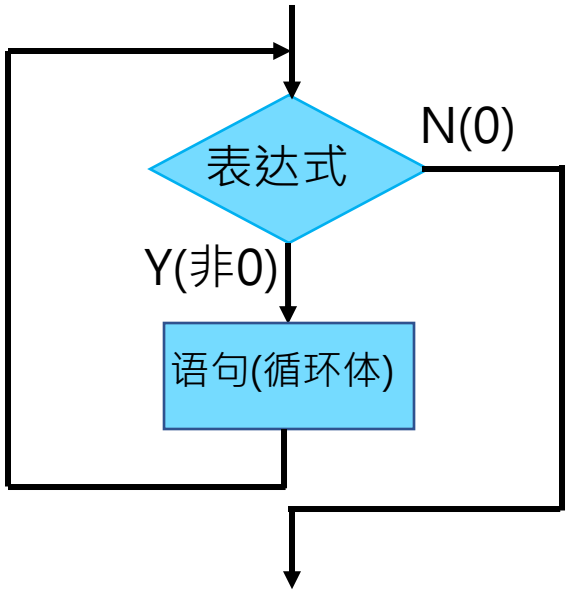



```
while(表达式)
{
        语句;
}
```




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

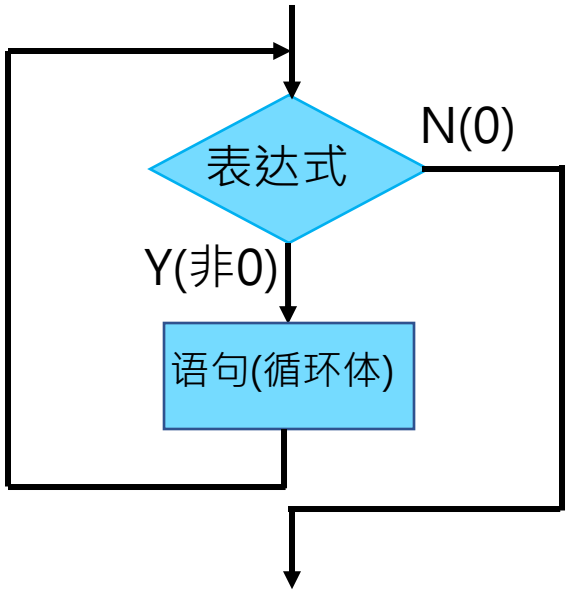
While 循环语句




```
while(表达式)
{
        语句;
}
```

- 计算表达式的值，当值为真（非0）时，执行循环体语句，一旦条件为假，就停止执行循环体。如果条件在开始时就为假，那么不执行循环体语句直接退出循环
- 用途：用于循环次数不定的循环。循环是否结束取决于某一个条件是否成立

While 循环语句




```
while(表达式)
{
        语句;
}
```

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

1 循环初值

2 循环条件 | 终止条件

3 循环体 | 循环变量增值

- 计算表达式的值，当值为真（非0）时，执行循环体语句，一旦条件为假，就停止执行循环体。如果条件在开始时就为假，那么不执行循环体语句直接退出循环
- 用途：用于循环次数不定的循环。循环是否结束取决于某一个条件是否成立

求：$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$　　$\frac{x^n}{n!} < 0.000001$ 时结束。

求：$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$　　$\frac{x^n}{n!} < 0.000001$ 时结束。

```
ex=0;
p = 1;
while (p>0.000001)
{
   ex += p;
   计算新的p;
}
```

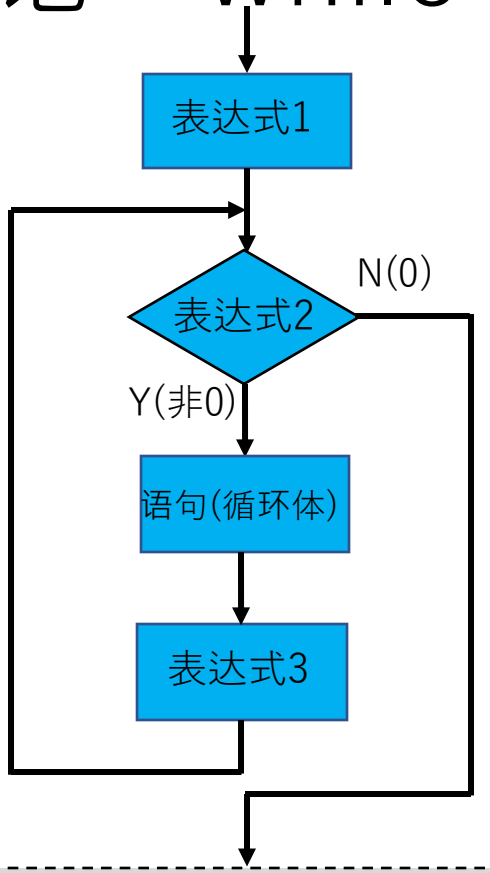
```
int main()
{
  double ex, x, p;//ex存储e^x的值，p保存当前项的值
  int i;

  cout << "请输入x: "; cin >> x;

  ex=0; p=1; i=0;

  while (p > 1e-6){
       ex += p;
        ++i;
       p = p * x / i;
 }
 cout << "e的" << x << "次方等于: " << ex << endl;
 return 0;
}
```

讨论：while 与 for 有哪些关键区别？

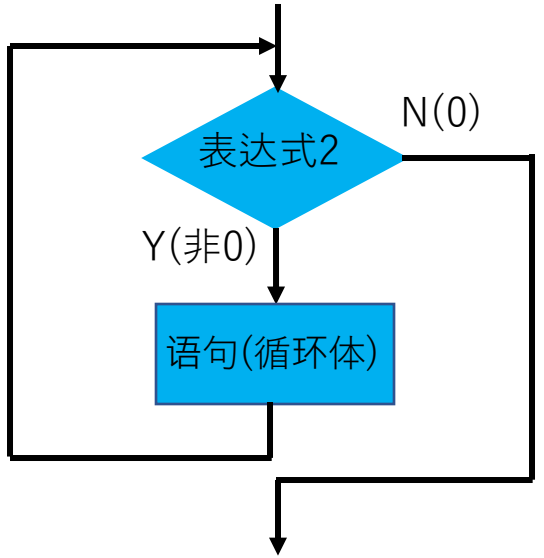
- 表达式数量不同，1个 vs. 3个
- while的表达式控制循环，与for的第二个表达式类似
- while不管循环变量的初始化与递增或递减

while 表达式只负责条件判断，初始化与条件改变在表达式之外完成！

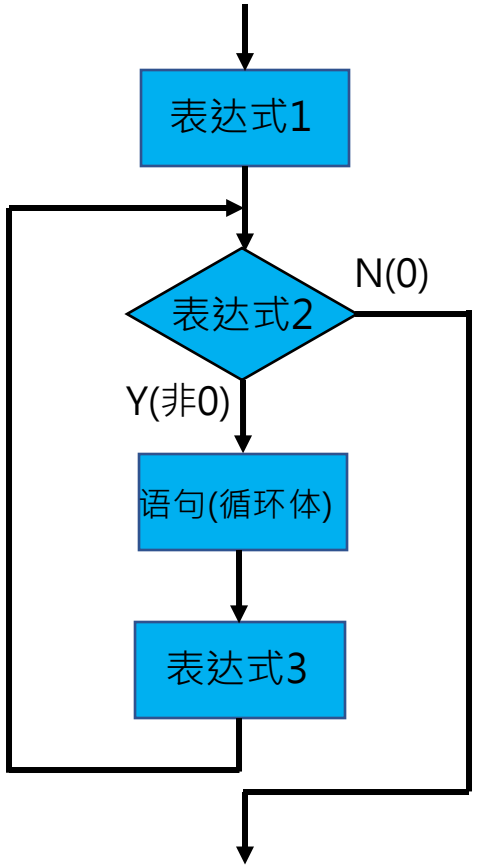
讨论：while 与 for 有哪些关键区别？



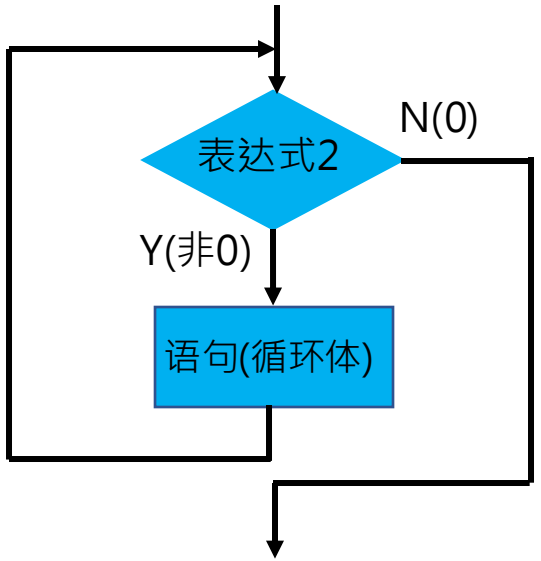

```
for(表达式1; 表达式2; 表达式3)
{
        语句;
}
```




```
while(表达式)
{
        语句;
}
```




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




```
while(表达式)
{
        语句;
}
```

- for适用于已知循环次数的场景
- 未知循环次数的循环可否等价转化?

课堂练习：for-while 以及 while-for转换

```
for(double x = a + dlt/2; x < b;) {
    integral + =(x*x + 5*x +1)*dlt;
    x += dlt;
}
```

```
double x = a + dlt/2;
while(x < b){
    integral + =(x*x + 5*x +1)*dlt;
        x += dlt;
}
```

```
while(p > 1e-6){
    ex += p;
    ++i;
    p = p*x/i;
}
```

```
for(; p > 1e-6;){
    ex += p;
        ++i;
        p = p*x/i;
}
for (p = 1; p > 1e-6; p*=x/i) {
        ex += p;
        ++i;
}
```