电子信息学院

算法与数据结构
(基于现代C++的方法及实践)
ALGORITHM & DATA STRUCTURE IN MODERN C++

第3章 遍历、迭代与递归

王文伟 Wang Wenwei, Dr.-Ing.
Tel: 18971562600
Email: wwwang@aliyun.com
课程QQ群：珞珈EIS数据结构与算法，668792335

电子信息学院 Table of Contents

本章位置

本章首先介绍循环、遍历、迭代和递归的相关概念，然后在对比中分析各自的不同特性，揭示循环、遍历、迭代和递归在算法实现中存在广泛的应用。

电子信息学院 Table of Contents

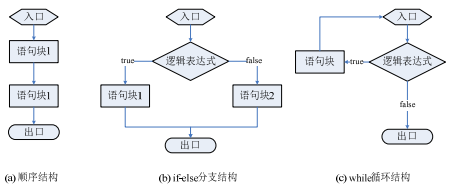
3.0 Introduction

- 数据处理常要求遍历整个集合。遍历：按照某种次序访问集合中的所有元素，并且每个元素仅被访问一次。
- 复杂问题的求解常包含基本操作的重复运行，重复基本操作的常用方式有迭代和递归。迭代一般利用循环结构，通过某种递推式，不断更新变量新值，直到得到问题的解为止。计算机程序往往用大量的迭代来解决复杂的问题。
- 递归则是算法中存在自调用，将大问题化为相同结构的小问题来求解。递归是一种有效的算法设计方法，是解决许多复杂问题的重要方法。

3.1 高级编程语言中的循环结构与遍历操作

- 重复执行一段代码的最基本方法是将它放在循环（loop）结构中，循环结构有三个要素：循环变量、循环体、循环终止条件。
- 循环：在满足循环条件时，循环体内的代码被重复运行，一直到终止条件达到，才结束整个循环结构。

3.1.1 循环结构



(a) 顺序结构 (b) if-else分支结构 (c) while循环结构

---

for语句的定义格式举例如下：

```
for(int i=0; i<10; i++){
    <语句块>;
}
```

基于范围的for语句的定义格式举例如下：

```
for (string& s: args){//enumerable集合
    <语句块>;
}
```

while语句的定义格式举例如下：

```
int i=0;
while(i<10){
    <语句块>;
    i++;  }
```

基于范围的for语句

- 从编写代码的角度，基于范围的for语句比其他循环语句更简洁方便地表达对集合所有元素的遍历操作。
- 在循环构造上免除定义循环计数变量及对循环终止条件的测试，语句简洁而且能防止不经意的错误。
- 能在基于范围的for语句中遍历的数据集合称为可枚举的（enumerable）集合，又称作可迭代的（iterable）集合，C++数组以及标准库中拥有begin()和end()函数的多种容器等都是可枚举的类型。

3.1.2 C++中遍历容器的常用范式

1) 基于下标的遍历：index++: [0, count)

```
int n = 100, sum = 0;
for(int i=0;i<n;i++)
   sum += a[i];
```

```
int n = 100, sum = 0;
for(int i=0; i<v.size(); i++)
   sum += v[i];
```

2) 基于迭代器的遍历：*iterator++: [first, last)

3) 基于范围的for循环遍历：item: container

```
for(auto it= v.begin(); it!=v.end(); it++)
    doSomething(*it);
```

需定义迭代器变量及对循环终止条件的测试，可以通过容器类的begin()和end()成员函数为迭代器变量赋初值和终值。用auto可以简化类型声明。迭代器变量在使用上形同指针。C++标准库算法广泛支持迭代器。

3.2 迭代

- 迭代（iterate）的原义是一种不断用变量的旧值递推新值的过程。迭代过程一般利用循环结构，让变量从初值出发，通过某种递推式，不断更新变量新值，直到得到问题的解为止。
- 高层问题的求解包含一些基本操作的重复运行。在循环体代码中，与求解问题相关的变量在每一轮中将根据某种规则而更新，并作为下一轮循环计算的初始值。整个循环结构通过一个迭代的过程来完成数学中的递推公式所表达的功能。
- 迭代常用来指根据递推公式循环演进逐步接近结果的编程思想，以迭代过程为显著特点的算法，就时常归为迭代算法。

使用迭代思想完成算法的实现要解决三个方面的问题

1. 迭代变量的确定：在可以用迭代解决的问题中，至少存在一个直接或间接地不断由旧值递推出新值的变量，称为迭代变量。
2. 建立迭代公式：迭代公式确定迭代关系，是指如何从变量的前一个值推出其下一个值的公式（或关系）。
3. 对迭代过程进行控制：迭代过程的控制通常分为两种情况：一种是所需的迭代次数是个确定的值，可以预先计算出来，此时可以使用一个固定的循环来控制迭代过程。另一种是所需的迭代次数无法预先确定。此时应进一步分析结束迭代过程的条件，在每一轮迭代末尾根据对结束条件的检测来控制迭代过程的进行。

迭代算法

- 包含某种循环迭代过程的算法，有时称为迭代算法。数值分析中有大量的迭代算法，如：牛顿迭代法，高斯迭代消元法，最速下降法等。
- 在这些算法中，所谓迭代，就是从一个初始估计$\boldsymbol{x}^{(k)}$出发，按照某种规则（又称递推公式）求出后继点$\boldsymbol{x}^{(k+1)}$，用$k+1$代替k，重复以上过程，这样便产生点列$\{\boldsymbol{x}^{(k)}\}$，在一定条件下它收敛于原问题的解。

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

求多元函数最小值的梯度下降法

基本迭代公式

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$$

- 其中$\boldsymbol{d}^{(k)}$是从$\boldsymbol{x}^{(k)}$出发的函数值下降方向，称之为搜索方向，沿这样的函数值下降方向迭代，在一定条件下收敛于函数的极小值的点。α_k是控制迭代速度的参数，称之为搜索步长。

[例3.2]利用迭代方法计算阶乘函数$f(n) = n!$

$p = p \times i$

```
int factorial(int n) {
   int p = 1;
   for (int i = 2; i <= n; i++) {
      p *= i;
    }
    return p;
}
```

```
int main() {
  int n = 6;
  cout<<n<<
   "! = "<<
  factorial(n)
     <<endl;
  return 0;
}
```


3.3 递归recursion

- C/C++一类语言中，若一个函数直接或间接地调用自己，则称这个函数是递归函数；
- 递归（recursion）是数学定义和计算中的一种思维方式，用对象自身来定义一个对象；在程序设计中常用递归方式来实现一些问题的求解。
- 递归可以出现在算法和数据结构的定义中。
  - 存在自调用的算法称为递归算法。
  - 若一个对象用它自己来定义自己，则称这个对象是递归的。

1. 递归算法（recursive algorithm）
2. 递归数据结构（recursive data structure）

3.3.1. 递归算法

- 递归算法将待求解的问题推到比原问题更简单且解法相同或类似的问题的求解。只要解决了子问题，那么原问题就迎刃而解。
- 不断分解原问题，直至分解后的子问题可以直接解决时，就停止分解。这些可以直接求解的问题称为递归结束条件。
- 数学中常见函数的递归定义和计算。一方面给出被定义函数在某些自变量处的值，另一方面则给出由已知的被定义函数值逐步计算未知的被定义函数值的规则。

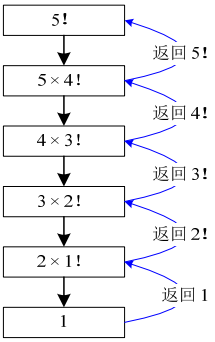
阶乘函数 $$n! = \begin{cases} 1 & n = 0,1 \\ n \times (n-1)! & n \geq 2 \end{cases}$$

Fibonacci数列 $$f(n) = \begin{cases} n & n = 0,1 \\ f(n-1) + f(n-2) & n \geq 2 \end{cases}$$

[例3.3] 阶乘函数$n!$的递归定义式

$$n! = \begin{cases} 1 & n = 0,1 \\ n \times (n-1)! & n \geq 2 \end{cases}$$

```
int f(int n){
  if(n==0)
    return 1;
  else {
   return n * f(n-1);
  }
}
```

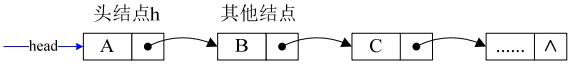


3.3.2. 递归与迭代的比较

- 递归的特点是把一个复杂的算法分解成若干相同的、可重复的步骤，思路清晰、代码简洁、逻辑比较容易理解。
- 递归意味着大量的函数调用，时间效率和空间效率比较低。
  - 递归函数每一次调用自身的时候，该函数都没有退出，在函数调用过程中，系统将在系统栈中为函数分配临时工作空间，当递归深度越深，系统栈空间的占用就越大，可能造成系统栈的溢出。
  - 调用函数因需传递参数及保护现场，也会引起一定的运行时间开销。
- 迭代算法思路上可能没有递归算法简洁，但是其运行时间正比于循环次数，而且没有调用函数引起的额外时间和内存空间开销，因而算法的时间效率和空间效率都很高。
- 一般来说，递归都可以用迭代来代替，如果某算法有迭代式和递归式两种实现，则从执行效率出发，选择使用该算法的迭代式实现。

3.3.3. 递归数据结构

- 有些数据结构是递归定义的。例：树（tree）结构：树T是由n个结点组成的有限集合，它或者是棵空树，或者包含一个根结点和零或若干棵互不相交的子树。
- 单向链表结点也可以递归定义为：
  Node＝(data, next_Node)
- 链表Z＝(头结点h,子链表Z_h)
  h代表头结点，Z_h代表头结点的链域指向的子链表



使用递归的方式，定义链表就只需要一种数据结构类型。

本章学习要点

1. 熟练掌握高级编程语言中的循环结构。
   while，do-while，for，基于范围的fo。
2. 理解遍历，掌握常用的几种范式。
3. 理解迭代，掌握使用迭代思想的三个步骤。
4. 理解递归的特性，递归和迭代算法的不同特性。