线性结构(Linear Structure)

2019 年 3 月 18 日

1 线性结构(Linear Structure)

线性结构的特点 是在数据元素的非空有限集中，1）存在唯一的“第一个元素”；2）存在唯一的“最后一个”元素；3）除“第一个”元素之外，每个数据元素均只有一个前驱；4）除“最后一个”元素外，每个数据元素均只有一个后继。 [1]

线性结构的主要内容

1. 线性表（顺序表+链表）
2. 栈和队列
3. 串

当线性表中的数据元素本身也是一个数据结构时，线性表又可以包含 [1] [2]

1. 数组
2. 广义表

但因为这些只是在原数据结构上的扩展，原理上并无区别，不过多介绍。

2 数组和向量(Array and Vector)

线性表（Linear List）的概念包括*顺序存储表*和*链式存储表* [1] [3]。其中顺序存储表本质上与数组或向量的概念一致，这里称之为数组。

数组（Array） 如果集合S由n个元素组成，各个元素之间具有一定线性次序，则可将它们存放于一段物理位置连续的存储空间中，并称之为数组(Array)。 [2]

向量（Vector） 是对数组的一种抽象和泛化，向量使用“循秩（rank）访问”方式[1]。集合中各个元素类型不一定相同，且不一定可以相互比较大小（没有某数值属性）。 [2]

数组或向量，有人直接将其称之为*顺序表* [4]，也有人认为数组是顺序表在“表中数据元素本身也是一个数据结构”的条件下的扩展，而非顺序表 [1]。在不同书籍中，对这一概念的表述与分类略有差别，但本质上是同一概念。

其优点是存储利用率高，存取速度快。

2.1 抽象数据类型接口（ADTI）

对于数组，除构造和析构函数外，只需提供存储和抽取操作。

```
explicit Array(int size);  // 构造函数
~Array();                  // 析构函数
T & operator [](int i);      // 访问第i元素，存储和抽取操作
int GetSize();               // 获取数组长度
```

2.2 数据类型的实现

数组类，需要至少包含一个数组指针，指向数组首地址。同时常需要一个变量记录数组长度，数组长度可以用于数组的声明，甚至进一步用于数组长度的调节等。

数组的操作较为简单，只需注意类构造函数声明出的数组，在析构函数中需要及时释放。

[1]秩与数组下标概念类似。如果某元素前驱元素共r个，则其秩也为r。秩在概念上可以表示该元素创建的先后，因此比数组下标的概念更为抽象与泛化

3 链表(Linked List)

线性表（Linear List）的概念包括*顺序存储表*和*链式存储表* [1] [3]。其中顺序存储表本质上与数组或向量的概念一致，这里只介绍链表（Linked List）。

链表（Linked List） 虽然其各个元素在逻辑上是连续的，但是在物理存储位置上未必连续。上一个元素（结点，node）中包含指向下一个元素位置的*指针域*，由此可以将一个表，链式的串联起来。

链表的实现 一个链表中包含了零个或多个结点（Node），因此一个链表对象包含零个或多个结点（Node）对象，这种关系在面向对象方法中叫做*聚合*（*Aggregation*）关系 [4]。对于聚合关系，通常需要定义多个类（class）来共同实现，一般可以采用两种方式：定义*复合类*（*Composite Class*）和定义*嵌套类*（*Nested Class*）。其中复合类就是分别定义两个类AB，并在类B中，将另一个类A声明为类B 的*友元*（*friend*）。这样类B便开放了类A 的访问权限，类A可以访问B 类中的私有数据成员。而嵌套类就是在一个类A中定义类B。[2]

链表的优点是可以随时延长或缩短；但其缺点是如若要访问某一个元素，必先遍历其所有前驱元素。

3.1 单链表（Singly Linked List）

单链表是最简单的链表，也称线性链表。单链表中每个数据元素占用一个结点（Node），每个结点由两个域组成，一个域存放数据元素，一个存放指向下一个结点的指针。

表示单链表通常需要两个类（class），即结点（ListNode）类和链表（List）类。有必要的话，还需要一个游标（Iterator）类，用于遍历（Traversal）链表。

3.1.1 抽象数据类型接口（ADTI）

```
SinglyLinkedList();      // 构造函数，构造空表
~SinglyLinkedList();     // 析构函数，清空链表
```

[2]详细参看 [4]3.1.2节

```
int Insert(int pos, T x); // 在pos位置前，插入一个新元素x
int Remove(int pos);    // 删除pos位置上元素，后继元素接上
int Display();       // 显示表内元素，用于监控操作正确与否
```

3.1.2 数据类型的实现

插入和删除 对于插入操作，可能出现3种情况：1）待插入位置是链表第一个位置；2）待插入位置是链表最后一个位置；3）待插入位置非头非尾。

清空 因为单链表没有指向上一个元素的指针域，因此清空链表操作很难通过反复删除最后一个元素来实现，但可以通过反复删除第一个元素实现。

带附加头结点的单链表 上述实现过程，无论是插入还是删除，代码中均需要区分插入位置和删除位置是否是第一个位置。代码不整洁，加入头结点之后，单链表的插入和删除无需区分是否是第一个位置。

3.2 循环链表（Circular List）

循环链表的尾结点指向不是NULL，而是指向第一个元素，使得链表形成一个环状。这种链表可以从任意一点开始，遍历链表中所有元素。

3.2.1 抽象数据类型接口（ADTI）

3.2.2 数据类型的实现

3.3 *静态链表（Static Linked List）

3.3.1 应用举例：多项式相加

3.4 双向链表（Doubly Linked List）

3.4.1 应用举例：稀疏矩阵

4 栈(Stack)

栈和顺序表（数组）是不同的数据类型。栈只允许在表的末端，也即栈顶（*top*），进行插入和删除操作。而另一端栈底（*bottom*）不允许进行操作。

栈符合先进后出（LIFO）[3]的特性，顺序栈也也叫作后进先出的顺序表。 [4]

栈可以有两种实现方式，分别是顺序存储方式，和链式存储方式。但因链式存储方式灵活，所以通常采用链式存储方式。

4.1 抽象数据类型接口（ADTI）

除构造和析构操作外，栈应该有压栈(push)操作和出栈(pop)操作。

```
explicit Stack(int maxsize); // 构造函数
~Stack();                    // 析构函数
int Push(T element);  // 压栈操作
T Pop();             // 出栈操作
```

4.2 数据类型的实现：顺序栈

栈需要一个一维数组存放数据，一个指针指向当前位置。另外需要一个最大尺寸，用于限定栈的尺寸。

压栈（Push） 时，指针上移，指向新的位置。如果超出栈的最大尺寸，无法继续压栈，则返回错误信息。

出栈（Pop） 时，指针下移，指向新的位置，同时返回之前指针指向的元素。如果栈空，无法继续出栈，返回错误信息。

如果程序中有多个栈，有的栈会快速膨胀，有的相对空闲，因此会造成存储空间上的浪费。但是可以使用两个栈，相向伸展 [4]，相互调剂，灵活性强。但为了避免空间浪费，可以使用链式栈。

代码清单：

`sequential_stack.h`

[3] Last In First Out

sequential_stack.cpp

4.3 数据类型的实现：链式栈

在程序中需要同时使用多个栈的情况下，采用链式栈可以提高效率。

压栈（Push） 链式栈和链表结构上类似。链式栈的栈顶在表头位置。当需要压栈时，就在表头位置插入一个数据元素。

出栈（Pop） 时，在表头位置删除一个数据元素，同时返回该元素。

代码清单：

linked_stack.h

linked_stack.cpp

4.4 应用举例：括号匹配

输入一串字符串，输出匹配的括号和没有匹配的括号。

代码清单：

bracket_matching.cpp

5 队列(Queue)

队列只允许在一端插入，而在另一端删除元素。其中可以插入新元素的一端叫做队尾（rear），允许删除的一端叫做队头（front）。队列所具有的特性叫做先进先出FIFO（First In First Out）。

5.1 抽象数据类型接口（ADTI）

除构造析构外，需提供入队和出队操作。另外提供一个现实队列内容的函数，用于检查。

```
Queue();
~Queue();
void Enqueue(T x);
T Dequeue();
void Display();
```

5.2 数据类型的实现：循环队列(顺序存储)

队列可以使用数组实现，但是随着队头（front）和对尾（rear）的移动，当队头和队尾均移动到数组的一端，有可能会出现数组存储空间不够用的情况。为了解决这种问题，队列的顺序存储实现可以将数组的前端和后端连接起来，形成环形的表，即在逻辑上形成一个环，称之为循环队列（*circular queue*）

代码清单：

```
sequential_queue.h
sequential_queue.cpp
```

5.3 数据类型的实现：链式队列(链式存储)

链式队列使用链表实现。队头（front）指向链表第一个元素，队尾（rear）指向链表最后一个元素。插入新元素时，在队尾插入；而删除一个元素时，在队头插入。

链式队列没有队列满而溢出的情况，因此相对顺序存储方式，有着灵活高效的优点。

代码清单：

linked_queue.h

linked_queue.cpp

5.4 应用举例：打印二项展开式

二项式各阶的展开式的各项系数，可以组成一个杨辉三角形（*Pascal's triangle*）。

```
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5   10  10   5   1
        ....
```

杨辉三角形

杨辉三角形下一层的值，等于三角形上一层的两值的和。

实现过程是，先将上一层的各数值入队。根据杨辉三角形的性质，依次计算下一层的数值，并入队。与此同时，当上一层的数值使用完毕后[4]，就将其出队。这样，当下一层各个数值完全入队后，上一层数值刚好完全出队。队列保存了当前层的各个数值。

代码清单：

pascals_triangle.cpp

5.5 优先级队列

与普通队列不同的是，普通队列每次出队的都是最早入队的元素（FIFO，先进先出），而优先级队列每次出队的都是优先级最高的元素。这种队列叫做优先级队列，也称优先权队列。

最小优先级队列，每次查找，并出队的是拥有最小优先级的元素；而最大优先级队列，每次查找并出队都是拥有最大优先级的元素。

[4]计算完下一层对应的数值

5.5.1 抽象数据类型接口（ADTI）

与普通队列的接口一致，但需要增加一个调整函数，用于根据各元素优先级，调整各元素位置。因此，可以继承队列类。

```
Queue();
~Queue();
void Enqueue(T x);
T Dequeue();
void Adjust();
void Display();
```

5.6 双端队列（Double-ended queue）

与普通队列不同的是，双端队列的两端均可以进行插入和删除操作。

6 串(String)

串，字符串，是由零个或多个字符的顺序排列所组成的数据结构。其元素是字符，而长度可变。

计算串的长度时，不包含引号，也不包括结束符[5]。

长度为零的串叫做空串；而串内所有元素为空格的，叫做空白串。

非空串中，从其中任意位置开始的连续几个字符所组成的串，称为子串。子串的在原串中的位置由子串第0个字符确定。

6.1 抽象数据类型接口（ADTI）

串需要提供诸如取子串，串拼接等操作。

```
String(); // 构造函数
~String(); // 析构函数

String& operator () (int pos, int len);  // 取子串，截取从pos开始，长度为len的子串
String& operator = (String& x);  // 给字符串赋值
String& operator += (String& x); // 串连接
```

6.2 数据类型的实现

6.2.1 基本操作

主要涉及取子串，赋值，串拼接，操作较为简单，参考代码即可。

6.2.2 串匹配

在串中找到某个子串。

简单的匹配算法：B-F算法 从串中第0个字符开始，依次和子串中字符进行对比。如果所有字符均相同，则认为匹配成功，返回相应的位置；如果有字符不相同，则后移一位，从下一个字符开始，重新和子串进行对比。直至找到正确的子串或匹配失败。

[5]结束符占一个字符

该算法不断回溯，重复操作。时间复杂度为$O(n*m)$，其中原串长度为n，子串长度为m。

改进的匹配算法：KMP算法

参考文献

[1] 严蔚敏. 数据结构（C语言版）. 北京: 清华大学出版社, 2007.

[2] 邓俊辉. 数据结构（C++语言版）(第三版). 北京: 清华大学出版社, 2013.

[3] 李春葆. 数据结构考研指导. 北京: 清华大学出版社, 2002.

[4] 殷人昆. 数据结构：用面向对象方法与C++描述. 北京: 清华大学出版社,1999.