**第一章 绪论**

（6）

int fun(int n) {

    if(n % 2 == 0)

        return -n/2;

    else

        return (n-1)/2 + 1;

}

（7）O(n2)

（10）1）线性结构 2）集合结构 3）集合结构

（11）1）树形结构 2）图形结构 3）树形结构

**第二章 线性表**

（1）虚函数是为了实现C++面向对象的多态性。多态性允许你使用基类的指针或引用来操作派生类的对象，同时调用派生类中的特定实现。定义一个虚析构函数意味着当通过基类指针删除对象时，会首先调用派生类的析构函数，然后调用基类的析构函数。对于采用链接实现的线性表，可以确保所有结点的内存都被正确释放。

（2）没有头结点的单链表在插入和删除数据时，需要特别处理第一个数据元素。这是因为头元素所在的结点没有前驱结点，需要单独考虑。而有了头结点后，头元素的前驱结点就是头结点，使得所有存储数据元素的结点都有前驱结点，从而可以统一处理，简化算法。

（5）顺序存储：插入操作最坏O(n)，最好O(1)；删除操作最坏O(n)，最好O(1)；查找操作 O(n)。链接存储：插入操作O(1)；删除操作O(1)；查找操作O(n)。

（6）2）B 4）B

（8）顺序表类中存在动态变量，若以线性表类型对象作为返回值将调用拷贝构造函数，而seqList类没有定义拷贝构造函数，使用默认的拷贝构造函数将使动态变量访问出现错误结果。

**第三章 队列与栈**

（3）

（4）n-1；(rear + 1) % maxSize == front

（5）不正确。为了效率起见，在循环队列中出队（删除元素）并不会修改元素值，只会移动front指针。在入队操作中，rear会递增，并在到达数组末尾后循环至数组开头继续向后，此时指向的元素值并非初始化时设定的特殊值，而是之前已被删除的元素。

（6）1）能，操作如下：进栈A，B，C，D，E；出栈E，D，C，B，A。2）能，操作如下：进栈A，B，C；出栈C；进栈D，E；出栈E、D、B、A。3）能，操作如下：进栈A；出栈A；进栈B；出栈B；进栈C；出栈C；进栈D；出栈D；进栈E；出栈E。

（7）计算机在处理表达式时，可以按自左至右顺序读入表达式的各个运算数、运算符。读入运算数时可以直接输出，因为运算数的相对位置在常规表达式与后缀表达式中是一致的。运算符位置则因为优先级不同发生了变化。读入运算符时，由于不知道能否运算，需要先把它保存起来。如果后面一个运算符比它优先级低，才可输出刚才保存起来的运算符，表示它可以运算了，即相应的表达式才可转换为后缀的形式。

使用栈进行后缀表达式转换的方法如下：

设立一个用于保存运算符的栈，首先将栈清空。自左至右依次读入运算符和运算数，然后执行以下操作：

1. 若读入的是运算数，立即输出。
2. 若读入的是左括号，则进栈。
3. 若读入的是右括号，则对应的左括号和右括号之间的运算符都可进行运算。将栈中的运算符依次出栈并输出，直到遇到相应的左括号为止。将左括号出栈但不输出。
4. 如果读入的是运算符比较其与栈顶运算符的优先级：

* 如果栈为空，或者栈顶为左括号，或者当前运算符优先级高于栈顶运算符，则将当前运算符进栈。
* 否则，将栈顶运算符出栈并输出，然后继续比较新的栈顶运算符，直到遇到上述情况之一。

1. 在读入操作结束时，将栈中所有的剩余运算符依次出栈输出。

1）

|  |  |  |
| --- | --- | --- |
| **操作** | **输出** | **栈** |
| 读入操作数6，输出 | 6 | [] |
| 读入运算符/，进栈 | 6 | [/] |
| 读入操作数3，输出 | 6 3 | [/] |
| 读入运算符×，比较优先级，栈顶运算符/出栈并输出，×进栈 | 6 3 / | [×] |
| 读入操作数4，输出 | 6 3 / 4 | [×] |
| 读入运算符+，比较优先级，栈顶运算符×出栈并输出，+进栈 | 6 3 / 4 × | [+] |
| 读入操作数1，输出 | 6 3 / 4 × 1 | [+] |
| 读入完毕，栈中所有运算符出栈输出 | 6 3 / 4 × 1 + |  |

2）

|  |  |  |
| --- | --- | --- |
| **操作** | **输出** | **栈** |
| 读入(，是左括号，进栈 |  | [(] |
| 读入1，是操作数，输出 | 1 | [(] |
| 读入+，是运算符，进栈 | 1 | [(, +] |
| 读入3，是操作数，输出 | 1 3 | [(, +] |
| 读入)，是右括号，弹出并输出栈顶运算符直到遇到左括号 | 1 3 + | [] |
| 读入×，是运算符，进栈 | 1 3 + | [×] |
| 读入(，是左括号，进栈 | 1 3 + | [×, (] |
| 读入9，是操作数，输出 | 1 3 + 9 | [×, (] |
| 读入–，是运算符，进栈 | 1 3 + 9 | [×, (, –] |
| 读入2，是操作数，输出 | 1 3 + 9 2 | [×, (, –] |
| 读入×，是运算符，优先级高于栈顶–，进栈 | 1 3 + 9 2 | [×, (, –, ×] |
| 读入4，是操作数，输出 | 1 3 + 9 2 4 | [×, (, –, ×] |
| 读入)，是右括号，弹出并输出栈顶运算符直到遇到左括号 | 1 3 + 9 2 4 × – | [×] |
| 读入–，是运算符，优先级低于栈顶×，将栈顶×弹出并输出，–进栈 | 1 3 + 9 2 4 × – × | [–] |
| 读入6，是操作数，输出 | 1 3 + 9 2 4 × – × 6 | [–] |
| 读入/，是运算符，进栈 | 1 3 + 9 2 4 × – × 6 | [–, /] |
| 读入2，是操作数，输出 | 1 3 + 9 2 4 × – × 6 2 | [–, /] |
| 读入完毕，栈中所有运算符出栈输出 | 1 3 + 9 2 4 × – × 6 2 / – | [] |

（8）

1）(9 - (5 + 2)) × 3

2）5 × (7 - 2 × 3) + 8 / 2

（9） 用栈来完成后缀表达式的计算过程如下：遇到操作数时，将其压入栈中。遇到运算符时，从栈中弹出所需数量的操作数（两个），然后进行运算，并将运算结果压入栈中。重复上述步骤，直到处理完整个后缀表达式。最终栈中应该只剩下一个元素，即表达式结果。

（11）D() D() A() D() B() C()

（12）C

**第四章 树与优先级队列**

（1）n-1

（2）

（3）这棵二叉树的每个结点都只有一个子结点，或者是左子结点，或者是右子结点，但不会同时存在。树的形状类似于一个单链表。

（10）

第五章 集合与静态查找表

第六章 动态查找表

第七章 排序

第八章 外部查找与排序

第九章 图

第十章 最小生成树与最短路径问题

第十一章 通用算法