# 《数据结构》实验报告

|  |  |
| --- | --- |
| 姓名： 吴泽同 | 学号： 084622109 |
| 班级： 计算机221 | 日期： 2023/9/24 |
| 程序名： 实验3 栈和队列的应用 | |

### 一、上机实验的问题和要求：

1. 栈和队列综合应用：进制转换**（必做题）**

使用栈和队列结构，将用户输入的非负十进制数x，将其转换成二进制数输出。如（3.25）10=（11.01）2

1. 队列的应用：延时队列**（必做题）**

延时队列DelayQueue是一种元素有序的队列结构，常用于多任务系统、缓存系统等。延时队列中每个元素有delay值，最早到期的元素排在队头，最早出队列；当元素入队列时需要插入适当的位置，保持延时队列的有序性。

若某多任务系统，多个用户提交了任务集合Task={task1,task2,…,taskn}，每个任务taski包含任务编号TaskNumi和可以容忍的最大等待时间TimeWaiti，服务器端设置延时队列，当接收到任一用户任务后立即加入延时队列，服务器从延时队列中按序执行任务。

请使用常规队列的结构和操作模拟延时队列，要求通过继承顺序队列或链队列实现延时队列类，实现延时队列的入队列EnDelayQueue、出队列DeDelayQueue操作，延时队列有最大容量限制。

给一些样例数据和输入输出案例，不要代码

3、**栈和队列的应用**后缀表达式：教材P97实验2**（选做题）**

后缀表达式。运算符在两个运算对象的中间(如 4+2)称为中缀表达式运算符在两个运算对象的后面(如 4 2+)称为后缀表达式,也称逆波兰式(reversePolish notation)。例如,中缀表达式(4+2) \* 3-5对应的后缀表达式为42+3 \* 5-.对算术表达式的后缀形式仅做一次扫描即可得到表达式的运算结果,而无须考虑优先级和括号等因素。因此,很多编译程序在对表达式进行语法检查的同时,将其转换为对应的后缀形式。将一个中缀表达式转换为对应的后缀表达式只需用一个栈存放运算符,中缀表达式(4+2)\* 3-5转换对应的后缀表达式的过程如表 3-2 所示。设计算法实现将中缀表达式转换为对应的后缀表达式。

### 二、程序设计的基本思想，原理和算法描述：

（包括程序的结构，数据结构，输入/输出设计，符号名说明等）

1、栈和队列综合应用：进制转换**（必做题）**

实现思路：

使用栈和队列结构可以实现将十进制数转换为二进制数的过程。以下是一个基本的思路：

接收用户输入的非负十进制数 x。

创建一个空栈和一个空队列，用于存储计算过程中的中间值和最终结果。

将 x 逐位进行二进制转换，直到 x 变为 0。

反复进行以下步骤：

将 x 对 2 取余数（即 x % 2），得到当前位的二进制值。

将余数入栈。

将 x 除以 2 取整数部分（即 x / 2），作为下一轮的被除数。

当 x 变为 0 时，从栈顶开始依次将数字出栈，并入队列。

依次从队列中取出数字，即为转换后的二进制数。

1、进制转换

（1）基本思想：将十进制数x分离出整数部分n和小数部分m（即x=n.m），再将n和m分别转换成二进制形式n’和m’，最后得到x的二进制表示n’.m’

（2）分离十进制数x的整数部分和小数部分：

double x,m ; int n;

①分离整数部分：

方法一：强制类型转换，n = (int)x;

方法二：使用math.h头文件的floor函数，n = floor (x);

②分离小数部分：m = x – n;

（3）十进制整数n转换成二进制表示：除2取余数，逆序输出，直到商为0

while( n!=0) { int k=n%2; n=n/2; } //每次求出的k即为结果

（4）十进制小数m转换成二进制表示：乘2取整数，正序输出，直到积为0

while(m!=0) { m = m\*2; int k = (int) m; m = m-k; }//每次求出的k即为结果

（5）输出结果：先输出整数部分二进制，输出小数点，再输出小数部分二进制

2、队列的应用：延时队列**（必做题）**

实现思路：

1. 创建一个延时队列类（例如`DelayQueue`），并继承顺序队列或链队列的实现。

2. 在延时队列类中添加一个私有数据成员`maxDelay`，表示最大延时时间。

3. 在入队操作`EnDelayQueue()`中，对元素的延时时间进行判断：

- 如果元素的延时时间小于等于最大延时时间`maxDelay`，则需要找到合适的位置插入元素，以保持队列的有序性。可以遍历队列中已有的元素，找到第一个延时时间大于待插入元素的位置，并将其插入该位置。

4. 在出队操作`DeDelayQueue()`中，直接调用父类的出队操作，将队头元素出队。

5. 根据需要，可以添加其他操作，如获取队头元素、判断队列是否为空等。

3、后缀表达式：教材P97实验2**（选做题）**

实现思路：

将中缀表达式转换为后缀表达式的思路如下：

1. 创建一个栈，用于存放运算符。

2. 创建一个空字符串，用于存放后缀表达式。

3. 从左到右遍历中缀表达式的每个字符。

4. 如果字符是操作数（数字），则直接添加到后缀表达式中。

5. 如果字符是运算符，则与栈顶运算符进行比较：

- 如果栈为空或栈顶为左括号"("，则将运算符入栈。

- 如果栈顶运算符的优先级低于当前运算符，或者栈顶为左括号"("，则将当前运算符入栈。

- 否则，将栈顶运算符出栈并添加到后缀表达式中，重复此步骤直到满足上述条件，然后将当前运算符入栈。

6. 如果字符是左括号"("，则将其入栈。

7. 如果字符是右括号")"，则将栈中的运算符依次出栈并添加到后缀表达式中，直到遇到左括号"("为止。然后将左括号出栈，但不添加到后缀表达式中。

8. 遍历结束后，将栈中剩余的运算符依次出栈并添加到后缀表达式中。

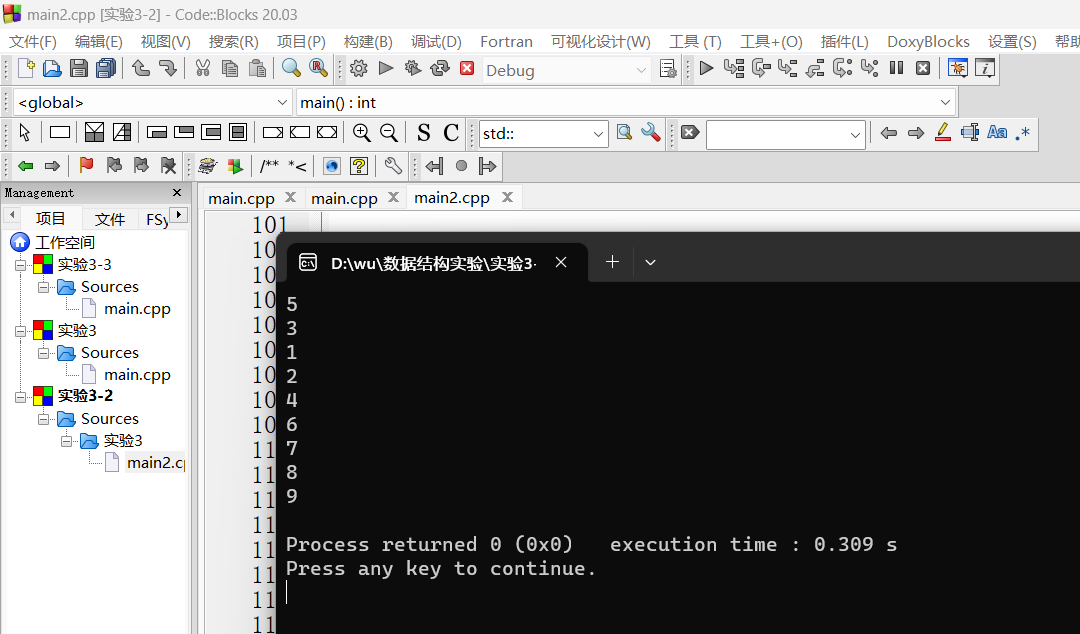
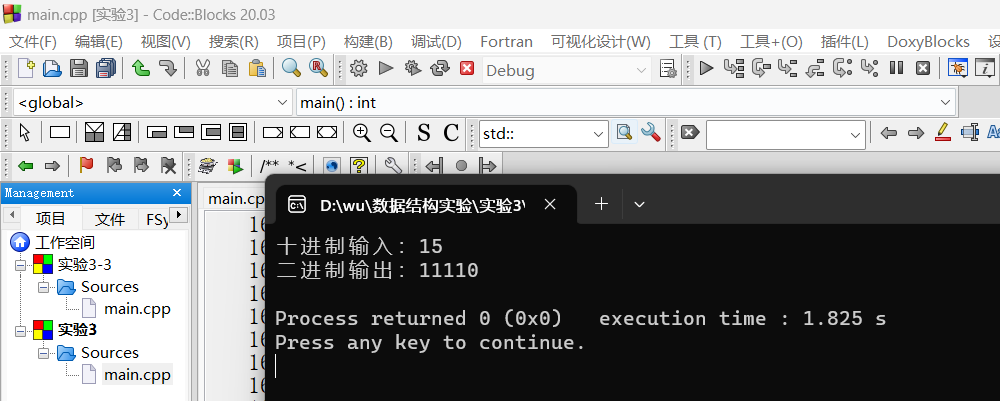
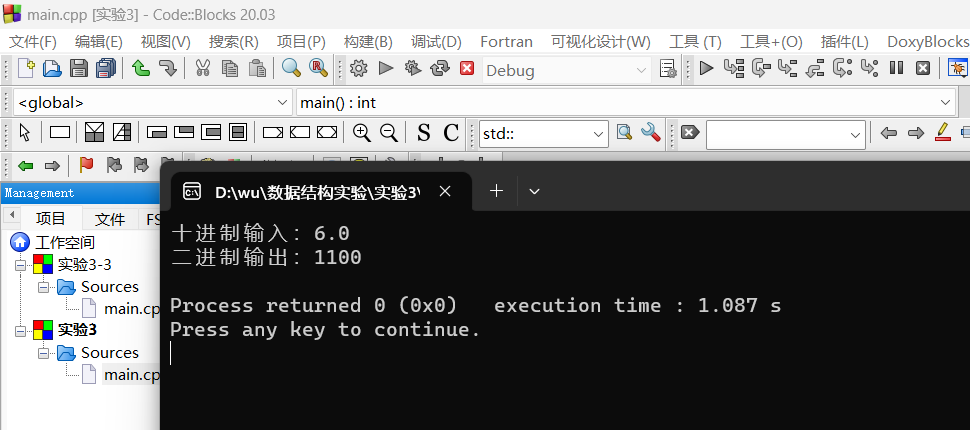
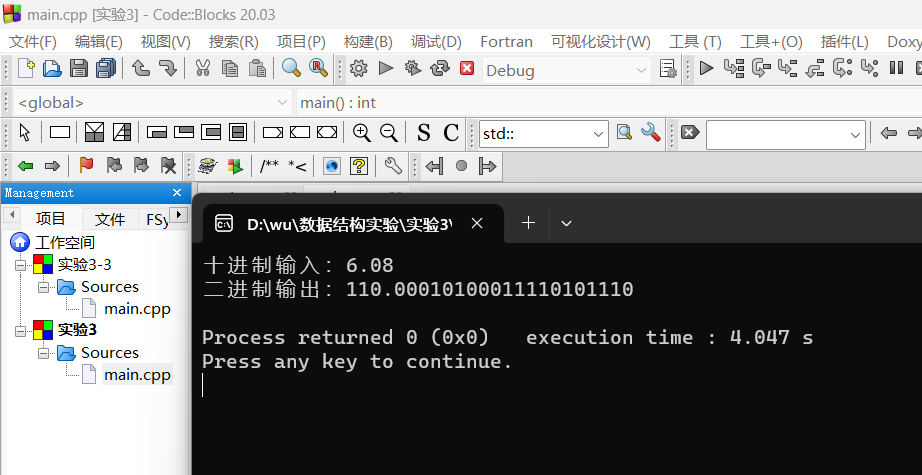
9. 后缀表达式即为转换后的结果。

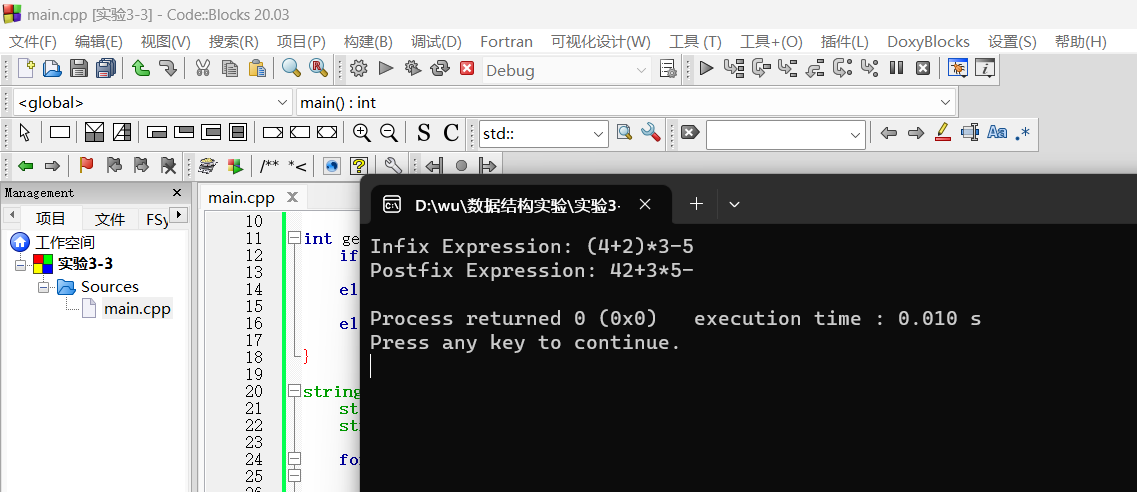
### 三、源程序及注释（说明每个文件的文件名即可，电子档的\*.h和\*.cpp文件压缩传到FTP上）：

实验3.1.zip实验3-2.zip实验3-3.zip

### 四、运行输出结果：

（可以将运行结果抓图贴至此处）





### 五、调试和运行程序过程中产生的问题及采取的措施：

1. 对于栈和队列综合应用的进制转换，可能出现栈或队列操作时的错误，例如入栈操作未成功将元素入栈或出队列操作未成功将元素出队列。解决方法是检查代码中与栈和队列相关的操作，并确保其正确性。

2. 在延时队列的实现过程中，需要注意判断元素的延时时间和最大延时时间的大小关系，以确定是否需要插入合适的位置。同时，还需要确保队列的有序性。可以通过添加适当的判断条件和遍历队列进行比较来解决这个问题。

3. 在后缀表达式转换过程中，需要正确处理运算符的优先级和括号的匹配。可以使用栈来存储运算符，并根据运算符的优先级来进行比较和出栈操作。还需要注意处理括号的情况，确保括号的匹配和顺序。在处理运算符时，可以使用一个映射表来存储每个运算符对应的优先级，方便比较和判断。

4. 在调试和运行程序过程中，可能会出现一些逻辑错误或边界情况的处理问题。可以通过添加合适的测试数据和边界值来进行调试，同时也可以使用调试工具来跟踪程序的执行过程，并查看变量的值和计算结果，以找出错误的原因并进行修正。

### 六、对算法的程序的讨论、分析，改进设想，其它经验教训：

1. 在栈和队列综合应用的进制转换中，使用栈和队列结构能够有效地将十进制数转换为二进制数，具有较高的效率和简洁的逻辑。

2. 延时队列的实现可以通过继承顺序队列或链队列来进行，通过添加最大延时时间的判断和合适的插入位置来保持队列的有序性。这样可以方便地模拟延时队列的功能。

3. 后缀表达式的转换过程中，可以使用栈来存储运算符，并通过比较优先级和处理括号来确定运算符的顺序。这种转换方法可以简化表达式的求值过程，减少了需要考虑的因素。

4. 在编写算法和程序时，需要考虑各种边界情况和特殊情况，确保程序的正确性和鲁棒性。同时，可以使用注释和合适的命名来提高代码的可读性和可维护性。在调试过程中，可以使用合适的测试数据和调试工具，以及错误处理和异常捕获机制来提高程序的健壮性。

### 七、对数据结构教学的意见和建议：

1. 数据结构教学可以注重理论与实践相结合，通过实际应用场景的案例来讲解不同数据结构的应用和实现思路，既加深了学生对数据结构的理解，也提高了他们的编程实践能力。

2. 在课堂上可以进行一些互动式的讨论或小组活动，让学生参与到问题的解决过程中，增强他们的动手能力和团队合作能力。

3. 可以引导学生思考数据结构的选择与设计，根据具体问题的需求选择合适的数据结构，并掌握其特点和使用方法。同时，要着重培养学生的算法设计能力，使其能够灵活运用所学的数据结构解决实际问题。

4. 强调代码规范和良好的编程习惯，让学生在实践中养成良好的代码风格和调试技巧，提高代码的可读性和可维护性。同时，也要加强对程序调试和错误处理的教学，培养学生解决问题的能力。