实验任务2.1：正规表达式转NFA算法及实现

一、实验目的

1. 掌握正规表达式与有限自动机的基本概念和转换方法。
2. 了解非确定有限自动机（NFA）的构建过程。
3. 熟悉编程实现正规表达式到NFA转换的算法。
4. 提高编程能力和算法设计的技能。

二、实验内容

1. **理论背景**：正规表达式是一种用于描述词法单元的形式化表示法，而NFA是一种用于词法分析的状态机。正规表达式可以通过算法转化为NFA，从而实现对字符串的模式匹配。
2. **任务描述**：实现正规表达式到NFA的转换算法，并验证生成的NFA对给定输入字符串的接受性。同时，设计适合NFA的数据结构，为后续NFA转DFA、DFA最小化等实验任务提供基础支持。
3. **实验步骤**：
   - 解析输入的正规表达式。
   - 构建对应的NFA，包括处理基本符号、连接、并联（或操作）、闭包（星号操作）等运算。
   - 设计并实现合理的数据结构表示NFA，如状态集合、转移关系、初始状态和接受状态。
   - 对NFA进行模拟，验证其是否接受给定的输入字符串。
4. **案例分析**：给定一个简单的正规表达式（如`a(b|c)*`），手动推导其NFA，并用程序实现自动生成NFA的过程。

三、实验要求

1. **输入输出要求**：
   - 输入：正规表达式和多个测试字符串。
   - 输出：生成的NFA状态集合及其转换关系，指明每个测试字符串是否被NFA接受。
2. **算法要求**：
   - 支持基本的正规表达式运算符，如连接（`ab`）、或（a|b）、闭包（a*`）。
   - 实现Thompson构造法，将正规表达式分解为基本操作，然后逐步合成NFA。
3. **数据结构要求**：
   - 设计合理的数据结构来表示NFA（如图的表示方式），应包括状态集、状态转移表、初始状态和接受状态的表示。
   - 数据结构需具备扩展性，以便在后续实验中使用，如NFA到DFA的转换、DFA的最小化。
   - 考虑实现状态的唯一标识符，支持对状态进行增删查操作的高效实现。
4. **程序要求**：
   - 使用C/C++、Java、Python等语言编写程序，代码结构清晰，具备良好的注释。

- 提供详细的实验报告，包括算法设计、实现过程、测试结果和问题分析。

5. **实验报告要求**：【整合到最后提交的个人所有实验报告中，加上目录】
   - 描述实验目的和内容。
   - 解释算法实现的步骤和数据结构的设计思路。
   - 给出测试用例和结果，分析测试数据的正确性。
   - 总结实验的收获和遇到的挑战。

四、实验指南

1. **准备工作**
   - 复习正规表达式和NFA的相关理论知识，特别是Thompson构造法的原理。
   - 了解常见的数据结构（如图的表示方法）在编译器中的应用。
   - 安装编程环境（如Python的IDE，C/C++的编译器等），熟悉相关编程工具的使用。
2. **实验步骤**

- **步骤1**：解析输入的正规表达式，将其拆分为基本符号和运算符。
  - 查阅相关文献，实现正规表达式从中缀形式变换为后缀形式
- **步骤2**：按照Thompson构造法的规则，实现以下几种基本转换：
  - 单个符号的NFA。
  - 两个NFA的连接（拼接）。
  - 两个NFA的并联（或操作）。
  - 一个NFA的闭包（星号操作）。
- **步骤3**：合并所有子NFA，构建完整的NFA。
- **步骤4**：设计合理的数据结构表示NFA，如使用邻接表或状态转移矩阵表示转换关系。
- **步骤5**：实现NFA的模拟算法，判断输入字符串是否被NFA接受。
- **步骤6**：进行测试，使用几个正规表达式和测试字符串验证程序的正确性。

3. **注意事项**
   - 确保算法处理优先级问题，如星号优先级高于连接，连接高于或运算。
   - 数据结构的设计要考虑后续任务的需求，如状态和转换关系的查找效率。
   - 在程序中增加异常处理和边界情况的测试。
4. **扩展任务（可选）**

- 增加可视化功能，利用Graphviz等库，实现NFA和DFA的可视化展示。
- 支持扩展的正规表达式运算符，如`+`（一次或多次重复）和`?`（零次或一次）。
- 优化NFA结构，尝试将其转化为最小化的DFA。

五、参考资料

- **编译原理教材**：大多数编译原理教材都会涵盖正规表达式和NFA的转换算法。
- **在线课程**：如Coursera、edX等平台上的编译原理课程。
- **技术博客和教程**：知乎或CSDN等技术平台上有较多关于正规表达式和NFA转换的详细解释和示例。
- **开源项目**：查看GitHub等代码托管平台上的开源项目，了解其他人是如何实现这一算法的。