数据结构实验二说明

1 概览

本次实验包含两个独立的编程任务，分别重点考察**栈**和**队列**这两种线性数据结构的应用。学生需要完成两个核心程序：

1. **表达式求值**：实现一个能处理加、减、乘、除、乘方和括号的表达式计算器，核心是使用栈来解析和计算表达式。
2. **机器人吃金币**：在一个二维网格中，计算机器人在有限步数内能获取的金币最大总价值，核心是使用队列进行广度优先搜索（BFS）来探索所有可能的移动路径。

2 预备知识

2.1 栈及其应用

栈是一种**后进先出** 的线性数据结构。本次实验的表达式求值部分将直接应用栈。关键点包括：

- **中缀表达式转后缀表达式**：利用栈处理运算符的优先级。
- **后缀表达式求值**：利用栈存储操作数并进行计算。
- 你需要理解栈的基本操作：入栈、出栈、访问栈顶元素，并能在C++中实现一个栈类。

2.2 队列及其应用

队列是一种**先进先出** 的线性数据结构。本次实验的机器人吃金币部分将应用队列。关键点包括：

- **广度优先搜索**：队列是实现BFS算法的核心数据结构，用于按层次遍历所有可能的移动状态，确保找到最短路径或最优解。
- 你需要理解队列的基本操作：入队、出队、访问队首元素，并能在C++中实现一个队列类。

2.3 C++ 类与对象

两个任务均通过C++类来实现。请确保你对以下概念有清晰理解：

- 类的声明与定义（ `.h` 头文件和 `.cpp` 源文件）。
- 构造函数与析构函数。
- `public` 、 `private` 成员的访问控制。

- 成员函数的实现与调用。

3 实验框架

实验框架已经提供了基本的项目结构、类定义和函数声明。你需要在指定的位置补全代码以实现功能。

项目主要目录结构如下：

```
Lab2/
├── include/            // 头文件目录
│   ├── list.h          // 线性表虚基类
│   ├── stack.h         // 栈类声明 (需实现)
│   ├── queue.h         // 队列类声明 (需实现)
│   └── unordered_set.h // 简单的集合类（已实现）
├── src/                // 源代码目录
│   ├── 1_Test/         // 用于验证头文件实现的数据结构的测试文件（无需修改）
│   ├── 2_Expression/   // 表达式求值任务
│   │   ├── calculator.cpp // 计算器核心逻辑 (需实现)
│   │   └── main.cpp    // 主程序（已实现）
│   └── 3_Robot/        // 机器人吃金币任务
│       ├── walk.cpp    // 机器人行走逻辑 (需实现)
│       └── main.cpp    // 主程序（已实现）
└── structures/         // 编译辅助文件（无需修改）
```

4 实验任务

4.1 表达式求值 (`src/2_Expression/calculator.cpp`)

本部分要求实现一个支持四则运算、乘方和括号的表达式计算器。

1. **`bool judge()` 函数**
   检查表达式的合法性。需要完成两项检查：
   - **括号匹配**：确保所有左括号都有对应的右括号，且顺序正确。
   - **运算符合法性**：确保运算符不会连续出现（例如，"++", "*/"是非法的）。
2. **`struct element get_ans()` 函数**
   这是计算器的核心函数，负责计算表达式的值。建议的实现思路是“中缀转后缀”再“后缀表达式求值”，两者都需使用栈。你可能需要实现并调用以下辅助函数：
   - `struct element read_num()`：从表达式字符串中读取一个完整的数字（整数或浮点数）。
   - `int priority(char c1, char c2)`：比较两个运算符的优先级。

- `struct element operate(struct element e1, char op, struct element e2)`：执行具体的运算 `e1 op e2` 。

4.2 机器人吃金币（`src/3_Robot/walk.cpp`）

本部分要求计算机器人在有限移动步数内能获得的金币最大价值。

1. **`int compute_distance(int i, int x, int y)` 函数**
   计算从当前坐标 `(x, y)` 移动到第 `i` 个金币所在位置所需的最少步数。移动规则是每次可沿x轴或y轴方向移动1或2格。
2. **`int get_value()` 函数**
   这是本部分的核心函数，计算机器人能获得的最大金币价值。推荐使用**广度优先搜索** 配合队列实现：
   - 将机器人的初始状态（当前位置、剩余步数、已获得价值）入队。
   - 每次从队列中取出一个状态，尝试所有可能的移动（向四个方向移动1或2格），生成新状态。
   - 如果新位置有金币且之前未在同一步数下到达过该位置（需使用集合进行状态去重），则更新获得的价值。
   - 循环直到队列为空或所有步数用完，记录过程中出现的最大价值。

4.3 数据结构实现（`include/stack.h`，`include/queue.h`）

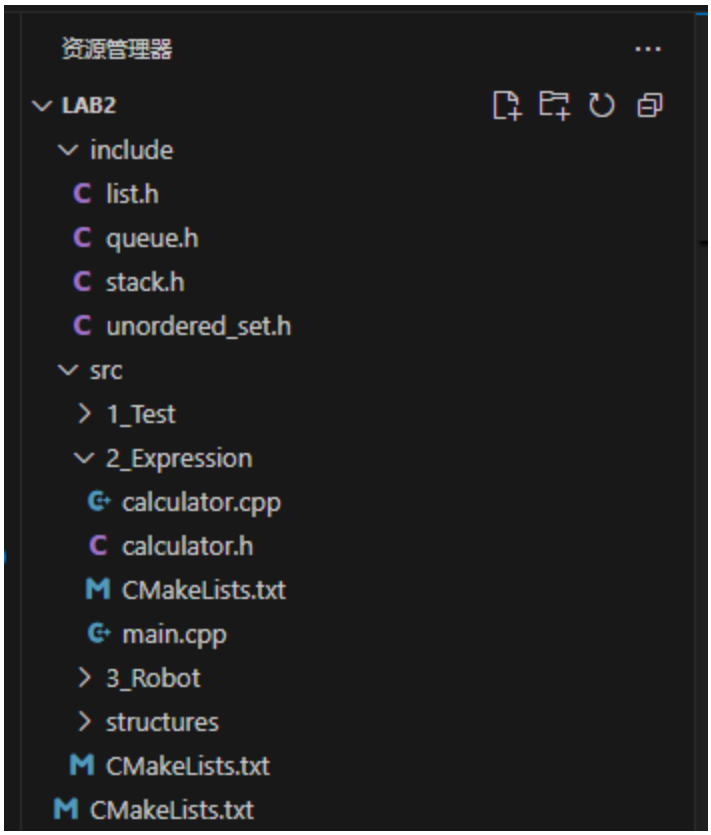
你需要选择使用数组或链表，完整实现栈和队列类中声明的所有成员函数，包括：

- `empty()`，`clear()`
- `push(data)`，`pop()`
- `top()` (对于栈) / `front()` (对于队列)

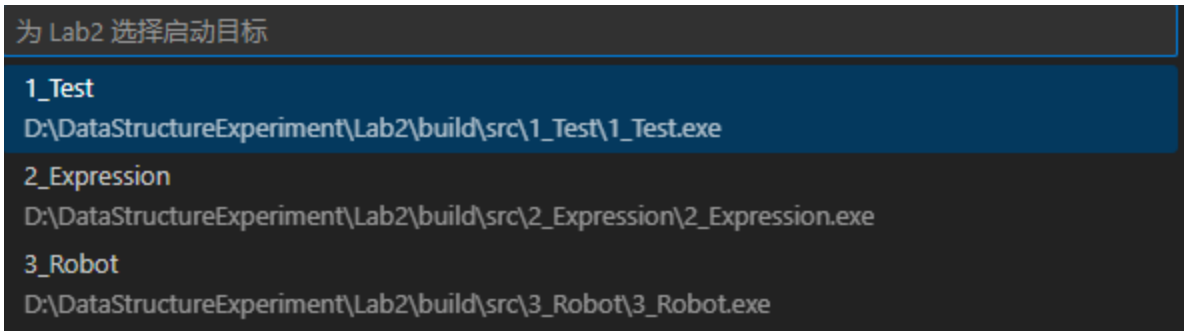
实现完成后可通过 `1_Test` 的 `main.cpp` 检验实现的数据结构操作是否正确，可根据自己需要修改对应的测试。

5 实验运行

使用vscode打开到 `Lab2` 的目录（应包含 `include` 文件夹、 `src` 文件夹和 `CMakeList.txt` 文件）：



然后使用vscode的CMake插件进行构建时，可以选择对应的子项目：



注意：在验收前，你需要进行相应的测试，以验证程序的健壮性，可以尝试自己手动构建测试集进行测试。本次实验检查会使用助教自备的测试集进行测试已测试功能完整性。

6 实验分数安排与检查

6.1 分数安排

- 完成 `include/stack.h` 的实现：**1分**
- 完成 `include/queue.h` 的实现：**1分**
- 完成 `src/2_Expression/calculator.cpp` 的实现：**4分**
- 完成 `src/3_Robot/walk.cpp` 的实现：**3分**
- 现场检查提问：**1分**
- **总分：10分**

6.2 现场检查

- 实验检查截止日期：**2025年11月7日**（请根据实际情况修改）。
- 请在截止日期前的实验课上联系助教进行检查。
- 检查时，需要现场运行两个程序并解释关键代码逻辑。
- 逾期未检查者，本次实验成绩记为0分。

7 附录

7.1 表达式求值算法参考

算法核心参考严蔚敏《数据结构》教材中“表达式求值”案例。请注意，本实验新增了乘方运算 ^ ，其优先级高于乘除。

7.2 机器人问题思路提示

此问题是一个典型的带约束的路径规划问题。由于移动步数有限，暴力枚举所有路径不可行。使用BFS的优势在于它可以按“步数”这个维度层层扩展，确保在步数耗尽前探索所有可达状态，从而找到最优解。关键点在于设计合适的数据结构来记录状态（坐标、剩余步数、当前价值）并进行高效的去重。