数据结构 Lab2

2024 年 10 月 31 日

1. 实验要求

1.1 表达式求值 Expression（stack.h 及 calculator.cpp）

本题来自严蔚敏教材第二版 3.2 节案例 3.3，算法参考严蔚敏教材第一版 3.2.5 节，注意实验中多了一个乘方运算。参考资料详见群文件。

编写程序实现表达式求值，表达式中可能包含的符号有（非负）整数、（非负）浮点数、四则运算符（+-*/）、乘方符号（^）、小括号（英文括号），计算规则符合 C 语言规范（例如 3/2=1，而 3.0/2=1.5）。

为达到此目的共需要完成以下步骤：

1. 为后续的计算进行一定数据结构上的准备，该步骤包括对 stack 数据结构的编写，助教提供的框架下支持用数组完成该数据结构的编写，同学们可以选择直接使用数组实现栈数据结构，也可以自己增设新变量用链表实现。

2. 判断表达式的合法性，该步骤包括测试括号匹配、运算符不连续出现两部分，该功能通过 calculator::judge() 函数实现。

3. 实现表达式求值，该功能通过 calculator::get_ans 函数实现，同学们可能需要类中的其他函数，也可能需要补充一定 private 类型变量（在 private 类型中定义的变量可以理解为全局变量，但在 calculator.h/cpp 之外不能使用）。各函数可能提供的功能见框架简介一项。

1.2 机器人吃金币 Robot（queue.h 及 walk.cpp）

在一个平面上有一个机器人（位于 (0, 0) 坐标）和 n 个金币（分别有各自的位置坐标和价值），机器人共有 m 次行动机会。编写程序实现一个机器人吃金币的最优策略（吃到的金币总价值最高）。

机器人的每回合的行动方式为：

1. 每次行动只能沿 x 轴或 y 轴（或其反方向）方向移动，每次移动可以移动 1 或 2 个单位。举例来说，从 (0, 0) 移动到 (3, 2) 需要 3 次行动，分别为向 x 轴移动 2 步、向 x 轴移动 1 步、向 y 轴移动 2 步。

2. 机器人吃金币需要在回合结束时准确落在金币所在的坐标处。举例来说，若机器人在某回合中从 (0, 0) 经过 (0, 1) 移动到 (0, 2)，那么它 不能 吃到位于 (0, 1) 的金币。

3. 拾取金币不消耗行动机会，同一个金币仅能获得一次（但同坐标可以有多枚金币，回合结束时可以全部获得）。

为达到此目的共需要完成以下步骤：

1. 为后续的计算进行一定数据结构上的准备，该步骤包括对 queue 数据结构的编写，助教提供的框架下支持用数组完成该数据结构的编写，同学们可以选择直接使用数组实现队列数据结构，也可以自己增设新变量用链表实现。

2. 实现机器人的最优策略计算，为了简便计算最后只需要输出最终总最大金币价值即可。同学们可能需要类中的其他函数，也可能需要补充一定 private 类型变量。各函数可能提供的功能见框架简介一项。

2. 评分标准

同学们需要完成

include/stack.h（1 分）

include/queue.h（1 分）

src/2_Expression/calculator.cpp（4 分）

src/3_Robot/walk.cpp（4 分）

中的所有 TODO 部分。检查时，需要各位同学运行 Expression 和 Robot 的可执行文件，注意运行时要将部分 main.cpp 中的代码中的注释符号删除以实现自主输入测试。

本次实验的 ddl 为 11.15（星期五）。

3. 框架简介

3.1 include/.

该文件夹内包含一个线性表（虚）类（list.h），一个栈类（stack.h），一个队列类（queue.h）和一个独立的无序集合类（unordered_set.h）。有经验的同学或许发现其内部的函数与 stl 库中的函数类似，这里单独写出是为了让同学们练习一下栈和队列内方法的实现。虚类的含义是这个类没有实际意义、也不能创造实例，但可以被继承。在《数据结构》课程中，线性表作为虚类可以被理解为是栈和队列的实际载体，而栈和队列是线性表的一种特殊举例。

首先介绍 unordered_set 类，这个类已经封装完成，只实现了最简单的方法，包括向集合中插入元素（insert）、查找集合中是否有某个元素（find)、清空集合（clear）以及检查集合是否为空（empty）。

接下来介绍 list.h 中定义的各个虚函数和方法，为方便大家理解，这里使用“该数据结构”代替“线性表”，因为以下描述中将 list 代换成 stack 或 queue 都是适用的。

empty()

这是一个返回 bool 值的方法，若该数据结构为空则返回 $true$，否则返回 $false$。

clear()

这是一个无返回值的方法，意思是将该数据结构清空。

push(data_base push_data)

这是一个无返回值的函数，意思是将 push_data 按该数据结构的规则放入该数据结构中（例如栈的规则是放在栈顶、队列的规则是放在队尾）。

pop()

这是一个无返回值的方法，意思是按该数据结构的规则弹出该数据结构中（例如栈的规则是弹出栈顶、队列的规则是弹出队首）。

top()

这是一个返回该数据结构规则下的第一个元素（例如栈的规则是返回栈顶、队列的规则是返回队首）。

3.2 src/1_Test/.

该文件下的 main.cpp 文件是一个样例，整个文件夹不需要同学们编辑，这个文件夹能在大家在处理 c++ 代码时提供一定参考，也可以测试 include/. 中的代码是否正确。

考虑到部分同学是初次接触 c++，该文件中的语句都做了详细的注释。

```cpp
#include <iostream>

// .h is okay here, please don't include the 'cpp's here
#include "stack.h"
#include "queue.h"

// Maybe add the following line to the program, it announces the namespace of the rest of the program. You may see another way in list.h
// using namespace std;

int main() {
    // Initialize a practice of a class, in this case a stack or a queue
    DATA_STRUCTURE::stack<int> p;

    // Push 0 to 9 into the structure
    int n = 10;
    for (int i = 0; i < n; i++)
        p.push(i);

    // Check if the structure is empty
    // Actually it has to be not empty because we have just push 10 items into the structure
    std::cout << p.empty() << std::endl << std::endl;

    while (!p.empty()) {
        // Print the items in the structure
        std::cout << p.top() << std::endl;

        // After printing the item, pop it out
```

```
28            p.pop();
29        }
30        return 0;
31    }
```

3.3 src/2_Expression/.

该文件包含一个类的声明与实现以及一个 main.cpp，其中 main.cpp 已经写好，实现了两个既定表达式以及一个持续输入的表达式的展示和计算。注意 calculator.cpp 文件内的各函数多少都有一些非常简单的返回值，这是为了能让整个项目通过编译，大家在完成实验时需要酌情删减。接下来主要介绍 calculator 类中各个函数的含义。

calculator(*) 与 ~calculator()

这是构造函数与析构函数，不重要也不关键。这里简单提一下构造函数，可以参照同文件下的 main.cpp 对比阅读。声明一个类时如果其附带的参数符合某个构造函数的构成，则按这个构造函数下的内容对类进行初始化，构造函数的初始化在整个类的初始化最后完成。

std::string get_expr()

这是一个返回表达式本身的方法，该函数已经写好，这个函数存在的意义是将某个私有变量作为返回值被返回，这些往往作为 debug 或者类间传输参数的接口，这里的意义是作出示范，方便各位同学学习如何依靠检查输出值的方法 debug。

bool legal()

这是 calculator 类中的核心函数，旨在判断 expr 表达式是否合法，非法的表达式可能出现的问题有括号不匹配、连续出现运算符（例如 3-*2）。

struct element get_ans()

这是 calculator 类中的核心函数，旨在计算表达式的值，需要用到下述私有函数。

struct element read_num()

这是一个私有函数，该函数的功能可以是读取字符串的某个数（可能多位、可能包含小数点）。

int priority(char c1, char c2)

这是一个私有函数，该函数的功能可以是检测 c1 和 c2 两个字符哪个字符的计算优先级高，不设计为 bool 类型是因为有可能两个字符优先级相等。

int priority_regular(char c)

这是一个私有函数，该函数的功能可以是返回一个常规运算符的运算优先级，是 priority 函数的辅助函数。

struct element operate(struct element element1, char c, struct element element2)

这是 calculator 类中的另一个核心函数，该函数的功能可以是计算 element1 (c) element2 的值，其中 element 是数、c 是计算符。

3.4 src/3_Robot/.

该文件包含一个类的声明与实现以及一个 main.cpp，其中 main.cpp 已经写好，实现了机器人最优策略的计算。注意 walk.cpp 文件内的各函数多少都有一些非常简单的返回值，这是为了能让整个项目通过编译，大家在完成实验时需要酌情删减。接下来主要介绍 walk 类中各个函数的含义。

walk(*) 与 ~walk()

构造和析构函数，与 calculator 类类似。

void print_para()

这是一个简单的测试方法，测试构造函数的正确性，可以与 calculator.h 中的构造函数对比阅读。

int compute_distance(int i, int x, int y)

这是一个私有函数，该函数的功能可以是计算第 i 号金币与坐标 (x, y) 的步数，返回值是从 (x, y) 到精确落到第 i 号金币位置共需要的步骤数。

int get_value()

这是 walk 类中的核心方法，旨在计算机器人能取得的最大金币价值，需要用到上述私有函数。

3.5 structures/.

这个文件夹内的文件不重要，其功能简而言之是编译”stack.h”，”queue.h” 两个类文件。请不要修改!!!