Basic Interpreter - 2022 动手指南

deadline: 第13周 周一 23：00

负责助教
@JerryGJX
@lillian039

1. 前言

BASIC 是一门解释性编程语言，本次大作业要求你用c++实现这个语言中的一些功能，来完成一个 Minimal 版本的 BASIC 解释器。

该解释器应具有的功能有：

- 立即解释某些语句并运行
- 执行特定控制语句
- 按照行数升序依次运行的大程序

关于评测：

我们进行基本的鲁棒性检测，但是不进行性能检测，你只需要通过下发的数据点即可。

注意：

由于已知没有 mac 用户，所以该教程完全基于此前提，如果有需要在 mac 上运行，请联系助教。

2. 下发文件解释

```
├── Basic
|   ├── Basic.cpp
|   ├── Utils
|   |   ├── error.cpp
|   |   ├── error.hpp
|   |   ├── strlib.cpp
|   |   ├── strlib.hpp
|   |   ├── tokenScanner.cpp
|   |   └── tokenScanner.hpp
|   ├── evalstate.cpp
|   ├── evalstate.hpp
|   ├── exp.cpp
|   ├── exp.hpp
|   ├── parser.cpp
|   ├── parser.hpp
```

```
|   ├── program.cpp
|   ├── program.hpp
|   ├── statement.cpp
|   └── statement.hpp
├── Basic-Demo-64bit
├── CMakeLists.txt
├── LICENSE
├── Minimal BASIC Interpreter - 2021.pdf
├── README.md
├── StanfordCPPLib
├── Test
└── score.cpp
```

2.1 你需要补充代码的部分

```
├── Basic
|   ├── Basic.cpp
|   ├── evalstate.cpp
|   ├── evalstate.hpp
|   ├── exp.cpp
|   ├── exp.hpp
|   ├── parser.cpp
|   ├── parser.hpp
|   ├── program.cpp
|   ├── program.hpp
|   ├── statement.cpp
|   └── statement.hpp
```

2.2 你需要学习使用的部分

```
├── Basic
|   ├── Utils
|   |   ├── error.cpp
|   |   ├── error.hpp
|   |   ├── strlib.cpp
|   |   ├── strlib.hpp
|   |   ├── tokenScanner.cpp
|   |   └── tokenScanner.hpp
```

注意：

这部分文件正常情况下你**不应修改**，如确信这部分文件有问题，请找出问题后向助教反馈。

学习方法：

`strlib.hpp` 和 `tokenScanner.hpp` 源于 `StanfordCPPLib`，使用方法与 `StanfordCPPLib` 中对应文件相同，请通过阅读文件中的注释，以及查阅官方文档 The StanfordCPPLib package 自行学习。

`error.hpp` 的使用请参考 C++ 异常处理 - 菜鸟教程。请务必熟悉代码中的异常处理方法，包括助教给出的代码。

声明：

`strlib.hpp` 和 `tokenScanner.hpp` 是助教在 `StanfordCPPLib` 基础上进行过修改的版本，基本排除了由于编译器版本导致的问题，同时也基本修复了原代码的 `memory leak` 问题。

2.3 本地评测方法

```
├── Basic-Demo-64bit
├── CMakeLists.txt
├── Test
└── score.cpp
```

评测数据：

`Test` 文件中的 100 个数据点，这些文件和oj中测评的数据点是相同的。

评测原理：

与标程对拍 （指进行相同输入看输出是否相同）

标程：

`Basic-Demo-64bit` 这是使用已有的正确代码编译出的一个可执行文件

评测要求：

你的可执行文件路径与名字为`cmake-build-debug/code` ，也就是CLion默认的产生可执行文件的位置。

若路径有问题或你有独特的需求，你可以自行修改score.cpp中的路径或者测评代码来满足你的需求。

评测脚本使用方法：

编译 `score.cpp`，然后运行编译产生的文件，即可。

e.g 命令行中输入：

```
g++ -o score score.cpp
```

```
./score -f
```

评测结果：

```
Trace "Test/trace85.txt" ... Pass
Trace "Test/trace86.txt" ... Pass
Trace "Test/trace87.txt" ... Pass
Trace "Test/trace88.txt" ... Pass
Trace "Test/trace89.txt" ... Pass
Trace "Test/trace90.txt" ... Pass
Trace "Test/trace91.txt" ... Pass
Trace "Test/trace92.txt" ... Pass
Trace "Test/trace93.txt" ... Pass
Trace "Test/trace94.txt" ... Pass
Trace "Test/trace95.txt" ... Pass
Trace "Test/trace96.txt" ... Pass
Trace "Test/trace97.txt" ... Pass
Trace "Test/trace98.txt" ... Pass
Trace "Test/trace99.txt" ... Pass
100 / 100 trace(s) passed.
Final Score: 10.0
```

2.4 解释器实现步骤及待实现文件介绍

注意：这是助教实现上的设计建议，你可以完全抛开现有框架自己实现，只需在 CR 时与助教说明即可。

- **读入指令并切片**

目的： 将指令按照一定规律划分，以便于使用。

实现方法： 学习使用 `tokenscanner` 来帮助你完成这一步。

实现位置： `Basic.cpp`

- **Lexer 词法检查**

目的： 检查获得的切片是否满足类型要求。

实现方法： 学习使用 `tokenscanner` 来帮助你完成这一步。

实现位置： `Basic.cpp`

e.g.

```
111         --> a correct integer
1a1         --> an incorrect integer
```

- **Parser 句法检查**

目的： 检查获得的指令是否满足指令的格式要求。

实现方法： 借助词法检查，以及切片类型进行检查。

实现位置： `Basic.cpp`

e.g.

```
END         --> a correct END command
END Hahaha  --> an incorrect END command
```

- **指令存储**

目的： 用于实现大程序中的跳转指令，指令打印等等。

实现方法： 使用数据结构，如 map， set 等，请自行设计。

实现位置： `program.cpp, program.hpp`

- **指令运行**

目的： 执行指令

实现方法： 完成 `statment.hpp` 中的类的设计和实现，在 `Basic.cpp` 中通过调用 `statment` 及其子类的 execute 函数实现。

实现位置： `statment.cpp, statment.hpp`

2.5 关于 OJ 评测

本次大作业**使用github进行测评。**提交时，你的github仓库下需要有一个Basic文件夹，一个提供的StanfordCPPLib文件夹，与一个提供的CMakeLists.txt。

在oj的提交栏提交你的github仓库的地址，如 `https://github.com/your_user_name/your_repo_name`

如果**Compile Error**并且报错类似下面的样子，

`Cloning into 'your_repo_name'... fatal: unable to access 'https://github.com/your_user_name/your_repo_name/': Failed to connect to github.com port 443: Connection timed out`

或

`Cloning into 'your_repo_name'... fatal: unable to access 'https://github.com/your_user_name/your_repo_name/': GnuTLS recv error (-110): The TLS connection was non-properly terminated.`

可能是因为github访问速度较慢。你只需要将**提交的地址改为** `https://github.com.cnpmjs.org/your_user_name/your_repo_name` 形式的地址就可以进行镜像加速从而解决这个问题了。

测评文件内容默认是Test文件夹下提供的文件。这些文件和oj中测评的数据点是相同的。在这些测试点下，你的程序需要输出与标程相同的结果。

2.6 关于 code review

- 请务必保证 code review 时你的最新版本代码不仅仅保存在 ACM OJ 上，否则将酌情扣分。
- 请务必保证你完全清楚你在写什么，且严禁抄袭。

2.7 关于答疑

由于该作业本身实现难度较低，同时该作业的主要目的是 锻炼各位派生类的使用，文档的阅读能力同时让各位初步了解解释器的设计，所以答疑原则上以对于文档的理解为主，实现方面的细节（如派生类的用法，指针的用法等）请自己复习/学习。（当然如果实在debug搞不定助教们也会帮忙的~~）

2.8 Utils 修改日志

- 2022/11/13 修复 `stream >> std::ws` 在不同版本的编译器中产生的不同操作现象。
- 2022/11/13 修复 `tokenscanner` 自带内存泄露的问题。