CS235101
Data Structures
Homework 1

2016/10/18 10:00am

~

2016/11/01 10:00am

(Hard deadline)

Target

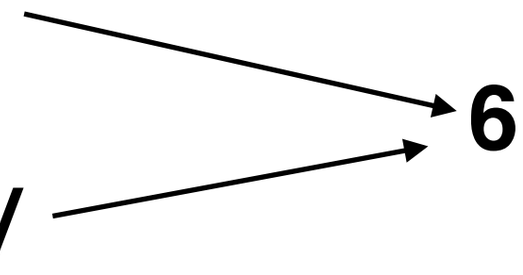
- The target of the homework is to evaluate an infix expression.
  - Operands range from 0 to 9.
  - The set of operators contains **( ) + - * /**
    - /: integer division, e.g. 5/2=2
  - No space character in-between.
- e.g. 1+(7*1+2)*3

Target

- To implement the evaluation, you are asked to implement 4 functions below.
  - The conversion from infix to prefix.
  - The conversion from infix to postfix.
  - The evaluation of prefix.
  - The evaluation of postfix.

Target

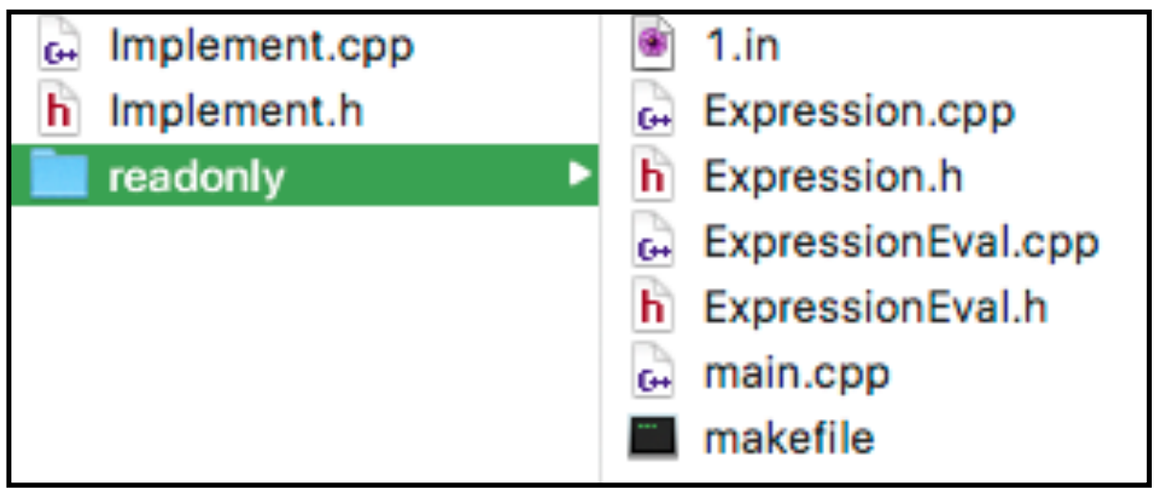
- There is no parenthesis in prefix and postfix expressions.
  - prefix: **(1+2)*6/3** -> **/*+1263**
  - postfix: **(1+2)*6/3** -> **12+6*3/**



- For more details of the algorithms, please refer to the slides, **2016(Fall)_DS - 04. Stacks & Queues**.

File Structure

hw1/



- **readonly**
  - makefile
  - 1.in is a released test file.
  - main.cpp contains a testing function.
  - **class** Expression represents the DS of expression.
  - **class** ExpressionEval specifies the functions to be implemented.
- **class** Implement : contains your implementation.

Expression

```
//The data structure that stores the expression.
class Expression
{
public:
    //the size of the expression
    unsigned sz;

    //char array that stores the expression
    char *data;

    //constructors and destructor
    // ~

    //assignment operator with respect to a cstring
    // all chars are copied to data except the end of cstring, \0.
    Expression& operator=( const char *e );

    //comparison operator between two expressions
    bool operator==( const Expression &e );

    //erase data
    void clear();

    //resize the space of data
    void resize(unsigned sz);

    //print the expression
    friend std::ostream& operator<<( std::ostream &os, const Expression &e );
};
```

ExpressionEval

```
//Expression Evaluation
//It contains the functions that you have to override in implement.h/.cpp.
class ExpressionEval
{
public:
    //-----------------------------------------------------------------
    // convert the input infix expression to prefix one
    //-----------------------------------------------------------------
    virtual void infix2Prefix(Expression &prefix, const Expression &infix);

    //-----------------------------------------------------------------
    // convert the input infix expression to postfix one
    //-----------------------------------------------------------------
    virtual void infix2Postfix(Expression &postfix, const Expression &infix);

    //-----------------------------------------------------------------
    // evaluate the input prefix expression
    //-----------------------------------------------------------------
    virtual int evalPrefix(const Expression &prefix);

    //-----------------------------------------------------------------
    // evaluate the input postfix expression
    //-----------------------------------------------------------------
    virtual int evalPostfix(const Expression &postfix);
};
```

Implement

Implement.h

```
class Implement : public ExpressionEval
{
public:

    // add your code here
    //----------------------

    //....

    //----------------------
};
```

Implement.cpp

```
#include "Implement.h"

// add your code here
//----------------------

//...

//----------------------
```

1.in

```
7*3-5¬
-*735¬
73*5-¬
16¬
```

// infix
// prefix
// postfix
// value

```
(1+2)*3*(2+3/1)-
**+123+2/31¬
12+3*231/+*¬
45¬
```

// infix
// prefix
// postfix
// value

```
5+2*1+(4-5)¬
++5*21-45¬
521*+45-+¬
6¬
```

// infix
// prefix
// postfix
// value

```
1+(7*1+2)*3¬
+1*+*7123¬
171*2+3*+¬
28
```

// infix
// prefix
// postfix
// value

Evaluation

```
// test data
Expression infix = infixStr.c_str();
Expression prefix = prefixStr.c_str();
Expression postfix = postfixStr.c_str();
int value = std::atoi( valueStr.c_str() );

// generate your results
Expression urprefix, urpostfix;
inst.infix2Prefix( urprefix, infix );
inst.infix2Postfix( urpostfix, infix );
int valuePre  = inst.evalPrefix( prefix );
int valuePost = inst.evalPostfix( postfix );

// compare results
bool prefixPass = prefix==urprefix;
bool postfixPass = postfix==urpostfix;
bool valuePass = value==valuePre && value==valuePost;

if( !prefixPass )
    throw "[Wrong Prefix]";

if( !postfixPass )
    throw "[Wrong Postfix]";

if( !valuePass )
    throw "[Wrong Evaluation]";
```

Messages

- **[Undefined ExpressionEval::*******]**
  - The function is not implemented.
- **[Wrong Prefix], [Wrong Postfix], [Wrong Evaluation]**
  - Wrong Answer
- **[Accepted]**

STL is not allowed.

- <string> <stack> <queue> are not allowed.
- If you try to include the above headers, your source files **WILL NOT** be compiled properly during TA's evaluation.

Submission

1. **make clean**

   - remove object files (*.o) and the executable

2. Archive your source codes (whole hw1 folder) into a zip file named **[studentID]_hw1.zip**

   - e.g. 104062999_hw1.zip

3. Submit the zip file to ilms system BEFORE the deadline.