과목: 자료구조

교수: 최재현 교수님

자료구조 <프로젝트>

- Calculator 클래스 작성 -

홍지훈

이름: 홍지훈

학과: 소프트웨어학부

분반: 나

학번 : 20201777

0. 과제

수식을 후위표기식으로 변경하고, 계산하는 Calculator 클래스 작성

-- 과제3 에서 작성한 List 클래스를 template 를 추가하여 변경

-- 과제4 에서 작성한 String 클래스를 사용

클래스의 기본 정의는 아래와 같음

class Calculator

{

   private :

      List<String> tokens;

      int errCode;                  // 발생된 오류코드 값 : 0 -> 오류 없음 , 다른 값 -> 오류 있음

      int value;                     // 계산된 값

      String postfix:              // 후위표기식 = 최초 공백으로 초기화

      int makePostFix();         // postfix 로 변경하는 함수 :

                                       // 오류 없는경우, 0, 오류가 있는 경우, 1을 반환

                                       // 변경결과는 postfix 변수에 저장, 오류시 적절한 코드를 errCode 에 저장 (오류코드는 각자가 정의)

      int evaluation();            // postfix 를 계산하는 함수 :

                                       // 계산된 값을 구함, 오류 없는경우 0, 오류가 있는 경우, 1을 반환

                                       // 계산된 값은 value 에 저장, 오류시 적절한 코드를 errCode 에 저장 (오류코드는 각자가 정의)

  public :

      int getErrorCode();                             // 오류코드 반환

      int setExpression(const char\* expr);       // expr에 전달된 수식(중위표기식)을 postfix로 변경하고 계산하는 함수

                                                            //    오류 없는경우, 0, 오류가 있는 경우, 1을 반환

       String getPostFix();                            // 변환된 후위표기식을 반환   --> 오류가 있을경우 최초값인 공백이 리턴

       int getValue();                                   // 수식 오류있음 --> 예외발생

                                                             // 수식 오류없음 --> 결과값 리턴

};

위에 기본정의된 함수외에 각자 필요하다고 생각하다고 하는 함수는 얼마든지 추가하여 구현가능

<< setExpression 예시 >>

   if ( !makePostfix )

        return evaluation();

   else

        return 1;

<< main 함수 예시 >>

char expr[1000];

Calculator c;

cout << "수식을 입력하시오 : ";

cin.getline(expr, 1000);

if ( !c.setExpression(expr) )

{

    String posfix = c.getPostFix();

    try {

       cout << "후위표기식 : " << postfix  << "결과값 : " << c.getValue() << endl;

    } catch (const char\* errmsg)

    {

          cout << errmsg << endl;

     }

}

※ List<String> 은 expression 을 postfix 로 변경하기 expession 전달된 문자열을 토큰리스트로 변경하는 데 사용

    즉, 괄호, 연산자, 피연산자를 구분해서 하나의 단위로 저장

    (토큰 : 공백 또는 의미있는 단위로 잘려진 문자열 )

   예) (3+5)/10 을 List<String>으로 변경 :    tokens[0] = "("

                                                            tokens[1] = "3"

                                                            tokens[2] = "+"

                                                            tokens[3] = "5" ......

※ Postfix 로 변경할때 List<String> 에 저장된 토큰들을 가지고, 강의내용에서 설명한 알고리즘을 적용하여 변경

1. 소스코드

1-1. Stack.h

#pragma once

#include <iostream>

using namespace std;

//cpp파일을 따로 나누면 오류가 생겨서 헤더파일에 합쳤습니다.

template<class T>

class Stack

{

private:

    T\* stack;

    int top;

    int capacity;

public:

    Stack(int stackCapacity = 10);

    bool IsEmpty() const;

    T& Top() const;

    void Push(const T& item);

    void Pop();

    void print();

};

template <class T>

void ChangeSize1D(T\*& a, const int oldSize, const int newSize)

{

    if (newSize < 0) throw "New length must be >= 0";

    T\* temp = new T[newSize];

    int number = min(oldSize, newSize);

    copy(a, a + number, temp);

    delete[] a;

    a = temp;

}

template<class T>

Stack<T>::Stack(int stackCapacity) : capacity(stackCapacity)

{

    if (capacity < 1) throw "Stack capacity must be > 0";

    stack = new T[capacity];

    top = -1;

}

template<class T>

inline bool Stack<T>::IsEmpty() const { return top == -1; }

template<class T>

inline T& Stack<T>::Top() const

{

    if (IsEmpty()) throw "Stack is empty";

    return stack[top];

}

template<class T>

void Stack<T>::Push(const T& x)

{

    if (top == capacity - 1)

    {

        ChangeSize1D(stack, capacity, 2 \* capacity);

        capacity \*= 2;

    }

    stack[++top] = x;

}

template<class T>

void Stack<T>::Pop()

{

    if (IsEmpty()) throw "Stack is empty. Cannot delete.";

    stack[top--].~T();

}

template<class T>

void Stack<T>::print()

{

    for (int i = 0; i <= top; i++) {

        cout << stack[i] << " ";

        if (i == top)

            continue;

        cout << "-> ";

    }

    cout << endl;

}

1-2. Queue.h

#pragma once

#include <iostream>

using namespace std;

//cpp파일을 따로 나누면 오류가 생겨서 헤더파일에 합쳤습니다.

template<class T>

class Queue

{

private:

    T\* queue;

    int front;

    int rear;

    int capacity;

public:

    Queue(int queueCapacity = 10);

    bool IsEmpty() const;

    T& Front() const;

    T& Rear() const;

    void Push(const T& item);

    void Pop();

    void print();

};

template <class T>

Queue<T>::Queue(int queueCapacity) : capacity(queueCapacity)

{

    if (capacity < 1) throw "Queue capacity must be > 0";

    queue = new T[capacity];

    front = rear = 0;

}

template <class T>

inline bool Queue<T>::IsEmpty() const { return front == rear; }

template <class T>

inline T& Queue<T>::Front() const

{

    if (IsEmpty()) throw "Queue is empty. No front element";

    return queue[(front + 1) % capacity];

}

template <class T>

inline T& Queue<T>::Rear() const

{

    if (IsEmpty()) throw "Queue is empty. No rear element";

    return queue[rear];

}

template <class T>

void Queue<T>::Push(const T& x)

{

    if ((rear + 1) % capacity == front)

    {

        T\* newQueue = new T[2 \* capacity];

        int start = (front + 1) % capacity;

        if (start < 2)

            copy(queue + start, queue + start + capacity - 1, newQueue);

        else

        {

            copy(queue + start, queue + start + capacity - 1, newQueue);

            copy(queue, queue + rear + 1, newQueue + capacity - start);

        }

        front = 2 \* capacity - 1;

        rear = capacity - 2;

        capacity \*= 2;

        delete[] queue;

        queue = newQueue;

    }

    rear = (rear + 1) % capacity;

    queue[rear] = x;

}

template <class T>

void Queue<T>::Pop()

{

    if (IsEmpty()) throw "Queue is empty. Cannot delete.";

    front = (front + 1) % capacity;

    queue[front].~T();

}

template <class T>

void Queue<T>::print()

{

    for (int i = front+1; i <= rear; i++)

    {

        cout << queue[i] << " ";

        if (i == rear)

            continue;

        cout << "-> ";

    }

    cout << endl;

}

1-3. Main.cpp

#include "Stack.h"

#include "Queue.h"

using namespace std;

int main(void) {

    //Stack

    cout << "Stack" << endl;

    Stack<int> s1;

    cout << "s1 is Empty? => " << s1.IsEmpty() << endl;

    s1.Push(5);

    s1.Push(10);

    s1.Push(15);

    s1.print();

    cout << "s1 Top => " << s1.Top() << endl;

    s1.Pop();

    s1.print();

    cout << "s1 Top => " << s1.Top() << endl;

    //Queue

    cout << endl << "Queue" << endl;

    Queue<int> q1;

    cout << "q1 is Empty? => " << q1.IsEmpty() << endl;

    q1.Push(5);

    q1.Push(10);

    q1.Push(15);

    q1.print();

    cout << "q1 Front => " << q1.Front() << endl;

    cout << "q1 Rear => " << q1.Rear() << endl;

    q1.Pop();

    q1.print();

    cout << "q1 Front => " << q1.Front() << endl;

    cout << "q1 Rear => " << q1.Rear() << endl;

    return 0;

}

2. 실행 화면

