개요 코딩 결과

과제 110 - Exception - Stack

제출 마감일: 2023-06-18 23:59
업로드 가능한 파일 수: 3
제출 방식: 개인

과제 110 은 keylog 분석 및 jplag 를 이용한 표절 검사가 상시 수행됩니다.

- 키로그 파일(p111.csv)에 소스 코드 이름이 제출 파일명과 일치하지 않으면 로그 점수가 계산되지 않습니다.
- 키로그 파일(p111.csv)에 키보드 입력 정보가 없는 경우 로그 점수가 계산되지 않습니다. (로그가 IdeState 나 Action 타입 밖에 없는 경우가 자주 있습니다.)
- 구현이 작은 소스 코드는 주어진 코드를 복사하지 말고 직접 타이핑 하는 것을 권장 드립니다.
- 계속 잘 안될 때에는 초심으로 돌아가 activitytracker 를 clear 하고, 소스 파일명을 제출 파일명과 일치 시킨 후 직접 키보드로 코딩하는 것을 추천 드립니다.

문제

주어진 Stack 템플릿 클래스에 예외 처리를 추가하시오

예외 발생 타입은 StackException 클래스를 사용하시오

<제약조건>

- 실습 113 을 먼저 해결해야 합니다.
- Stack의 push, pop 멤버 함수에서 적절한 예외를 던지도록 구현하시오
- Stack의 복사생성자와 복사 대입 연산자는 삭제되어 사용할 수 없음
- Stack의 소멸자는 반드시 noexcept 표시를 하시오
- Stack의 생성자와 소멸자에서 RAII 를 준수하도록 하시오
- StackException 은 std::runtime_error 클래스를 상속받도록 하시오
- StackException의 소멸자는 noexcept 표시를 하시오
- main() 함수에서는 StackException 예외를 처리하는 핸들러를 정의하시오
- Stack, StackException 클래스의 멤버 함수를 모두 구현하시오
- 주어진 헤더 파일의 클래스 선언을 준수하여 멤버 함수를 구현하시오
- Stack.h 에서는 throw 만 하시오
- 소스코드 명이나 라인 넘버를 출력하고 싶을 때는, gcc 컴파일러의 "The standard predefined macros" 를 참고하세요

https://gcc.gnu.org/onlinedocs/cpp/Standard-Predefined-Macros.html

Stack.h ----------

```
#ifndef EXCEPTION_STACK_H
#define EXCEPTION_STACK_H

#include "StackException.h"
#include <iostream>
template<typename T>
class Stack {
public:
        Stack(int sz) ;                 //객체 초기화 및 필요한 자원을 획득하시오
        ~Stack() noexcept ;             //사용한 자원을 해제하시오
        // 복사 생성자와 복사 대입 연산자를 삭제하시오
        void push(T c) { s[top++] = c ; }    //예외 발생 시 StackException 던지시오
        T pop() { T r = s[--top]; return r ; } //예외 발생 시 StackException 던지시오
        void print() const ;     //스택의 모든 원소를 '\n' 으로 구분하여 출력하시오
private:
    int size = 0;   int top = 0;
    T* s = nullptr;
} ;
#endif //EXCEPTION_STACK_H
```

StackException.h ----------

```
#ifndef EXCEPTION_STACKEXCEPTION_H
#define EXCEPTION_STACKEXCEPTION_H
#include <stdexcept>
#include <string>
#include <sstream>
class StackException : public std::runtime_error {
public:
   StackException(const std::string& msg, const char *file, size_t line)
         : std::runtime_error(msg), _msg{msg},  _file_name(file), _line(line) {
      std::ostringstream oss;
```

```
        oss  << file << ":" << line << ":" << "Exception: " << msg << std::endl;
        _msg = oss.str();
    }
    ~StackException() noexcept {}
    const char* what() const noexcept {
        return _msg.c_str();
    }

private:
    std::string _msg;
    std::string _file_name;
    size_t _line = 0;
};
#endif //EXCEPTION_STACKEXCEPTION_H
```

입출력

```
// main() 함수에서 StackException 예외를 처리하는 핸들러를 정의해서 예외 메시지를 출력해 보세요
//case 1 (입력 1일 때 실행되는 코드 예)
int main() {
    Stack<char> stack(1);
    stack.push('a');
    stack.pop(); stack.pop();
    stack.print();
}
==> 출력 예: main.cpp:[라인번호]:Exception: Stack is empty

//case 2
int main() {
    Stack<char> stack (1);
    stack.push('a');
    stack.push('b');
```

```
    stack.print();
}
```

==> 출력 예: main.cpp:[라인번호]:Exception: Stack is full!

```
//case 3
int main() {
    Stack<int> stack(1);
    stack.push(1);    stack.pop();
    stack.push(2);
    stack.print();     // 2 출력됨
    stack.pop();
}
```

==> 출력 예: 2

```
//case 4
int main() {
    Stack<int> stack(2);
    stack.push(1); stack.push(2);
    stack.pop();   stack.pop();
    stack.push(3);
    stack.print();   // 3 출력됨
}
```

==> 출력 예: 3

제출파일

Stack.h

StackException.h

p111.csv