

ICE2004 자료구조론 과제2

제 목

자료구조 H/W

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2021년 10월 23 일

학부 정보통신공학과

학년 2

성명 김민겸

학번 12201863



1. **개요**

주어진 ArrayStack 코드를 바탕으로, 아래의 함수들이 정상 동작함을 보이고 Stack에서 push, pop을 진행할 때 StackFullException과 StackEmptyException 클래스를 이용해 예외처리를 구현하는 것이 과제였다.

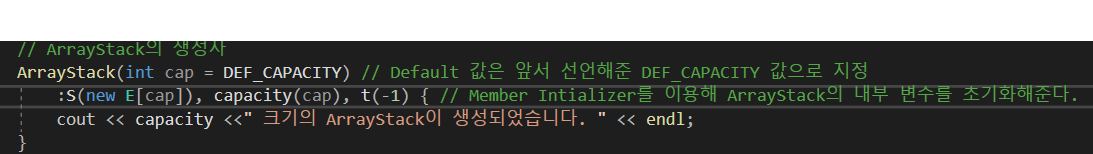
* Size() : 스택이 현재 가지고 있는 요소의 개수를 반환한다.
* Push() : 스택에 새로운 요소를 넣는다.
* Pop() : 스택의 가장 마지막 요소를 뺀다.
* Top() : 스택의 가장 마지막 요소의 값을 반환한다.

이 함수들 중 Push는 스택이 가득 찼을 때 실행되는 경우 StackFullException을 이용해 더 이상 스택 내부에 요소가 들어갈 수 없음을 알려준다. 그리고 Top, Pop 함수가 스택이 텅 비어있을 때 실행되는 경우 StackEmptyException을 이용해 반환할 스택 내의 마지막 요소가 없음을 알려준다.

이번 과제에서는 Exception Handling 부분에서 많은 오류를 마주쳤다. 익숙하지 않은 파트이다 보니 예외처리 클래스를 사용하는 것에서 시행착오를 많이 겪었다. 이 부분을 위주로 코드 분석과 해결 과정을 기록하려 한다. **2. 구현상 특징**

1. **생성과 소멸**

기본적인 ArrayStack을 구현하는 코드이다. 클래스의 생성자부터 보면,



Member Initializer를 이용해서 내부 멤버 변수를 초기화한다. ArrayStack의 클래스 멤버 변수는 다음과 같다. 이때 포인터 자료형인 S는 new 연산자를 이용해 초기화한다.

∙ E 포인터 자료형 변수 S : 스택에 값을 담을 공간, 하나의 컨테이너를 의미한다.

∙ int 자료형 capacity : 스택의 총 용량을 의미한다.

∙ int 자료형 t : 스택에 쌓여 있는 데이터의 개수를 의미한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

소멸자에는 클래스 객체가 소멸되었음을 알려주고, 동적할당한 S 변수를 해제해준다.

1. **스택 함수 기능 구현**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

∙ size() : 현재 스택의 개수를 나타내는 t에서 1을 더하여 (-1부터 시작하므로) 반환한다.

∙ empty() : 현재 스택이 비어있는 상태인지 확인하는 함수, bool타입으로 현자 위치가 0보다 작은 지 여부를 반환한다.

∙ full() : 현재 스택이 가득 차 있는 상태인지 확인하는 함수, bool 타입으로 현재 용량을 확인하는 size() 함수를 호출해 반환된 값이 capacity와 동일한지 여부를 반환한다.

∙ top() : 자료형에도, 함수 선언에도 const 키워드를 붙여서 변경 여지를 없앤다. if문을 이용해 empty() 상태이면 스택 요소를 반환하지 못하므로 StackEmptyException 클래스를 throw한다. Empty()상태가 아니라면 S 스택 컨테이너의 t번째 요소를 반환한다.

∙ push() : E 자료형 값을 매개변수로 받아 1을 증가시킨 t 인덱스의 S 컨테이너에 e 매개변수를 대입한다. 이때 스택이 가득 찬 상태였을 때 예외처리를 하기 위해 if문에서 full()함수를 호출해, StackFullException 클래스를 throw한다.

∙ pop() : S 컨테이너에서 제일 상위에 있는 값을 꺼내는 함수이다. 현재 위치의 인덱스를 1 감소시키는 것으로 이를 구현한다. 이때도 역시 스택이 텅 비어있을 경우 pop을 수행하지 못하므로 예외처리를 한다. Empty()함수가 true인 경우에 StackEmptyExcpetion 클래스를 throw한다.

1. **RuntimeException.h 구현**

**텍스트이(가) 표시된 사진

자동 생성된 설명** **텍스트이(가) 표시된 사진

자동 생성된 설명**

String 자료형의 에러 메시지 변수를 멤버 변수로 하는 RuntimeException 클래스를 다음과 같이 구현하였다. 생성자는 string 매개변수를 받아 errorMsg 멤버변수를 초기화하도록 하였고, getMessage() 에서 멤버변수를 반환하는 함수를 구현했다.

이를 public 상속받는 StackEmptyException, StackFullException 클래스를 만들었으며 모두 MemerInitializer를 이용해 상위 클래스인 RuntimeException의 생성자를 호출하였다. 이 두개의 클래스는 상황마다 다른 error Message를 받아서 콘솔에 출력하는 역할을 수행한다.

1. **Try catch문을 이용한 main함수에서 예외 처리 구현**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 try catch문을 두 가지 케이스(StackEmptyException, StackFullException이 호출되는 경우)로 나누어서 작성하였다.

첫 번째 케이스에서는 스택이 가득 찬 상태에서 push를 진행해서 StackFullException이 일어나도록 하였다. 이 오류에 의해서 push 함수 내부의 if문이 실행되고, StackFullException클래스의 객체가 생성된다.

두 번째 케이스에서는 스택이 텅 빈 상태에서 pop을 진행했을 때 StackEmptyException이 일어나는 경우이다. 이때 pop 함수 내부에서 if문이 실행되어 StackEmptyException 클래스 객체가 생성된다.

1. **ostream 연산자 오버로딩**

****

RuntimeException 객체의 errorMsg를 출력하는 역할을 하는 <<연산자 오버로딩이다. RuntimeException 객체가 피연산자로 왔을 때, 객체의 getMessage 함수를 호출해 멤버변수 errorMsg를 출력해준다.

**3.실행화면 캡쳐 이미지**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

1. **고찰**

Throw() Specifier이라는 개념에 대해 새롭게 알게 되었다.

텍스트, 시계, 벽, 탑재이(가) 표시된 사진

자동 생성된 설명

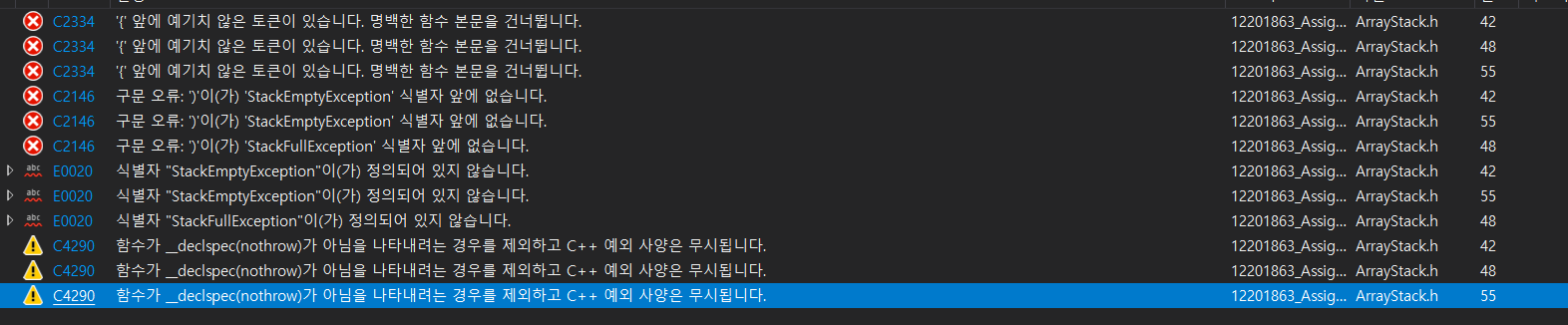
텍스트, 벽, 녹색, 탑재이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

다음 세 가지 함수를 보면, 함수의 이름 옆에 **throw ([예외처리 클래스 이름])** 다음과 같은 문구가 있는 것을 알 수 있다. 이는 ‘throw 한정라’라 하며, 함수가 정의 부분에서 예외로 던질 수 있는 typeid 즉 자료형을 인자로 받는다. 이떄 typeid가 클래스나 구조체라면 상속받는 자식 역시 예외로 던질 수 있는 타입으로 지정하는 것이 특징이다.

또한 RuntimeException 클래스의 정의 부분이, 이 예외처리 클래스를 사용하는 함수의 뒤에 있다면 컴파일 오류가 발생하는 것을 알게 되었다.



Throw 한정자의 개념을 알기 전에는 선언에 붙어있는 throw 구문 탓에 오류가 이러한 오류가 발생한다고 생각하였다. 그러나, RuntimeException 클래스와 그 자식들을 다른 헤더 파일에 넣고 ArrayStack.h의 가장 상단에 include 하였더니 컴파일 시점에서 예외처리 클래스가 모두 생성되었다. 그리고 그 이후에 push, pop, top 함수에서 이를 사용할 수 있게 되어 정상적으로 컴파일이 되는 것을 알 수 있었다.

이에 더하여, template를 사용했을 때 선언과 정의의 분리 작업 과정에서도 어려움을 겪었다. 과제에서 주어진 헤더 파일에서는 선언과 정의를 분리했지만, 분리한 상태에서는 컴파일이 잘 되지 않았다. 이 원인은, 헤더파일에 template를 선언만 하게 된다면 컴파일 과정에서 template가 어떤 자료형인지 컴파일러가 알 수 없게 되기 때문이다. 따라서 template를 사용할 시점에는 선언과 동시에 정의해주어야 컴파일이 정상적으로 진행되고, 에러가 뜨지 않는 것이다.