



C++ - 모듈 09

STL

요약: 이 문서
에는 C++ 모듈의 모듈 09 연습이 포함되어 있습니다.

버전: 1.5

내용물

나	소개	2
II	일반 규칙	삼
III	모듈별 규칙	5
IV 연습 00: 비트코인 거래소		6
V 연습 01: 역 폴란드 표기법		8
VI 연습 02: PmergeMe		10
VII 제출 및 동료 평가		13

제1장

소개

C++는 Bjarne Stroustrup이 C 프로그래밍 언어 또는 "C with Classes"(출처: [Wikipedia](#))의 확장으로 만든 범용 프로그래밍 언어입니다.

이 모듈의 목표는 객체 지향 프로그래밍을 소개하는 것입니다. 이것이 C++ 여정의 출발점이 될 것입니다. OOP를 배우려면 많은 언어를 권장합니다. 우리는 오랜 친구인 C에서 파생되었기 때문에 C++를 선택하기로 결정했습니다. 이는 복잡한 언어이고 작업을 단순하게 유지하기 위해 코드는 C++98 표준을 준수합니다.

우리는 현대 C++가 여러 측면에서 많이 다르다는 것을 알고 있습니다. 따라서 능숙한 C++ 개발자가 되고 싶다면 42 Common Core 이후에 더 나아가는 것은 여러분에게 달려 있습니다!

제2장

일반 규칙

컴파일 중

- C++ 및 -Wall -Wextra -Werror 플래그를 사용하여 코드를 컴파일합니다.
- -std=c++98 플래그를 추가해도 코드는 여전히 컴파일되어야 합니다.

형식 및 명명 규칙

- 연습 디렉토리의 이름은 다음과 같이 지정됩니다: ex00, ex01, ... , 엑스
- 필요에 따라 파일, 클래스, 함수, 멤버 함수 및 속성의 이름을 지정합니다.
지침.
- 클래스 이름은 UpperCamelCase 형식으로 작성합니다. 클래스 코드가 포함된 파일은
항상 클래스 이름에 따라 이름이 지정됩니다. 예: ClassName.hpp/
ClassName.h, ClassName.cpp 또는 ClassName.hpp. 그런 다음 벽돌 벽을 나타내는 "BrickWall"
클래스 정의가 포함된 헤더 파일이 있는 경우 해당 이름은 BrickWall.hpp가 됩니다.
- 달리 지정하지 않는 한 모든 출력 메시지는 새 줄로 끝나야 합니다.
문자로 표시되어 표준 출력에 표시됩니다.
- 안녕 Norminette! C++ 모듈에는 코딩 스타일이 적용되지 않습니다. 당신이 좋아하는 것을
따라갈 수 있습니다. 하지만 동료 평가자가 이해할 수 없는 코드는 채점할 수 없는 코드라는
점을 명심하세요. 깨끗하고 읽기 쉬운 코드를 작성하기 위해 최선을 다하십시오.

허용/금지

더 이상 C로 코딩하지 않습니다. C++를 배울 시간입니다! 그러므로:

- 표준 라이브러리의 거의 모든 것을 사용할 수 있습니다. 따라서 이미 알고 있는 내용을 고수하는 대신 익숙
한 C 함수의 C++ 버전을 최대한 많이 사용하는 것이 현명한 것입니다.
- 단, 다른 외부 라이브러리는 사용할 수 없습니다. 이는 C++11(및 파생 형식) 및 Boost 라이브러리가 금지됨
을 의미합니다. 다음 함수도 금지됩니다: *printf(), *alloc() 및 free(). 사용하면 성적이 0이 되고 그게 전
부입니다.

- 달리 명시적으로 명시하지 않는 한 사용 네임스페이스 <ns_name> 및 친구 키워드는 금지되어 있습니다. 그렇지 않으면 성적은 -42가 됩니다.
- STL은 모듈 08과 09에서만 사용할 수 있습니다. 즉, 그때까지는 컨테이너 (벡터/목록/맵 등)와 알고리즘 (<algorithm> 헤더를 포함해야 하는 모든 것)이 없다는 의미입니다. 그렇지 않으면 성적은 -42가 됩니다.

몇 가지 디자인 요구 사항

- C++에서도 메모리 누수가 발생합니다. 메모리를 할당할 때(새 키워드), 메모리 누수를 피해야 합니다.
- 모듈 02부터 모듈 09까지 수업은 정통 방식으로 설계되어야 합니다. 달리 명시적으로 언급된 경우를 제외하고 표준 형식.
- 헤더 파일에 있는 모든 함수 구현(함수 템플릿 제외)은 실습에서 0을 의미합니다.
- 각 헤더를 다른 헤더와 독립적으로 사용할 수 있어야 합니다. 따라서 필요한 모든 종속성을 포함해야 합니다. 그러나 포함 가드를 추가하여 이중 포함 문제를 피해야 합니다. 그렇지 않으면 성적은 0이 됩니다.

나를 읽어주세요

- 필요한 경우(예: 코드 분할) 몇 가지 추가 파일을 추가할 수 있습니다. 이러한 과제는 프로그램에 의해 확인되지 않으므로 필수 파일을 제출하는 한 자유롭게 확인하십시오.
- 때로는 연습 지침이 짧아 보이지만 예를 통해 지침에 명시적으로 기록되지 않은 요구 사항.
- 시작하기 전에 각 모듈을 완전히 읽으십시오! 정말로, 그렇게 하세요.
- 오딘과 토르의 작품! 머리를 써라!!!



많은 클래스를 구현해야 합니다. 지루해 보일 수도 있지만, 좋아하는 텍스트 편집기를 스크립트로 작성할 수 없다면 말이죠.



연습을 완료할 수 있는 어느 정도의 자유가 주어집니다. 그러나 필수 규칙을 따르고 게으르지 마십시오. 당신은 유용한 정보가 많이 그리워요! 주저하지 말고 읽어보세요. 이론적 개념.

제3장

모듈별 규칙

이 모듈의 각 연습을 수행하려면 표준 컨테이너를 사용해야 합니다.

컨테이너를 사용하면 나머지 모듈에서는 사용할 수 없습니다.



해당 주제를 수행하기 전에 해당 주제를 전체적으로 읽어 보는 것이 좋습니다.
수업 과정.



각 운동마다 최소한 하나의 용기를 사용해야 합니다.
두 개의 컨테이너를 사용해야 하는 연습 02는 예외입니다.


-Wall, -Wextra 및 -Werror 플래그를 사용하여 소스 파일을 필수 출력으로 컴파일할 각 프로그램에 대해 Makefile을 제출해야 합니다.

C++를 사용해야 하며 Makefile을 다시 연결해서는 안 됩니다.

Makefile에는 최소한 \$(NAME), all, clean, fclean 및 re 규칙이 포함되어야 합니다.

제4장

연습 00: 비트코인 거래소

	운동 : 00
비트코인 거래소	
제출 디렉터리 : ex00/ 제출할 파일 :	
Makefile, main.cpp, BitcoinExchange.{cpp, hpp}	
금지된 기능 : 없음	

특정 날짜에 특정 금액의 비트코인 가치를 출력하는 프로그램을 만들어야 합니다.

이 프로그램은 비트코인 가격을 나타내는 csv 형식의 데이터베이스를 사용해야 합니다.
시간이 지남에 따라. 이 데이터베이스는 이 주제와 함께 제공됩니다.

프로그램은 평가할 다양한 가격/날짜를 저장하는 두 번째 데이터베이스를 입력으로 사용합니다.

프로그램은 다음 규칙을 준수해야 합니다.

- 프로그램 이름은 btc입니다.
- 프로그램은 파일을 인수로 사용해야 합니다.
- 이 파일의 각 줄은 "날짜 | 값" 형식을 사용해야 합니다.
- 유효한 날짜는 항상 연-월-일 형식입니다.
- 유효한 값은 부동 소수점 또는 0에서 1000 사이의 양의 정수여야 합니다.



이 연습을 검증하려면 코드에 컨테이너를 하나 이상 사용해야 합니다. 적절한 방법으로 발생할 수 있는 오류를 처리해야 합니다.
에러 메시지.

다음은 input.txt 파일의 예입니다.

```
$> 헤드 input.txt 날짜 | 값
2011-01-03 | 3
2011-01-03 | 2
2011-01-03 | 1
2011-01-03 | 1.2
2011-01-09 | 1
2012-01-11 | -1
2001-42-42 2012-01-11
| 1 2012-01-11
| 2147483648 $>
```

프로그램은 입력 파일의 값을 사용합니다.

프로그램은 곱한 값의 결과를 표준 출력에 표시해야 합니다.
데이터베이스에 표시된 날짜에 따른 환율로 계산됩니다.



입력에 사용된 날짜가 DB에 없으면 DB에 포함된 가장 가까운 날짜를 사용해야 합니다. 상위 날짜가 아닌 하위 날짜를 사용하도록 주의하세요.

다음은 프로그램의 사용 예이다.

```
$> ./btc 오
류: 파일을 열 수 없습니다. $> ./btc
input.txt 2011-01-03 => 3
= 0.9 2011-01-03 => 2 = 0.6

2011-01-03 => 1 = 0.3
2011-01-03 => 1.2 = 0.36
2011-01-09 => 1 = 0.32
오류: 양수가 아닙니다.
오류: 잘못된 입력 => 2001-42-42 2012-01-11
=> 1 = 7.1
오류: 숫자가 너무 큼니다. $>
```



경고: 이 연습을 검증하는 데 사용하는 컨테이너는 이 모듈의 나머지 부분에서 더 이상 사용할 수 없습니다.

제5장

연습 01: 역 폴란드 표기법

	운동 : 01
RPN	
제출 디렉터리 : ex01/ 제출할 파일 :	
Makefile, main.cpp, RPN.{cpp, hpp}	
금지된 기능 : 없음	

다음 제약 조건을 사용하여 프로그램을 작성해야 합니다.

- 프로그램 이름은 RPN입니다.
- 프로그램은 반전된 폴란드 수학 표현을 인수로 사용해야 합니다.
멘션.
- 이 연산에 사용되고 인수로 전달되는 숫자는 항상 10보다 작습니다. 계산 자체는 물론 결과에도 이 규칙이 고려되지 않습니다.
- 프로그램은 이 표현식을 처리하고 올바른 결과를 출력해야 합니다.
표준 출력.
- 프로그램 실행 중 오류가 발생하면 오류 메시지가 표시되어야 합니다.
표준 출력에 표시됩니다.
- 프로그램은 "+ - / *" 토큰을 사용하여 작업을 처리할 수 있어야 합니다.



이를 검증하려면 코드에 하나 이상의 컨테이너를 사용해야 합니다.
운동.



대괄호나 소수를 관리할 필요가 없습니다.

다음은 표준 사용의 예입니다.

```
$> ./RPN "8 9 * 9 - 9 - 4 - 1 +" 42 $> ./RPN "7 7 * 7 -" 42
```

```
$> ./RPN "1 2 * 2 / 2 * 2 4 - +" 0
```

```
$> ./RPN "(1 + 1)"
```

```
오류 $>
```




경고: 이전 연습에서 사용한 컨테이너는 여기서 금지됩니다. 이 연습을 검증하는 데 사용한 컨테이너

이 모듈의 나머지 부분에서는 사용할 수 없습니다.

제6장

연습 02: PmergeMe

	운동 : 02
PmergeMe	
제출 디렉터리 : ex02/ 제출할 파일 :	
Makefile, main.cpp, PmergeMe.{cpp, hpp}	
금지된 기능 : 없음	

다음 제약 조건을 사용하여 프로그램을 작성해야 합니다.

- 프로그램 이름은 PmergeMe 입니다.
- 프로그램은 양의 정수 시퀀스를 인수로 사용할 수 있어야 합니다.
- 프로그램은 양의 정수를 정렬하려면 병합-삽입 정렬 알고리즘을 사용해야 합니다.
순서.



명확히 하려면 Ford-Johnson 알고리즘을 사용해야 합니다.

- 프로그램 실행 중 오류가 발생하면 표준 출력에 오류 메시지가 표시되어야 합니다.



이 연습을 검증하려면 코드에 두 개 이상의 서로 다른 컨테이너를 사용해야 합니다. 프로그램은 최소한 3000개의 서로 다른 정수를 처리할 수 있어야 합니다.



각 컨테이너에 대해 알고리즘을 구현하여 일반 함수를 사용하지 않는 것이 좋습니다.

다음은 표준 출력에 한 줄씩 표시해야 하는 정보에 대한 몇 가지 추가 지침입니다.

- 첫 번째 줄에는 명시적인 텍스트와 정렬되지 않은 긍정문을 표시해야 합니다.
정수 시퀀스.
- 두 번째 줄에는 명시적인 텍스트와 정렬된 긍정적인 텍스트를 표시해야 합니다.
정수 시퀀스.
- 세 번째 줄에는 양의 정수를 정렬하는 데 사용되는 첫 번째 컨테이너를 지정하여 알고리즘에서 사용되는 시간을 나타내는 명시적인 텍스트를 표시해야 합니다.
순서.
- 마지막 줄에는 양의 정수 시퀀스를 정렬하는 데 사용되는 두 번째 컨테이너를 지정하여 알고리즘에서 사용되는 시간을 나타내는 명시적인 텍스트를 표시해야 합니다.



분류를 수행하는 데 사용된 시간 표시 형식은 자유지만 선택한 정밀도는 항목을 명확하게 볼 수 있어야 합니다.
사용된 두 컨테이너의 차이점

다음은 표준 사용의 예입니다.

```
$> ./PmergeMe 3 5 9 7 4 이전: 3 5 9
7 4 이후: 3 4 5 7 9 std::[...]를
사용하여 5개의 요소 범위를 처리
하는 데 걸리는 시간 : 0.00031 us 5개의 범위를 처리하는 데 걸리는 시간 std::[...] 포함 요소 : 0.00014 us $> ./
PmergeMe `shuf -i 1-100000 -n 3000 | tr "\n" " "` 이전: 141 79 526 321 [...]

이후: 79 141 321 526 [...]
std::[...]를 사용하여 3000개 요소 범위를 처리하는 데 걸리는 시간 : 62.14389 us std::[...]를 사용하여 3000개 요소 범위를
처리하는 데 걸리는 시간 : 69.27212 us $> ./PmergeMe "-1" "2"

오류 $>
# OSX 사용자의 경우: $> ./
PmergeMe `jot -r 3000 1 100000 | tr "\n" " "` [...] $>
```



이 예에서는 시간 표시가 의도적으로 이상합니다.

물론 정렬 부분과 데이터 관리 부분 모두에서 모든 작업을 수행하는 데 사용된 시간을 표시해야 합니다.



경고: 이전 연습에서 사용한 컨테이너는 다음과 같습니다.
여기서는 금지되어 있습니다.



중복과 관련된 오류에 대한 관리는 귀하의 몫입니다.
재량.

제7장

제출 및 동료 평가

평소처럼 Git 저장소에 과제를 제출하세요. 방어 중에는 저장소 내부 작업만 평가됩니다. 주저하지 말고 풀더와 파일의 이름이 올바른지 다시 확인하세요.