



C++ - 모듈 06

C++ 캐스트

요약:

이 문서에는 C++ 모듈 중 모듈 06의 연습 문제가 포함되어 있습니다.

버전: 6.2

콘텐츠

I	소개	2
II	일반 규칙	3
III	추가 규칙	5
IV	연습 00: 스칼라 유형 변환하기	6
V	연습 01: 직렬화	9
VI	연습 02: 실제 유형 식별하기	10
VII	제출 및 동료 평가	11

1장 소개

C++는 비야른 스트로스트룹이 C 프로그래밍 언어, 즉 "클래스가 있는 C"의 변형으로 만든 범용 프로그래밍 언어입니다(출처: [위키백과](#)).

이 모듈의 목표는 **객체 지향 프로그래밍**을 소개하는 것입니다. 이것이 C++ 여정의 출발점이 될 것입니다. OOP를 배우기 위해 많은 언어가 권장됩니다. C++는 여러분의 오랜 친구인 C에서 파생된 복잡한 언어이기 때문에 C++를 선택하기로 결정했으며, 코드를 단순하게 유지하기 위해 C++98 표준을 준수할 것입니다.

최신 C++는 여러 측면에서 많이 다르다는 것을 알고 있습니다. 따라서 능숙한 C++ 개발자가 되고 싶다면 42개의 공통 코어를 넘어서는 것은 여러분에게 달려 있습니다!

2장 일반 규칙

컴파일

- c++와 -Wall -Wextra -Werror 플래그를 사용하여 코드를 컴파일합니다.
- std=c++98 플래그를 추가하면 코드가 계속 컴파일됩니다.

서식 지정 및 이름 지정 규칙

- 연습 디렉터리의 이름은 ex00, ex01, ..., exn
- 가이드라인에서 요구하는 대로 파일, 클래스, 함수, 멤버 함수 및 속성의 이름을 지정하세요.
- 클래스 이름을 **대문자 대소문자** 형식으로 작성합니다. 클래스 코드가 포함된 파일은 항상 클래스 이름에 따라 이름이 지정됩니다. 예를 들어 ClassName.hpp/ClassName.h, ClassName.cpp 또는 ClassName.hpp. 예를 들어 벽돌 벽을 의미하는 "BrickWall" 클래스의 정의가 포함된 헤더 파일이 있는 경우 그 이름은 BrickWall.hpp가 됩니다.
- 달리 지정하지 않는 한 모든 출력 메시지는 새 줄 문자로 끝나야 하며 표준 출력에 표시되어야 합니다.
- *안녕, 노미넷!* C++ 모듈에는 코딩 스타일이 적용되지 않습니다. 좋아하는 스타일을 따를 수 있습니다. 하지만 동료 평가자가 이해할 수 없는 코드는 채점할 수 없는 코드라는 점을 명심하세요. 깔끔하고 읽기 쉬운 코드를 작성하기 위해 최선을 다하세요.

허용/금지

더 이상 C로 코딩하지 않습니다. C++로 전환할 때입니다! 그러므로:

- 표준 라이브러리의 거의 모든 것을 사용할 수 있습니다. 따라서 이미 알고 있는 것을 고수하는 대신 익숙한 C++ 버전의 C 함수를 최대한 많이 사용하는 것이 현명할 것입니다.
- 그러나 다른 외부 라이브러리는 사용할 수 없습니다. 즉, C++11(및 파생된 형식)과 Boost 라이브러리는 금지됩니다. 다음 함수도 금지됩니다: `*printf()`, `*alloc()` 및 `free()`. 이 함수를 사용하면 성적이 0점이 됩니다.

- 명시적으로 달리 명시되지 않는 한, 네임스페이스 <ns_name>과 친구 키워드는 금지되어 있습니다. 그렇지 않으면 성적은 -42점이 됩니다.
- **모듈 08과 09에서만 STL을 사용할 수 있습니다.** 즉, 그때까지는 **컨테이너**(벡터 /리스트/맵 등)와 **알고리즘**(<알고리즘> 헤더를 포함해야 하는 모든 것)을 사용할 수 없습니다. 그렇지 않으면 성적이 -42점이 됩니다.

몇 가지 설계 요구 사항

- 메모리 누수는 C++에서도 발생합니다. 메모리를 할당할 때(새로운 키워드)의 경우 **메모리 누수**를 방지해야 합니다.
- 모듈 02부터 모듈 09까지, **명시적으로 달리 명시된 경우를 제외하고는** 클래스를 **정통 정석 형식으로** 설계해야 합니다.
- 헤더 파일에 포함된 모든 함수 구현(함수 템플릿 제외)은 연습 문제에서 0을 의미합니다.
- 각 헤더를 다른 헤더와 독립적으로 사용할 수 있어야 합니다. 따라서 필요한 모든 종속성을 포함해야 합니다. 그러나 **포함 가드**를 추가하여 이중 포함 문제를 피해야 합니다. 그렇지 않으면 성적이 0점이 됩니다.

읽기

- 필요한 경우(예: 코드를 분할하는 경우) 추가 파일을 추가할 수 있습니다. 이러한 과정은 프로그램에서 확인하지 않으므로 필수 파일을 제출하는 한 자유롭게 추가할 수 있습니다.
- 때로는 연습 지침이 짧아 보이지만 예제를 통해 지침에 명시적으로 작성되지 않은 요구 사항을 확인할 수 있습니다.
- 시작하기 전에 각 모듈을 완전히 읽어보세요! 정말 그렇게 하세요.
- 오딘과 토르의 힘으로! 머리를 써!!!



많은 클래스를 구현해야 합니다. 좋아하는 텍스트 편집기를 사용할 수 없다면 지루해 보일 수 있습니다.



연습을 완료하는 데는 어느 정도의 자유가 주어집니다. 그러나 필수 규칙을 따르고 게으르지 마세요. 유용한 정보를 많이 놓칠 수 있습니다! 주저하지 말고 이론적 개념에 대해 읽어보세요.


제3장 추가 규칙

다음 규칙은 전체 모듈에 적용되며 선택 사항이 아닙니다.

각 연습에 대해 유형 변환은 하나의 특정 유형의 캐스팅을 사용하여 해결해야 합니다.
방어 중에 선택 사항이 확인됩니다.

제4장

연습 00: 스칼라 유형 변환하기

	운동 00
스칼라 유형 변환	
체크인 디렉토리 : <i>ex00/</i>	
제출할 파일 : 메이크파일, *.cpp, *.{h, hpp}	
허용된 함수: 문자열에서 정수, 실수 또는 복수로 변환하는 모든 함수. 이 함수는 도움이 되지만 모든 작업을 수행하지는 않습니다.	

가장 일반적인 형식의 C++ 리터럴의 문자열 표현을 매개변수로 받고 그 값을 다음 일련의 스칼라 유형으로 출력하는 정적 메서드 "convert"를 하나만 포함하는 ScalarConverter 클래스를 작성합니다:

- char
- int
- float
- double

이 클래스는 아무것도 저장할 필요가 없으므로 사용자가 인스턴스화할 수 없어야 합니다. 문자 매개변수를 제외하고는 십진수 표기법만 사용됩니다.

문자 리터럴의 예: 'c', 'a', ...

간단하게 표시할 수 없는 문자는 입력으로 사용해서는 안 된다는 점에 유의하세요. 표시 가능한 문자로 변환할 수 없는 경우 정보 메시지를 인쇄합니다.

int 리터럴의 예: 0, -42, 42...

부동 소수점 리터럴의 예: 0.0f, -4.2f, 4.2f...

이러한 의사 리터럴도 처리해야 합니다(과학의 경우): -inff, +inff

및 `nanf`.

이중 리터럴의 예: `0.0`, `-4.2`, `4.2...`

이러한 의사 리터럴도 처리해야 합니다(재미로): `-inf`, `+inf` 및 `nan`.

클래스가 예상대로 작동하는지 테스트하는 프로그램을 작성합니다.


먼저 매개변수로 전달된 리터럴의 유형을 감지하고 문자열에서 실제 유형으로 변환한 다음 다른 세 가지 데이터 유형으로 **명시적으로** 변환해야 합니다. 마지막으로 아래와 같이 결과를 표시합니다.

변환이 의미가 없거나 오버플로우가 발생하면 사용자에게 유형 변환이 불가능하다는 메시지를 표시합니다. 숫자 제한 및 특수 값을 처리하기 위해 필요한 헤더를 포함하세요.

```
./convert 0
char: 표시할 수 없는
int: 0
플로트:
0.0F 더블:
0.0
./convert 나노
문자: 불가능 인
트: 불가능 플로
트: 나노프 더블
나노
./convert 42.0f
문자: ""
int: 42
float: 42.0F
더블: 42.0
```


제5장

연습 01: 직렬화

	운동 : 01
	직렬화
체크인 디렉토리 : <i>ex01/</i>	
제출할 파일 : 메이크파일, *.cpp, *.{h, hpp}	
금지된 기능 : 없음	

사용자가 어떤 식으로든 초기화할 수 없는 Serializer 클래스를 다음과 같은 정적 메서드로 구현합니다:

```
uintptr_t serialize(Data* ptr);
```

포인터를 받아 부호 없는 정수 유형인 uintptr_t로 변환합니다.

```
데이터* 역직렬화(uintptr_t raw);
```

이 함수는 부호 없는 정수 매개변수를 받아 데이터에 대한 포인터로 변환합니다.

클래스가 예상대로 작동하는지 테스트하는 프로그램을 작성합니다.


비어 있지 않은(데이터 멤버가 있다는 의미) 데이터 구조를 만들어야 합니다.

Data 객체의 주소에 serialize()를 사용하고 반환값을 deserialize()에 전달합니다. 그런 다음 역직렬화()의 반환값이 원래 포인터와 동일한지 확인합니다.

데이터 구조의 파일을 제출하는 것을 잊지 마세요.

제6장

연습 02: 실제 유형 식별하기

	운동 : 02
	실제 유형 식별
	체크인 디렉토리 : <i>ex02/</i>
	제출할 파일 : 메이크파일, *.cpp, *.{h, hpp}
	금지 함수 : <code>std::typeid</code>

공용 가상 소멸자만 있는 **Base** 클래스를 구현합니다. **Base**를 공개적으로 상속하는 빈 클래스 **A**, **B**, **C**를 3개 생성합니다.



이 네 가지 클래스는 정통 정석 형식으로 디자인할 필요가 없습니다.

다음 기능을 구현합니다:

```
Base * generate(void);
```

A, **B** 또는 **C**를 무작위로 인스턴스화하고 인스턴스를 **Base** 포인터로 반환합니다. 무작위 선택 구현에는 원하는 것을 자유롭게 사용하세요.

```
void identify(Base* p);
```

p가 가리키는 객체의 실제 유형인 "**A**", "**B**" 또는 "**C**"를 인쇄합니다.

```
void identify(Base& p);
```

이 함수는 **p**가 가리키는 객체의 실제 유형인 "**A**", "**B**" 또는 "**C**"를 인쇄합니다. 이 함수 내에서 포인터를 사용하는 것은 금지되어 있습니다.

`typeid` 헤더를 포함하는 것은 금지되어 있습니다.

모든 것이 예상대로 작동하는지 테스트하는 프로그램을 작성합니다.

제 7장

제출 및 동료 평가

평소처럼 Git 저장소에 과제를 제출하세요. 방어 중에는 저장소 내의 작업만 평가됩니다.
폴더와 파일의 이름이 정확한지 다시 한 번 확인하시기 바랍니다.



16D85ACC441674FBA2DF65190663E136253996A5020347143B460E2CF3A3784D794B
104265933C3BE5B62C4E062601EC8DD1F82FEB73CB17AC57D49054A7C29B5A5C1D8
2027A997A3E24E387

