



C++ - モジュール 06

C++キャスト

概要

本書は、C++モジュールのうち、Module 06の演習を収録したものです。

バージョン : 5

内容

I	はじめに	2
II	一般規定	3
III	追加ルール	5
IV	練習問題00：スカラー型の変換	6
V	練習問題01：シリアライズ	8
VI	練習問題02：実数型の識別	9

第一章 はじめに

C++は、*Bjarne Stroustrup*が「C with Classes」（出典：[Wikipedia](#)）というプログラミング言語の元祖として作った汎用プログラミング言語である。

これらのモジュールの目標は、オブジェクト指向プログラミングを紹介することです。これは、あなたのC++の旅の出発点となるでしょう。オブジェクト指向を学ぶには、多くの言語が推奨されます。この言語は複雑な言語であり、物事を単純化するために、あなたのコードはC++98標準に準拠することになります。

私たちは、現代のC++が多く側面で大きく異なっていることを認識しています。ですから、もしあなたが熟練したC++開発者になりたいのであれば、42のコモンコアの後にさらに進むのはあなた次第なのです。

第II章 総則

コンパイル

- `c++`と`-Wall -Wextra -Werror`フラグでコードをコンパイルします。
- フラグ `-std=c++98` を追加しても、あなたのコードはコンパイルできるはずです。

フォーマットと命名規則

- 演習用のディレクトリは、`ex00`, `ex01`, ...のように命名されます。 `exn`
- ファイル、クラス、関数、メンバ関数、属性にガイドラインに必要な名前を付ける。
- クラス名は、**UpperCamelCase** 形式で記述します。クラスコードを含むファイルには、常にクラス名に従った名前が付けられます。例えば例えば、`ClassName.hpp/ClassName.h`, `ClassName.cpp`, または `ClassName.tpp`.例えば、レンガの壁を表すクラス "`BrickWall` "の定義を含むヘッダーファイルがあれば、そのファイル名は`BrickWall.hpp`となります。
- 特に指定がない限り、すべての出力メッセージは改行文字で終了し、標準出力に表示されなければならない。
- さよならノルミネット!C++モジュールでは、コーディングスタイルは強制されません。あなたの好きなものに従えばいいのです。しかし、同僚評価者が理解できないコードは、評価できないコードであることを心に留めておいてください。きれいで読みやすいコードを書くために、最善を尽くしてください。

許可/禁止

あなたはもうCでコーディングしていない。C++の時代だ!というわけで。

- 標準ライブラリのほとんどすべてを使用することが許されています。したがって、すでに知っているものに固執するのではなく、使い慣れたC関数のC++的なバージョンをできるだけ使うのが賢い方法でしょう。
- ただし、それ以外の外部ライブラリは使用できません。つまり、C++11（および派生形式）とBoostライブラリは禁止されています。以下の関数も禁止されています。`*printf()`、`*alloc()`、`free()`。これらを使用した場合、成績は0点、それで終わりです。

- 明示的に指定しない限り、`using`名前空間`<ns_name>`と友人キーワードは禁止です。そうでない場合、あなたの成績は-42となります。
- **STL の使用はモジュール 08 のみ許可されています。**つまり、コンテナ (`vector/list/map/` など) とアルゴリズム (`<algorithm>` ヘッダを含む必要があるもの) はそれまで使わないでください、ということです。さもないと、あなたの成績は -42 になります。

いくつかのデザイン要件

- C++でもメモリリークは発生します。メモリを確保するときに (`new` キーワード)を使用する場合は、メモリリークを回避する必要があります。
- モジュール02からモジュール08までは、特に明記されている場合を除き、正教会の正書法で授業を設計する必要があります。
- ヘッダーファイルに書かれた関数の実装は（関数テンプレートを除いて）、演習では0を意味します。
- それぞれのヘッダーは、他のヘッダーと独立して使用できるようにする必要があります。従って、必要な依存関係はすべてインクルードしなければなりません。しかし、インクルードガードを追加することによって、二重インクルードの問題を回避しなければなりません。そうでなければ、あなたの成績は0点となります。

リードミー

- 必要であれば、いくつかのファイルを追加することができます（コードを分割するためなど）。これらの課題は、プログラムによる検証を行わないので、必須ファイルを提出する限り、自由に行ってください。
- 演習のガイドラインは短く見えても、例題には明示的に書かれていない要件が示されていることがあります。
- 始める前に、各モジュールを完全に読んでください本当に、そうしてください。
- オーディンによって、ツールによって!頭を使え!!!



多くのクラスを実装する必要があります。これは、お気に入りのテキストエディタでスクリプトを書くことができる人でなければ、退屈に思えるかもしれません。



演習をこなすには、ある程度の自由度が与えられています。ただし、必須のルールは守り、怠慢は禁物です。多くの有益な情報を見逃すこととなりますよ。理論的な概念については、ためらわずに読んでください。


第三章 追加規定

以下のルールは、モジュール全体に適用され、オプションではありません。

各演習では、特定の1種類のキャストを使って型変換を解かなければなりません。
選択した内容は、防衛時に確認されます。

第四章

練習問題00：スカラー型の変換

	エクササイ ズ00 スカラー型の変換
ターンインディレクトリ : <i>ex00/</i>	
提出するファイル: <i>Makefile</i> 、 <i>*.cpp</i> 、 <i>*.{h, hpp}</i> 。	
使用可能な関数： 文字列を <i>int</i> , <i>float</i> , <i>double</i> に変換する関数。こ れは助けになるが、すべての仕事をするわけではない。	

C++ のリテラルを最も一般的な形で文字列表現したものをパラメータとするプログラムを作成しなさい。このリテラルは、*char*、*int*、*float*、*double* のいずれかのスカラー型に属さなければならない。*char* 型のパラメータを除いて、10 進法のみが使用されます。

char リテラルの例 : *'c'*、*'a'*、....
簡単のために、表示不可能な文字は入力として使うべきではないことに注意してください。*char* への変換が表示不可能な場合、情報提供のメッセージを表示します。

int リテラルの例 : 0, -42, 42....*float* リテラル

の例 : 0.0*f*, -4.2*f*, 4.2*f*....
これらの疑似リテラルも扱わなければなりません (ご存知のように、科学のためです)。*-inff*, *+inff*
と *nanf*.

ダブルリテラルの例 : 0.0, -4.2, 4.2....
これらの疑似リテラルも扱わなければなりません (お遊びです)。*-inf*、*+inf*、*nan* です。


まず、パラメータとして渡されたリテラルの型を検出し、それを文字列から実際の型に変換し、次に他の3つのデータ型に**明示的に**変換する必要があります。最後に、以下のように結果を表示します。

変換が意味をなさないか、オーバーフローする場合、型変換が不可能であることを知らせるメッセージを表示する。数値の制限や特殊な値を扱うために必要なヘッダを含めてください。

```
./convert 0
char:表示不可 int: 0
float: 0.0f
double: 0.0
./convert nan
char: 不可 int: 不
可 float: nanf
double: nan
./convert 42.0f
char: '*'
int: 42
float:42.0f
double42.0
```


第五章

練習問題01：シリアライズ

	演習：01 シリアライズ
	データインディレクトリ： <i>ex01/</i>
	提出するファイル： <i>Makefile</i> 、 <i>*.cpp</i> 、 <i>*.{h, hpp}</i> 。
	禁止されている機能：なし

以下の機能を実装する。

`uintptr_t serialize(Data* ptr);`

ポインタを受け取り、符号なし整数型 `uintptr_t` に変換する。

データ*のデシリアライズ(`uintptr_t raw`)。

符号なし整数のパラメータを受け取り、`Data`へのポインタに変換する。

作成した関数が期待通りに動作することを確認するためのプログラムを作成します。


空でない（データメンバを持つという意味）データ構造を作成する必要があります。

`Data` オブジェクトのアドレスに対して `serialize()` を使用し、その戻り値を `deserialize()` に渡します。そして、`deserialize()` の戻り値が元のポインタと等しいことを確認します。

データ構造のファイルを提出することを忘れないでください。

第六章

練習問題02：実数型の識別

	演習：02
リアルタイプの識別	
ターンインディレクトリ：ex02/	
提出するファイル：Makefile、*.cpp、*.{h、hpp}。	
禁制の関数：std::typeinfo	

パブリックな仮想デストラクタのみを持つ **Base** クラスを実装しなさい。Baseを継承した3つの空のクラス **A**, **B**, **C** を作成しなさい。



この4つのクラスは、正統派の正典形式で設計する必要はありません。

以下の機能を実装する。

ベース * generate(void);

A, B, Cのいずれかをランダムにインスタンス化し、そのインスタンスをBaseポインタとして返します。ランダムチョイスの実装は自由に行ってください。

void identify(Base* p);

p が指すオブジェクトの実際の型、"A"、"B"、"C" を表示する。

void identify(Base& p);

p が指すオブジェクトの実際の型 ("A", "B", "C") を表示します。この関数の内部でポインタを使用することは禁じられています。

typeidヘッダを含めることは禁止されています。

すべてが期待通りに動作することをテストするプログラムを書いてください。