



C++ - モジュール 01

メモリ割り当て、メンバへのポインタ、参照
、switch文

概要

このドキュメントには、C++ モジュールのモジュール01の練習問題が含まれています。

バージョン: 10

内容

I	はじめに	2
II	一般規定	3
III	エクササイズ00: BraiiiiinnnzzzzZ	5
IV	練習01: もっと頭を使おう!	6
V	エクササイズ02: HI THIS IS BRAIN	7
VI	エクササイズ03: 不必要な暴力	8
VII	練習04: セッドは敗者のためのもの	10
VIII	練習05: ハール2.0	11
IX	練習06: ハールフィルター	13
X	提出と相互評価	14

第一章 はじめに

C++は、*Bjarne Stroustrup*（ビャルネ・ストルストラップ）により、C言語の発展形として開発された汎用プログラミング言語である（出典：[Wikipedia](#)）。

これらのモジュールの目的は、**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。OOPを学ぶには多くの言語が推奨されています。これは複雑な言語であり、物事をシンプルに保つために、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++が多くの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、もしあなたが熟達した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です！

第II章 一般規定

コンパイル

- `c++`と`-Wall -Wextra -Werror`フラグでコードをコンパイルする。
- フラグ`-std=c++98`を追加しても、コードはコンパイルできるはずだ。

フォーマットと命名規則

- 演習用のディレクトリは次のように命名されます: `ex00`, `ex01`, ..., `exn`.
`exn`
- ファイル名、クラス名、関数名、メンバ関数名、属性名は、ガイドラインの要求に従ってください。
- クラス名は**UpperCamelCase**形式で記述する。クラス・コードを含むファイルは常にクラス名に従って命名されます。例えば例えば、
`ClassName.hpp/ClassName.h`、`ClassName.cpp`、`ClassName.hpp`。例えば、
`ClassName.hpp/ClassName.hpp/ClassName.cpp/ClassName.hpp`のようになります。
- 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。
- さようならノルミネット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。あなたの好きなスタイルに従ってください。しかし、相互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よう、ベストを尽くしてください。

許可/禁止

あなたはもうC言語でコーディングしていない。C++の出番だ！ だから

- 標準ライブラリのほとんどすべてを使うことが許されている。したがって、すでに知っていることに固執するのではなく、使い慣れたC関数のC++っぽいバージョンをできるだけ使うのが賢いやり方だろう。
- ただし、それ以外の外部ライブラリーは使用できない。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る。以下の関数も禁止されている：`*printf()`、`*alloc()`、`free()`です。これらを使用した場合、あなたの成績は0点となり、それで終わりです。

- 明示的に指定されていない限り、using 名前空間 <ns_name> と友達キーワードは禁止。さもないと、あなたの成績は-42点となる。
- **モジュール 08 と 09 でのみ STL を使用できます。**つまり、それまでは **コンテナ**（vector/list/map/など）や **アルゴリズム**（<algorithm>ヘッダーを含む必要があるもの）を使用しないこと。そうでなければ、成績は-42点となる。

いくつかの設計要件

- メモリ・リークはC++でも発生する。メモリを確保するときに（new キーワード）、**メモリー・リーク**を避けなければならない。
- モジュール02からモジュール09まで、**特に明記されている場合を除き**、あなたのクラスは**正教会正書式**でデザインされなければなりません。
- 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）、練習にとっては0を意味する。
- それぞれのヘッダーは、他のヘッダーから独立して使えるようにしなければならない。したがって、必要な依存関係をすべてインクルードしなければならない。しかし、**インクルード・ガード**を追加することで、二重インクルードの問題を避けなければなりません。そうでなければ、あなたの成績は0点になってしまいます。

続きを読む

- 必要であれば（コードを分割するためなど）ファイルを追加しても構いません。これらの課題はプログラムによって検証されるわけではないので、必須ファイルを提出する限り、自由に行ってください。
- 時には、練習のガイドラインが短く見えても、例題を見れば、指示には明示的に書かれていない要件がわかることもある。
- 始める前に各モジュールを完全に読むこと！ 本当にそうしてください。

- オーディンによって、ソーによって！頭を使え！




たくさんのクラスを実装しなければならない。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ない限り、退屈に思えるかもしれない。



練習をこなすにはある程度の自由が与えられている。しかし、義務的なルールは守り、怠けてはいけません。多くの有益な情報を見逃すことになります！理論的な概念を読むことをためらわないでください。

第三章

エクササイズ00: BraiiiiinnnnzzzzZ

	エクササイズ: 00
BraiiiiinnnnzzzzZ	
ターンイン・ディレクトリ: <i>ex00/</i>	
提出するファイル: Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, newZombie.cpp, randomChump.cpp	
禁止されている関数: なし	

まず、ゾンビクラスを実装します。これは文字列のプライベート属性名を持っています。

ゾンビクラスにメンバ関数 `void announce(void);` を追加する。 ゾンビは以下のように自分自身をアナウンスします:

<名前>: BraiiiiinnnnzzzzZ...

角括弧 (<と>) は表示しないでください。Fooというゾンビの場合、メッセージは次のようになります:

フー BraiiiiinnnnzzzzZ...

次に、以下の2つの関数を実装する:

- ゾンビ* newZombie(std::string name);
ゾンビを作成し、名前を付け、関数のスコープ外で使えるように返す。
- void randomChump(std::string name);


ゾンビを作成し、名前を付け、ゾンビは自分自身を公表する。

さて、この練習の実際のポイントは何だろうか？ ゾンビをスタックとヒープのどちらに割り当てるのがよいかを判断しなければならない。

ゾンビは不要になったら破棄しなければなりません。デストラクタはデバッグのためにゾンビの名前のメッセージを表示しなければなりません。

第四章

練習01：もっと頭を使う！

	エクササイズ：01
もっと頭を使ってくれ！	
ターンイン・ディレクトリ：ex01/	
提出ファイル：Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, zombieHorde.cpp	
禁止されている関数：なし	

ゾンビの大群を作る時が来た！

以下の関数を適切なファイルに実装する：

```
ゾンビ    zombieHorde( int N, std::string name );
```

一回の割り当てでN個のゾンビ・オブジェクトを割り当てなければならない。そして、ゾンビを初期化し、それぞれにパラメータとして渡された名前を付けます。

この関数は最初のゾンビへのポインタを返します。


zombieHorde() 関数が期待通りに動作することを確認するために、独自のテストを実装してください。

ゾンビの一人一人にアナウンス()を呼び出してみてください。

すべてのゾンビを削除し、**メモリリーク**がないかチェックすることをお忘れなく。

第五章

エクササイズ02: HI THIS IS BRAIN

	エクササイズ: 02
ターンイン・ディレク	こんにちは、ブレインです。
トリ: <i>ex02/</i>	
提出するファイル: Makefile、main.cpp	
禁止されている関数: なし	

を含むプログラムを書きなさい:

- HI THIS IS BRAIN "に初期化された文字列変数。
- stringPTR: 文字列へのポインター。
- stringREF: 文字列への参照。

あなたのプログラムは印刷しなければならない:

- 文字列変数のメモリアドレス。
- stringPTRが保持するメモリアドレス。
- stringREFが保持するメモリアドレス


。そして

- 文字列変数の値。
- stringPTR が指す値。
- stringREF が指す値。

それだけだ。この練習のゴールは、まったく新しいと思われがちなりファレンスを、より理解しやすくすることだ。ちょっとした違いはありますが、これはすでにやっていること、つまりアドレス操作のための別の構文です。

第六章

エクササイズ03： 不必要な暴力

	練習：03
不必要な暴力	
ターンイン・ディレクトリ：ex03/	
提出ファイル：Makefile, main.cpp, Weapon.{h, hpp}, Weapon.cpp, HumanA.{h, hpp}, HumanA.cpp, HumanB.{h, hpp}, HumanB.cpp	
禁止されている関数：なし	

を持つ武器クラスを実装する：

- 文字列であるプライベート属性タイプ。
- 型への const 参照を返す getType() メンバ関数。
- setType()メンバ関数は、パラメータとして渡された新しい型を使って型を設定する。

次に、2つのクラスを作成します：HumanAと**HumanB**だ。どちらもWeaponと名前を持っている。また、メンバ関数attack()を持ち、これを表示する（もちろん、角括弧は付けない）：

<名前>は<武器タイプ>で攻撃する。

HumanAとHumanBは、この2つの小さなディテールを除けばほとんど同じである：

- HumanAはコンストラクタでWeaponを取るが、HumanBは取らない。
- 人間Bは常に武器を持っているとは限らないが、人間Aは常に武装している。
 -

もしあなたの実装が正しければ、以下のコードを実行すると、両方のテストケースについて、「粗末なスパイク付きこん棒」による攻撃と、「他の種類のこん棒」による2回目の攻撃が表示されます：

```
int main()
{
    {
        Weapon club = Weapon("crude spiked club");

        HumanA bob("Bob", club);
        bob.attack();
        club.setType("some other type of club"); bob.attack()
    ;
    }
    {
        Weapon club = Weapon("crude spiked club");

        HumanB jim("Jim");
        jim.setWeapon(club);
        jim.attack();
        club.setType("some other type of club"); jim.attack()
    ;
    }

    0を返す;
}
```


メモリーリークのチェックをお忘れなく。



どのような場合にWeaponへのポインタを使うのがベストだと思いますか？ ウェポンへの参照は？ なぜですか？ この練習を始める前に考えてみてください。

第七章

練習04：セッドは敗者のためのもの

	練習：04
セッドは敗者のもの	
ターンイン・ディレクトリ：ex04/	
提出するファイル：Makefile、main.cpp、*.cpp、*.{h、hpp}。	
禁止されている関数：std::string::replace	

ファイル名と2つの文字列s1とs2である。


ファイル<ファイル名>を開き、その内容を新しいファイルにコピーする。
<ファイル名>.replaceで、s1が出現するたびにs2に置き換える。

Cのファイル操作関数の使用は禁止されており、不正行為とみなされる。
std::stringクラスのメンバ関数は、replaceを除いてすべて使用できます。賢く使ってください！

もちろん、予期せぬ入力やエラーにも対処すること。プログラムが期待通りに動くことを確認するために、自分でテストを作成して提出しなければならない。

第VIII章 練習問題05

： ハール2.0

	練習： 05
ハール2.0	
投入ディレクトリ： <i>ex05/</i>	
提出するファイル： Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp	
禁止されている関数： なし	

ハールを知ってる？ みんな知ってるよね？ そうでない方のために、ハールがどのようなコメントをしているか、以下でご覧ください。レベル別に分類されています：

- 「DEBUG」レベル：デバッグ・メッセージには文脈情報が含まれる。主に問題診断に使用されます。

例 「7XLダブルチーズ・トリプル・ピクルス・スペシャル・ケチャップバーガーにベーコンを追加するのが大好きです。本当にそう思う！ ”

- 「INFO」レベル：これらのメッセージには広範な情報が含まれている。本番環境でのプログラム実行をトレースするのに役立つ。

例 「ベーコンを追加するとお金がかかるなんて信じられない。私のハンバーガーにはベーコンが足りない！ そうしてくれたら、追加なんて頼まないよ！ ”

- "WARNING "レベル：警告メッセージは、システムに潜在的な問題があることを示します。ただし、対処することも無視することもできま

す。

例 「私はベーコンをタダで食べる資格があると思う。私は何年も通っているのですが、あなたは先月からここで働き始めたのですから」。

- 「**ERROR**」レベル：これらのメッセージは、回復不可能なエラーが発生したことを示す。これは通常、手作業が必要な重大な問題です。
例 「こんなことは受け入れられない！ 今すぐマネージャーと話したい。」

ハールを自動化するんだ。Harlはいつも同じことを言うので、難しくはないだろう。以下のプライベート・メンバー関数を持つ**Harl**クラスを作らなければならない：

- `void debug(void);`
- `void info(void);`
- `void warning(void);`
- `void error(void);`

Harlには、パラメータとして渡されたレベルに応じて上記の4つのメンバ関数を呼び出すパブリック・メンバ関数もある：


無効 `complain(std::string level);`

この練習の目的は、**メンバー関数へのポインター**を使うことです。これは提案ではない。Harlは、if/else if/elseの森を使わずに文句を言わなければならない。それは二度考えない！

ハールがよく文句を言うことを示すテストを作成し、提出しなさい。上記のコメントの例を題材に使っても、自分でコメントを選んでもよい。

第IX章

練習06： ハールフィルター

	練習： 06
ハールフィルター	
ターンイン・ディレクトリ： <i>ex06/</i>	
提出するファイル： Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp	
禁止されている関数： なし	

Harlの言うことすべてに注意を払いたくないこともあるだろう。聞きたいログのレベルに応じて、Harlの発言をフィルターするシステムを導入する。

つのレベルのうちの1つをパラメータとするプログラムを作成する。このプログラムは、このレベル以上のすべてのメッセージを表示する。例えば

```
警告"[ 警告 ]。
ただでベーコンが食べられるんだ。
私は何年も通っているのに対して、あなたは先月からここで働き始めた。

[ERROR]
今すぐマネージャーと話したい。

$ ./harlFilter "今日の疲れがよくわからない。" "おそ
らく取るに足らない問題に文句を言っているのたろ
う"]。
```

ハールに対処する方法はいくつかあるが、最も効果的なのはスイッチを切ることだ。

実行ファイルにharlFilterという名前を付ける。

この練習では、switch文を使わなければならないし、もしかしたら発見するか

もしれない。



練習問題06を解かなくても、このモジュールに合格することができます。

第X章

提出と相互評価

通常通り、Gitリポジトリに課題を提出してください。あなたのリポジトリ内の作品だけが、ディフェンス中に評価されます。フォルダ名やファイル名が正しいかどうか、ためらわずに再確認してください。



????????????XXXXXXXXXX = \$3\$4f1b9de5b5e60c03dcb4e8c7c7e4072c

