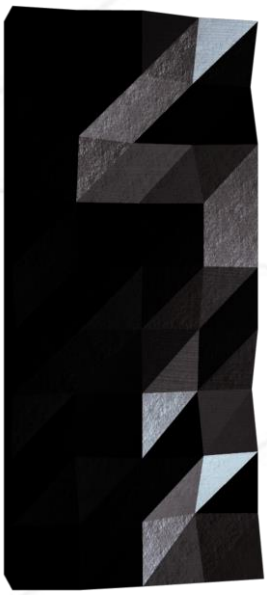

C++ - モジュール 01

メモリ割り当て、メンバへのポインタ、リファレンス、switch文

概要

本書は、C++モジュールのうち、Module 01の演習問題を収録しています。

バージョン：9

内容

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++が多くの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フラグでコンパイルしてください。
- 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。さもなければ、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、トールによって!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になりますよ。　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

エクササイズ00：BraiiiiinnnzzzZ

	演習：00
BraiiiiinnnzzzZ	
ターンインディレクトリ : *exoo/*	
提出するファイル:Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, newZombie.cpp, randomChump.cpp.	
禁止されている機能 :なし	

まず、**Zombie**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。
Zombie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>:ブレイニーインズズズ...

角括弧（<と>）は表示しない。Fooという名前のゾンビの場合、メッセージは次のようになります。

フー。ブライインズズズ...

次に、以下の2つの機能を実装する。

- Zombie* newZombie(std::string name);
  ゾンビを作り、名前を付け、それを返すので、関数スコープの外で使うことができます。
- void randomChump(std::string name);
  ゾンビを作り、名前を付け、そのゾンビが自分自身を発表する。

さて、この演習の実際のポイントは何でしょうか？どのような場合にゾンビをスタックとヒープのどちらに割り当てるのが良いかを判断する必要があります。

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、ゾンビの名前を含むメッセージを表示しなければなりません。

第四章

Exercise 01: Moar brainz!

	演習：01
Moar brainz!	
ターンインディレクトリ : *ex01/*	
提出するファイル:Makefile、main.cpp、Zombie.{h, hpp}、Zombie.cpp、zombieHorde.cpp	
禁止されている機能 :なし	

ゾンビの大群を作る時が来た!

適切なファイルに次のような関数を実装してください。

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

N個のZombieオブジェクトを一度に割り当てる必要があります。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なければなりません。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()をかけてみてください。

ゾンビをすべて削除し、**メモリリークが**ないかをチェックすることもお忘れなく。

第五章

エクササイズ02：HI THIS IS BRAIN

	演習：02
脳みそです	
ターンインディレクトリ	
: *ex02/*	
提出するファイル:Makefile、main.cpp	

禁止されている機能 :なし

含むプログラムを書いてください。

- 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。
- 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、その後に

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

これだけです、トリックはありません。この演習の目的は、全く新しいと思われる参照を解明することです。ちょっとした違いはありますが、これはアドレスの操作という、すでにやっていることの別の構文です。

第六章

練習問題03：不要な暴力

	演習：03
不必要な暴力	
ターンインディレクトリ : *ex03/*	
提出するファイル:Makefile、main.cpp、Weapon.{h, hpp}、Weapon.cpp、HumanA.{h, hpp}、HumanA.cpp、HumanB.{h, hpp}、HumanB.cpp。	
禁止されている機能 :なし	

持つWeaponクラスを実装する。

- プライベートな属性タイプであり、文字列である。
- type への const リファレンスを返す getType()メンバ関数である。
- setType()は、パラメータとして渡された新しい型を用いて型を設定するメンバー関数である。

ここで、2つのクラスを作成します。HumanAと**HumanBです**。どちらも武器と名前を持っています。また、メンバ関数attack()を持ち、（もちろん、角括弧を除いて）表示します。

<名前＞が＜武器種＞で攻撃する。

HumanAとHumanBは、この2つの小さな点を除けば、ほとんど同じです。

- HumanA はコンストラクタで Weapon を受け取りますが、HumanB は受け取りません。
- 人間Bは**常に**武器を持っているとは**限りませんが**、人間Aは**常に**武器を持っています。

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の棍棒」による攻撃と、「他の種類の棍棒」による2回目の攻撃が出力されます。

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

どのような場合にWeaponへのポインタを使用するのが良いと思いますか？ また、Weaponへの参照は？ なぜでしょうか？ この演習を始める前に考えておいてください。

```
    Weapon club = Weapon("crude spiked club");

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

  0を返します。
}
```

第七章

Exercise 04: セドは敗者のためにある

	演習：04
セドは負け組のもの	
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}.	
禁制関数：std::string::replace	

ファイル名、2つの文字列s1、s2の順に3つのパラメータを受け取るプログラムを作成しなさい。
ファイル <filename> を開き、その内容を新しいファイルにコピーします。
<ファイル名>.replaceで、s1が出現するたびにs2に置き換えます。

C言語のファイル操作関数を使用することは禁止されており、不正行為とみなされます。std::string クラスのメンバ関数は replace 以外はすべて使用可能です。賢く使いましょう!

もちろん、予期せぬ入力やエラーに対処すること。プログラムが期待通りに動くかどうか、自分でテストを作って提出する必要があります。

第VIII章 演習05：ハール2.0

	演習：05
ハール2.0	
ターンインディレクトリ : *ex05/*	
提出するファイル:Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp。	
禁止されている機能 :なし	

ハールを知ってるか？みんな知ってるよね？そうでない方のために、ハールがどんなコメントをするのか、以下にご紹介します。レベル別に分類しています。

- "**DEBUG** "レベル。デバッグメッセージには文脈情報が含まれます。これらは主に問題の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は広範な情報を含んでいる。実稼働環境でのプログラム実行をトレースするのに役立ちます。
  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十分なベーコンを入れていないじゃないか。そうしてくれたら、追加で頼まないよ！*"

- "**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可能です。
  例*"私はベーコンをタダで食べるに値すると思うんだ。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のだから*"

- "**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します。これは、通常、手動での介入を必要とする重大な問題です。
  例*"これは容認できない!今すぐマネージャーと話したい。*"

ハールを自動化するのですね。いつも同じことを言うので、難しくはないでしょう。以下のプライベートメンバー関数を持つ**Harl**クラスを作成する必要があります。

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

また、Harl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を備えています。

```
void    complain( std::string level );
```

この演習の目標は、**メンバー関数へのポインタを**使用することです。これは提案ではありません。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のです。それは二度考えません!

Harlがたくさん文句を言うことを示すテストを作って提出する。例のコメントを利用することができます。

第IX章

練習問題06：ハールフィルタ

	演習：06
ハールフィルター	
ターンインディレクトリ : *ex06/*	
提出するファイル:Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp。	
禁止されている機能 :なし	

時には、Harlが言うことすべてに注意を払いたくないこともあるでしょう。聞きたいログレベルに応じて、Harlの発言にフィルターをかけるシステムを導入してください。

4つのレベルのうち1つをパラメータとするプログラムを作成します。このレベル以上のすべてのメッセージを表示します。例えば、以下のようになります。

```
$> ./harlFilter "WARNING"
[ WARNING ]です。
ただでベーコンを余らせるのは当然だと思う。
私は何年も通っているのですが、あなたは先月から働き始めたんですね。

[ ERROR ]です。
これは容認できない、今すぐマネージャーと話したい。

$> ./harlFilter "今日の疲れは何なのかよくわからない
..."[ おそらく、取るに足らない問題の愚痴 ]。
```

ハールへの対処法はいくつかありますが、最も効果的なのは「SWITCH OFF」です。

実行ファイルにharlFilterという名前をつけます。

この演習では、switch文を使わなければなりませんし、もしかしたら発見もあるかもしれません。

このモジュールは、演習06を行わずに合格することができます。