[Prog2] Programming C++ (C6) Exercise Guide (Ex01)

Monday, Thursday 3rd period.

演習の授業（C６クラス）について

- この演習に関する情報は以下の Wiki で提供する

http://hare.u-aizu.ac.jp/prog2/2017/index.php

- 演習問題については、このWikiに載っているものを使用する。

講義のページにあるものは扱わない（AY2016より）

主な変更理由：

1. 講義・演習・販売教科書の不一致
2. 出題の意図や趣旨が一意に解釈できない、解答・採点共に困惑する設問
3. 設問中の誤り個所の長年の放置
4. 学生からの不満、授業の意義に関する質問

演習の授業（C６クラス）について

- **教科書は講義のページおよびスライドで指示された本を使用**

Accelerated C++ 効率的なプログラミングのための新しい定跡
(小林健一郎訳、ピアソンエデュケーション)

が、売店に売っていない！

⇒ 実はこの本は既に絶版

⇒ 形式的に、オライリー本が売店で販売されている

しかし、C5/6の講義・演習は Accelerated C++ に準拠したものとなっている。

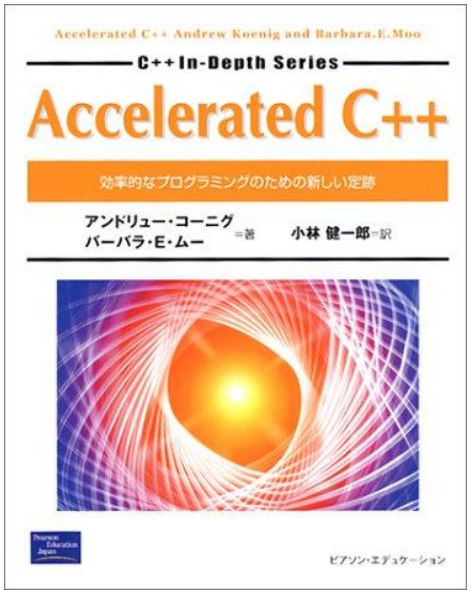
演習の授業（C６クラス）について

- オライリー本では、探したい内容を見つけるのに時間を浪費

（欲しい情報が順番に並んでいない）

- 演習問題によっては「**教科書を読んで…**」となっているが、

Accelerated C++ の説明の流れの上でのデータ構造の定義等が含まれているので、オライリー本を読んでも絶対に分かるはずがない



というわけで、授業に必要な分だけ

随時コピーをとったものをpdfで配布します

演習の趣旨（というより教科書の流れ？）

- 前半戦（ユーザー）

C言語に比べて、より扱いやすくなった機能の紹介

・ストリーム入出力

・string（文字列）

・vector, list, map（動的に可変なデータ構造）

プログラミングを効率的にする既存のツールを利用するだけ

- 後半戦（エンジニア）

C++を通してオブジェクト指向を学習（抽象化された設計）

・クラス定義、インヘリタンス、ポリモーフィズム

・string や vector の構造解析

プログラミングを効率的にするツールを自分で新たに作り出す

演習の流れ

※注意※
講義ではクラスの話題がガンガン投入される。
試験にもバンバン出ているそう。
特に前半は、中間試験への準備を想定して
講義の進捗とは独立して演習を進めます。

- **前半戦（ユーザー）**

10/02	月	Ex01: Getting started / Working with string	ストリーム入出力(cin,cout), string型
10/05	木	Ex02: Looping and counting	STLコンテナ(vector)
10/12	木	Ex03: Working with batchs of data	STLコンテナ(list)
10/16	月	Ex04: Organizing programs and data	STLコンテナ(map)
10/19	木	Ex05: Using sequential containers and analyzing strings	オブジェクト指向基礎(クラスの概念)
10/23	月	Ex06: Using library algorithms	オブジェクト指向基礎(コンストラクタ/デストラクタ)
10/26	木	Ex07: Using associative containers	オブジェクト指向基礎(継承)

- **後半戦（エンジニア）**

11/02	木	Ex08: Writing general function	関数の抽象化（テンプレート）
11/06	月	Ex09: Define new types	クラス設計補足ほか
11/09	木	Ex10: Managing memory and low-level data structures	関数ポインタ/ファイルIOなど
11/13	月	Ex11: Defining abstract data types	STLコンテナの解析(vector)/メモリマネジメント
11/16	木	Ex12: Making class objects act like values	STLコンテナの解析(string)
11/20	月	Ex13: Using inheritance and dynamic binding	クラス定義のオーバーロード/オーバーライド

後半はほぼ、講義/真の教科書通り

演習課題について

- 各回、wikiにある問題に解答する
- 提出用に、問題ごとに指定されたファイル名で作成する
- 指定のディレクトリ構造でファイルを保存しておく

```
~/Prog2
        + /Ex01
            + /ex01-1.cc
            + …
        + /Ex02
        + /Ex03
        + …
```

- 指定のスクリプトを起動して、期限までに自分で提出する
- 時々存在する、(optional)と書かれた問題の提出は任意　⇒ ボーナス問題

演習課題について

- 締切日時： 次回授業の前日まで？

⇒ 日曜および水曜の23:59:59:999まで

- これを過ぎると（特別な事情を除き）***一切提出不可***

- 正規の提出方法以外は採点できないようになっている
- 提出スクリプト cppsub が提出に関する窓口

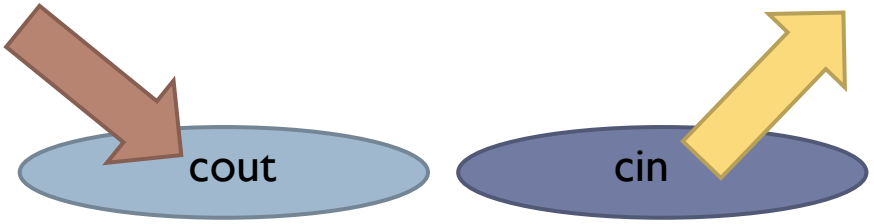
Ex01 について： 入出力ストリーム

- 入力(cin)・出力(cout) ストリームの使用

・C言語の printf , scanf に相当

std::cout << 出力したいもの

std::cin >> 入力したいもの



※ >> および << の使い方を間違えないように

cout は 出力用の穴
→ 画面に出したいものを穴の中に入れるイメージ

cin は入力用の穴
→穴の中（キーボードから入力したもの）に
手を入れて取り出すイメージ

Ex01 について： 入出力ストリーム

```
std::cout << a << b << c << d;
std:: cin >> a >> b >> c >> d;
```

のように、<< や >> を**連続して繋げると**、複数の要素を
同時に出力・入力できる（変数を書いた順番で入出力）

★C++における改行
⇒ std::endl を使う（ printf()の書式における ¥n と同じ ）
他の変数と同じように、endl もストリームに流す

```
// 変数 x の値を出力して、改行
std::cout << x << std::endl;
```

Ex01 について：文字列型の扱い

- string による文字列の扱い

【変数宣言】　　**std::string 変数名**

string型を使えば、C言語の文字列配列よりも文字列操作が
簡単になる場面が多い

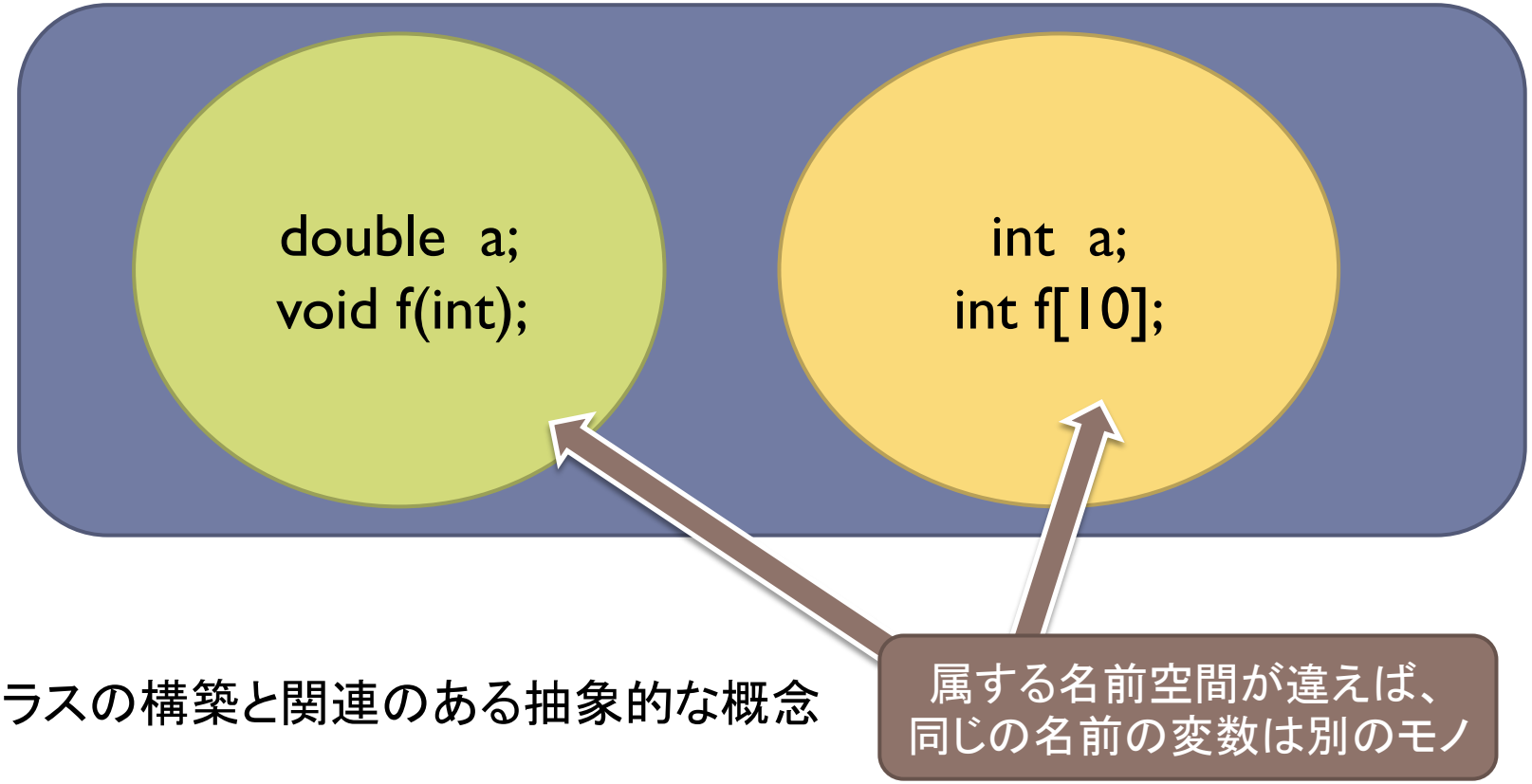
b = “ai”; c = “zu”; のとき　**a = b + c;** とすると、
a = “aizu” となる。

※**文字列変数における加法**は**文字列の連結操作**として
定義されている = strcat()
代入 a = b は C++では可能（Cにおける、strcpy(b, a); ）

a.size() = 4; // strlen に相当（文字列の長さを返す関数）

Ex01 について：名前空間

- C言語の感覚なら、通常プログラム内に全く同じ名前の変数や関数は存在してはいけない
- C++の場合、**名前空間（変数が管理されている領域）が異なれば**許される

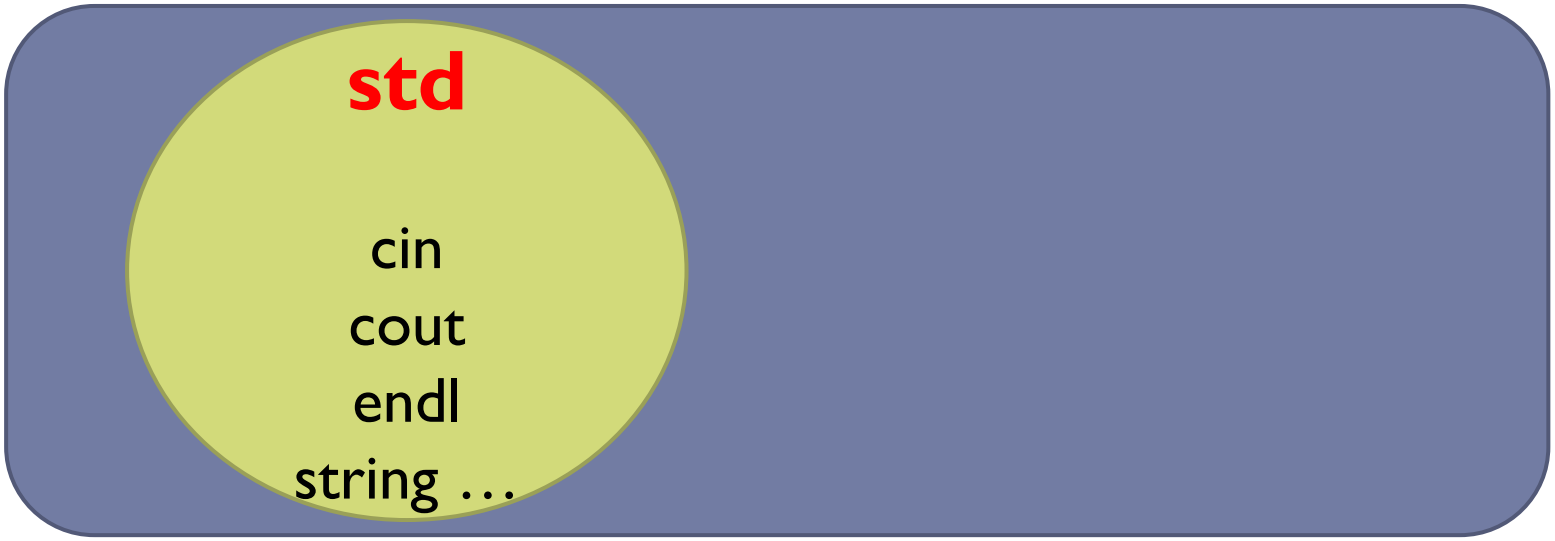


- クラスの構築と関連のある抽象的な概念

Ex01 について：名前空間

- 入出力ストリームをはじめとする標準機能は、**std** という名前空間に属する
- cin や cout の正式名称は **std::cin** ， **std::cout**

（ **stdという名前空間にいる** cin / cout だよ）



- 名前空間は自分で定義することもできる
- よく使うものに対して、毎回正式名称を書くのは大変なので、**誤解がないのならば**

using namespace std;

を最初に書いてしまえば、以後 **std:: を省略できる**。（暗黙に stdの中だと認識させる）

Ex01 について：using namespace std;

- 文字列を1つ入力させて、そのままそれを出力＆改行する（namespace なし）

```
std::string a;
std::cin >> a;
std::cout << a << std::endl;
```

- using namespace std; を使う と…（「これから出てくるものは std と思え」）

```
string a;
cin >> a;
cout << a << endl;
```

名前の被りが起こらないもの（特に、標準で使用される名前など）はとても楽に