ソフトウェア演習２B

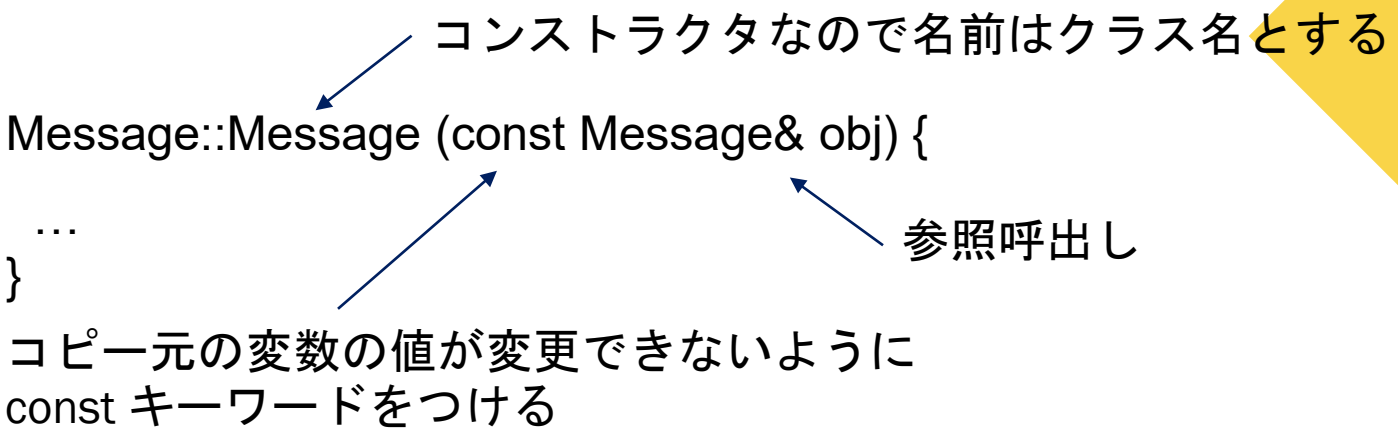
コピーコンストラクタ

コピーコンストラクタ

- 変数のコピーが行われるときに呼び出されるコンストラクタ
  - 明示的に実装しない場合にはデフォルトコピーコンストラクタが呼び出されて、メンバ変数の値がそのままコピーされる。
  - メンバ変数がポインタ変数の場合は、アドレスがコピーされてそのアドレスにある変数の値が共有されるので注意が必要である。
    - 一方の変数の値を書き変えるともう一方の変数の値も書き換わってしまう。
- メンバ変数にポインタ変数がある場合には明示的にコピーコンストラクタを実装する必要がある

コピーコンストラクタの実装

- コピーコンストラクタの実装は以下の例のようにする



vectorクラス

- 配列のような機能をもつコンテナクラス
- 配列は宣言時に要素数が固定されるが、vectorクラスでは要素数は可変
- あらゆるデータ型、クラスの変数を要素として持つためにテンプレート機能を使って実装されている
- 使用するためにはvectorクラスを定義したファイル vector（vector.hではない） をインクルードする

```
#include <vector>
```

vectorクラス:インスタンス生成

- 要素にするデータ型を指定して変数を宣言する

```
std::vector<int> container_of_int;
std::vector<std::string> container_of_string;
```

- vectorクラスの変数も要素として持てる

```
std::vector<std::vector<int>> container_of_container_of_int;
```

vectorクラス:要素の追加

- vectorクラスの以下のメンバ関数を用いて要素を追加する
  - push_back()
  - emplace_back()

```
std::vector<int> container_int;
for (int n = 0; n < 10; n++) container_int.push_back (n);
```

vectorクラス:要素へのアクセス

- 各要素にアクセスするためには配列のように演算子[]を使用する

```
std::vector<int> container_int;
for (int n = 0; n < 10; n++) container_int.push_back (n);
for (int n = 0; n < container_int.size(); n++) std::cout << container_int[n] << std::endl;
```

要素数を返す関数

vectorクラス:反復処理

- 各要素を演算子[]を使用して参照できるので、以下のような for文を使ってる反復処理を行うことができる

```
for (int n = 0; n < a.size(); n++) std::cout << a[n] << std::endl;
```

- iteratorというクラスを用いることで添え字を考えることなく反復処理を行うことができる

```
for (std::vector<int>::iterator it = a.begin(); it != a.end(); it++)  std::cout << *it << std::endl;

for (auto it = a.begin(); it != a.end(); it++)  std::cout << *it << std::endl;
```

- 特別なfor文の構文を使う

```
for (int elem : a) std::cout << elem << std::endl;
for (int& elem: a) ...
for (auto elem: a) ...
for (auto& elem: a) ...
```

  - 参照渡し（＆）にすると要素を書き変えることができる