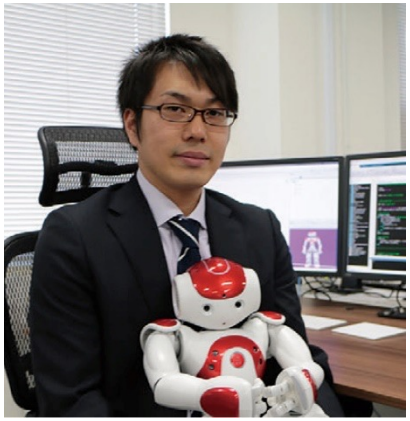
担当：
知能情報学分野
増山 直輝



HP: https://masuyama-lab.github.io/index_ja.html

情報工学演習 I

C++の演習2（クラス）

今日の内容

- string
- クラス
- ファイル入出力

string

string

実はstringというクラス

[C言語の場合]
char str[10];
など

- C++で使える、文字列を扱う便利な方法
  - 文字数を事前に決めてメモリを確保する必要が無い
  - 文字列の扱いが容易
    - “=”や“+”といった記号で操作可能
  - char*型に変換でき、C言語の関数も使用可能
    - c_str()

[C言語の場合]
string.hをインクルードして、
文字列を操作する関数を使う

参考：http://ppp-lab.sakura.ne.jp/ProgrammingPlacePlus/cpp/library/002.html

stringの例（前半）

ex6_string.cpp

```
#include <iostream>
#include <string>          // C++のstringを使うため
#include <string.h>        // Cの文字列操作関数を使うため
using namespace std;

int main() {
  string str1;             // "" で初期化
  string str2 = "abc";     // "abc" で初期化
  string str3("def");      // "def" で初期化

  str1 = str3 + "333";     // str1の変更
```

stringの例（後半）

ex6_string.cpp

```
  // 文字列の表示
  cout << "str1 = " << str1 << endl
       << "str2 = " << str2 << endl
       << "str3 = " << str3 << endl;

  // 文字列の長さの表示（C++スタイルとCスタイルの比較）
  cout << "str1.length() = " << str1.length() << endl
       << "strlen(str1.c_str()) = " << strlen(str1.c_str()) << endl;

  return 0;
}
```

クラス

クラス

- クラスは、特定の「もの」を表す型
- クラスのオブジェクトを作ることができる
- オブジェクトには、データを格納でき、関数を呼ぶことができる

加速する

クラス：スバル レヴォーグ

データ
- 排気量
- 色
- EyeSightの有無
- …

関数
- 加速する
- 減速する
- 窓を開ける
- 窓を閉める
- …

オブジェクト１

- 1.6l
- 赤
- 無

オブジェクト２

- 2.0l
- 黒
- 有

（運転席の）窓を開ける

クラスを使う利点１

- データと操作の一体化
  - クラスによって保持するデータや呼び出せる関数が異なる
    - クラス「レヴォーグ」は「排気量」というデータを持つが、クラス「体温計」にはない
    - クラス「体温計」は「熱を測る」という操作が可能だが、クラス「レヴォーグ」 にはできない
  - クラス内の関数はクラス内のデータに対して作用するので、意味が明確になる

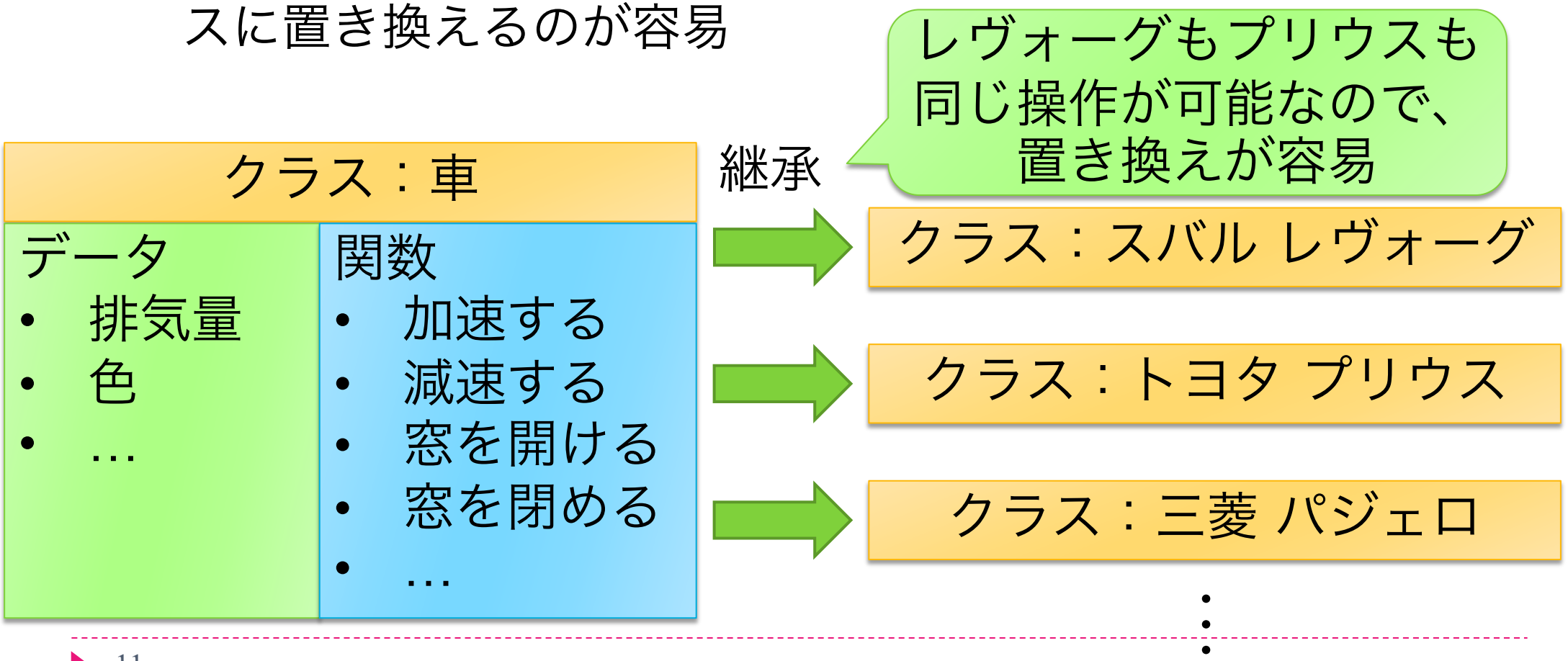
クラスを使う利点２

大規模なプログラムを
書くときに便利

- データの隠蔽（カプセル化）
  - オブジェクト内のデータに直接アクセスできなくすることができる
    →予期せぬバグを防ぐことができる
    →デバッグ時の問題の切り分けが容易
    →複数人での分業が容易

クラスを使う利点３

- 過去のソースコードの再利用可能性が向上
  - 例：特定のクラスだけを別のプログラムに再利用
  - 「継承」という仕組みを使えば、同じ性質を持つ別のクラスに置き換えるのが容易



残高照会プログラム（クラスを使わない場合）

ex7_wo_class.cpp

```
#include <string> // stringを使うために必要
#include <iostream> // 入出力に必要
using namespace std; // お約束

int main() {
  string suzuki_name = "鈴木龍一"; // 鈴木さんの名前
  int suzuki_balance = 123000; // 鈴木さんの残高
  string tanaka_name = "田中恵美"; // 田中さんの名前
  int tanaka_balance = 256000; // 田中さんの残高

  suzuki_balance += 10000; // 鈴木さんの残高を10000円増やす
  tanaka_balance -= 2000; // 田中さんの残高を2000円減らす

  cout << suzuki_name << “様の残高は" << suzuki_balance << "円です．" << endl;
  cout << tanaka_name << "様の残高は" << tanaka_balance << "円です．" << endl;

  return 0;
}
```

残高照会プログラム（クラスを使う場合）

ex8_w_class.cpp

```
#include <string>
#include <iostream>
using namespace std;

class Account {
public:
  string name; // 名前
  int balance; // 残高
};

int main() {
  Account suzuki; // 鈴木さんの口座のオブジェクト
  Account tanaka; // 田中さんの口座のオブジェクト
  suzuki.name = "鈴木龍一"; // 鈴木さんの名前
  suzuki.balance = 123000; // 鈴木さんの残高
  tanaka.name = "田中恵美"; // 田中さんの名前
  tanaka.balance = 256000; // 田中さんの残高

  suzuki.balance += 10000; // 鈴木さんの残高を10000円増やす
  tanaka.balance -= 2000; // 田中さんの残高を2000円減らす

  cout << suzuki.name << "様の残高は" << suzuki.balance << "円です。" << endl;
  cout << tanaka.name << "様の残高は" << tanaka.balance << "円です。" << endl;

  return 0;
}
```

クラスの宣言

実は構造体とほぼ同じ書式

- クラス名
  - ここでは銀行口座を意味する「Account」
- データメンバ（クラスに属する変数）
  - ここでは、以下の2つ
    - string型のname
    - int型のbalance

クラスの宣言は”class”から始まる

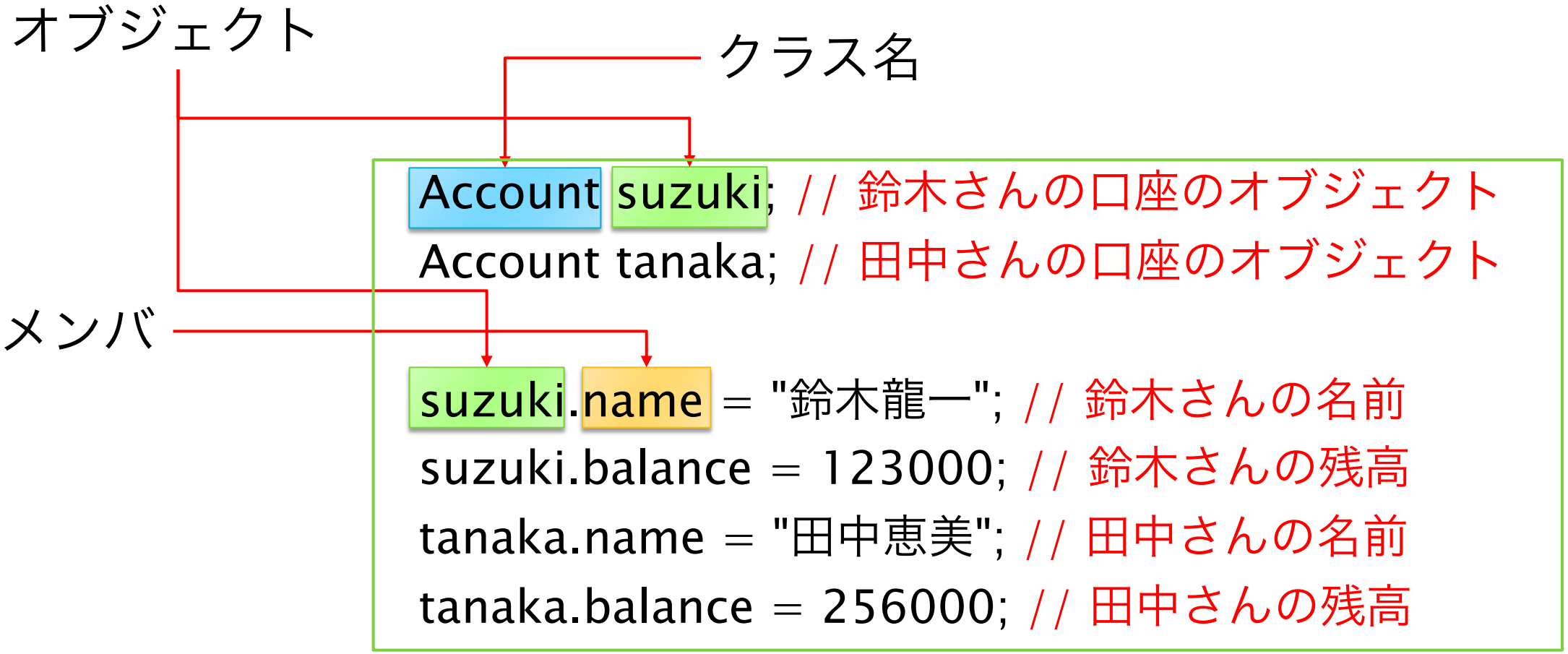
ひとまずお約束

```
class Account {
public:
  string name; // 名前
  int balance; // 残高
};
```

セミコロンを忘れない！

インスタンスとも言う

オブジェクトの宣言とデータメンバの操作



参考：classって実は構造体？

- classをstructに変えても
  同じ事ができる
  （試してみよう）

```
struct Account {
public:
  string name; // 名前
  int balance; // 残高
};
```

- C言語では、構造体を宣言するとき、
  struct Account suzuki;
  のようにした
- C++では、structを省略して、
  Account suzuki;
  とできる

残高照会プログラム（メンバ関数を使う場合）

ex9_w_class2.cpp

```
#include <string>
#include <iostream>
using namespace std;

class Account {
private:
  string name; // 名前
  int balance; // 残高

public:
  // コンストラクタ
  Account(string _name, int _balance) {
    name = _name; // 名前を初期化
    balance = _balance; // 残高を初期化
  }
  // 名前を調べる
  string get_name() {
    return name;
  }
  // 残高を調べる
  int get_balance() {
    return balance;
  }
  // 預ける
  void deposit(int amnt) {
    balance += amnt;
  }
  // おろす
  void withdraw(int amnt) {
    balance -= amnt;
  }
};
```

残高照会プログラム（メンバ関数を使う場合）

続き

ex9_w_class2.cpp

```
int main() {
  Account suzuki(“鈴木龍一”, 123000); // 鈴木さんの口座のオブジェクト作成
  Account tanaka("田中恵美", 256000); // 田中さんの口座のオブジェクト作成

  suzuki.deposit(10000); // 鈴木さんの残高を10000円増やす
  tanaka.withdraw(2000); // 田中さんの残高を2000円減らす

  cout << suzuki.get_name() << "様の残高は"
       << suzuki.get_balance() << "円です．" << endl;
  cout << tanaka.get_name() << "様の残高は"
       << tanaka.get_balance() << "円です．" << endl;

  return 0;
}
```

クラス・その２

- クラスはデータメンバだけでなく、
  メンバ関数も持てる

- クラスの宣言again
  - メンバ関数
    （クラスに属する関数）
    - ここでは、以下の4つ
      - get_name
      - get_balance
      - deposit
      - withdraw

```
 // 名前を調べる
  string get_name() {
    return name;
  }
 // 残高を調べる
  int get_balance() {
    return balance;
  }
  // 預ける
  void deposit(int amnt) {
    balance += amnt;
  }
  // おろす
  void withdraw(int amnt) {
    balance -= amnt;
  }
};
```

クラス・その２（続き）

- データメンバ、メンバ関数へのアクセス制御

これ以降に宣言されるものは
クラス外からもアクセス可能

```
class Account {
public:
  string name; // 名前
  int balance; // 残高
};
```

これ以降に宣言されるものは
クラス外からアクセス不可能

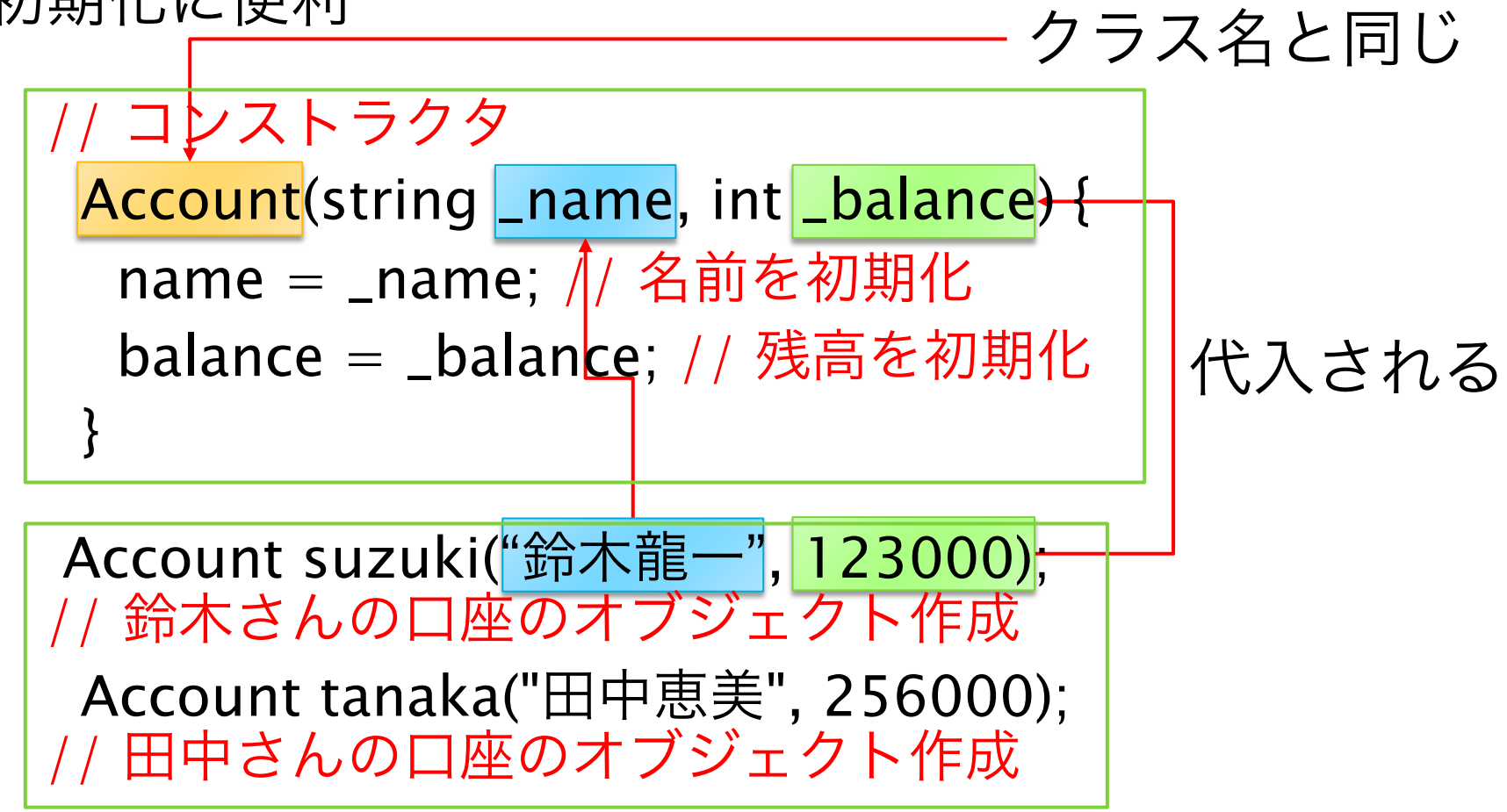
```
class Account {
private:
  string name; // 名前
  int balance; // 残高
};
```

試しに、ex8_w_class.cppのpublicを
privateにしてみると、エラーになる

第12回でも勉強します

クラス・その２（さらに続き）

- コンストラクタ
  - オブジェクト作成時に呼ばれる関数
  - 変数の初期化に便利



ファイル入出力

C++でのファイルの入出力

- ファイルの入出力を司るクラスがある
  - 入力：ifstream
  - 出力：ofstream

- よくある使い方
  - オブジェクトを作るときにファイルを開く

```
// 入力ファイルを開く
ifstream fin("ex10_input.txt");
```

  - オブジェクトを破棄するとファイルを閉じる
    - オブジェクトはスコープの範囲外に出ると破棄される
      （サンプルファイルでは，「 exit(0); 」で定義．）

ファイルの入出力のオプション

- ファイルを開くときにオプションを指定できる

ios::in	読み取り用にファイルを開く (ifstreamのデフォルト値)
ios::out	書き込み用に新規ファイルを開き、 既存ファイルは上書きする (ofstreamのデフォルト値)
ios::trunc	書き込み用に新規ファイルを開き、 既存ファイルは上書きする (ofstreamの場合、ios::outと等価)
ios::app	書き込み用に既存ファイルを開き、 ファイルの最後に追記する （ファイルが存在しなければ作成する）
ios::ate	ファイルを開いて、ファイルの末尾に移動
ios::binary	バイナリモード （改行コードの変換を行わない）

ファイルの入出力のメンバ関数

- open: ファイルを開く
- close: ファイルを閉じる
- eof：ファイルの終端に達したらtrueを返す
- seekg: [入力ストリーム用] ファイルの読み取り位置を変更する
- seekp: [出力ストリーム用] ファイルの書き込み位置を変更する
- get, readなど: データの読み込み
- put, writeなど: データの書き込み

参考：https://ja.cppreference.com/w/cpp/io

ファイル入出力の例（1/3）

ex10_file_io.cpp

```
#include <iostream>
#include <fstream>          // ファイル入出力に使う
#include <stdlib.h>         // exitのため

using namespace std;
```

ファイル入出力の例（2/3）

ex10_file_io.cpp

```
int main() {
  // 入力ファイルを開く
  ifstream fin("ex10_input.txt");
  if (!fin) {       // ファイルが開けたかどうかの確認
    cout << "Cannot open input file." << endl;
    exit(0);
  }

  // 出力ファイルを開く(追記モード)
  ofstream fout("ex10_output.txt", ios::app);
  if (!fout) {     // ファイルが開けたかどうかの確認
    cout << "Cannot open output file." << endl;
    exit(0);
  }
```

finはクラスifstreamのオブジェクト

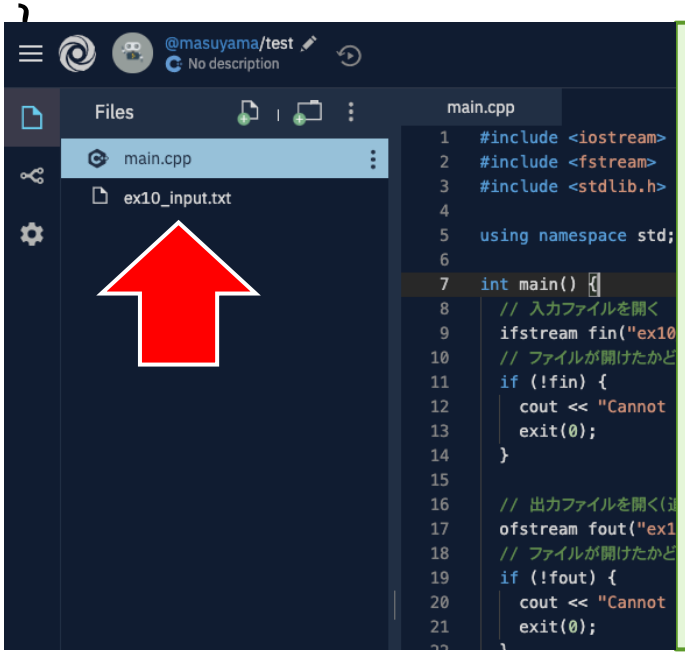
foutはクラスofstreamのオブジェクト

ファイル入出力の例（2/3）

ex10_file_io.cpp

```
int main() {
  // 入力ファイルを開く
  ifstream fin("ex10_input.txt");
  if (!fin) {       // ファイルが開けたかどうかの確認
    cout << "Cannot open input file." << endl;
    exit(0);
  }
```

finはクラスifstreamのオブジェクト



Repl.itのFilesに「ex10_input.txt」をドラッグ＆ドロップする.
上記ファイルが無いと,
"Cannot open input file."
となる.

ファイル入出力の例（3/3）

ex10_file_io.cpp

```
  // 入力ファイルの最後まで順番に値を読み込み、
  // それぞれ10を足して出力ファイルに追記する
  int val;
  // ファイルの最後までのループ
  while ( !fin.eof() ) {
    fin >> val;
    fout << val+10 << " ";
  }
  fout << endl;

  return 0;
}
```

eof()：ファイルの終端に達したらtrueを返す関数