構造体からクラスへ

オブジェクトとは？

- 関数とは，あるデータに対する特定の処理単位
  - 関数とデータは相互に密接な関係がある
- 関数とデータを併せてパッケージ化したものがクラス
  - データのみをパッケージ化したのが構造体
  - 処理機能（メンバ関数）と，処理される対象（メンバ変数）をひとまとめにすると，プログラムを理解しやすくなる
- 生成されたクラスの実体（オブジェクト）に処理を加える事によるプログラム記述が，オブジェクト指向プログラミング
  - 実体としてのオブジェクトは、インスタンスとも呼ばれる

クラス宣言（まずはメンバ変数から）

```
struct Health {
    char  *name; /* 対象者の氏名 */
    float  height, weight; /* 身長・体重 */
    float  BMI; /* 計算した肥満度 */
};
```

```
class Health {
public: // 以下のメンバを公開
    char  *name; // メンバ変数
    float  height, weight;
    // 肥満度は省略（関数で値を返す）
};
```

「//」はコメント行を表す（// 以降の文字列はその行だけ読み飛ばされる）

クラス宣言（メンバ関数を加える）

```
class Health {
public:
    char  *name; // メンバ変数（氏名）
    float  height, weight; // メンバ変数（身長，体重）

    float getBMI () { // BMI 値を計算して返す
        return weight / (height * height);
    }
};
```

初期化（構築子）の宣言

```
class Health {
public:
    char  *name;
    float  height, weight;
    Health () { // 引数無しの構築子. 戻り値の型宣言は不要
        name = null; height = weight = 0.0;
    }
    Health (char *n, float h, float w) { // 引数有り
        name = n; height = h; weight = w;
    }
    float    getBMI () {...
```

引数の与え方によって，対応する構築子が自動的に選択される

クラスの生成と使い方

実体（オブジェクト）生成後にメンバ変数を指定する場合

```
Health health; // 引数無しの構築子が呼ばれる
health.name = “taro”;
health.height = 1.7; health.weight = 50;
printf (“BMI = %f¥n”, health.getBMI());
```

実体（オブジェクト）生成時にメンバ変数を指定する場合

```
Health health (“taro”, 1.7, 50);
printf (“BMI = %f¥n”, health.getBMI());
```

new 演算子を用いてオブジェクトの領域を動的に確保する場合

```
Health *health = new Health (“taro”,1.7, 50);
printf (“BMI = %f¥n”, health->getBMI());
```

クラス宣言（2次元座標上の点）

```
class Point {
public: // 以下のメンバを公開
    int x, y; // メンバ変数
    Point () { x = y = 0; } // 引数無し構築子
    Point (int _x, int _y) { // 引数有り
        x = _x; y = _y:
    }
    Point* median (Point p) { // 中点を生成して返す
        return new Point ((x+p.x) / 2, (y+p.y) / 2);
    }
};
```

クラスの生成と使い方

実体（オブジェクト）生成後にメンバ変数を指定する場合

```
Point p0, p1; // 引数無し構築子
p0.x = 1; p0.y = 2; p1.x = 3; p1.y = 6;
Point *cp = p0.median (p1);
printf (“median = %d, %d¥n”, cp->x, cp->y);
```

実体（オブジェクト）を動的に生成する場合

```
Point *p0 = new Point (1, 2);
Point *p1 = new Point (3, 6);
Point *cp = p0->median (*p1);
// 2, 4 が出力される
printf (“median = %d, %d¥n”, cp->x, cp->y);
```

メンバ変数を隠す事ができる

```
class Health {
private:
    char  *name;
    float  height, weight;

public:
    Health (char *n, float h, float w); {
        name = n; height = h; weight = w;
    }
    float getBMI () { // BMI 値を計算して返す
        return weight / (height * height);
    }
};
```

この宣言以降のメンバは全て参照できなくなる

この宣言以降のメンバは全て参照可能となる

private宣言の結果．．．

Health health (); // 引数無しの構築子が呼ばれる

× health.name = “taro”;

× health.height = 1.7;

× health.weight = 50;

// これだけが実行可能

◯ printf (“BMI = %f¥n”, health.getBMI());

メンバ変数を隠す理由

- メンバ変数はオブジェクトの内部状態を表し，他のクラスからその値を更新するには，メンバ関数を用いる様にする
- 内部状態に対して，値の参照と更新の許可を設定する
  - 参照を許可する場合：get変数名()のメンバ関数を作成
  - 更新を許可する場合：set変数名()のメンバ関数を作成
- 読み書きの際に，補足的な前/後処理を追加できる
  - 参照時：メンバ変数に無い値の即時計算（具体例：BMI）
  - 更新時：不正な値のチェックとデフォルト値への置き換え

更新用関数の導入

```
class Health {
private:
    char  *name;
    float  height, weight;
public:
    void setHeight (float h) {
        if (h > 0. && h < 3.) // 3m以上の人はいない
            height = h;
    }
    void setWeight (float w) {
        if (w > 0. && w < 300.) // 300kg以上は除外
            weight = w;
    }
};
```

参照用関数の導入

```
class Health {
private:
    char  *name;
    float  height, weight;
public:
    char* getName () {
        return name;
    }
    float getHeight () {
        return height;
    }
    // 体重の値は，外部からは参照させない…
};
```

ヘッダファイルとソースファイルの分離

クラス名.h と命名されたファイル（Point.h）に保存する

```
class Point {
public:
      int x, y;
      Point ();
      Point (int _x, int _y);
      Point* median (Point p);
};
```

メンバ関数は宣言のみを記述し，
{ と } で囲まれる本体部は省略する

クラス名.cpp と命名されたファイル（Point.cpp）に保存する

```
Point::Point () { x = y = 0; }
Point::Point (int _x, int _y) {
          x = _x; y = _y:
}
Point* Point::median (Point p) {
          return new Point ((x+p.x) / 2, (y+p.y) / 2);
}
```

関数名の前に「クラス名::」の
prefix を記述する（例 Point::）

まとめ

- クラスとは，構造体に専用関数を加えたもの
  - 同名の関数でも，引数が異なると別の関数とみなされる
  - ヘッダファイル（.h）に入出力定義を記述し，本体部分はソースファイル（.cpp）に記述する
  - ヘッダファイルで使用方法を公開し，ソースファイルで内部を隠蔽
- クラスから，その実体を生成するには構築子を用いる
  - 構築子はクラスと同じ名前の関数（通常は公開する）
  - 引数を指定できる（省略もできる）
  - 領域の動的確保には new 演算子（malloc の代用品）
- クラスの実体に対する処理単位がメンバ関数
  - メンバ変数は内部状態を保持するためのもの
  - メンバ変数は他のクラスから隠すのが安全（private宣言）
  - メンバ変数の操作用関数を適宜用意する（getter, setter）