

構造体とクラス

ゲームプログラミングA #02

向井 智彦

本日の目標

- 構造体の利用法・作成法の習得
- クラスの利用法・作成法の習得

先週のおさらい

- 4つの演習課題と2つの発展課題
 - 解答例を 01/example フォルダに公開
- コンパイル失敗や, 出力結果が異なる提出物については個別に連絡済

構造体

基本型(おさらい)

- char, int, short, long
- unsigned char, unsigned int, unsigned short, unsigned long
- float, double
- bool

配列（おさらい）

- 同じ基本型をまとめて管理
 - `int a0, a1, a2, a3, ..., a9;` → `int a[10];`
 - `float b0, b1, b2, b3, b4;` → `float b[5];`
- 添え字を通じてアクセス
 - `char carray[20];` → `carray[0] ~ carray[19]`
 - `int iarray[10];` → `iarray[0] ~ iarray[9]`
 - `double darray[5]` → `darray[0] ~ darray[4]`

構造体:=複合型

基本型が異なるデータのまとめ

1. int a;

2. float b;

3. double c;

4. char d;

1. struct data

2. {

3. int a;

4. float b;

5. double c;

6. char d;

7. };

※文法は後ほど説明

現実な例

- 数学
 - 2次元座標, 3次元座標, 複素数, ベクトル?
- 個人データ
 - 氏名, 年齢, 生年月日, 住所, マイナンバー...
- 家族構成
 - 父, 母, 子, 孫...
- 地理データ
 - 都市名, 人口, 面積, 標高

Quick exercise 1

- 日野キャンパスを表す「数字」を複数挙げて下さい
 - － 例) 学生数

Quick exercise 2

- 先ほど挙げた「日野キャンパスを表す『数字』」から5つほど選択し、それらをC言語の変数で表すために、
それぞれに適した型と
わかりやすい変数名を考えて下さい.

Quick exercise 3

作成した5つの変数を，空欄に記入して下さい．

```
struct Campus7Number
```

```
{
```

```
};
```

構造体の文法

struct 構造体タグ名

{

変数1の型 変数1の名前;

変数2の型 変数2の名前;

...

変数nの型 変数nの名前;

};

構造体の使い方(C++)

基本型の変数宣言

- `int a;`
- `float b;`
- `char c;`

配列型の変数宣言

- `int a[10];`
- `float b[10];`
- `char c[10];`

構造体の変数宣言

- `PlayerEntity player;`
- `EnemyEntity enemy;`
- `Weapon weapon;`
- `Campus7Number hino;`

構造体タグ名 + 変数名

本日用いる構造体

```
struct Vector2 // 2次元ベクトル
{
    double x;
    double y;
};
```

X座標とY座標へのアクセス

1. `Vector2 v;` // Vector2変数vの宣言

2. `v.x = 10;`

// v + ピリオド + 要素名 x

3. `v.y = 20;`

// v + ピリオド + 要素名 y

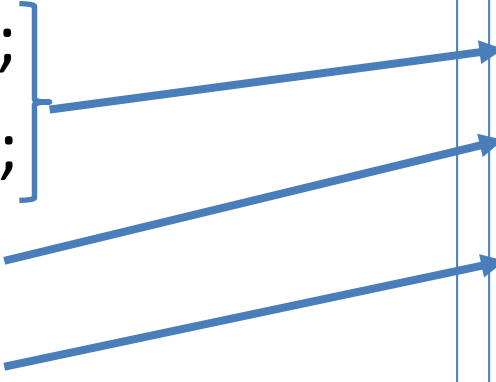
4. `printf(“%f, %f¥n”, v.x, v.y);`

```
struct Vector2
{
    double x;
    double y;
};
```

同じ働きをするプログラム1

```
1. double x;  
2. double y;  
3. x = 10.0;  
4. y = 20.0;  
5. printf("%f, %f¥n",x,y);
```

```
1. Vector2 v;  
2. v.x = 10.0;  
3. v.y = 20.0;  
4. printf("%f, %f¥n", v.x,  
           v.y);
```



構造体できること・できないこと

できること

- 構造体の代入
 1. `struct Data { int a; int b;};`
 2. `Data x, y;`
 3. `y.a = 0; y.b = 0;`
 4. `x = y;`
- 配列を含む構造体の代入
 1. `struct Data { int a[10]; };`
 2. `Data x, y;`
 3. `for (int i=0; i<10; y.a[i++]=1);`
 4. `x = y;`

できないこと

- 構造体同士の比較, 演算
 - `x != y`
 - `x == y`
 - `x = y * 2;`
- (※演算子オーバーライド
することで可能に)

構造体の配列

- `Vector2 array[100];`
 - Vector2構造体が, 100個並んだデータ
 - 添え字＋ピリオド＋要素名 を通じて各構造体の要素にアクセス
 - `array[0].x`
 - `array[10].y`
 - `array[22].x`
 - `array[99].y` など

同じ働きをするプログラム2

```
1. int x0, y0, x1, y1;  
2. x0 = 0;  
3. y0 = 1;  
4. x1 = 2;  
5. y1 = 3;
```

```
1. Vector2 p0, p1;  
2. p0.x = 0;  
3. p0.y = 1;  
4. p1.x = 2;  
5. p1.y = 3;
```

```
1. Vector2 p[2];  
2. p[0].x = 0;  
3. p[0].y = 1;  
4. p[1].x = 2;  
5. p[1].y = 3;
```

メリット:

意味単位にまとまる
変数名が少ない

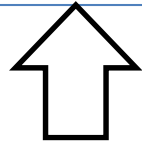
関数の基本形と構造体

関数の基本形

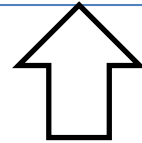
戻り値の型名 `main(void) { return 0; }`



戻り値の型名 関数名 (引数リスト) { 処理; }



Vector2



Vector2 a

基本型を引数とする関数

例：ベクトルの長さの計算

```
1. #include <stdio.h>
2. #include <math.h>
3.
4. double CalcDistance(
        double x, double y)
5. {
6.     return sqrt(x * x
7.                 + y * y);
8. }
```

```
9. int main(void)
10. {
11.     double x = 10.0;
12.     double y = 5.0;
13.     double len =
14.         CalcDistance(x, y);
15.     printf("%f¥n", len);
16.     return 0;
17. }
```

構造体を引数とする関数

例：原点からの距離の計算

```
1. #include <stdio.h>
2. #include <math.h>
3.
4. double CalcDistance(
           Vector2 p)
5.
6. {
7.     return ???;
8. }
```

```
9. int main(void)
10. {
11.     Vector2 p;
12.     p.x = 10;
13.     p.y = 5;
14.     double len =
           CalcDistance(p);
15.     printf("%f¥n", len);
16.     return 0;
17. }
```

ミニ演習

- vector3.cpp: 3次元ベクトル構造体の作成
 - TODO 1 ~ 5 まで完成させてコミット & プルリク
- 演習時間20分程度
 - 間に合わなかった部分は自習時間に

クラス

クラス(C++)

- オブジェクト指向プログラミングの根幹
- 構造体と同じ
 - デフォルトのアクセス設定の違いのみ(後述)
 - 複数の変数や関数を要素とする複合体
 - 言い換えれば構造体も関数をメンバーにできる
- C++における構造体とクラスの機能は同一
 - 意図的な設計, 可読性の点で使い分け

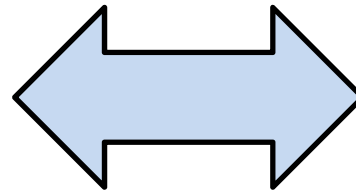
構造体とクラス

以降の説明は「クラス」についてのみ言及

```
struct Data1
{
    ...
};

struct Data2
{
    private:
        ...
};
```

同じ



```
class Data1
{
    public:
        ....
};

class Data2
{
    ....
};
```

メンバ関数

- 必須メンバ関数
 - コンストラクタ: 変数宣言時に自動呼出し
 - デストラクタ: 変数を使わなくなったら自動呼出し
- その他のメンバ関数
 - コピーコンストラクタ(次週)
 - 演算子オーバーライド(次週以降?)
 - 任意のメンバ関数

来週の予告

- 続・クラス
 - 引数付きコンストラクタ
 - コピーコンストラクタ
- カプセル化
 - アクセス指定子
 - constメンバ関数
- ポリモーフィズム
 - 引数が異なるメンバ関数
 - 演算子オーバーライド

コード例

- vector2class.cpp
 - 2次元ベクトルクラス
- vector3class.cpp
 - 3次元ベクトルクラス(演習課題)