２－１２：構造体

構造体とは、複数のデータを宣言・管理するための方法です。たとえデータの型が異なっていても、まとめて宣言することができます。

宣言するべき変数が多量になるとき、関連する複数の変数をまとめて、グループ分けすることなどに利用できます。

・使用法

1. 構造体の宣言(ヘッダーの class の上に書く)

```
struct 構造体の名前{
        メンバ(構造体内の変数のこと);
        メソッド(構造体内の関数のこと、メンバ関数とも);
};
```

＊1：構造体の宣言時、{}の最後に;(セミコロン)が必要になります。

構造体の宣言ではその構成を決定します。しかしこれではまだ変数は作成されていません。

2. 構造体変数の宣言(通常の変数と同じように書く)

```
struct 構造体の名前 構造体変数の名前;
```

＊2：ここでの struct は省略可。

(struct 構造体の名前)で変数の型(int など)と同じように考えます。

これで実際に使えるようになりました。

3. 実際の使用方法

```
構造体変数の名前.メンバ
構造体変数の名前.メソッド
```

＊3.(ドット)を挟む必要があります。これは Builder のコンポーネントに使われる->と意味的には同じです。

これで変数や関数を書いた場合と同じように扱われます。

では、実際に構造体を使ったプログラムを書いてみましょう。

1. 構造体の宣言(ヘッダーの class の上に書く)

```
struct Human{
        int Age;
        double Height;
        double Weight;
};
```

2. 構造体変数の宣言(ヘッダーの private に書く)

```
struct Human Student;
```

3. 実際の使用方法

```
Student.Age = 18;
Label1->Caption = Student.Height;
if(Student.Weight ==…
```

構造体の概念は、パソコンのフォルダのようなとらえ方をすると分かりやすいでしょう。基本的に大きなまとまりから順に表記していきます。

構造体のメンバとして配列を組み込むこともできます。(やり方は省略します)また、構造体そのものを配列にすることもできます。

```
struct Human Student[10];
```

宣言や、使用方法は普通の型で配列を宣言した場合と同じです。

```
Student[0].Age  Student[1].Height…
```

また構造体の中に構造体を入れることもできます。

```
struct School{
        struct Human Student;
};
```