データ構造入門及び演習
5回目：構造体

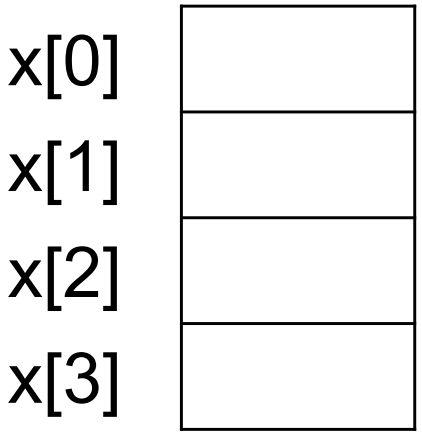
2014/05/16

担当：見越　大樹
61号館304号室

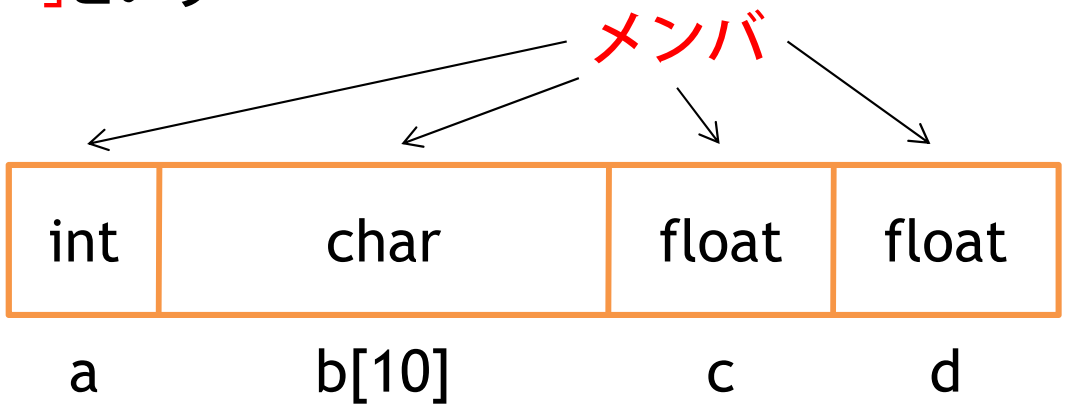
構造体 (Structure)

- 配列とは？
  - 同じ型のデータをまとめて格納するもの
  - 宣言方法：int　x[4];

整数型　名前　サイズ



- 構造体とは？
  - 「異なる型」の変数や配列をまとめて格納するもの
  - 要素のひとつひとつを「メンバ」という



構造体の使用例

- 学生名簿の例：
  - 学生番号，氏名，身長，体重　…　データの型が異なる

学生番号	101	114	199	整数：int
氏名	阿部 一郎	鈴木 健二	渡辺 隆史	文字：char[]
身長(cm)	178.5	167.5	175.0	小数：float
体重(kg)	63.5	53.0	82.4	小数：float

構造体の宣言

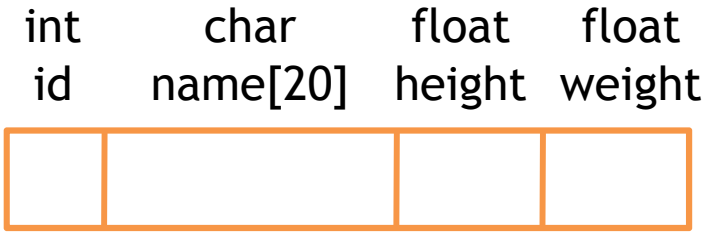
1. 構造体テンプレートの宣言
   - どのような型の変数や配列を1つにまとめるかを決定する

```
struct student {
    int id;
    char name[20];
    float height;
    float weight;
};
```

構造体テンプレート名(タグ)

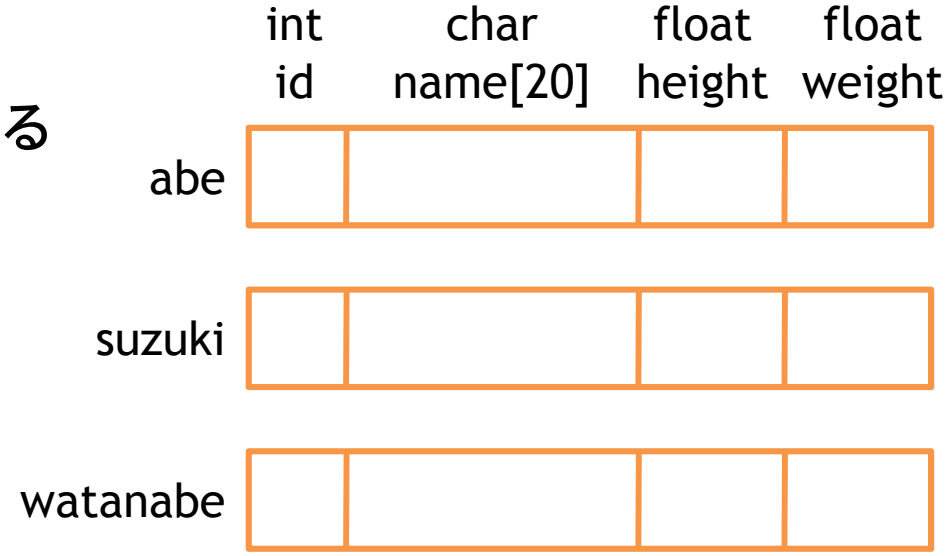
メンバ(構成要素)

← セミコロン



2. 構造体変数の宣言
   - テンプレートを持った変数を決定する

```
struct student abe;
struct student suzuki;
struct student watanabe;
```



構造体の初期化

```
// 宣言
struct student {
    int id;
    char name[20];
    float height;
    float weight;
};
```

	int id	char name[20]	float height	float weight
abe	101	阿部一朗	178.5	63.5
suzuki	114	鈴木健二	167.5	53.0
watanabe	199	渡辺隆史	175.0	82.4

```
// 初期化
struct student abe = { 101, “阿部一朗”, 178.5, 63.5 };
struct student suzuki = { 114, “鈴木健二”, 167.5, 53.0 };
struct student watanabe = { 199, “渡辺隆史”, 175.0, 82.4 };
```

構造体の宣言~配列を使用する場合~

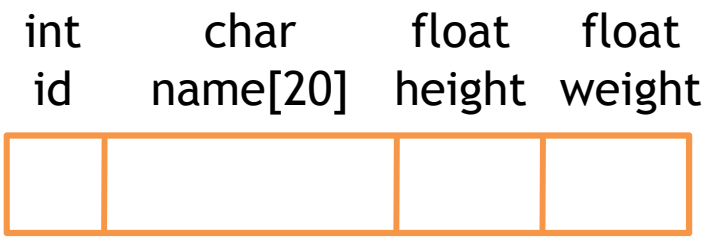
1. 構造体テンプレートの宣言
   - どのような型の変数や配列を1つにまとめるかを決定する

```
struct student {
    int id;
    char name[20];
    float height;
    float weight;
};
```

構造体テンプレート名(タグ)

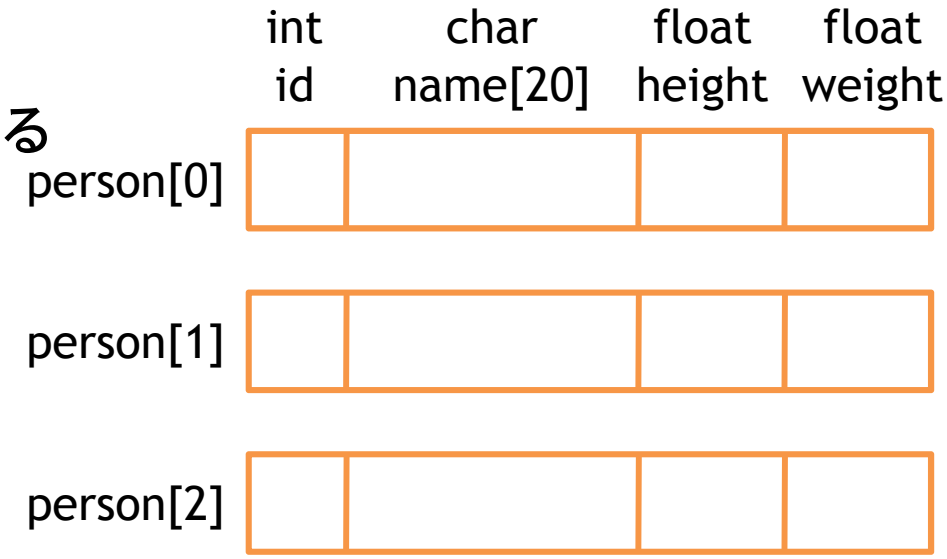
メンバ(構成要素)

← セミコロン



2. 構造体変数の宣言
   - テンプレートを持った変数を決定する

```
struct student person[3];
```



構造体の初期化 ~配列を使用する場合~

```
// 宣言
struct student {
    int id;
    char name[20];
    float height;
    float weight;
};

// 初期化
struct student person[] = {
    { 101, “阿部一朗”, 178.5, 63.5 },
    { 114, “鈴木健二”, 167.5, 53.0 },
    { 199, “渡辺隆史”, 175.0, 82.4 }
};
```

	int id	char name[20]	float height	float weight
person[0]	101	阿部一朗	178.5	63.5
person[1]	114	鈴木健二	167.5	53.0
person[2]	199	渡辺隆史	175.0	82.4

構造体メンバの参照法

- 構造体変数のメンバを参照するには，「. (ピリオド)」を使用する

```
printf( “学生番号：%d　氏名：%s　身長：%f　体重：%f¥n”,
                abe.id,  abe.name,  abe.height,  abe.weight );
```

構造体変数　↑　メンバ

ピリオド

- 構造体変数へ値を代入するときも同様

```
abe.height = 179.2;
abe.weight = 65.4;
strcpy( abe.name, “阿倍一郎”);
```

	int id	char name[20]	float height	float weight
abe	101	阿部一朗	178.5	63.5

サンプルプログラム

```
#include "stdio.h"

struct  student{
      int  id;
      char  name[20];
      float  height;
      float  weight;
};

void  main()
{
      struct  student  abe = { 101, “阿部一郎”, 178.5, 63.5 };
      printf("学生番号：%d  氏名：%s  身長：%.1f  体重：%.1f¥n",
            abe.id,  abe.name,  abe.height,  abe.weight);
}
```

メイン関数の外に書く

関数の引数として構造体を使う

```
#include <stdio.h>

struct student
{
    int  id;
    char  name[20];
    float  height;
    float  weight;
};
```

練習問題：
青字の部分を関数PrintStudentで表す・

```
int main(void)
{
    struct  student  abe =
        { 101, “阿部一朗”, 178.5, 63.5 };

    printf( “学生番号：%d¥n”,  abe.id );
    printf( “氏名：%s¥n”,  abe.name) ;
    printf( “身長：%f¥n”,  abe.height );
    printf( “体重：%f¥n”,  abe.weight );

    abe.height  +=  0.6;
    abe.weight  -=  2.0;

    printf( “学生番号：%d¥n”,  abe.id );
    printf( “氏名：%s¥n”,  abe.name );
    printf( “身長：%f¥n”,  abe.height );
    printf( “体重：%f¥n”,  abe.weight );

    return  0;
}
```

練習問題

BMIの計算式：
BMI = 体重(kg)/{身長(m)}^{2}

- BMI(体格指数)を計算・表示するプログラムを作成しなさい.

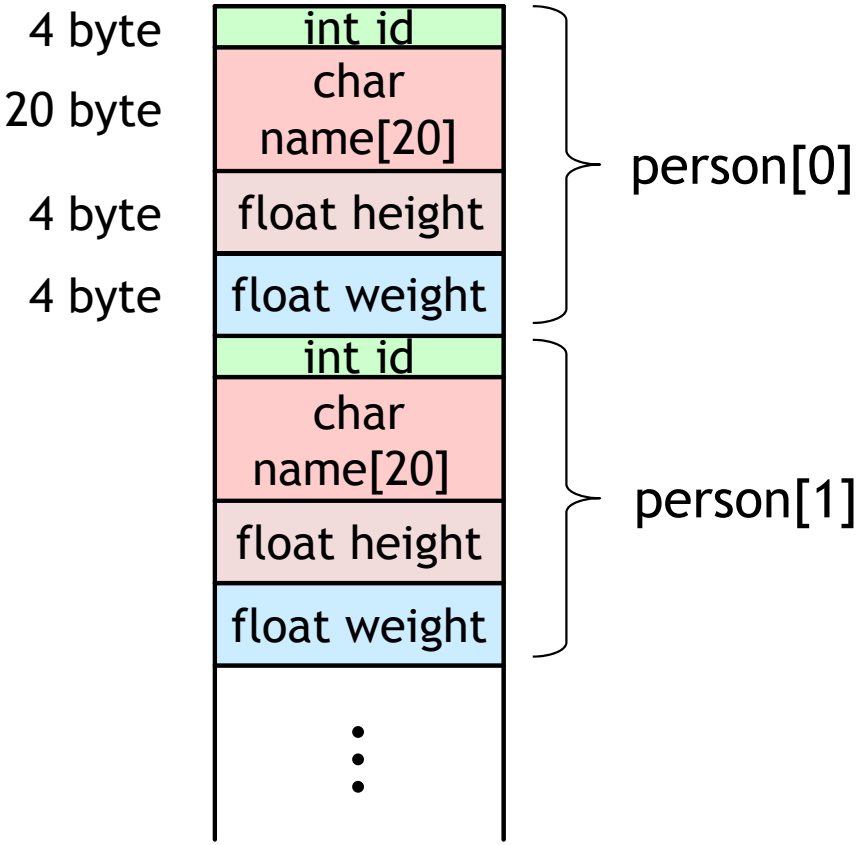
```
#include  <stdio.h>
#define  NUM  3
struct student {
    int  id;
    char  name[20];
    float  height;
    float  weight;
};

int main(void){
    int  i;
    double  bmi[3];  //各人のBMI
    struct  student  person[] = {
        { 101, “阿部一朗”, 178.5, 63.5 },
        { 114, “鈴木健二”, 167.5, 53.0 },
        { 199, “渡辺隆史”, 175.0, 82.4 },
    };
```

```
    // BMIの計算
    for (i=0;i<NUM;i++) {
        // ここにBMIを計算する
        // プログラムを記述する
        bmi[i]=

    }
    // 計算結果の表示
    for(i=0; i<NUM; i++) {
        // ここにBMIを表示する
        // プログラムを記述する
        printf(“

    }
    return 0;
}
```

構造体のメモリ上の構造

- 構造体変数のメンバは，異なったデータ型であっても，連続したアドレスに格納される

```
struct  student {
      int  id;
      char  name[20];
      float  height;
      float  weight;
};

struct  student  person[3];
```




```
int  a, b, c;
a = sizeof( person[0] );      →   a = 32
b = sizeof( person );         →   b = 96
c = sizeof( student );        →   c = 32
```

typedef

- typedef:　型の名前を定義する

struct student → typedef → MyStudent

```
struct student {
    int id;
    char name[20];
    float height;
    float weight;
};

struct student person[5];
```

```
typedef struct student {
    int id;
    char name[20];
    float height;
    float weight;
} MyStudent;

MyStudent person[5];
```

省略可

← 新しい型の名前

サンプルプログラム

```
#include <stdio.h>

typedef  struct  student {
    int  id;
    char  name[20];
    float  height;
    float  weight;
} MyStudent; //← 新しい型の名前


void  PrintStudent (MyStudent  p )
{
    printf( “学生番号：%d¥n”,  p.id );
    printf( “氏名：%s¥n”,  p.name) ;
    printf( “身長：%f¥n”,  p.height );
    printf( “体重：%f¥n”,  p.weight );
}
```

```
int main(void)
{
    MyStudent  person[] = {
        { 101, “阿部一朗”, 178.5, 63.5 },
        { 114, “鈴木健二”, 167.5, 53.0 },
        { 199, “渡辺隆史”, 175.0, 82.4 },
    };

    for(int i= 0; i < 4; i++){
        PrintStudent(perosn[i]);
    }

    return 0;
}
```

構造体を指し示すポインタ

- 基本的な考え方は変数に対するポインタと同じ
- 宣言する場合は，ポインタ名の前に「＊」をつける

```
//  構造体テンプレートの宣言
struct  student {
      int  id;
      char  name[20];
      float  height;
      float  weight;
};
//  ポインタの宣言
struct  student  *tmp;
```

- ポインタへのアドレス代入の方法：

```
struct  student  abe;
tmp = &abe;
```

ポインタを使った構造体の参照

- ポインタを使って構造体のメンバを参照するには，
「-> (アロー演算子)」を使う

```
printf( “学生番号：%d　氏名：%s　身長：%f　体重：%f¥n”,
        tmp->id,  tmp->name,  tmp->height,  tmp->weight );
```

- 以下のように書いても同じ意味

```
printf( “学生番号：%d　氏名：%s　身長：%f　体重：%f¥n”,
        (*tmp).id, (*tmp).name, (*tmp).height, (*tmp).weight );
```

「*tmpはポインタtmpが指し示す変数」

- 書き方が煩雑なので，通常はアロー演算子を使う

サンプルプログラム

```
#include "stdio.h"

struct  student{
      int  id;
      char  name[20];
      float  height;
      float  weight;
};

void  main()
{
      struct  student  abe = { 101, “阿部一郎”, 178.5f, 63.5f };
      struct  student  *tmp;
      tmp = &abe;
      printf("学生番号：%d  氏名：%s  身長：%.1f  体重：%.1f¥n",
            tmp->id,  tmp->name,  tmp->height,  tmp->weight);
      printf("学生番号：%d  氏名：%s  身長：%.1f  体重：%.1f¥n",
            (*tmp).id,  (*tmp).name,  (*tmp).height,  (*tmp).weight);
}
```

練習問題　２点間の距離を求めるプログラムを完成させよ

```
#include <stdio.h>
#include <math.h>

struct point {
    double x;
    double y;
};

double  kyori( struct  point  zahyo[2] )
{
    double z;
    z = sqrt(




    return z;
}
```

```
int  main( void )
{
    int  i;
    double  px, py, dis;
    struct  point  zahyo[2];

    for( i=0;i<2;i++ ){
        printf("¥n Input data %d(x,y):",i);
        scanf(“%lf %lf”, &px, &py);
        zahyo[i].x=px;
        zahyo[i].y=py;
    }
    dis = kyori( zahyo );
    printf("¥n Distance is %f¥n",dis);
    return 0;
}
```

2点(X1,Y1),(X2,Y2)間の距離

$=\sqrt{(X1-X2)^2+(Y1-Y2)^2}$