復習

関数のプロトタイプ宣言(3)

```
#include        <stdio.h>

float menseki(float a, float b);

int main
{
    float kekka;
    kekka = menseki(3, 4);
    printf("s=%f¥n", kekka);
}

float menseki(float a, float b)
{
    float s;
    s = a * b / 2;
    return s;
}
```

関数のプロトタイプ宣言

呼出し

定義

ソースプログラム上で，関数の呼び出し前に，関数の定義か，プロトタイプ宣言のどちらかが絶対必要

printf()関数やscanf()関数では不要？

↓

No！　絶対必要

printf()やscanf()のプロトタイプ宣言はstdio.hファイルに定義されている.

プロトタイプ宣言は関数定義の予告編

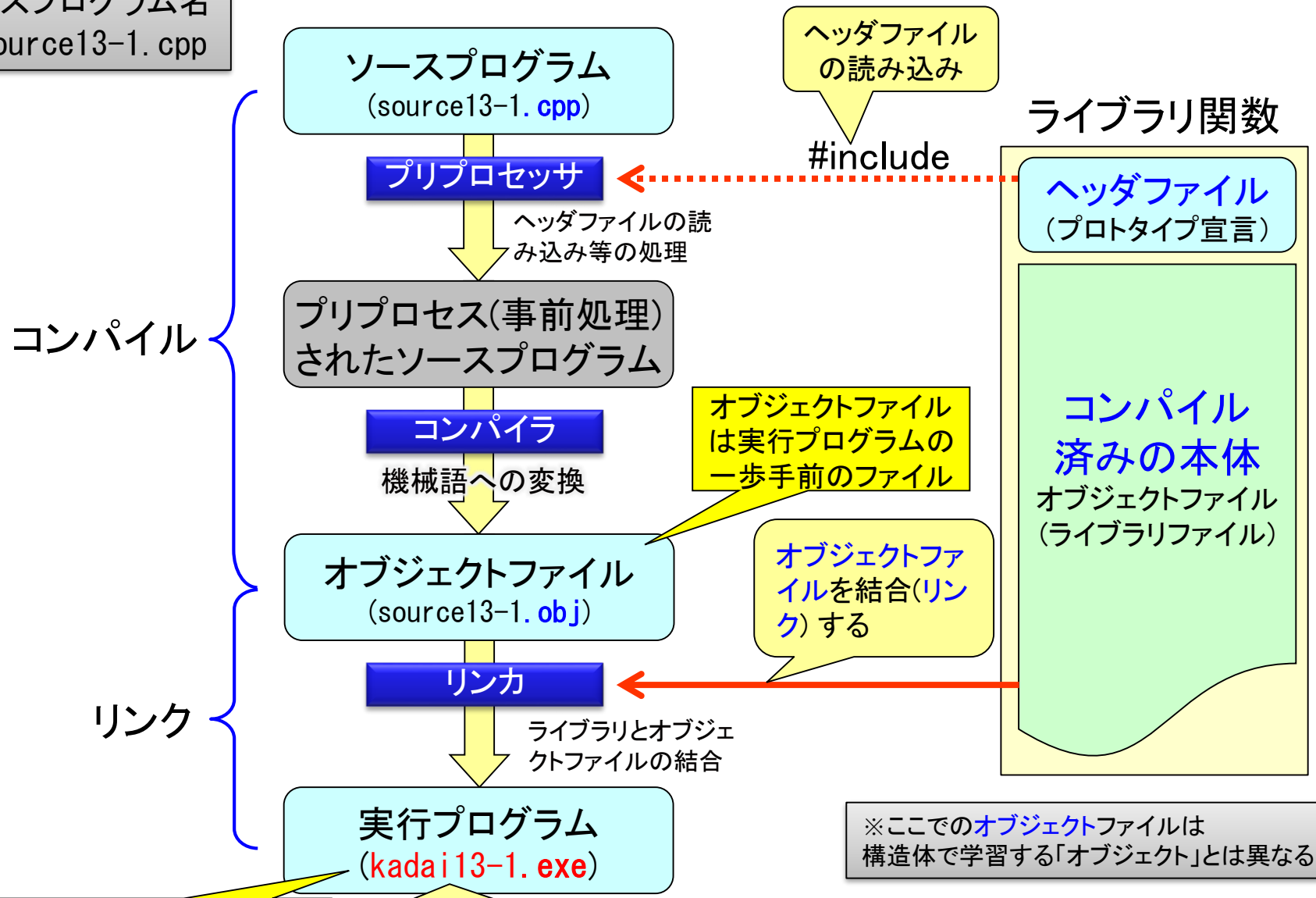
プロトタイプ宣言　＝　関数定義の1行目

プロトタイプ宣言は関数の使用方法(引数の数と型，戻り値の型)を示している

実行プログラムができるまで

復習

プロジェクト名
kadai13-1
ソースプログラム名
source13-1.cpp



※ここでのオブジェクトファイルは
構造体で学習する「オブジェクト」とは異なる

.exeファイルだけでプログラムが実行できる！

プロジェクトフォルダ内のDebugまたはReleaseフォルダにできる

コンパイル＋リンク＝ビルド と呼んでいる

プリプロセッサ命令

復習

プリプロセッサ命令は
#で始まる

```
#include    <標準ヘッダファイル名>
#include    "自作ヘッダファイル名"
#define     マクロ名 マクロ文字列
```

スペース(空白)またはタブ

→ ヘッダファイルを読み込む

→ マクロ名をマクロ文字列で置き換える

ソースファイル:kadai12.cpp

```
#include    "rei12.h"
#define     teisu    35
#define     moji     'B'+15
int main(void)
{
    int x,y;
    x = teisu + 22 + moji;
    func1(x);
    y = func2(teisu, moji);
}
```

プリプロセス

プリプロセッサの処理後

```
void func1(int a);
float func2(int a, float b);

int main(void)
{
    int x,y;
    x = 35 + 22 + 'B'+15;
    func1(x);
    y = func2(35, 'B'+15);
}
```

コンパイル

自作ヘッダファイル:rei12.h

```
void func1(int a);
float func2(int a, char b);
```

キャストによる変数型の変換

復習

```
#include    <stdio.h>
int main(void)
{
    int a = 2, b = 3;
    double x1, x2, y1, y2;

    x1 = 1 / 3;
    x2 = a / b;
    printf("x1 =%5.2f, x2 =%5.2f¥n", x1, x2);

    y1 = 1.0 / 3;
    y2 = (double) a / b;
    printf("y1 =%5.2f, y2 =%5.2f¥n", y1, y2);
}
```

```
x1 = 0.00, x2 = 0.00
y1 = 0.33, y2 = 0.67
続行するには何かキー. . .
```

整数同士の演算では少数以下は切り捨て
1/3 = 0.333… ⇒ 0

整数変数同士の演算でも少数以下切り捨て

浮動小数点と整数の演算では全て
浮動小数点として計算
1.0/3 = 1.0/3.0 ⇒ 0.333…

整数変数同士の演算だが,
浮動小数点で計算させたい

キャストによる型の変換

キャストの書き方
(変数型) 変数
変数を変数型に変換する

キャストを用いた四捨五入

復習

```
#include    <stdio.h>
int main(void)
{
    int c0, c1, c2, c3, c4;
    double x1, x2;

    x1 = 3.4999;
    c0 = x1;
    c1 = (int) x1;

    x2 = 3.5000;
    c2 = (int) x2;
    printf("c1 = %d, c2 = %d¥n", c1, c2);

    c3 = (int) (x1 + 0.5);
    c4 = (int) (x2 + 0.5);
    printf("c3 = %d, c4 = %d¥n", c3, c4);
}
```

```
c1 = 3, c2 = 3
c3 = 3, c4 = 4
続行するには何かキー. . .
```

warning C4244: '=' : 'double' から 'int' への変換です。データが失われる可能性があります。

キャストを用いているので警告が出ない！

キャストを用いた四捨五入
0.5を足して整数型に変換すると小数以下の四捨五入になる

3.4999 + 0.5 = 3.9999 ⇒ 3

3.5000 + 0.5 = 4.0000 ⇒ 4

プログラミング技術の発達

初期の時代

暗黒時代．モジュール化困難．
理解しやすいプログラム，再利用可能なプログラムができなかった．

初期のFORTRAN，初期のBASIC，etc…

構造化プログラミング

ループ制御，選択制御の整理．
ローカル変数の導入　→　モジュール化が容易．
理解しやすいプログラム，再利用しやすいプログラムができるようになった．

C言語，Pascal，etc…

オブジェクト指向プログラミング

一度作成したモジュールの徹底した再利用が可能．
プログラミングの考え方の整理

C++，Java，C#，Objective-C，Python

- 数値計算
- シミュレーション
- 機械制御(ロボット)

科目名
基礎プログラミング
応用プログラミング実習
コンピュータグラフィックス

現代のプログラミング技術

- グラフィカルユーザーインターフェース(GUI)
- アプリケーション(アプリ)
- 人工知能

科目名
プログラミングプロジェクト演習

オブジェクト指向

オブジェクトを中心としたプログラムの考え方
オブジェクト　＝　物，全ての事物・事象

オブジェクトの例

人間　人間を表わす要素
性別，名前，年齢，身長，・・・，etc.

機械　機械を表わす要素
名前，重さ，部品の数，値段

実験データ　実験データを表わす要素
データ値の個数，データの値

構造体　ー　オブジェクト指向の第一歩

いくつかの変数をまとめて新しい変数型を作る

例えば

複素数型

実部と虚部のfloat型の２変数からできている

日付型

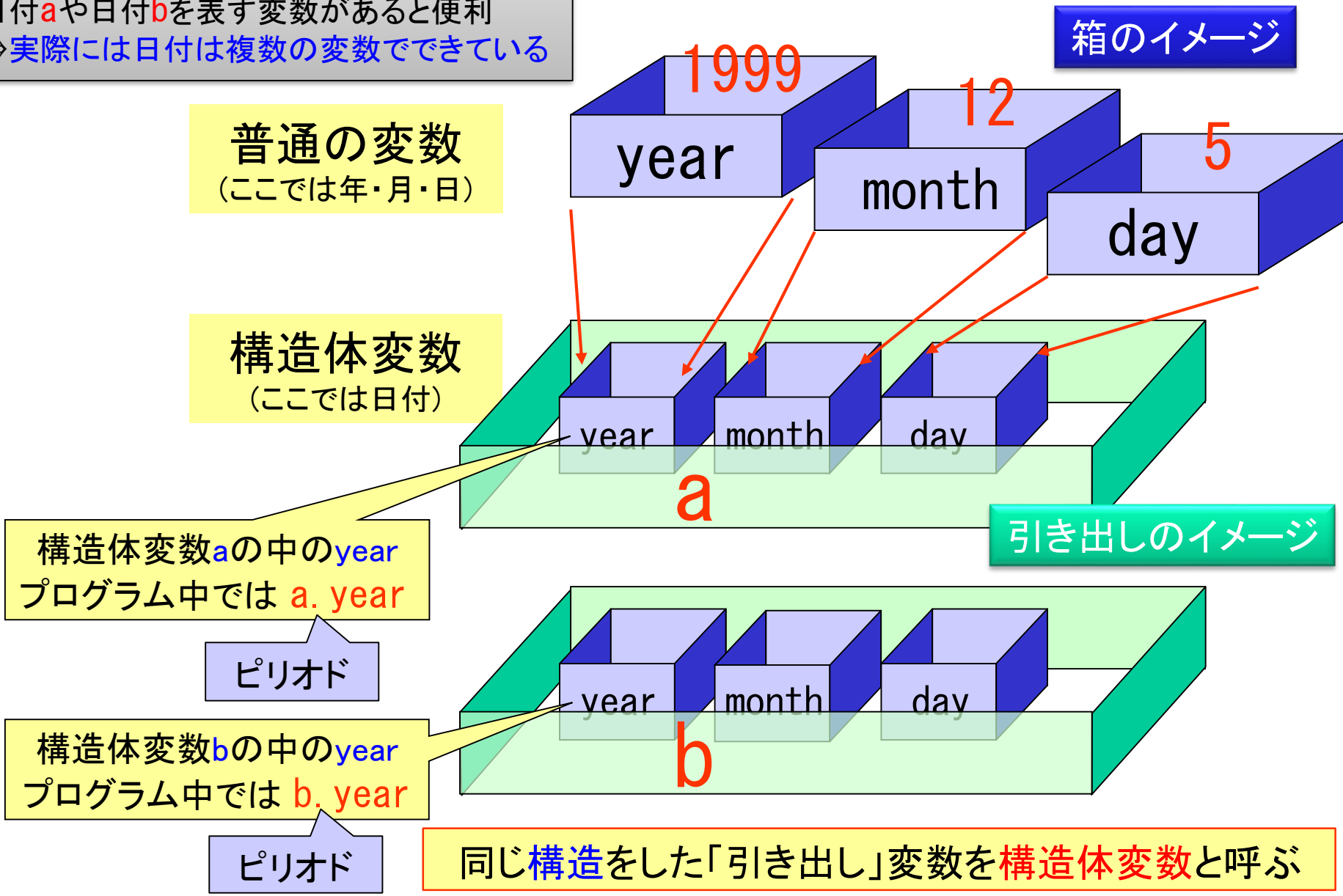
年，月，日を表すint型の３変数からできている

学生身体データ型

学籍番号(int型変数)
身長(float型変数)
体重(float型変数)
名前(文字列変数，char配列)
誕生日(日付型構造体)

構造体変数

例えば，課題が
「A君の誕生日をa，B君の誕生日をbとし，どちらが早く生まれたか比べなさい・・・」
日付aや日付bを表す変数があると便利
⇒実際には日付は複数の変数でできている



同じ構造をした「引き出し」変数を構造体変数と呼ぶ

構造体のプログラム例(1) ― 日付型

キーワード
struct

構造体内の変数をメンバ変数と呼ぶ

Date型の構造体の定義 → Dateは変数名ではなく設計図名

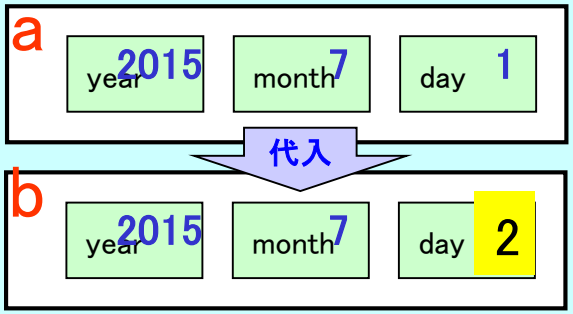
```
#include <stdio.h>

struct  Date
{
    int year;
    int month;
    int day;
};

void main()
{
    Date a, b;
    a.year = 2015;  a.month = 7;   a.day = 1;
    b = a;
    b.day = 2;
    printf("初めの日付は%d年%d月%d日, ", a.year, a.month, a.day);
    printf("次の日付は%d年%d月%d日¥n", b.year, b.month, b.day);
}
```

Date型の構造体変数aとbの宣言

a.year は構造体変数aのメンバ変数yearを意味する

構造体変数aをbに代入

b.day は構造体変数bのメンバ変数dayを意味する



初めの日付は2015年7月1日, 次の日付は2015年7月2日

構造体のプログラム例(2)

```
struct  Date
{
    int year;
    int month;
    int day;
};

void main()
{
    Date a = {1998, 6, 28};
    Date b;
    b = {1998, 6, 29};
    Date c[20];
    c[15].year = 1998;
    c[15].month = 6;
    c[15].day = 28;

    printf("今日は%d年%d月%d日¥n", c[15].year, c[15].month, c[15].day);
}
```

構造体変数を宣言すると同時に初期化

同時ではないのでダメ
エラー

Date型構造体の配列の宣言

Date型構造体配列のインデックス15の要素に値を代入

※この例では配列の添え字15には特に意味はない

構造体のプログラム例(4)

```
#include <stdio.h>

struct  Date
{
  int year;
  int month;
  int day;
};

struct Student
{
  int number;  //学籍番号
  char name[20]; //名前
  float height;  //身長
  float weight;  //体重
  Date birthday; //誕生日
};
```

Student型構造体のメンバとして文字列を指定

Student型構造体のメンバとしてDate型構造体変数を指定

Student型構造体sを仮引数とする関数の定義

引き出しの内容(情報)を丸ごと引数として受け取るイメージ

```
void printSt(Student s)
{
  printf("%d番%s(%d年%d月%d日生)",
  s.number,  s.name, s.birthday.year,
  s.birthday.month,  s.birthday.day);
  return;
}

void main()
{ Student matsu = {1020, "matsushima",
                   163.5, 62.0,
                   {1970, 2, 28}};

  printSt(matsu);
  printf("¥n");
}
```

Student型構造体の変数sの内容を画面表示

代入(コピー)

注意!
s.number ○
number ×

Student型構造体変数matsuを実引数とした関数呼び出し

1020番matsushima(1970年2月28日生)

構造体のプログラム例(5)

```
#include <stdio.h>

struct  Date
{
  int year;
  int month;
  int day;
};

struct Student
{
  int number;  //学籍番号
  char name[20]; //名前
  float height;  //身長
  float weight;  //体重
  Date birthday; //誕生日
};

void printSt(Student s);
```

printSt()関数の定義はこの例では省略しているが実際は必要

```
void check(Student a)
{
  printSt(a);
  if (a.weight + 110 > a.height)
  {  printf("は太りすぎだ！¥n");  }
  else
  {  printf("はいい感じだ！¥n");  }
  return;
}

void main()
{ Student  matsu = {1020, "matsushima",
                          163.5, 62.0,
                          {1970, 2, 28}};
  check(matsu);
}
```

注意！
a.weight ○
weight ×

1020番matsushima(1970年2月28日生)は太りすぎだ！

構造体と配列を組み合わせたプログラム例(1/2)

- 構造体変数の中に配列が含まれている場合(配列を含む構造体)
- 構造体変数が配列になっている場合(構造体変数の配列)

```
#include    <stdio.h>
struct BaitoKanri    //バイト管理構造体
{
    int id;          //バイトの個人番号
    char name[20];   //バイトの氏名(文字列)
    int money[12];   //1～12月に支払ったバイト料(1000円単位)
};

// バイト一人分の番号，氏名，支払額の表示する関数
void PrintPerson(BaitoKanri baito)
{
    int i;
    printf("%2d   %-10s", baito.id, baito.name);
    for (i = 0; i < 12; i++)
    {
        printf(" %2d", baito.money[i]);
    }
}

// バイト一人分の1年間の支払額を計算する関数
int Sum(BaitoKanri a)
{
    int i, sum = 0;
    for (i = 0; i < 12; i++)
    {
        sum = sum + a.money[i];
    }
    return sum;
}
```

配列を含む構造体の定義
この構造体にバイト一人あたりの情報をすべて詰め込んで管理する

この関数を使うと構造体変数baitoの(一人分の)情報を画面表示できる

%-10sは文字列を10文字左詰めで出力

この関数は，構造体変数aの(一人分の)1年間の給料総額を返す

```
番号 氏名         毎月の給料(単位:千円)                    合計
-----------------------------------------------------------
10   Matsushima 10 15 12 10 15 20 20 15 11 15 12 10 : 165
15   Abe        10 15 14 20 15 20 20 16 11 15 42 10 : 208
16   Trump      20 15 13 15 15 30 20 15 11 25 12 10 : 201
20   Putin      12 15 12 20 15 20 10 17 11 15 12 10 : 169
続行するには何かキーを押してください . . .
```

構造体と配列を組み合わせたプログラム例(2/2)

```
#include    <stdio.h>
struct BaitoKanri    //バイト管理構造体
{
    int id;          //バイトの個人番号
    char name[20];   //バイトの氏名(文字列)
    int money[12];   //1～12月に支払ったバイト料(単位1000円)
};
```

構造体変数の配列宣言

一人分の情報

```
int main(void)
{
    BaitoKanri list[4] = { {10, "Matsushima", {10, 15, 12, 10, 15, 20, 20, 15, 11, 15, 12, 10}},
                           {15, "Abe", {10, 15, 14, 20, 15, 20, 20, 16, 11, 15, 42, 10}},
                           {16, "Trump", {20, 15, 13, 15, 15, 30, 20, 15, 11, 25, 12, 10}},
                           {20, "Putin", {12, 15, 12, 20, 15, 20, 10, 17, 11, 15, 12, 10}},
                         };
    printf("番号 氏名        毎月の給料(単位:千円)                    合計¥n");
    printf("-------------------------------------------------------------¥n");
    int i;
    for (i = 0; i < 4; i++)
    {
        PrintPerson(list[i]);
        printf(" :%4d¥n", Sum(list[i]));
    }
}
```

ここでは4人分の情報で初期化

list[i]は，インデックスiの要素(一つの構造体変数)で一人分の情報
これは配列引数ではないことに注意!

```
番号 氏名        毎月の給料(単位:千円)                    合計
-------------------------------------------------------------
10   Matsushima 10 15 12 10 15 20 20 15 11 15 12 10 : 165
15   Abe        10 15 14 20 15 20 20 16 11 15 42 10 : 208
16   Trump      20 15 13 15 15 30 20 15 11 25 12 10 : 201
20   Putin      12 15 12 20 15 20 10 17 11 15 12 10 : 169
続行するには何かキーを押してください . . .
```