ce-8. 構造体，レコードデータファイル

（Cプログラミング応用，全１４回）

https://www.kkaneko.jp/cc/c/index.html

金子邦彦

例題１．バイナリファイル形式のファイルからのデータ読み込み

- 次のような名簿ファイル（バイナリファイル形式）を読み込んで，画面に表示するプログラムを作る

name	age	address
Ken	**20**	**NewYork**
Bill	**32**	**HongKong**
Mike	**35**	**Paris**

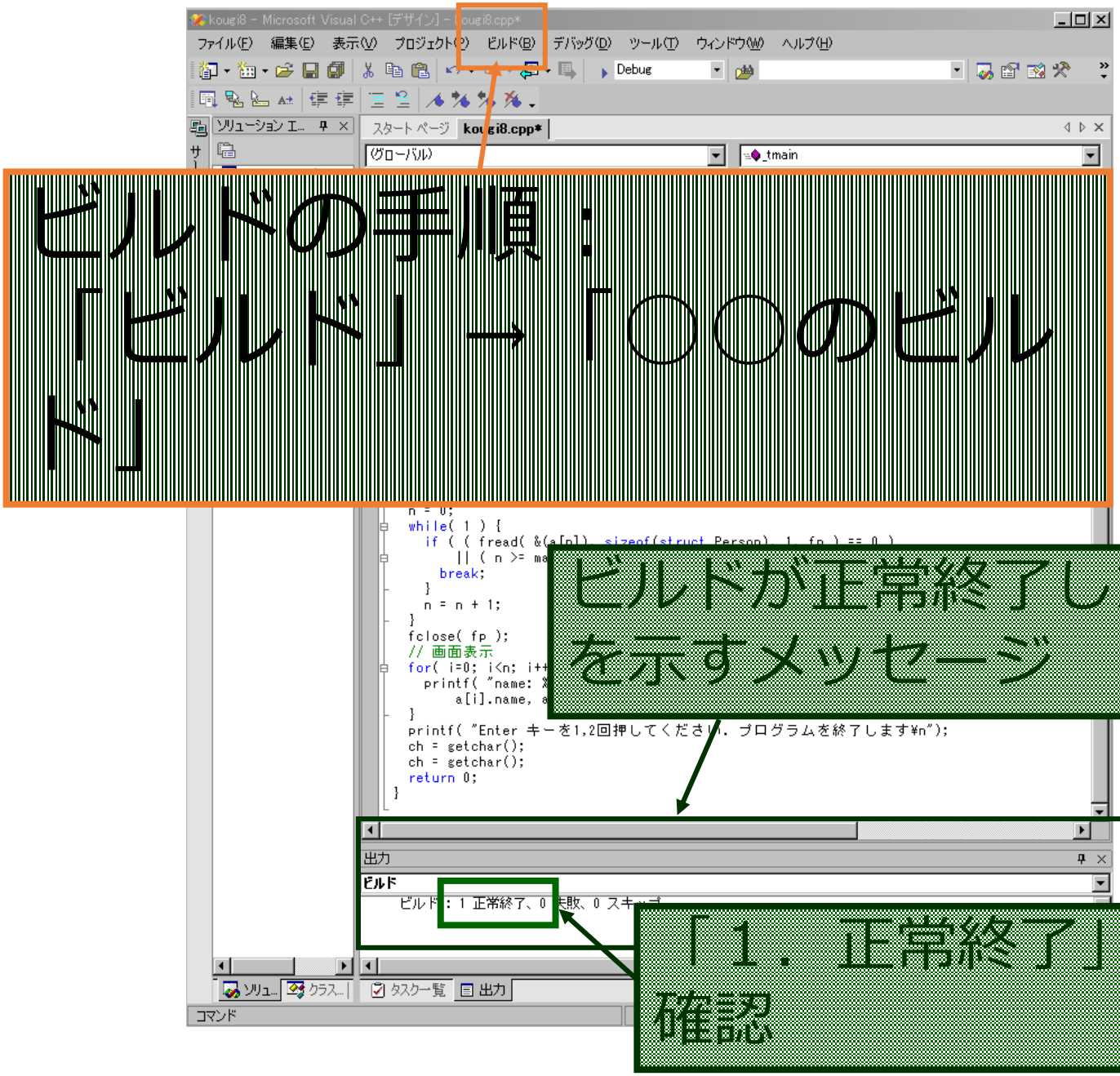
名簿ファイル

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
struct Person {
    char name[20];
    int age;
    char address[20];
};
int main()
{
  const int max_lines = 100;
  const char file_name[] = "z:¥¥PersonData.bin";
  struct Person a[max_lines];
  FILE *fp;
  int n;
  int i;
  int ch;
  // データファイル読み込み
  fp = fopen( file_name, "r" );
  if ( fp == NULL ) {
    fprintf( stderr, "ファイル %s のオープンに失敗しました" );
    return -1;
  }
  n = 0;
  while( 1 ) {
    if ( ( fread( &(a[n]), sizeof(struct Person), 1, fp ) == 0 )
        || ( n >= max_lines ) ) {
      break;
    }
    n = n + 1;
  }
  fclose( fp );
  // 画面表示
  for( i=0; i<n; i++ ) {
    printf( "name: %s, age: %d, address: %s¥n",
        a[i].name, a[i].age, a[i].address );
  }
  printf( "Enter キーを1,2回押してください. プログラムを終了します¥n");
  ch = getchar();
  ch = getchar();
  return 0;
}
```

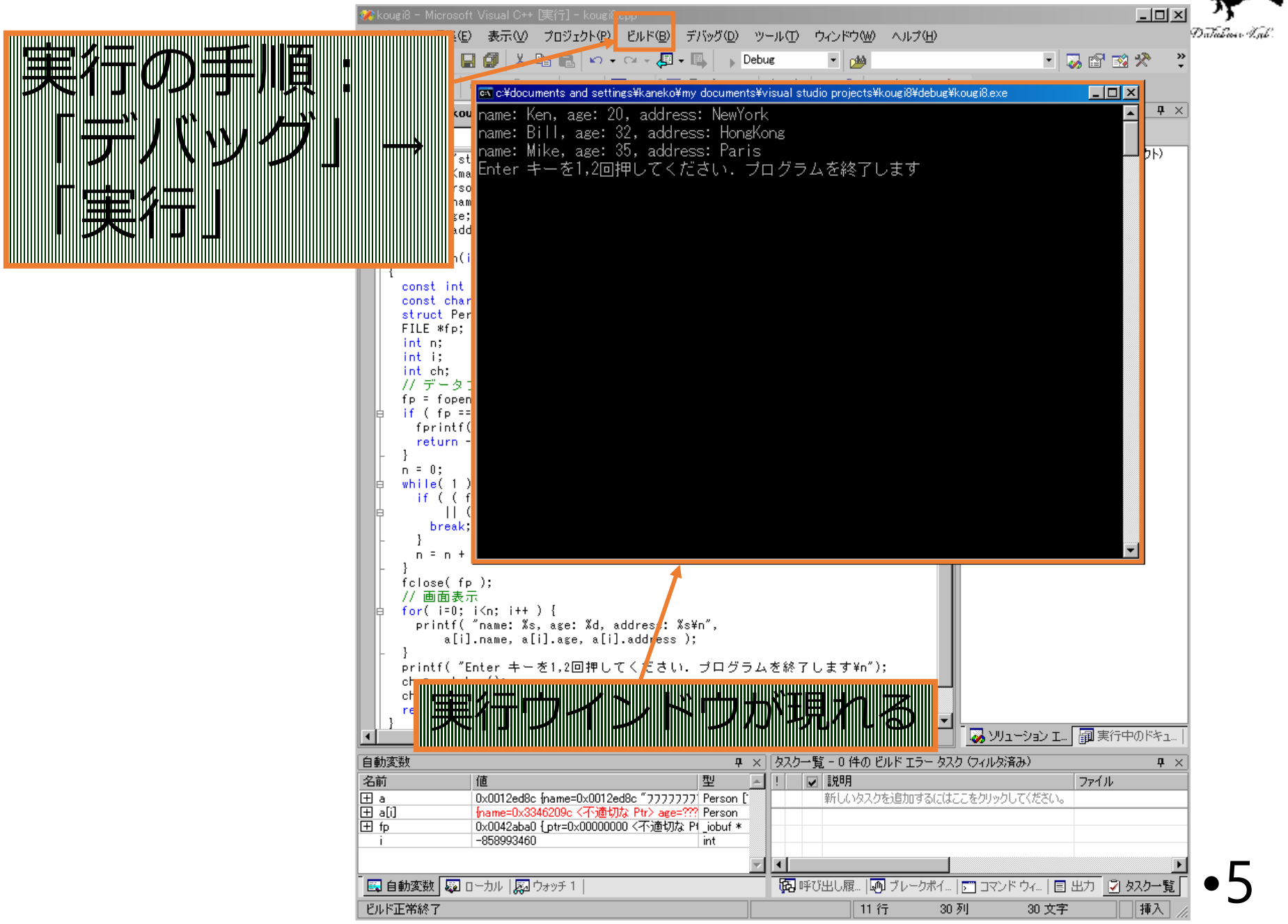
この fread はバイナリファイル形式のファイルを読み出す

Person構造体を，　１度に１つ

ビルド後の画面



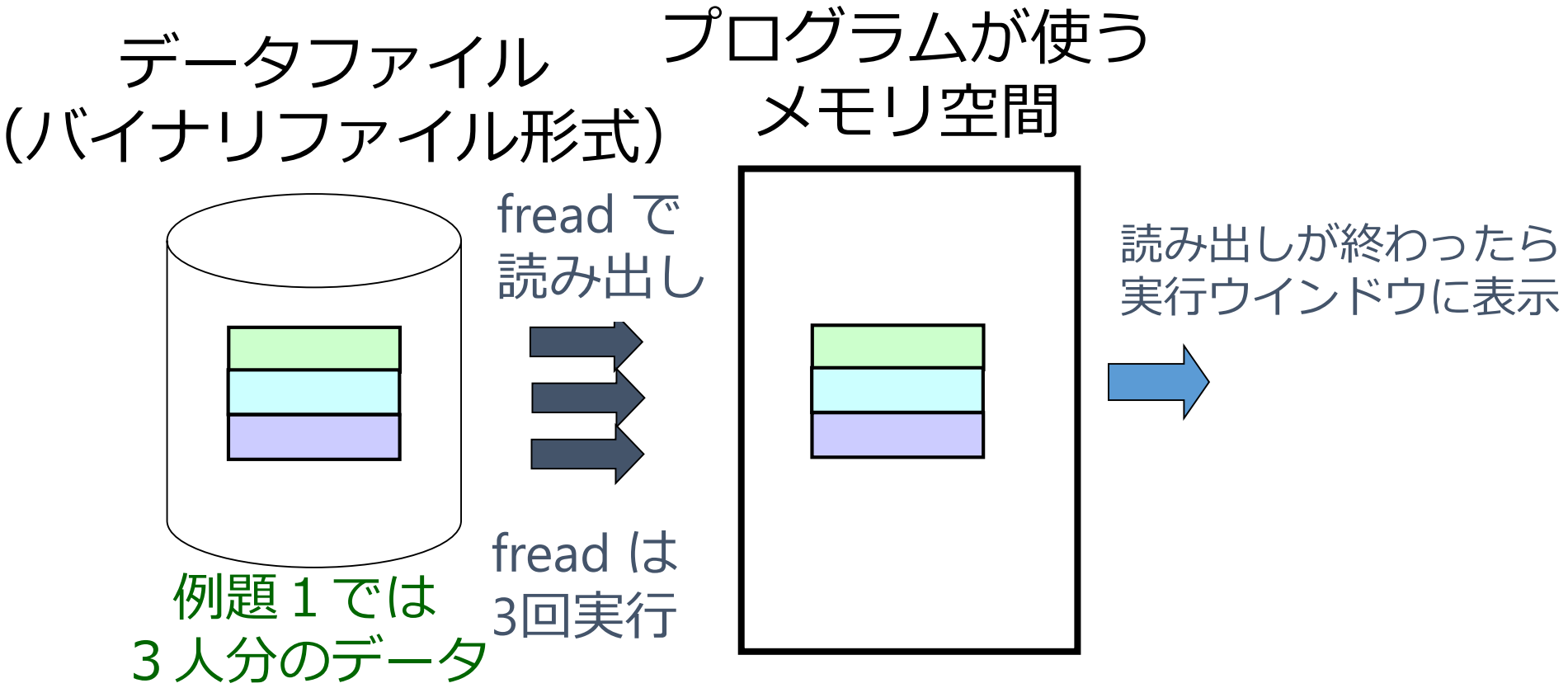
実行中の画面

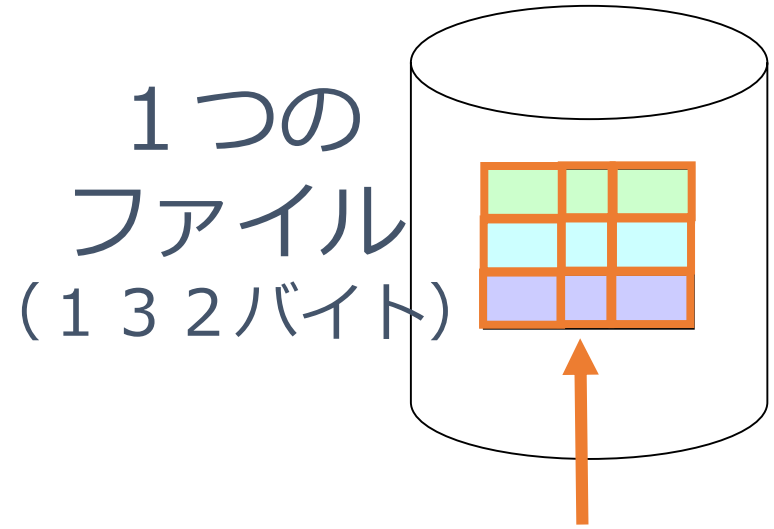

実行の手順：
「デバッグ」→
「実行」
c:¥documents and settings¥kaneko¥my documents¥visual studio projects¥kougi8¥debug¥kougi8.exe
name: Ken, age: 20, address: NewYork
name: Bill, age: 32, address: HongKong
name: Mike, age: 35, address: Paris
Enter キーを1,2回押してください．プログラムを終了します
実行ウインドウが現れる

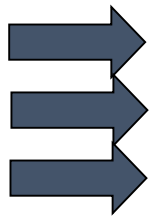
実行結果の例

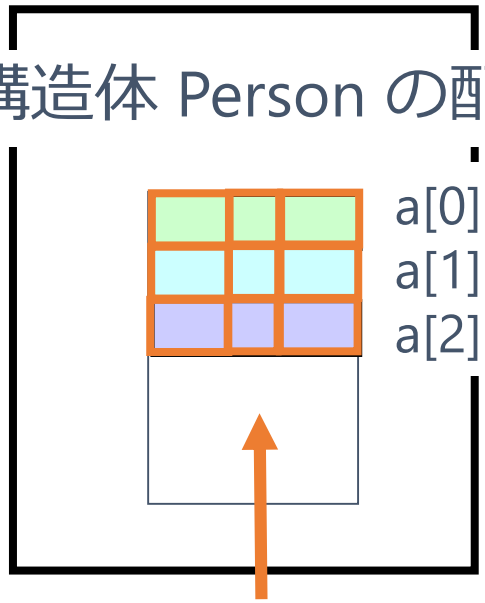
```
name: Ken, age: 20, address: NewYork
name: Bill, age: 32, address: HongKong
name: Mike, age: 35, address: Paris
Enter キーを1,2回押してください．プログラムを終了します
```

例題１のプログラムが行っていること







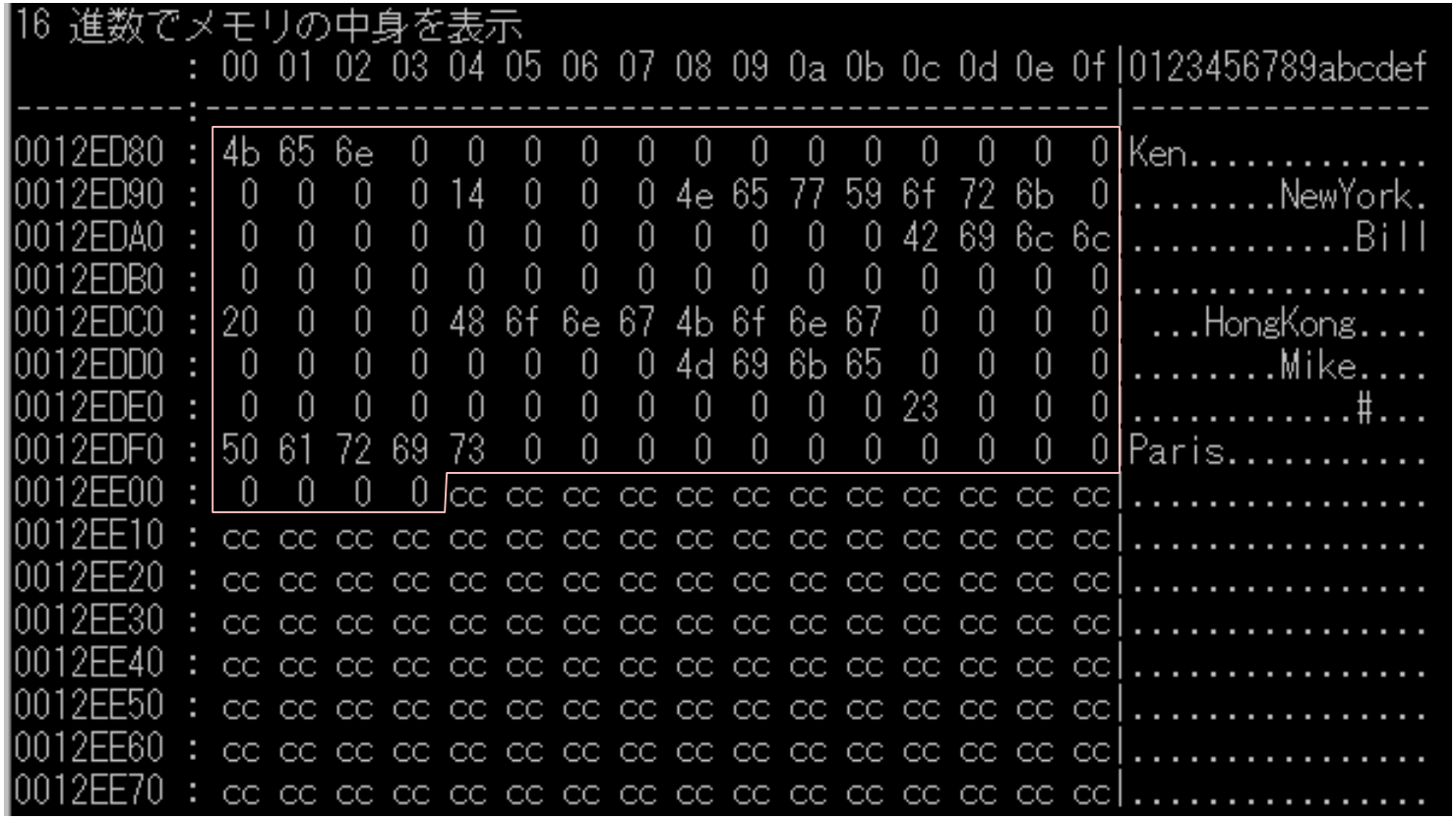


name	age	address
Ken	**20**	**NewYork**
Bill	**32**	**HongKong**
Mike	**35**	**Paris**

name	age	address
Ken	**20**	**NewYork**
Bill	**32**	**HongKong**
Mike	**35**	**Paris**

name, age, address の３つの「メンバ」から構成される

実際のメモリの中身



16 進数でメモリの中身を表示

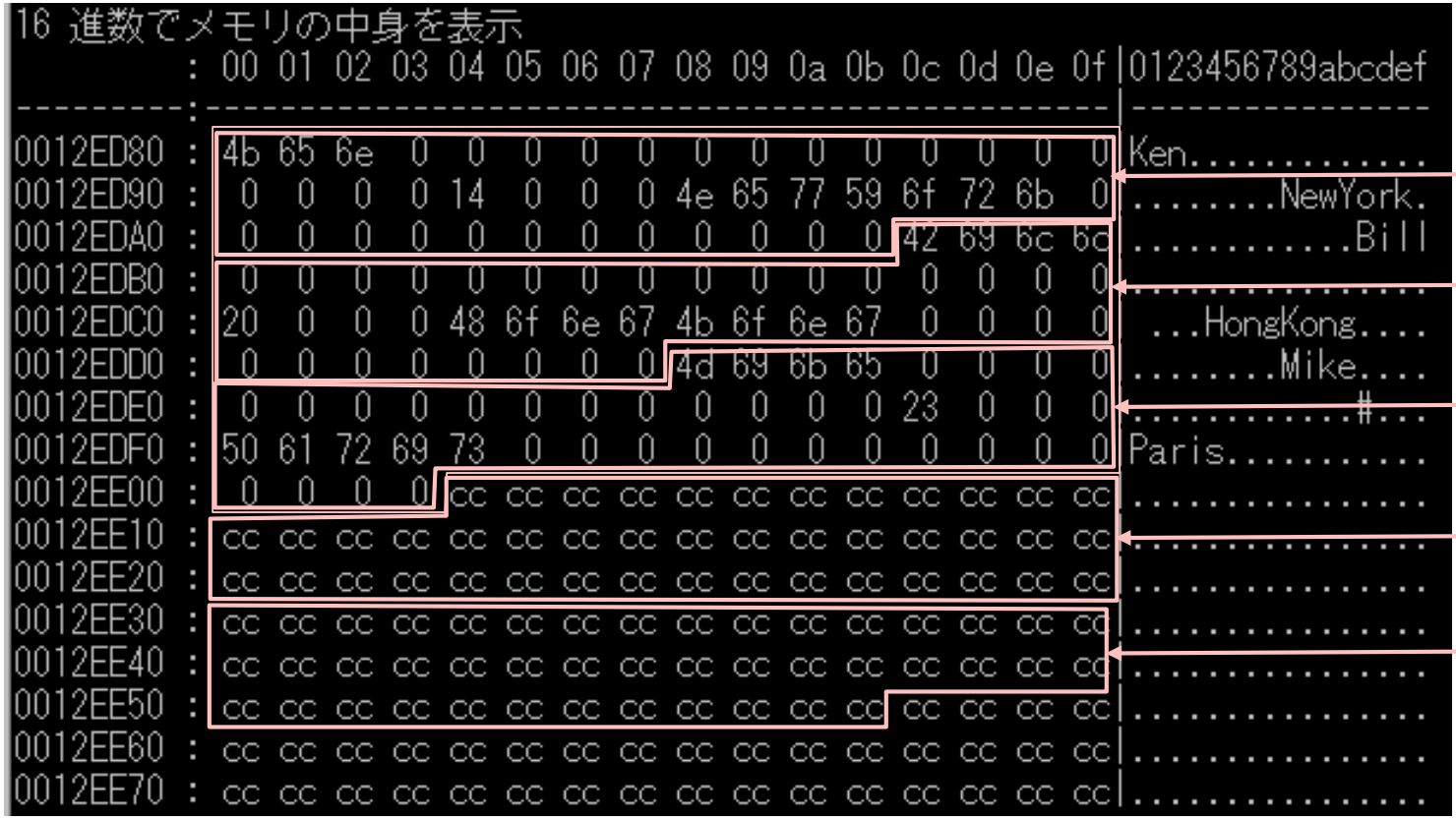
```
         : 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f|0123456789abcdef
---------:-------------------------------------------------|----------------
0012ED80 : 4b 65 6e  0  0  0  0  0  0  0  0  0  0  0  0  0|Ken.............
0012ED90 :  0  0  0  0 14  0  0  0 4e 65 77 59 6f 72 6b  0|........NewYork.
0012EDA0 :  0  0  0  0  0  0  0  0  0  0  0  0 42 69 6c 6c|............Bill
0012EDB0 :  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0|................
0012EDC0 : 20  0  0  0 48 6f 6e 67 4b 6f 6e 67  0  0  0  0| ...HongKong....
0012EDD0 :  0  0  0  0  0  0  0  0 4d 69 6b 65  0  0  0  0|........Mike....
0012EDE0 :  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0  0|............#...
0012EDF0 : 50 61 72 69 73  0  0  0  0  0  0  0  0  0  0  0|Paris...........
0012EE00 :  0  0  0  0 cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE10 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE20 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE30 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE40 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE50 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE60 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
0012EE70 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc|................
```

元々のファイルサイズ：　１３２バイト
実メモリでのサイズ：　１３２バイト

バイナリファイル形式の性質

name	age	address
Ken	**20**	**NewYork**
Bill	**32**	**HongKong**
Mike	**35**	**Paris**

実際のメモリの中身



a[0]

a[1]

a[2]

a[3]

a[4]

元々のファイルサイズ：　１３２バイト
実メモリでのサイズ：　１３２バイト

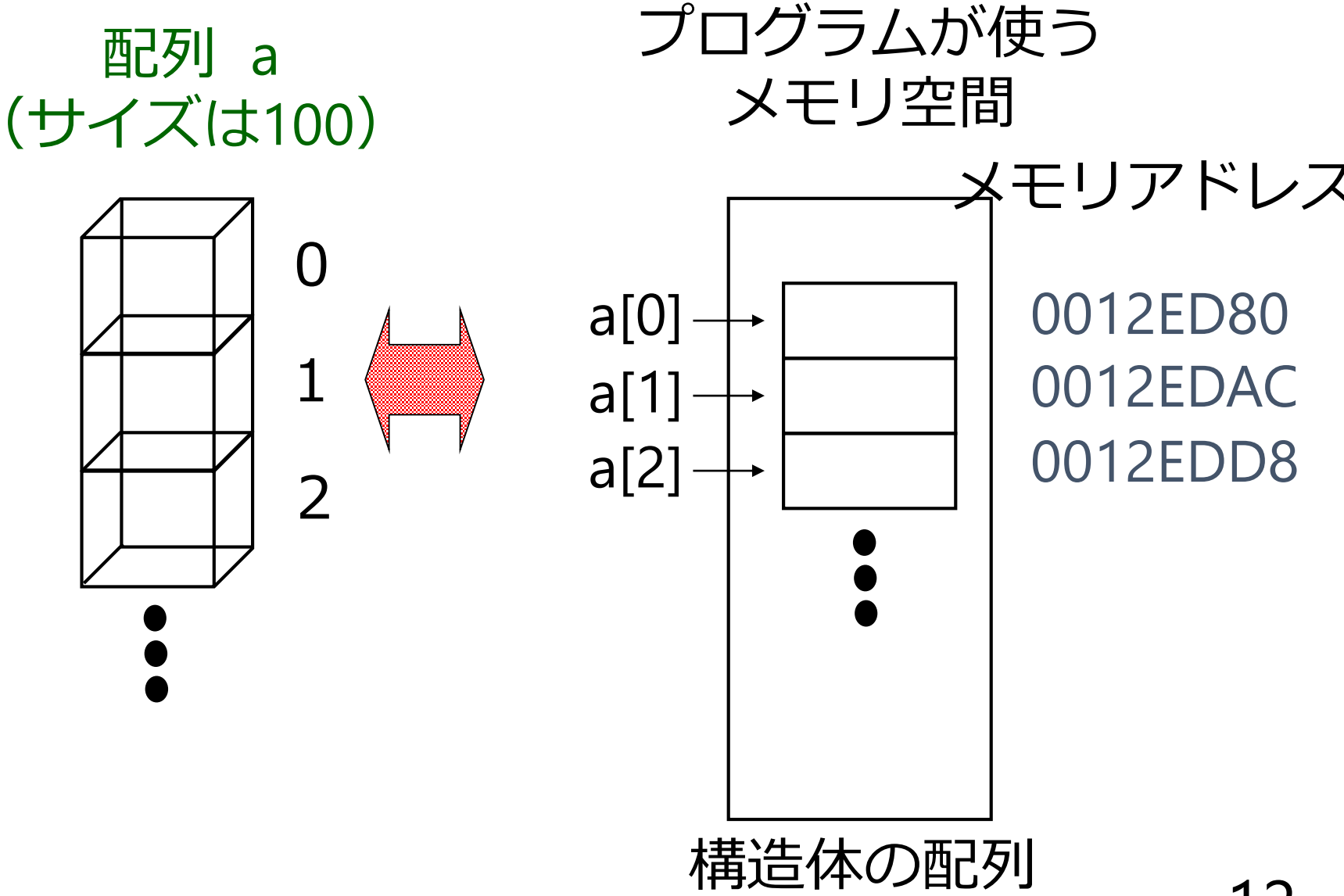
バイナリファイル形式の性質

name	age	address
Ken	**20**	**NewYork**
Bill	**32**	**HongKong**
Mike	**35**	**Paris**

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
struct Person {
    char name[20];
    int age;
    char address[20];
};
int main()
{
  const int max_lines = 100;
  const char file_name[] = "z:¥¥PersonData.bin";
  struct Person a[max_lines];
  FILE *fp;
  int n;
  int i;
  int ch;
  // データファイル読み込み
  fp = fopen( file_name, "r" );
  if ( fp == NULL ) {
    fprintf( stderr, "ファイル %s のオープンに失敗しました" );
    return -1;
  }
  n = 0;
  while( 1 ) {
    if ( ( fread( &(a[n]), sizeof(struct Person), 1, fp ) == 0 )
        || ( n >= max_lines ) ) {
      break;
    }
    n = n + 1;
  }
  fclose( fp );
  // 画面表示
  for( i=0; i<n; i++ ) {
    printf( "name: %s, age: %d, address: %s¥n",
        a[i].name, a[i].age, a[i].address );
  }
  printf( "Enter キーを1,2回押してください. プログラムを終了します¥n");
  ch = getchar();
  ch = getchar();
  return 0;
}
```

構造体 **Person** の定義

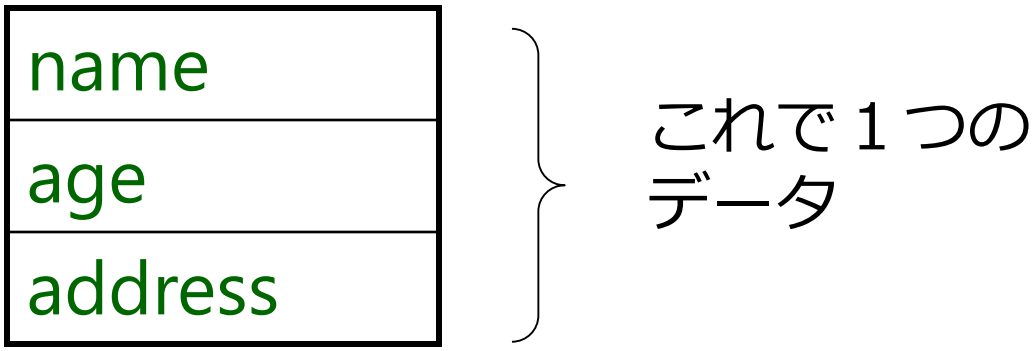
配列とメモリアドレス


配列 a
（サイズは100）
0
1
2
プログラムが使う
メモリ空間
メモリアドレス
a[0]
a[1]
a[2]
0012ED80
0012EDAC
0012EDD8
構造体の配列

構造体（struct）

- データの集合を一つのデータ型として扱う仕組み。新たな型名を付けて登録することができる。
- 構造体の要素をメンバと呼び、それぞれに型を指定し、名前を付ける。
- 構造体のメンバとして他の構造体を含むことが許される。特に、同じ構造体をメンバとするもの（再帰的構造体）も許される。

構造体の例



例題１の
構造体

演算子 「.」 の意味

- 構造体のメンバを指定

  例題１では

  a[i].name

  a[i].age

  a[i].address

演算子 -> の意味

- ポインタの指す構造体のメンバを指定

```
z->re_part = x;
```

- z が struct complex なる構造体を指すポインタであるとして、
- そのメンバ re_part に x の値を代入。

例題２．構造体データのメモリ配置を見る

- 下記の構造体 ImaginaryNumber について，メモリアドレスを表示してみる
  - １つの構造体は 16 バイト

real_part	imaginary_part
3.1	**-2.4**
4.5	**5.1**
-2.4	**6.3**

8バイト　　8バイト

```
#include "stdio.h"
#include <math.h>
struct ImaginaryNumber {
  double real_part;
  double imaginary_part;
};
int main()
{
  struct ImaginaryNumber a[] = {{3.1, -2.4},
                                {4.5, 5.1},
                                {-2.4, 6.3}};
  int i;
  int ch;
  for (i=0; i<3; i++ ) {
    printf( "複素数 %3.1f + %3.1f i¥n", a[i].real_part, a[i].imaginary_part );
  }
  for (i=0; i<3; i++ ) {
    printf( "address(a[%d]) = %p¥n", i, &(a[i]) );
  }
  printf( "Enter キーを1,2回押してください. プログラムを終了します¥n");
  ch = getchar();
  ch = getchar();
  return 0;
}
```

a[0] に 3.1, -2.4 を
a[1] に 4.5, 5.1 を
a[2] に -2.4, 6.3 をセット

「%3.1f」は, 小数点以上は最大 3 桁,
小数点以下は最大 1 桁の表示

「%p」はメモリアドレスの表示

「&」はメモリアドレスの取得

メモリアドレス表示

実行結果の例

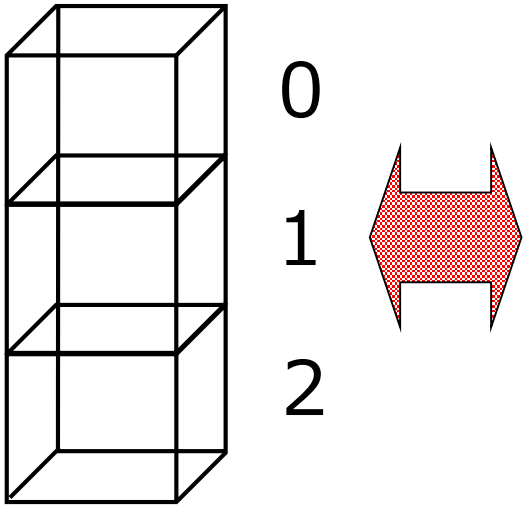
```
複素数 3.1 + -2.4 i
複素数 4.5 + 5.1 i
複素数 -2.4 + 6.3 i
address(a[0]) = 0012FEA8
address(a[1]) = 0012FEB8
address(a[2]) = 0012FEC8
Enter キーを1,2回押してください．プログラムを終了します
```

表示された
メモリアドレス*

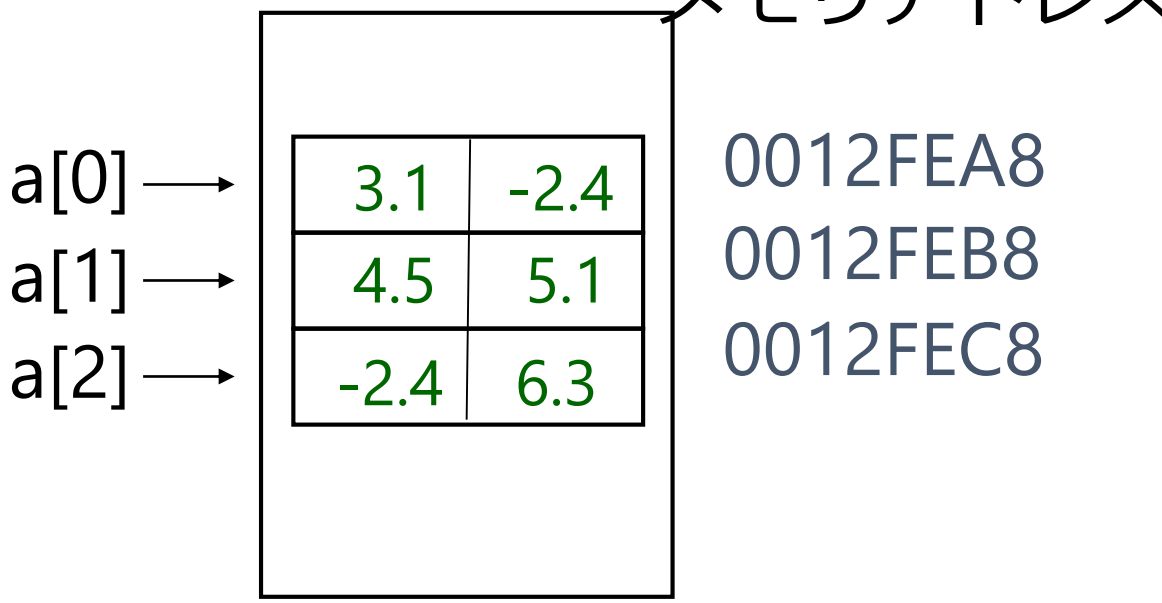
メモリアドレスの値が
ここでの「例」と違っている
ことはある（動作は正しい）

配列とメモリアドレス

配列 a
（サイズは３）

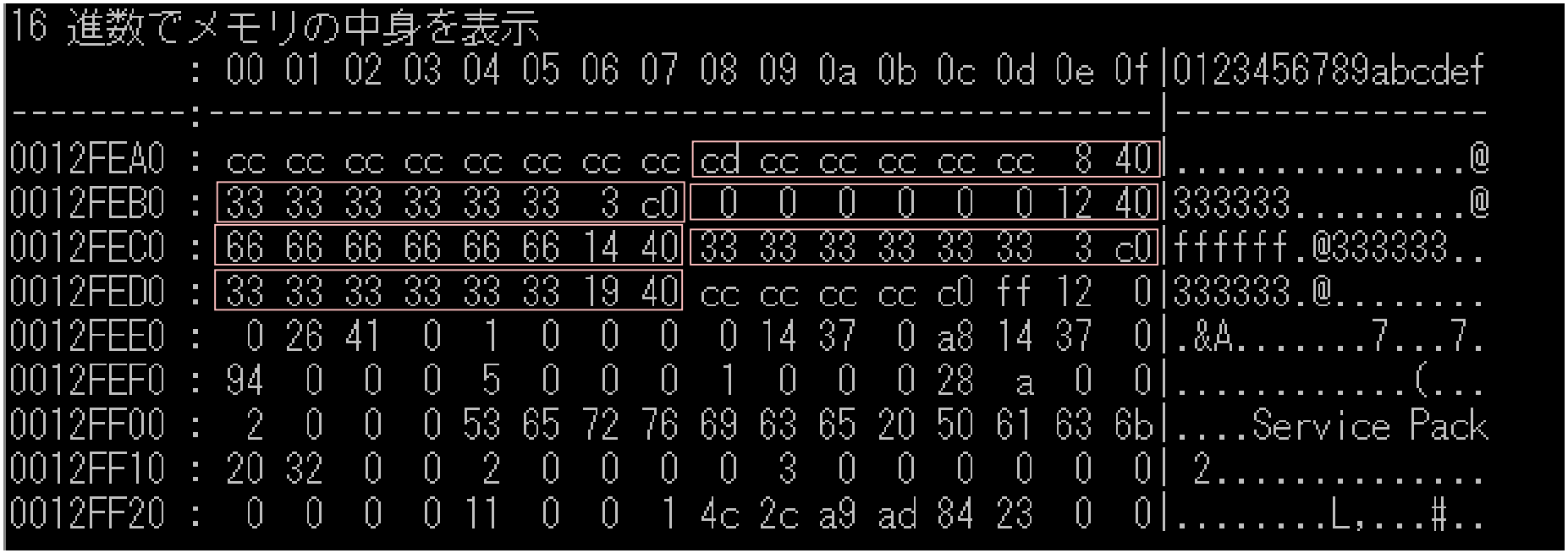


プログラムが使う
メモリ空間



構造体の配列

実際のメモリの中身



double 型の変数は
8 バイトになっている

メモリアドレスの値が
ここでの「例」と違っている
ことはある

```
#include "stdio.h"
#include <math.h>
struct ImaginaryNumber {
  double real_part;
  double imaginary_part;
};
int main()
{
  struct ImaginaryNumber a[] = {{3.1, -2.4},
                                {4.5, 5.1},
                                {-2.4, 6.3}};
  int i;
  int ch;
  for (i=0; i<3; i++ ) {
    printf( "複素数 %3.1f + %3.1f i¥n", a[i].real_part, a[i].imaginary_part );
  }
  for (i=0; i<3; i++ ) {
    printf( "address(a[%d]) = %p¥n", i, &(a[i]) );
  }
  printf( "Enter キーを1,2回押してください. プログラムを終了します¥n");
  ch = getchar();
  ch = getchar();
  return 0;
}
```

構造体 ImaginaryNumber の型宣言

構造体の配列 a の宣言と初期化

構造体のメンバ

構造体の型宣言

- 構造体には，名前がある
- それぞれのデータ（メンバという）は，名前と型（データの種類のこと）がある.

```
struct Person {
    char name[20];
    int age;
    char address[20];
};
```

名前

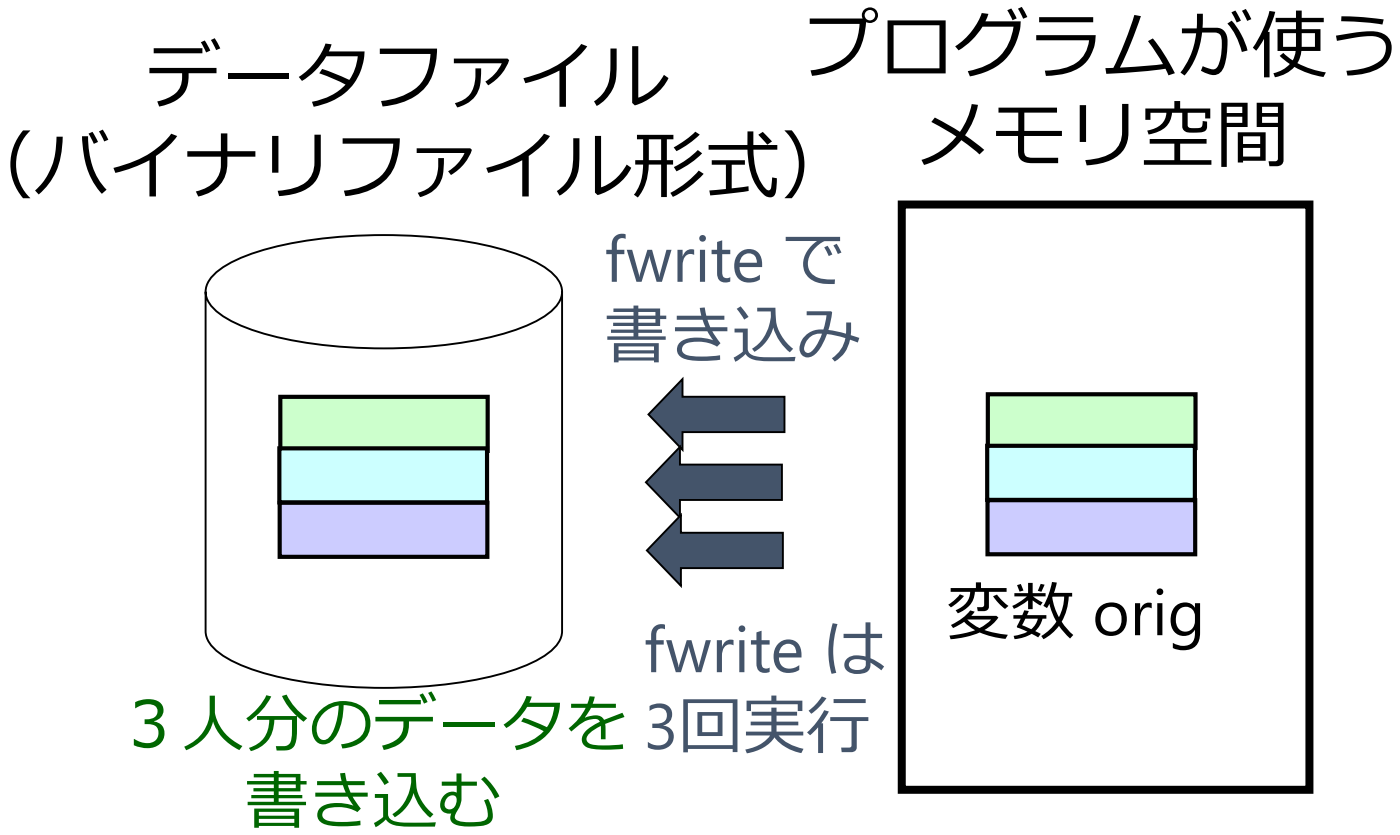
メンバ

参考．バイナリファイル形式のファイル読み出し，書き込み

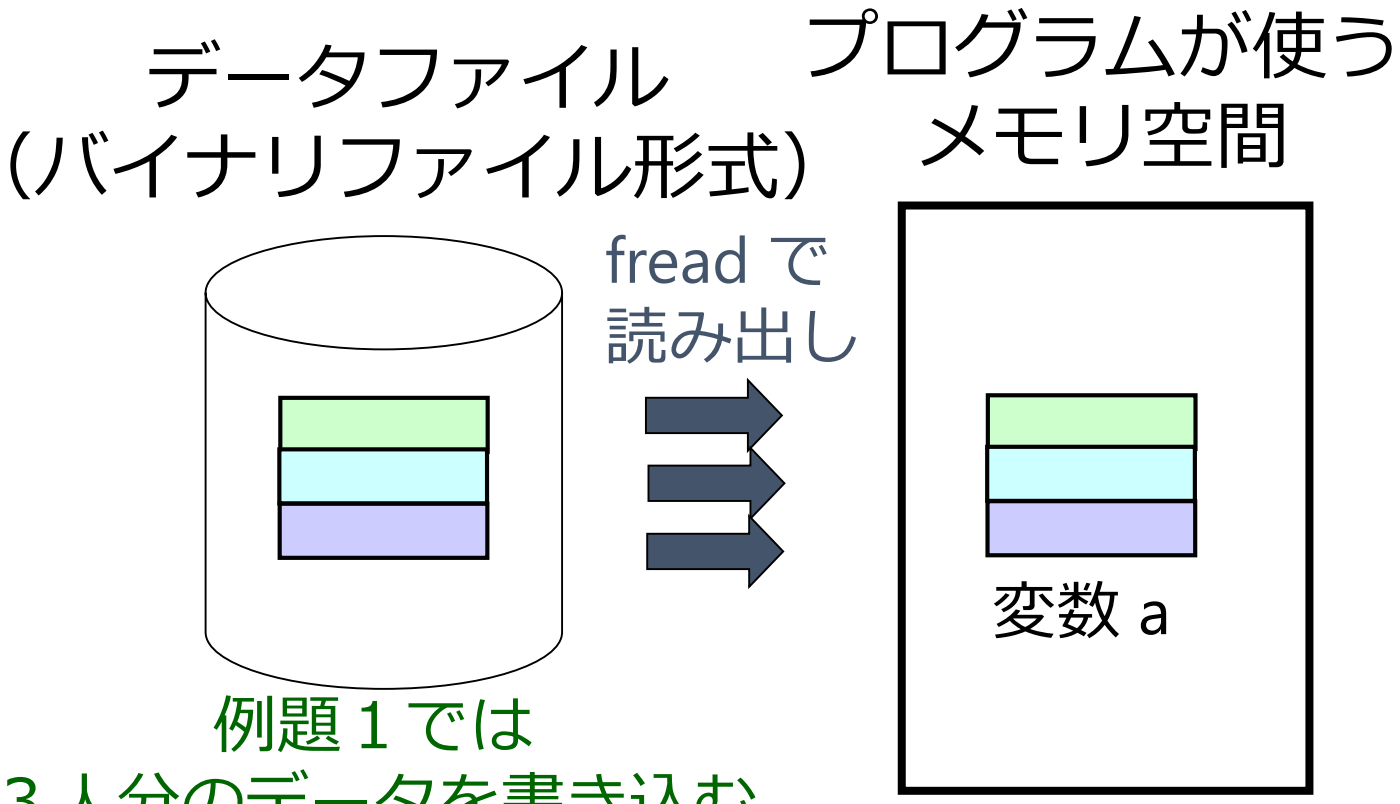
- 最初にファイルの書き込みを行い，次に，書き込んだファイルの読み出しを行うプログラム

　　→　練習用の見本として示す

- 最初は書き込み


データファイル
（バイナリファイル形式）
プログラムが使う
メモリ空間
fwrite で
書き込み
変数 orig
fwrite は
3回実行
3 人分のデータを
書き込む

- 次は読み出し


データファイル
(バイナリファイル形式)
fread で
読み出し
プログラムが使う
メモリ空間
変数 a
例題１では
３人分のデータを書き込む

```
#include "stdio.h"
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
struct Person {
    char name[20];
    int age;
    char address[20];
};

int main()
{
  const int max_lines = 100;
  const char file_name[] = "z:¥¥PersonData.bin";
  struct Person a[max_lines];
  FILE *fp;
  int n;
  int i;
  int ch;

  struct Person orig[3] = { {"kaneko", 38, "hazozaki"}, {"ken",
20, "kaizuka"}, {"mike", 30, "tenjin" } };

  printf( "書きます．ファイル名は %s¥n", file_name );
// データファイル書き込み
  fp = fopen( file_name, "w" );
  if ( fp == NULL ) {
    fprintf( stderr, "ファイル %s のオープンに失敗しました" );
    return -1;
  }
  // 書き出すのは３個（i = 0, 1, 2)
  for ( i = 0; i < 3; i++ ) {
    fwrite( &(orig[i]), sizeof(struct Person), 1, fp );
  }
  fclose( fp );

  printf( "読みます．ファイル名は %s¥n", file_name );
  // データファイル読み出し
  fp = fopen( file_name, "r" );
  if ( fp == NULL ) {
    fprintf( stderr, "ファイル %s のオープンに失敗しました" );
    return -1;
  }
  n = 0;
  while( 1 ) {
    if ( ( fread( &(a[n]), sizeof(struct Person), 1, fp ) == 0 )
                        || ( n >= max_lines ) ) {
      break;
    }
    n = n + 1;
  }
  fclose( fp );

  // 画面表示
  for( i=0; i<n; i++ ) {
    printf( "name: %s, age: %d, address: %s¥n",
                        a[i].name, a[i].age, a[i].address );
  }
  printf( "Enter キーを1,2回押してください. プログラムを終了します¥n");
  ch = getchar();
  ch = getchar();
  return 0;
}
```

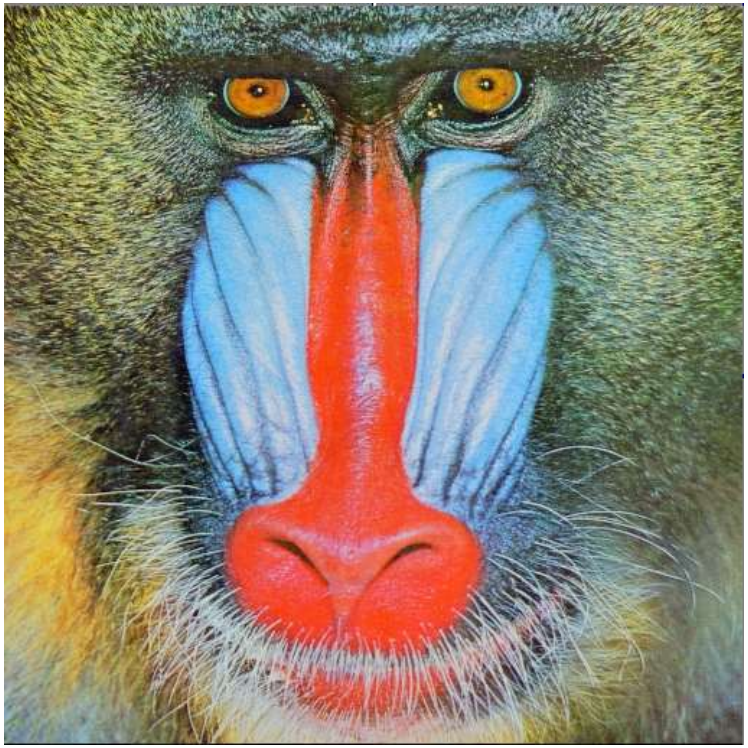
最初は書き込み

次は読み出し

例題３．Windows ビットマップファイルの読み出し，書き込み

- ２４ビットカラーの Windows ビットマップファイルについて，簡単な計算を行ってみる

実行結果の例



z:¥Mandrill.bmp

z:¥done.bmp

新しく生成されるファイル

- 画素の r, g, b 値

  x > 0 に対して

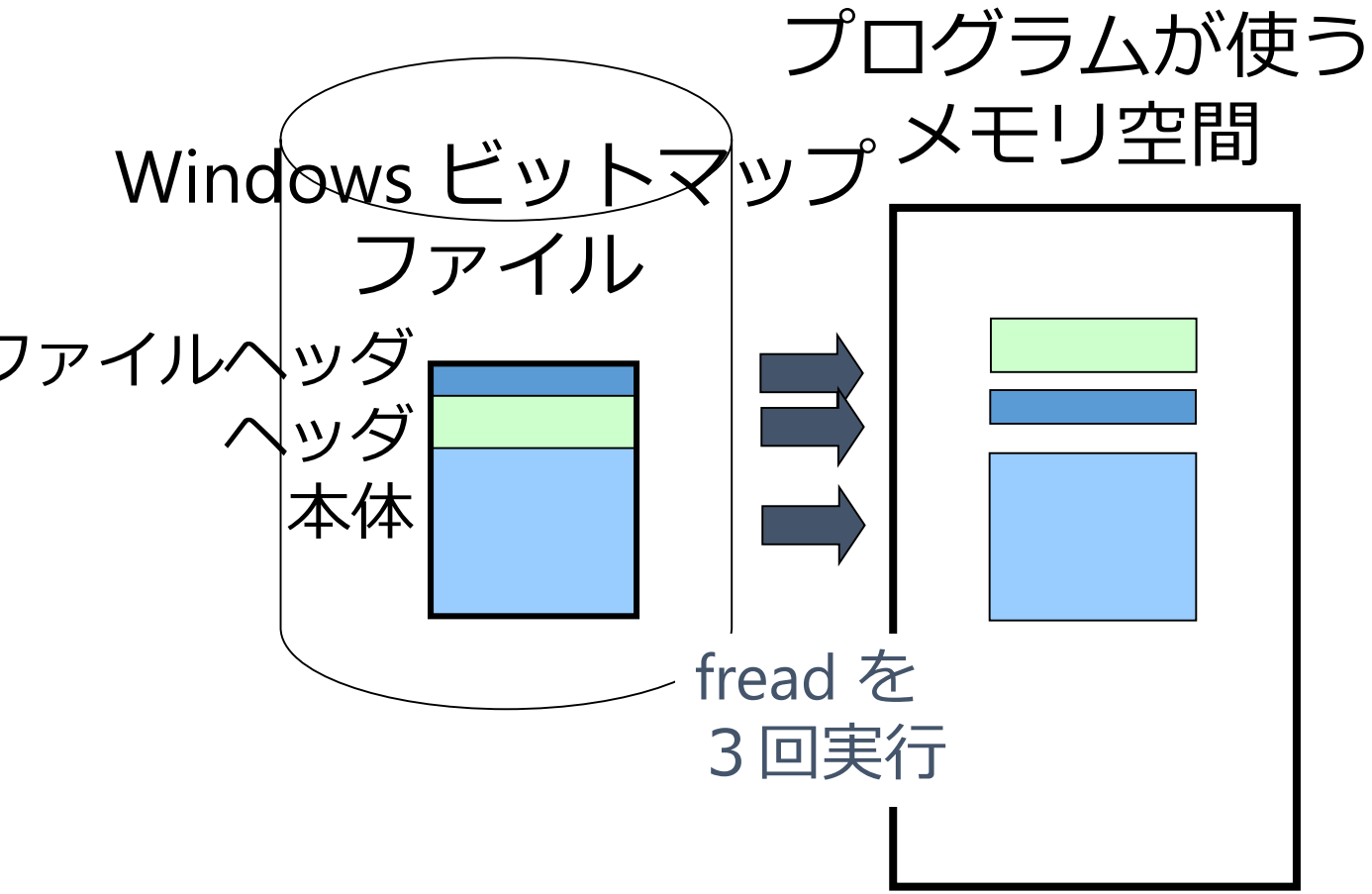
  $$r'(x, y) = | r(x,y) - r(x-1, y) |$$
  $$g'(x, y) = | g(x,y) - g(x-1, y) |$$
  $$b'(x, y) = | b(x,y) - b(x-1, y) |$$

  横方向の差分（差分の量が大きいほど，画素は「明るく」なる）

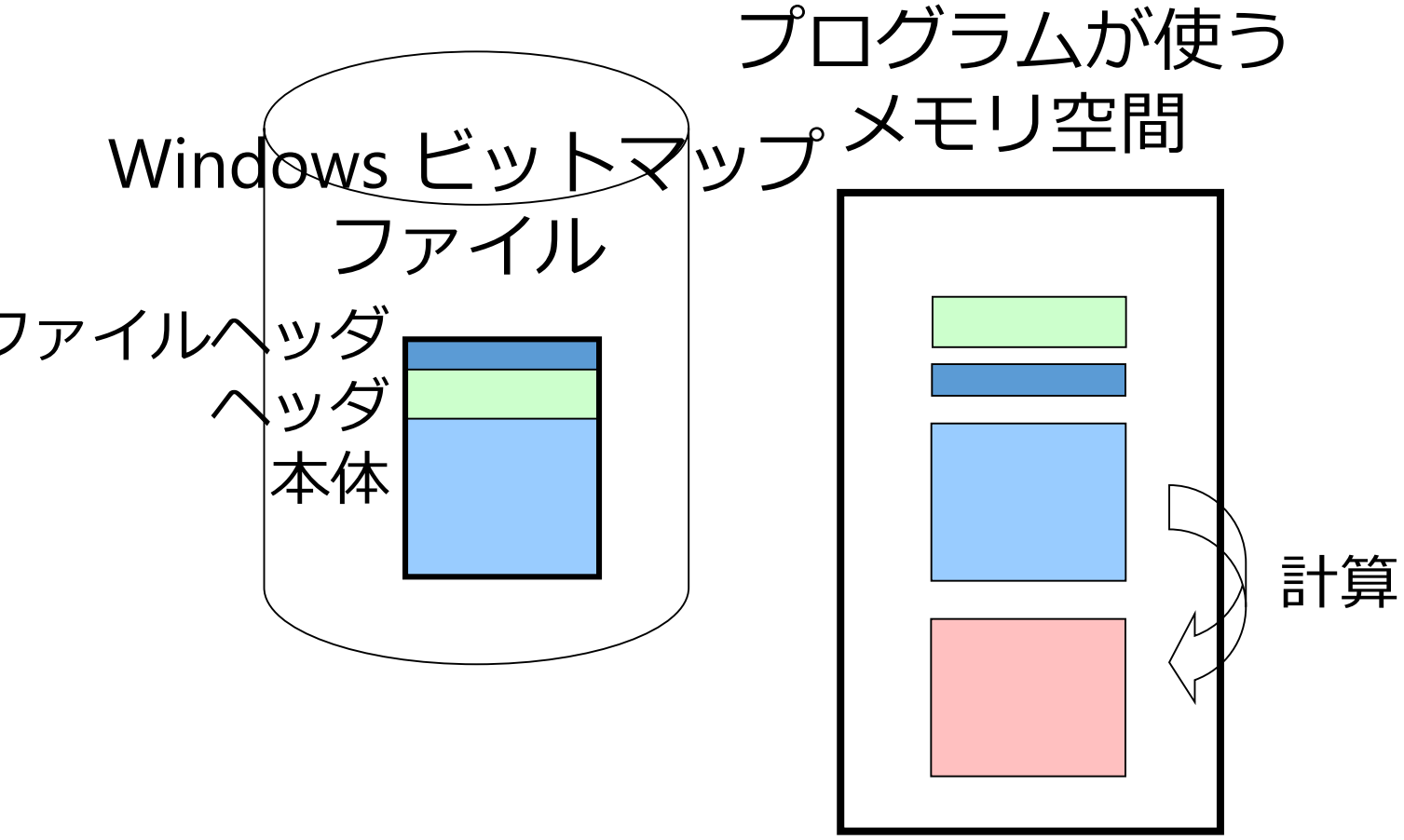
  x = 0 に対して

  $$r'(x, y) = 0$$
  $$g'(x, y) = 0$$
  $$b'(x, y) = 0$$

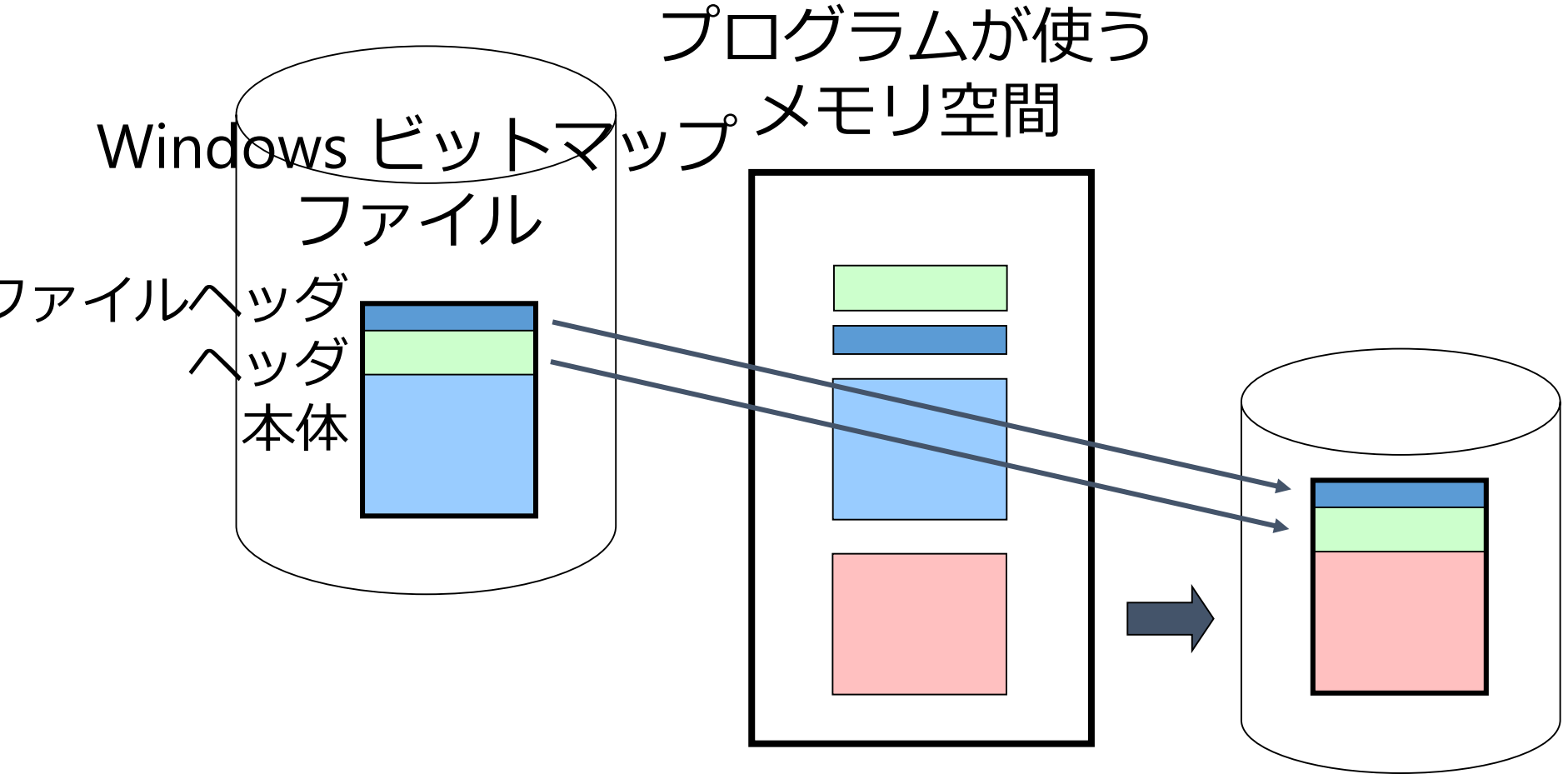
例題３のプログラムが行っていること



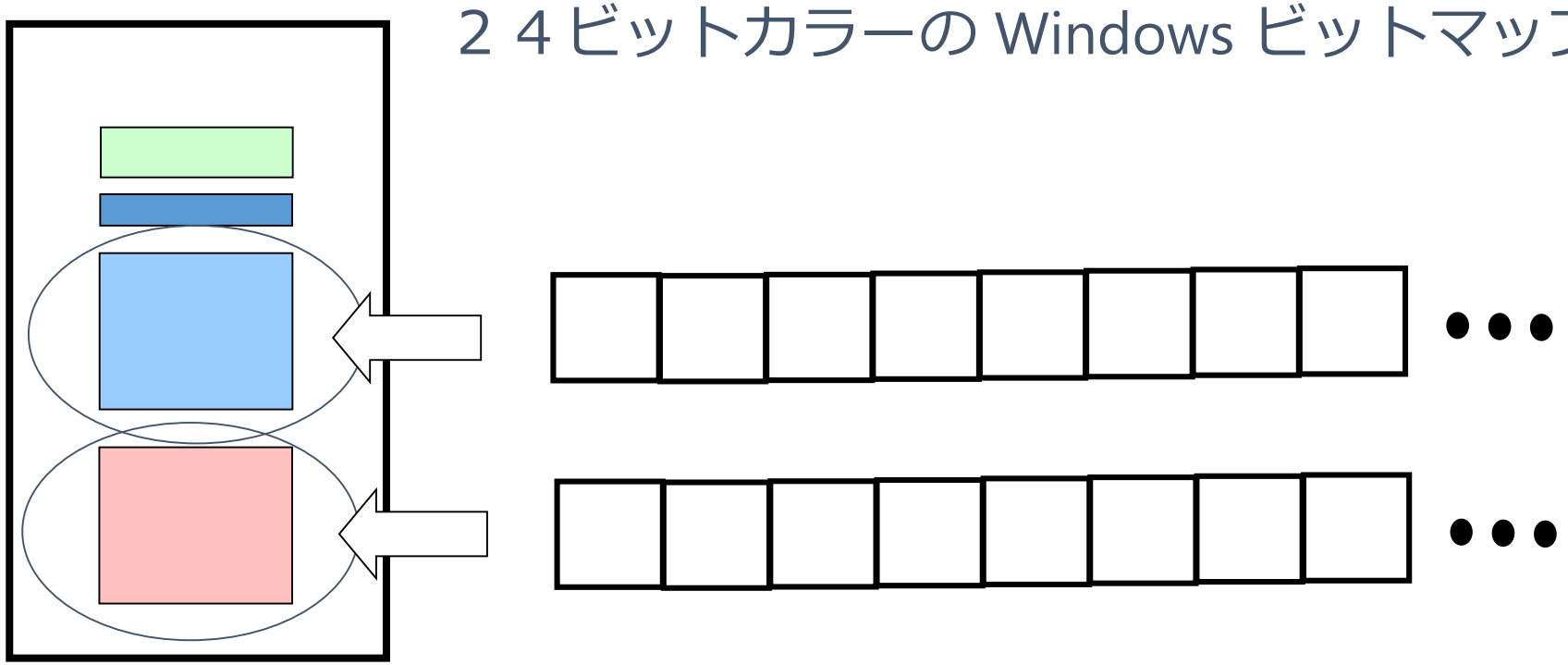
例題３のプログラムが行っていること



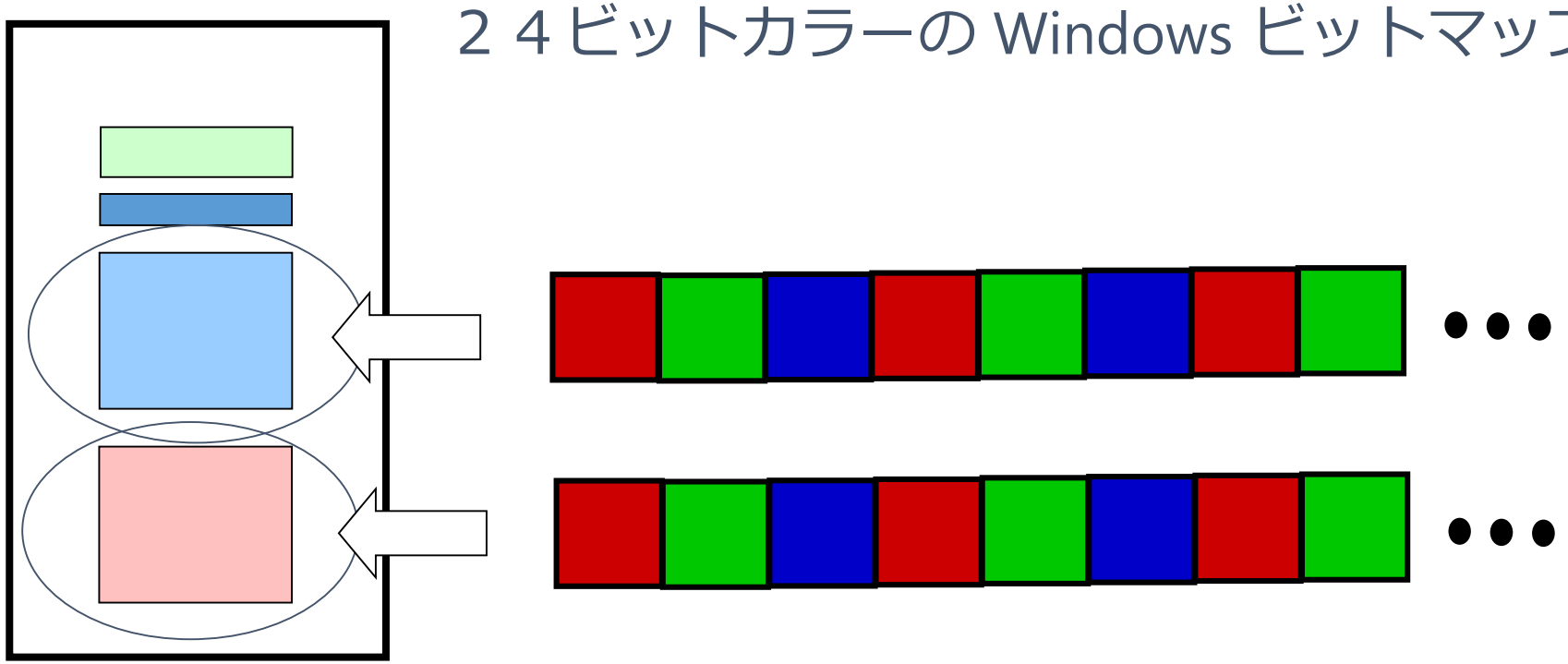
例題３のプログラムが行っていること



画像本体の実メモリ上での配置



画像本体の実メモリ上での配置



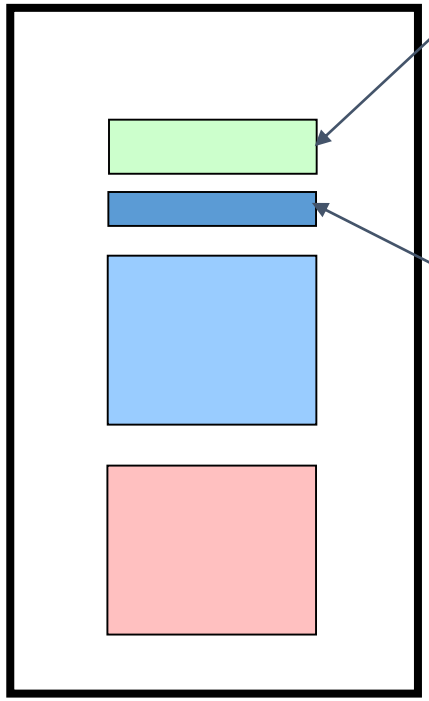
２４ビットカラーの Windows ビットマップ

実メモリ上では，バイト列のデータ

画素 (x,y) のデータは，先頭から 3xy バイト目にある

画像本体の実メモリ上での配置

２４ビットカラーの Windows ビットマップ



ファイルヘッダ

```
typedef struct tagBITMAPFILEHEADER {
  unsigned short bfType;
  unsigned long bfSize;
  unsigned short bfReserved1;
  unsigned short bfReserved2;
  unsigned long bfOffBits;
} BITMAPFILEHEADER
```

ヘッダ

```
typedef struct tagBITMAPINFOHEADER{
  unsigned long biSize;
  long biWidth;
  long biHeight;
  unsigned short biPlanes;
  unsigned short biBitCount;
  unsigned long biCompression;
  unsigned long biSizeImage;
  long biXPixPerMeter;
  long biYPixPerMeter;
  unsigned long biClrUsed;
  unsigned long biClrImporant;
} BITMAPINFOHEADER;
```