ce-6. ファイル，配列

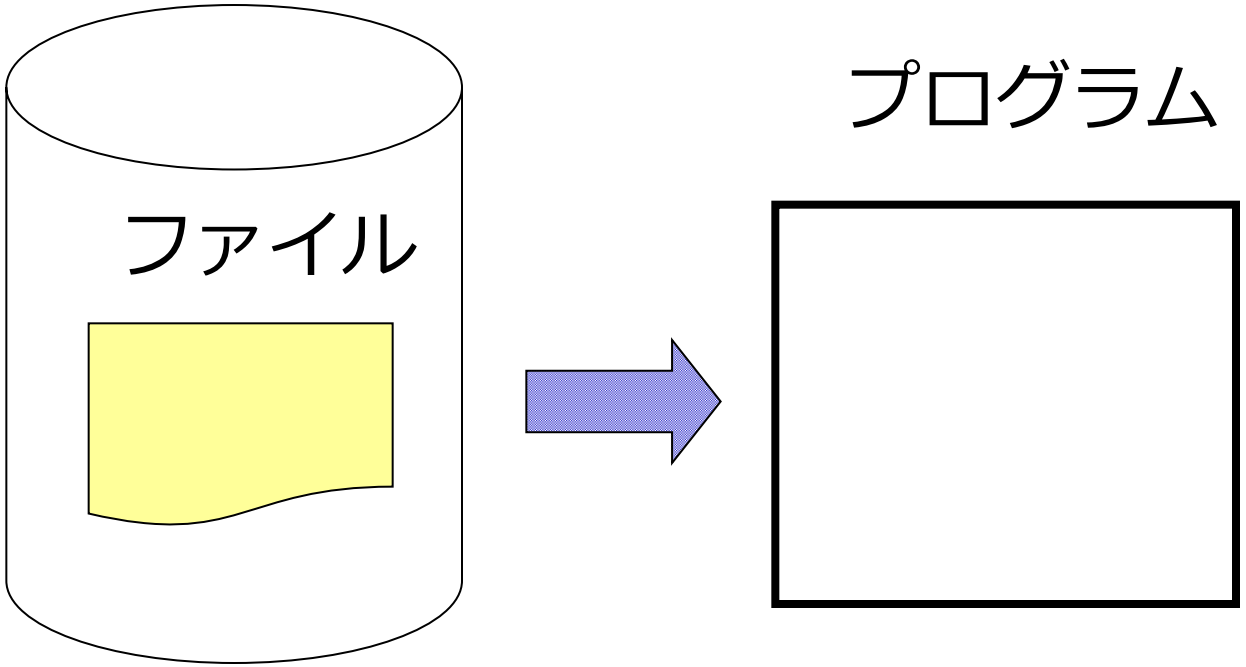
（C プログラミング応用）（全１４回）

URL: https://www.kkaneko.jp/pro/c/index.html

金子邦彦

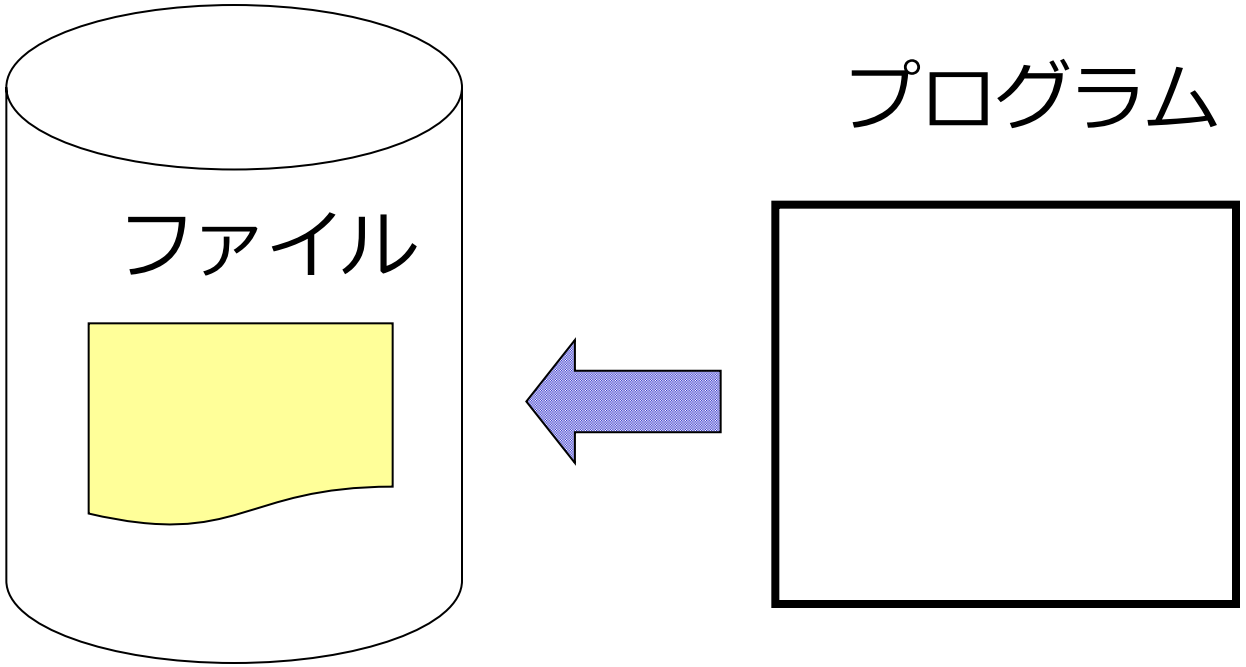
ファイル処理

ファイル読み込み



- ファイルの中身は変わらない

ファイル書き出し



- ファイルの中身が変わる
- ファイルは伸び縮みすることがある

例題１．テキスト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（テキストファイル形式）を読み込んで，１列目の氏名と，３列目の住所だけを表示するプログラムを作る
  - 各データは，半角の空白文字（１つまたは複数）で区切られる

```
金子邦彦 1200/01/01 福岡市東区箱崎３丁目      392-123-8234
○○×× 1300/12/31 福岡市東区貝塚団地        492-252-7188
●●■■ 0800/05/31 福岡市東区香椎浜１丁目    592-824-7144
```

３行のテキストファイル

テキストファイル形式

- 行単位での読み書きに意味がある。
- 人間が「目で見て」読むことができるファイル。

```
MPL 40
  pattern1 = 786
  pattern2 = 1
  pattern3 = 979
  pattern4 = 0
```

テキストファイルの例

```
<FF>0<FF><E0>^@
^PJFIF^@^A^B^A^
@<96>^@<96>^@
^@<FF><E0>~JFXX
^@^P<FF>00<FF>U
^@^C^@^
```

テキストファイルでない
バイナリファイルの例
（画像のファイル）

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
  char line[100];
  char name[100];
  char birth[100];
  char address[100];
  FILE *in_file;
  int ch;
  in_file = fopen("d:¥¥Book1.txt", "r");
  if ( in_file == NULL ) {
    return 0;
  }
  while( fgets( line, 100, in_file ) != NULL ) {
    sscanf_s( line, "%s %s %s", name, birth, address );
    printf( "name=%s, address=%s¥n", name, address );
  }
  fclose(in_file);
  ch = getchar();
  ch = getchar();
  return 0;
}
```

例題１の手順

１．準備

演習用のデータファイル d:¥Book1.txt を各自で作成

（本資料のページ９，１０，１１，１２，１３）

２．ビルドと実行

例題１のプログラムを各自で実行し，実行結果を確認

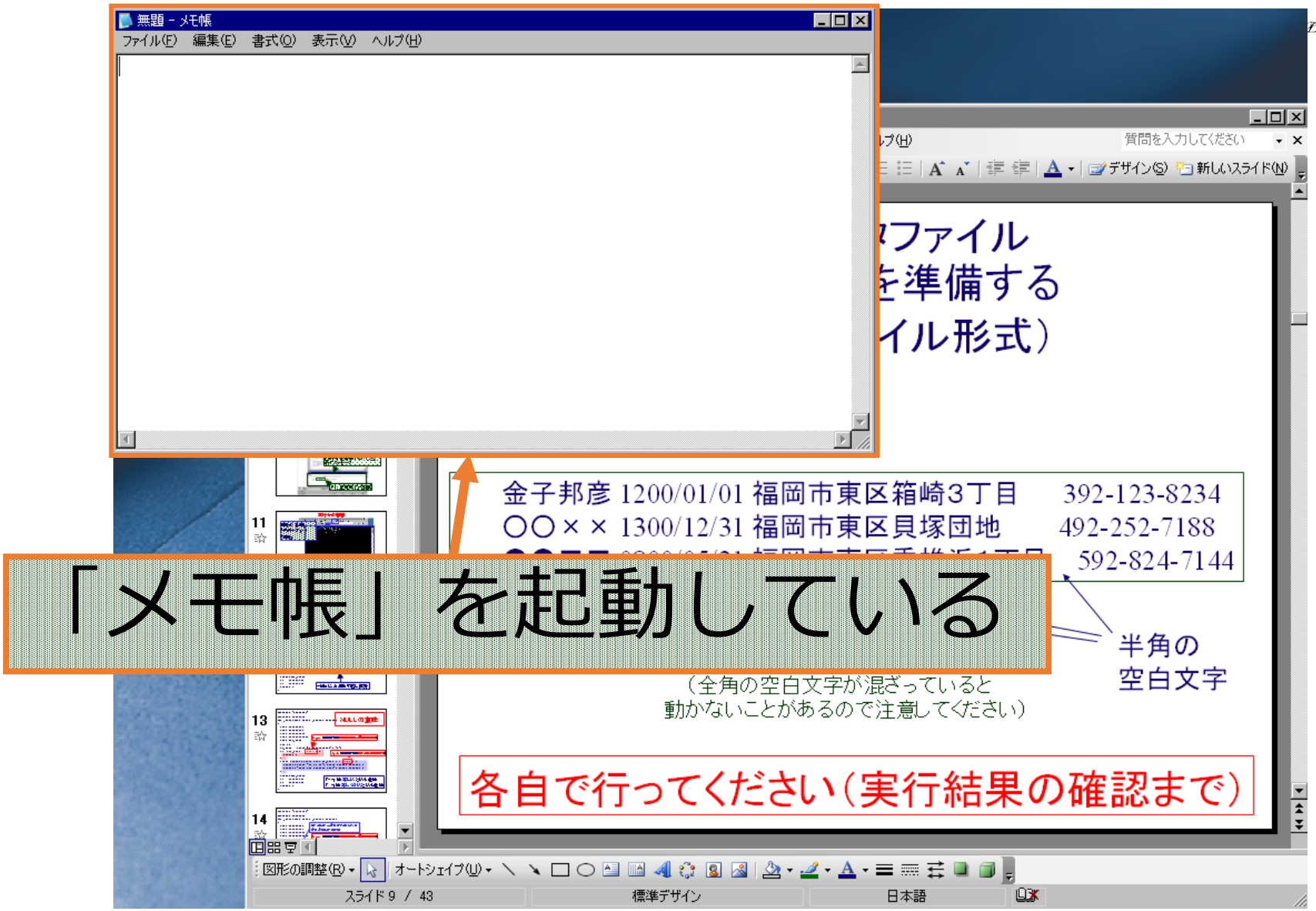
各自で行ってください
（実行結果の確認まで）

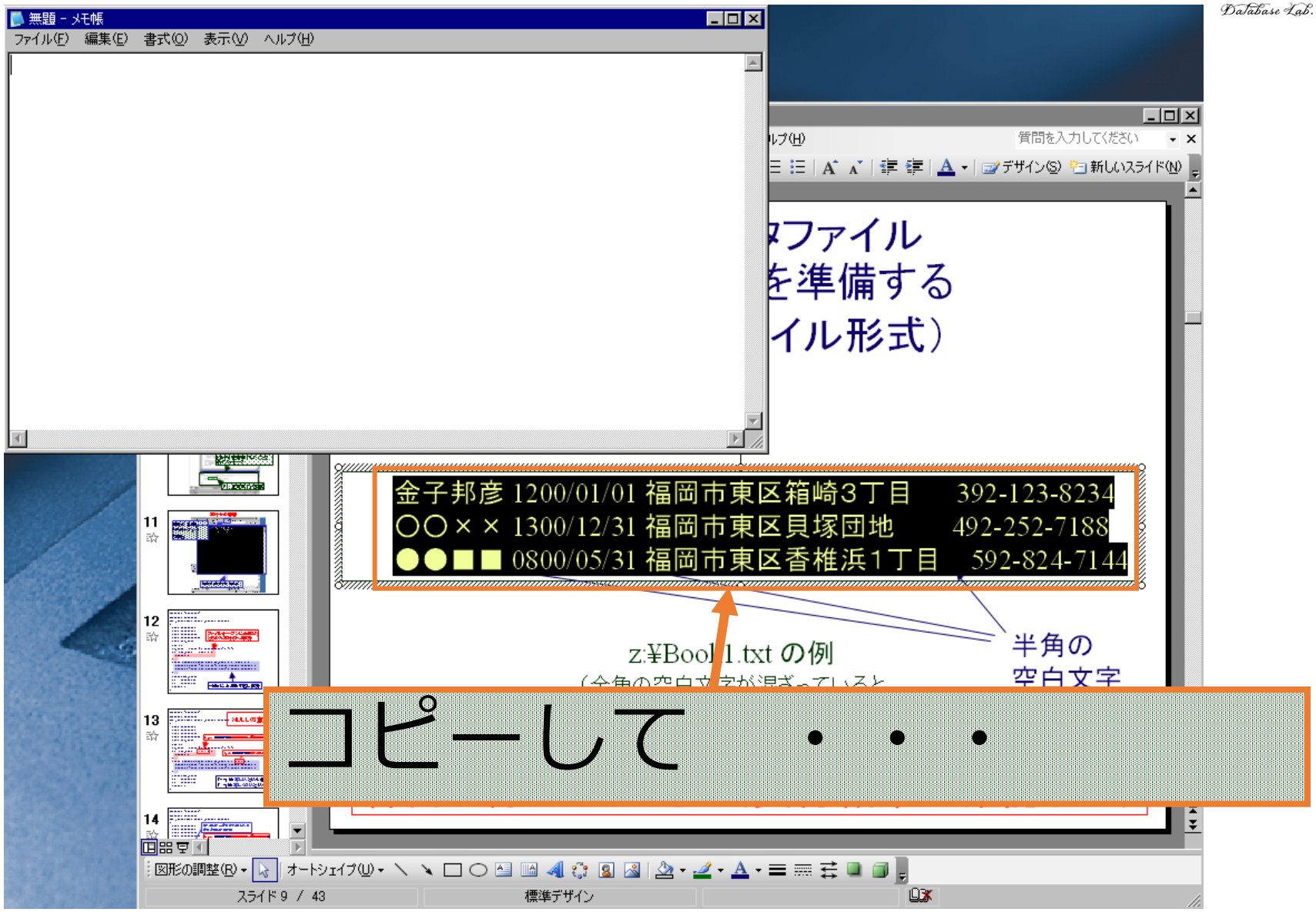
まず，データファイル d:¥Book1.txt を準備する （テキストファイル形式）

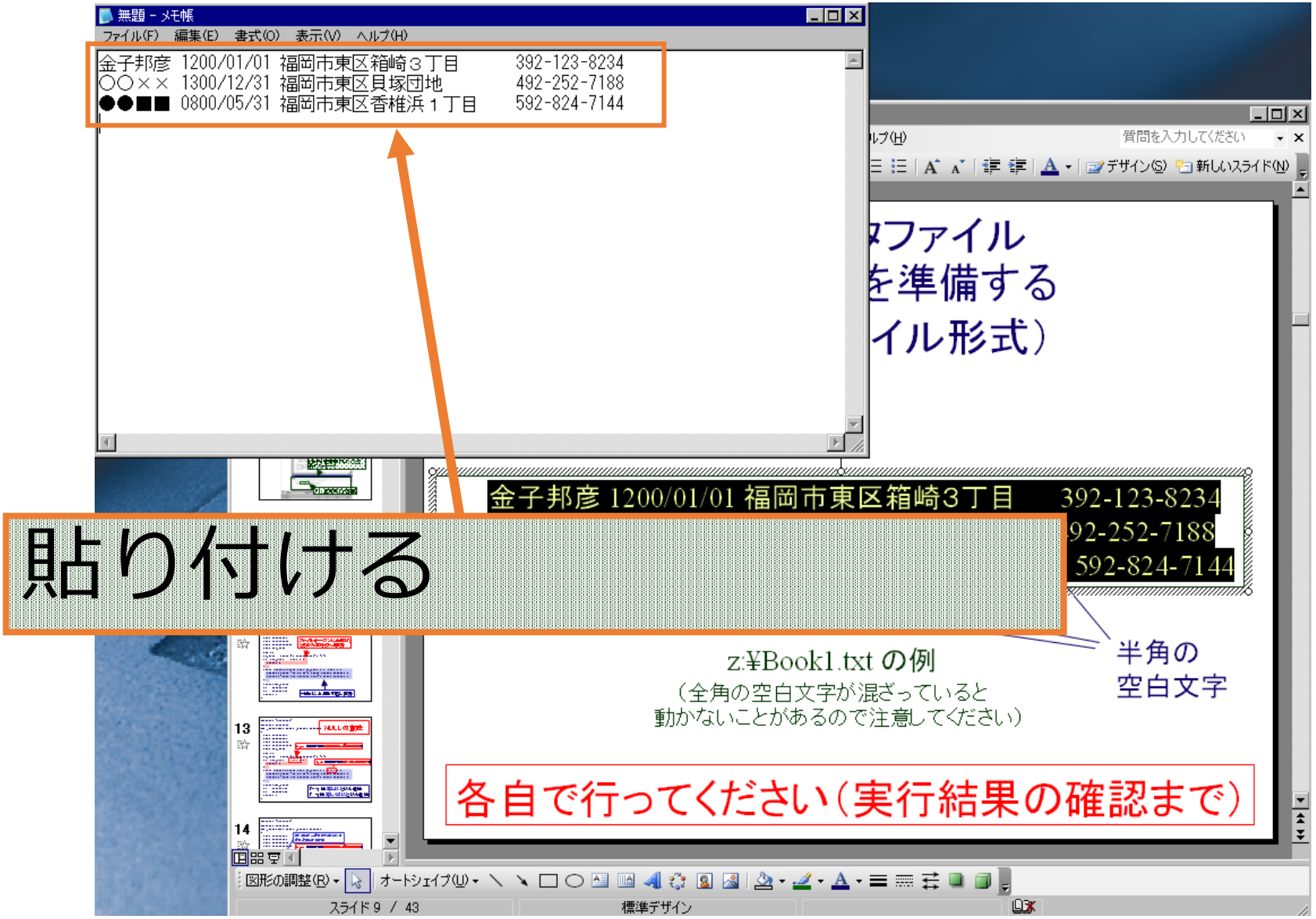
金子邦彦 1200/01/01 福岡市東区箱崎３丁目　　392-123-8234
○○×× 1300/12/31 福岡市東区貝塚団地　　492-252-7188
●●■■ 0800/05/31 福岡市東区香椎浜１丁目　592-824-7144

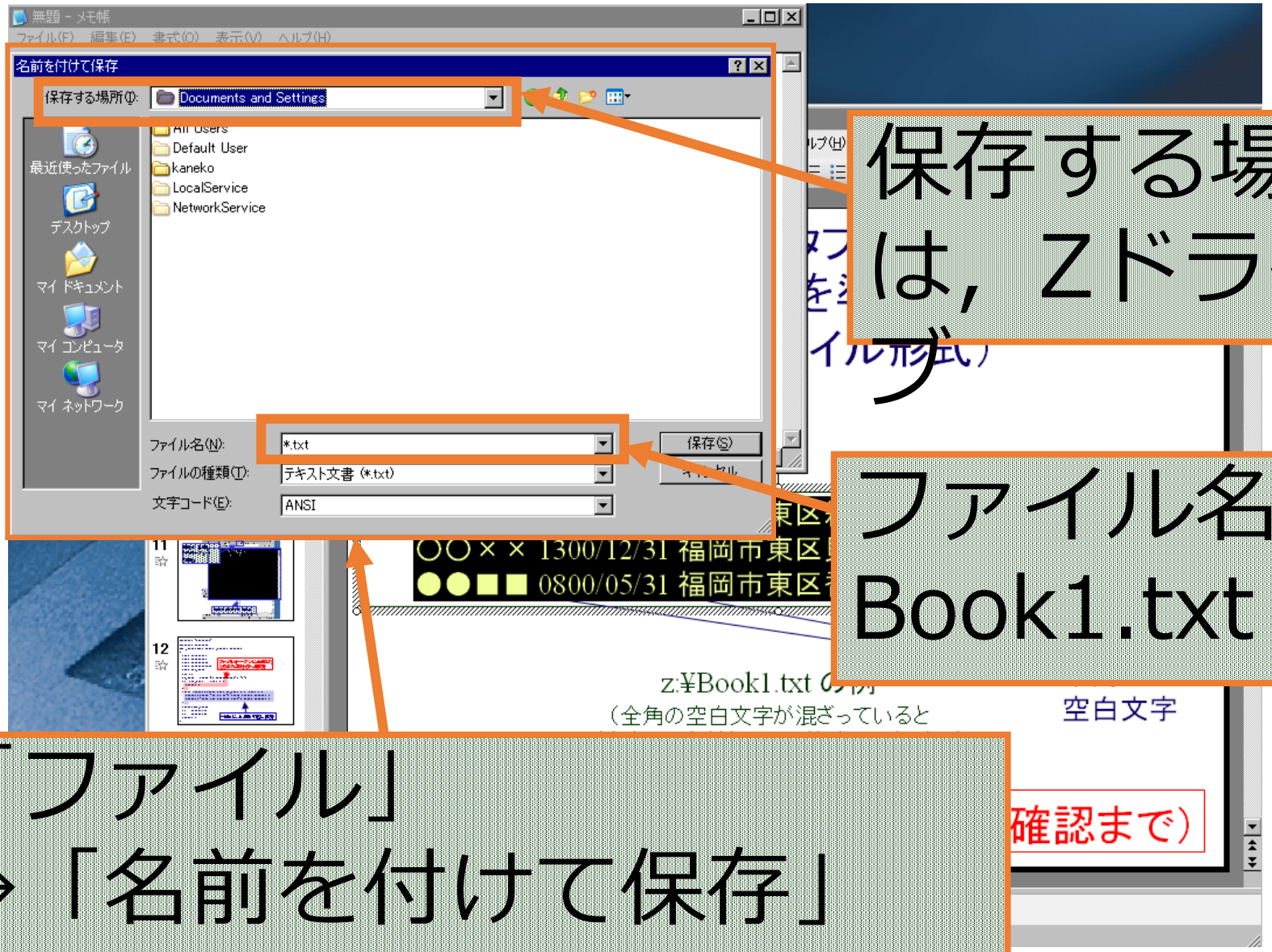
半角の
空白文字

d:¥Book1.txt の例
（全角の空白文字が混ざっていると
動かないことがあるので注意してください）

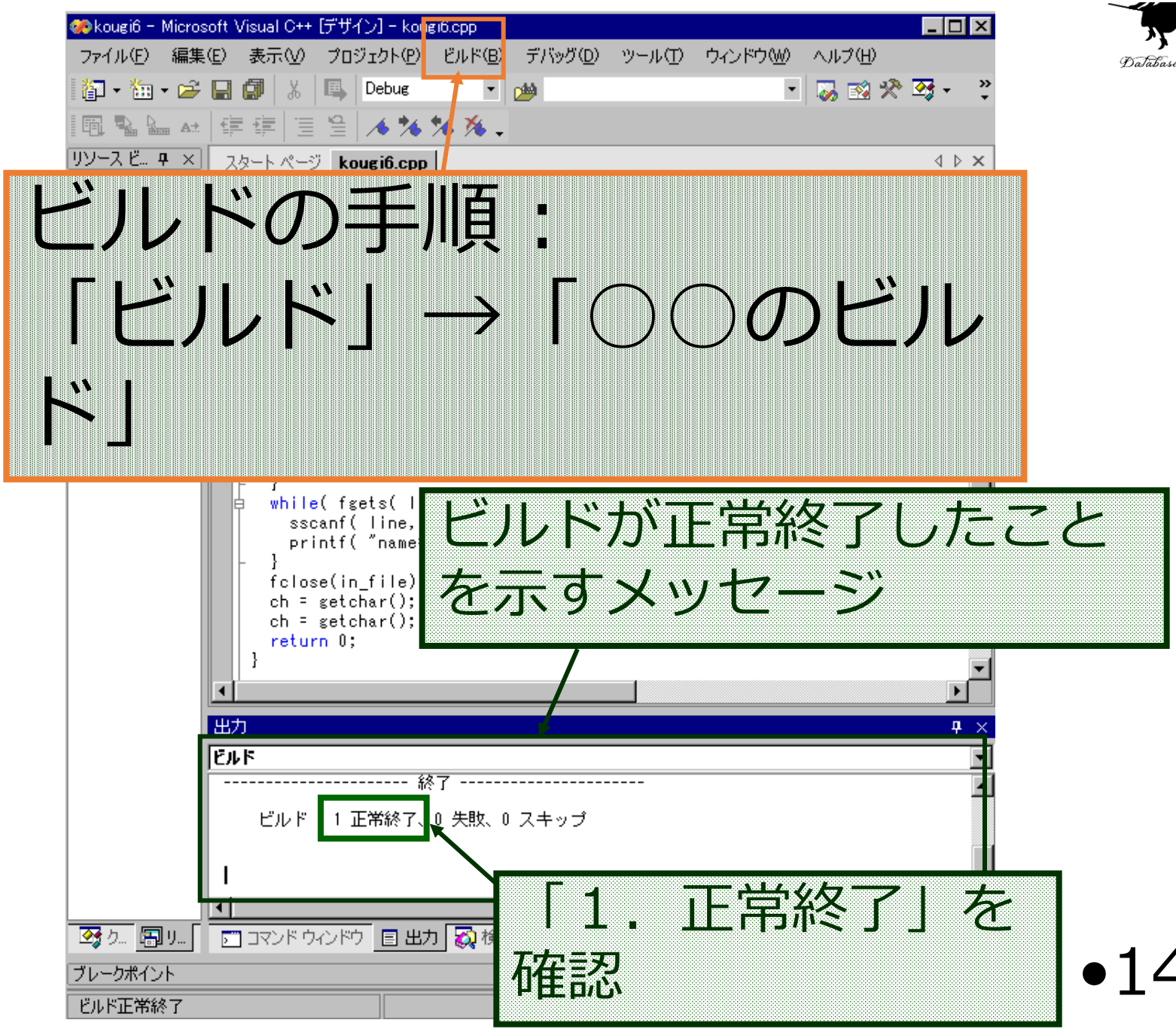

無題 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
ファイル
を準備する
イル形式）
金子邦彦 1200/01/01 福岡市東区箱崎3丁目 392-123-8234
○○×× 1300/12/31 福岡市東区貝塚団地 492-252-7188
592-824-7144
半角の
空白文字
（全角の空白文字が混ざっていると
動かないことがあるので注意してください）
各自で行ってください（実行結果の確認まで）
「メモ帳」を起動している
スライド 9 / 43
標準デザイン
日本語


無題 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
金子邦彦 1200/01/01 福岡市東区箱崎3丁目 392-123-8234
○○×× 1300/12/31 福岡市東区貝塚団地 492-252-7188
●●■■ 0800/05/31 福岡市東区香椎浜1丁目 592-824-7144
半角の
空白文字
コピーして ・・・
スライド 9 / 43
標準デザイン

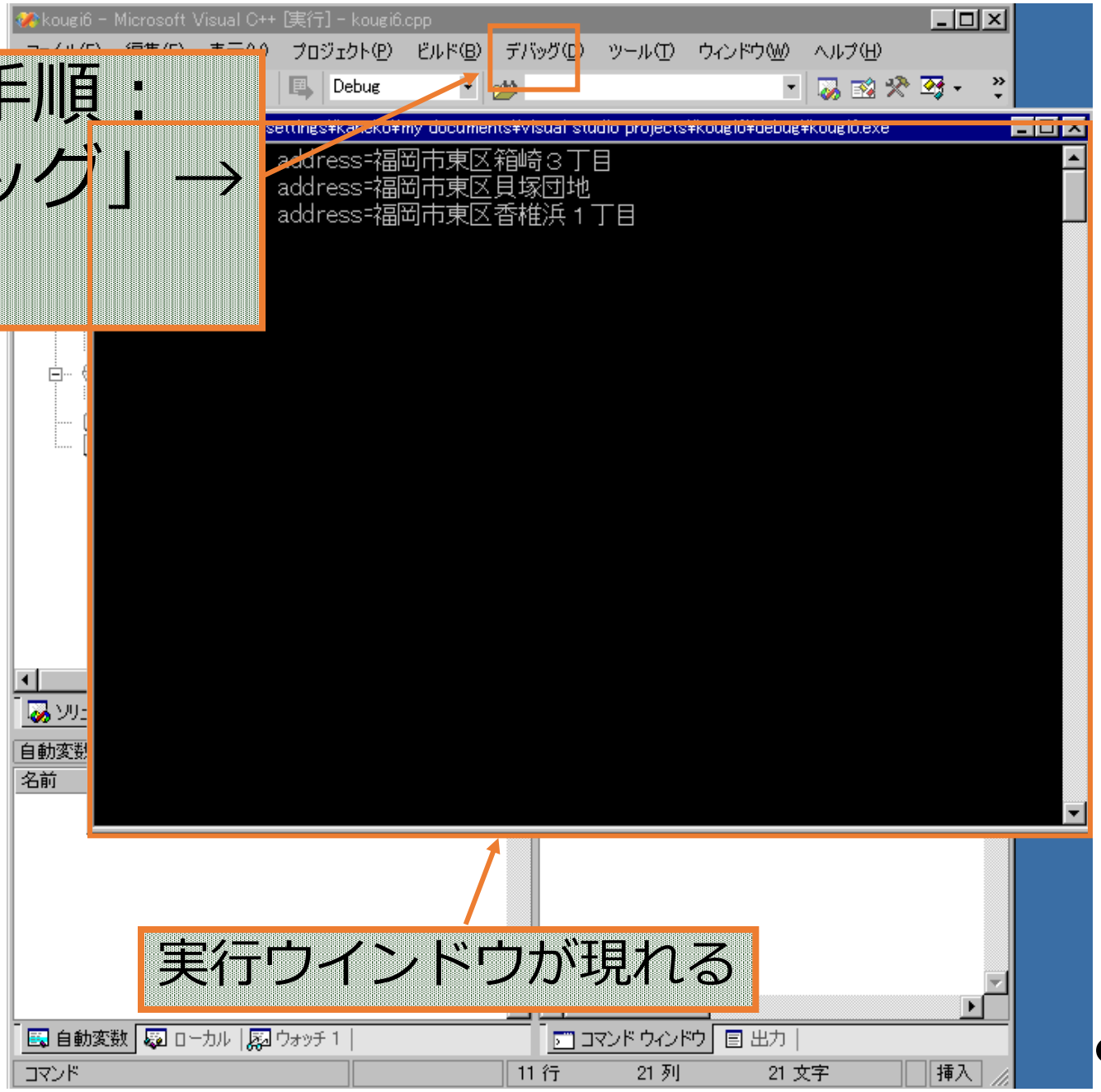

無題 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
金子邦彦 1200/01/01 福岡市東区箱崎３丁目 392-123-8234
○○××　1300/12/31 福岡市東区貝塚団地 492-252-7188
●●■■　0800/05/31 福岡市東区香椎浜１丁目 592-824-7144
金子邦彦 1200/01/01 福岡市東区箱崎3丁目 392-123-8234
92-252-7188
592-824-7144
貼り付ける
z:¥Book1.txt の例
（全角の空白文字が混ざっていると
動かないことがあるので注意してください）
半角の
空白文字
各自で行ってください（実行結果の確認まで）
スライド 9 / 43
標準デザイン


保存する場所は，Zドライブ
ファイル名は Book1.txt
「ファイル」
→「名前を付けて保存」
名前を付けて保存
保存する場所(I):
Documents and Settings
Default User
kaneko
LocalService
NetworkService
ファイル名(N):
*.txt
ファイルの種類(T):
テキスト文書 (*.txt)
文字コード(E):
ANSI
保存(S)
z:¥Book1.txt
(全角の空白文字が混ざっていると
空白文字

ビルド後の画面


kougi6 - Microsoft Visual C++ [デザイン] - kougi6.cpp
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
Debug
kougi6.cpp
ビルドの手順：
「ビルド」→「○○のビルド」
while(fgets(l
sscanf(line,
printf("name
}
fclose(in_file)
ch = getchar();
ch = getchar();
return 0;
}
ビルドが正常終了したことを示すメッセージ
出力
ビルド
終了
ビルド 1 正常終了、0 失敗、0 スキップ
コマンド ウィンドウ
出力
ブレークポイント
ビルド正常終了
「１．正常終了」を確認

実行中の画面


実行の手順：
「デバッグ」→
「実行」
kougi6 - Microsoft Visual C++ [実行] - kougi6.cpp
プロジェクト(P)
ビルド(B)
デバッグ(D)
ツール(T)
ウィンドウ(W)
ヘルプ(H)
Debug
address=福岡市東区箱崎３丁目
address=福岡市東区貝塚団地
address=福岡市東区香椎浜１丁目
実行ウインドウが現れる
自動変数
ローカル
ウォッチ 1
コマンド ウィンドウ
出力
コマンド
11 行
21 列
21 文字
挿入

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
  char line[100];
  char name[100];
  char birth[100];
  char address[100];
  FILE *in_file;
  int ch;
  in_file = fopen("d:¥¥Book1.txt", "r");
  if ( in_file == NULL ) {
    return 0;
  }
  while( fgets( line, 100, in_file ) != NULL ) {
    sscanf_s( line, "%s %s %s", name, birth, address );
    printf( "name=%s, address=%s¥n", name, address );
  }
  fclose(in_file);
  ch = getchar();
  ch = getchar();
  return 0;
}
```

fopen 関数で，NULL は「ファイルオープンの失敗」

ファイルオープンに失敗したときのみ実行される部分

while による繰り返し

fgets 関数で，NULL は「ファイルの終わり」

ファイル操作

- ファイルのオープンとクローズ
  - fopen　　ファイルの読み書きを行う前に、ファイルはオープンされねばならない
  - fclose　　ファイルの読み書きが終わったら、ファイルはクローズされねばならない
- ファイルの読み込み
  - fgets　　１行単位の読み込み
  - fread　　バイト単位での読み込み（１バイト，複数バイト）
- ファイルの書き出し
  - fputs　　１行単位での書き出し
  - fwrite　バイト単位での書き出し（１バイト，複数バイト）
  - fprintf　整形しての書き出し

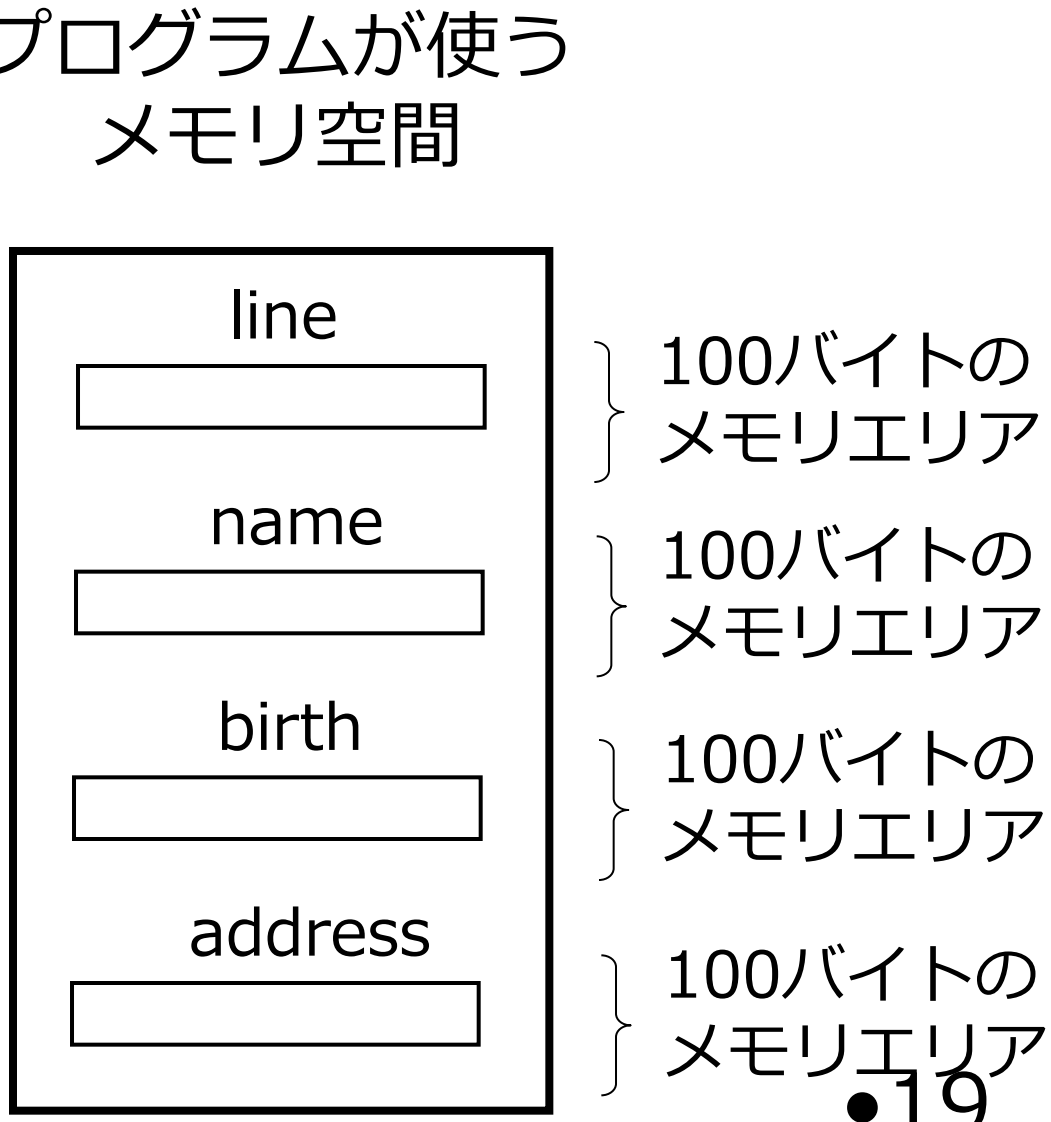
オープンモード

- “r” モード
  - 読み込みモード
  - 引数fileで指定したファイルが存在しないか，読み込み不可能な場合には，オープンすることができない.
- “w” モード
  - 書き出しモード
  - 引数fileで指定したファイルが存在しない場合には，ファイルが新たに作成される．ファイルがすでに存在した場合，ファイル中のデータはすべて捨てられる（ファイルの長さは０になる）．

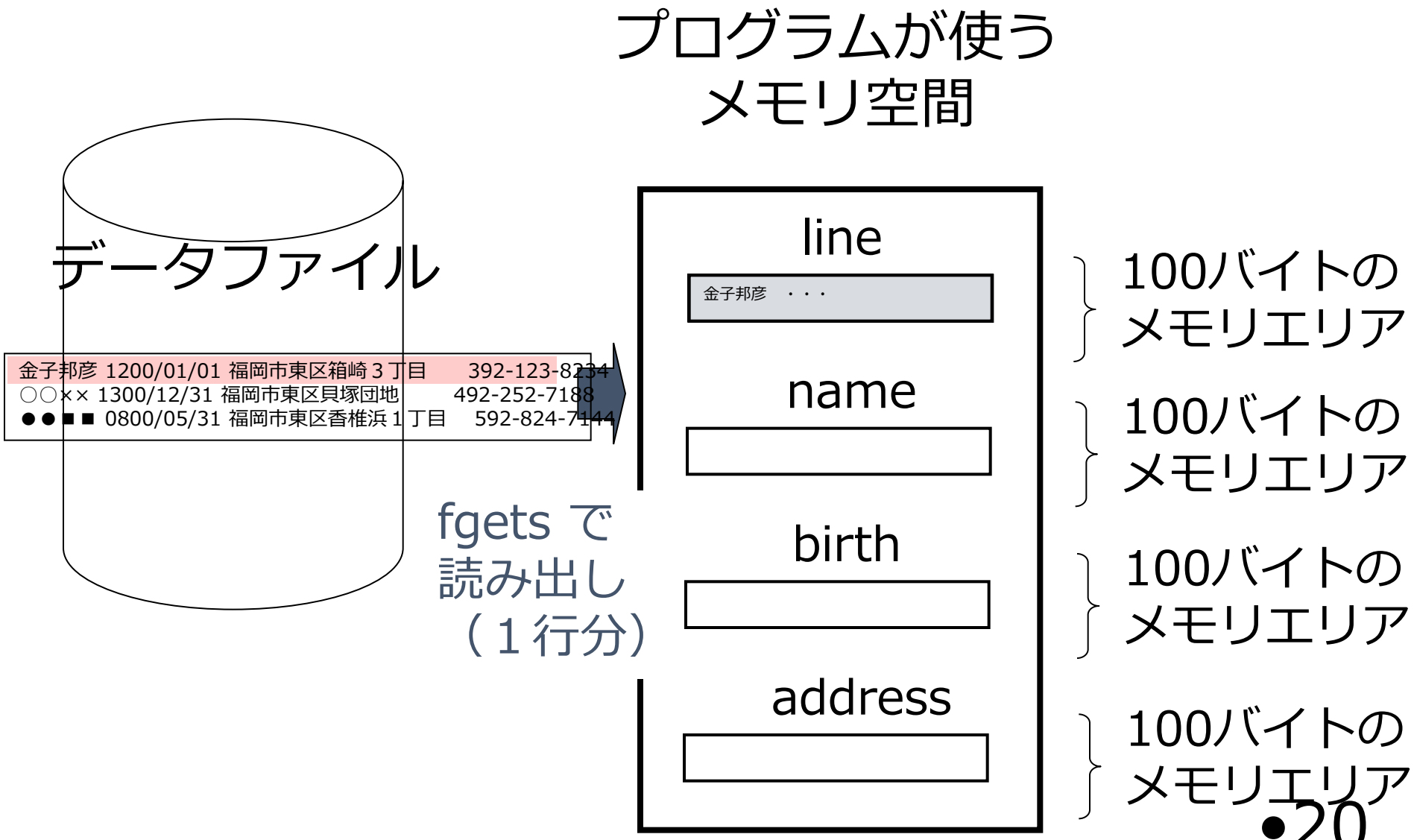
```
in_file = fopen("d:¥¥Book1.txt", "r");
```

ファイル名（文字列）　オープンモード（文字列）

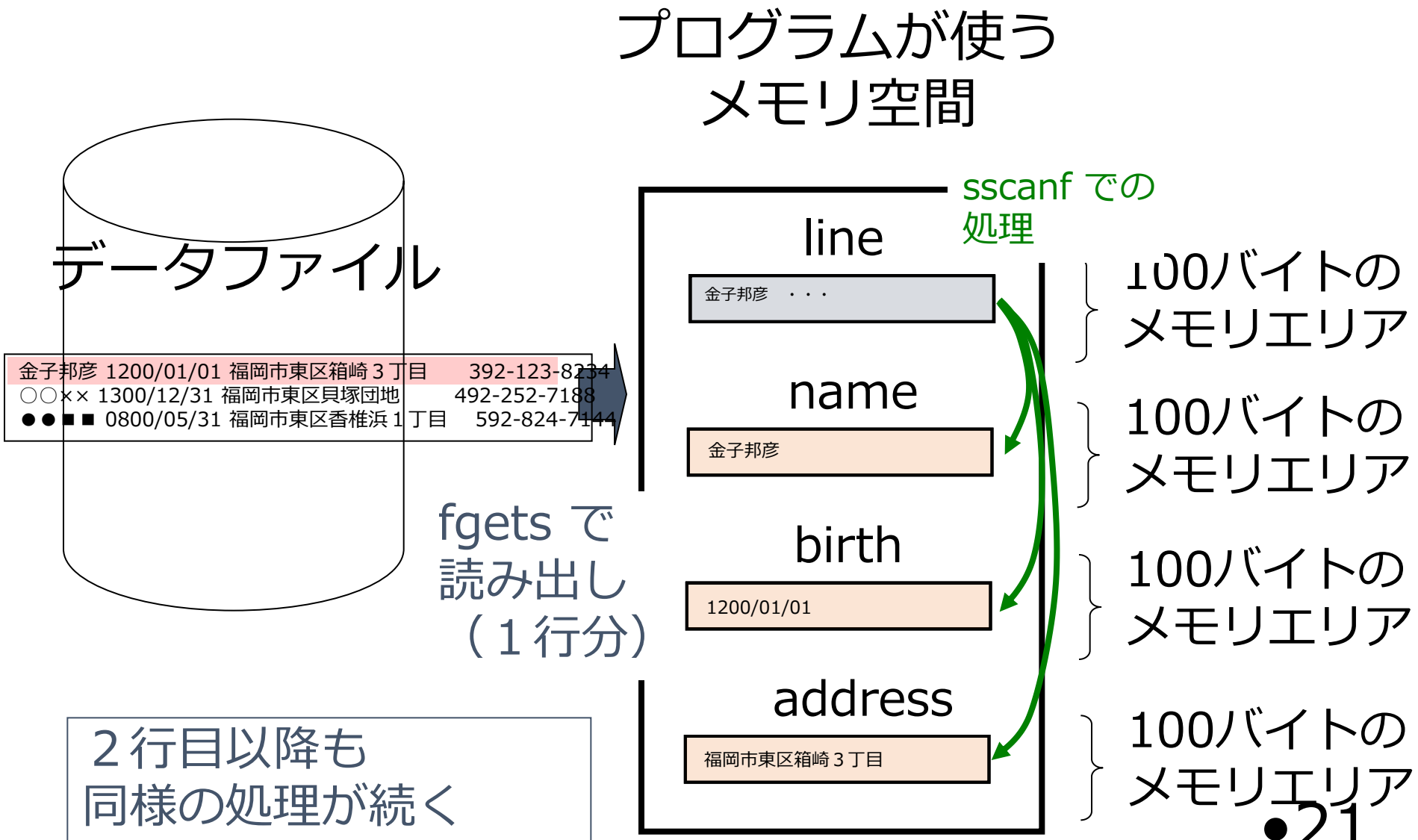
例題１のプログラムが行っていること



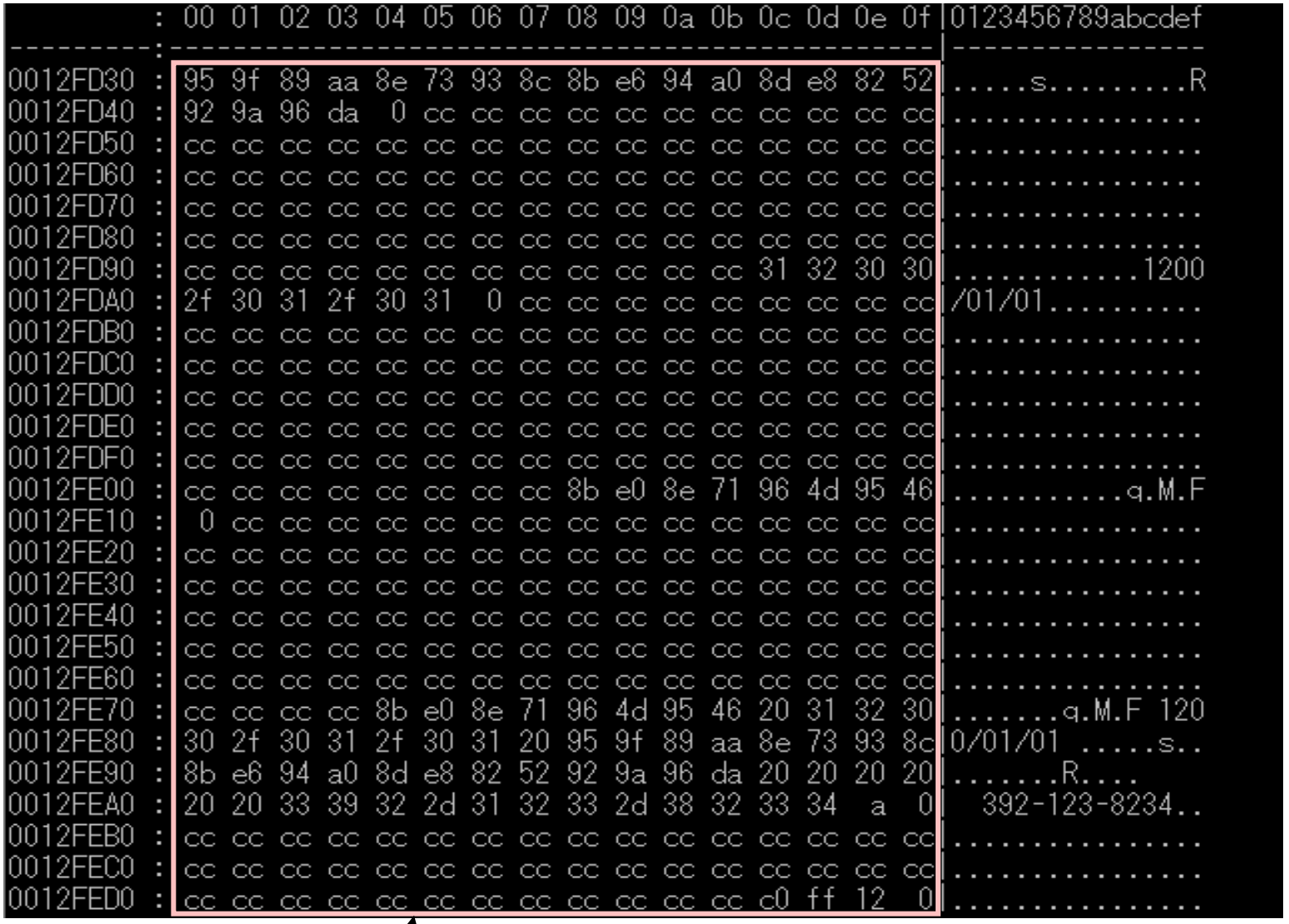
例題１のプログラムが行っていること



例題１のプログラムが行っていること



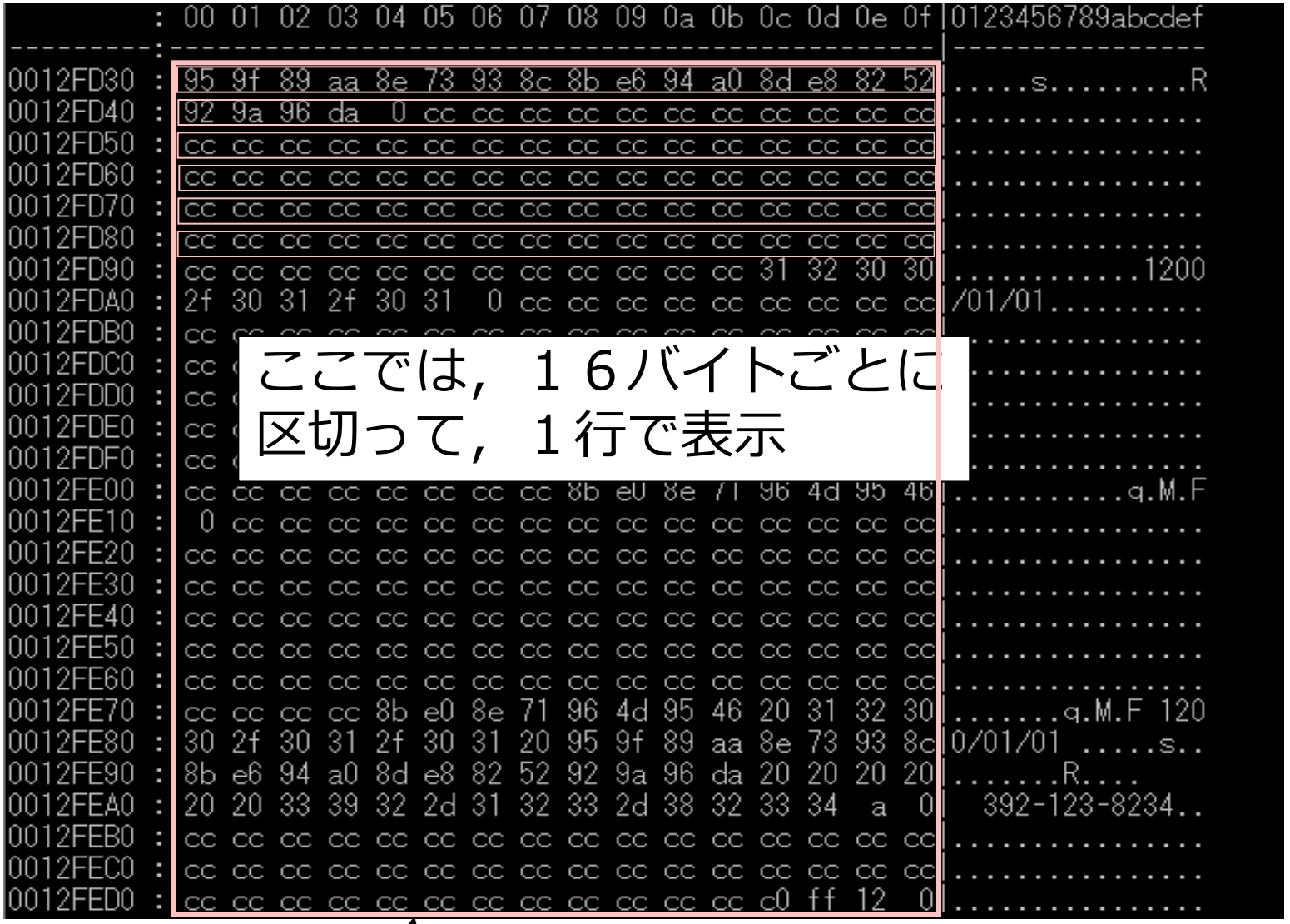
実際のメモリの中身



```
         : 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f |0123456789abcdef
---------:-------------------------------------------------|----------------
0012FD30 : 95 9f 89 aa 8e 73 93 8c 8b e6 94 a0 8d e8 82 52 |.....s.........R
0012FD40 : 92 9a 96 da  0 cc cc cc cc cc cc cc cc cc cc cc |................
0012FD50 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FD60 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FD70 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FD80 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FD90 : cc cc cc cc cc cc cc cc cc cc cc cc 31 32 30 30 |............1200
0012FDA0 : 2f 30 31 2f 30 31  0 cc cc cc cc cc cc cc cc cc |/01/01..........
0012FDB0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FDC0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FDD0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FDE0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FDF0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE00 : cc cc cc cc cc cc cc cc 8b e0 8e 71 96 4d 95 46 |...........q.M.F
0012FE10 :  0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE20 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE30 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE40 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE50 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE60 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FE70 : cc cc cc cc 8b e0 8e 71 96 4d 95 46 20 31 32 30 |.......q.M.F 120
0012FE80 : 30 2f 30 31 2f 30 31 20 95 9f 89 aa 8e 73 93 8c |0/01/01 .....s..
0012FE90 : 8b e6 94 a0 8d e8 82 52 92 9a 96 da 20 20 20 20 |.......R....
0012FEA0 : 20 20 33 39 32 2d 31 32 33 2d 38 32 33 34  a  0 |  392-123-8234..
0012FEB0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FEC0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |................
0012FED0 : cc cc cc cc cc cc cc cc cc cc cc cc cc c0 ff 12  0|................
```

メモリの中身を画面表示したもの

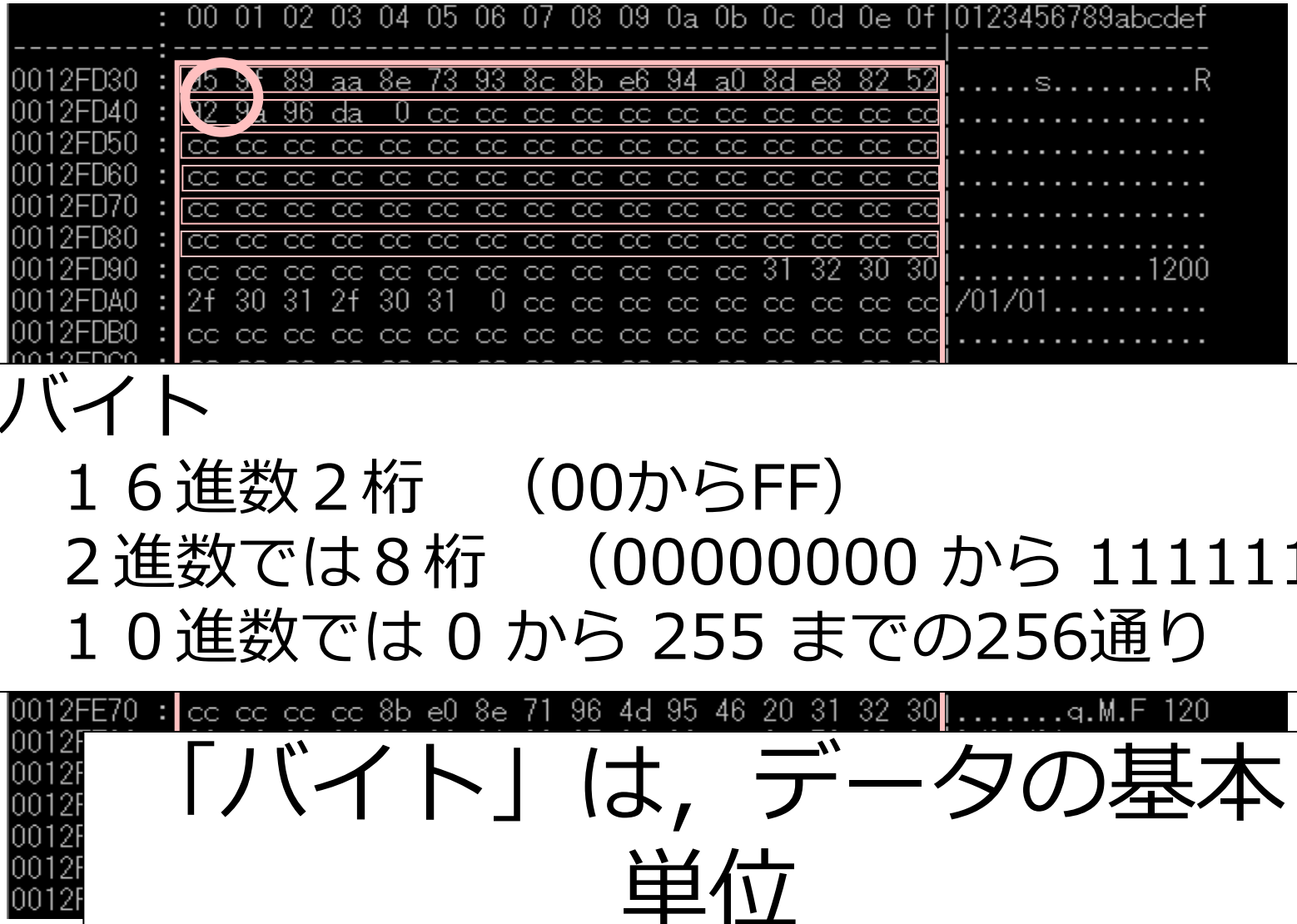
実際のメモリの中身



ここでは，１６バイトごとに区切って，１行で表示

メモリの中身を画面表示したもの

実際のメモリの中身



１バイト

１６進数２桁　（00からFF）

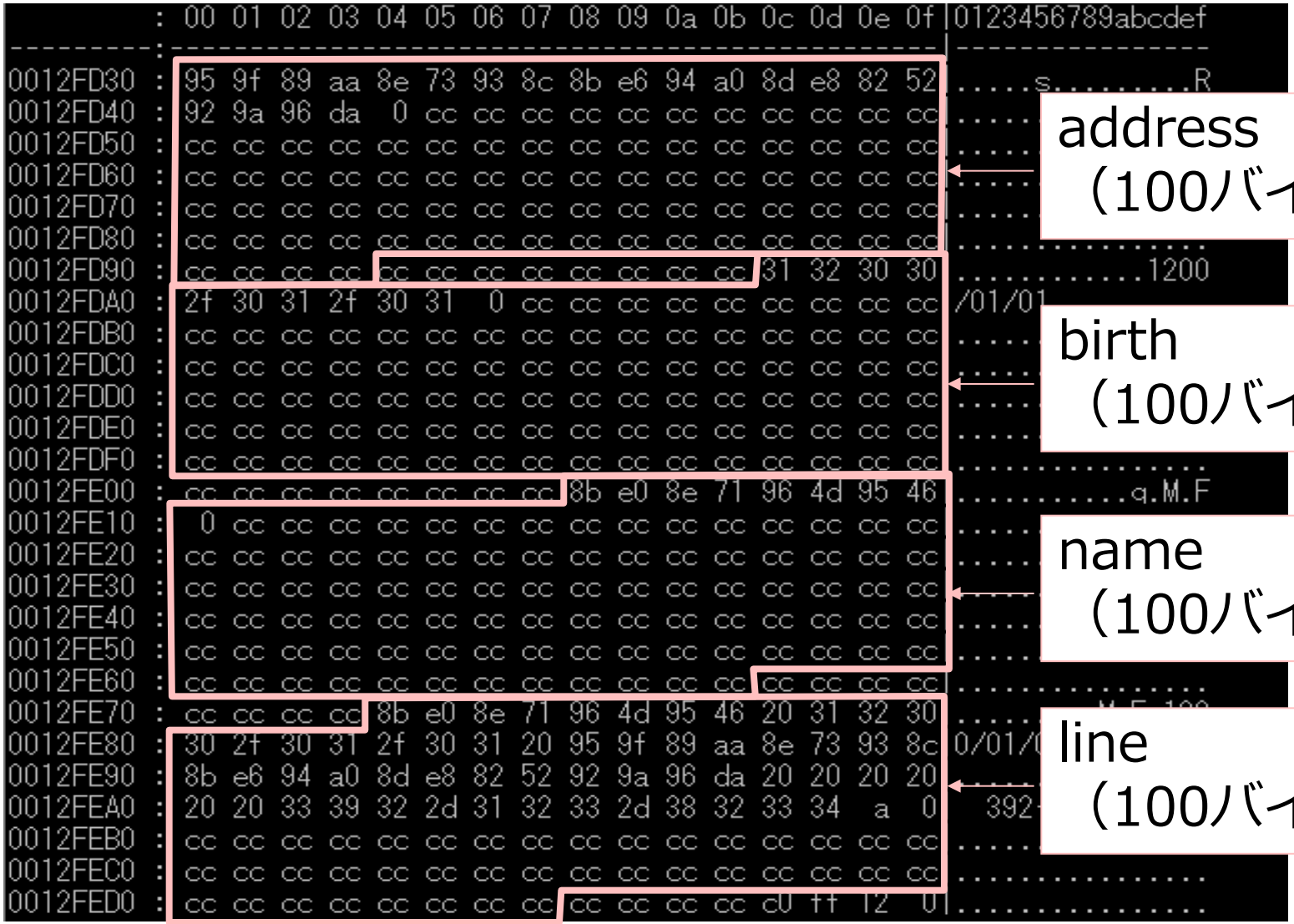
２進数では８桁　（00000000 から 11111111）

１０進数では 0 から 255 までの256通り

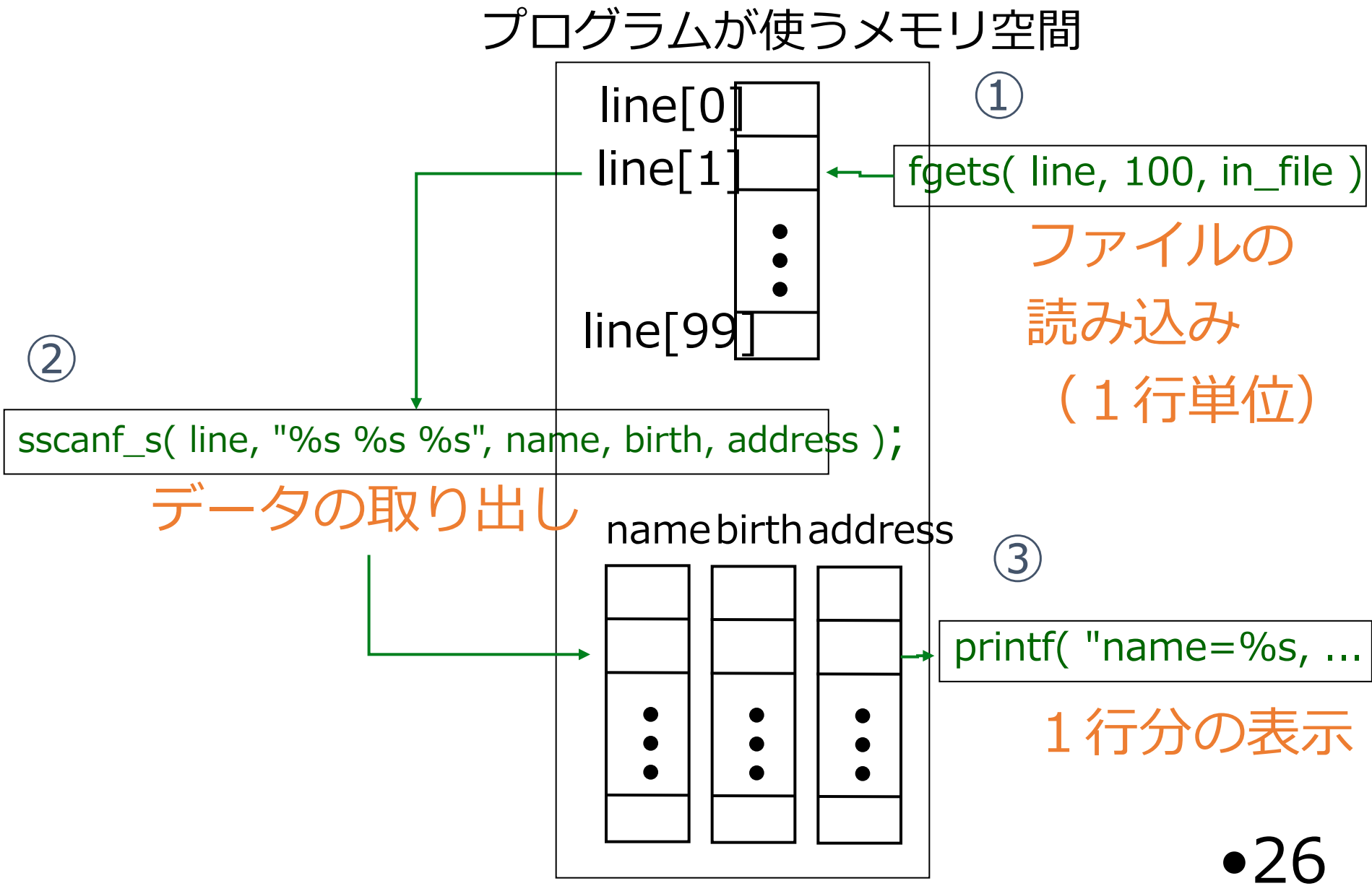
「バイト」は，データの基本単位

メモリの中身を画面表示したもの

実際のメモリの中身

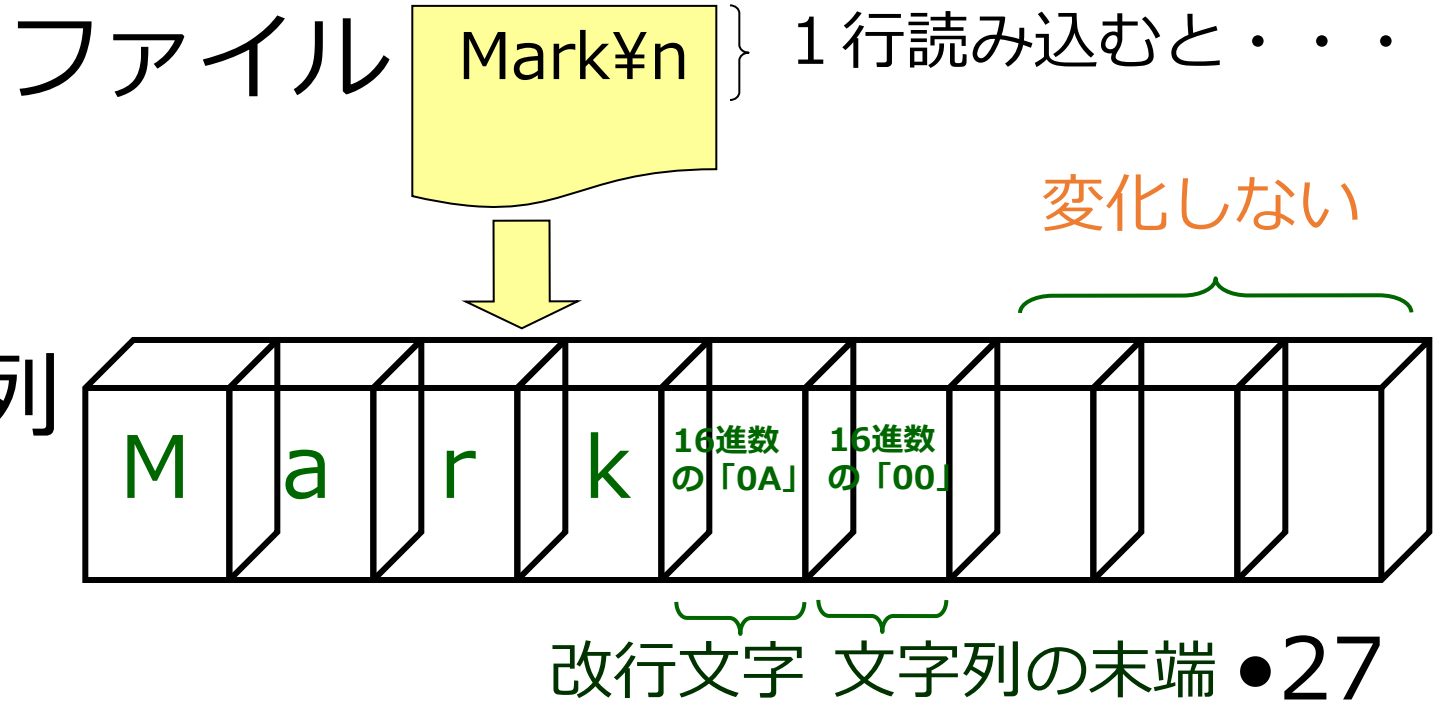

: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 0123456789abcdef
0012FD30 : 95 9f 89 aa 8e 73 93 8c 8b e6 94 a0 8d e8 82 52 .....s.....R
0012FD40 : 92 9a 96 da 0 cc cc cc cc cc cc cc cc cc cc cc .....
0012FD50 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FD60 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FD70 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FD80 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FD90 : cc cc cc cc cc cc cc cc cc cc cc cc 31 32 30 30 1200
0012FDA0 : 2f 30 31 2f 30 31 0 cc cc cc cc cc cc cc cc cc /01/01
0012FDB0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FDC0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FDD0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FDE0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FDF0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE00 : cc cc cc cc cc cc cc cc 8b e0 8e 71 96 4d 95 46 .....q.M.F
0012FE10 : 0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE20 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE30 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE40 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE50 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE60 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FE70 : cc cc cc cc 8b e0 8e 71 96 4d 95 46 20 31 32 30 .....
0012FE80 : 30 2f 30 31 2f 30 31 20 95 9f 89 aa 8e 73 93 8c 0/01/0
0012FE90 : 8b e6 94 a0 8d e8 82 52 92 9a 96 da 20 20 20 20 .....
0012FEA0 : 20 20 33 39 32 2d 31 32 33 2d 38 32 33 34 a 0 392-
0012FEB0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FEC0 : cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc .....
0012FED0 : cc cc cc cc cc cc cc cc cc cc cc cc c0 ff 12 0 .....
address
（100バイト）
birth
（100バイト）
name
（100バイト）
line
（100バイト）

プログラムとデータ


プログラムが使うメモリ空間
line[0]
line[1]
line[99]
①
fgets(line, 100, in_file)
ファイルの
読み込み
（１行単位）
②
sscanf_s(line, "%s %s %s", name, birth, address);
データの取り出し
name
birth
address
③
printf("name=%s, ...
１行分の表示

fgetsの振る舞い

- ファイルの１行読み込み
  - ファイルの一行分を読み込んで、末端の¥0を付ける
  - ファイルには、各行の終わりに、改行文字（¥n)が付いている（目には見えない）
  - 読み込み先（文字の配列）のサイズが、ファイルの１行の長さより長いときは、「残りの部分」は変化しない

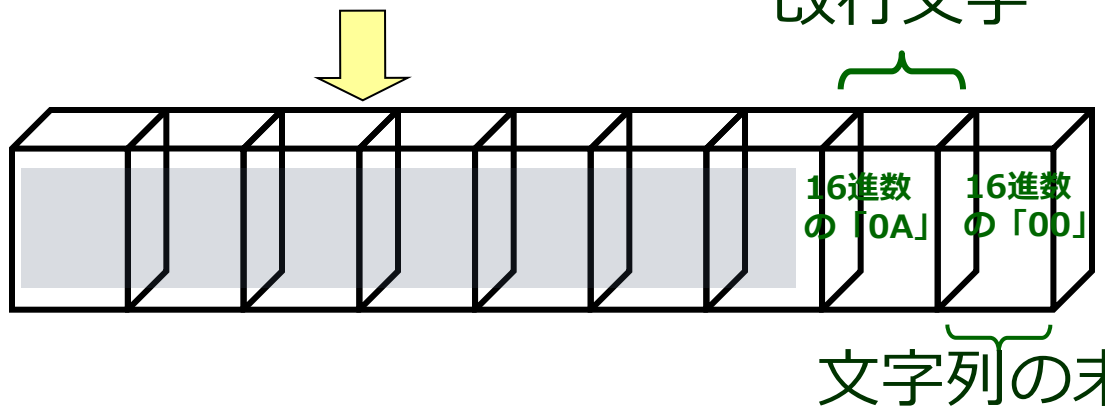


fgets での「100」

```
fgets( line, 100, in_file )
```

１００バイトに達したら
行末になっていなくても読み込み終了せよ

ファイル



文字の配列

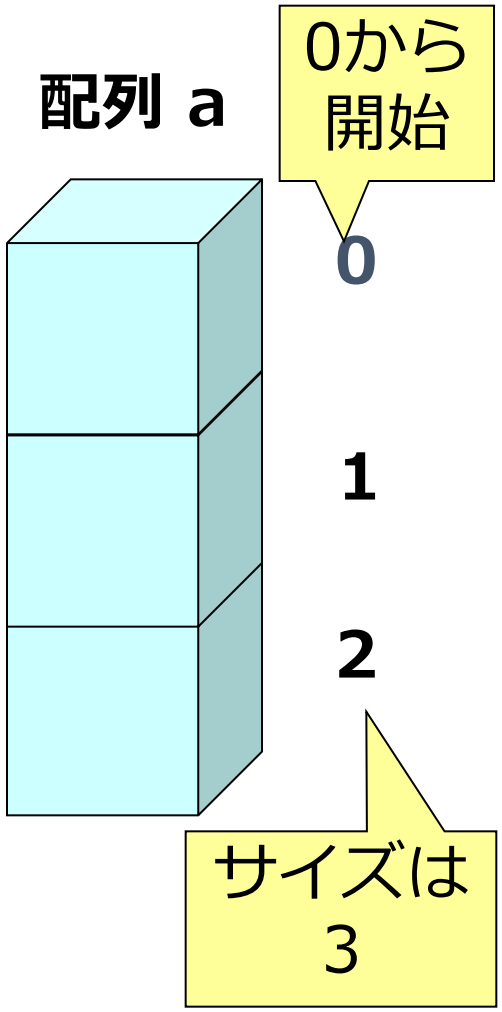
配列のサイズが１００ならば、読み込めるデータの本体（「改行文字」を除く）は，最大で９８文字まで

配列

一次元配列

- 配列の要素には型がある
- 0から始まる番号がついたデータの並び

例） int, char, double など



例題２．ベクトルの内積

- ベクトル（1.9, 2.8, 3.7）と，ベクトル（4.6, 5.5, 6.4）の内積を表示するプログラムを作る
  - ２つのベクトルの内積の計算のために，サイズ３の一次元配列を２つ使う

例題２：ベクトルの内積

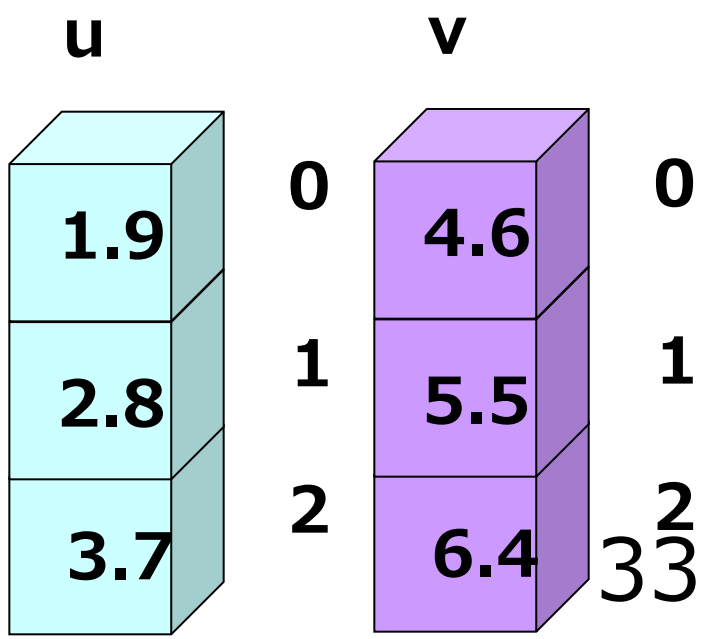
```
#include "stdio.h"
#include <math.h>
int main()
{
  int i;
  double ip = 0.0;
  double u[]={1.9, 2.8, 3.7};
  double v[]={4.6, 5.5, 6.4};
  int ch;
  for (i=0; i<3; i++) {
    ip = ip + u[i]*v[i];
  }
  printf("内積=%f¥n", ip);
  ch = getchar();
  ch = getchar();
  return 0;
}
```

浮動小数を扱う double 型

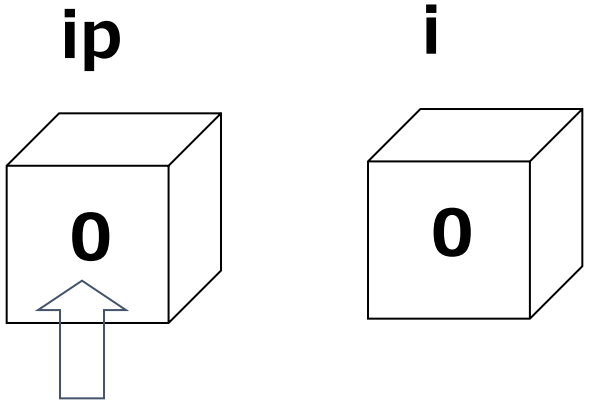
```
#include "stdio.h"
#include <math.h>
int main()
{
  int i;
  double ip = 0.0;
  double u[]={1.9, 2.8, 3.7};
  double v[]={4.6, 5.5, 6.4};
  int ch;
  for (i=0; i<3; i++) {
    ip = ip + u[i]*v[i];
  }
  printf("内積=%f¥n", ip);
  ch = getchar();
  ch = getchar();
  return 0;
}
```

メモリ確保および初期化が行われる。

変数 u, v は，浮動小数を要素とする配列で，サイズは 3

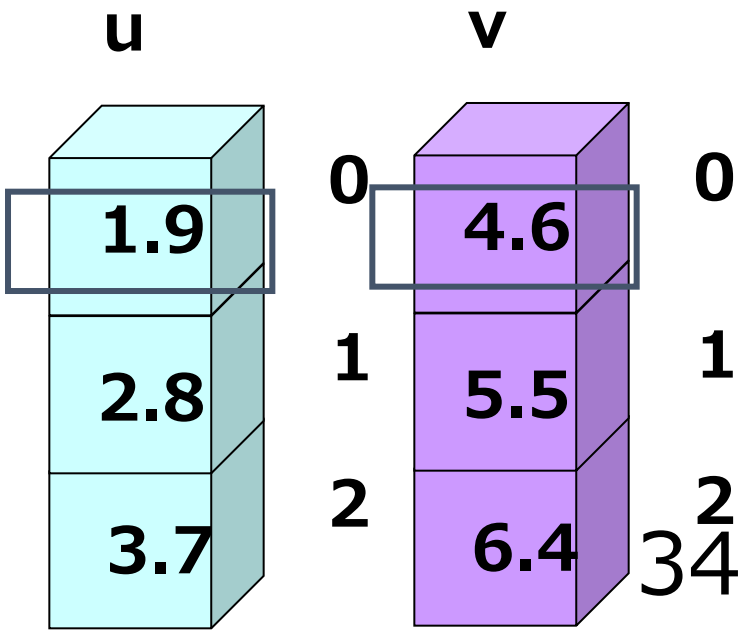


i = 0, 1, 2 での繰り返し
（3回繰り返し）
最初は i = 0

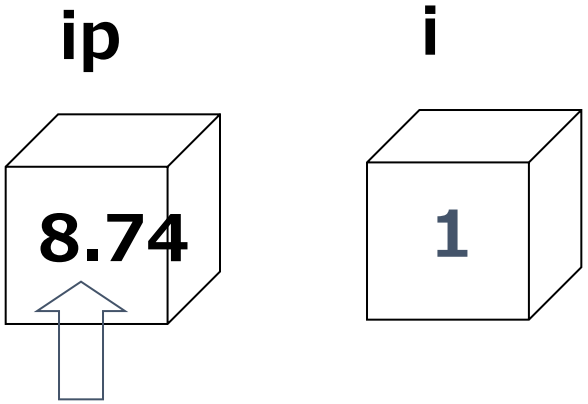



```
for (i=0; i<3; i++) {
  ip = ip + u[i]*v[i];
}
```

i = 0 のときは
ip + u[0]*v[0]

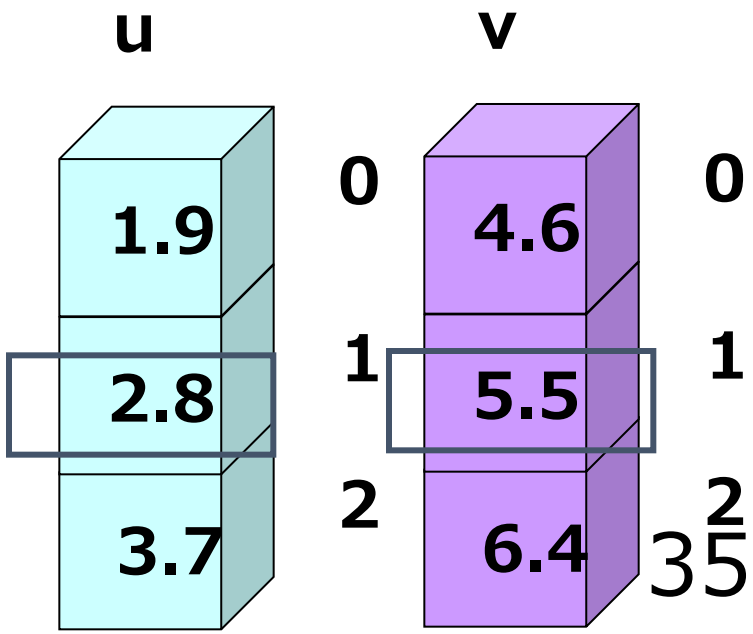


i = 0, 1, 2 での繰り返し
（3回繰り返し）
次は i = 1

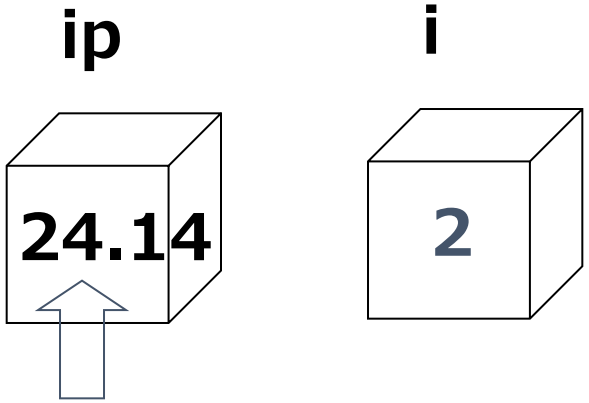



```
for (i=0; i<3; i++) {
  ip = ip + u[i]*v[i];
}
```

i = 1 のときは
ip + u[1]*v[1]



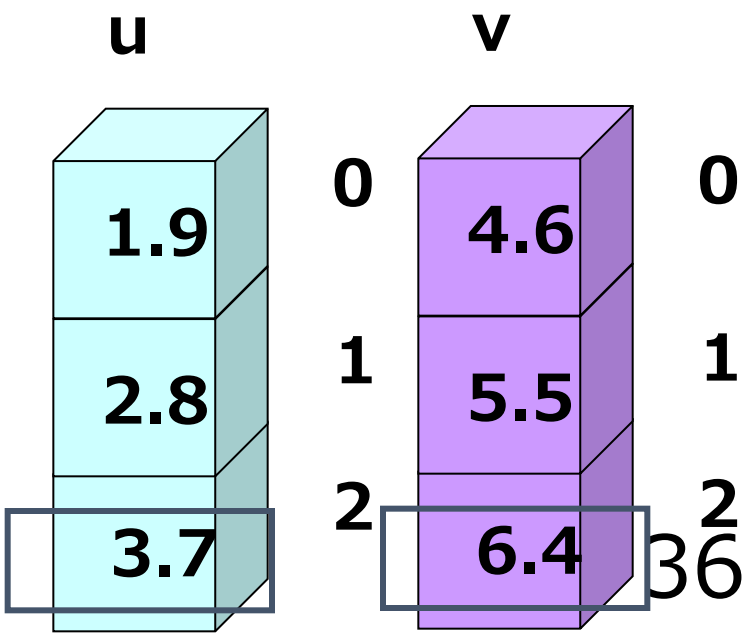
i = 0, 1, 2 での繰り返し
（3回繰り返し）
次は i = 2



24.14 + 3.7 * 6.4

```
for (i=0; i<3; i++) {
  ip = ip + u[i]*v[i];
}
```

i = 2 のときは
ip + u[2]*v[2]



ベクトルの内積

```
for (i=0; i<3; i++) {
  ip = ip + u[i]*v[i];
}
```

	i の値	繰り返し条件式が成り立つか	ip の値
繰り返し1回目	i = 0	i < 3 が成り立つ	ip = ip + u[0] * v[0];

つまり ip の値は u[0]*v[0]

繰り返し2回目	i = 1	i < 3 が成り立つ	ip = ip + u[1] * v[1];

つまり ip の値は u[0]*v[0] + u[1]*v[1]

繰り返し3回目	i = 2	i < 3 が成り立つ	ip = ip + u[2] * v[2];

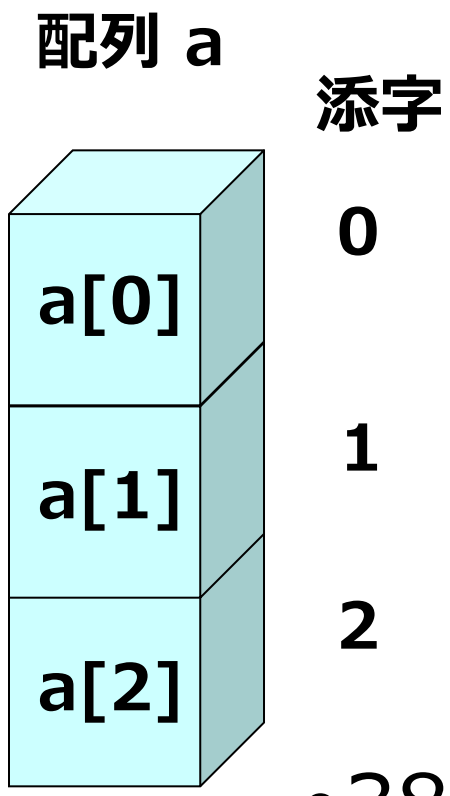
つまり ip の値は u[0]*v[0] + u[1]*v[1]
+u[2]*v[2]

繰り返し4回目	i = 3	i < 3 が成り立たない

配列の使い方

- 添字をつけて、普通の変数のように使う。

```
int a[3];
a[0] = i;
a[1] = 5;
fscanf(“%d”,&a[2]);
j = a[2];
printf(“%d %d¥n”,a[0],a[1]);
```



次のプログラムを実行してみなさい

```
#include "stdio.h"
#include <math.h>
int main()
{
  int i;
  double ip = 0.0;
  double u[]={1.9, 2.8, 3.7};
  double v[]={4.6, 5.5, 6.4};
  int ch;
  for (i=0; i<3; i++) {
    ip = ip + u[i]*v[i];
  }
  printf("内積=%f¥n", ip);
  ch = getchar();
  ch = getchar();
  return 0;
}
```

例題３．棒グラフを描く

- 整数の配列から，その棒グラフを表示するプログラムを作る.
  - ループの入れ子で，棒グラフの表示を行う

例題３：棒グラフ

```
#include "stdio.h"
#include <math.h>
int main()
{
  int i;
  int j;
  int ch;
  int a[]={6,4,7,1,5,3,2};
  for (i=0; i<7; i++) {
    for (j=0; j<a[i]; j++) {
      printf("*");
    }
    printf("¥n");
  }
  ch = getchar();
  ch = getchar();
  return 0;
}
```

配列の宣言

配列からの読み出し

棒グラフを書く

実行結果の例

```
******

****

********

*

******

***

**
```

多重ループ

- ループを入れ子構造にする。

外側のループ

```
for (i=0 ; i < n ;i++)
{
    内側のループ
    for (j=0 ; j < m ;j++)
    {

    }
}
```

i=0

i=1

i=n-1

i	j
0	0
0	1
0	2
⋮	⋮
0	m-1
1	0
1	1
1	2
⋮	⋮
1	m-1
⋮	⋮
n-1	m-1