Programming II 0x05

多次元配列(2011.05.19)

塩澤秀和 http://vilab.org

関数に配列を渡す（復習）

```
/* リスト5-17を一部変更(p.108) */
#include <stdio.h>
int gokei(int b[5]);

int main(void)
{
  int a[5], sum;
  int i;

  for(i = 0; i < 5; i++){
    printf("a[%d]=", i);
    scanf("%d", &a[i]);
  }

  sum = gokei( a );
  printf("sum=%d\n", sum);
  return 0;
}

/* 総和を計算する関数 */
int gokei(int b[5])
{
  int i;
  int sum = 0;

  for(i = 0; i < 5; i++) {
    sum += b[i];
  }
  return sum;
}
```

実引数では配列名だけを書く

仮引数では同じ型の配列変数を宣言して受け取る

関数で配列を書き換える(復習)

```
/* 関数内で配列の全要素を2倍する */
/* リスト5-18 (p.110) を一部変更    */
#include <stdio.h>
/* 定数マクロの利用例 */
#define N 3

void multi2(int b[N]);

int main(void)
{
  int i, a[N];

  for(i = 0; i < N; i++) {
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
  }

  multi2( a );

  for(i = 0; i < N; i++) {
    printf("a[%d] = %d\n",
           i, a[i]);
  }
  return 0;
}
```

aの内容が変わっていることを確認

```
/* 配列の全要素を2倍 */
void multi2(int b [N])
{
  int i;

  for(i = 0; i < N; i++) {
    b[i] = b[i] * 2;
  }
}
```

仮引数

実引数

関数内で配列の中身を書き換えられる！

C言語の文字列（復習）

- 文字列は文字（char）の配列（p.114）
  - 先頭の文字から1文字ずつ配列に入れて、最後の文字の次に、ヌル文字（'\0'＝文字コード0の文字）をつける
  - char str[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };　← 5文字＋'\0'
  - 最低限、文字列の長さ＋1文字分の長さの配列が必要

- 文字列の初期化（p.115）
  - char str[6] = "Hello";
  - char str[ ] = "Hello";
  - char str[10] = "Hello";

H	e	l	l	o	\0

H	e	l	l	o	\0	空	空	空	空

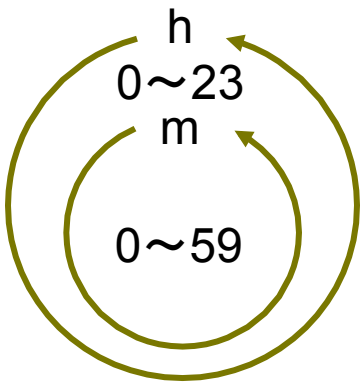
  - 「= "文字列"」で初期化できるのは、変数定義のときだけ
    - 初期化以外では標準ライブラリ関数の strcpy でコピーする

多重ループ

- ループのなかでループする

h が1増えるごとに
m は60回ループする

```
for (h = 0; h < 24; h++) {
  for (m = 0; m < 60; m++) {
    printf("%02d時%02d分\n", h, m);
  }
}
```



順々に全部表示

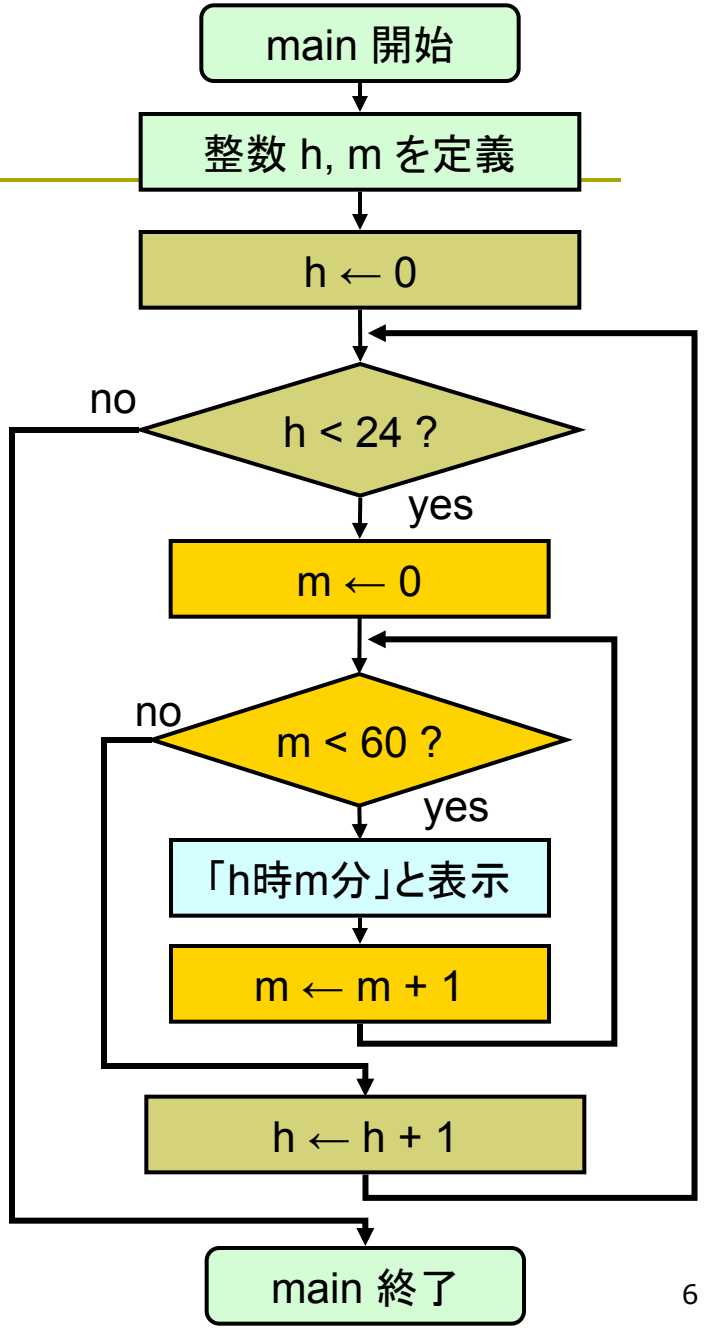
	h=0	h=1	h=2	…	h=22	h=23
m=0	00時00分	01時00分	02時00分	…	22時00分	23時00分
m=1	00時01分	01時01分	02時01分	…	22時01分	23時01分
m=2	00時02分	01時02分	02時02分	…	22時02分	23時02分
…	…	…	…	…	…	…
m=59	00時59分	01時59分	02時02分	…	22時59分	23時59分

2重ループの例

```
#include <stdio.h>

int main(void)
{
  int h;   /* 時 0～23 */
  int m;   /* 分 0～59 */

  for (h = 0; h < 24; h++) {
    for (m = 0; m < 60; m++) {
      printf("%d時%d分\n", h, m);
    }
    getchar(); /* [Enter]キーを待つ */
  }
  return 0;
}
```



フローチャートを
書いてみよう

九九の表示

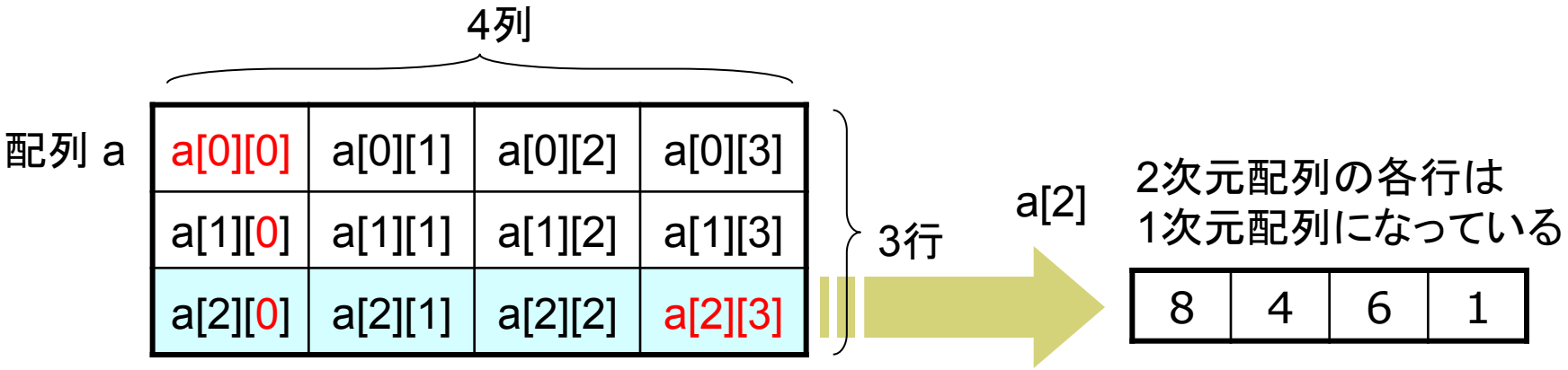
```
#include <stdio.h>

int main(void)
{
  int i, j;

  i = 1;
  while (i <= 9) {
    printf("%dの段\n", i);
    j = 1;
    while (j <= 9) {
      printf("%d×%d = %d\n",
             i, j, i * j);
      j++;
    }
    printf("\n");
    i++;
  }
  return 0;
}
```

多次元配列

- 多次元配列の定義（p.121）
  - 配列名[要素数1][要素数2]...;
    int a[3][4];　　← a[0][0] ～ a[2][3] の2次元配列
    int b[4][4][4];　← b[0][0][0] ～ b[3][3][3] の3次元配列
  - 初期化方法
    int a[3][4] = { { 3, 6, 2, 4 }, { 2, 9, 1, 8 }, { 8, 4, 6, 1 } };



- 多次元配列は「配列の配列」
  - 2次元配列を構成する各行は1次元配列になっている

フローチャートを
書いてみよう

2次元配列の例

```
// 2次元配列を表のように表示
#include <stdio.h>

int main(void)
{
  int a[3][3] = {{ 1, 2, 3 },
    { 4, 5, 6 }, { 7, 8, 9 }};
  int i, j;

  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %2d", a[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

文字列の配列

```
#include <stdio.h>

int main(void)
{
  char names[5][10] = {
    "Oda", "Mori", "Takeda",
    "Uesugi", "Shimazu" };
  int i;

  for (i = 0; i < 5; i++) {
    printf("初期値: %s\n", names[i]);
    printf("変更 -> ");
    scanf("%s", names[i]);
  }
  for (i = 0; i < 5; i++) {
    printf("変更後: %s\n", names[i]);
  }
  return 0;
}
```

最大9文字＋ヌル文字

names[0]	O	d	a	\0						
names[1]	M	o	r	i	\0					
names[2]	T	a	k	e	d	a	\0			
names[3]	U	e	s	u	g	i	\0			
names[4]	S	h	i	m	a	z	u	\0		

文字列は配列なので
その配列は文字型の
2次元配列になる

names[i]というのが
1次元配列の名前
（文字列は&不要）

演習問題

5a.「2次元配列の例」のフローチャートを書きなさい。

5b. かけ算の九九を“表”の形式（数が縦横に並んだ形式）で表示するプログラムを作成しなさい（配列は使わないこと）。

5c. 配列 double A[2][2], B[2][2] を2行2列の行列とみなしてキーボードから値を読み込み、行列の和 C=A+B を計算して double C[2][2] に代入してから、C の各要素を画面に表示するプログラムを作成しなさい。

5d. 下記の表の数値部分を表す2次元配列 int point[4][5] を初期化し、各チームの勝ち点を合計の欄（point[?][4]）に集計して表全体を表示するプログラムを作成しなさい。

	対NED	対JPN	対DEN	対CMR	合計
オランダ(NED)	(0)	3	3	3	
日本(JPN)	0	(0)	3	3	
デンマーク(DEN)	0	0	(0)	3	
カメルーン(CMR)	0	0	0	(0)	

- 次回までの課題：教科書p.127～145 を予習（入力＆実行）

デバッガの活用

- デバッガ（debugger）
  - プログラムのバグ（bug 誤り）を見つけるためのツール
  - プログラム実行中の変数などの状態を見ることができる

- デバッガの代表的機能
  - **ブレークポイント**:　プログラム実行中に指定場所で一時停止させる
  - **ステップ実行**:　　プログラムを1行1行止めながら実行させる

- VC++でのブレークポイントの使い方
  1. 一時停止させたい行にカーソル（カレット）をもっていく
  2. メニューから [デバッグ]→[ブレークポイントの設定/解除] で設定する
  3. 同様に、mainの「return 0;」の行にもブレークポイントを置くとよい
  4. [デバッグ]→[デバッグ開始] でプログラムを実行開始する