第13回
プログラミング及び実習 II

第12回課題の解答例

ex12-1：変数のスコープ

```
#include <stdio.h>

void fa(void);
int a = -1;


int main(void) {
  printf("グローバル変数のa=%d\n", a); / ここにスコープ情報を追加
  int a = 1;
  {
    int a = 10;
    printf("main()関数内の内側スコープのa=%d\n", a); // ここにスコープ情報を追加
    fa();
    a++;
  }
  a++;
  printf("main()関数内の外側スコープのa=%d\n", a); // ここにスコープ情報を追加
  return 0;
}

void fa(void) {
  {
    int a = 100;
    {
      int a = 200;
      printf("fa()関数内の内側スコープのa=%d\n", a); // ここにスコープ情報を追加
      a++;
    }
    printf("fa()関数内の外側スコープのa=%d\n", a); // ここにスコープ情報を追加
  }
  a++;
  printf("グローバル変数のa=%d\n", a); / ここにスコープ情報を追加
}
```


ex12-2：テント写像

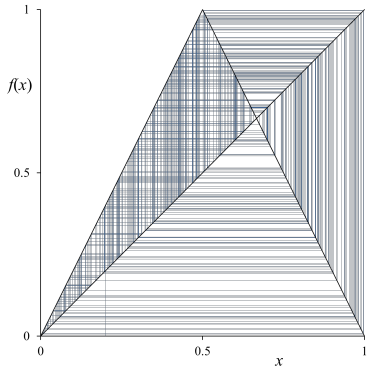
```
#include <stdio.h>
#define N 20

double tent(double);

int main(void) {
  double x;
  do {
    scanf("%lf", &x);
  } while (x < 0 || 1 < x);

  int i=0;
  while (i<N) {
    printf("%.20f\n", x);
    i++;
    x = tent(x);
  }
  return 0;
}

double tent(double x) {
  if (x < 1/2.)
    return 2*x;
  else
    return 2*(1-x);
}
```



ex12-3：配列変数を並び替える関数

```c
#include <stdio.h>

void ssort(int[], int);

int main() {
    int n;
    scanf("%d", &n);
    if (n < 1) return 1;

    int a[n];
    for (int i=0; i<n; i++)
        scanf("%d", &a[i]);

    ssort(a, n);

    for (int i=0; i<n; i++)
        printf("%d\n", a[i]);

    return 0;
}
```

```c
void ssort(int a[], int n) {
    for (int i=0; i<n-1; i++) {
        for (int j=i+1; j<n; j++) {
            if (a[i] > a[j]) {
                int tmp = a[i];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
}
```


ex12-4：配列変数の値を検索する関数

```c
#include <stdio.h>

int vsearch(const int[], int, int);

int main(void) {
  int n;
  scanf("%d", &n);
  if (n < 1) return 1;

  int a[n], v;
  for (int i=0; i<n; i++)
    scanf("%d", &a[i]);

  scanf("%d", &v);
  printf("%d\n", vsearch(a, n, v));
  return 0;
}

int vsearch(const int a[], int n, int s) {
  int cnt = 0;
  for (int i=0; i<n; i++)
    if (a[i] == s) cnt++;
  return cnt;
}
```


ex12-5：配列変数のヒストグラムを求める関数

```c
#include <stdio.h>
#define N 11

void histgram(const int[], int, int[]);

int main() {
    int n;
    scanf("%d", &n);
    if (n < 1) return 1;

    int a[n];
    for (int i=0; i<n; i++) {
        do {
            scanf("%d", &a[i]);
        } while (a[i] < 0 || a[i] > 100);
    }
    int h[N];
    for (int i=0; i<N; i++) h[i] = 0;
    histgram(a, n, h);

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

void histgram(const int a[], int n, int h[]) {
    int val;
    for (int i=0; i<n; i++) {
        h[a[i]/10]++;
    }
}
```


ex12-6：配列変数のヒストグラムを求める関数2

```c
#include <stdio.h>

void histgram2(const int[], int, int[], int);

int main() {
    int n, m;
    scanf("%d", &n);
    if (n < 1) return 1;

    int a[n];
    for (int i=0; i<n; i++) {
        do {
            scanf("%d", &a[i]);
        } while (a[i] < 0 || a[i] > 100);
    }
    scanf("%d", &m);
    if (n < 1) return 2;

    int bin = 100/m + 1;
    int h[bin];
    for (int i=0; i<bin; i++) h[i] = 0;

    histgram2(a, n, h, m);

    for (int i=0; i<bin; i++)
        printf("%3d-%3d:%3d\n", i*m, (i+1)*m-1 > 100 ? 100 : (i+1)*m-1, h[i]);
    return 0;
}

void histgram2(const int a[], int n, int h[], int m) {
    for (int i=0; i<n; i++) {
        h[a[i]/m]++;
    }
}
```

第13回の内容

配列変数（再）

2次元配列

（多次元配列）

配列変数とは（再）

```
int main() {
```

```
  int a;
```
a という名前の整数型の変数を1つ用意

```
  int b[10];
    :
```
b[0], b[1], … , **b[9]** という名前の整数型の変数を合計10こ用意

a

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]

b という整数型の変数は存在しない。b[0] や b[9] が一つ一つの変数。

b[n] と宣言すると、b[0] から b[n-1] までが使える。b[n] は使えない。

配列変数の [ ] の中の何番目かを表す0以上の整数を「添字」と呼ぶ。

第13回の内容

配列変数（再）

2次元配列

（多次元配列）

2次元配列変数

```
int main() {
```

```
  int b[10];
```
大きさ10の整数型の配列変数 b を宣言

```
  int c[3][10];
```
大きさ3×10 の2次元配列変数 c を宣言
30個の整数型配列 c[0][0], c[0][1],…

a	b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]

c[0][0]	c[0][1]	c[0][2]	c[0][3]	c[0][4]	c[0][5]	c[0][6]	c[0][7]	c[0][8]	c[0][9]
c[1][0]	c[1][1]	c[1][2]	c[1][3]	c[1][4]	c[1][5]	c[1][6]	c[1][7]	c[1][8]	c[1][9]
c[2][0]	c[2][1]	c[2][2]	c[2][3]	c[2][4]	c[2][5]	c[2][6]	c[2][7]	c[2][8]	c[2][9]

2次元配列の初期化

基本は全ての要素への逐一代入

```
for (i=0; i<3; i++) {
    for (j=0; j<10; j++) {
        c[i][j] = i*j;
    }
}
```

配列なので宣言時に定数で初期化することもできる

```
int c[3][10] = {
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},     // c[0]
    {0, 1, 2, 3, 4, 5, 6, 7, 8, 9},     // c[1]
    {0, 2, 4, 6, 8, 10, 12, 14, 16, 18} // c[2]
};
```

定数での初期化から c[i][j] == (c[i])[j] という構造が見える
（大きさが10の c[0], c[1], c[2] という3つの一次元配列が積み重なって c[3][10] になっている）

2次元配列の初期化

もし、要素数より多くの値で初期化すると

```
int c[3][2] = {{0, 1, 2}, {3, 4}, {5, 6}};
```

コンパイル時に警告(Warning)される(エラーではない)

もし、要素数より少ない値で初期化すると

```
int c[3][2] = {{0}, {3}, {5}};
```

以降の要素は全て0で初期化される。

宣言時の要素数を省略すると、初期化値の数でサイズが決まる（省略できるのは先頭のサイズだけ！）

```
int c[][2] = {{0, 1}, {2, 3}, {4, 5}}; // → int c[3][2]

// 2つめを省略すると数値を何個ずつ区切るべきかコンパイラが分からない
```


2次元配列の例：九九表の参照

```
for (i=0; i<10; i++) {
   for (j=0; j<10; j++) {
       a[i][j] = i*j;
   }
}
scanf("%d%d", &i, &j);
printf("%d*%d=%d\n", i, j, a[i][j]);
```

2つのパラメータで指定されるようなデータの参照に便利

平面座標・画像データ・行列など

2次元配列を引数とする関数の宣言と定義

```
#include <stdio.h>

int sum(int[][5], int);

int main(void) {
  int a[][5] = {{1,2,3,4,5}, {6,7,8,9,10}}; // a[2][5]
  sum(a, 2);
  return 0;
}

int sum(int a[][5], int n) {
  int sum = 0;
  // 配列 a[][] の n*5 個の要素を sum に加算
  return sum;
}
```

右側のサイズは省略できない！
（要素定数による初期化と同じ理由）

sum() に配列変数の名前とサイズを渡す

右側のサイズは省略できない！

C99 に準拠していれば可変長配列の形で書けないこともない（宣言の n は省略可能）

```
#include <stdio.h>

int sum(int n, int m, int[n][m]); // プロトタイプ宣言
        :
int sum(int n, int m, int a[n][m]) { // 関数定義
   // a[][] の n*m 個の要素を加算して返す
}
```

第13回の内容

配列変数（再）

2次元配列

（多次元配列）

多次元配列変数：3次元の例

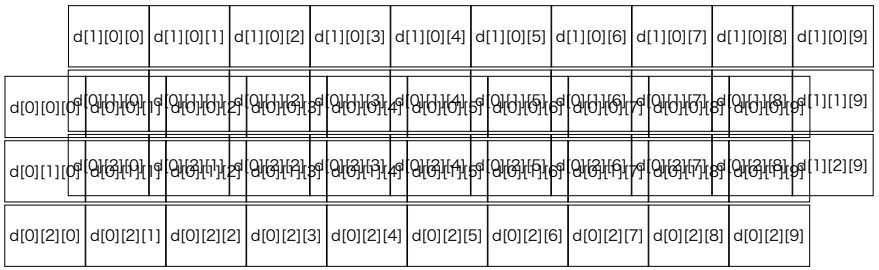
```
int main() {

  int c[3][10];

  int d[2][3][10];
      :
```

大きさ3×10 の2次元配列変数 c
30個の整数型配列 c[0][0], c[0][1],…

大きさ2×3×10 の3次元配列変数 d を宣言
60個の整数型配列 d[0][0][0], c[0][0][1],…



3次元配列を引数とする関数の宣言と定義

```
#include <stdio.h>

int sum(int□[3][4], int);

int main(void) {
  int a□[5][3][4] = {  // a[2][3][4]
                     {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}},
                     {{13,14,15,16}, {17,18,19,20}, {21,22,22,24}
                    };
  sum(a, 2);
  return 0;
}

int sum(int a□[3][4], int n) {
  int sum = 0;
  // 配列 a の n*3*4 個の要素を sum に加算
  return sum;
}
```

最初の要素数しか省略できない！
（要素定数による初期化と同じ理由）

最初の要素数しか省略できない！

C99 に準拠していれば可変長配列の形で書けないこともない（宣言の n は省略可能）

```
#include <stdio.h>

int sum(int n, int m, int l, int[n][m][l]); // プロトタイプ宣言
        :
int sum(int n, int m, int l, int a[n][m][l]) { // 関数定義
   // a の n*m*l 個の要素を加算して返す
}
```