

シラバス

13,14. 1次元配列

配列による同型多数データを扱う処理の実現方法の理解

事前学修：授業スライド「第7回 1次元配列」と教科書p.114～p.118, p.109～p.110の熟読。(60分)

事後学修：授業スライドの用語とその意味の理解。配列の利点, C言語における配列の宣言と初期化, for文と配列を用いた多数データの計算, #define文を説明できること。(60分)

演習課題のプログラムの完成と提出。(180分)

プログラミングの基礎及び演習

日本大学 工学部 情報工学科

プログラミングの基礎 第7回

■ 配列

- 配列とは？
 - 配列の宣言
 - 配列の要素の使い方と初期化
 - 例題：配列の合計と平均計算
 - 問題
 - 配列サイズの変更と#define
- ※教科書 p.114～118, p.109～110

同じ種類のデータの扱い

■ 同じ種類のデータを多数扱うときにはどうするか？

□ 例： 10人の学生のテストの点数

□ 方法 1：

- 変数 $a, b, c, d, e, f, g, h, i, j$ という 10 個の変数に代入

□ 方法 2：

- 変数 $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$ という 10 個の変数に代入

□ 方法 3：

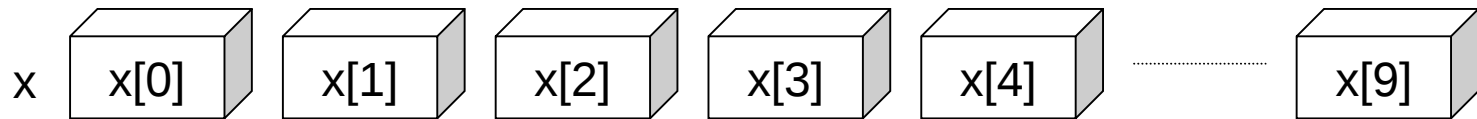
- 変数 x_i ($i=0, 1, 2, \dots, 9$) の形でまとめる扱ことはできないか

– 参考例： 平均値の計算

$$average = \frac{1}{10} \sum_{i=0}^9 x_i$$

配列とは

- 番号付きで並べられた変数
 - それぞれの要素には同じ型の値が格納される
 - 同じ種類のデータをまとめて扱うのに便利
- 配列全体を表すための名前（配列名）を設定



- 個々の要素は番号（添字）を使って参照
 - 配列名[添字] で表わす
 - 個々の要素は変数と同様に扱える
 $x[5] = 80$; など
 - C言語では添字は0から始まる

配列の宣言

- 配列を使う際には、最初に宣言が必要
 - 通常の変数と同様
- 配列の宣言方法

型名 配列名[要素の数];

- 要素の数: データの個数
 - ・ 例: 10人分の点数なら10個
 - 要素の数は定数（数値）で指定する
（変数で指定しない）
- 例
 - `int score[10];` int型で、要素数10個の配列scoreを宣言
 - `double x[100];` double型で、要素数100個の配列xを宣言

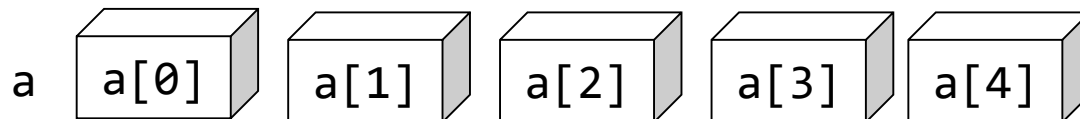
重要： 配列の添字

■ 配列の添字は0から始まる

□ 1 ではないことに注意！

□ 例： `int a[5];` という宣言をした場合：

- `a[0]`, `a[1]`, `a[2]`, `a[3]`, `a[4]` の5つが使える
- `a[1]`, `a[2]`, `a[3]`, `a[4]`, `a[5]` ではない！



「添字」は, 「要素番号」「インデックス」と呼ばれることもある

プログラミングの基礎 第7回

■ 配列

- 配列とは？
 - 配列の宣言
 - 配列の要素の使い方と初期化
 - 例題：配列の合計と平均計算
 - 問題
 - 配列サイズの変更と#define
- ※教科書 p.114～118, p.109～110

配列の初期化 (1)

- 代入文を使って, 1つずつ代入する

```
int score[4];  
score[0] = 3;  
score[1] = 2;  
score[2] = 4;  
score[3] = 5;
```



例題9.1: 配列の値の合計の計算

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i;
```

```
    int score[4];
```

} 要素数 4 の配列scoreの宣言

```
    int total = 0;
```

} 合計を格納する変数totalの宣言と初期化

```
    score[0] = 3;
```

```
    score[1] = 2;
```

```
    score[2] = 4;
```

```
    score[3] = 5;
```

} scoreの各要素への値の代入.
要素数 4 なので, [0], [1], [2], [3]である.
score[4]は使えない.

```
    for( i = 0; i <= 3; i++ )
```

```
        total += score[i];
```

```
    printf("Totalscore: %d\n", total);
```

```
    return 0;
```

```
}
```

合計の計算

$i \leq 4$ ではない

$i < 4$ ならOK

配列の初期化 (2)

■ scanfを使って代入する方法

```
int score[4];  
for( i = 0; i <= 3; i++ ) {  
    printf( "score[%d] = ", i );  
    scanf( "%d", &score[i] );  
}
```

実行結果:

score[0] = 3 Return

score[1] = 2 Return

score[2] = 4 Return

score[3] = 5 Return



例題9.1: 配列の値の合計の計算

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i;
```

```
    int score[4]; } 要素数 4 の配列scoreの宣言
```

```
    int total = 0; } 合計を格納する変数totalの宣言と初期化
```

```
    for( i = 0; i <= 3; i++ ) {
```

```
        printf( "score[%d] = ", i );
```

```
        scanf( "%d", &score[i] );
```

```
    }
```

```
    for( i = 0; i <= 3; i++ )
```

```
        total += score[i];
```

```
    printf("Totalscore: %d\n", total);
```

```
    return 0;
```

```
}
```

scoreの各要素への値の代入。
要素数 4 なので、
[0], [1], [2], [3]である。
score[4]は使えない。

合計の計算
 $i \leq 4$ ではない
 $i < 4$ ならOK

配列の初期化 (3)

■ 配列の宣言と同時にまとめて初期化する

■ 例 :

□ `int score[4] = { 3, 2, 4, 5 };`

- 要素[0]から順に, 3, 2, 4, 5という値が代入される.

□ `int score[4] = { 1 };`

- 先頭の要素に1が代入される.
- 省略した要素は0として扱われる.

□ `int score[] = { 3, 2, 4, 5 };`

- 要素数を省略 ⇒ 値の数が4個なので, 要素数も4
- 注意: 値を初期化しないときは省略不可
 - エラー: `int score[];`



例題9.1: 配列の値の合計の計算

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i;
```

```
    int score[4] = { 3, 2, 4, 5 };
```

```
    int total = 0; } 合計を格納する変数totalの宣言と初期化
```

```
    for( i = 0; i <= 3; i++ )
```

```
        total += score[i];
```

```
    printf("Totalscore: %d¥n", total);
```

```
    return 0;
```

```
}
```

要素数 4 の配列scoreの宣言と
同時に初期化する

．．．初期化子による初期化

合計の計算

$i \leq 4$ ではない

$i < 4$ ならOK

配列の初期化 (4)

■ まとめて初期化する際の注意事項

- まとめて代入できるのは、配列宣言のときだけ
- 通常の代入の代わりには使えない

■ 例：

- 以下の場合にはエラーになる

```
int score[4];  
score[4] = { 3, 2, 4, 5 };
```



プログラミングの基礎 第7回

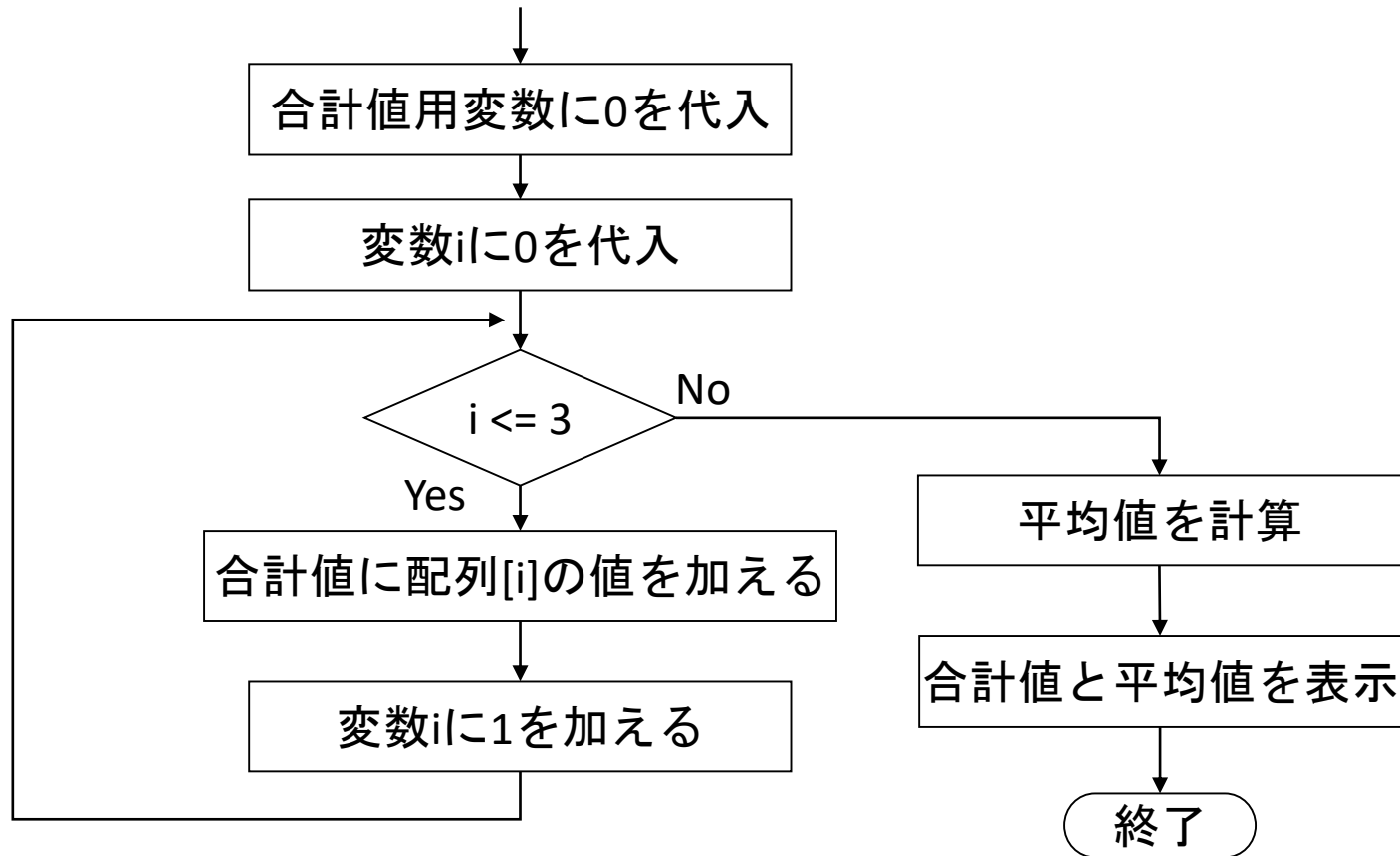
■ 配列

- 配列とは？
 - 配列の宣言
 - 配列の要素の使い方と初期化
 - 例題：配列の合計と平均計算
 - 問題
 - 配列サイズの変更と#define
- ※教科書 p.114～118, p.109～110

例題： 配列の合計の計算 (1)

- 要素数 4 の配列の値の合計を求めるには？
 1. 合計値を入れておくための変数を用意する
 2. 合計値用の変数に0を代入しておく
 3. 変数*i*に0を代入する・・・配列の添字は0から
 4. 変数*i*が3以下である間，以下の処理を繰り返す
 - 合計値用の変数の値に配列[*i*]の値を加える
 - 変数*i*に1を加える
 5. 平均値を計算する
 6. 合計値と平均値を表示する

例題： 配列の合計と平均の計算 (2)





例題: 配列の値の合計と平均の計算

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i;
```

```
    double ave;
```

```
    int score[4];
```

```
    int total = 0;
```

```
    score[0] = 3;
```

```
    score[1] = 2;
```

```
    score[2] = 4;
```

```
    score[3] = 5;
```

```
    for( i = 0; i <= 3; i++ )
```

```
        total += score[i];
```

```
    ave = (double)total/4;
```

```
    printf("Total score: %d¥n", total);
```

```
    printf(("Average score: %f¥n", ave);
```

```
    return 0;
```

```
}
```

要素数 4 の配列scoreの宣言

合計を格納する変数totalの宣言と初期化

scoreの各要素への値の代入.

要素数 4 なので, [0], [1], [2], [3]である.

score[4]は使えない.

合計の計算

$i \leq 4$ ではない

$i < 4$ ならOK

配列の注意事項 (1)

■ 添字が範囲外の場合の動作 (1)

誤

```
for( i = 0; i <= 4; i++ )  
    total += score[i];
```

正

```
for( i = 0; i <= 3; i++ )  
    total += score[i];
```

□ 間違っているが、コンパイルエラーにならない

(C言語のコンパイラは配列の範囲をチェックしない)

上の誤ったプログラムを実行すると、
score[4]が範囲外のため、
正しくない合計値が表示される可能性が高い。



例題9.1: 配列の値の合計の計算

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i;
```

```
    int score[4] = { 3, 2, 4, 5 };
```

```
    int total = 0; } 合計を格納する変数totalの宣言と初期化
```

```
    for( i = 0; i <= 4; i++ )
```

```
        total += score[i];
```

```
    printf("Totalscore: %d¥n", total);
```

```
    return 0;
```

```
}
```

要素数 4 の配列scoreの宣言と
同時に初期化する

．．．初期化子による初期化

合計の計算

$i \leq 4$ ではない

$i < 4$ ならOK

配列の注意事項 (2)

■ 添字が範囲外の場合の動作 (2)

- 配列scoreの要素数が4のとき、
score[5] = 10; と書いてしまった場合でも、
コンパイルエラーにならない

致命的なバグになる可能性があるので注意

配列の添字が範囲内にあるかどうかは、
プログラムを作成者が責任を持って確認せねばならない

問題

- 整数値を 5 個入力し，それを配列に格納する
- 配列の中から，最大値を探し，最大値が格納されている配列の添字を表示せよ.

実行例：

データ[0]の入力： 10

データ[1]の入力： 30

データ[2]の入力： 25

データ[3]の入力： 44

データ[4]の入力： 13

最大値となるのはデータ[3]です.

ヒント

- 配列の最大値を求めるには？
 - 先頭の要素[0]を「仮の最大値」とする.
 - 要素[1]から[4]まで，繰り返す
 - ・ 要素[i]と「仮の最大値」を比較し，
要素[i]の方が大きければ，
「仮の最大値」を変更する
 - 「仮の最大値」が「本当の最大値」である.

 - 添字を求めるには？
-

解答 (1)

「仮の最大値」の添字

```
int main(void) {  
    int i, max, max_index;  
    int array[5];
```

```
    for( i = 0; i < 5; i++ ) {  
        printf( "データ [%d]の入力: ", i );  
        scanf( "%d", &array[i] );
```

```
    }
```

初期値は0（先頭の要素）

```
    max = array[0];
```

```
    max_index = 0;
```

```
    for( i = 1; i < 5; i++ ) {
```

```
        if( max < array[i] ) {
```

```
            max = array[i];
```

```
            max_index = i;
```

```
        }
```

```
    }
```

```
    printf( "最大値となるのはデータ [%d]です¥n", max_index );
```

```
    return 0;
```

```
}
```

「仮の最大値」を変更するときに、同時に添字も変更

解答 (2)

```
int main(void) {  
    int i, max_index;  
    int array[5];  
  
    for( i = 0; i < 5; i++ ) {  
        printf( "データ [%d]の入力: ", i );  
        scanf( "%d", &array[i] );  
    }  
  
    max_index = 0;  
    for( i = 1; i < 5; i++ ) {  
        if( array[max_index] < array[i] )  
            max_index = i;  
    }  
    printf( "最大値となるのはデータ [%d]です¥n", max_index );  
    return 0;  
}
```

「仮の最大値」の添字

「仮の最大値」を省略することも可能



プログラミングの基礎 第7回

■ 配列

- 配列とは？
- 配列の宣言
- 配列の要素の使い方と初期化
- 例題：配列の合計と平均計算
- 問題
- 配列サイズの変更と#define

※教科書 p.114～118, p.109～110

配列のサイズを変更するには？

- 「データ数: 5個」 ⇒ 「データ数: 10個」
- どの部分を変更すれば良いか？

```
int main(void) {  
    int i, max_index;  
    int array[5];  
  
    for( i = 0; i < 5; i++ ) {  
        printf( "データ[%d]の入力: ", i );  
        scanf( "%d", &array[i] );  
    }  
  
    max_index = 0;  
    for( i = 1; i < 5; i++ ) {  
        if( array[max_index] < array[i] )  
            max_index = i;  
    }  
    printf( "最大値となるのはデータ [%d] です¥n", max_index );  
    return 0;  
}
```

配列のサイズを変更する

```
int main(void) {  
    int i, max_index;  
    int array[5];  
  
    for( i = 0; i < 5; i++ ) {  
        printf( "データ[%d]の入力: ", i );  
        scanf( "%d", &array[i] );  
    }  
  
    max_index = 0;  
    for( i = 1; i < 5; i++ ) {  
        if( array[max_index] < array[i] )  
            max_index = i;  
    }  
    printf( "最大値となるのはデータ[%d]です¥n", max_index );  
    return 0;  
}
```

3ヶ所変更する必要あり

- すべて10に変更する必要がある
- 変更を忘れると正しく動作しない

「確実にすべて変更」するにはどうすれば良いか？

#define

- プログラム中で用いる定数に名前を付ける
 - 定数を変更するときに、一括変更が可能になる
- プログラムの先頭部分に、以下のように記述する
 - プログラム中に「名前」を書けば、その「数値」に自動で置き換えられる

`#define 名前 数値`

- 例:
 - #define SIZE 5
プログラム中にSIZEと書くと、
「5」として扱われる



#defineを使う例 (1)

```
#include <stdio.h>
```

```
#define SIZE 5
```

#include の後に記述 (セミコロン不要)

```
int main(void) {  
    int i, max_index;  
    int array[SIZE];
```

```
    for( i = 0; i < SIZE; i++ ) {  
        printf( "データ[%d]の入力: " , i);  
        scanf( "%d", &array[i] );  
    }
```

```
    max_index = 0;  
    for( i = 1; i < SIZE; i++ ) {  
        if( array[max_index] < array[i] )  
            max_index = i;  
    }
```

```
    printf( "最大値となるのはデータ[%d]です\n", max_index );  
    return 0;
```

```
}
```

#defineを使う例 (2)

```
#include <stdio.h>
```

```
#define SIZE 10
```

ここを変えるだけで、
配列のサイズを変更できる

```
int main(void) {  
    int i, max_index;  
    int array[SIZE],
```

```
    for( i = 0; i < SIZE; i++ ) {  
        printf( "データ[%d]の入力: " , i);  
        scanf( "%d", &array[i] );  
    }
```

```
    max_index = 0;  
    for( i = 1; i < SIZE; i++ ) {  
        if( array[max_index] < array[i] )  
            max_index = i;  
    }
```

```
    printf( "最大値となるのはデータ[%d]です\n", max_index );  
    return 0;
```

```
}
```

変更不要

1ヶ所変更するのみ
間違いが少ない

#define: まとめ

- 配列のサイズなどの定数は,
#defineで定義しておく
 - 「複数の場所で使う」 数値は#defineするべき
 - 「5」や「10」のような数値を
直接記述してはいけない
- #defineで定義する名前は
「全部大文字」にするのが一般的である
 - 例: SIZE, NUMBER, MAX_LENGTH など
 - 変数名は「小文字」にする
⇒ 区別しやすくなる

【演習課題】

- 演習課題提出システムの「第7回」演習課題を実施
- 演習課題7-1, 7-2, 7-3, 7-4
 - 演習時間に，設計後に演習環境にリモートログインしプログラムを作成
 - 演習環境でコンパイル，テストし，完成
 - ソースファイル（7-x.c）をWinSCP等でPCへ転送
 - 課題提出システムへSubmitし各課題100点を目指す
 - 時間内に終了しない場合 ⇒ 宿題
- 授業翌週水曜日の午後5時までに必ず提出
- レポート，今回はなし

演習課題7-2, 7-3 ヒント

- データ入力 ("Input Data") および
データ一覧 ("Data List") の
データ番号(No. に続く1から10の数字)の
表示は、2桁として揃えて表示する
変換指定 %2d

演習課題7-3 ヒント

- 平均値は, 小数点以下2桁まで表示
`printf("average value : %.2f\n", ave);`
- 最小値と最大値については,
データの番号も表示する