

第8回 繰り返しⅢ （繰り返し応用）



1

今回の目標

- 配列とfor文の組み合わせ方を理解する。

☆与えられたデータの平均値を
求めるプログラムを作る。

2

for文と配列

for文と配列は大変相性が良い。
例えば、配列dataの中身を出力する場合に、
ループカウンタ(int型の変数)を配列の添え字として用いれば、
プログラムが非常にシンプルになる。

```
printf("%d¥n",data[0]);  
printf("%d¥n",data[1]);  
printf("%d¥n",data[2]);  
printf("%d¥n",data[3]);  
...  
printf("%d¥n",data[9]);
```

```
for(i=0; i<10; i++)  
{  
    printf("%d¥n",data[i]);  
}
```

配列の添え字には、int型の定数だけでなく、
int型の式¹が使える。
ループカウンタとの組合せは典型的な使い方。

3

プログラム例1: for文による配列要素表示の練習

```
/*print_array.c 配列要素表示の練習 コメント省略 */  
#include<stdio.h>  
/* マクロ定義 */  
#define SIZE 10 /* 配列の要素数 */  
  
int main()  
{  
    int i; /* ループカウンタ*/  
    int data[SIZE]; /* データを格納する配列 */  
  
    /* データの初期化 */  
    data[0]=0;  
    data[1]=1;  
    data[2]=2;  
    /* 次に行く */
```

4

```
data[3]=3;
data[4]=4;
data[5]=5;
data[6]=6;
data[7]=7;
data[8]=8;
data[9]=9;

/* 配列の中身の表示 */
for(i=0; i<SIZE; i++)
{
    printf("data[%2d]=%2d¥n",i,data[i]);
}

return 0;
}
```

5

配列へのデータの入力

配列にデータを入力する場合にも、forループを用いると便利である。

例えば、int型配列dataにデータを格納する場合に、ループカウンタを配列の添え字として用いて、繰り返しscanf関数を呼び出せば、プログラムが単純になる。

```
scanf("%d", &data[0]);
scanf("%d", &data[1]);
scanf("%d", &data[2]);
scanf("%d", &data[3]);
...
scanf("%d", &data[9]);
```



```
for(i=0; i<10; i++)
{
    scanf("%d", &data[i]);
}
```

6

プログラム例2: for文による配列要素入出力の練習

```
/* 配列へのデータ格納実験 scan_array.c      コメント省略
*/
#include<stdio.h>

/* マクロ定義 */
#define SIZE 3 /* 配列の要素数 */

int main()
{
    int i; /* ループカウンタ*/
    int data[SIZE]; /* データを格納する配列 */

    /* 次に行く */
}
```

7

```
/* データの入力 */
for(i=0; i<SIZE; i++)
{
    scanf("%d", &data[i]);
}

/* 配列の中身の表示 */
for(i=0; i<SIZE; i++)
{
    printf("data[%2d] = %2d¥n", i, data[i]);
}

return 0;
}
```

8

プログラム例3の原理: ベクトルの足し算

2つの n 次元ベクトル

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

の足し算を行うプログラムを作る。

$$\mathbf{a} + \mathbf{b} = \begin{pmatrix} a_0 + b_0 \\ a_1 + b_1 \\ \vdots \\ a_{n-1} + b_{n-1} \end{pmatrix}$$

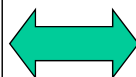
9

ベクトルの足し算

ベクトル $\mathbf{a}$, $\mathbf{b}$ はプログラム中では配列に保存される。
演算結果はベクトル $\mathbf{c}$ に保存する(プログラム中ではこれも配列)。

ベクトルの足し算は、成分ごとの足し算を繰り返すことで計算される。

$$\begin{aligned} c_0 &= a_0 + b_0 \\ c_1 &= a_1 + b_1 \\ &\vdots \\ c_{n-1} &= a_{n-1} + b_{n-1} \end{aligned}$$



$$\begin{aligned} c_i &= a_i + b_i \\ (i &= 0, 1, \dots, n-1) \end{aligned}$$

10

プログラム例3: 配列要素を用いた反復計算の練習

```
/*ベクトルの足し算 add_vector.c      コメント省略 */
#include<stdio.h>

/* マクロ定義 */
#define SIZE 3 /* 配列の要素数 */

int main()
{
    int i; /* ループカウンタ*/
    double vector_a[SIZE]; /* 入力ベクトルa */
    double vector_b[SIZE]; /* 入力ベクトルb */
    double vector_c[SIZE]; /* ベクトルの和c=a+b */

    /* 次に行く */
}
```

11

```
/* ベクトルの成分の入力 */
printf("ベクトルaを入力して下さい\n");
for(i=0; i<SIZE; i++)
{
    scanf("%lf", &vector_a[i]);
}

printf("ベクトルbを入力して下さい\n");
for(i=0; i<SIZE; i++)
{
    scanf("%lf", &vector_b[i]);
}

/* 次に行く */
```

12

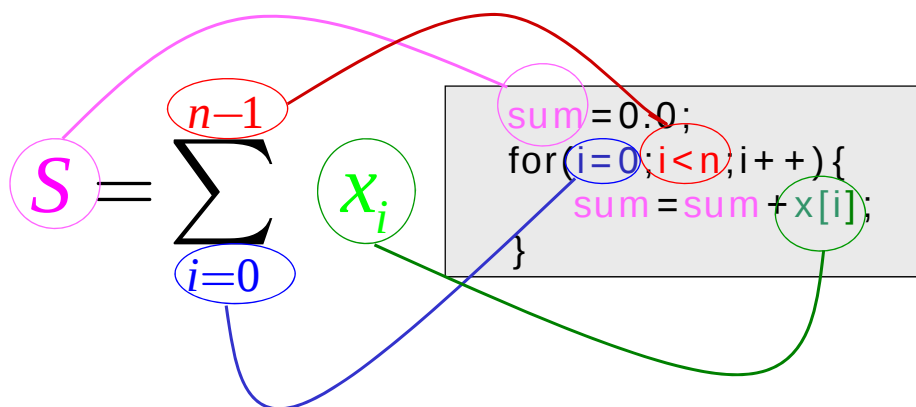
```
/* ベクトルの足し算 */
for(i=0; i<SIZE; i++)
{
    vector_c[i]=vector_a[i]+vector_b[i];
}

/* 結果の表示 */
for(i=0; i<SIZE; i++)
{
    printf("c[%d]=%6.2f\n",i,vector_c[i]);
}

return 0;
}
```

13

数学記号と繰り返し記号の対応(総和計算)



数学にも計算の繰り返しを表す記号がいくつかある。
計算の繰り返しは、for文を用いて記述できる。

14

プログラム例4の原理: 平均

n 個のデータ $x_0, x_1, \dots, x_{n-1}$ の平均値は次式で計算できる。

$$\begin{aligned}\bar{x} &= \frac{x_0 + x_1 + \dots + x_{n-1}}{n} \\ &= \frac{\sum_{i=0}^{n-1} x_i}{n}\end{aligned}$$

15

プログラム例4: 平均値を求めるプログラム

```
/*
    作成日: yyyy/mm/dd
    作成者: 本荘太郎
    学籍番号: B00B0xx
    ソースファイル: average.c
    実行ファイル: average
    説明: n個の実数に対し、その平均値を求めるプログラム。
    入力: まずデータ数として、標準入力から
           データの個数を表す1つの正の整数nを入力。
           続いて、データとして、標準入力からn個の実数を
           任意の順番で入力。
    出力: 標準出力に入力されたn個のデータの平均値を出力。
           平均値は実数である。

*/
/*  次のページに続く  */
```

16

```
#include <stdio.h>
/*マクロ定義*/
#define DATANUM 1000 /* データ個数の上限 */
int main()
{
    /* 変数宣言 */
    int n; /* データ数 */
    int i; /* ループカウンタ、配列dataの添字 */
    double ave; /* データの平均値 */
    double sum; /* データの総和 */
    double data[DATANUM]; /* データを入れる配列 */

    /* データ数の入力 */
    printf("データ数を入力して下さい。¥n");
    printf("n = ? ¥n");
    scanf("%d", &n);
    /* 次のページに続く */
}
```

17

```
/* データ数nのチェック */
if(n <= 0)
{
    /* 不正な入力: データ数が0以下 */
    printf("データが無いのですね。¥n");
    return -1;
}
/* これ以降では、nは正の整数 */
if(n > DATANUM)
{
    /* 不正な入力: 上限オーバー */
    printf("データが多すぎます。¥n");
    return -1;
}
/* これ以降では、nは1からDATANUMまでの整数 */
/* 次のページに続く */
```

18

```
/* 標準入力から配列へデータを読み込む/  
for(i=0; i<n; i++)  
{  
    /* n個の実数データの入力 */  
    printf("data[%d] = ?", i);  
    scanf("%lf", &data[i]);  
}  
  
/* 次のページに続く */
```

19

```
/* 総和の計算 */  
/* 初期設定 */  
sum=0.0;  
for(i=0; i<n; i++)  
{  
    sum = sum+data[i];  
}  
  
/* 平均の計算 */  
ave = sum/(double)n;  
  
/* 平均値の出力 */  
printf("平均値は %.2fです。¥n", ave);  
return 0;  
}
```

20

プログラム例4の実行結果

```
$make  
gcc average.c -o average  
$ ./average  
データ数を入力して下さい。  
n = ?  
3  
data[0] = ? 4.0  
data[1] = ? 2.0  
data[2] = ? -3.0  
平均値は 1.00 です。  
$
```