電子 1(木曜クラス)

第 11 回課題 T11

(ポインタ:教科書第 14 章、2007/6/28(木))

基本問題

T11-1:配列の引数

(本提出期限 2007/6/28(木)17:40、再提出期限 2007/7/12(木)14:30)

提出物:Makefile、ソースファイル (arg_array.c)、入力ファイル (arg_array.in)、出力ファイル (arg_array.out)

他の関数内の 1 次元配列に標準入力から値を読み込む関数および、他の関数内の 1 次元配列の値を標準出力に出力する関数を作成せよ。これらの関数を利用して、標準入力からデータを読み込み、標準出力へデータを出力するプログラムを作成せよ。

ただし，次の実行例を参考にして，後ろに示す要求仕様を満たすように作成すること.

arg_array.in の例

```
5

10.4
2.3
-4.3
7.1
6.0
```

実行例

```
b08b0xx@tyy:~/T11/1$./arg_array < arg\_array.in
読み込む要素数は?
n=?

配列に 5 個の double 値を読み込みます。

配列の 5 個の double 値を表示します。
10.4
2.3
-4.3
7.1
6.0

b08b0xx@tyy:~/T11/1$
```

要求仕様 (T11-1)

全体的な仕様:

- 他の関数中の 1 次元配列に，標準入力から double 値を読み込む関数 scan_array を作れ。scan_array の仕様は後の説明を参照.
- 他の関数中の 1 次元配列の各要素を標準出力に小数点以下 2 桁まで出力する関数 print_array を作れ。print_array の仕様は後の説明を参照.
- データを読み込む配列 array は main 関数のローカル変数とすること。
- 配列 array には最高 100 個までのデータを保存できるようにすること．この値はプログラムの先頭でマクロ定義し、マクロ定義の部分だけを書き換えるだけで変更できるようにすること。すなわち，次のようなマクロ定義がしてあること.

```
/*マクロ定義*/
#define N_MAX (100) /*データ数の上限，一次元配列の要素数*/
```

- グローバル変数を用いないこと。
- 読み込むデータ数を蓄える変数 n を main 関数のローカル変数として用意する．main 関数内で標準入力から n にデータ数を読み込むこと．main 関数内ではデータ数だけを読み込こみ，データの値を標準入力から読み込んではいけない.
- 不正な配列要素を参照しないように，データ数 n のチェックを main 関数内で行なうこと.
- main 関数中から，関数 scan_array を 1 度呼び出すこと．呼び出す際には適切に実引数を設定すること.
- main 関数中から，関数 print_array を 1 度呼び出すこと．呼び出す際には適切に実引数を設定すること.

他の関数中の一次元配列に指定個数の値を標準入力から読み込む関数 scan_array の仕様:

- 次のプロトタイプ宣言を持つ.

```
/*配列に標準入力から指定個数分値を読み込む関数*/
void scan_array(int n, double array[N_MAX]);
```

- 仮引数は次の意味を持つ.
  - 仮引数 n はデータ数とし，1 以上の自然数が与えられるとする．呼び出された段階で常に $1 \leq$ n $\leq$ N_MAX が満たされるとして良く，数関 scan_array 中で n のチェックを行なう必要は必要はない.
  - 仮引数 array は標準入力から読み込む n 個の値を格納するための 1 次元配列 (の先頭要素のアドレス) とし，配列の各要素は double 型の値とする.
  - マクロ N_MAX は，仮引数で与えられる配列 array の要素数として，適切に定義されているとする.
- 戻り値は無い．すなわち，戻り値の型は void 型とする.
- 仮引数で与えられる配列 array の array[0] から array[n-1] までの n 要素に，適切に標準入力から double 値を読み込む副作用を持つ.
- 関数 scan_array 中では標準出力による出力は行なってはならない．例えば，printf 文による表示を行ってはならない.

他の関数中の一次元配列の指定個数の値を標準出力に出力する関数 print_array の仕様:

- 次のプロトタイプ宣言を持つ.

```
/*配列の指定個数分の要素を標準出力に出力する関数*/
void print_array(int n, double array[N_MAX]);
```

- 仮引数は次の意味を持つ.
  - 仮引数 n はデータ数とし，1 以上の自然数が与えられるとする．呼び出された段階で常に $1 \leq$ n $\leq$ N_MAX が満たされるとして良く，関数 print_array 中で n のチェックを行なう必要は必要はない.
  - 仮引数 array は，n 個の要素が格納された配列 (の先頭要素のアドレス) とし，配列の各要素は double 型の値とする．呼び出された段階で array[0] から array[n-1] までに値が格納されているとする.
  - マクロ N_MAX は，仮引数で与えられる配列 array の要素数として，適切に定義されているとする.
- 戻り値は無い．すなわち，戻り値の型は void 型とする.
- 仮引数で与えられる配列 array の array[0] から array[n-1] までの, n 要素を小数点以下 2 桁まで，標準出力に出力する副作用を持つ.
- 関数 print_array 中では標準入力による入力は行なってはならない．例えば，scanf 文による表示を行ってはならない.

応用問題

T11-2:ベクトルの足し算
(本提出期限 2007/7/5(木)14:30、再提出期限 2007/7/12(木)14:30)

提出物：Makefile、ソースファイル (add_vector.c)、入力ファイル (add_vector.in)、出力ファイル (add_vector.out)

配列に格納されたベクトルの和を計算する関数を作成し、その関数を用いてベクトルの和を計算せよ。

ただし，次の実行例を参考にして，後ろに示す要求仕様を満たすように作成すること．

add_array.in の例

```
3

10.4
2.3
-4.3

2.2
-1.5
-3.0
```

実行例

```
b08b0xx@tyy:~/T11/2$./add_vector < add_vector.in
ベクトルの次数は？
n=?

3 次ベクトルに double 値を設定します。

3 次ベクトルに double 値を設定します。

ベクトルの和を計算中。

3 次ベクトルを表示します。
  12.6
   0.8
  -7.3

b08b0xx@tyy:~/T11/2$
```

要求仕様 (T11-2)

全体的な仕様:

- プログラムを通じ，全ての n 次元ベクトルは 1 次元配列に格納する．ただし，異なるベクトルは異なる 1 次元配列に格納する．
- 2 本のベクトル同士の和を計算する関数 add を作れ。add の仕様は後の説明を参照.
- 他の関数中の 1 次元配列に，標準入力から double 値を読み込む関数 scan_array を作れ。scan_array の仕様は，基本問題の scan_array と同様とする．
- 他の関数中の 1 次元配列の各要素を標準出力に小数点以下 2 桁まで出力する関数 print_array を作れ。print_array の仕様は基本問題の print_array と同様とする．
- ベクトルを格納する 1 次元配列は，全て main 関数のローカル変数とすること。
- ベクトルは最高 100 次まで扱えるようにすること．この値はプログラムの先頭でマクロ定義し、マクロ定義の部分だけを書き換えるだけで変更できるようにすること。すなわち，次のようなマクロ定義がしてあること．

```
/*マクロ定義*/
#define DIM_MAX (100) /*ベクトルの最高次数，一次元配列の要素数*/
```

- グローバル変数を用いないこと。
- ベクトルの次数を格納する変数 n を main 関数のローカル変数として用意する．main 関数内で標準入力から n に次数を読み込むこと．
- 不正な配列要素を参照しないように，次数 n のチェックを main 関数内で行なうこと．
- 関数 add を 1 度呼び出すことで，2 本のベクトルの和を計算せよ．呼び出す際には適切に実引数を設定すること．
- 被演算ベクトルに値を設定する際には，関数 scan_array を呼び出すことで行なうこと．呼び出す際には適切に実引数を設定すること．
- 演算結果ベクトルの内容を表示するには，関数 print_array を呼び出すことで行なうこと．呼び出す際には適切に実引数を設定すること．

2 つのベクトルの和を計算する関数 add の仕様:

- 次のプロトタイプ宣言を持つ.

```
/* ベクトルの和演算を行う関数 */
int add(int n,
  double operand1[DIM_MAX], double operand2[DIM_MAX],
  double result[DIM_MAX]);
```

- 仮引数は次の意味を持つ.
  - 仮引数 n はベクトルの次数とし，1 以上の自然数が与えられるとする．呼び出された段階で常に $1 \leq$ n $\leq$ DIM_MAX が満たされるとして良く，関数 add 中で n のチェックを行なう必要は必要はない.
  - 仮引数 operand1 は被演算ベクトル (和演算の左辺) を格納している 1 次元配列の先頭要素のアドレスを受け取る．被演算ベクトル (和演算の左辺) は double 型の値を要素として持つ n 次元ベクトルとする.
  - 仮引数 operand2 は被演算ベクトル (和演算の右辺) を格納している 1 次元配列の先頭要素のアドレスを受け取る．被演算ベクトル (和演算の右辺) は double 型の値を要素として持つ n 次元ベクトルとする.
  - 仮引数 result は，和演算結果で得られるベクトルを格納するための 1 次元配列の先頭要素のアドレスを受け取る．和演算結果ベクトルは double 型の値を要素として持つ n 次元ベクトルとする.
  - マクロ DIM_MAX は，仮引数で与えられる各配列の要素数として，適切に定義されているとする.
- 戻り値は無い．すなわち，戻り値の型は void 型とする.
- 仮引数 operand1 で与えられたベクトルと，仮引数 operand2 で与えられたベクトルの和を仮引数 result の 1 次元配列に格納する副作用を持つ.
- 関数 add 中では，標準入出力による入出力は行なわないこと．例えば，scanf 文による標準入力からの読み込みや，printf 文による標準出力への出力は行なわないこと.