電子 1(木曜クラス)

第 12 回課題 T12

(新しい型と構造体:教科書第 16 章、2005/07/21(Thu.))

基本問題

T12-1:複素数 (本提出期限 2005/07/21(Thu.)17:40、再提出期限 2005/07/28(Thu.))

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

複素数の四則演算 (加算、減算、乗算、除算) を行なうプログラムを作成せよ。入力は 2 つの複素数を実部、虚部の順に標準入力から行なう。出力は 4 つの演算結果すべてを標準出力に行なう。ただし、プログラムは以下の仕様を満たすように作成すること。

(1) 複素数は配布資料の Complex 型の構造体を用いて表現すること。

(2) 複素数を標準入力より入力させて、それを戻り値とする関数を作成せよ。また、複素数を標準出力へ出力する関数を作成せよ。ただし、これら関数は後で示すプロトタイプ宣言を持つようにすること。

(3) 複素数の四則演算を行なう関数をそれぞれ作成せよ。ただし、これらの関数は、後で示すプロトタイプ宣言を持つようにすること。

```
================================================================
/*プロトタイプ宣言*/
/*
標準入力から 2 つの double 値を順に読みとり、
最初の double 値を実部、2 番目の double 値を虚部として持つ
複素数を返す関数。

引数::
なし (void)

戻り値::
標準入力から受け取った複素数 (Complex 型)
*/
Complex scan_complex(void);



/*
引数で与えられた複素数 z を (実部, 虚部) の形式で
標準出力に出力する関数。

引数::
z:表示すべき複素数
```

```
戻り値::
なし (void)
*/
void print_complex(Complex z);


/*
2 つの複素数を加算する関数

引数::
operand1:(被演算項である複素数 1)
operand2:(被演算項である複素数 2)

戻り値::
加算結果の複素数 (operand1+operand2)
*/
Complex add_complex(Complex operand1,Complex operand2);


/*
2 つの複素数を減算する関数

引数::
operand1:(被演算項である複素数 1)
operand2:(被演算項である複素数 2)

戻り値::
減算結果の複素数 (operand1-operand2)
*/
Complex sub_complex(Complex operand1,Complex operand2);


/*
2 つの複素数を乗算する関数

引数::
operand1:(被演算項である複素数 1)
operand2:(被演算項である複素数 2)

戻り値::
乗算結果の複素数 (operand1*operand2)
*/
```

```
Complex mul_complex(Complex operand1,Complex operand2);


/*
2 つの複素数を除算する関数

引数::
operand1:(被演算項である複素数 1)
operand2:(被演算項である複素数 2)

戻り値::
除算結果の複素数 (operand1/operand2)
*/
Complex div_complex(Complex operand1,Complex operand2);
=============================================================
```

実行例 1

```
~/T12/1$ ./complex
複素数 a=? (空白区切り)
1 2
複素数 b=? (空白区切り)
3 4
(  1.00,  2.00)+(  3.00,  4.00)=(  4.00,  6.00)
(  1.00,  2.00)-(  3.00,  4.00)=( -2.00, -2.00)
(  1.00,  2.00)*(  3.00,  4.00)=( -5.00, 10.00)
(  1.00,  2.00)/(  3.00,  4.00)=(  0.44,  0.08)
~/T12/1$ ./vector
```

応用問題

T12-2:ベクトル
(本提出期限 2005/07/28(Thu.)17:40、再提出期限 2005/07/28(Thu.))

提出物 : Makefile、ソースファイル (vector.c)、入力ファイル (vector.in)、出力ファイル (vector.out)
3 次元ベクトルの内積と外積を求めるプログラムを作成せよ。入力は、2 つのベクトルの要素 (実数) を x 成分、y 成分、z 成分の順に、main 関数内で標準入力から行なう。出力は、2 つのベクトルの内積の値と外積の x,y,z 成分を、main 関数内から標準出力に行なう。ただし、プログラムは以下の仕様を満たすように作成すること。

(1) 3 次元ベクトルは構造体を用いて表現すること。ただし、この構造体は後で示す Vector 型を持つようにすること。

(2) 3 次元ベクトルの各成分 (実数) を x 成分、y 成分、z 成分の順に、標準入力より入力させてそれを戻り値として返す関数を作成せよ。また、3 次元ベクトルを標準出力へ出力する関数を作成せよ。ただし、これら関数は後で示すプロトタイプ宣言を持つようにすること。

(3) 2 つの 3 次元ベクトルの内積を求める関数を作成すること。ただし、これらの関数は、後で示すプロトタイプ宣言を持つようにすること。

(4) 2 つの 3 次元ベクトルの外積を求める関数を作成すること。ただし、これらの関数は、後で示すプロトタイプ宣言を持つようにすること。

```
================================================================
/*構造体テンプレート*/
struct vector
{
double x;/* x 成分*/
double y;/* y 成分*/
double z;/* z 成分*/
};

/*型定義*/
typedef struct vector Vector; /*Vector 型の定義*/


/*プロトタイプ宣言*/

/*
3 次元ベクトルを標準入力から読み込む関数。
x 成分、y 成分、z 成分の順に読み込み、
それらの成分を持つ 3 次元ベクトルを戻す。

引数::
なし (void)
```

```
戻り値::
一つの 3 次元ベクトル (Vector 型)
*/
Vector scan_vector(void);


/*
引数で与えられた 3 次元ベクトルを
(x 成分,y 成分,z 成分) の形式で標準出力に出力する関数。

引数::
v:表示すべき 3 次元ベクトル

戻り値::
なし (void)
*/
void print_vector(Vector v);


/*
2 つの 3 次元ベクトルの内積を計算する関数

引数::
operand1:(被演算項であるベクトル 1)
operand2:(被演算項であるベクトル 2)

戻り値::
内積結果のスカラー (operand1・operand2、double 型)
*/
double inner_product(Vector operand1,Vector operand2);


/*
2 つの 3 次元ベクトルの外積を計算する関数
引数::
operand1:(被演算項であるベクトル 1)
operand2:(被演算項であるベクトル 2)


戻り値::
外積結果の (operand1 × operand2、Vector 型)
```

```
*/
Vector outer_product(Vector operand1,Vector operand2);
=====================================================================
```

実行例 1

```
~/T12/2$ ./vector
(Ax,Ay,Az)?
1 2 3
(Bx,By,Bz)?
3 2 1
A・B= 10.00
A × B=( -4.00,  8.00, -4.00)
~/T12/2$
```

実行例 2

```
~/T12/2$ ./vector < vector.in
(Ax,Ay,Az)?
(Bx,By,Bz)?
A・B= 20.00
A × B=( -1.00,  2.00, -1.00)
~/T12/2$
```

なお、入力の例として、/home/student/submit/T12/2/vector.in というファイルを用意したので、コピーして利用して下さい。