演習課題

課題 1

サンプルプログラムをコンパイル・実行して動作を確認せよ．さらに，適宜修正してその実行結果を確認せよ．

課題 2

サンプルプログラム sample1.f90 <chap09_sample1> を参考にして，数学定数を定義するモジュールを作成せよ．メインプログラムも作成して，動作を確認すること．特に use によるモジュール参照， only の使い方などを理解し，定数値の書き換えが出来ない (コンパイルエラーとなる) ことを確かめよ．数学定数としては例えば π, $\sqrt{\pi}$, e などを定義すれば良い．

課題 3

sample5.f90 <chap09_sample5> のモジュール mod_vector で定義されている 2 次元のベクトル構造型 type(vector2) を拡張し，以下の演算子を定義せよ．(以下で a, b は共に type(vector2) で宣言されたものとする．)

▷ - 演算子: 倍精度実数と type(vector2) の間で以下のような演算が出来るようにする．

```
b = a - 1.0_8 ! b%x = a%x - 1.0_8; b%y = a%y - 1.0_8; と同値になるように
b = 1.0_8 - a ! b%x = 1.0_8 - a%x; b%y = 1.0_8 - a%y; と同値になるように
```

▷ * 演算子: ベクトル積を返す演算子とする．(従って以下の 2 行が同じ結果となる．)

```
write(*,*) a * b
write(*,*) a%x * b%y - a%y * b%x
```

▷ = 演算子: 以下のような代入文を実行できるようにする．

```
a = 1.0_8             ! x, y に同じ値 (1.0) を代入
b = (/2.0_8, 1.0_8/) ! x には 2.0, y には 1.0 を代入
```

ここで b = (/2.0_8, 1.0_8/) の右辺は長さ 2 の倍精度実数型の配列である．

課題 4

mod_vector をさらに拡張し， x, y, z を要素に持つ 3 次元ベクトルを表す構造型 type(vector3) を新たに定義せよ．また，この型に対する +, -, *, = などの演算子を定義せよ．(ただし, * 演算子は通常のベクトル積を返すものとし，それ以外は type(vector2) の自然な拡張となるようにすること．)

さらにサブルーチン show をオーバーロードし，type(vector2), type(vector3) のどちらの変数も引数に取れるようにせよ．

これによって，例えば

```
type(vector2) :: a, b, c
type(vector3) :: x, y, z

write(*, fmt='(a)') '--- vector2 ---'

a = (/1.0_8, 0.0_8/)
b = 1.0_8
c = a + b

call show(a + b)
call show(a - b)

write(*, fmt='("a * b = ", f12.4)') a*b

write(*, fmt='(a)') '--- vector3 ---'

x = (/1.0_8, 0.0_8, 0.0_8/)
y = 1.0_8
z = x * y

call show(x)
call show(y)
call show(z)
call show(x+y)
call show(x-y)
```

のような記述が可能になるはずである．

課題 5

有理数を表す構造体を扱うモジュールを作成せよ．有理数同士の和差積商の演算子もそれぞれ定義すること．さらに，代入演算子で長さ 2 の整数配列を受け取れるようにし，また有理数を標準出力に表示するサブルーチン (例えば `show`) も作成せよ．(有理数の分子および分母は整数であるので 2 つの整数型を持つ構造体として定義すればよい．)

```
type(rational) :: a, b

a = (/1, 4/)
b = (/2, 5/)

write(*, fmt='(a)', advance='no') 'a     = '
call show(a)

write(*, fmt='(a)', advance='no') 'b     = '
call show(b)
```

```
write(*, fmt='(a)', advance='no') 'a + b = '
call show(a+b)

write(*, fmt='(a)', advance='no') 'a - b = '
call show(a-b)

write(*, fmt='(a)', advance='no') 'a * b = '
call show(a*b)

write(*, fmt='(a)', advance='no') 'a / b = '
call show(a/b)
```

例えば上のようなプログラムをコンパイルして実行した結果が以下のようになればよい．約分も忘れずにすること．最大公約数を求める関数もしくはサブルーチンを用いると良い．(絶対値に注意すること．)

```
$ ./a.out
a     =      1 /      4
b     =      2 /      5
a + b =     13 /     20
a - b =     -3 /     20
a * b =      1 /     10
a / b =      5 /      8
```