第 7 章 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

与えられた倍精度実数 $a(>0)$ の平方根の近似値を返す関数を実装せよ．ただし平方根は以下のような逐次近似で計算するものとする．

$$x_{n+1} = \frac{1}{2}\left(\frac{a}{x_n} + x_n\right)$$

ここで x_n は $\sqrt{a}$ の n 番目の近似値である．初期値としては $x_0 = a$ を与え，反復は例えば $\epsilon = 10^{-5}$ に対して，$\|x_{n+1} - x_n\| < \epsilon\|x_n\|$ となるまで繰り返せば良い．(反復の途中結果は必要ないので x_n に配列を使う必要はないことに注意せよ．)

実装した関数と組み込み関数 `sqrt` の結果を比較し， ϵ で与えた精度の範囲内で正しいことを確認すること．例えば，以下は標準入力から与えられた数値 (この例では 2.0) の平方根を計算して表示する例である．

```
$ ./a.out
2.0
sqrt(x) =    1.4142135623730951
approx  =    1.4142135623746899
```

課題 3

テストの点数が整数配列から与えられた時にヒストグラムを作成するサブルーチン `histogram` を実装せよ．例えば点数配列とビン幅を入力とし，作成されたヒストグラムの各ビンの中央値，各ビン内の人数を出力とする以下の様な形式のサブルーチンを作成すればよい．ただし，与える整数は $0 \le n < 100$ とするが，もしこの範囲を超えた入力があった場合にはエラーを表示して終了すること．

```
subroutine histogram(score, binw, binc, hist)
  implicit none
  integer, intent(in)  :: score(:)  ! 点数 (人数分)
  integer, intent(in)  :: binw      ! ビンの幅 (例えば 10 点)
  real(8), intent(out) :: binc(:)   ! ビンの中央値 (例えば 5, 15, ..., 95)
  integer, intent(out) :: hist(:)   ! 各ビン内の人数

  ! ここでヒストグラムを作成

end subroutine histogram
```

この場合のように固定幅のビンでヒストグラムを作成するのは簡単である．i 番目の点数がヒストグラムの j 番目の要素に入るとすると， j は

```
j = score(i) / binw + 1
```

のように求められる (複雑な if 文による分岐は必要ない!)．このインデックスが配列 hist の上限と下限の間に収まっていなければエラーとすれば良い．

このサブルーチンを用いて， score2.dat からデータを読み込みヒストグラム作成するプログラムを作成せよ．またその結果を gnuplot で図示せよ (with boxes を用いると良い)．出力結果は例えばビン幅を 10 とした場合には以下のようになる．

```
$ ./a.out < score2.dat
  5.0000000000000000                0
  15.000000000000000                3
  25.000000000000000               26
  35.000000000000000               63
  45.000000000000000              152
  55.000000000000000              248
  65.000000000000000              254
  75.000000000000000              159
  85.000000000000000               80
  95.000000000000000               15
```

なお score2.dat の形式は 5 章の課題 3 の score1.dat と同じであるが，データは異なるものになっている．

同様に範囲外のデータを含む score3.dat を読みこませると

```
$ ./a.out < score3.dat
  Invalid input
```

のようなエラーを表示するように実装せよ．

課題 4

データ $x_1, x_2, \ldots, x_n$ を大きさの順に並べ替える処理をソートという．ソートのアルゴリズムの中でも一番簡単なのがバブルソートと呼ばれるものである．これは以下の処理を $n-1$ 回繰り返すことで実現される．

> $i = 1, \ldots, n-1$ まで順に x_i と x_{i+1} の大小を比較し, $x_i > x_{i+1}$ なら順番を入れ替える

この処理を k 回行うと x_i のうち k 番目に大きい要素が x_{n-k+1} に配置されるので， $n-1$ 回繰り返すことで全ての要素を並び替えることが出来る．

ただし 1 回目の処理が終了した時点で最大値が x_n になっており，この要素については並べ替えの必要がない．従って 2 回目は $x_1, \ldots, x_{n-2}$ までを処理すれば十分である．同様に m 回目の処理が終了した時点で $x_{n-m+1}, \ldots, x_n$ までの位置は確定しているので， $m+1$ 回目には $x_1, \ldots, x_{n-m}$ までの処理を行えばよい．これによって比較回数を $(n-1)^2$ 回から半分に減らすことができる．

このバブルソートによって整数配列をソートするサブルーチン bsort を実装せよ．これは例えば以下のような形になるだろう．

```
subroutine bsort(array)
  implicit none
  integer, intent(inout) :: array(:) ! 配列にはソートされた結果が代入される

  ! バブルソート

end subroutine bsort
```

作成したプログラムを用いて rand.dat のデータをソートし，結果が正しいことを確認せよ．ここでもデータファイルの形式は上の score2.dat と同一である．従って同様にリダイレクトで

```
$ ./a.out < rand.dat
```

のように読みこませれば良い．

課題 5

データ $x_1, x_2, \ldots, x_n$ からある値 X に等しいものを探しだす処理を探索という．その中でも以下は二分探索と呼ばれるアルゴリズムである．

▷ まず $x_i (i = 1, \ldots, n)$ をソートする．
▷ $l = 1, r = n$ として以下の処理を繰り返す．
[1] $l > r$ ならば失敗 (見つからなかったので処理を終える)．
[2] $m = (l + r)/2$ とし，$X = x_m$ なら成功 (見つかったので処理を終える)．
[3] $X > x_m$ なら $l = m + 1$ ， $X < x_m$ なら $r = m - 1$ として [1] に戻る．

これをサブルーチン bsearch として実装せよ．

また open 文で rand.dat からデータを読み込み， bsort で配列をソートした後に bsearch で実際に適当な値の探索を行うプログラムを作成せよ．例えば標準入力から与えられた値をソート後の配列から探索を行い，結果を表示するようにすれば良い．この時の実行結果は以下のようになる．(この例では 10 や 2004 が入力値である．)

```
$ ./a.out
Input an integer : 10    # キーボード入力
      10 was not found !

$ ./a.out
Input an integer : 2004  # キーボード入力
    2004 was found at index      100
```

なおサブルーチン bsearch は例えば以下のような形式とすればよい．見つかった場合には探しだす値と等しい値が格納されている配列のインデックスを，見つからなかった場合には-1 を仮引数 idx に代入して返すようにするとよいだろう．

```
subroutine bsearch(array, var, idx)
  implicit none
  integer, intent(in)  :: array(:) ! ソートされた配列
  integer, intent(in)  :: var      ! 探したい値
  integer, intent(out) :: idx      ! 見つかった要素へのインデックス

  ! 二分探索

end subroutine bsearch
```

課題 6

1 行に 1 つの英単語が記述されているファイルを読み込み，英単語を辞書順にソートして出力するプログラムを作成せよ．空白行かファイルの終端に達するまで全ての単語を読み込むこと．ただし英単語は 10 文字以下，また読み込む単語の数は 100 個以下であると仮定してよい．words.txt を入力として作成したプログラムの動作を確認せよ．

実行結果は例えば以下のようになるだろう．(空白行の後の NOT_TO_BE_READ は読まれていないことに注意!)

```
$ ./a.out < words.txt
*** before sort ***
tomato     banana     apple      grape      watermelon plum
*** after sort ***
apple      banana     grape      plum       tomato     watermelon
```

なお，文字列の大小比較は辞書順となる (例えば`'apple' < 'banana'`) ので，`bsort` を一部修正するだけで文字列のソートが出来る．文字型変数の配列は例えば

```
character(len=10) :: char_array(100)
```

などのように宣言すれば良い．ここで `char_array` は 10 文字分の文字列を保持する長さ 100 の配列である．

課題 7 †

与えられたファイルに含まれるアルファベットの各文字の出現回数をヒストグラムとして標準出力に表示するプログラムを作成せよ．ファイルの長さは任意とする．即ちファイルの終端まで全ての文字を読み込まなければならない (空白行があっても読み込み続ける)．ただし以下の条件を満たすこと．

- ▷ アルファベット `a-z`，`A-Z` 以外は無視してよい．
- ▷ 大文字と小文字は区別しない．
- ▷ 文字数があまりに多い時にはヒストグラムを適当に規格化すること．

wikipedia.txt (Wikipedia より引用) を処理した時には例えば以下のような出力となるだろう．この例では最大で 60 個の`'o'` が出力されるように規格化してある．各アルファベットの後の括弧内の数字は文字数を表している．(wikipedia.txt で試す前に words.txt などの小さいデータで試した方が良いだろう．)

```
$ ./a.out < wikipedia.txt
A( 110):oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
```

```
B(   9):ooooo
C(  44):oooooooooooooooooooooooo
D(  32):oooooooooooooooo
E(  99):ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
F(  31):oooooooooooooooo
G(  38):oooooooooooooooooooo
H(  32):oooooooooooooooo
I(  87):ooooooooooooooooooooooooooooooooooooooooooooooooooo
J(   2):o
K(   2):o
L(  38):oooooooooooooooooooo
M(  49):ooooooooooooooooooooooooooo
N(  91):ooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
O(  81):ooooooooooooooooooooooooooooooooooooooooooo
P(  39):ooooooooooooooooooooo
Q(   0):
R(  93):ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
S(  63):ooooooooooooooooooooooooooooooooo
T(  79):ooooooooooooooooooooooooooooooooooooooooooooo
U(  37):ooooooooooooooooooo
V(  14):oooooooo
W(   5):ooo
X(   1):o
Y(  15):oooooooo
Z(   0):
```

組込み関数 ichar ， char を用いるとよい．また規格化の際には四捨五入をする関数 nint を用いることが出来る．

特に仕様は限定しないが，関数やサブルーチンを使った分り易いプログラムを目指すこと．