第４回(5/14)

８．配列と文字列

〔１〕１次元配列

例題１－２２

5 個の整数要素をもつ配列を初期化して配列全体を出力し，その合計と平均を求めるプログラムをつくりなさい。ただし，5 個の数値を 13，56，27，44，39 とする。

▼出力結果１－２２

```
13    56    27    44    39
合計 = 179, 平均 = 35.80
```

考え方

配列の初期化は，宣言したときに行う。ループ処理によって 1 番目の要素から順次累算して合計を求める。平均値は整数でなく実数となるので，キャスト演算子によって型変換を行う。

▼プログラム１－２２

```
01      /* E1-22 */
02      /* 配列の合計と平均 */
03      #include <stdio.h>
04      #define   SIZE   5
05
06      main()
07      {
08          static int dim[SIZE] = {13, 56, 27, 44, 39};
09          int i, sum;
10          float ave;
11          sum = 0;
12          for (i = 0 ; i < SIZE ; i++) {
13              sum += dim[i];
14              printf("%5d", dim[i]);
15          }
16          ave = (float) sum / SIZE;
17          printf("¥n 合計 = %d,平均 = %.2f¥n", sum, ave);
18      }
```

《キャスト演算子》

次の形式を取る。

（*type-name*） expression

キャストは expression の値を *type-name* で指定された型に強制的に変換する。ただし，expression 自身は変化しないことに注意する。キャスト演算子は他の単項演算子と同じ優先度を持つ。

練習問題１６

１．10 個の適当な整数を配列に格納し（初期化），これらの順序を入れ換えて（並びを反転する），元の配列内容と，並びを反転した後の配列内容を合わせて出力するプログラムをつくりなさい。

《ヒント》例えば，最初の d[0]と最後の d[9]の値を入れ換える手順は次のようになる。
作業用の変数 dm を用意して
① d[0]の内容を dm に退避
② d[9]の内容を d[0]に代入
③ dm に退避している内容を d[9]に代入

▼出力例

元の配列：
13　56　27　44　39　22　51　8　87　96
反転した配列：
96　87　8　51　22　39　44　27　56　13

練習問題１６

２．10 個の適当な整数を配列に格納し（初期化），最大値と最小値をみつけるプログラムをつくりなさい。

《ヒント》この問題の擬似コードは,次のようになる。

```
最初の要素をとりあえず最大値，最小値とする
配列要素の添え字の数を進める
while（まだ配列がある）{
    if（その配列要素は最大値より大きい）
        その要素を最大値とする
    else if（その配列要素は最小値より小さい）
        その要素を最小値とする
    else
        なにもしない
    配列要素の添え字を進める
}
配列全体，および最大値と最小値を出力する
```

▼出力例

```
10 個の整数の配列：
  47  29  64  15  97  38  50  88  31  76
最大値 = 97,  最小値 = 15
```

練習問題１６

３．10 個の実数データをキーボードから入力し，配列 data[10]に格納しなさい。そして，各要素の小さい方からの順位を決定して，別の配列 rank[10]に格納しなさい。最後に，入力データ data[10]と順位 rank[10]をともに出力しなさい。

《ヒント》順位を格納する配列 rank[10]は，全要素を１で初期化しておくとよい。
そして，自分より小さな data 要素があったら，rank 要素を１プラスする。

▼出力例

キーボードから実数データを 10 個入力して下さい！
1 番目：0.53↵
2 番目：1.2↵
3 番目：55.2↵
4 番目：10.4↵
5 番目：3.45↵
6 番目：8.12↵
7 番目：0.1↵
8 番目：100.0↵
9 番目：357.0↵
10 番目：7.6↵

10 個の実数並びと小さい方からの順位：

0.53	1.20	55.20	10.40	3.45	8.12	0.10	100.00	357.00	7.60
2	3	8	7	4	6	1	9	10	5

練習問題１６

４．次のプログラムは，10 個の整数データを降順に整列させるものである。①～⑤を埋めて正しく動作するようにしなさい。

```
/* Q16-4 */
/* 降順ソート */
#include <stdio.h>

main()
{
    static int dd[10] = { 27, 89, 45, 18, 55,
                          64, 92, 73, 34, 88 };
    int jj, kk, [ ① ] ;
    for (jj = 0; jj < 10; jj++)
        printf("%4d", dd[jj]);
    printf("¥n");
    for (jj = 0; jj < 10-1; jj++) {
        for ( [ ② ] ; kk < 10; kk++) {
            if (dd[jj] < [ ③ ] ) {
                dm = dd[jj];
                [ ④ ] = dd[kk];
                dd[kk] = [ ⑤ ] ;
            }
        }
    }
    for (jj = 0; jj < 10; jj++)
        printf("%4d", dd[jj]);
    printf("¥n");
}
```

練習問題１６

５．(難問) 配列 a の n 個の互いに異なる要素 a[i](i = 0, 1, ..., n-1)が次のように昇順：

a[0] < a[1] < ... < a[n-1]

に並んでいる。このとき，キーx がこの配列 a のどこにあるかを探索する。この問題を２分探索 (binary search) により解くことを考える。

２分探索のおおまかな戦略は次の通りとなる。

「まず，列の中央の要素から始める。もし，この要素が探している x と異なっていれば，x がその要素に比べて小さいかまたは大きいかによって，もとの列の左または右半分の探索に移行し，同じ処理を繰り返す。」

しかし，この記述だけでは，不正確であり，また不完全でもある。そこで問題を明確にして，キーx がこの配列 a のどこにあるか，ない場合はどこにいれるべきか，の位置 site を次式：

site = 0　x ≤ a[0] のとき

site = n　x > a[n-1] のとき

site = j　j (1 ≤ j ≤ n-1)に対して，a[j-1] < x ≤ a[j]

で決定することとする。これにより，２分探索アルゴリズムが正確に記述される。

次のプログラムは２分探索をインプリメントしたものである。正しく動作するように①～④を埋めなさい。

```
/* Q16-5 */
/* ２分探索 */
#include <stdio.h>

main()
{
    static int a[10] = { 2, 9, 12, 18, 35,
                         37, 55, 63, 84, 99 };
    int i;
    int site, x, n = 10;
    int middle, left = 0, right = n - 1;

    printf("次の整数並びがあります¥n");
    for (i = 0; i < n; i++)
        printf("%d¥t", a[i]);
```

```
    printf("¥n¥n キーを入力してください：");
    scanf("%d", &x);

    if (x <= a[left]) {
        site = 0;
    } else if (    ①    ) {
        site = n;
    } else {
        while (right - left > ② ) {
            middle = (right + left) / 2;
            if (    ③    )
                right = middle;
            else
                left = middle;
        }
        site =    ④    ;
    }

    if (a[site] == x)
        printf("あなたが入力したキー %d は %d 番目にあります¥n", x, site + 1);
    else
        printf("あなたが入力したキー %d は %d 番目に入ります¥n", x, site + 1);
}
```

〔２〕文字型配列

例題１－２３

文字型配列の各要素がどのような値になるかを調べるプログラムをつくりなさい。

▼出力結果１－２３

```
tokyo
str[0] = t, (74)
str[1] = o, (6f)
str[2] = k, (6b)
str[3] = y, (79)
str[4] = o, (6f)
str[5] =  , (0)

edo
str[0] = e, (65)
str[1] = d, (64)
str[2] = o, (6f)
str[3] =  , (0)
str[4] = o, (79)
str[5] =  , (0)
```

考え方

文字列を扱うライブラリ関数である strcpy 関数を使用するため，ヘッダファイル string.h の include が必要である。文字列配列へ文字列がセットされる様子を確認するため，長さの異なる 2 種類の文字列を準備する。for 文で文字型配列の全要素をその値（JIS コード，16 進数）とともに出力する。

▼プログラム１－２３

```
01      /* E1-23 */
02      /* 文字型配列の要素 */
03      #include <stdio.h>
04      #include <string.h>
05
06      main()
07      {
08          char str[6];
09          int i;
10          strcpy(str, "tokyo");
11          printf("¥n%s¥n", str);  /* %s 変換での引数は char 型配列の配列名か
12                                      char 型のポインタ変数 */
13          for (i = 0 ; i < 6 ; i++)
14              printf("str[%d] = %c, (%x)¥n", i, str[i], str[i]);
15          strcpy(str,"edo");
16          printf("¥n¥n%s¥n", str);
17          for (i = 0 ; i < 6 ; i++)
18              printf("str[%d] = %c, (%x)¥n", i, str[i], str[i]);
19      }
```

練習問題１７

１．文字型配列を宣言して，次のように文字列を初期化しなさい。

```
static char str[] = "I am a student.";
```

この文字列の長さ（文字数）を計数（カウント）するプログラムをつくりなさい。但し，文字列長を計算するライブラリ関数 strlen は使用しないこと。

《ヒント》空白や記号「’‘」等も文字列の一部として取扱い，正しい長さを出力すること。
文字列の終端を表すナル文字（¥0）はカウントしない。

〔３〕文字列処理

例題１－２４

出力結果１－２４のように，24 文字以内の英文字列を 2 組（文字列 1 と文字列 2）入力すると，それぞれの文字列の長さを出力するプログラムをつくりなさい。次に，その文字列の長さを比較して出力するプログラムをつくりなさい。すなわち，文字列１のほうが長い場合は「○○○＞△△」，等しい場合は「○○＝△△」，短い場合は「○○＜△△△」のように出力すること。

▼出力結果１－２４

文字列１は？mouse↵
文字列２は？multimedia↵

mouse の文字数は 5 です。
multimedia の文字数は 10 です。

mouse ＜ multimedia

考え方

文字列の長さを求めるには，strlen 関数を用いる。

▼プログラム１－２４

```
01      /* E1-24 */
02      /* 文字列処理 */
03      #include <stdio.h>
04      #include <string.h>
05
06      main()
07      {
08          char a[25], b[25];
09          int x, y;
10
11          printf("文字列１は？");
12          scanf("%s", a);  /* 配列名 a は配列 a の先頭アドレスを表す定数 */
13          printf("文字列２は？");
14          scanf("%s", b);
15
16          x = strlen(a);
17          y = strlen(b);
18          printf("¥n%s の文字数は%d です。¥n", a, x);
19          printf("%s の文字数は%d です。¥n", b, y);
20
21          if (x - y > 0)
22              printf("¥n%s > %s¥n", a, b);
23          else if (x == y)
24              printf("¥n%s = %s¥n", a, b);
25          else
26              printf("¥n%s < %s¥n", a, b);
27      }
```

《補足例題》

次のプログラムは，自分の年齢を入力すると，“私の年齢は，○○歳です。”と出力するものである。プログラムの①，②を埋めて，正しく動作するように完成してみましょう。

```
#include <stdio.h>
#include <  ①  .h>

main()
{
    char a[20] = "私の年齢は，";
    char b[5];
    printf("私の年齢は？");
    scanf("%s", b);
    ②  (a, b);
    printf("¥n%s 歳です¥n", a);
}
```

９．ポインタ

〔１〕ポインタとは

Ｃでは,アドレス値を代入できるポインタ変数が用意されている。ポインタ変数をうまく利用すれば，すっきりしたプログラムの記述ができるが，実際の記憶領域を直接操作するので，思いもかけない結果が生じることがある。
ポインタの概念を理解するために，次の例を考えてみる。
10 号室（アドレス）の田中さんの部屋（変数として確保された記憶領域）に荷物（データ）を届ける場合を考えよう。
この場合，2 種類の荷物の行き先を表示する方法があり，
①「田中様」（変数名表示）
②「10 号室行」（アドレス表示）
である。
①では，荷札に「田中様」とあるため配達人はそれを見て荷物を田中さん本人に手渡す。
②では,「10 号室行」とあるため配達人は箱（ポインタ変数）に入っている鍵（アドレス値）で 10 号室の扉を開けて荷物を置いてくる。
荷物が届くという結果は同じであるが，ポインタを用いたときは田中さん本人の了解（変数名の使用）がなくてもよいことに注意する。
箱に入っている鍵を取り替えれば，どの部屋にでも荷物を届けることができるし，荷物を受け取ることもできる。

〔２〕ポインタの基礎

単項演算子&はメモリ中のオブジェクトのアドレスを与える。例えば，次のように

```
char *p, c = 'A';
p = &c;
```

を実行するとポインタ変数 p に c のアドレスが代入される。&演算子はメモリ中のオブジェクト（変数および配列の要素）にのみ適用できる。従って，&演算子は式や定数，レジスタ変数には適用できないことに注意する。

一方，単項演算子*は，間接演算子または逆参照演算子と呼ばれる。これをポインタ変数に適用すると，そのポインタが指すオブジェクトの中身を取り出すことができる。

次のコードで，ポインタ変数の宣言のしかたと，&と*の使い方を見てみよう。

```
int x = 1, y = 2, z[10];
int *ip;         /* ip は int 型へのポインタ変数である */
ip = &x;         /* ip はいま x（のアドレス）を指す */
y = *ip;         /* y はこれで（2 から）1 になる */
*ip = 0;         /* x はこれで（1 から）0 になる */
ip = &z[0];      /* ip はいま z[0]（のアドレス）を指す */
```

例題１－２５

各種の変数（char, int, float）でポインタを用いた場合，それらの変数に割り当てられるアドレス（ポインタの値）とポインタが示す内容（データ値）を出力するプログラムをつくりなさい。

▼出力結果１－２５

ｃのアドレス = 12FF7C, ｃ = C
ｉのアドレス = 12FF74, ｉ = 35
ｆのアドレス = 12FF6C, ｆ = 12.340000

考え方

ポインタ変数の宣言を行い，データ値が代入されている変数のアドレスをセットする。ポインタ変数の値とポインタが指す変数の値を printf 関数で書式を合わせて出力する。

▼プログラム１－２５

```
01      /* E1-25 */
02      /* ポインタのはたらき */
03      #include <stdio.h>
04
05      main()
06      {
07          char c, *pc;
08          int i, *pi;
09          float f, *pf;
10          c = 'C';
11          i = 35;
12          f = 12.34;
13          pc = &c;
14          pi = &i;
15          pf = &f;
16          printf("c のアドレス = %X, c = %c¥n", pc, *pc);
17          printf("i のアドレス = %X, i = %d¥n", pi, *pi);
18          printf("f のアドレス = %X, f = %f¥n", pf, *pf);
19      }
```

練習問題１７

２．次のコードは間違っていて，変数 q の値が正しく 1234.34 と表示されません。まず，ここでのポインタの使い方の誤りがどこにあるかを指摘しなさい。次に，このコードでは実際にどんな処理が実行されているかを考えて，1234.34 とうまく表示されない理由を説明しなさい。

《注意》Linux 上の gcc でコンパイルして下さい。警告が出ますが，実行ファイルは作成されます。Windows 上の Visual C++ではコンパイルが通りません。

```
/* Q17-2 */
/* ポインタの型とポインタの指すオブジェクト */
#include <stdio.h>

main()
{
    int *p;
    double q, temp;

    temp = 1234.34;

    p = &temp;
    q = *p;   /* q に値を代入しようとしているが… */

    printf("temp = %f¥n", temp);
    printf("q = %f¥n", q);   /* これでは 1234.34 は表示されない */
}
```