5　C 言語によるプログラミング 5: あたらしいデータの型の構築

目 次

5　C 言語によるプログラミング 5: あたらしいデータの型の構築　i

5.1　概説 [1, 2, 3] 69

5.2　配列 [1, 3] 69

5.2.1　整数型の配列 69

5.2.2　文字配列 74

図 目 次

5.1 概説 [1, 2, 3]

§ 2 では、いろいろな変数の型を宣言し、さらに初期化して、ソースプログラムでもちいた。これらの変数の型が、変数の基本要素、言わば、基本ブロックである。この基本ブロックを組み合わせて、複雑なデータ構造をもったあらたな変数の型をつくりだすことができる。化学で原子を組み合わせて分子ができるようなものである。あるいは、生物学で、塩基の組み合わせで遺伝子が構成されることを想起してもよいであろう。

これらのあらたな変数の型にはつぎのようなものがある。

(i) おなじ型の変数を連続的にメモリ空間に配置したもの (『配列』)。

(ii) 複数個の変数 (おなじ型、あるいは別の型を問わず) をまとめて、組にして、ひとつの変数の型にしたもの (『構造体』)。

(iii)(ii) でつくった変数を基本単位にして、(i) の配列を構成する (『構造体配列』)。これは、データベースの原型である。

上記の (i)~(iii) は、ポインタ変数 (ほかの変数の住所 = アドレス メモリ位置を記憶している変数と密接に関連している。しかし、ここでは、ポインタ変数を顕に出さない形で話をすすめる。

5.2 配列 [1, 3]

5.2.1 整数型の配列

おなじ型の変数を連続的にメモリ空間に配置したものを「配列」という。おなじ型の変数を、おなじ目的で、複数個つかうことがよくある。このような場合、変数の宣言をいちいちしなくても、配列としてまとめて宣言しておけば便利である。ここでは整数型変数の配列をかんがえているが、ほかの型の配列も基本的なことはおなじである。

宣言と初期化

例 1

```
/* array1.c */

#include <stdio.h>

main()
{
   int i; /*   配列の添字    */
   int a[13] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
    /*  整数型配列の宣言および初期化
   int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
        としてもよい。     */

   printf("ソースコードに用意された配列を印字します。\n\n");
   printf("順方向\n");
   for (i = 0; i <= 12; i++)
   {
      printf("a[%3d]: %3d\n", i, a[i]);
   }
```

```
   printf("\n 逆方向\n");
   for (i = 12; i >= 0; i--)
   {
      printf("a[%3d]: %3d\n", i, a[i]);
   }
   printf("\n");

}
```

実行例

$ ./a.out ソースコードに用意された配列を印字します。

順方向

```
a[  0]:   1
a[  1]:   2
a[  2]:   3
a[  3]:   4
a[  4]:   5
a[  5]:   6
a[  6]:   7
a[  7]:   8
a[  8]:   9
a[  9]:  10
a[ 10]:  11
a[ 11]:  12
a[ 12]:  13
```

逆方向

```
a[ 12]:  13
a[ 11]:  12
a[ 10]:  11
a[  9]:  10
a[  8]:   9
a[  7]:   8
a[  6]:   7
a[  5]:   6
a[  4]:   5
a[  3]:   4
a[  2]:   3
a[  1]:   2
a[  0]:   1
```

整数型配列の宣言・初期化の例

```
int a[100]; (宣言)
a[2] = 3; (初期化)
int a[3] = {1, 2, 3}; (宣言・初期化)
int a[] = {1, 2, 3}; (宣言・初期化)
```

上記の配列を例にとると、配列の添字と要素の対応は以下のようになっていることがわかる。添字は 0 ～ (配列のおおきさ − 1) の範囲を走ることに注意が必要である。

i	0	1	2	3	4	5	6	7	8	9	10	11	12
a[i]	1	2	3	4	5	6	7	8	9	10	11	12	13

表 1: 配列の添字と要素

標準入力による初期化

標準入力による初期化は、ふつうの変数をあつかうときとおなじである。つぎの例は整数を逐次入力して、後でまとめて出力する。また、そのなかの最大値とそれが何番目に入力したものか、をあたえるものである。

例 2

```
/* array2.c */

#include <stdio.h>

main()
{
   int i; /*   配列の添字   */
   int a[100];   /*  整数型配列の宣言   */
   int i_aMax; /*  最大の配列要素をあたえる添字 i   */
   int i_max; /* 添字 i じたいの最大値   */

   printf("整数を逐次入力してください。その最終的な和をもとめます。\n");
   printf("終了するときは整数以外の文字を入力してください: \n");

   i = 0;
   i_aMax = 0;
   while (scanf("%d", &a[i]) == 1)
   {
      if (a[i_aMax] < a[i])
      {
          i_aMax = i;
      }
```

```
      i++;
   }
   i_max = i - 1;

   printf("入力された整数は以下のとおりです。\n");

   for (i = 0; i <= i_max; i++)
   {
      printf("a[%d]: %d\n", i, a[i]);
   }

   printf("最大値は%d 番目に入力した a[%d]: %d です。\n", i_aMax + 1, i_aMax,
a[i_aMax]);
}
```

実行例

```
$ ./a.out
整数を逐次入力してください。その最終的な和をもとめま す。
終了するときは整数以外の文字を入力してください:
1
13
110
25
58
a
入力された整数は以下のとおりです。
a[0]: 1
a[1]: 13
a[2]: 110
a[3]: 25
a[4]: 58
最大値は 3 番目に入力した a[2]: 110 です。
```

つぎの例は、10 進整数の各桁からその整数を再現するものである。だが、いっぱんの p 進数にもつかえることはあきらかである。べき乗 (累乗) を逐次的にあつかうこの旨い方法はしばしばつかわれる。これには、ホーナー (Horner) の方法という呼び名がついている [4]。

例 3

```
/* array3.c */

#include <stdio.h>
main()
{
```

```
   int i = 0; /*  配列の添字  */

   int a[100]; /*  整数の各桁の数  */

   int number; /*  最終的に得る整数の各桁の数  */

   printf("整数を逐次入力してください。\n");
   printf("それらを各けたとする整数をもとめます。\n");
   printf("終了するときは整数以外の文字を入力してください: \n");

   for (number = 0; scanf("%d", &a[i]) == 1; i++)
   {
      number = number * 10 + a[i];
   }

   printf("その整数は、%d です。\n", number);
}
```

実行例 1

```
$ ./a.out
整数を逐次入力してください。
それらを各桁とする整数をもとめます。
終了するときは整数以外の文字を入力してください:
1
2
3
a
その整数は、123 です。
```

一気に 123a と入力してもおなじ結果を得る。

実行例 2

```
$ ./a.out
整数を逐次入力してください。
それらを各けたとする整数をもとめます。
終了するときは整数以外の文字を入力してください:
123a
その整数は、123 です。
```

この方法では、$123 = 1 \times 10^2 + 2 \times 10 + 3$ とするのはなく、
$123 = ((1 \times 10 + 2) \times 10) + 3$ というように、べき乗を逐次的にあつかっていることがわかる。また、この結果を得るためには、ことさら配列をつかわなくともよい、ということはあきらかである。というのも、いまの場合は最終的な答えを出力すればよいのである。したがって、コンパイラが最後まで各桁の数字をつかう必要はないからである。

問題によっては、あとで各桁の数字をつかうときもある。このときは、もちろん、配列が効いてくる。

5.2.2 文字配列

C では「文字列」は「文字配列」としてあつかわれる。「文字列型」という変数の型はない。

宣言と初期化

例 4

```
/* charray1.c */

#include <stdio.h>

main()
{
   int i; /*   配列の添字   */
   char a[20] = "BiwakoKusatsu";
    /*  文字配列の宣言および初期化。
    これは、char a[20] = {'B', 'i', 'w', 'a', 'k', 'o', 'K', 'u', 's',
'a', 't', 's', 'u', '\0'};
    を略記したものとみることができる。
      また、char a[] = "BiwakoKusatsu"; としてもよい。     */

   printf("ソースコードに用意された文字配列を印字します。\n\n");
   printf("まず、一行で書くと、\n");
   printf("%s", a);
   printf("\n となります。\n\n");
   printf("つぎに、縦方向に一文字ずつ出力すると、\n\n");
   printf("順方向\n");
   for (i = 0; i <= 12; i++)
   {
      printf("a[%3d]: %c\n", i, a[i]);
   }

   printf("\n 逆方向\n");
   for (i = 12; i >= 0; i--)
   {
      printf("a[%3d]: %c\n", i, a[i]);
   }
   printf("\n となります。\n");


}
```

実行例

```
$ ./a.out
ソースコードに用意された文字配列を印字します。
```

まず、一行で書くと、
BiwakoKusatsu
となります。

つぎに、縦方向に一文字ずつ出力すると、

順方向
```
a[  0]: B
a[  1]: i
a[  2]: w
a[  3]: a
a[  4]: k
a[  5]: o
a[  6]: K
a[  7]: u
a[  8]: s
a[  9]: a
a[ 10]: t
a[ 11]: s
a[ 12]: u
```

逆方向
```
a[ 12]: u
a[ 11]: s
a[ 10]: t
a[  9]: a
a[  8]: s
a[  7]: u
a[  6]: K
a[  5]: o
a[  4]: k
a[  3]: a
a[  2]: w
a[  1]: i
a[  0]: B
```

となります。

文字配列の宣言・初期化の例

```
char a[100]; (宣言)
a[2] = 'o'; (初期化)
char a[6] = "Kyoto"; (宣言・初期化)
char a[] = "Kyoto"; (宣言・初期化)
```

上記の文字配列の添字と要素は以下の如くになっている。

i	0	1	2	3	4	5
a[i]	K	y	o	t	o	\0

表 2: 文字配列の添字と要素

最後に文字列の終わりをしめす記号であるヌル文字「\0」が入っていることに注意する必要がある。これが文字列を閉じてくれるのである。文字配列の宣言するとき、ヌル文字の分も配列のおおきさに込められている。

標準入力による初期化

標準入力での書式指定では、%s とする。ただし、ほかの変数のときのように、**読み込む変数に&をつける必要はない。**この点くれぐれも御注意を!!!

つぎの例は、キーボードから入力した文字列を、メモリのおなじところに逆順に上書きするものである [5]。

例 5

```
/* charray2.c */
/* reverse */

#include <stdio.h>
#include <string.h>    /* strlen */

main()
{
   int i, j; /*   配列の添字    */
   int n; /*   文字列の長さ    */
   char a[100];  /*  文字配列の宣言。    */
   char temp; /* 文字列を逆にするときの文字の臨時格納庫    */
   printf("文字列を入力してください。メモリ内のおなじ位置で、\n");
   printf("逆に並べ替えて出力します。\n");
   scanf("%s", a);

   n = strlen(a) - 1;
   j = n;

   for (i = 0; i < j; i++)
   {
      j = n - i;

      temp = a[i];
      a[i] = a[j];
      a[j] = temp;
   }
```

```
   printf("あたえられた文字列の順を逆にしてできた文字列は\n%s\n です。\n", a);
}
```

```
$ ./a.out
文字列を入力してください。メモリ内のおなじ位置で、
逆に並べ替えて出力します。
Kaguragaoka
あたえられた文字列の順を逆にしてできた文字列は
akoagarugaK
です。
```

文字列入力のとき、scanf では、空白が文字 列の切れ目と認識される。
`scanf("%s %s %s", a, b, c);` として、
`Kyoto Osaka Kobe`
を入力すると文字列 a, b, c にそれぞれ Kyoto, Osaka, Kobe が入る。だが、空白も込めて 1 行そのままひとつの文字列に入れるにはどうしたらよいか? これには、関数 `fgets` をつかうとよい。以下の例をみよう。

例 6

```
/* toupepr-string1.c */
/* 文字列を一行入力 */
/* ( ) のなかの文字を大文字にする。 */
/* 複数の括弧の場合 */

#include <stdio.h>
#include <ctype.h>
#include <string.h>

main()
{
   int i;
   char a[100];

   printf("(英数) 文字列を一行あたえてください。");
   printf("丸括弧 () に入った文字列を大文字にします。一行あたえてください。\n");
   fgets(a, 80, stdin);  /* 文字列を一行標準入力して、配列 a に入れる。 */
   printf("いまの文字列を一文字ずつみていくと、つぎのようになります。\n");

   for (i = 0; i <= strlen(a); i++)
   {
      printf("a[%3d]: %c %x\n", i, a[i], a[i]);
   }

   i = 0;
   while (a[i] != '(' )
```

```
   {
      i++;               /* 左括弧「(」に出会うまですすむ */
   }

   i++;          /* 左括弧「(」のつぎの文字の配列の添え字 */

   while (a[i] != ')' )     /* 右括弧「)」に出会うまでくりかえす。*/
   {
      a[i] = toupper(a[i]); /* 大文字にする。*/
      i++;                       /* つぎの文字にすすむ。*/
   }

   a[strlen(a) - 1] = '\0'; /* 改行コードを消す。*/

   printf("\n あらたな文字列は、\n%s です。\n", a);

   printf("一文字ずつみていくと、つぎのようになります。\n");

   for (i = 0; i <= strlen(a); i++)
   {
      printf("a[%3d]: %c %x\n", i, a[i], a[i]);
   }

}
```

$ ./a.out(英数) 文字列を一行あたえてください。丸括弧 () に入った文字列を大文字にします。一行あたえてください。

```
Kyoto (Osaka) Kobe
いまの文字列を一文字ずつみていくと、つぎのようになります。
a[  0]: K 4b
a[  1]: y 79
a[  2]: o 6f
a[  3]: t 74
a[  4]: o 6f
a[  5]:   20
a[  6]: ( 28
a[  7]: O 4f
a[  8]: s 73
a[  9]: a 61
a[ 10]: k 6b
a[ 11]: a 61
a[ 12]: ) 29
a[ 13]:   20
a[ 14]: K 4b
a[ 15]: o 6f
```

```
a[ 16]: b 62
a[ 17]: e 65
a[ 18]:
 a
a[ 19]:  0
```

あらたな文字列は、
Kyoto（OSAKA）Kobe です。
一文字ずつみていくと、つぎのようになります。

```
a[  0]: K 4b
a[  1]: y 79
a[  2]: o 6f
a[  3]: t 74
a[  4]: o 6f
a[  5]:   20
a[  6]: ( 28
a[  7]: O 4f
a[  8]: S 53
a[  9]: A 41
a[ 10]: K 4b
a[ 11]: A 41
a[ 12]: ) 29
a[ 13]:   20
a[ 14]: K 4b
a[ 15]: o 6f
a[ 16]: b 62
a[ 17]: e 65
a[ 18]:  0
```

実行結果からもわかるように、fgets で文字列を一行入力したときは、文字列の末端「\0」のまえに改行コード「\n」が来ていることに注意する必要がある。これを取り除くために、

```
a[strlen(a) - 1] = '\0';
```

としているのである [6]。

参考文献

[1] [7] の第 5 章。

[2] [7] の第 6 章。

[3] J. May & J. Whittle (武舎 広幸　訳)『Symantec C++ for the Macintosh トレーニングブック (Symantec C++ for the Macintosh)』（翔泳社、1994 年)、第 8 章。

[4] [8] の § 1.8。

[5] [7] の p 76。

[6] [8] の pp 150-1。

[7] B. R. Kernighan & D. M. Ritchie (石田 晴久　訳)『プログラミング言語C･第2版 (The C programming Language)』(共立出版、1989 年)。

[8] L. Ammeraal (小山 裕徳 訳) 『C で学ぶデータ構造とプログラム (Programs and Data Structures in C)』(オーム社、1995 年)。