プログラムの書き方の注意

だんだん大きなプログラムを作成するようになってきたので、自分にも他人にも分かり易く書くことが重要になる。特に注意することとして次の 2 つの事項がある。

- **コメント**
  コメントは、プログラムを作っている最中はその必要性をあまり感じないが、一度できたものをあとで読もうとすると、たとえ自分が作ったものであってもコメントがないと苦労することが多々ある。ましてや、レポートの課題などで他人に読ませるものにはコメントが必須となる。コメントの付け方に規則はない。授業のプリントや本を参考にして読む人の立場に立ったコメントを書いて下さい。

- **字下げ**
  字下げ (indent, indentation) もあくまでも人間がプログラムを理解する助けとして行うもので、規則はない。たとえば、

```
      for(i=1; i<=10; i++) {
  printf("%d", i);
   j=i;
                            }
```

  や、

```
      for(i=1; i<=10; i++) {
      printf("%d", i);
      j=i;
      }
```

  などと書くよりも、

```
      for(i=1; i<=10; i++) {
         printf("%d", i);
         j=i;
      }
```

  とした方が、一目で for 文のかたまり (ブロック) が分かる。C 言語は、関数の集まりで構成され、各関数内では文ブロックの集まりで出来ているから、字下げをしてそのブロックの範囲を明示することはプログラムを読む大きな助けになる。

#define マクロについて

配列の大きさなど、問題のサイズによって変わるところに具体的な数を書いてしまうと、変更が大変である。プログラムの保守の点から、変更する箇所は少なければ少ないほどよいし、変更する箇所がプログラムの途中にあると読み直すのが大変である。

C 言語では、このような場合に便利な記法として #define マクロ [1]というものがある。

#define マクロは、

[1]マクロ名で示される文字列を他の文字列で置き換える機能をマクロ機能という。

`#define` 記号定数 文字列

のように記述する。C 言語プログラム中の記号定数は、一般の変数と区別するため、大文字で書く習慣がある。以下の例で分かるように、#define マクロは、プログラム本体中に特定の値を埋め込まないで処理するのに非常に役に立つ。

12. 配列とポインタ

配列とポインタは密接な関係がある。具体例で見ていこう。

[例題 1]　0 が来るまで配列に数を読み込んでいき、0 が来た時点でそれまでの入力を出力するプログラム例。(配列の最後の要素には 0 が入るようにする)

まず、前期の知識を使って配列の添字を変えていくことで処理する方法。

```
#include <stdio.h>
#define ARRAYSIZE 100

main()
{
  int a[ARRAYSIZE], i;

  scanf("%d", &a[0]);
  for(i=0; a[i]!=0; ) {                 /* 0 が入力されたら終了 */
    i++;
    scanf("%d", &a[i]);
  }

  for(i=0; a[i]!=0; i++)
    printf("%8d", a[i]);
}
```

次に、配列名が領域の先頭番地を表すということと、`a[i]` ≡ `*(a+i)` となることを使って書き換える。

```
#include <stdio.h>
#define ARRAYSIZE 100

main()
{
  int a[ARRAYSIZE], i;

  scanf("%d", a);                       /* 最初の数の入力 */
  for(i=0; *(a+i)!=0; ) {               /* 0 が入力されたら終了 */
    i++;
    scanf("%d", (a+i));
  }
```

```
  for(i=0; *(a+i)!=0; i++)
    printf("%8d", *(a+i));
}
```

次に、ポインタ変数を使ってすっきりと書き換える。

```
#include <stdio.h>
#define ARRAYSIZE 100

main()
{
  int a[ARRAYSIZE], *p;

  scanf("%d", a);
  for(p=a; *p!=0; ) {      /* a は配列領域の先頭番地なので，p=a といった代入が可能 */
    p++;                   /* ポインタを 1 つ進めて次の要素を指すようにする */
    scanf("%d", p);        /* p は番地なので，&はつけない */
  }

  for(p=a; *p!=0; p++)
    printf("%8d", *p);
}
```

[練習 1]　上のプログラムで配列要素の総和を求めるようにしなさい。

[練習 2]　上のプログラムで配列要素の最大値を求めるようにしなさい。

[例題 2]　配列要素の最後に 0 が入っているような配列を受けとって、全要素の和を返す関数 arraysum を作りなさい。

```
#include <stdio.h>
int arraysum(int *);

int main(void)
{
  int a[]={1,2,3,4,5,6,0};

  printf("%d", arraysum(a));
}

int arraysum(int b[])
{
  int sum=0, i;

  for(i=0; b[i]!=0; i++)
    sum = sum + b[i];

  return sum;
}
```

関数 arraysum をポインタ表示で書くと以下のようになる。

```
int arraysum(int b[])
{
  int sum=0, i;

  for(i=0; *(b+i)!=0; i++)
    sum = sum + *(b+i);

  return sum;
}
```

または

```
int arraysum(int *b)
{
  int sum=0;

  for( ; *b != 0; b++)
    sum = sum + *b;

  return sum;
}
```

[練習 3]　配列要素の最後に 0 が入っているような配列を受けとって、要素の最大値を返す関数 arraymax、全要素の平均値を返す関数 arrayave を作りなさい。

[練習 4]　整数要素の配列を受取り、全要素の平均値に一番近い要素を求める関数を作りなさい。(ヒント: 以下のプログラムが参考になるだろう)

```
 1  /* ctrl-d が入力されるまで整数を標準入力から読み込み，入力数のうちで平均値に一番近い数を出力するプログラム 複数の数が同じ距離で平均値に近い時はどれか一つ出力する */
 2
 3  #include <stdio.h>
 4  #define MAXARRAY 100          /* 配列のサイズ */
 5
 6  main()
 7  {
 8    int closest, i, n, sum;
 9    int a[MAXARRAY];
10    float avg = 0.0;
11
12    /* ctrl-d が入力されるまで入力し，配列に格納 */
13    for (n = 0; n < MAXARRAY && scanf("%d", &a[n]) != EOF; n++) ;
14
15    /* 平均値の計算 */
16    sum = 0;
17    for (i = 0; i < n; i++)
18      sum = sum + a[i];
19    avg = (float)sum / n;
20
21    /* 平均に一番近い要素を求める．同じ距離の場合にはインデックスの小さい方 */
22    closest = 0;           /* インデックスを記憶する */
23    i = 1;
24    while (i < n) {
25      if ((a[i]-avg)*(a[i]-avg) < (a[closest]-avg)*(a[closest]-avg))
26        closest = i;
27      i++;
28    }
29
30    printf("%d が平均に一番近い要素 (平均値は%f)\n", a[closest], avg);
31  }
```