計算機言語 I 第 7 回
for 文を用いたループ (繰り返し)

この講義資料は, 次の場所にあります.

`http://www.math.u-ryukyu.ac.jp/~suga/gengo/2018-1/07.pdf`

1 ご託宣

前回に続いて, ループの記述法を述べます. 今回は for 文で, awk でも同じことを学びました.

ループを書く際に, どの文を用いるかのを決める一つの基準を前回述べましたが, それを再掲しておきます (あくまでも一つの基準で, これが万能というわけではない).

- for 文を用いたほうがわかりやすい場合.
  - ループに入る前に終了条件 (あるいは, 実行する回数) がわかっている.
  - 繰り返しの次の回の実行内容が, 前の回の実行内容とあまり変化しない.
- do – while 文, while 文を用いたほうがわかりやすい場合.
  - ループの終了条件が, 実行中に決まる.(事前に実行回数が決まらない.)
  - 繰り返しの次の回の実行内容が, 前の回から大きく変わることがある.

2 for 文

次のように記述します.

```
for ( 処理の初期条件の設定 ; 終了条件 ; 次のループに入る際の設定 ) {
    処理 1:
    処理 2:
    ...
}
```

for 文が多く利用される典型的な繰り返しは,「あらかじめ定まった回数だけ特定の動作を繰り返す.」という形で, 繰り返しの回数を記録する変数 (カウンタ (counter) 変数) を用意して, 次のように書く場合です. 次の例では, i がカウンタ変数です.

```
int i ;
for ( i=1  ; i<= 繰り返し回数  ; i++ ) {
    処理 1:
    処理 2:
    ...
}
```

for 文と同値な内容の while 文は, 次になります.

```
処理の初期条件の設定 ;
while (終了条件){
    処理 1;
    処理 2;
    ...
    次のループに入る時の設定;
}
```

for 文の「初期条件の設定」,「終了条件」,「次のループに入る時の設定」は全て省略できます.
例えば,

```
for ( ; ; ){
    処理 1;
    処理 2;
    ...
}
```

は, ブロック内の処理を永遠に続ける「無限ループ」になります.

2.1 無限ループ

今では, 多くのアプリケーションは, 無限ループで動いています. Window システムでのアプリケーションは, C 言語で記述すると次のようなソースコードになっています.

次で, `while (1)` というのは, while 文を用いた無限ループの作り方で, アプリケーションの終了は, break 文を利用して, 無限ループから抜け出します.

```
main()
{
    アプリケーションの初期設定;

    while (1){
        マウスやキーボードの入力を取得する;

        if (終了をする指示が選ばれた)
            break;

        マウスやキーボード入力に対応する処理;
    }

    終了時の処理;
}
```

3 ループを書く

前回も述べましたが, プログラミング言語を学ぶ際には,

- 文法を理解して, 他の人が書いたプログラムの動作が理解できるようになる.
- 要求された処理や計算を実行するプログラムを書けるようになる.

の 2 つの目標があります. これらのうち,「プログラムを書く」の部分では, 「できる限り明解な」プログラムを書くべきです. 教科書の while 文の例は, while 文の文法の説明には良いかもしれませんが, ループの記述としては, for 文の方が明解になります.

ここでは, それらについて述べていきます. 例によって, ブロックが不要な場合でも, ブロックを用いたプログラムを記述します.

3.1 whlile を for に書き換える.

List 4-5(p. 81)

このプログラムは, 読み込まれた数から 0 までの出力して最後に改行の出力ですから, for 文を用いて次のようにする方が明解です. 入力値を保持する変数名は, num(number の略) に変更しました (no (nombre の略?) という変数名もセンスが悪い).

```
#include <stdio.h>

int main(void)
{
    int num;

    printf("正整数を入力して下さい: ");
    scanf("%d", &num);

    for ( ; num >=0 ; num--){
        printf("%d ", num);
    }
    printf("\n");

    return 0;
}
```

List 4-6 (p. 82), List 4-8 (p. 84) **の問題点**

これらでは, デクリメントを利用してプログラムを短くしています. このようなプログラムを読む能力は身につける方が良いですが, このようなプログラムを書いてはいけません.

何がダメかというと. List 4-6 の出力部分の

```
printf("%d ", no--);
```

の部分です. no-- ですが, 変数 no がデクリメント (1 減ら) されます. ところで, そのデクリメントは, どのタイミングで実行されるのでしょう?

後置デクリメント (後置インクリメント) では, その変数の値にアクセスしたあとデクリメント (インクリメント) が実行されると, 文法で定まっています. したがって, 上の printf 文では, no の値を印刷したあと, no の値が減らされます. つまり, 上の 1 つの文は, 次の 2 つの文と同等です.

```
printf("%d ", no);
no--;
```

このことを知っていなければ, 上のプログラムが正しく動作するかの判断ができません. 現在では,「このようなプログラムの記述はすべきではない.」という認識が普通になっています. つまり,「1 つの実行文に 2 つ以上のことをさせるな.」です. 2 つのことを実行すると, 実行順によって結果が異なるのはよくあることです (ズボンとパンツの関係). 1 つの実行文に 2 つ以上のことを書くのは, それらの実行順に結果が依存しない場合に限るべきです (スカートとパンツの関係).

ちなみに, 前置のデクリメントを使った,

```
printf("%d ", --no);
```

は, 次の 2 つの文と同じになります.

```
--no;
printf("%d ", no);
```

同じことは, p. 84, List 4-8 でも言えます.

```
whilie (no-- >= 0)
```

の部分です. このプログラムの動作は理解できるようになるべきですが, このようなプログラムを自分から書いてはいけません.

List 4-7 (p. 83), List 4-9 (p.86)

それぞれ for 文を用いたプログラムが, p. 90, List 4-11, p. 93, List 4-13 で例示されています.

3.2 do – while として良いと思える例

List 4-10 (p.88)

このプログラムは, do – while ループとして, 適切な使い方の例になっています. そのアルゴリズムも含めて, よく理解して下さい.

3.3 多重ループ

ループも入れ子構造を用いて多重化できます. p. 96, List 4-16 ですが, ブロックを用いていないため, 読みやすさが落ちています. 次のようにブロック化しておけば, ループの範囲が明確になります.

```
#include <stdio.h>

int main(void)
{
   int i, j;

   for ( i=1 ; i <= 9 ; i++){
      for ( j=1 ; j <= 9 ; j++){
         printf("%3d", i*j);
      }
      putchar('\n');
   }

   return 0;
}
```

putchar と文字定数

C の基本変数型に char 型というのがあるというのを, 以前に述べました. 今利用している C の処理系では, これは, 1 つの Ascii 文字を保持するための変数です.

putchar() は, printf(), puts(), scanf() と同様, ライブラリ関数です. その動作は, (Ascii 文字の) 1 文字を標準出力に出力します (put character の略, character は文字の意味). つまり引数に char 型の値を受け取る関数で, その文字コードに対応する文字を出力します.

C でも文字定数や文字列定数は次のように記述します.

文字列定数 2 重引用符で囲まれる. "ABC" のような形.

文字定数 1 文字単位で, 1 重引用符で囲まれる. 'A' のような形. 基本的に Ascii コードの文字. ほとんどの処理系では, 'A' は Ascii コードの対応する数の意味です.

したがって, "A"は 1 文字からなる文字列で, 文字定数ではありません. 文字列と 1 文字は明確に区別がありますが, それについては, 後の講義で解説します. 'AB' はエラーになります.

レポート問題

1. p. 95, 演習 4-19, 件名: enshu 4-19.
2. p. 100, 演習 4-20, 件名: enshu 4-20.
3. p. 101, 演習 4-21, 件名: enshu 4-21.

締め切りは, 6 月 4 日.