# 基本制御構造 (繰り返し)

山本昌志*

2004 年 9 月 10 日

## 1　先週の復習と今週の内容

### 1.1　先週の復習

順次と選択、繰り返しがプログラムの基本制御構造である。このうち、順次と選択について、先週、述べた。順次は、文を並べ、それが上から下へと実行される、いつものお決まりのパターンである。選択は少し複雑で、次の 5 つの構文を学習した。

- もし、制御文が正しければ、文を実行する

```
if(制御式) 文;
```

- もし、制御文が正しければ、文 1→ 文 2→ 文 3 を実行する

```
if(制御式){
   文 1;
   文 2;
   文 3;
}
```

- もし、制御文が正しければ、文 1→ 文 2→ 文 3 を実行する。さもなければ (制御文が誤り)、文 4→ 文 5→ 文 6 を実行する。

```
if(制御式){
   文 1;
   文 2;
   文 3;
}else{
   文 4;
   文 5;
   文 6;
}
```

*国立秋田工業高等専門学校　電気情報工学科

- もし、制御文 1 が正しければ、文 1→ 文 2 を実行する。さもなければ、制御文 2 が正しければ、文 3→ 文 4 を実行する。さもなければ、制御文 3 が正しければ、文 5→ 文 6 を実行する。さもなければ (全てが誤り)、文 7→ 文 8 を実行する。

```
if(制御式 1){
   文 1;
   文 2;
}else if(制御式 2){
   文 3;
   文 4;
}else if(制御式 3){
   文 5;
   文 6;
}else{
   文 7;
   文 8;
}
```

- もし、式の値が 1 ならば文 1→ 文 2 を実行し、式の値が 2 ならば文 3→ 文 4 を実行し、式の値が 5 ならば文 5→ 文 6 を実行し、式の値がいずれでもないときは文 7→ 文 8 を実行する。

```
switch(式){
  case 1:
    文 1;
    文 2;
    break;
  case 2:
    文 3;
    文 4;
    break;
  case 5:
    文 5;
    文 6;
    break;
  default:
    文 7;
    文 8;
}
```

## 1.2 今週の学習内容

基本制御構造の残り、繰り返しについて学習する。繰り返しの構造は、図 1 の通りである。ここで学習する繰り返しの構文は、

| | |
|---|---|
| `for` | 前判定繰り返し。あらかじめ繰り返し回数が分かっているときに、使われることが多い。 |
| `while` | 前判定繰り返し。繰り返し回数が分からないときに、使われることが多い。 |
| `do while` | 後判定繰り返し。繰り返し回数が分からないときに、使われることが多い。 |

である。

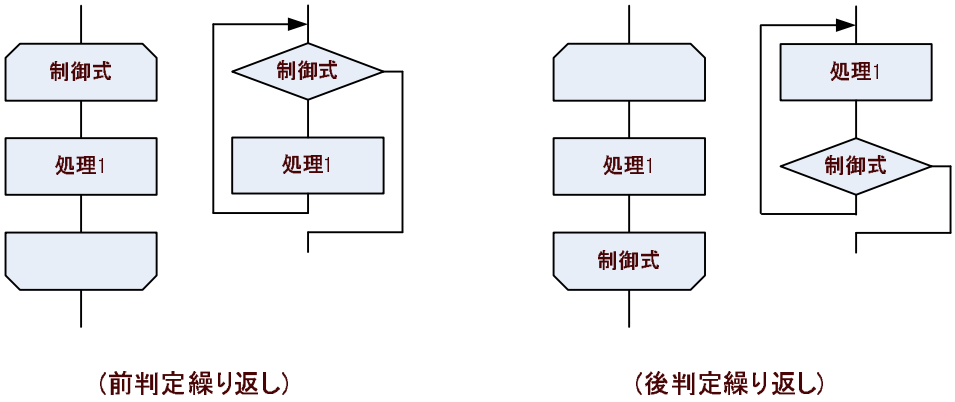


図 1: 繰り返しの構造

## 2　基本制御構造 (繰り返し)

繰り返しの構文に使われる C 言語の命令は、次の通りで、英語の意味を書いておく。

| | |
|---|---|
| for | ～の間 |
| do | する (C 言語では命令形と考える) |
| while | ～する間 |
| continue | 続ける |
| break | 遮断する |

### 2.1　`for` 文

繰り返しの回数が予め分かっているときに、しばしば使われる `for` 文を説明する。「初期値は○○、条件式が正しければ、ループを繰り返し、条件を再設定する。これは、条件式検査 → ループ → 条件の再設定を繰り返す。条件式が誤りになれば、そのループから抜け出す」という構文に使われる。これは、次のように、書く。

書式

```
for(初期値設定式; 継続条件式; 再設定式){
   文 1;
   文 2;
   文 3;
}
```

これは、「継続条件が正しい限り、文 1 と文 2、文 3 を実行する」となる。もし、制御式が誤り (偽) であれば、これら文は実行されず、ブロックの外側に出る。図 2 にこの構文のフローチャートを示す。

以下のようなプログラムが、この構文の使用例である。

```
for(a=1; a<=100; a++){
   printf("a=%d\n",a);
   sum=sum+a;
   printf("sum=%d\n",sum);
}
```

この構文の実行直前まで、sum=0 ならば、1〜100 までの和を計算することができる。見慣れない a++は、a=a+1 と同じで、a の値を 1 増加させている。これをインクリメントと言う。

構文の内容は、次の通りである。

- 初期値として、「a=1」と設定する。
- もし、a が 100 以下ならば、
  - 「a=値」と表示する。
  - 「sum+a を計算し、その結果を sum に代入」を実行する。
  - 「sum=値」と表示する。
  - 「a の値を+1 増加」を実行する。
- 一つ前の、アイテム「もし、a が 100 以下ならば ···」に戻る。

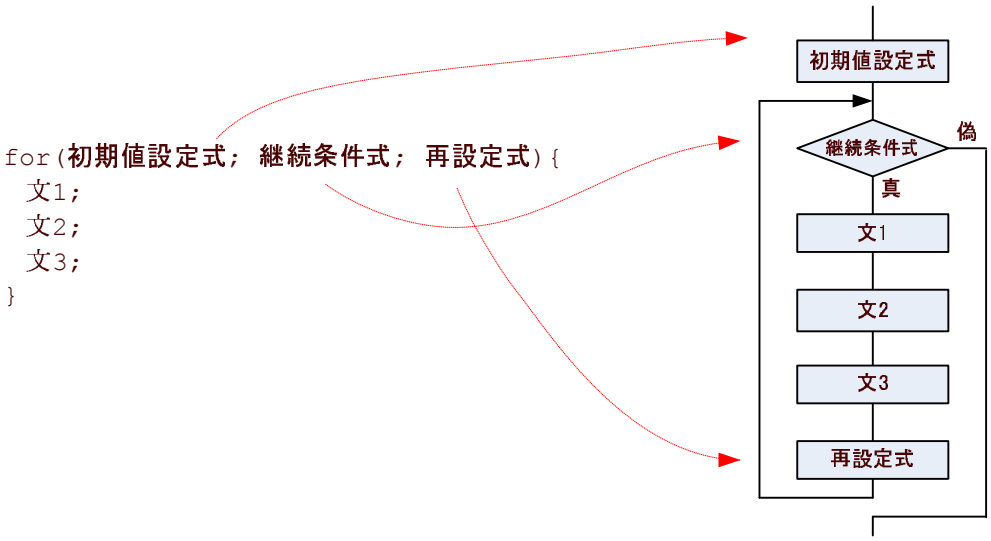


図 2: for の前判定繰り返し文

## 2.2 while 文

これも、for 文同様、前判定繰り返しであるが、予め繰り返し回数が分からないときには、while 文が使われることが多い。「条件式が正しければ、ループ繰り返す。条件式が誤りになれば、そのループから抜け出す」という構文に使われる。次のように、書く。

書式

```
while(継続条件式){
   文 1;
   文 2;
   文 3;
}
```

これは、「継続条件が正しい限り、文 1 と文 2、文 3 を実行する」となる。もし、制御式が誤り (偽) であれば、これら文は実行されず、ブロックの外側に出る。図??にこの構文のフローチャートを示す。

以下のようなプログラムが、この構文の使用例である。

```
while(sum<=10000){
   sum=sum+n;
   printf("n=%d\n",n);
   n++;
}
```

この構文の実行直前まで、sum=0 かつ n=1 ならば、

$$\texttt{sum} = 1 + 2 + 3 + \cdots + n$$

を計算する。ただし、この繰り返し文を抜けたときには、sum の値は 10000 を越えている。

構文の内容は、次の通りである。

- もし、sum が 10000 以下ならば、
  - 「n=値」と表示する。
  - 「sum+n を計算し、その結果を sum に代入」を実行する。
  - 「n=値」と表示する。
  - 「n の値を+1 増加」を実行する。
- 一つ前の、アイテム「もし、sum が 10000 以下ならば ···」に戻る。

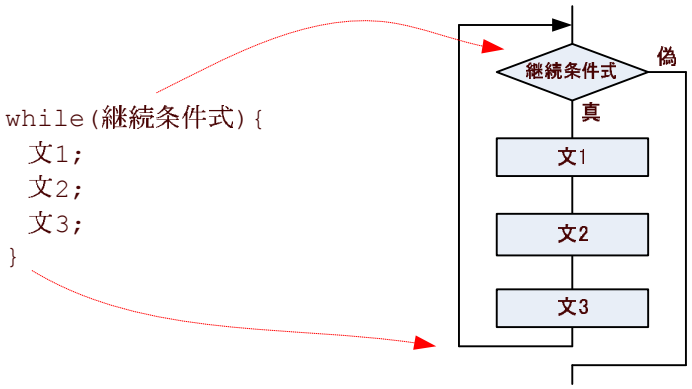


図 3: while の前判定繰り返し文

## 2.3 do while 文

これも、後判定繰り返しで、予め繰り返し回数が分からないときに使われることが多い。「ループ内を実行し、継続条件式が正しければ、さらにループを繰り返す。条件式が誤りになれば、そのループから抜け出す」という構文に使われる。次のように、書く。

書式

```
do{
   文 1;
   文 2;
   文 3;
}while(継続条件式);
```

これは、「文 1 と文 2、文 3 を実行し、継続条件が正しければ、これを繰り返す」となる。もし、制御式が誤り (偽) であれば、ブロックの外側に出る。図 4 にこの構文のフローチャートを示す。

以下のようなプログラムが、この構文の使用例である。

```
do{
   sum=sum+n;
   printf("n=%d\n",n);
   n++;
}while(sum<=10000);
```

この構文の実行直前まで、sum=0 かつ n=1 ならば、

$$\mathtt{sum} = 1 + 2 + 3 + \cdots + n$$

を計算する。ただし、この繰り返し文を抜けたときには、sum の値は 10000 を越えている。

構文の内容は、次の通りである。

- 以下を実行する。
  - 「n=値」と表示する。
  - 「sum+n を計算し、その結果を sum に代入」を実行する。
  - 「n=値」と表示する。
  - 「n の値を+1 増加」を実行する。
- もし、sum が 10000 以下ならば、一つ前の、アイテム「以下を実行する」に戻る。

do while 文と whil 文の動作はよく似ているが、「最初から継続条件式が誤り」の場合に違いが生じる。違いは、

| | |
|---|---|
| do while | 最初から条件式が誤りでも、ループブロックを 1 回は実行する。 |
| while | 最初から条件式が誤りの場合、ループブロックは実行されない。 |

である。

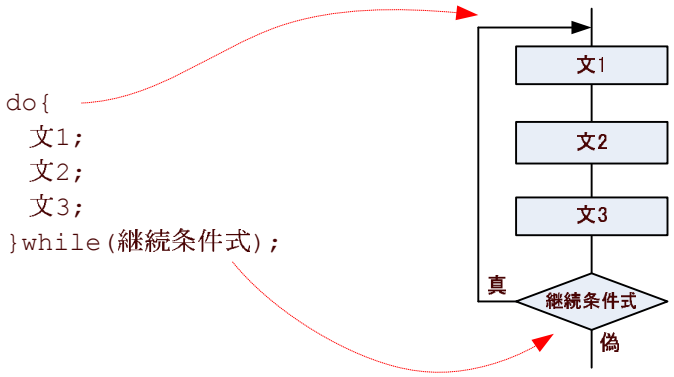


図 4: do while の後判定繰り替え詩文

# 3　ループのスキップと脱出

## 3.1　スキップ (continue)

場合によっては、ループブロックの文を実行させたくない場合がある。このとき、continue 文を使う。通常は if 文を伴って、次のように書く。

書式

```
while(継続条件式){
   文 1;
    if(制御式) continue;
   文 2;
   文 3;
}
```

これは、「継続条件が正しい限り、文 1 と文 2、文 3 を実行する。ただし、制御式が正しければ文 2 と文 3 はスキップする。」となる。当然、制御式が誤り (偽) であれば、これら文は実行されず、ブロックの外側に出る。図 5 にこの構文のフローチャートを示す。ここでは、while 文に、continue を用いているが、for や do while 文にも使える。いずれの構文でも、contine 文に出会うと、それ以降のループブロックが実行されない。

以下のようなプログラムが、この構文の使用例である。

```
while(sum<=10000){
   sum=sum+n;
   n++;
   if(sum <= 9000)contine;
   printf("sum=%d\n",n);
}
```

この構文の実行直前まで、sum=0 かつ n=1 ならば、

$$\mathrm{sum} = 1 + 2 + 3 + \cdots + n$$

を計算する。ただし、この繰り返し文を抜けたときには、sum の値は 10000 を越えている。

構文の内容は、次の通りである。

- もし、sum が 10000 以下ならば、
  - 「sum+n を計算し、その結果を sum に代入」を実行する。
  - 「n の値を+1 増加」を実行する。
  - もし、sum が 9000 以下ならば、ループブロックの最後にスキップする。
  - 「sum=値」と表示する。
- 一つ前の、アイテム「もし、sum が 10000 以下ならば ···」に戻る。

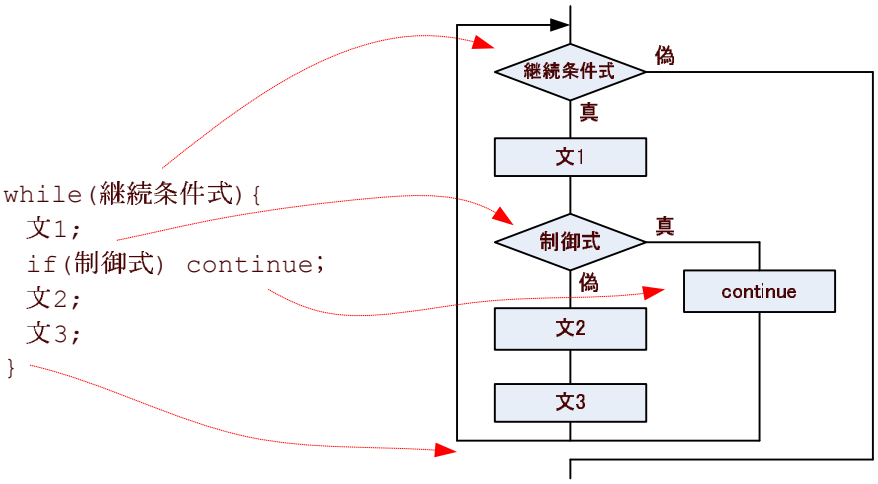


図 5: contine を使って、ループブロックをスキップする構文

## 3.2 脱出 (break)

場合によっては、継続条件式が正しくても、構文から抜け出たい場合がある。このとき、break 文を使う。通常は if 文を伴って、次のように書く。

書式

```
while(継続条件式){
   文 1;
    if(制御式) break;
   文 2;
   文 3;
}
```

これは、「継続条件が正しい限り、文 1 と文 2、文 3 を実行する。ただし、制御式が正しければ、この構文から抜ける」となる。当然、制御式が誤り (偽) であれば、これら文は実行されず、ブロックの外側に出る。図 6 にこの構文のフローチャートを示す。ここでも、while 文に、break 文を用いているが、for や do while 文にも使える。いずれの構文でも、break 文に出会うと、構文から抜け出る。。

以下のようなプログラムが、この構文の使用例である。

```
while(1){
   sum=sum+n;
   n++;
   if(sum >=  10000)break;
   printf("sum=%d\n",n);
}
```

この構文の実行直前まで、sum=0 かつ n=1 ならば、

$$\mathtt{sum} = 1 + 2 + 3 + \cdots + n$$

を計算する。ただし、break により、sum の値が 10000 以上になると、この構文から完全に抜け出す。。

構文の内容は、次の通りである。

- 継続条件式はいつも 1(真) なので、以下を実行する。
  - 「sum+n を計算し、その結果を sum に代入」を実行する。
  - 「n の値を+1 増加」を実行する。
  - もし、sum が 10000 以上ならば、構文から抜ける。
  - 「sum=値」と表示する。
- 一つ前の、アイテム「継続条件式はいつも 1(真) なので、$\cdots$」に戻る。

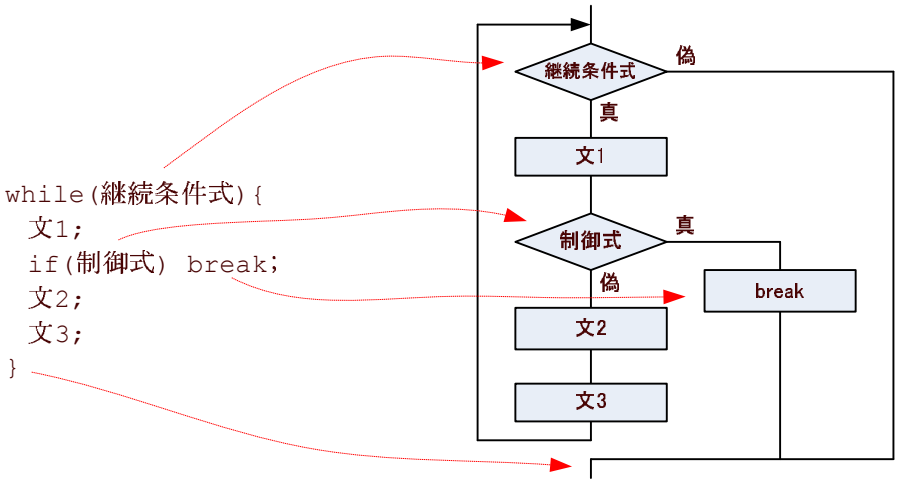


図 6: break 文を用いた構文からの脱出

# 4 練習問題

次の練習問題のプログラムを作成せよ。コンパイル可能なソースプログラムを書くこと。#include <stdio.h>も int main() も全て記述せよと言うことである。

試験も近いので、これは課題にしない。

## 4.1 for

1. キーボード入力と計算結果出力

- n の値をキーボードから読み込む。
- for 文を用いて、

$$\texttt{sum} = 1 + 2 + 3 + 4 + \cdots + n$$

  を計算する 。
- 計算結果を「sum=値」と表示する。

## 4.2 while

1. キーボード入力と計算結果出力
   - n の値をキーボードから読み込む。
   - while 文を用いて、

$$\texttt{sum} = 1 + 2 + 3 + 4 + \cdots + n$$

     を計算する 。
   - 計算結果を「sum=値」と表示する。

## 4.3 do while

1. キーボード入力と計算結果出力
   - n の値をキーボードから読み込む。
   - do while 文を用いて、

$$\texttt{sum} = 1 + 2 + 3 + 4 + \cdots + n$$

     を計算する 。
   - 計算結果を「sum=値」と表示する。