5　制御文

5.1　構造化プログラミング

構造化プログラミング (structured programming) は 1960 年代末にダイクストラ (E.W.Dijkstra) によって段階的詳細化 (stepwise refinement) と併せて提唱されたプログラミング方法論 (programming methodology) である。「プログラムのテキストの構造は、計算の構造を反映することが望ましい」という理念に基づいている。プログラムを構造化する目的は、プログラムを理解するための知的な労力がプログラムの長さに比例するようにすることである。

構造化プログラミングでは、プログラムの制御構造は順次、選択、繰り返し (反復) を組み合わせて作る。

順次 (sequence)1 個以上の文を順に実行する。

選択 (selectin) 条件により実行する文を選択する。

単純選択 P が真ならば S を実行する。

二分岐選択 P が真ならば S_1 を実行し、そうでなければ S_2 を実行する。

多分岐選択 P_1 が真ならば S_1 を実行し、P_2 が真ならば S_2 を実行し、...。

繰り返し (repetition)

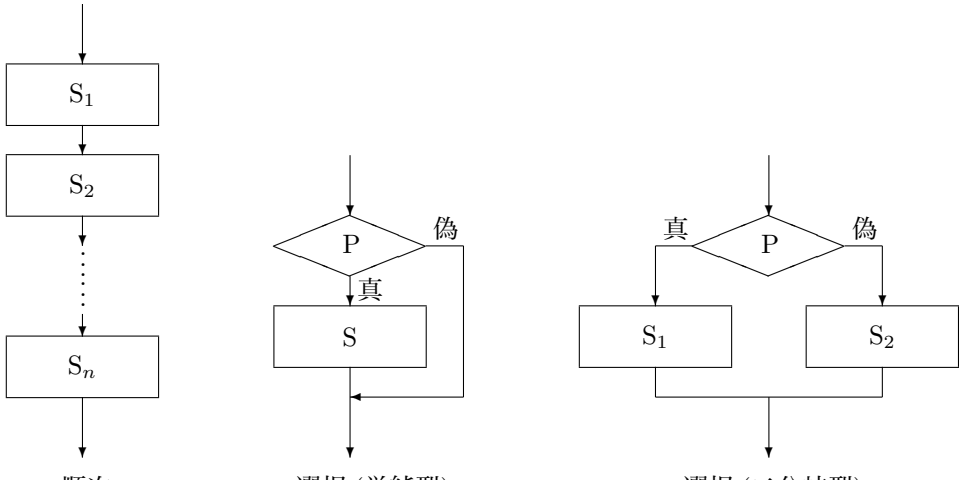
前判定反復 P が偽になるまで S を繰り返す (=P が真の間 S を繰り返す)。P の評価は S の実行の前に行う。P が最初から偽であれば S は 1 回も実行しない。

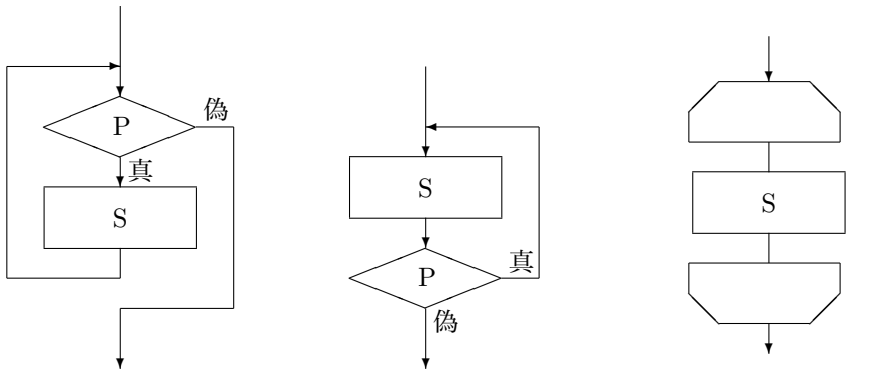
後判定反復 P が偽になるまで S を繰り返す。P の評価は S の実行の後に行う。S は少なくとも 1 回は実行される。

所定回反復 一定回数 S を繰り返す

繰り返し実行される S は、C ではループ本体 (loop body) という。

フローチャート (flowchart 流れ図) を用いて表わすと次のようになる。長方形の箱は処理を表わし、菱形は分岐を表わす。





前判定反復　　後判定反復　　所定回反復

`while (P) S`　　`do S while (P);`　　`for(E1;E2;E3) S`

上記の図はいずれも入り口が 1 つで出口も 1 つであるので、単一の作用と考えることができ、S や S_i を上記の図で置き換えることにより、選択や繰り返しを入れ子にすることもできる。

C では、P は式で与えられる。S、S_i は文 (選択文や繰り返し文を含む) または複合文 (2 つ以上の文を中括弧 {} でくくり 1 つの文のように扱うもの。) である。

演習 5.1 [基本情報処理技術者試験 2003 年度春期午前]
構造化プログラミングにおいて、プログラムを作成するときに用いる三つの制御構造はどれか。

ア 繰返し、再帰、順次　　イ 繰返し、再帰、選択
ウ 繰返し、順次、選択　　エ 再帰、順次、選択

5.2 選択文

5.2.1 if 文

`if` 文の構文規則は、

```
if (式) 文
```

である。

式の値が 0 と等しくないとき (真のとき) 文が実行される。式の値が 0 と等しいとき (偽のとき) は何も実行されない。文は単一の文であっても複合文あっても良い。文が単一の文のときは文の最後にセミコロン; が必要である。

式には、定数、変数、算術演算子で表される式、論理演算子、関係演算子や等価演算子で表される式、等がある。関係演算子や等価演算子で表される式は真のとき値は 1 で、偽のとき値は 0 である。定数、変数、算術演算子で表される式は値が 0 でないとき真、0 のとき偽と約束する。

演習 5.2 次の printf 文 は実行されるか。
(1) `if (2+3) printf("真です。\n");`
(2) `if (2.6-1.2) printf("真です。\n");`
(3) `if (2-3+1) printf("真です。\n");`
(4) `if (2>3) printf("真です。\n");`
(5) `if (2<3) printf("真です。\n");`
(6) `if (0) printf("真です。\n");`

例 5.1 次のプログラムは、キーボードから整数を入力するとその数が奇数か偶数かを表示する。偶奇の判定は剰余演算子%を用い 2 で割りきれるか否かを調べている。num%2==0 は num%2 が 0 のとき、すなわち num が偶数のときは、0==0 となるので、1 が返される。num が正の奇数のときは 1==0 となり、num が負の奇数のときは-1==0 となるので、num が奇数のときは 0 が返される。

```
/* even-odd.c */
#include <stdio.h>

int main(void)
{
  int num;
  printf("整数を入力してください:  ");
  scanf("%d", &num);

  if (num%2==0) printf("%d は偶数です。\n",num);
  if (num%2!=0) printf("%d は奇数です。\n",num);

  return 0;
}
```

even-odd.c の if 文は、

```
if (!(num%2)) printf("%d は偶数です。\n",num);
if (num%2) printf("%d は奇数です。\n",num);
```

のように表わすこともできる。

演習 5.3 整数を入力し、正か 0 か負かを表示するプログラムを作れ。

演習 5.4 2 つの int 型整数 a,b を入力し、b が 0 でなければ a/b と a%b を表示するプログラムを作れ。

5.2.2 if-else 文

if-else 文の構文規則は

```
if(式) 文 1 else 文 2
```

である。式の値が 0 と等しくないとき (真のとき) 文 1 が実行され、式の値が 0 と等しいとき (偽のとき) は文 2 が実行される。文 1、文 2 が単一の文のときは文の後にセミコロンをつける。

例 5.2 even-odd.c の 2 つの if 文を if-else 文を用いて書き直す。

```
if (num%2==0) printf("%d は偶数です。\n",num);
else printf("%d は奇数です。\n",num);
```

演習 5.5 西暦の年が 4 で割り切れるときオリンピックが開催される。西暦の年を入力し、オリンピックが開催されるかされないかを表示するプログラム olympic.c を作れ。
(year が 4 で割り切れるは year%4 == 0 と表現できる。)

演習 5.6 2 数とその和を入力し、正しければ正解と表示し、正しくなければ、正しい答えを表示するプログラム dri$\ell\ell$1.c を作れ。

`if` 文と `if-else` 文をまとめて `if` 文 (if statement) と呼ぶ。`if` の直後の () で囲まれた式を制御式 (controlling expression) という。

`if` 文の (式) および `else` の後の文は副文 (substatement) と呼ばれる。副文が `if` 文であってもよい。`else` はその `else` の前で最も近い位置にある `if` と結び付く。

演習 5.7 整数を入力し、正 (0 より大きい) ならば、「正です。」、負 (0 より小さい) ならば、「負です。」、0 に等しければ「零です。」と表示するプログラム sign.c を作れ。

5.2.3 switch 文

多分岐選択文として switch 文がある。

```
switch (式) 文
```

式は制御式 (controlling expression) と呼ばれ、整数型をとる。文はスイッチ本体 (switch body) と呼ばれる。

```
switch(式){
  case 定数式:
    文
    break;
  case 定数式:
    文
    break;
  .
  .
  default:
    文
    break;
}
```

定数式は整数定数である。スイッチ本体のなかに同じ値の定数式があってはならない。制御式の値と case の後の定数式が一致する文が実行される。一致する値が見つからなかったときは、default の文が実行される。一致する値が見つからず、default も無い場合は何も実行されない。

switch 文のなかで break 文を実行すると、switch 文が終了する。case 定数式の後の文の中で break 文を省略すると、次の case 定数式の文も実行される。switch 文のなかに break 文が一つもない場合は、対応する case 以下 default を含めすべての文が実行される。一致する値も default もないときは何も実行されない。

ラベル付き文

ラベル付き文 (labeled statement) には次の 3 通りがある。

```
識別子:文
```

```
case 定数式:文
```

```
default:文
```

「識別子」、「case」、「default」をラベル (label) という。「case 定数式:文」と「default:文」は、switch 文の中でのみ用いる。「識別子:文」は goto 文の分岐先として用いられることが多い。

例 5.3 次の例は switch 文の簡単な例である。

```
/* switch.c */
#include <stdio.h>

int main(void)
{
  int i;

  printf("1 から 3 までの整数を入力してください:");
  scanf("%d", &i);

  switch(i){
    case 1:
      printf("1 が入力されました。\n");
      break;
    case 2:
      printf("2 が入力されました。\n");
      break;
    case 3:
      printf("3 が入力されました。\n");
      break;
    default:
      printf("認識できない数字です。\n");
  }

  return 0;
}
```

演習 5.8 正の整数を入力し、5 で割りきれるときは「5 の倍数です」、5 で割って 1 余るときは「5 で割ると 1 余ります」、5 で割って 2 以上余るときは「5 で割ると 2 以上余ります」と出力するプログラムをつくれ。

5.3 繰り返し文

ループ本体 (loop body) と呼ばれる文を、制御式が 0 と等しくなるまで繰り返す。

5.3.1 for 文

for 文の構文規則[15]は次の通り。

```
for (式 1 ; 式 2 ; 式 3 ) 文
```

文が複合文のときはセミコロンは不要である。式 1 はループが始まる前に 1 回だけ実行される。式 2 は制御式で、値が 0 以外 (真) ならループを続け、値が 0(偽) ならループを終了して、ループの次の文を実行する。式 3 はループ本体を実行した後、制御式の評価 (次の条件判定) を行う前に実行される。

5.3.1.1 for 文 — 無限ループなど

for 文において、式 1、式 2、式 3 はそれぞれ省略しても良い。式 2(制御式) を省略したときは 0 でない定数によって置き換えられる。したがって、

```
for( ; ; )
```

は

```
for( ; 1 ; )
```

と同じで無限ループとなる。無限ループは通常ループ本体に分岐文を置き、一定の条件でループを抜け出す。

[15] C99 では、式 1 の代わりに (`for(int i=0; i<10; i++)` というような) 宣言も認められている。コンパイルの際に東女ではオプションに `-std=c99` をつける。

5.3.1.2 for 文 — 一定回繰り返し

文を N 回繰り返すときは、

```
for (i=0 ; i<N ; i++) 文
```

とする。i++は i の値を 1 増やす増分演算子 (4.7.2 節をみよ) である。i= $0, 1, \ldots, N-1$ に対し文が実行され、i= N となったとき、制御式の値が 0 となるので for 文を終了し、次の文に進む。

例 5.4 Hello World を 10 回出力するプログラム。

```
/* hello10.c */
#include <stdio.h>

int main(void)
{
  int i;

  for(i=0 ;i<10 ;i++){
    printf("Hello, World.\n");
  }

  return 0;
}
```

例 5.5 1 から 10 までの和を求めるプログラム。

```
/* summation.c */
#include <stdio.h>

int main(void)
{
  int i, sum;

  sum = 0;
  for (i=1; i<= 10; i++){
    sum += i;
  }
  printf("総和は %d です。\n",sum);

  return 0;
}
```

sum += i; は、sum の値に i を加えた値を左辺の sum に代入するという意味である。
i と sum は以下のように変化する。

i	sum
1	1
2	3
3	6
⋮	⋮
10	55
11	ループから抜ける

このことを見るには、summation.c の for 文を

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

とすればよい。

演習 5.9 $1^2+2^2+3^2+...+100^2$(1 から 100 までの 2 乗の和) を求めるプログラム sum_square.c を作れ。

演習 5.10 $1+3+5+...+99$(1 から 99 までの奇数の和) を求めるプログラム sum_odd.c を作れ。

演習 5.11 自然数 n を入力し、$n!=1\times2\times\ldots\times n$(1 から n までの積) を求めるプログラム factorial.c を作れ。

例 5.6 5 個の int 型数を入力し、和と平均値を出力するプログラムを作れ。

```
/****************************************
* average1.c                            *
* N 個の int 型数を入力し和と平均値を求める。  *
* キャスト演算子を用いている。              *
****************************************/
#include <stdio.h>
#define N 5

int main(void)
{
  int i,a,sum;   /* sum を int 型と宣言している */
  double mean;

  sum=0;
  for(i=1;i<= N;i++){
    printf("%d 番目のデータ=",i);
    scanf("%d",&a);
    sum += a;
  }
  mean = (double) sum/N;   /* sum を double 型にキャストしている */

  printf("和は%d\n",sum);   /* 変換指定子は d */
  printf("平均値は%f\n",mean);

  return 0;
}
```

`#define N 5` は、翻訳に先立ち N をすべて 5 で置き換えさせる前処理指令である。

前処理指令

他のソースファイルを取り込んだり、マクロに置き換える指令で、翻訳の前に行われるので前処理 (preprocessing) と呼ばれる。

1. ソースファイル取り込み

   `#include` <ヘッダ名> 改行

   は、ヘッダの内容全体でこの指令を置き換える。

2. マクロに置き換え

   `#define` マクロ名 置換要素 改行

   `#define` 以降にマクロ名が出てきたら置換要素と置き換える。マクロ名の命名規則は変数名と同じであるが、変数名に小文字を用いることが多いのに対し、マクロ名には大文字を用いることが多い。

5.3.2 while 文

```
while(式) 文
```

式の値が 0 以外 (真) の間は文を繰り返す。(= 式の値が 0 になるまで文を繰り返す。) 式の値が最初から 0(偽) であれば文は 1 回も実行されない。

例 5.7 q が入力されるまで入力を繰り返す。

```
/* while.c */
#include <stdio.h>

int main(void)
{
  char c;
  scanf("%c",&c);

  while(c!='q')
    scanf("%c",&c);
  printf("q が入力されました \n");

  return 0;
}
```

for 文

```
for(式 1; 式 2; 式 3) 文
```

は、while 文を用いて次のように書き直すことができる。

```
式 1;
while(式 2){
    文
    式 3;
}
```

演習 5.12 2 つの整数とその和をキーボードから入力し、正解になるまで解答を要求するプログラム dri$\ell\ell$.c を作れ。

演習 5.13 $1+3+5+\cdots+99$(1 から 99 までの奇数の和) を求めるプログラムを while 文を用いて作れ。

5.3.3 do 文

```
do 文 while(式);
```

式の値が 0 以外 (真) の間は文を繰り返す。(= 式の値が 0 になるまで文を繰り返す。) 式の値が最初から 0(偽) であっても文は 1 回実行される。do 文は while 文と異なり最後にセミコロン; が必要である。

例 5.8 次のプログラムは、q が入力されるまで入力を繰り返す。getchar() はキーボードから 1 文字入力して、その文字コードを関数値として返す関数である。引数は取らない。

```
/* do-while.c */
#include <stdio.h>

int main(void)
{
  int c;

  do{
    c=getchar();
  }while(c!='q');

  printf("q が入力されました \n");

  return 0;
}
```

演習 5.14 正の整数 n を入力すると、$n\ n-1\ \ldots\ 1$ と出力するプログラム countdown.c を do 文を用いて作れ。

例 5.9 $1+2+3+4+\ldots+n>1000$ となる最小の自然数 n を求める。

```
/* summation2.c */
#include <stdio.h>

int main(void)
{
  int n=0,sum=0;

  do{
    n++;
    sum += n;
  }while(sum<1000);

  printf("1 から%d までの和は%d です。\n",n,sum);

  return 0;
}
```

演習 5.15 次の 2 題の中から 1 題を選び、プログラムを作り答えよ。

(1) 複利の利率年 1%の定期預金の元利合計が元本の 2 倍になるのは何年後か。

(Hint. 元本を a、単位期間当たりの利率を p とすると、n 回の単位期間を経て利子がついたときの元利合計は、複利の場合 $a(1+p)^n$ となる。)

(2) 定数 $e=2.718281828459045$ は次の無限級数の和

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots$$

で与えられる。e と $1+1/1!+\ldots+1/n!$ の差が $0.0000000001=10^{-10}$ より小さくなるような最小の自然数 n を求めよ。(計算は double 型で行うこと。$n!$ を一つのオブジェクトとして扱うとよい。)

5.4 分岐文

goto 文、continue 文、break 文、return 文を分岐文という。分岐文は、別の場所への無条件分岐を引き起こす。

5.4.1 goto 文

goto 文は、

```
goto 識別子;
```

の形をしている。識別子をラベルに付けた文に飛ぶ。

例 5.10 while.c を goto 文を使って書き直す。

```
/* goto.c */
#include <stdio.h>

int main(void)
{
  char c;
  while(1){
    scanf("%c",&c);
    if (c=='q') goto exit; /* goto 文 */
  }
  exit:printf("q が入力されました \n"); /* ラベル付き文*/

  return 0;
}
```

演習 5.16 break.c を goto 文を使って書き直せ。

5.4.2 continue 文

```
continue;
```

continue 文はループ本体の中におき、continue 文を実行するとすぐに次のループを実行する。

ループ	continue 文の次に実行
for	式 3、式 2(制御式)、ループの順
while	直ちに制御式
do	直ちに制御式

例 5.11 自然数 n を入力し、1 から n までの奇数のみの和を求めるプログラム。

```
1  /* continue.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6    int i,sum=0,n;
7
8    printf("1 から n までの奇数のみの和を求めます。n を入力してください。");
9    scanf("%d",&n);
10   for(i=1;i<=n;i++){
11     if (i%2==0) continue;
12     sum+=i;
13   }
14
15   printf("1 から
16   return 0;
17 }
```

11 行目の continue 文により、i が 0 で割り切れるとき (偶数のとき) は、sum+=i; を実行しない。

5.4.3 break 文

```
break;
```

break 文は、ループか switch 文 (p.43) でのみ使われる。break 文を実行すると、それを囲む最も内側のループまたは switch 文を終了させる。

例 5.12 例 5.8 を break 文を用いて書き直す。

```
1   /* break.c */
2   #include <stdio.h>
3
4   int main(void)
5   {
6     int c;
7
8     while(1){
9       c=getchar();
10      if (c=='q') break;
11    }
12
13    printf("q が入力されました \n");
14
15    return 0;
16  }
```

8 行目 `while(1)` は無限ループである。10 行目の break 文で脱出している。

5.5 条件演算子

簡単な if-else 文は代わりに条件演算子を使うことができる。

注 5.1 条件演算子は制御文ではない。

```
式１ ?  式２ :  式３
```

式 1 は論理式 (論理演算子から構成される式) である。最初に式 1 が評価され、0 以外 (真) ならば式 2 が評価されその結果が条件式全体の値になる。式 1 が 0(偽) ならば式 3 が評価されその結果が条件式全体の値になる。

例 5.13 ２つの整数を入力し小さくない方の値を求めるプログラム。

```
1   /* max.c */
2   #include <stdio.h>
3
4   int main(void)
5   {
6     int a,b,max;
7
8     printf("2 つの整数を入力してください。\n");
9     printf("a=");
10    scanf("%d", &a);
11    printf("b=");
12    scanf("%d", &b);
13    max = (a>b) ?  a :  b;
14    printf("小さくない方の値は %d です。\n",max);
15
16    return 0;
17  }
```

13 行目の右辺の値は、「`a>b` のときは `a` そうでなければ `b`」である。if-else 文を用いて書き直すと

```
if (a > b)
  max = a;
else
  max = b;
```

となる。

演習 5.17 3 つの整数を入力し、その最大数を求めるプログラムを条件演算子を用いて作成せよ。