制御の流れ

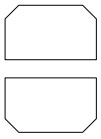
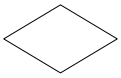
ここまでにはprintf、scanf、変数、四則演算など使ったプログラミングを習った。これらのプログラムの処理の流れは全て上から下と実行されており、これは「順次処理」と呼ばれている。また、四則演算では0で割り算を行わないようにif-else文を用いて処理を分岐させる方法についても習った。この処理を「条件判断処理」と言う。この他には「ループ処理」というものもある。これらの○○処理という呼び方は他にもいろいろある（条件分岐処理や繰り返し処理）。

プログラムの処理の流れを図で表現するものとして、フローチャートがある。本日はこのフローチャート（流れ図）を紹介し、条件判断処理の詳しい使い方を学習していく。

フローチャート

「フローチャート」はプログラムの流れを一目で分かるように図式化したものである。チャートは他にもNSチャート、PADなどがある。フローチャートの記号は日本工業規格（JIS）で定められている。表1にフローチャートで用いられる主な記号を示す。この他にも多くの記号がある。フローの最初と最後は「端子」を用いて明記し、処理の流れは原則として、「上から下」、「左から右」となる。逆行する場合や強調したい場合は矢印をつける。処理の流れを誤解されないように、線は交差させてはいけない。

表1　フローチャートで用いられる主な記号

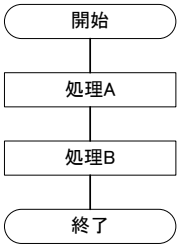
記　号	名称	説　　明	記　号	名称	説　　明
	端子	フローチャートの開始と終了を表す。		線	データや制御の流れを表す。流れの方向を明示したいときは矢印を付ける。
	処理	任意の処理（計算、代入など）を表す。記号中に処理内容を書く。		ループ端	ループの開始と終了を表す。記号のどちらかに終了条件を書く。また、必要に応じてループ名を書く。
	判断	一つの入口と複数の出口を持ち、記述された条件を判断して出口を選ぶ。			

処理の構造

フローチャートは基本的に３つの構造を組み合わせてアルゴリズムを記述する。

順次

処理が上から下に順番に並んでいる構造。



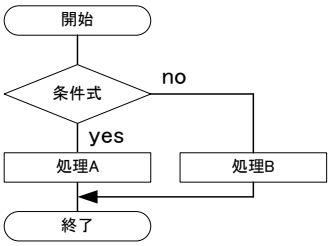

```
int main(void)
{
    syoriA( );    /*処理A*/
    syoriB( );    /*処理B*/

    return 0;
}
```

図1　順次

選択

条件によって処理が分岐する構造。条件は「yes（または true、真など）」か「no（または false、偽など）」で表す。分岐する形式によって「二分岐型」と「多分岐型」に分けることができる。C言語では「if文」、「switch

文」がこれにあたる。




```
int main(void)
{
    if(条件式)
    {
        syoriA( );    /*処理A*/
    }
    else
    {
        syoriB( );    /*処理B*/
    }

    return 0;
}
```

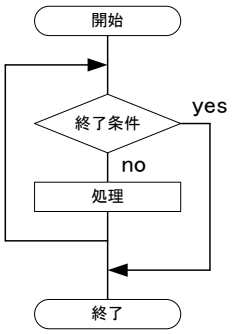
図２　選択

繰り返し

条件を満たしている間は処理を繰り返して実行する構造。C 言語では「while 文」、「for 文」、「do-while 文」がこれに当たる。条件を判定する位置によって「前判定型」と「後判定型」に分けることができる。

前判定型：処理を実行する前に条件の判定を行う。条件を満たさない場合には一度も処理を行わない場合もある。

後判定型：処理を実行してから条件の判定を行う。そのため、どのような条件でも一度は必ず処理を行う。



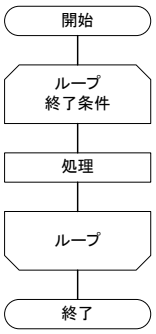
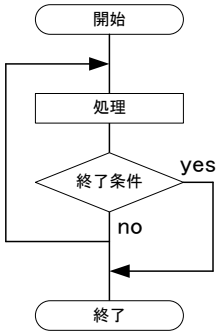


図３　前判定型



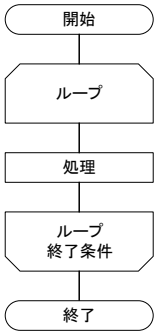


図４　後判定型

条件判断処理

if 文

それでは条件判断処理の具体的な処理を見ていこう。まずは if 文である。if 文は条件式が真だった場合に指定した処理を行う。偽の場合には指定した処理を行わず、次の処理に移る。

書式

if(条件式){
　　文;
}

使用例
#include <stdio.h> int main(void) { 　　int a;

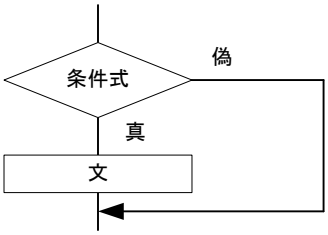


図５　if 文

注：　a *= -1; は a = a*(-1);と同じ処理である。

```
        printf("整数を入力してください：");
        scanf("%d", &a);

        if(a < 0){
                a *= -1;
        }

        printf("入力した値の絶対値は%dです¥n", a);

        return 0;
}
```

この省略した表記方法には、他には以下のものがある。

省略表記	同じ処理
a++;	a = a + 1;
a--;	a = a - 1;
a += b;	a = a + b;
a -= b;	a = a - b;
a *= b;	a = a * b;
a /= b;	a = a / b;
a %= b;	a = a % b;

{ }で囲まれた部分をブロックという。このブロック内の文が１つだけの場合には以下のように{ }を省略することができる。この例では条件式が真の場合だけ文１が実行される。文２については通常の順次処理が行われるので条件式の真偽に関わらず必ず実行される。しかし、慣れないうちは{ }を省略しないほうが無難である。また、どこが if 文であるのかわかりやすいように字下げ(インデント)を行う方が良い。

```
if(条件式)
        文1;
文2;
```

if-else 文

if 文は分岐の数によって多少書き方が変わる。ここでは条件式が真の場合と偽の場合とで異なる処理を行う if-else 文を紹介する。条件式が真のときには文１が処理され、偽のときに文２が処理される。つまり条件に従って文１か文２のどちらかを実行させたい場合に用いる。この場合も、ブロック { } 内の文が 1 つのみの場合は、{ } を省略することができる。

書式

```
if(条件式){
        文1;
}
else{
        文2;
}
```

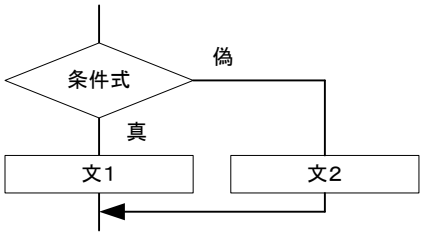


図６　if-else 文

使用例

```
#include <stdio.h>

int main(void)
{
        int a;

        printf("整数を入力してください：");
        scanf("%d", &a);

        if(a%2 == 0){
                printf("入力した値は偶数です¥n");
        }
```

```
        else{
                printf("入力した値は奇数です¥n");
        }

        return 0;
}
```

if-else if-else 文

複数の分岐を行いたい場合に用いる。以下の例では、条件式１が真のとき文１が実行される。条件式１が偽のときには条件式２を評価し、真のとき文２が実行される。このように次々と条件式を評価していき、すべての条件式が偽であった場合は最後の else 文のブロック内の文 m が実行される。else if 文はいくつでも設定でき、多分岐(多方向分岐)を行うことが可能である。また、最後の else 文が不要な場合は省略することができる。最後の else 文を省略すると、すべての条件式の評価が偽となった場合には、この構文で実行される文は無いことになる。

書式

```
if(条件式1){
        文1;
}
else if(条件式2){
        文2;
}
else if(条件式3){
        文3;
}
 :
 :
else if(条件式n){
        文n;
}
else{
        文m;
}
```

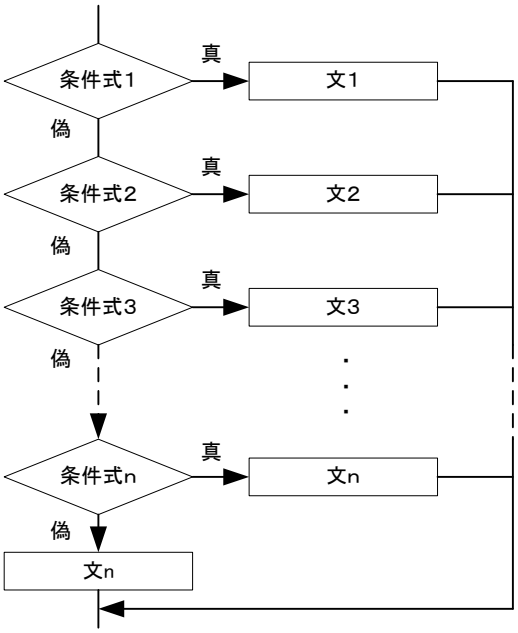


図７　if-else if-else 文

使用例

```
#include <stdio.h>

int main(void)
{
        int a;

        printf("整数を入力してください：");
        scanf("%d", &a);

        if(a < 0){
                printf("入力した値は負です¥n");
        }
        else if(a > 0){
                printf("入力した値は正です¥n");
        }
```

```
        else{
                printf("入力した値は0です¥n");
        }

        return 0;
}
```

switch 文

式と定数式との間で値が一致するか評価し、一致した定数式の case 部分にジャンプして対応する文を処理する。break 文が実行されると switch 文全体を終了する。また、どの case の定数式とも一致しない場合は default 部分に記述された文が実行される。default 部は不要なら省略できる。式には変数や計算式を使うことができる。定数式には整数、または文字を記述することができるが、実数や変数名、文字列は使えない。

書式

```
switch(式)
{
        case 定数式１;
           文１；
           break;
        case 定数式２;
           文２；
           break;
        case 定数式３;
           文３；
           break;
            :
            :
        default;
           文ｎ;
           break;
}
```

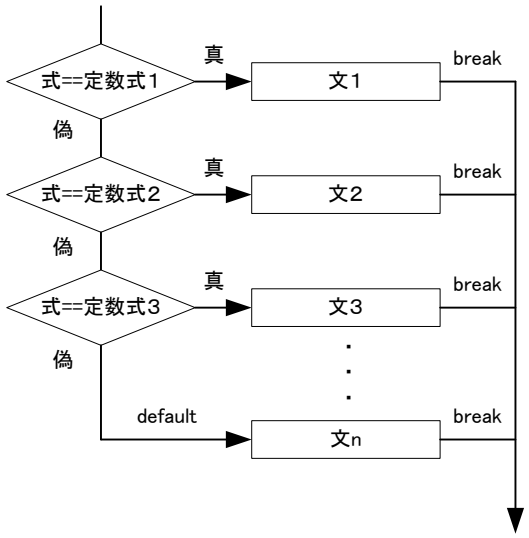


図８　switch 文

使用例

```
#include <stdio.h>

int main(void)
{
        int a;

        printf("1から3の範囲で値を入力してください：");
        scanf("%d", &a);

        switch(a)
        {
                case 1:
                        printf("入力した値は1です¥n");
                        break;

                case 2:
                        printf("入力した値は2です¥n");
                        break;
```

```
                case 3:
                        printf("入力した値は3です¥n");
                        break;

                default:
                        printf("範囲内の値を入力してください¥n");
        }

        return 0;
}
```

break 文は省略することができる。その場合の書式とフローチャートは以下のようになる。

書式

```
switch(式)
{
        case 定数式１;
           文１；
        case 定数式２;
           文２；
        case 定数式３;
           文３；
            :
            :
        default;
           文ｎ;
}
```

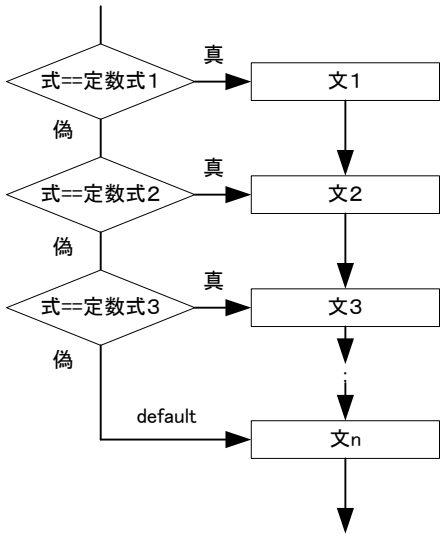


図９　break を省略した switch 文

演習

if 文、if-else 文、if-else if-else 文、switch 文の使用例で示したプログラムを作成し、出力を確認しなさい。また、switch 文の使用例において、break 文を全て削除した場合にどのような出力になるか確認しなさい。

課題２　次のプログラムを作成しなさい。

kadai2-1（20 点）

身長と体重を入力し、BMI を計算して表示するプログラムを作成しなさい。BMI とは Body Mass Index の略で肥満度の指標であり、身長と体重を用いて「BMI = 体重[kg]÷(身長[m]×身長[m])」として計算する。BMI は小数点以下一桁に調整して出力する。

実行結果例（斜体はキーボードから数値を入力する部分）

```
BMIを計算します。
身長(m)を入力してください。
1.84
体重(kg)を入力してください。
```

75.6

BMI=75.6kg÷(1.84m×1.84m)で計算します。

あなたの BMI は 22.3 です。25 を超えると注意してください。

次に if-else if-else 文を用いて、最後の出力結果の「25 を超えると注意してください。」の部分を条件分岐で以下のように表示を変える。

0 未満	入力に間違いがあります
18.5 未満	やせ気味です
25 未満	標準の範囲です
30 未満	肥満気味です
それ以外	だいぶ肥満です

kadai2-2 (20 点)

電卓を作る。整数で x と y の値を入力する。演算の内容は+、-、*、/、%のいずれかとする。演算の内容は整数で入力を行い、+ならば 1, -ならば 2, *ならば 3, /ならば 4, %ならば 5 とする。条件判断処理には switch 文を利用すること。

※%を printf で出力する場合は%%と書く。