第 3 章　条件分岐

本章では、プログラムの流れに分岐を作る「if」文と「switch」文を説明します。

3.1　if 文

「if」文は英語での意味のとおり、「もしも・・・ならば」という条件分岐を表します。条件分岐の例として、次の文を考えてみましょう。

もしも成績が 60 点以上ならば、単位がもらえる

皆さんに今後待ち受ける未来として、「成績が 60 点以上で単位がもらえる」と「成績が 59 点以下で単位がもらえない」の二通りが存在します。このように、特定の条件によって分岐が生じる（異なった未来が待ち受けている）ことを、**条件分岐**と言います。「降水確率が 30%以上なら傘を持って出かける」「身長が 130 cm 以上ならジェットコースターに乗れる」など、世の中のあらゆることは条件分岐の組み合わせでできています。

実際に、以下の仕様を持つプログラムを見てみましょう。

- 成績の入力を受け付ける
- 成績が 60 点以上なら「単位がもらえます。」と表示する

例 3-1

```c
#include <stdio.h>

int main(void) {
    int score;

    printf("成績を入力して下さい: ");
    scanf("%d", &score);

    if (score >= 60) {
        printf("単位がもらえます。\n");
    }

    return 0;
}
```

例 3-1 実行例

```
$ ./a
成績を入力して下さい: 80
単位がもらえます。

$ ./a
成績を入力して下さい: 50
```

このように if 文は、「条件式」と、条件式を満たしていた際の「処理」で構成されます。「条件式」には、「score が 60 点以上」を表す「score >= 60」が入っています。また、「処理」の範囲は main 関数と同じように波括弧 { } で囲って表します。ただし、処理が 1 行である場合には、波括弧を省略することも可能です。

```
if ( 条件式 ) {
    // 条件式を満たしていた場合の処理
}

if ( 条件式 )
    // 処理が 1 行であれば波括弧は省略可
```

3.2　条件の表し方

前節の例では、「以上」を表す記号として「>=」を用いました。このように、二つの数値の関係を表す記号を「関係演算子」（もしくは「比較演算子」）と言います。

関係演算子

関係演算子は、if 文や for, while 文（第 4 章）の条件式の中でとてもよく使われます。他にどのような関係演算子があるのか、以下の表を見てみましょう。

演算子	意味	数学記号での表記
`n > 60`	n が 60 よりも大きい	$n > 60$
`n < 60`	n が 60 よりも小さい	$n < 60$
`n >= 60`	n が 60 以上	$n \geq 60$
`n <= 60`	n が 60 以下	$n \leq 60$
`n == 60`	n が 60 と等しい	$n = 60$
`n != 60`	n が 60 と等しくない	$n \neq 60$

関係演算子を用いることで、数学で行うものと同じように、変数や数値の比較をすることが可能です。ただし、C 言語のプログラミングで使用可能な記号の数はかなり制限されているため、「≥」は「>=」、「≤」は「<=」、「≠」は「!=」のように表します。また、C 言語における「等しい」の関係演算子は、「==」と表すことに注意して下さい。第 2 章でも述べたように、C 言語上で「=」は代入演算子であり、「右辺のものを左辺に代入する」という意味になります。

論理演算子

if 文の条件式の構成要素には、「関係演算子」の他に「論理演算子」というものがあります。例えば、「成績が 60 点以上かつ 70 点未満の場合」のように、二つ以上の条件をまとめて扱いたい場合があります。数学では「$60 \leq n < 70$」のように書くことができますが、C 言語では「`60 <= n < 70`」のようにまとめて書くことはできません。このようなときは、以下の表に示す論理演算子を使います。

演算	意味	使用例	使用例の意味
`&&`	どちらも真	`60 <= n && n < 70`	n が 60 以上かつ 70 未満
`||`	どちらかが真	`n < 60 || 70 <= n`	n が 60 未満または 70 以上
`!`	否定	`!(n < 60)`	n が 60 以上でない（60 未満）

すなわち、「成績が 60 点以上かつ 70 点未満の場合」という条件分岐を作りたい場合は、「成績が 60 点以上」と「成績が 70 点未満」の二つの条件に分け、「&&」を用いて以下のようにします。

```
if (60 <= n && n < 70)
```

関係演算子と論理演算子を組み合わせれば、さまざまな条件を作ることが可能になります。また、数学で「&&」は「∩（論理積）」、「||」は「∪（論理和）」のことであり、それぞれ「アンド（and）」「オア（or）」と呼ばれます。

3.3　if〜else 文

ここまでで、if 文を使って「もし〜ならば、この処理をする」というプログラムが書けるようになりまりた。しかし、「そうでなければ、この処理をする」というプログラムを書くことも頻繁にあります。そのような場合は、「else」文を使います。実際の使用例を見てみましょう。

例 3-2

```c
#include <stdio.h>

int main(void) {
    int score;

    printf("成績を入力して下さい: ");
    scanf("%d", &score);

    if (score >= 60) {
        printf("単位がもらえます。\n");
    } else {
        printf("不可です。\n");
    }

    return 0;
}
```

例 3-2 実行例

```
$ ./a
成績を入力して下さい: 80
単位がもらえます。

$ ./a
成績を入力して下さい: 50
不可です。
```

例 3-1 からの変更点は、12〜14 行目の部分のみです。このプログラムは、成績が 60 点以上であるかどうかを確認し、そうであれば「単位がもらえます。」を、そうでない場合は「不可です。」を表示するものとなっています。このように、if 文の後に else 文を入れることで、if 文の条件を満たさなかった場合の処理を書くことができます。

```
if (条件式) {
    // 条件式を満たす場合の処理
} else {
    // 条件式を満たさない場合の処理
}
```

また、「else if」文を入れることで、以下のように細かい条件分岐を作ることもできます。else if 文は幾つでも追加することが可能です。

```
if (条件式 A) {
    // 条件式 A を満たす場合の処理
} else if (条件式 B){
    // 条件式 B を満たす場合の処理
} else if (条件式 C){
    // 条件式 C を満たす場合の処理
} else {
    // いずれの条件式も満たさない場合の処理
}
```

if〜else if〜else 文を使って、以下の仕様を持つプログラムを作るとどうなるか、見てみましょう。

- 成績の入力を受け付ける
- 成績が 70 点以上なら「余裕で単位がもらえます。」と表示する
- 成績が 60 点以上 70 点未満なら「ぎりぎり単位がもらえます。」と表示する
- 成績が 60 点未満なら「不可です。」と表示する

例 3-3

```
1  #include <stdio.h>
2
3  int main(void) {
4      int score;
5
6      printf("成績を入力して下さい: ");
7      scanf("%d", &score);
8
9      if (score >= 70) {
10         printf("余裕で単位がもらえます。\n");
11     } else if (score >= 60) {
12         printf("ぎりぎり単位がもらえます。\n");
13     } else {
14         printf("不可です。\n");
15     }
16
17     return 0;
18 }
```

ここで注目してもらいたいのは、12 行目の条件文です。この else if 文では単に「score が 60 点以上」という条件を書いていますが、この分岐点に来るということは直前の if 文で「score が 70 点以上」という条件を満たしていないので、必然的に「score は 70 未満」ということになります。すなわち、これだけで「成績が 60 点以上 70 点未満」という条件を作ることができるというわけです。もちろん、論理演算子「&&」を用いて明示的に「`60 <= score && score < 70`」を書いても構いません。

ホワイトボックステスト（網羅率の検証）

プログラムを書いた場合、必ず正しく動作するかをチェックしなければなりません。特に if 文や後述の switch 文を使う場合はプログラムに分岐ができるので、処理の分岐が正しく行われているか、各処理が正しく動作するかを検証する必要があります。この検証はホワイトボックステストと呼ばれ、テストの有効性は網羅率で表されます。代表的な「網羅性」を以下に示します。

- 分岐網羅・・・全ての分岐を必ず一度は実行すること
- 条件網羅・・・全ての条件の組み合わせを必ず一度は実行すること

以下のソースコードを考えてみましょう。

```
if (a < 5 && b < 10) {
    処理 A;
} else {
    処理 B;
}
```

このとき、「a = 0, b = 6」と「a = 6, b = 11」などの 2 つのケースでプログラムの動作を確認すれば、処理 A と処理 B が実行されることになり、分岐網羅を満たします。しかし、2 つの条件式の真偽の組み合わせは、4 通りあります。条件網羅を満たすには、さらに「a = 0, b = 11」「a = 6, b = 6」などのケースを試し、全ての条件式の真偽の組み合わせで動作を確認する必要があります。プログラムは複雑になるにつれ、エラーやバグが生じやすくなります。そのような自体を防ぐため、ソースコードが書けたら少なくとも「分岐網羅」、できれば「条件網羅」を満たすよう、ホワイトボックステストを実施するようにしましょう。

演習 3-1 (ex3-1.c)

例 3-3 を参考に、以下の仕様を持つプログラムを作成しなさい。

- 成績の入力を受け付ける
- 成績が 0 点未満または 100 点より大きい場合、
  「正しい値を入力して下さい。」と表示する
- 成績が 90 点以上 100 点以下なら「S 単位がもらえます。」と表示する
- 成績が 80 点以上 90 点未満なら「A 単位がもらえます。」と表示する
- 成績が 70 点以上 80 点未満なら「B 単位がもらえます。」と表示する
- 成績が 60 点以上 70 点未満なら「C 単位がもらえます。」と表示する
- 成績が 60 点未満なら「不可です。」と表示する

3.4　switch～case 文

プログラムの分岐を作るには、前節までで紹介した「if」文の他に、「switch」文があります。switch 文は、分岐の数が多いときによく使われる構文で、書式は以下に示すとおりです。

```
switch ( 式 ) {
    case 定数式 A:
        // 式と定数式 A が一致した場合の処理
        break;
    case 定数式 B:
        // 式と定数式 B が一致した場合の処理
        break;
    default:
        // 式がどの定数式とも一致しなかった場合の処理
        break;
}
```

書式を見ただけではわかりにくいと思うので、実際に if 文で書かれたソースコードを、switch 文に書き直すことを考えてみましょう。以下に、if 文による分岐を実装したコードを示します。

例 3-4

```
1  #include <stdio.h>
2
3  int main(void) {
4      char c;
5
6      printf("あなたの好きなスポーツは？\n");
7      printf("  (a) 野球\n");
8      printf("  (b) サッカー\n");
9      printf("  (c) 特にない\n");
10     scanf("%c", &c);
11
12     if (c == 'a') {
13         printf("やっぱり日本人なら野球ですよね。\n");
```

```
    } else if (c == 'b') {
        printf("サッカー、格好いいですよね。\n");
    } else if (c == 'c') {
        printf("特にないのですね。\n");
    } else {
        printf("不正な文字が入力されました。\n");
    }

    return 0;
}
```

このプログラムは、好きなスポーツに対応するアルファベットの入力を受け取り、結果を表示するものです。これを switch 文に書き換えると、次のようになります。

例 3-5

```
#include <stdio.h>

int main(void) {
    char c;

    printf("あなたの好きなスポーツは？\n");
    printf("  (a) 野球\n");
    printf("  (b) サッカー\n");
    printf("  (c) 特にない\n");
    scanf("%c", &c);

    switch (c) {
        case 'a':
            printf("やっぱり日本人なら野球ですよね。\n");
            break;
        case 'b':
            printf("サッカー、格好いいですよね。\n");
            break;
        case 'c':
            printf("特にないのですね。\n");
```

```
21              break;
22          default:
23              printf("不正な文字が入力されました。\n");
24              break;
25      }
26
27      return 0;
28  }
```

if 文のコードと比べてソースコードの行数は増えてしまっていますが、シンプルで読みやすい構造になっていることがわかると思います。重要な行を以下で説明していきます。

11 行目：　switch 文ではこのように、switch の後に括弧（）で何らかの式を囲います。式には整数を結果とするもののみが利用可能です（char 型は整数と同様に扱われます）。

12 行目：　分岐は case 文で作っていきます。case の後に半角スペースを空けて定数式を記述すると、switch の式と case の定数式が一致したとき、case の中に書かれたコードが実行されます。また、case 文の行はセミコロン「;」ではなくコロン「:」で終わることに注意して下さい。

14 行目：　case 文の中の処理が終わったら、break 文を用いて switch の { } から抜け出します。一度 case 文の中に入った場合、break 文で抜け出さない限り、それ以降に書かれているすべての case 文（および default 文）の処理が実行されます。

21 行目：　どの case にも当てはまらない場合の処理は、default 文の中に記述します。case 文での default 文は、if 文で言うところの else 文に相当します。

演習 3-2 (ex3-2.c)

例 3-5 のソースコードを元に、「野球」「サッカー」以外のスポーツ（例えばテニス）も選択できるできよう書き換えて下さい。また、break 文が無い場合にどのような挙動を示すか、実際に確認してみて下さい。

演習 3-3 (ex3-3.c)

演習 3-2 のソースコードを、大文字のアルファベットの入力にも対応したプログラムに改良して下さい。

ヒント： 以下のようなコードを書くと、例 3-5 の 14 行目のコード説明にある性質により、'a'を入力しても'A'を入力しても同じ結果を表示することができます。

```
case 'A':
case 'a':
    printf("やっぱり日本人なら野球ですよね。\n");
    break;
```

（補足： if 文を使って書く場合は、「if (c == 'A' || c == 'a')」のようにします。）

3.5　課題

課題 3-1

整数値の入力を受け付け、奇数であれば「奇数です。」、偶数であれば「偶数です。」と表示するプログラムを作って下さい。(if 文を利用)

ヒント： 偶数は 2 で割った余りが 0 になります。

課題 3-2

生まれ年を西暦で入力すると、その年の干支を表示するプログラムを作って下さい。(switch 文を利用)

ヒント： 西暦を 12 で割った余りが 0 のとき、干支は申です。

コラム 2　インデントスタイル

main 関数や if 文を利用する際、波括弧 {} で囲われた中身にインデント（字下げ）を入れ、処理の範囲をわかりやすくしてきました。このインデントを入れるスタイルは、実にさまざまなものが提案されています。

まだ