基本制御構造 (順次と選択)

山本昌志*

2004 年 9 月 3 日

1　先週の復習と本日の内容

先週は制御式について学習した。その内容は、次の通りであった。

- 制御式の計算結果は、0(偽) か 1(真) である。
- 制御式の演算子には次のようなものがある。

関係演算子	`<`　`<=`　`>=`　`>`
等価演算子	`==`　`!=`
論理演算子	`&&`　`||`　`!`

本日は、先週の制御式を利用した構文を主に学習する。主な内容は `if` 文であるが、`switch` 文についても述べる。

2　プログラミング技法とプログラミング言語の変遷

プログラミング言語の進化について、ハーバート・シルト著の「独習 C++」[1]には、次のように書かれている。分かりやすいので引用しておく。

> プログラミングには、初期の頃からさまざまな方法論が使われてきました。プログラミングの発展における節目では、ますます複雑になるプログラムに対処するために、新しいアプローチが開発されてきました。最初のプログラムは、コンピューターのフロントパネルにあるスイッチをオンとオフの間で切り替えることによって作成されました。しかし、この手法はごく短いプログラムにしか適さないことは明らかです。次にアセンブリ言語が開発され、これによってより長いプログラムを作成できるようになりました。次の進歩がもたらされたのは、1950 年代、最初の高レベル言語 (FORTRAN) が開発されたときでした。
>
> 高レベル言語を使うことによって、プログラマは数千行にわたるプログラムを作成することができるようになりました。しかし、初期に使われていたプログラミング方式はアドホック[2] であ

*国立秋田工業高等専門学校　電気情報工学科

[1]株式会社　翔泳社, ISBN4-7981-0318-7。第 3 版の P.3〜引用した。

[2]adhoc:特別の［に］ (an 〜 committee 特別［臨時］委員会) ; その場限り（の） (an 〜 policy 暫定策［方針］)

り、何でも許される方式でした。比較的小さなプログラムならばこれでもかまいませんが、大きなプログラムにこの方法を適用すると、読みづらい (そして保守しにくい)「スパゲティコード」が出来上がりました。1960 年代には、**構造化プログラミング言語**(structured programming language) の登場によって、スパゲティコードを排除することが可能になりました。この種の言語には Algol や Pascal があります。広い意味では C 言語も構造化言語であり、これまで読者が使用してきたプログラミング言語は大部分が構造化プログラミングと呼ばれる類のものでしょう。構造化プログラミングでは、厳密に定義された制御構造、コードブロックを使用し、「GOTO」を使わず (あるいは、少なくともその使用を最小限に抑え)、再帰とローカル変数をサポートする独立したサブルーチンを使用します。構造化プログラミングの要点は、プログラムをその構成要素に簡約することです。構造化プログラミングの手法を用いれば、凡庸なプログラマでも 50,000 行に及ぶ長さのプログラムを作成し、保守することができます。

構造化プログラミングは、適度な複雑さのプログラムに適用したときは優れた効果をもたらしますが、プログラムサイズがある程度に達すると限界が現れます。より複雑なプログラムを作成するためには、新しいプログラミング手法が必要となります。そこで開発されたのがオブジェクト指向プログラミングです。オブジェクト指向プログラミングでは、...

オブジェクト指向プログラミング言語、例えば C++などはここで学習ない。C 言語を使えば、凡庸なプログラマーでも 50,000 行のプログラムを開発、保守できるということなので、大体、間に合うでしょう。C++などのオブジェクト指向言語は便利そうなので、自分で学習するには良いであろう。

3 構造化プログラミング

3.1 基本制御構造

構造化プログラミングは、順次、選択、繰り返し (反復、ループ) の基本制御構造でプログラムを構築する方法である。これらを用いることで、無条件分岐文 (goto) 文を使わないでプログラミングでき、分かりやすいソースコードを書くことが出来る。

プログラムの構造は、この 3 つからできている。この 3 つを理解できれば、プログラムを書くことは容易になるであろう。たった 3 つである。これらのフローチャートを図 1〜図 3 に示す。

3.2 C 言語の制御式

ここで、制御式の真と偽について注意をしておく。先週述べたことであるが、

> C 言語では制御式の値が 0 のときのみ偽として取り扱われる。0 以外のとき真である。

を忘れてはならない。これは、C 言語全てに成り立つので、これ以降の構文も同じである。

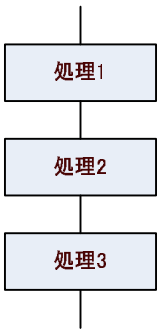

図 1: 順次の構造

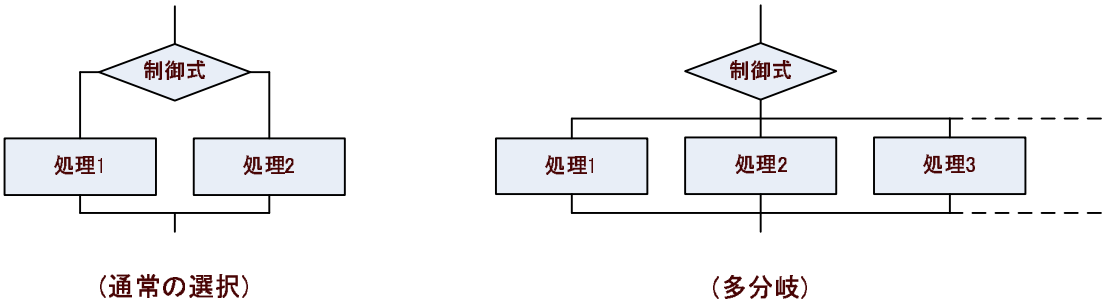

図 2: 選択の構造

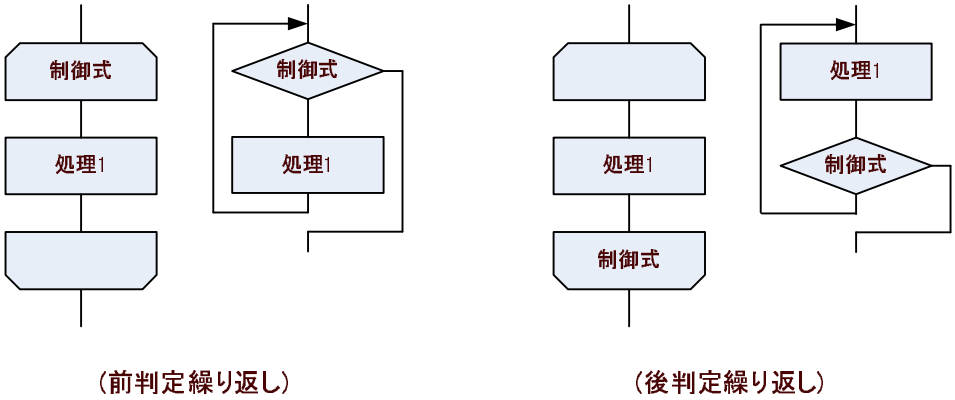

図 3: 繰り返しの構造

4　基本制御構造 (順次)

これは、もっとも基本的な構造で、文を上から下へと順次、実行していく。いつもおなじみの構造で、以下のように記述する。

書式

```
文 1;
文 2;
文 3;
```

文の数は、いくら書いてもよい。フローチャートで書くと、図 4 のようになる。以下のようなプログラムが、この構文の使用例である。

```
scanf("%d",&a);
b = a*a;
printf("%d*%d=%d\n",a,a,b);
```

- a の値をキーボードから読み込む。
- a*a を計算し、b に代入する。
- 「a*a=b」の各値を表示する。

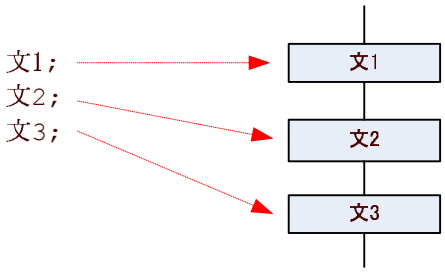

図 4: 順次の構文のフローチャート

5　基本制御構造 (選択)

C 言語の選択には、if と switch がある。ここでは、それぞれについて説明する。

5.1　if else 文

プログラム中で、「もし○○ならば、△△する」というような処理をしたい場合、if という命令を使う。また、「もし、○○ならば△△する、さもなければ□□する」という場合は、if と else を使う。ここでは、

この if や else の使い方を学習する。
分かっていると思うが、念のため、これらの意味を書いておく。

if	もしも〜ならば
else	さもなければ

5.1.1 処理が 1 つの場合

最初は一番単純な、「もし○○ならば、△△する」という構文を示す。とくに、△△の部分が 1 つの文で表せる場合である。このような場合は、次のように、書く。

書式

```
if(制御式) 文;
```

条件を表す○○の部分が制御式で、△△の部分を文で表すのである。これは、「制御式が正しい (真) ならば、文を実行する」となる。もし、制御式が誤り (偽) であれば、この文は実行されず次の行に移る。図 5 にこの構文のフローチャート (flow chart:流れ図) 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。

```
if(a<=10) printf("a は、10 以下です\n");
```

- a が 10 以下ならば、
  - 「a は、10 以下です」と表示する。

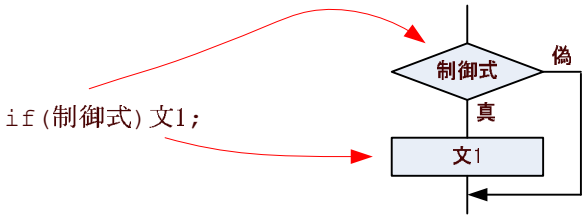

図 5: 制御式が真の場合、1 つの処理を実施する if 文

5.1.2 ブロックで処理する場合

先ほどの構文では、実行できる文は 1 個に限られる。「もし○○ならば、△△し、□□し、···」のように複数の文を実行したい場合がある。このようなときは、次に示すように、{ と } でくくり、ブロック化して、複数の文を書く。ブロック内には、任意の数の文を書くことができる。また、このブロック内には、順次や選択、繰り返しの文を書くことも可能である。

書式

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

これは、「制御式が正しい (真) ならば、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 6 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。

```
if(0<=a && a<=10){
  printf("a は、0 以上\n");
  printf("かつ\n");
  printf("a は、10 以下です\n");
}
```

- もし、a が 0 以上、かつ、10 以下ならば、
  - 「a は、0 以上」と表示する。
  - 「かつ」と表示する。
  - 「a は、10 以下です」と表示する。

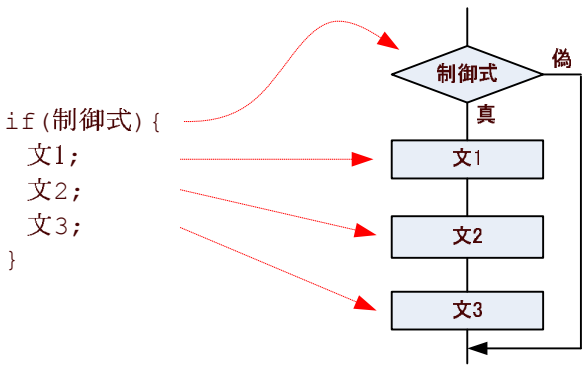

図 6: 制御式が真の場合、ブロックで処理を実施する if 文

5.1.3 2 分岐の場合

「もし○○ならば△△し、さもなければ□□する」というように、条件により二者択一の選択処理が必要な場合がある。これは、次のように書く。

書式

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

これは、「制御式が正しい (真) ならば、文 1 と文 2、文 3 を実行する。さもなければ、文 4 と文 5、文 6 を実行する。」となる。実行される文が複数であるので、ブロックになっていることに注意。文が 1 つの場合、、{ と } でくくり、ブロック化しなくても良い。図 7 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。

```
if(0<=a && a<=10){
  printf("a は、0 以上\n");
  printf("かつ\n");
  printf("a は、10 以下です\n");
}else{
  printf("a は、0 未満\n");
  printf("または\n");
  printf("a は、10 より大きい\n");
}
```

- もし、a が 0 以上、かつ、10 以下ならば、
  - 「a は、0 以上」と表示する。
  - 「かつ」と表示する。
  - 「a は、10 以下です」と表示する。
- さもなければ
  - 「a は、0 未満」と表示する。
  - 「または」と表示する。
  - 「a は、10 より大きい」と表示する。

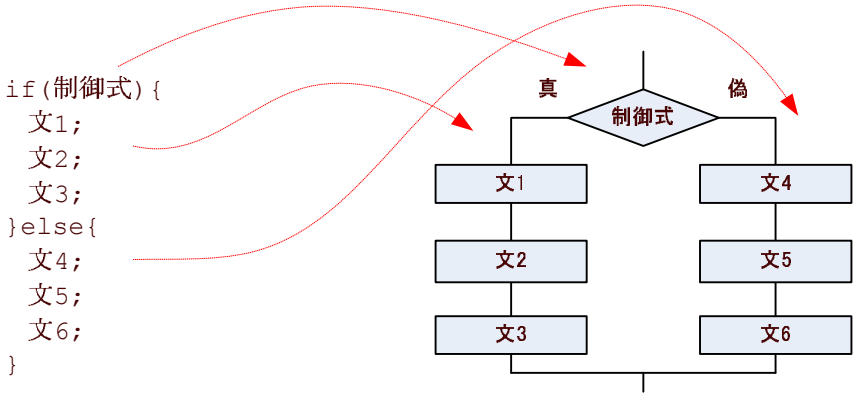

図 7: else を使って、二者択一の処理をする構文

5.1.4 連続制御の分岐

「もし○○ならば ⊛⊛ する。さもなければ、もし□□ならば ⊠⊠ する。さもなければ、もし △△ ならば ▽▽ する。さもなければ、◎◎ する。」というよう構文を書きたい場合がある。条件に合致しなければ、次の条件と、次々に条件を変ている。これは、次のように書く。

書式

```
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;
}
```

これは、「制御式 1 が正しい (真) ならば、文 1 と文 2 を実行する。さもなければ、制御式 1 が正しいならば、文 3 と文 4 を実行する。さもなければ、制御式 3 が正しいならば、文 5 と文 6 を実行する。さもなければ、文 7 と文 8 を実行する。」となる。

この構文のフローチャートを、図 8 に示す。このフローチャートを見てわかるように、最初に真となった制御式に続くブロック内が実行されのみである。それ以降、真になっても、そのブロックは実行されない。どの制御式も真にならない場合、最後の `else` のブロックが実行される。即ち、実行されるブロックは 1 個のみである。`else if` の段数をいくらでも増やせることは、言うまでもない。

また、elseが無い構文も許される。この場合、真となる制御式がない場合、どのブロックも実行されず、この構文から抜ける。

つぎのプログラムが、この構文の使用例である。

```
if(a < 0){
  printf("aは、0以下\n");
}else if (0 <= a && a < 1){
  printf("aは、0以上\n");
  printf("かつ\n");
  printf("aは、1未満\n");
}else if (1 <= a && a < 10){
  printf("aは、1以上\n");
  printf("かつ\n");
  printf("aは、10未満\n");
}else{
  printf("aは、10以上\n");
}
```

- もし、aが0未満ならば、
  - 「aは、0以下」と表示する。
- さもなければ、もし、aが0以上、かつ、1未満ならば
  - 「aは、0以上」と表示する。
  - 「かつ」と表示する。
  - 「aは、1未満」と表示する。
- さもなければ、もし、aが1以上、かつ、10未満ならば
  - 「aは、1以上」と表示する。
  - 「かつ」と表示する。
  - 「aは、10未満」と表示する。
- さもなければ
  - 「aは、10以上」と表示する。

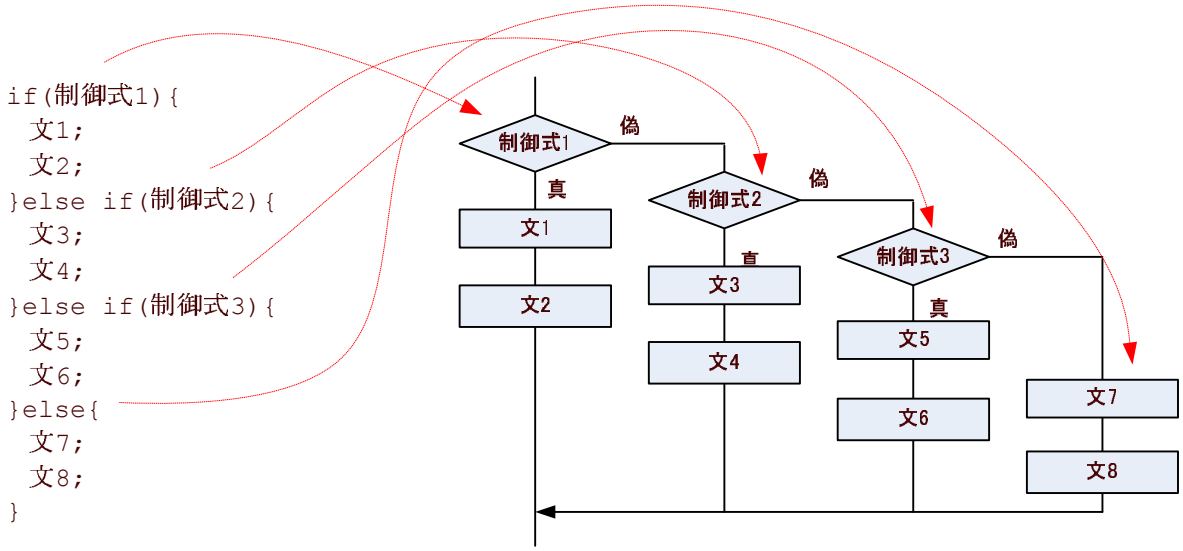

図 8: if else if else を使っての多段の選択

5.2 switch

if 文は、選択肢が少ない場合、わかりやすい記述ができる。しかし、選択肢が多くなると、記述は複雑になり、分かりにくいプログラムとなる。そのような場合は、if 文の代わりに switch 文を使うことができる。

この構文のフローチャートを、図 9 に示す。これは、式の値により、それにマッチしたブロック[3]が実行される。もし、どれもマッチしなければ、default が実行される。default 文は無くてもよいが、その場合はどのブロックも実行されない場合がある。

文の集まりのブロックの最後には、bread 文を書く。この break 文が無いと、マッチしたブロック以降の他のブロックも実行される。コードブロックを表す中括弧 {} が無いので、こうすることになっている。この break 文で switch 文の終わりを示す中括弧 (}) から

式や定数式の値の型は、int または char でなくてはならない。定数式の方は、コンパイル時に、評価できなくてはならない。

case の後の定数式は、ラベルである。ラベルの後は、コロン (:) をつける。文の終わりを示すセミコロン (;) ではない。

つぎのプログラムが、この構文の使用例である。

```
switch(a){
  case 1:
    printf("あなたは、1 と答えました。\n");
    printf("不正解です。\n");
    break;
```

[3]コードブロックではないので、中括弧 ({}) がない。

```
    case 2:
      printf("あなたは、2と答えました。\n");
      printf("不正解です。\n");
      break;
    case 5:
      printf("あなたは、5と答えました。\n");
      printf("正解です\n");
      break;
    default:
      printf("質問にまじめに答えろ。\n");
  }
```

- aが1ならば、以下を実行する。
  - 「あなたは、1と答えました。」と表示する。
  - 「不正解です。」と表示する。
  - switchの構文から抜ける。
- aが2ならば、以下を実行する。
  - 「あなたは、2と答えました。」と表示する。
  - 「不正解です。」と表示する。
  - switchの構文から抜ける。
- aが5ならば、以下を実行する。
  - 「あなたは、5と答えました。」と表示する。
  - 「正解です。」と表示する。
  - switchの構文から抜ける。
- どれにもマッチしなければ、以下を実行する。。
  - 「質問にまじめに答えろ。」と表示する。

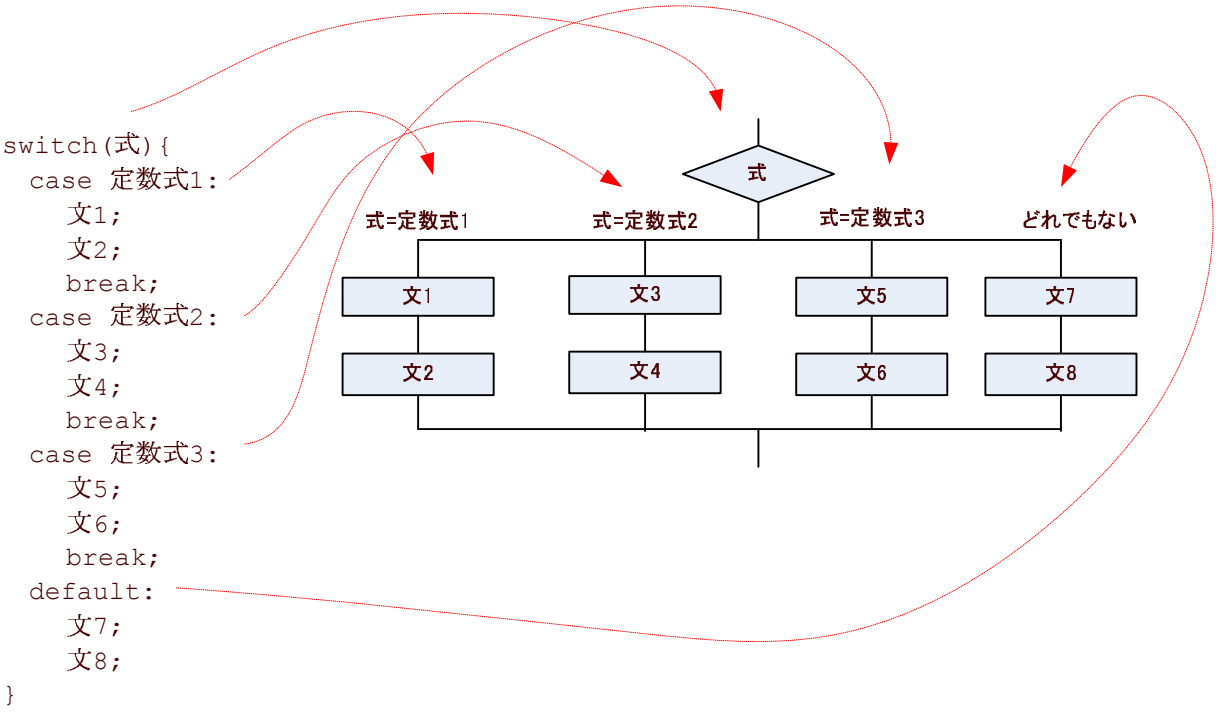

図 9: switch 文を使った構文。多くの選択肢がある場合。

6 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。

提出方法は、次の通りとする。

期限　次回の授業まで
用紙　A4
表紙　授業科目名「情報処理 I」
　　課題名「基本制御構造 (順次と選択)」
　　1E　学籍番号　氏名
　　提出日

6.1 順次

1. 表示
   - 「秋田県秋田市」と表示する。
   - 「秋田工業高等専門学校」と表示する。

2. キーボード入力と計算結果出力

    - b の値をキーボードから読み込む。
    - b+b を計算し、a に代入する。
    - 「b+b=a」の各値を表示する。

6.2 選択

6.2.1 処理が 1 つの場合

1. 大小関係の判定

    - 整数をキーボードから、読み込む。
    - a が 10 以上ならば、
        - 「a は、10 以上です」と表示する。

2. 正負の判定

    - 整数をキーボードから、読み込む。
    - 読み込んだ値が負ならば、
        - 「値は、負です」と表示する。

6.2.2 ブロックで処理する場合

1. 範囲の判断その 1

    - 整数をキーボードから、読み込む。
    - 読み込んだ値が 5 以上、20 以下ならば、
        - 「a は、5 以上」と表示する。
        - 「かつ」と表示する。
        - 「a は、20 以下です」と表示する。

2. 範囲の判断 2

    - 整数をキーボードから、読み込む。
    - 読み込んだ値が負、または、100 以上ならば、
        - 「値は、負」と表示する。
        - 「または」と表示する。
        - 「値は、100 以上です」と表示する。

6.2.3　2 分岐の場合

1. 整数の範囲
    - 整数をキーボードから、読み込む。
    - もし、a が 20 以上、かつ、100 以下ならば、
        - 「a は、20 以上」と表示する。
        - 「かつ」と表示する。
        - 「a は、100 以下です」と表示する。
    - さもなければ
        - 「a は、20 未満」と表示する。
        - 「または」と表示する。
        - 「a は、100 より大きい」と表示する。
2. 整数の計算
    - 整数 a をキーボードから、読み込む。
    - もし、a が 1 ならば、
        - 「a は、1 です」と表示する。
        - 「a は、負ではありません」と表示する。
    - さもなければ
        - 「a は、1 では有りません」と表示する。
        - 「a*a=a^2」の各値を表示する。

6.2.4　連続制御の分岐

1. 整数の範囲
    - 整数 a をキーボードから、読み込む。
    - もし、a が-1 未満ならば、
        - 「a は、-1 以下」と表示する。
    - さもなければ、もし、a が-1 以上、かつ、0 未満ならば
        - 「a は、-1 以上」と表示する。
        - 「かつ」と表示する。
        - 「a は、0 未満」と表示する。
    - さもなければ、もし、a が 0 以上、かつ、1 未満ならば
        - 「a は、0 以上」と表示する。

- 「かつ」と表示する。
- 「a は、1 未満」と表示する。

- さもなければ
  - 「a は、1 以上」と表示する。

2. 整数の大小関係
   - 整数 a をキーボードから、読み込む。
   - 整数 b をキーボードから、読み込む。
   - もし、a が b より小さければ、
     - 「a は、b より小さい」と表示する。
   - さもなければ、もし、a と b が等しければ
     - 「a と b は等しい」と表示する。
   - さもなければ
     - 「a は、b より大きい」と表示する。

6.3 switch

1. クイズ
   - 画面に「情報処理 I の担当教員は?」と表示する。
   - 画面に「1:山田　2:山上 3:山本」と表示する。
   - 整数 a をキーボードから、読み込む。
   - a が 1 ならば、以下を実行する。
     - 「山田ではありません。」と表示する。
     - 「不正解です。」と表示する。
     - switch の構文から抜ける。
   - a が 2 ならば、以下を実行する。
     - 「山上ではありません。」と表示する。
     - 「不正解です。」と表示する。
     - switch の構文から抜ける。
   - a が 3 ならば、以下を実行する。
     - 「情報処理 1 の担当は、山本です。」と表示する。
     - 「正解です。」と表示する。
     - switch の構文から抜ける。
   - どれにもマッチしなければ、以下を実行する。。
     - 「質問にまじめに答えろ。」と表示する。