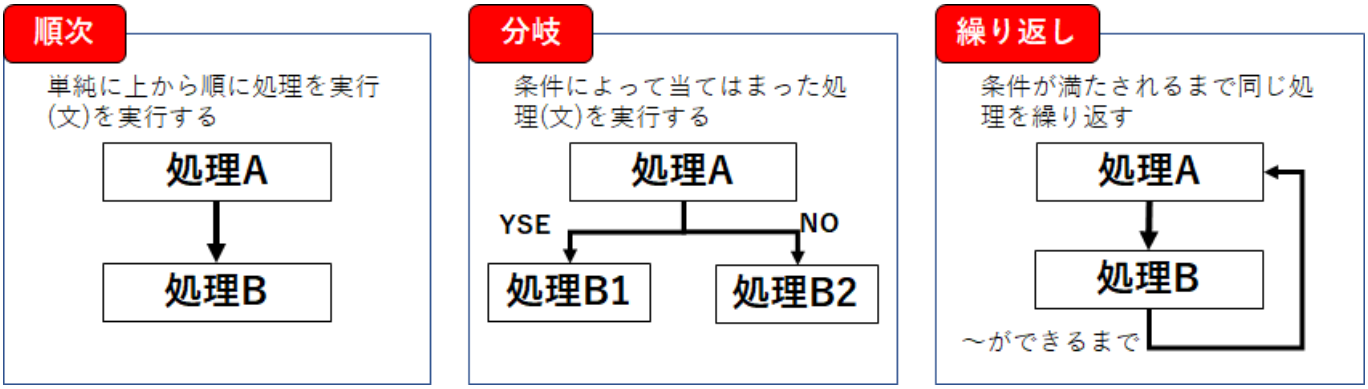
プログラムの流れ

代表的な制御構造

基本的にプロクラムが読まれる処理の順番が「 **上から順に1つずつ** 」実行されるのがルールとなっています。文を実行させる順番のことを **制御構造** (制御フロー)と呼び、代表的なものが以下の3つとなります。

1. 順次
2. 分岐
3. 繰り返し



1. 順次

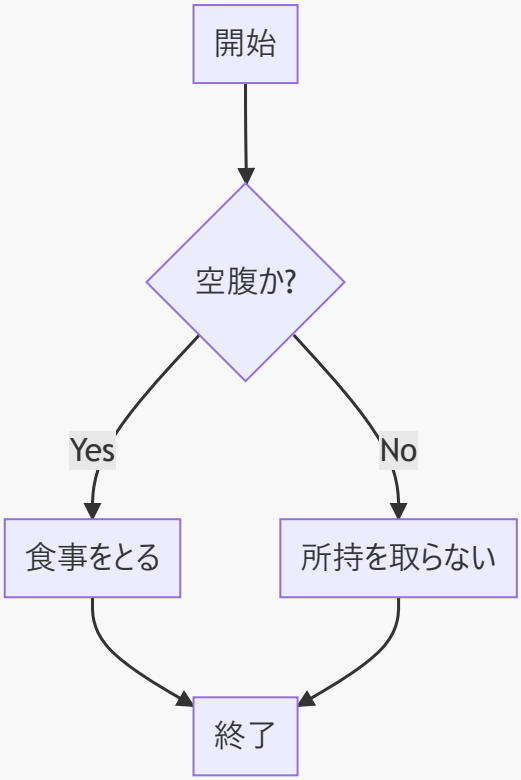
基本的な処理となる制御構造です。上記の画像上の処理Aに「分岐」や「繰り返し」処理が入るかもしれません。
この制御構造の上に様々の制御(プログラミング)が記述され処理されていきます。

2. 分岐

条件によって次に行いたい処理を分岐させる制御構造となります。
日常生活で例えるなら、

- もし「お腹が空いたら」食事をとる
- もし「お腹が空いていないなら」食事をとらない

このように**もし**「お腹が空いていたら」「お腹が空いていなかったら」でそのあとの「食事をとる」「食事をとらない」といった様に次に行う処理を **条件によって変化させている** 制御構造になっています。



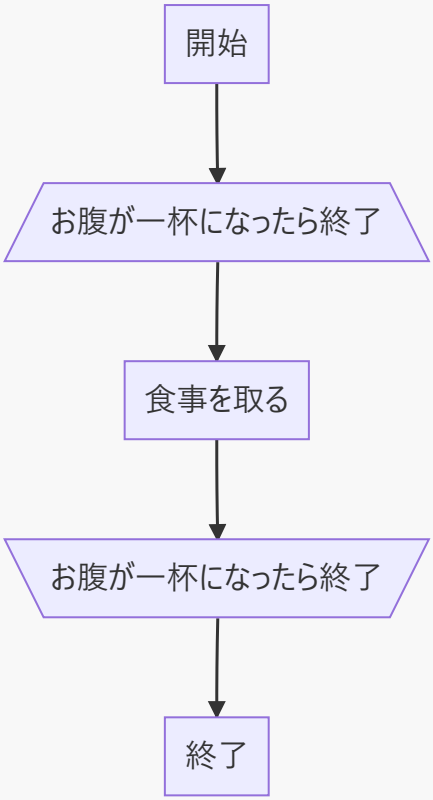
3. 繰り返し

条件が満たされるまで同じ処理をさせる制御構造となります。

日常生活で例えると、「お腹が空いている」状態が条件だとします。すると

- 食事を取る

この処理が条件を満たすまで何度も同じ処理を **繰り返し行い条件が満たされると処理が終了する** 制御構造となっています。



分岐処理(if文)を書いてみる

2. 分岐でも記載しましたが分岐処理は「**もしも～の場合**」は「この処理」をしますという動作になります。
基本的な構文で記述すると以下の様になる。

```
if (条件) {
    処理
}
```

- `if`
  分岐を指定する命令後
- (条件)
  分岐の条件を記述する。**条件として、booleanのture/falseの状態になる条件しか記述することができません。**

こんな例題があったとします。

> 国語、数学、英語の試験の点数の合計によって「合格/補習/不合格」を判定して表示するプログラムを作成してください。
> 以下が条件になります。
> ・合格: 合計点が240点以上の場合
> ・補習: 合計点が240点未満で150点以上の場合
> ・不合格: 150点未満の場合

この条件を満たすプログラミングを以下の様に作成してみました。
if文で条件式を3記述することで例題の条件を満たすことができます。

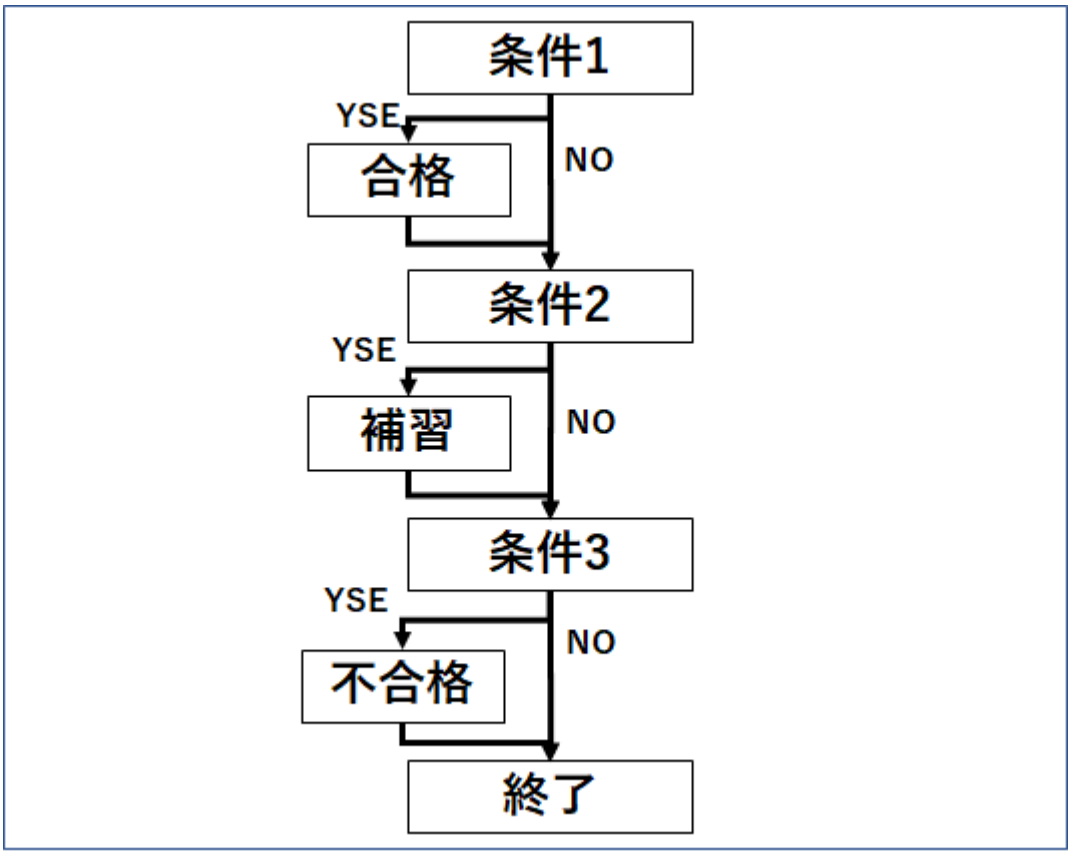
if文のみを使用して記述した場合

```
public class Main {
    public static void main(String[] args) {
        // 試験の点数によって合格/補習/不合格かを判定するコード
        // 国語の点数
        int kokugo = 50;
        // 数学の点数
        int sugaku = 50;
        // 英語の点数
        int eigo = 50;
        // 合計点: 150
        int goukei = kokugo + sugaku + eigo;

        // 条件式
        // 240点以上なら合格
        // 150点~240点未満なら補習
        // 150点未満なら封合格
        if(goukei >= 240) { // 条件式1
            System.out.println("合格");
        }
        if(goukei >= 150 && goukei < 240) { // 条件式2
            System.out.println("補習");
        }
        if(goukei < 150){ // 条件式3
            System.out.println("不合格");
        }

    }
}
```

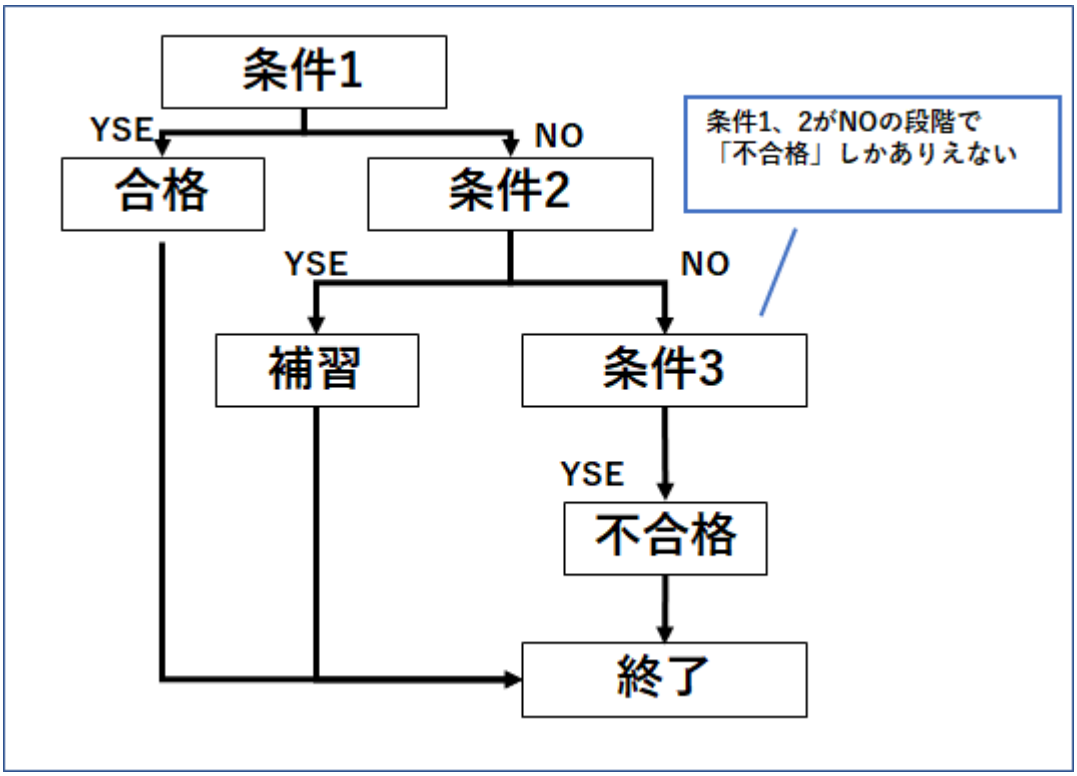
今回の処理手順を図にしてみると以下の様になります。



処理の順序は制御工場の順次により条件式1, 2, 3の順番で判定が行われます。今回は、`goukei`は150点のため条件式2のif文が実行されることになります。
しかし、仮に条件式2の'if(goukei >= 150 && goukei < 240)'の以上未満の条件を間違えて'if(goukei <= 150 && goukei < 240)'**240点未満と150点未満の条件を満たした場合**になってしまいますと条件式2と条件式3が実行されてしまいます。

else文なども併用して記述した場合

if文にも条件式をまとめることができます。今回の例題の場合、合計点数という同じ条件の一定点数の以上未満を判定しています。そして、例題の性質上「合格/補習/不合格」が2つ表示されることはありえない状態です。
処理手順をもっと単純に考えてみましょう。



このように条件式1が満たされた場合、下の条件式を判定する必要がなくなります。
この様な場合のif文の記述すると以下の様になります。

```
if(条件式1){
    //処理1
} else if(条件式2){
    // 処理2
} else {
    // 処理3
}
```

- `if(条件)`
  条件を満たすと、下記に記載している`else if`、`else`の条件式はスキップされます。
- `else if(条件)`
  `if(条件)`の判定が`false`(条件を満たせなかった)の場合、上から順に`else if`を判定していき条件を満たすと、それ以下の条件式はスキップします。
- `else`
  上記の条件がすべて満たせなかった場合、必ず実行されます。

```
public class Main {
    public static void main(String[] args) {
        // 試験の点数によって合格/補習/不合格かを判定するコード
        // 国語の点数
        int kokugo = 80;
        // 数学の点数
        int sugaku = 50;
        // 英語の点数
        int eigo = 60;
```

```
        // 合計点
        int goukei = kokugo + sugaku + eigo;

        // 条件式
        // 240点以上なら合格
        // 150点~240点未満なら補習
        // 150点未満なら封合格
        if(goukei >= 240) { // 条件式1
            System.out.println("合格");
        } else if(goukei >= 150) { // 条件式2
            System.out.println("補習");
        } else { // 条件式3
            System.out.println("不合格");
        }

    }
}
```

上記のコードの条件式を見るとgoukeiは190点となってます。
まず、制御構造の順次により条件式1, 2, 3の順番で判定が行われます。
条件式1ではgoukeiが190 >= 240(240点以上)を満たしていない、つまりfalseの判定になるため、{}の処理はスキップされます。
次に条件式2の190 >= 150(150点以上)は満たしている、つまりtrueの判定になるため、{}の処理が実行されます。
そして、条件式2が成立したことによりelseの条件式3はスキップされます。