Monaca

プログラミングの基本・応用
（ダイジェスト版）

アシアル株式会社
アシアル情報教育研究所
岡本 雄樹

- **制御構造とフローチャート**
  - draw.ioによるフローチャートの作図方法
  - 分岐と繰り返し
- **乱数**
- **WebAPI**

基本的プログラム

制御構造

■ 制御構造とは

- 手続き型のプログラミングで使用する処理の流れです。
- 順次・分岐・繰り返しの3つ基本的な制御構造を組み合わせることで、アルゴリズムを実現できます。
  - 実際には、変数なども必要です。

■ 順次：処理を上から下に逐次実行します

■ 分岐：条件に応じて処理を分岐します

■ 繰り返し：条件に応じて処理を繰り返し実行します

フローチャート

■ フローチャート(流れ図)

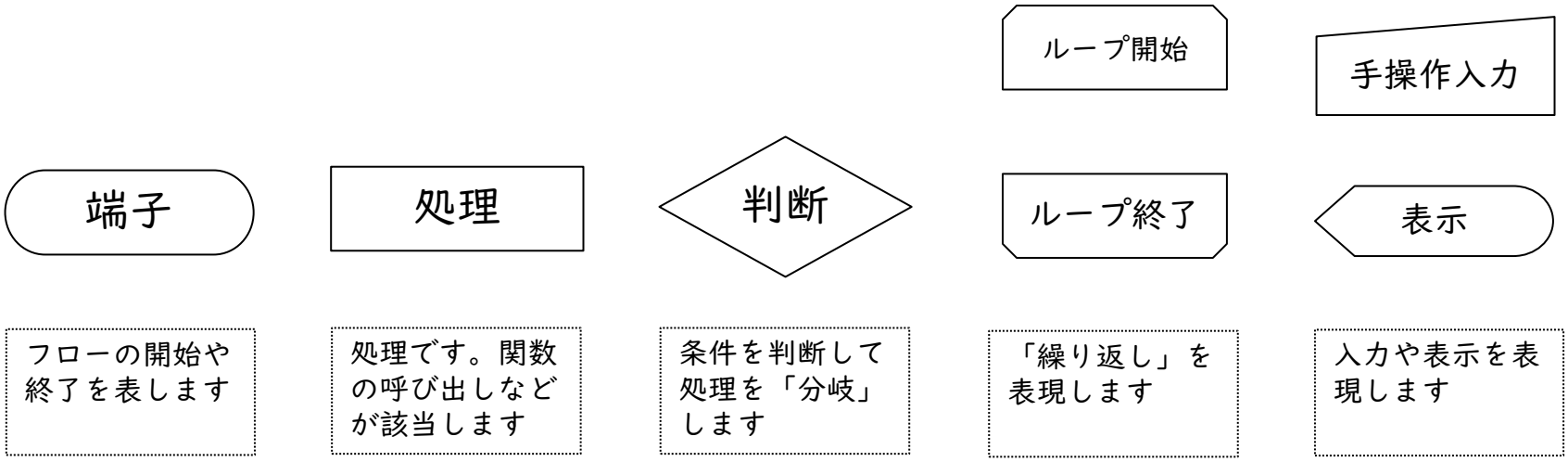
- 制御構造はフローチャートで図式化できます
  - 基本的には上から下へ、左から右へ流れていきます
  - 元々は工学や経営のために誕生したツールです
  - 日本ではJISで規格化もされています
- 高級言語が登場するもはるか昔に作られた図法のため、表現力には限界があります。
  - そのため、実際のプログラミングではあまり使われていません。

■ フローチャートの書き方(ツール編)

- 図は専用のツールで書くことをお勧めします
  - 図形と線が接続できるため
  - 特にプログラミング用の複雑な図は専用の作図ツールがあると楽です
- フローチャートは比較的簡単です
  - draw.ioのような汎用的な作図ツールでも十分、描けます
  - 簡単な図ならプレゼンテーションソフトでも可能

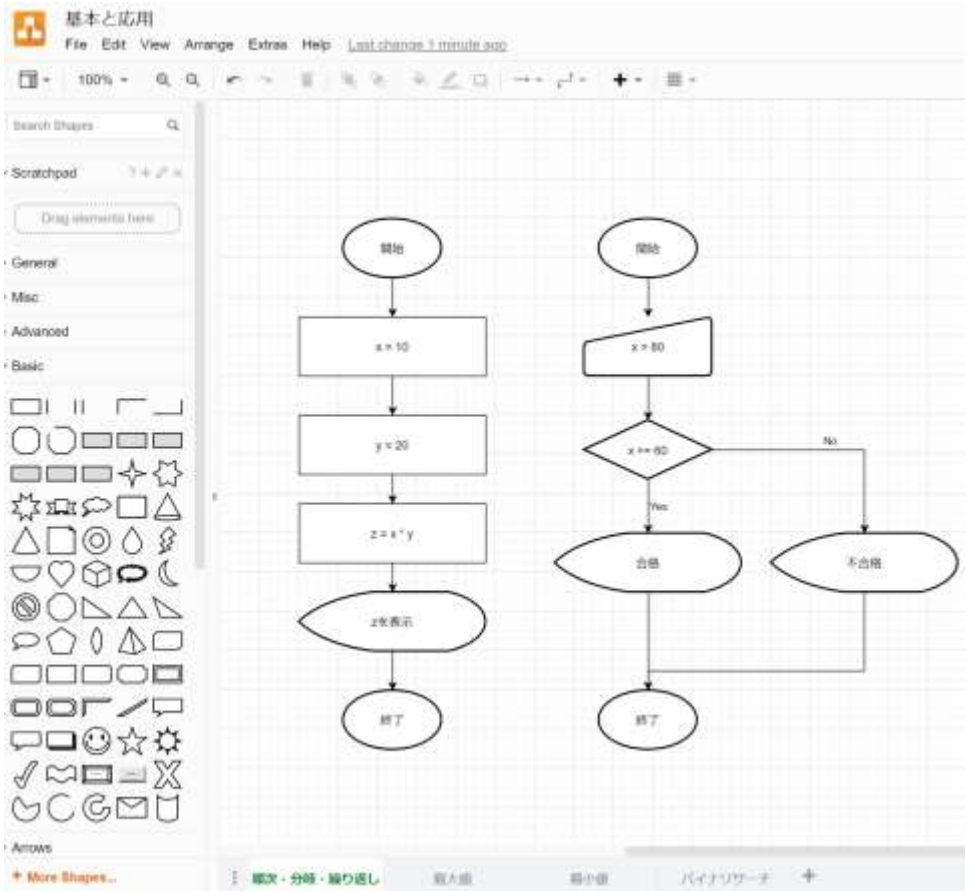
■ フローチャートの書き方(記号編)

- 主要な記号を紹介します
- 実は「判断」でループを表現することも可能です
- 入力や表示を全部「処理」記号で表しても理解は可能です



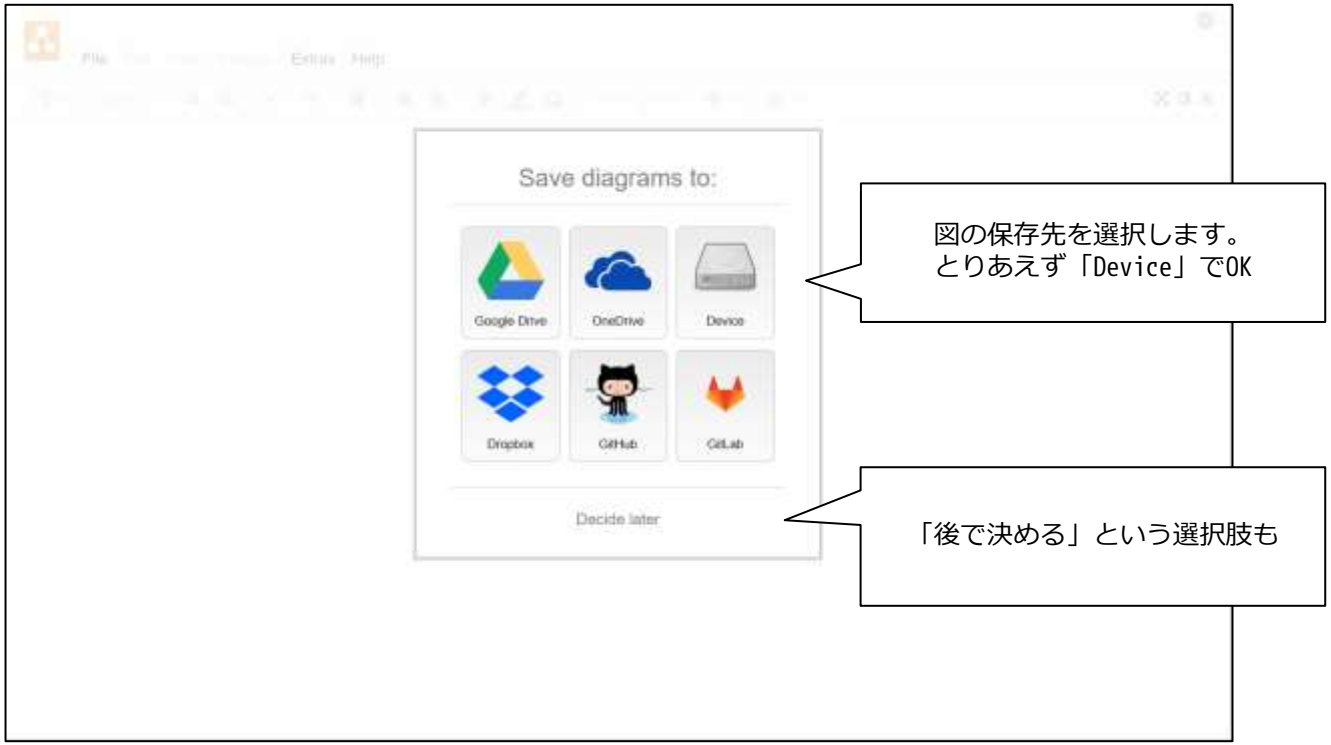
■ draw.ioによる作図入門

- 作図ツールを使うと部品や線の配置がとても楽です
- 研修用教材で紹介されているdraw.ioを紹介します
- ブラウザ上で動作します
- 会員登録・ログイン不要です



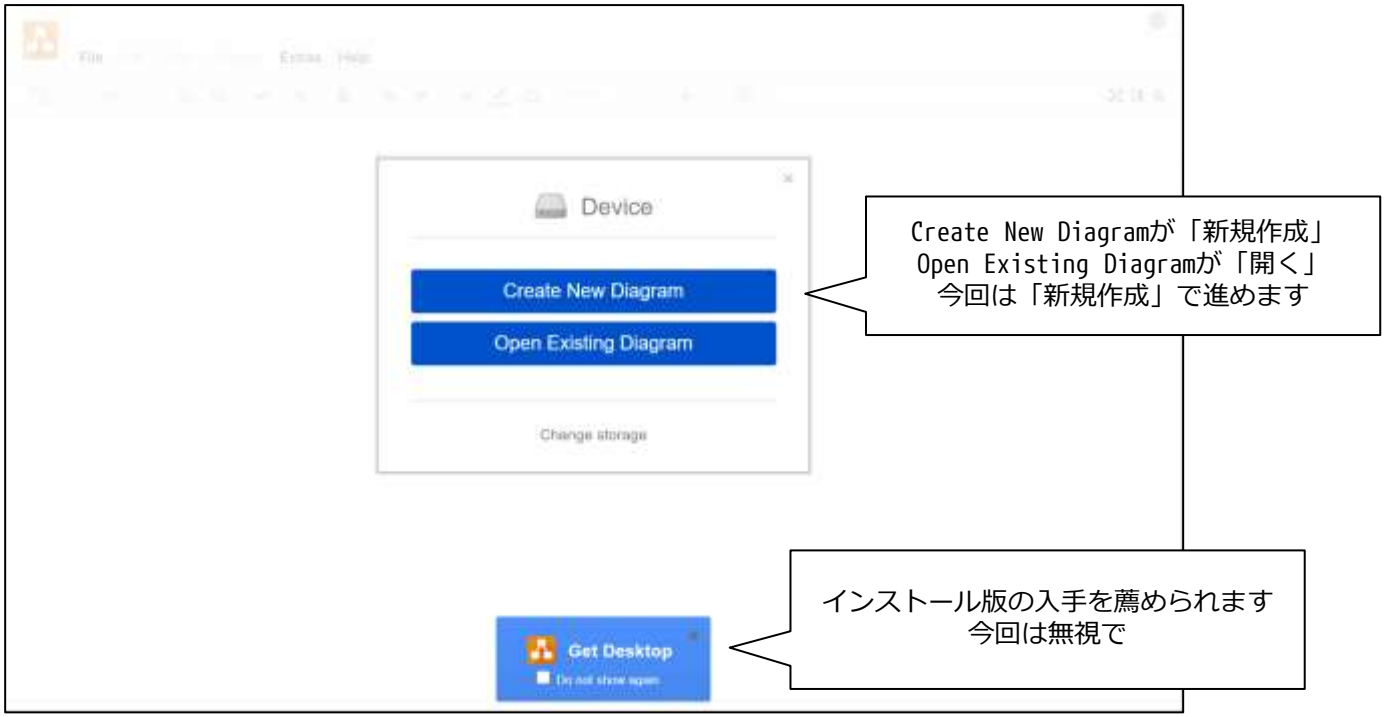
■ draw.ioの起動

- https://app.diagrams.net/ にアクセス
  - 旧URLは https://draw.io でした



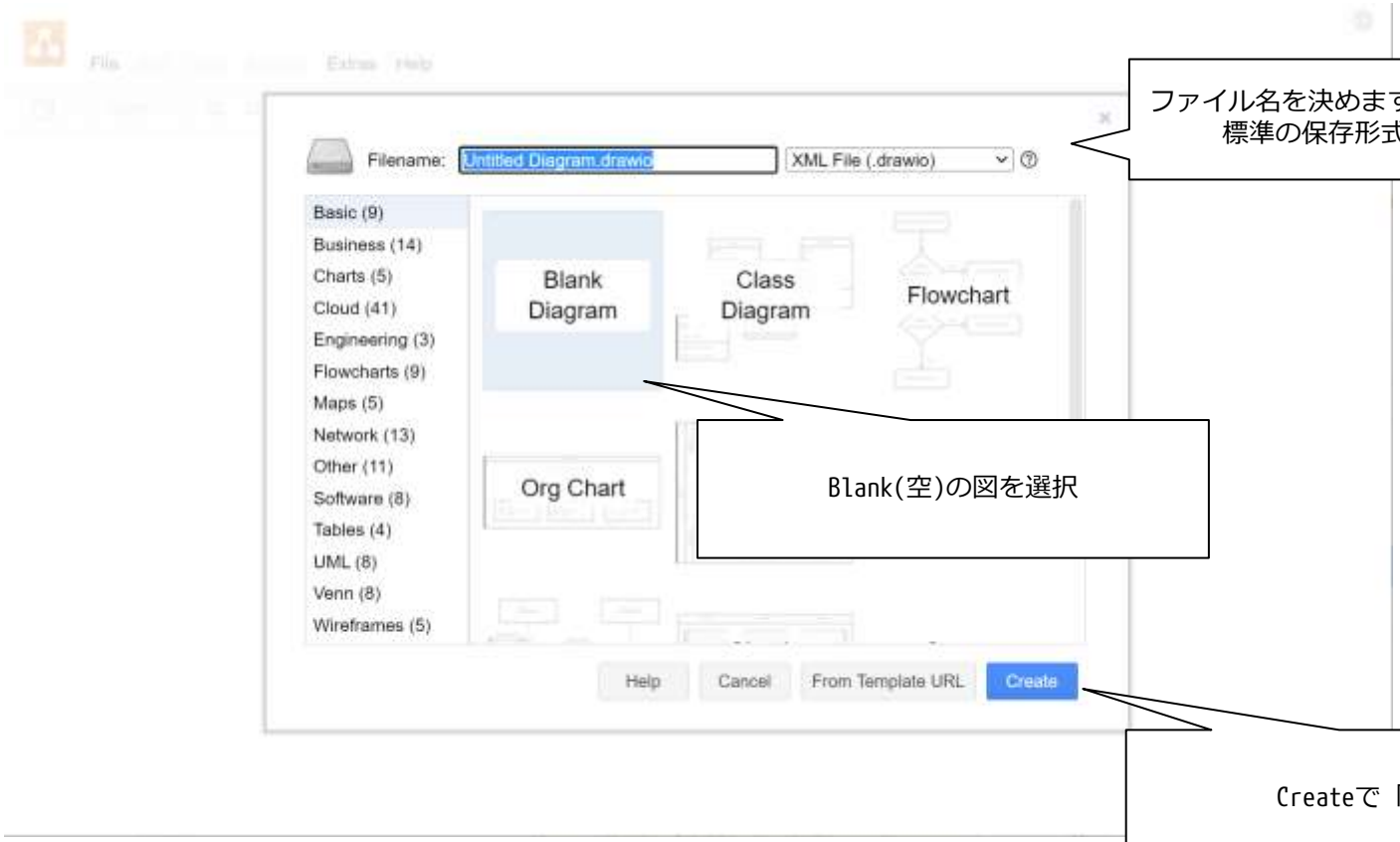
■ draw.ioの起動

└ 「新規作成」か既存ファイルを「開く」のか選択します



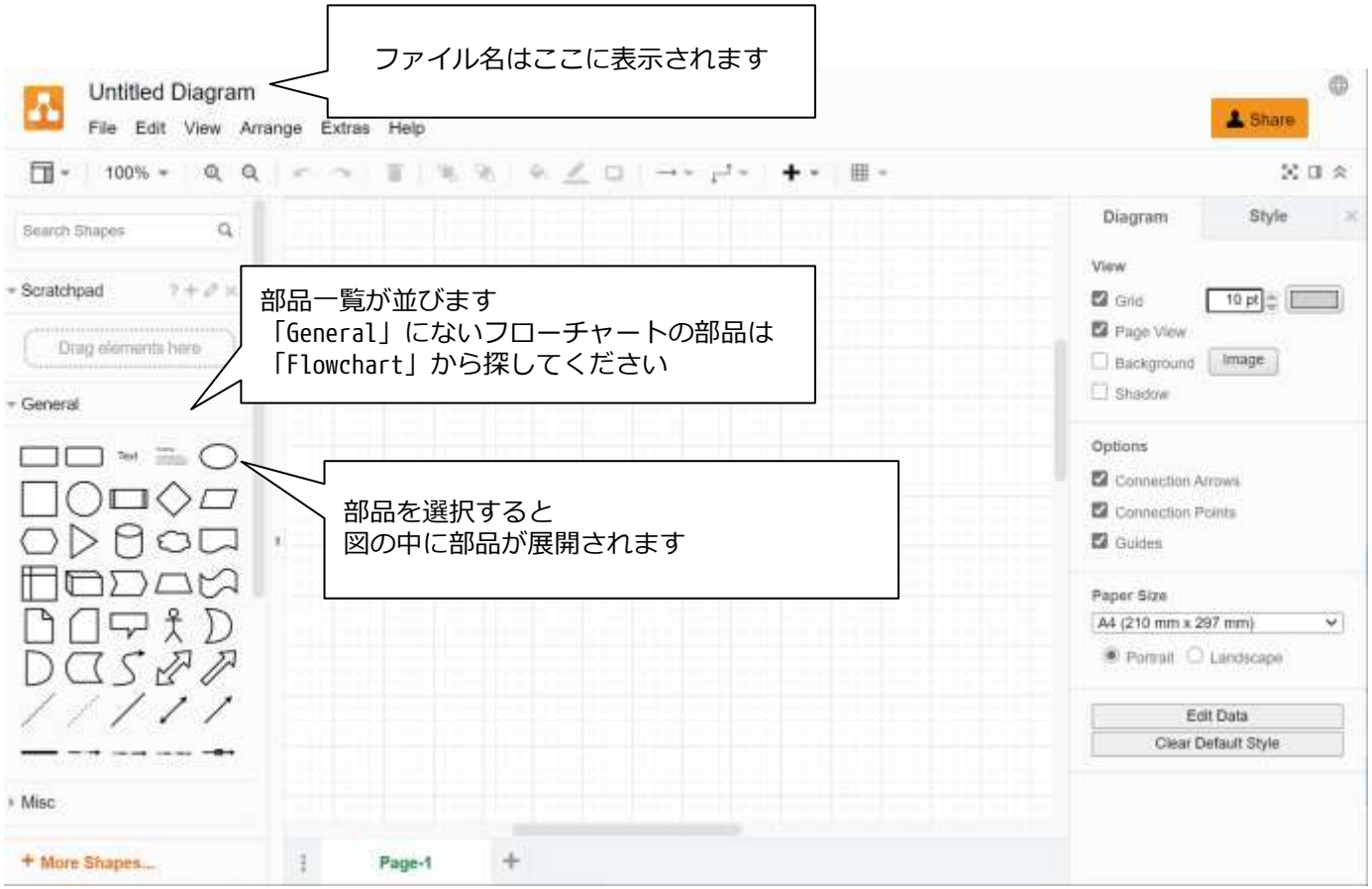
■ draw.ioの起動

└ ファイル名とテンプレートを選んで新規作成を行います



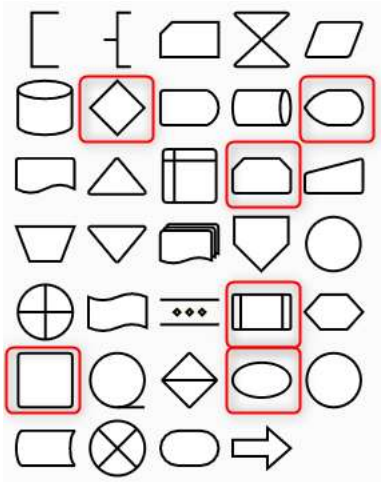
■ draw.ioの起動

└ 起動したらまずはフローチャート用の部品を探しましょう

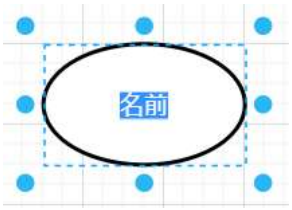


■ draw.ioの起動

- フローチャート用の部品は『Flowchart』としてまとまっています
  - 利用頻度の高い部品を□で囲んでおきました。
- 部品は図に展開後、ラベル名を付けることができます
  - ダブルクリックもしくはF12キーで編集可能



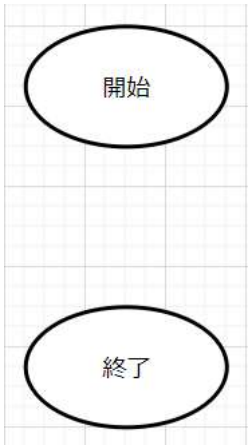
フローチャートの部品



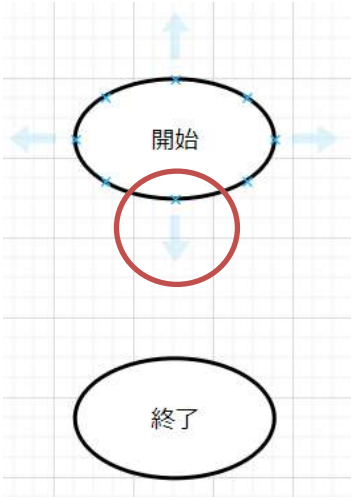
ラベル名の変更

■ 【実習】draw.ioの起動と基本操作

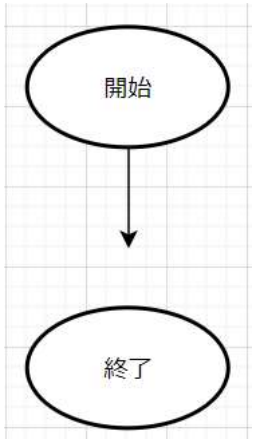
- 部品同士を線で結んでみましょう
  - 端子を二つ用意して線で結びます
  - ラベル名も変えてみましょう



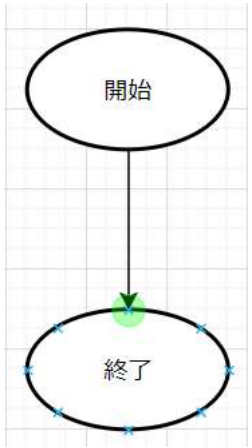
①部品を二つ用意します



②マウスカーソルをホバーさせ、矢印をクリック



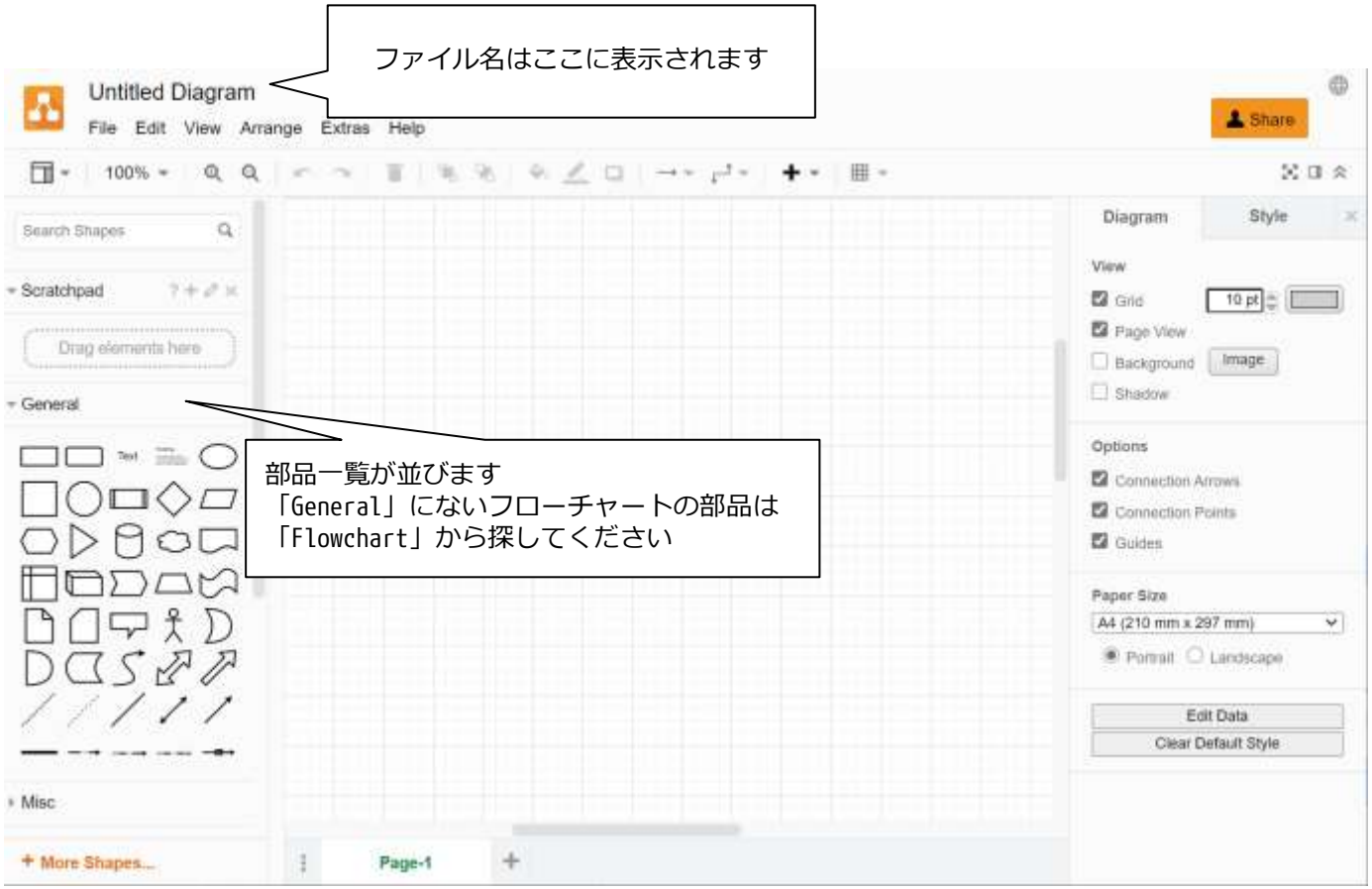
③部品から線が出ます。



④矢印の先を部品に接続して線を接続します。

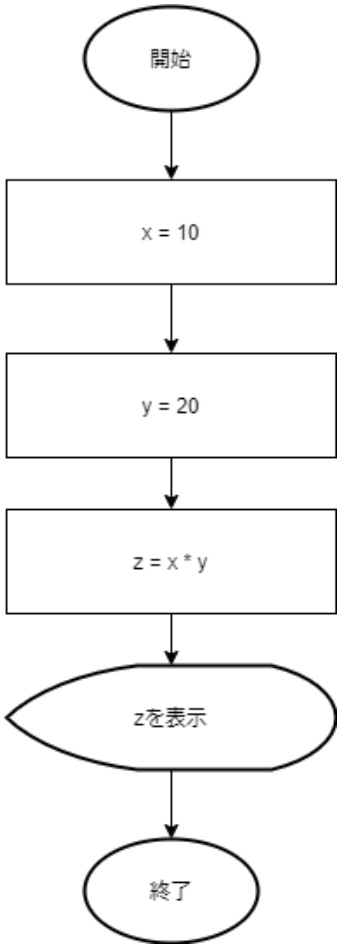
■ draw.ioの起動

└ 起動したらまずはフローチャート用の部品を探しましょう



■ 順次構造の作成

└ 図のような順次構造を作図してみましょう



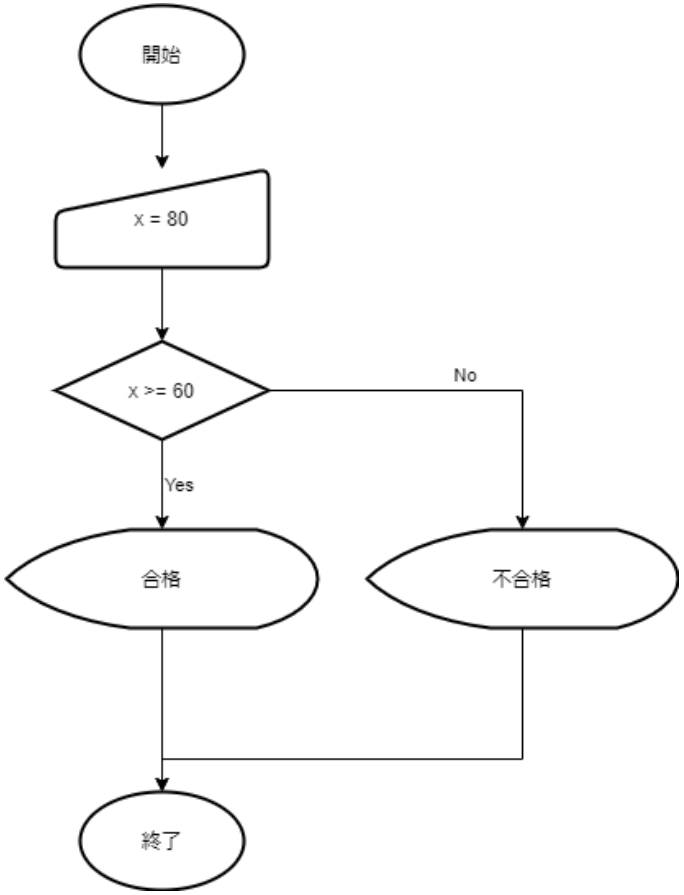
サンプルコード

```
x = 10;
y = 20;
z = x * y;

document.write(z);
```

■ 分岐の作成

└ 図のような分岐構造を作図してみましょう



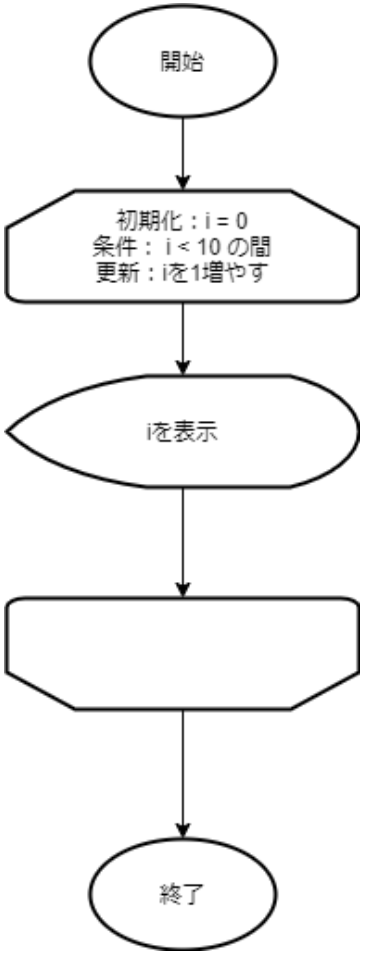
サンプルコード

```
x = 80;

if (x >= 60) {
    document.write("合格");
} else {
    document.write("不合格");
}
```

■ 繰り返しの作成

└ 図のような繰り返し構造を作図してみましょう



サンプルコード

```
for (i = 0; i < 10; i++) {
    document.write(i);
}
```

実行結果

```
0 1 2 3 4 5 6 7 8 9
```

解説

for文で「更新」が実施されるのは
ループ終端のタイミングです。
iの値が10になった次の回のループは条件が不一致となるため実行されず終了となります。

■ 変数：カウンタ変数

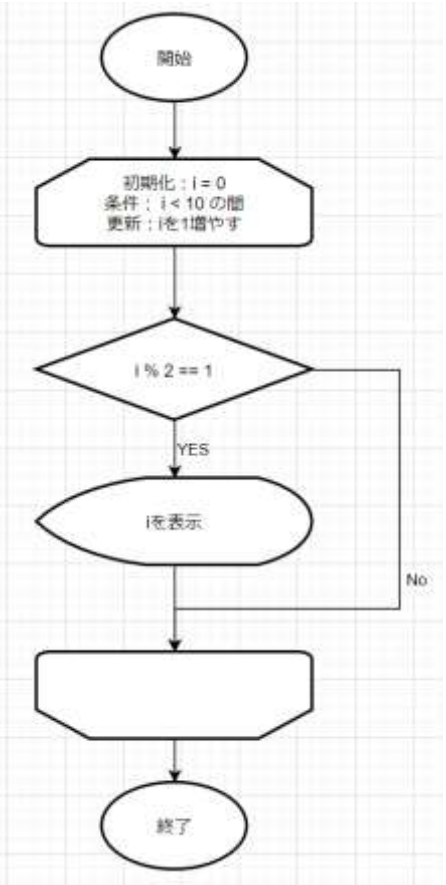
- 繰り返し処理で登場する「i」はカウンタ変数と呼びます
  - 歴史的経緯により「i」が使われています
  - 二重ループを書く場合、「i」とともに「j」が使われます
    - jの次は「k」です。
    - i, j, kが使われるのは慣習に過ぎません
    - カウンタ変数名は自由に付けても構いません。

■ 変数：変数のインクリメントとデクリメント

- カウンタ変数などで値を「1」増減したいケースが多々あります
  - 増やすことをインクリメント減らすことをデクリメントと呼びます
- 「i = i + 1」と記述できますが、面倒です
- そのため「i++」と記述できるようになっています
- インクリメント表記はプログラミングで多用するため、馴れてくれば嫌でも自然に頭の中に入ってくるようになります

■ ループでカウンタ変数が奇数の時だけ表示する

└ 図のような繰り返し構造を作図してみましょう



サンプルコード

```
for (i = 0; i < 10; i++) {
    if (i % 2 == 1) {
        document.write(i);
    }
}
```

実行結果

```
1 3 5 7 9
```

解説

整数を2で割ったときの余りが1のとき、整数は奇数です。if文で条件判定し、奇数の時だけiを表示させています。

■ 変数の宣言と型の話

- プログラミングで変数を使うときには「宣言」が必要です
  - スクリプト言語では宣言不要なものも多いです
- 変数の宣言ではデータ型も決める必要があります
  - が、スクリプト言語では型が動的なものも多いです
- スクリプト言語とは？
  - 簡易的な記述を行えるプログラミング言語全般を指す言葉です
    - JavaScript/VBA/Pythonはスクリプト言語に分類されます
  - 簡易的な目的で発明された言語も、発展すると高度な記述が行えるように進化する場合があります。
- 動的に型が決まることのデメリット
  - 意図しない型変換で不具合を作り込む可能性があります

JavaScriptにおける変数の宣言

- 数行程度の短いプログラムなら、あまり影響はありません
  - 文科省教材のサンプルコードでは、何も付けていません
  - 変数の影響範囲がプログラム全体に及ぶか一部に限定されるか、が変わります
    - スコープと呼びます、スコープの話は応用でします
- 色々なプログラムを組み合わせる場合は注意して下さい
  - 事故が少ないのは「let」です

何も付けない	何も付けずに宣言した変数はグローバルスコープとなり、関数の中からも読み書きが行えます。
var	グローバルスコープや関数スコープの変数を宣言します。
let	ローカルスコープやブロックスコープの変数を宣言します。
const	値を代入したら変更できません。いわゆる定数になります。

四則演算子

演算子	概要	条件式の例	結果
+	数値の加算	1 + 1	2
+	文字列の結合	"Hello" + "World"	"HelloWorld"
-	数値の減産	2 - 1	1
*	数値の乗算	2 * 2	4
/	数値の除算	10 / 2	5
%	数値の乗余算	9 % 2	1

複合代入演算子

演算子	概要	使用例	結果（a の値）
+=	左辺の値に右辺の値を加算したものを代入	a = 1; a += 2;	3
-=	左辺の値から右辺の値を減算したものを代入	a = 5; a -= 2;	3
*=	左辺の値に右辺の値を乗算したものを代入	a = 3; a *= 2;	6
/=	左辺の値を右辺の値で除算したものを代入	a = 10; a /= 2;	5
++	変数に1加算する（インクリメント）	a = 1; a++;	2
--	変数から1減算する（デクリメント）	a = 1; a--;	0

■ コメントの記述

- ソースコードにはコメントを記述できます
  - 1行コメントは「//」
  - 複数行にわたるブロックコメントは /* ～ */ で囲みます
- 無闇にコメントを付けるとメンテナンスが大変になるので程々に

■ 例

```
/**
 * 奇数を画面に表示するプログラム
 * バージョン 999
 */
for (i = 0; i < 10; i++) {
    // 2で割った余りが1ならば奇数なので表示する
    if (i % 2 == 1) {
        document.write(i);
    }
}
```

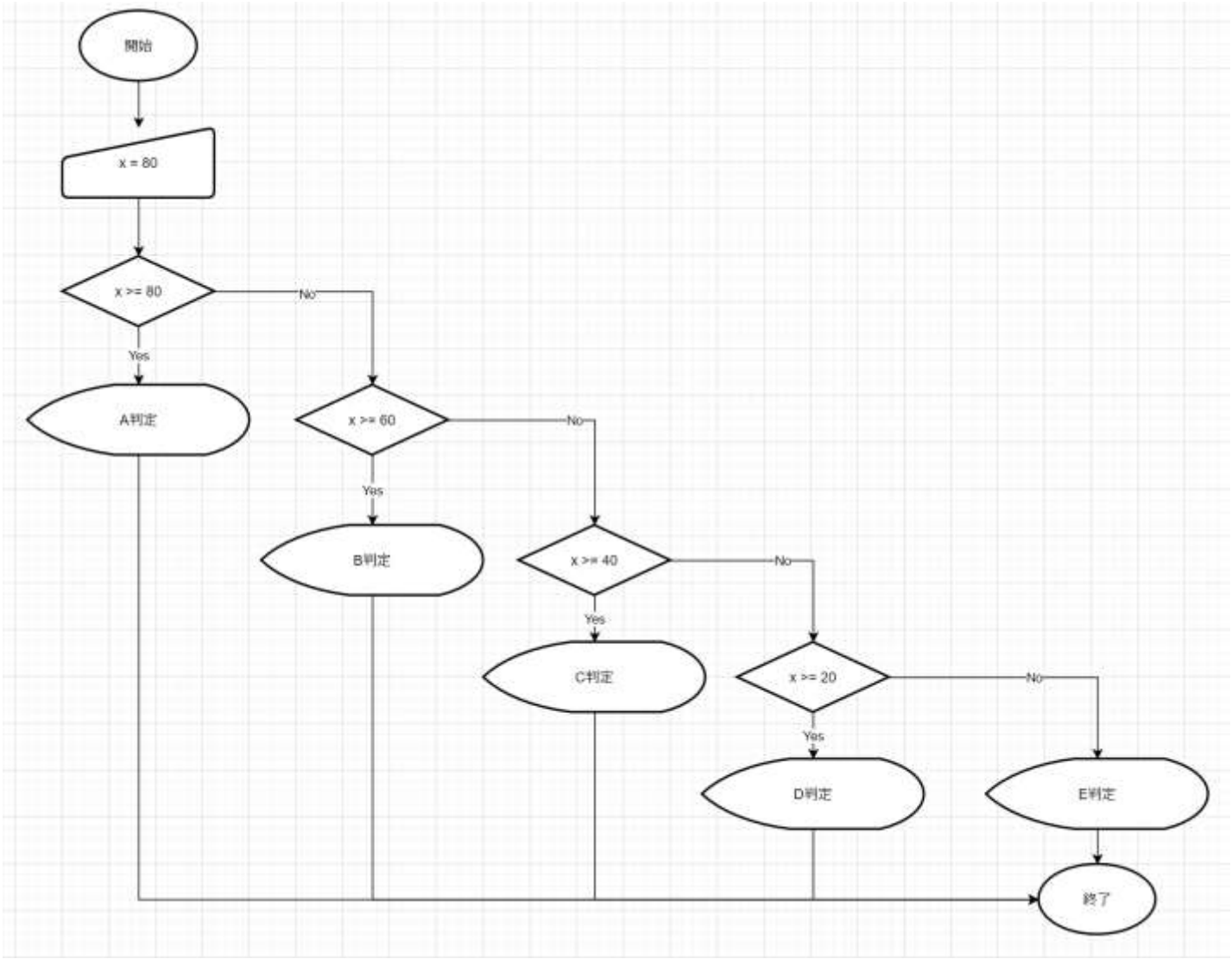
■ もっと条件分岐について学ぶ

- 多方向分岐（else if 文）
  - 1つめの条件式にあてはまらない場合に、次の条件式を試みます

■ 文法 else if 文の書き方

```
if (条件式1) {
    条件式1 が正しい場合に実行する処理
} else if(条件式2) {
    条件式2 が正しい場合に実行する処理
} else {
    条件式が正しくない場合に実行する処理
}
```

■ 実習：点数に応じてA～E判定を行うプログラムの作成

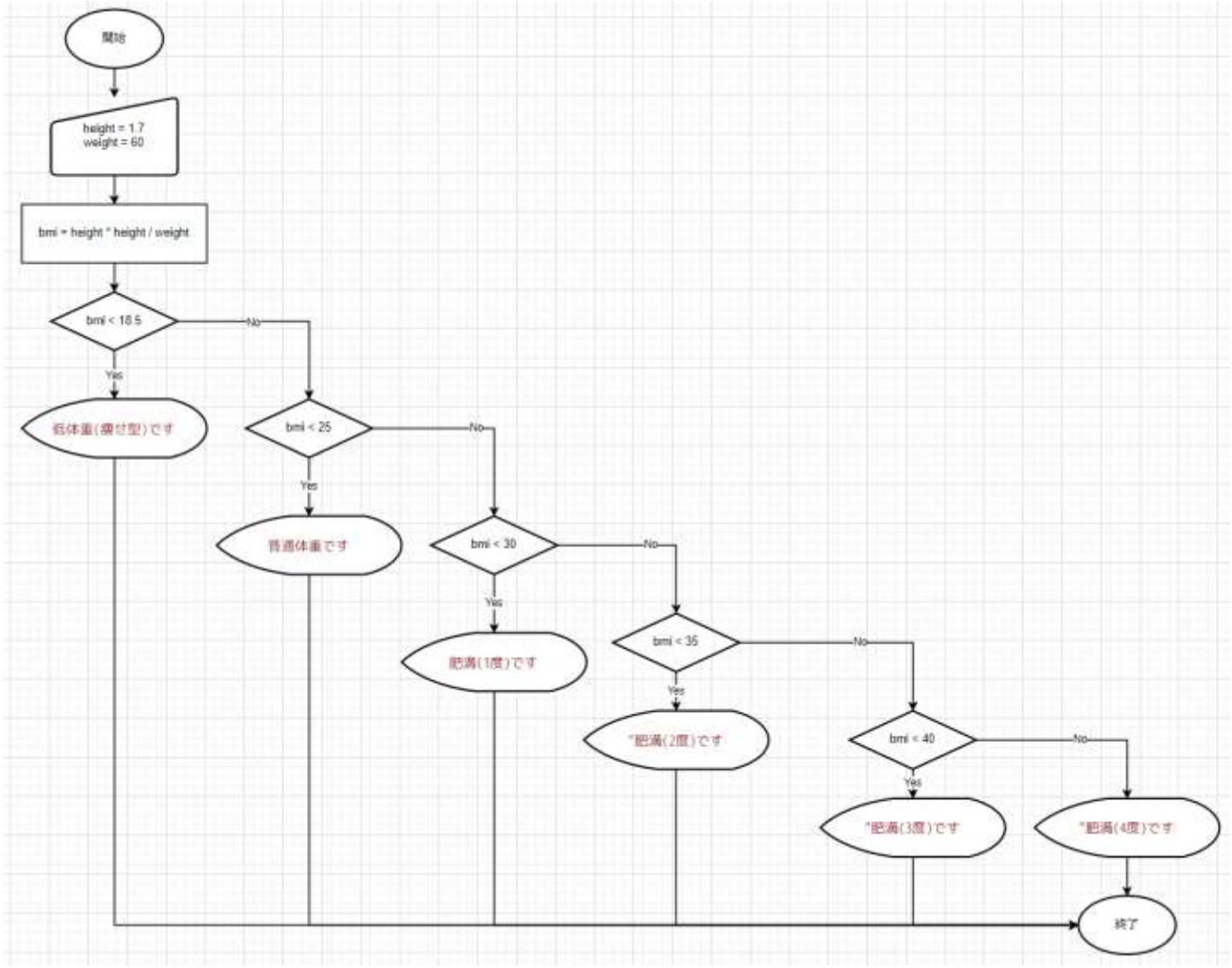


■ コード例

```
x = 80;

if (x >= 80) {
    document.write("A");
} else if (x >= 60) {
    document.write("B");
} else if (x >= 40) {
    document.write("C");
} else if (x >= 20) {
    document.write("D");
} else {
    document.write("E");
}
```

■ 実習：身長と体重に応じてBMI判定を行うプログラムの作成



■ コード例

```
height = 1.7; // m
weight = 60; // Kg
bmi = weight / (height * height);

if(bmi < 18.5) {
    document.write("低体重(痩せ型)です。");
} else if (bmi < 25) {
    document.write("普通体重です。");
} else if (bmi < 30) {
    document.write("肥満(1度)です。");
} else if (bmi < 35) {
    document.write("肥満(2度)です。");
} else if (bmi < 40) {
    document.write("肥満(3度)です。");
} else {
    document.write("肥満(4度)です。");
}
```

■ もっと繰り返しについて学ぶ

- while文
  - 条件式が一致する限りずっと処理を繰り返す
  - 注意しないと無限ループになります

■ 文法

```
while (条件式) {
    条件式 が正しい場合に実行する処理
}
```

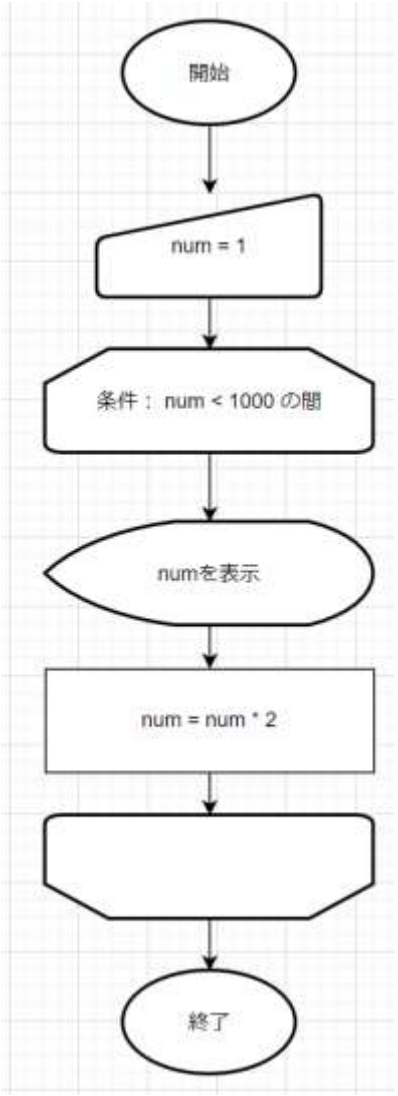
while文とfor文の比較

- 以下のfor文とwhile文は等価です
- 1行に初期化式と更新式をまとめられるのがfor文

```
for (i = 0; i < 10; i++) {
    document.write(i);
}
```

```
i = 0;
while (i < 10) {
    i++;
    document.write(i);
}
```

■ while文の例




```
num = 1;
while (num < 1000) {
    document.write(num,"<br>");
    num = num * 2;
}
```

■ 関数とは

- 関数は処理をひとまとめにしたものです
- 応用が利くよう、引数と戻り値の仕組みがあります
  - 『引数』を受け取ることができます
  - 処理結果を『戻り値』として返すことができます
  - 『引数』と『戻り値』は省略可能です
- 関数は自分で作る(定義する)こともできます

■ ブラウザ上で呼び出せる関数の例

- 引数のない関数や戻り値のない関数も存在します
- alertは画面にポップアップを行いますが、戻り値は特にありません
- 一方、promptは入力を求める文字列のポップアップを出しつつ、戻り値として受け取った文字列を返します。

関数	引数	戻り値
Math.random()	-	ランダムな値を返す
Math.floor()	数値	小数点を切り捨てて整数を返す
Math.ceil()	数値	小数点を切り上げて整数を返す
document.write()	文字列	-
alert()	文字列	-
prompt()	文字列	文字列

■ 乱数とは

- 一様乱数
  - 一定範囲内の数値が同じ確率で現れるような乱数です
- なお、計算では完全な乱数を生成することはできません
  - 真の乱数を作るためには、外部から乱数を生成するための種(シード)としてエントロピーを取得する必要があります。
    - エントロピーはコンピューターのハードウェアからも取得できます。

■ 乱数を生成する命令

```
Math.random();
```

■ 記述例:0～0,99999...の範囲の乱数を出力

```
r = Math.random();
console.log(r);
```

■ 記述例:0～9の整数を得る

```
r = Math.random();
r = Math.floor(r * 10); // parseInt()でもOK
```

■ 記述例:1～10の整数を得る

```
r = Math.random();
r = Math.floor(r * 10) + 1;
```

■ 10%の確率であたりがでるプログラムの例

```
a = 0;
r = Math.floor( 10 * Math.random() );
if (a == r) {
    console.log("あたり");
} else {
   console.log("はずれ");
}
```

■ くじ引き風にカスタムしたプログラムの例

```
a = promp("0～9の数字を入力してください");
a = parseInt(a);
r = Math.floor( 10 * Math.random() );
console.log("正解は", r, "選んだのは", a);
if (a == r) {
    console.log("あたり");
} else {
   console.log("はずれ");
}
```

■ 1〜6の範囲の乱数を求める関数

```
function random() {
    value = Math.floor(Math.random() * 6) + 1;
    return value;
}

random = random();
console.log(random);
```

■ 任意の範囲の乱数を求める関数

└ 更に、引数を省略した場合1～6の範囲で返す

```
function random(min = 1, max = 6) {
    length = max - min + 1;
    value = Math.floor(Math.random() * length) + min;
    return value;
}

random = random();
console.log(random);
```

WebAPI （郵便番号）

■ APIとは

- API ≒ 関数
  - OSやブラウザなどの機能をプログラミング言語で呼び出すときには、APIを経由して利用しています
  - 例：ファイルの保存など
- ブラウザにもAPIがあります(HTML5 API)
  - JavaScriptからHTML5 APIを呼び出すことで、位置情報の取得などが行えます

WebAPI

- Web経由でクラウドサービスの機能の一部を利用できるAPIをWebAPIと呼びます。以下、例です。
  - 郵便番号を元に住所を取得
  - 緯度経度を元に天気予報を取得
  - 画像を元に商品名を判定
- WebAPIの呼び出し
  - JavaScriptの「fetch」命令を使用すれば、簡単なWebAPIは簡単に利用できます
    - 結果はJSON形式などで取得できます
      - なお、WebAPIやJSONの話は4章でも登場します。

■ 郵便番号 WebAPIの例

└ https://api.anko.education/zipcode/?zipcode=1130033

リクエスト例

タイプ	値
URL	https://api.anko.education/zipcode/?zipcode=1130033
メソッド	GET

レスポンス

タイプ: JSON

キー	値
code	integer 郵便番号
prefcode	integer 都道府県コード
pref	String 都道府県名
city	String 市町村名
area	String 住所1

■ 郵便番号APIの呼び出しプログラム例

└ https://api.anko.education/zipcode/?zipcode=1130033

```
var url = 'https://api.anko.education/zipcode?zipcode=100-0013';
    fetch(url)
        .then(function(response) {
            return response.json();
        })
        .then(function(address) {
            document.write(address.pref + address.city + addre
        });
```

変数にWebAPIのアドレスを格納

WebAPIをfetchで呼び出し

成功結果からJSONだけを抽出

JSONの値を変数addressとして取得

JSONの値を画面に書き出し

※高等学校情報科 「情報 I 」教員研修用教材 3章 P125を参考に作成