Monaca

「情報Ⅰ」に向けたプログラミング研修会

～基本＆応用編～

アシアル株式会社
アシアル情報教育研究所
岡本 雄樹

Copyright © Asial Corporation. All Right Reserved.

はじめに

Copyright © Asial Corporation. All Rights Reserved.

自己紹介

- 名前
  - 岡本雄樹(アシアル情報教育研究所 所長)
- 著書
  - イラストでよくわかるPHP
  - WordPressプロフェッショナル養成読本
  - Monacaで学ぶはじめてのプログラミング
- メッセージ
  - 「コンピューター」「インターネット」「プログラミング」私は高校生の時にそれらと出会うことで人生が拓けました。先生方とMonacaによるアプリ開発を通じて、情報技術の活用方法や作品作りの楽しさを広めてまいります。

Copyright © Asial Corporation. All Rights Reserved.

■ 教材サイトのご案内

- あんこエデュケーション
  - https://anko.education/
- 教科情報研修資料
  - https://anko.education/joho

Copyright © Asial Corporation. All Rights Reserved.

■ プロジェクトのインポート

- 教材サイトに各種モデルのサンプルプロジェクトがあります
- Monacaでインポートして進めます
- 予めMonacaのログインと空きプロジェクト数の確保をお願いします

Copyright © Asial Corporation. All Rights Reserved.

■ 基本的プログラム

- 制御構造
- フローチャート
  - draw.ioによる作図方法
  - 分岐の作図
  - 繰り返しの作図
- 基本文法
  - 変数
  - 演算子
  - コメント
- 分岐処理
- 繰り返し処理

Copyright © Asial Corporation. All Rights Reserved.

基本的プログラム

Copyright © Asial Corporation. All Rights Reserved.

制御構造

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.

フローチャート

Copyright © Asial Corporation. All Rights Reserved.

■ 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。

Copyright © Asial Corporation. All Rights Reserved.

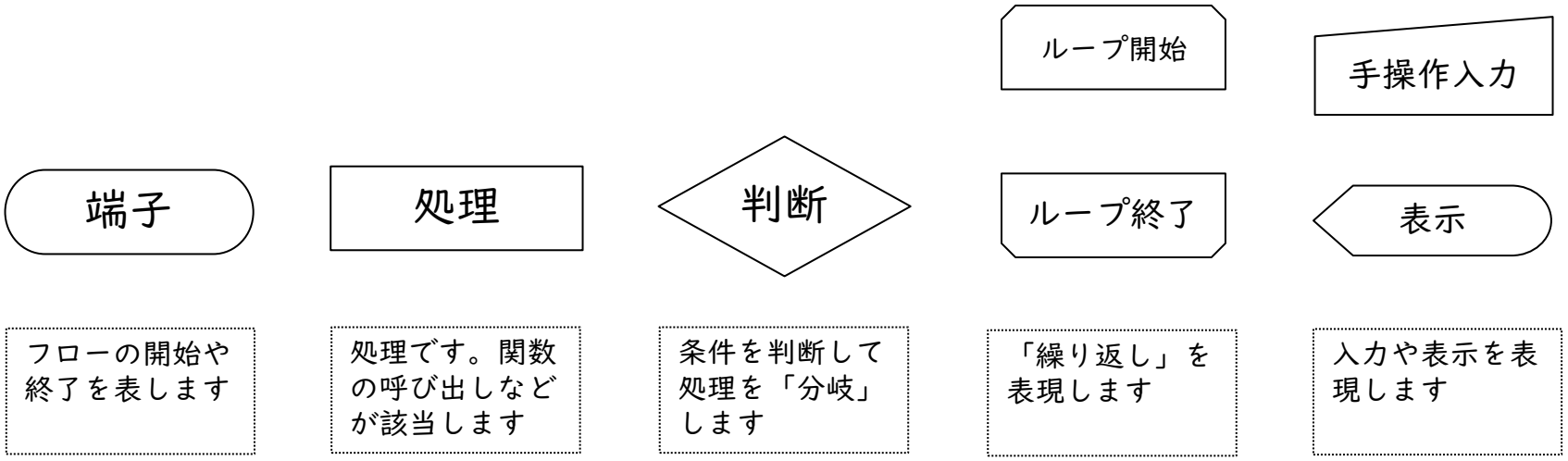
■ フローチャートの書き方(ツール編)

- 作図はソフトの利用をお勧めします
  - 図形と線の接続・移動が簡単
- フローチャートは汎用ソフトで対応可能
  - draw.ioのような汎用的な作図ツールでも十分、描けます
  - プレゼンテーションソフトにもフローチャートの図形は入っています
- プログラミング用の様々な図について
  - 専用の作図ツールも存在します

Copyright © Asial Corporation. All Rights Reserved.

■ フローチャートの書き方(記号編)

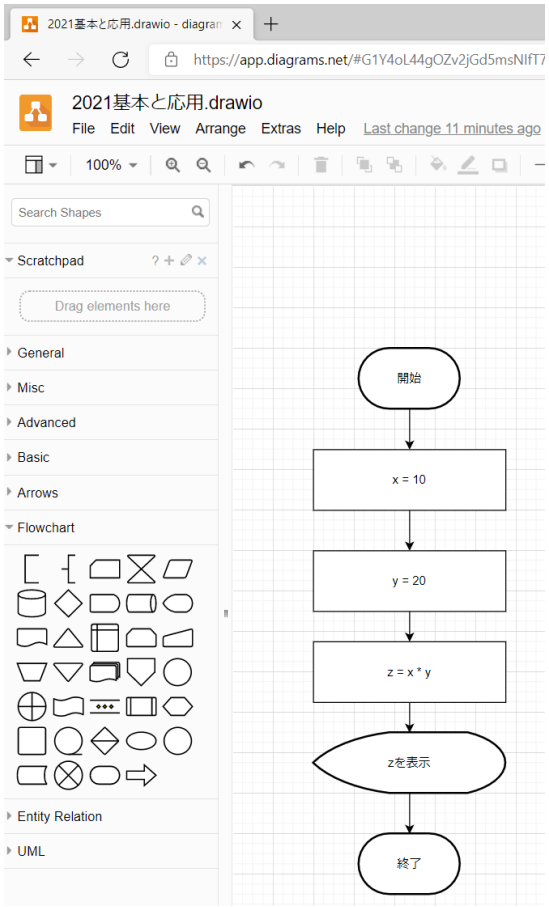
- 主要な記号を紹介します
- 実は「判断」でループを表現することも可能です
- 入力や表示を全部「処理」記号で表しても表現は可能です



Copyright © Asial Corporation. All Rights Reserved.

draw.io(diagrams.net)による作図入門

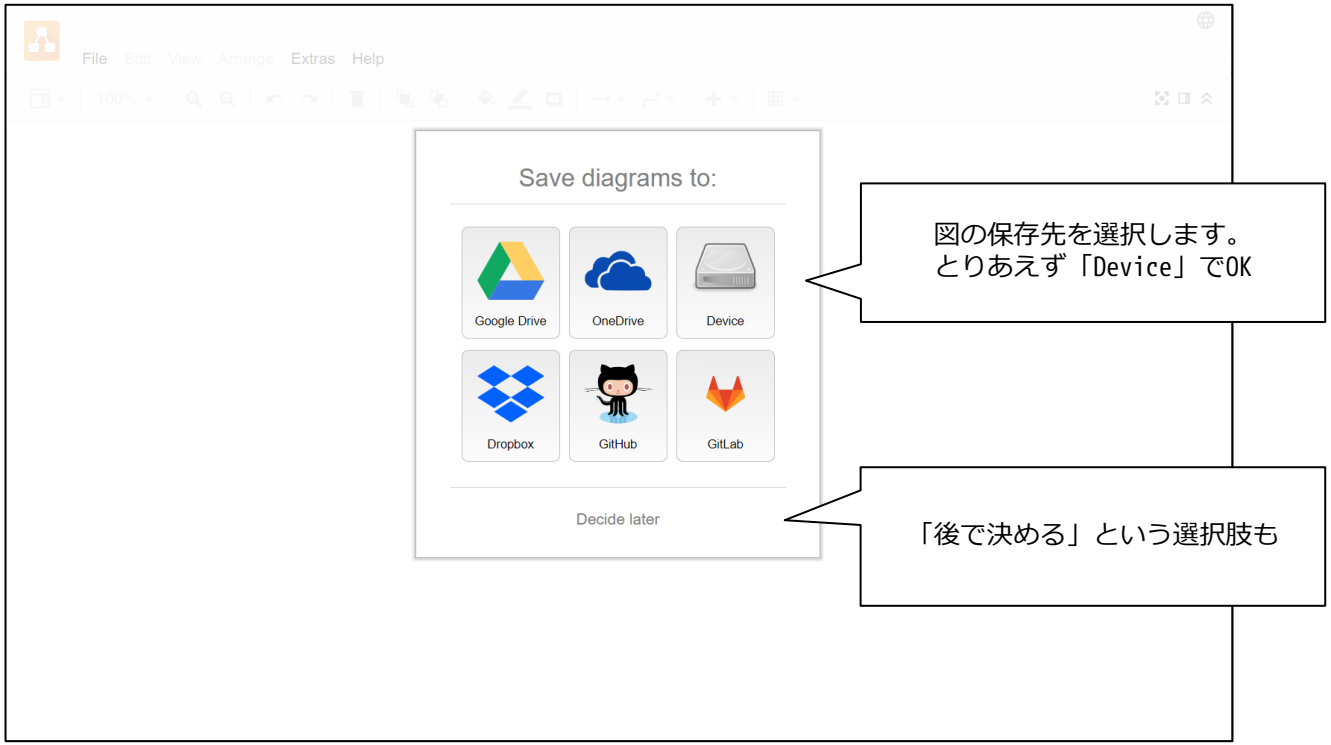
- 作図ツールを使うと部品や線の配置がとても楽です
- 研修用教材で紹介されているdraw.ioを紹介します
- ブラウザ上で動作します
- 会員登録・ログイン不要です



Copyright © Asial Corporation. All Rights Reserved.

■ draw.ioの起動

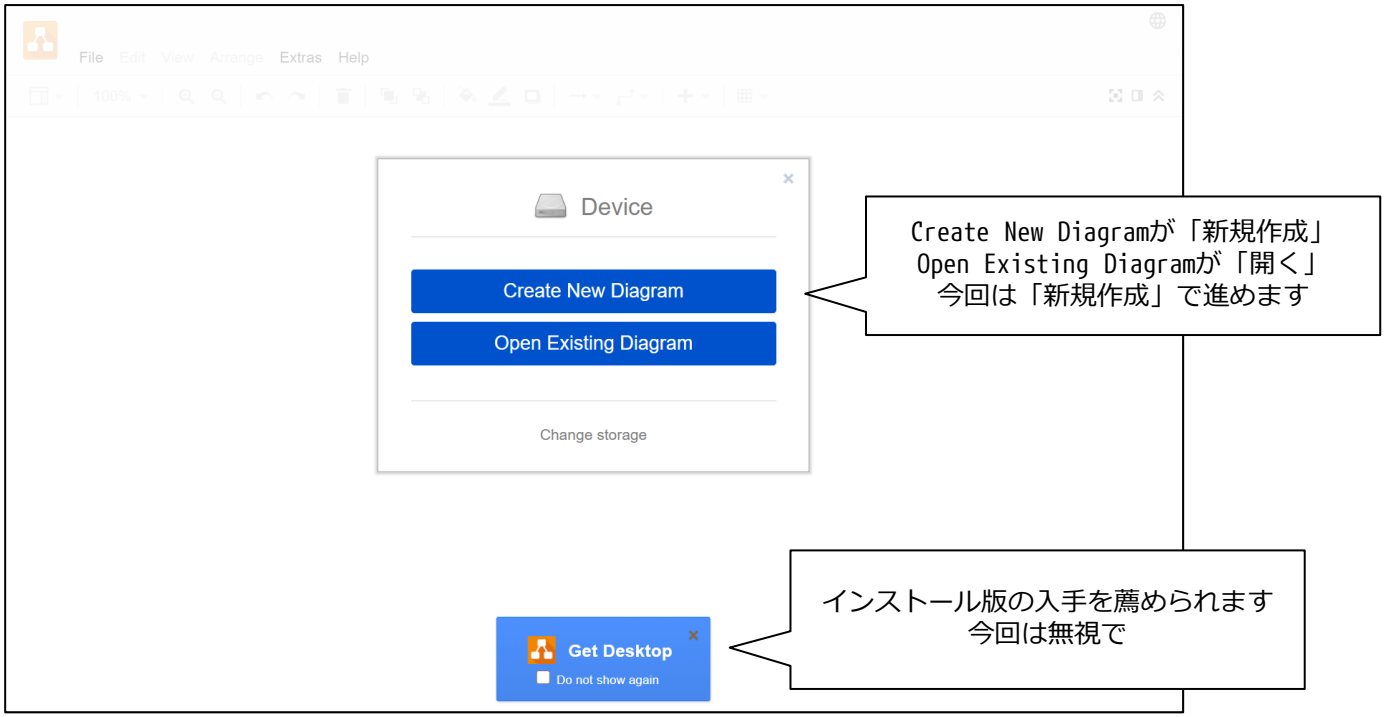
- https://app.diagrams.net/ にアクセス
  - 旧URLは https://draw.io でした



Copyright © Asial Corporation. All Rights Reserved.

draw.ioの起動

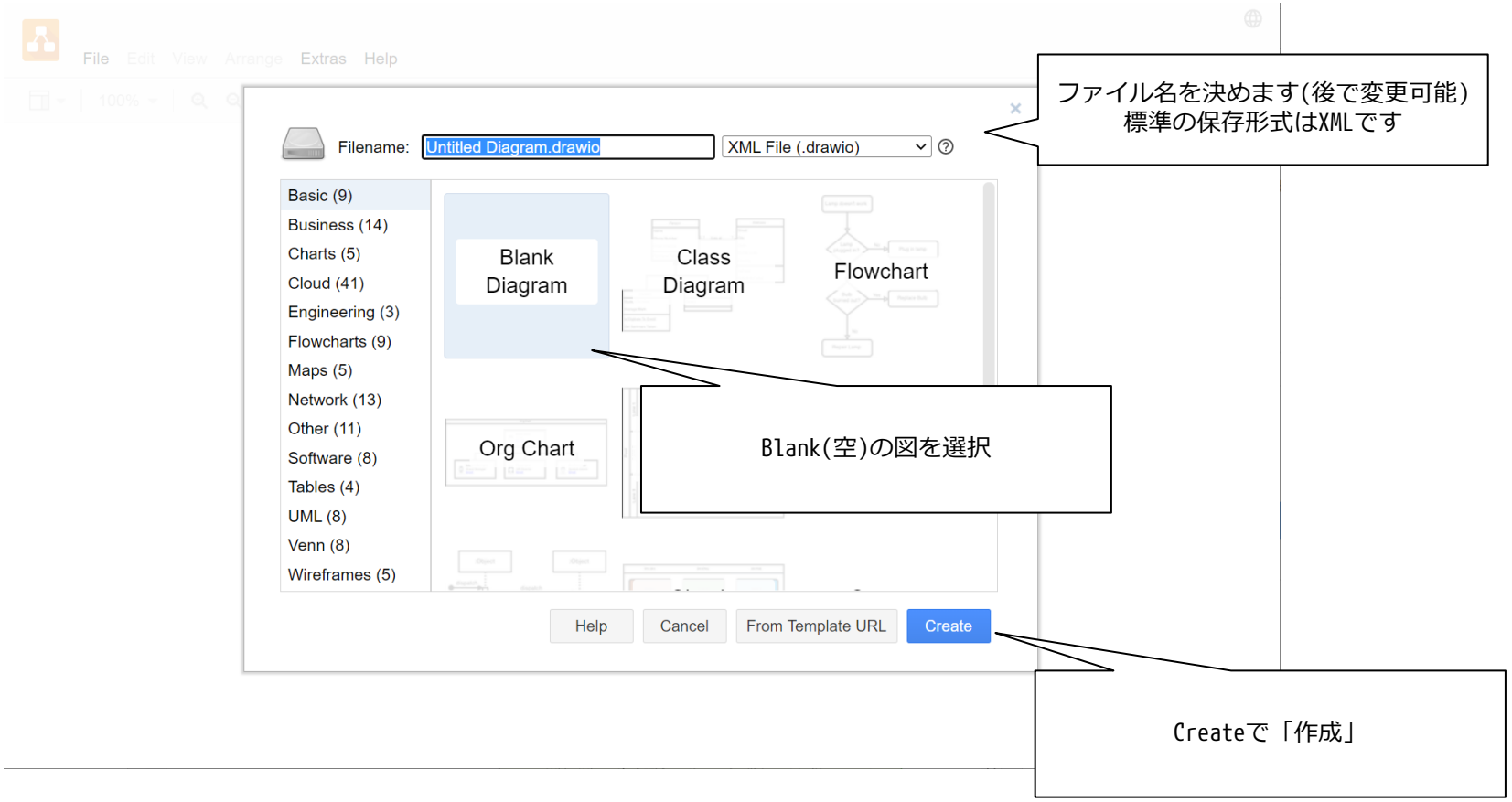
「新規作成」か既存ファイルを「開く」のか選択します



Copyright © Asial Corporation. All Rights Reserved.

draw.ioの起動

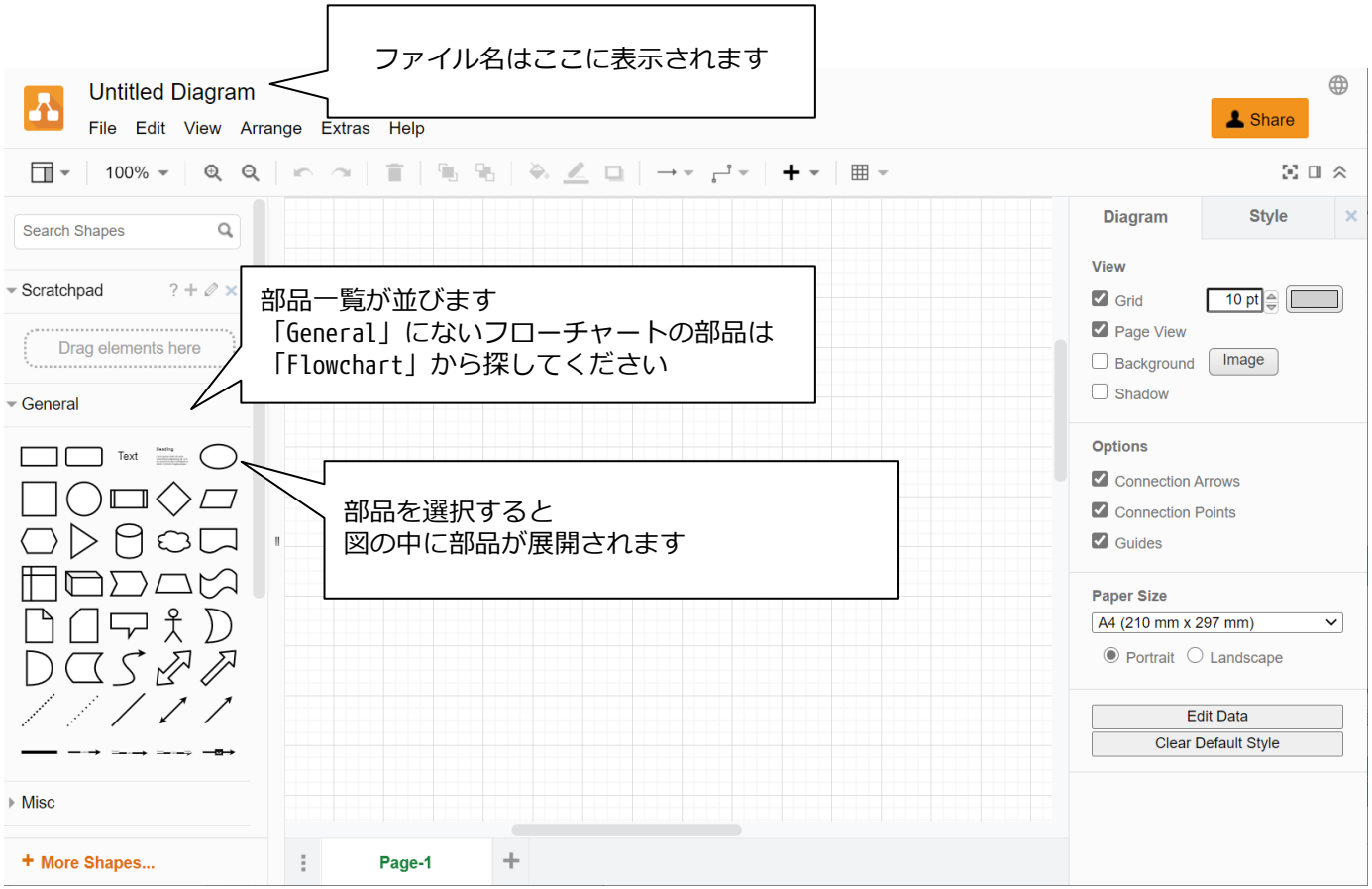
- ファイル名とテンプレートを選んで新規作成を行います



Copyright © Asial Corporation. All Rights Reserved.

draw.ioの起動

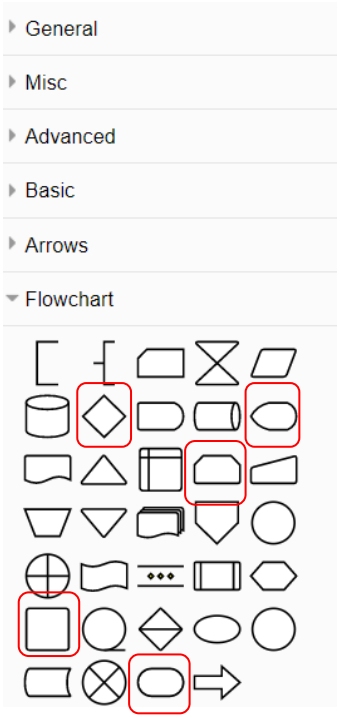
- 起動したらまずはフローチャート用の部品を探しましょう



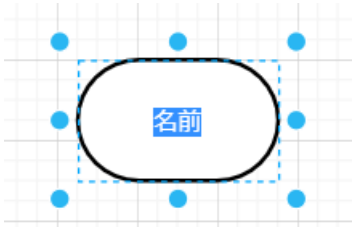
Copyright © Asial Corporation. All Rights Reserved.

■ 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
- 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『F2』キーで編集可能



フローチャートの部品

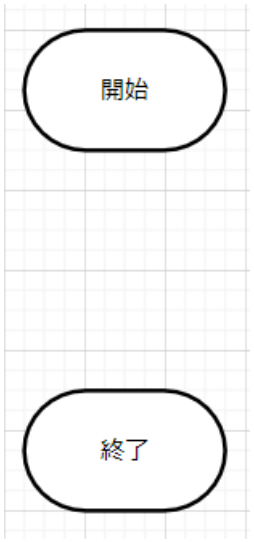


ラベル名の変更

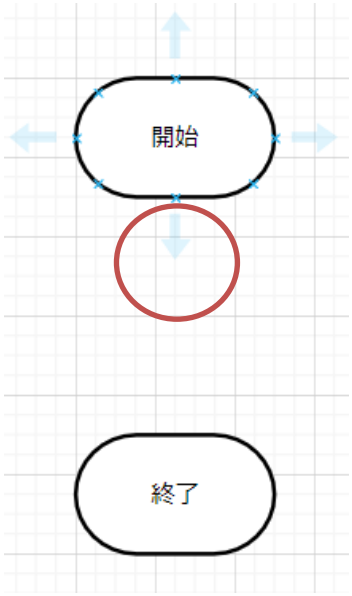
Copyright © Asial Corporation. All Rights Reserved.

■ 【実習】draw.ioの起動と基本操作

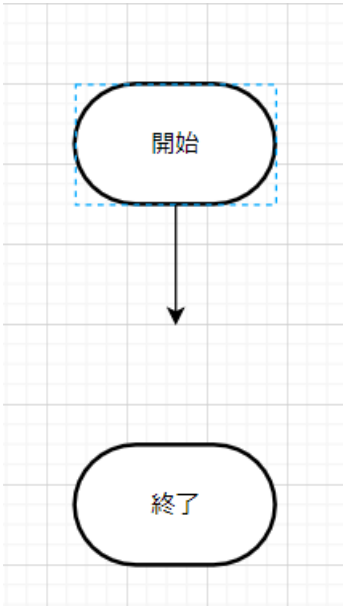
- 部品同士を線で結んでみましょう
  - 端子を二つ用意して線で結びます
  - ラベル名も変えてみましょう



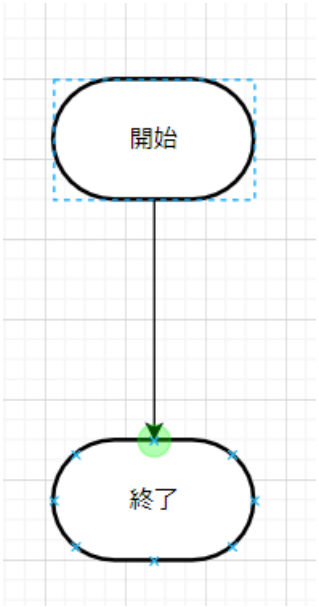
①部品を二つ用意します



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。

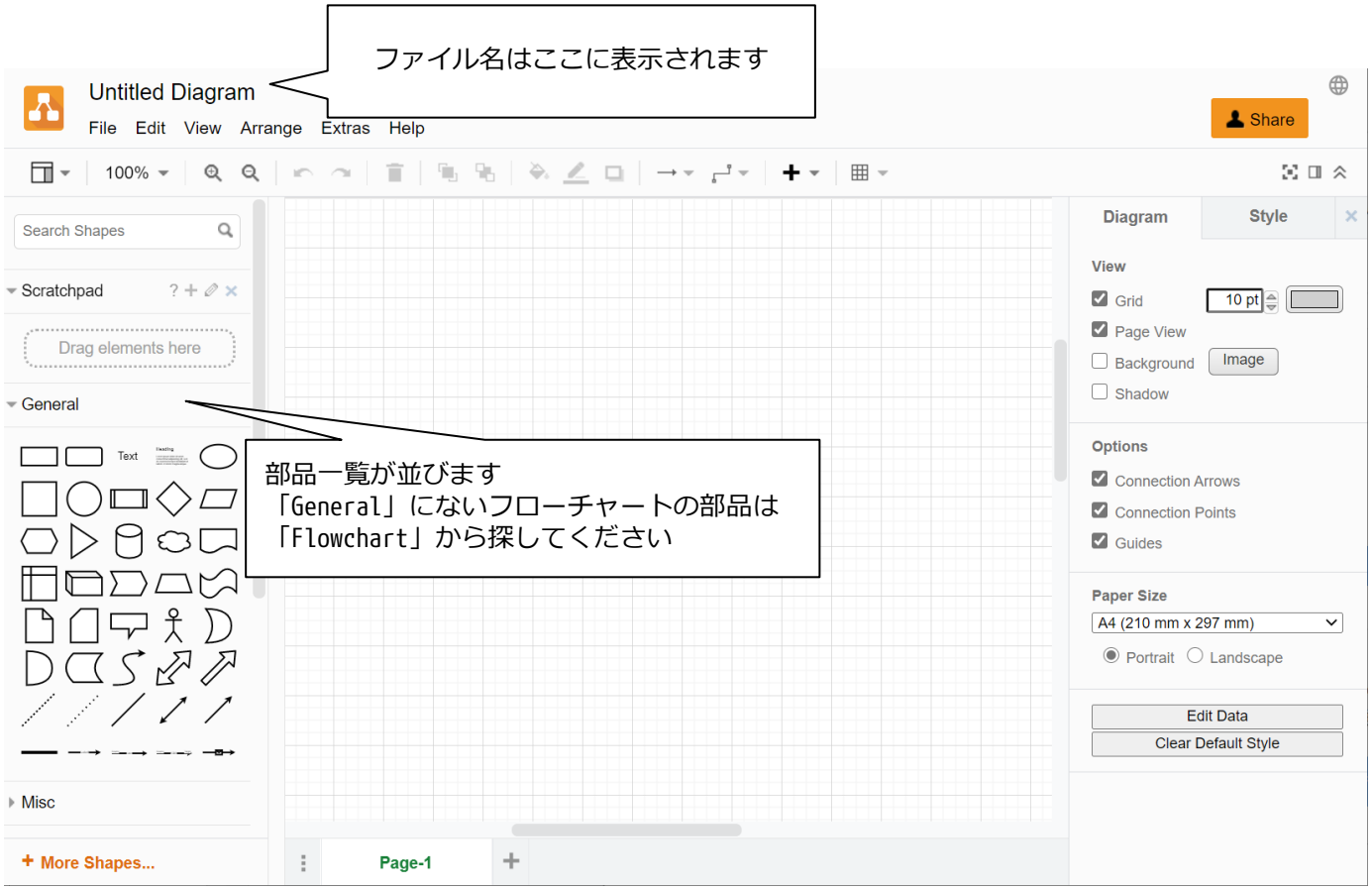


④矢印の先を部品に接続して線を接続します。

Copyright © Asial Corporation. All Rights Reserved.

draw.ioの起動

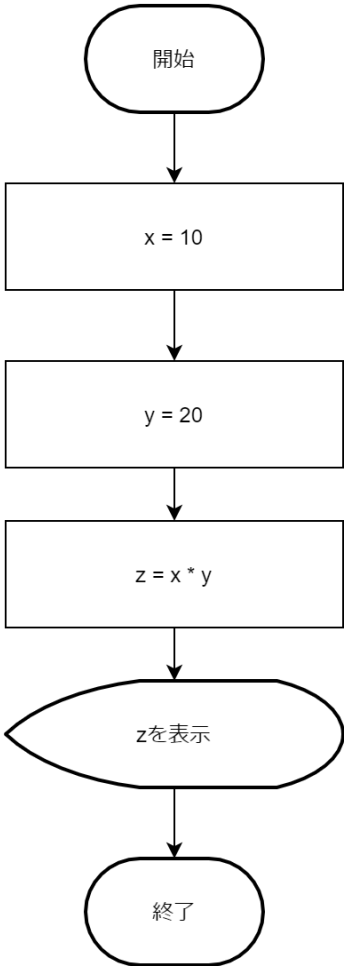
起動したらまずはフローチャート用の部品を探しましょう



Copyright © Asial Corporation. All Rights Reserved.

■ 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);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 疑似言語(文科省教材版)の対応表

- 文科省教材ではフローチャートに疑似言語が使用されています
  - 疑似言語の書き方は自由
  - 疑似言語の仕様の一つとして、DNCL(センター試験用手順記述標準言語)というものもあります

	疑似言語	JavaScript
変数に値を代入	x ← 0	x = 0;

変数の値を10増やす	x ← x + 10	x += 10;
		もしくは
		x = x + 10;

変数の値が60以上か	x ≧ 60	if (x >= 60)
奇数かどうか	i % 2 == 1	if (i % 2 == 1)

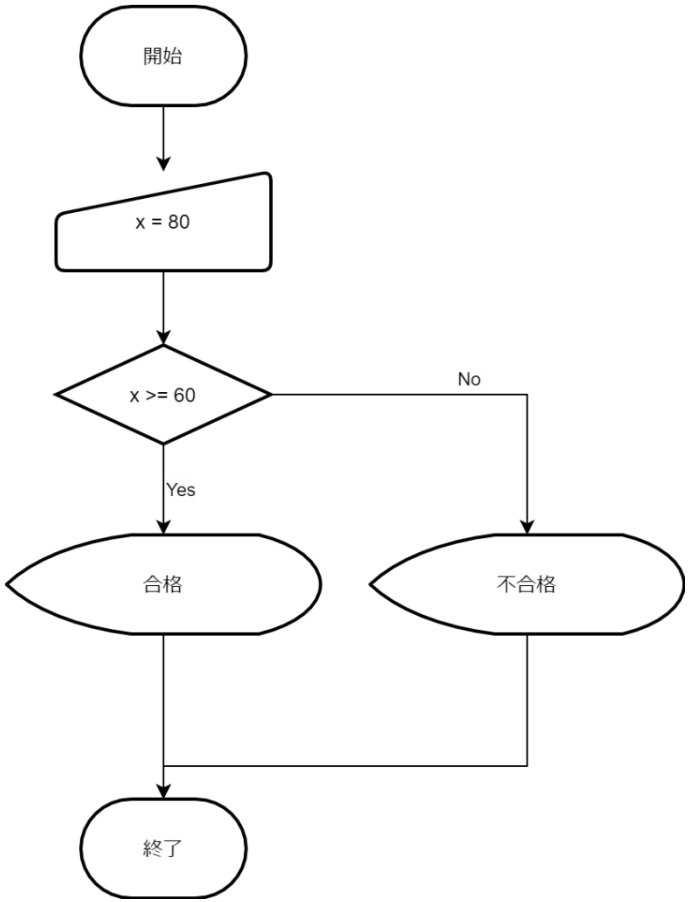
値を表示	x	document.write(x);
		もしくは
		console.log(x);
		alert(x);

繰り返し	i ← 1,2,3,4,5	for(i = 1; i < 6; i++)

Copyright © Asial Corporation. All Rights Reserved.

■ 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

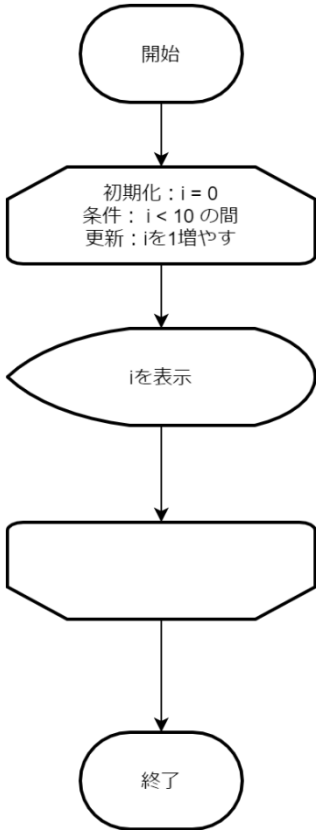
```
x = 80;

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

Copyright © Asial Corporation. All Rights Reserved.

■ 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。

Copyright © Asial Corporation. All Rights Reserved.

■ 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。

Copyright © Asial Corporation. All Rights Reserved.

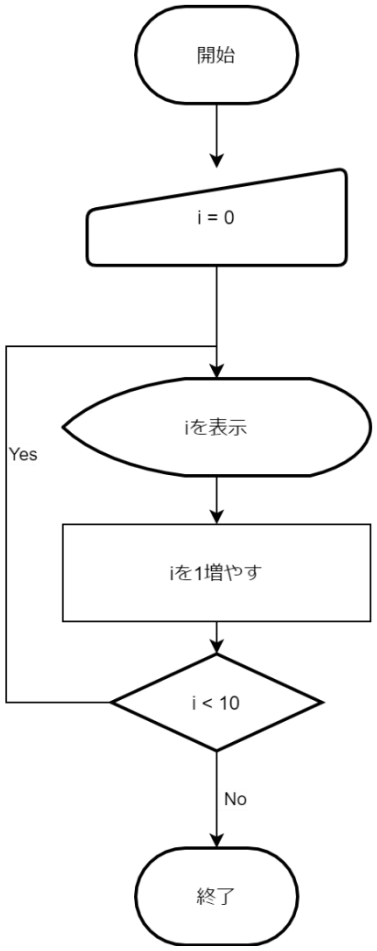
■ 変数：変数のインクリメントとデクリメント

- 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

Copyright © Asial Corporation. All Rights Reserved.

■ 繰り返しの作成②

└ 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

```
？？？？？
```

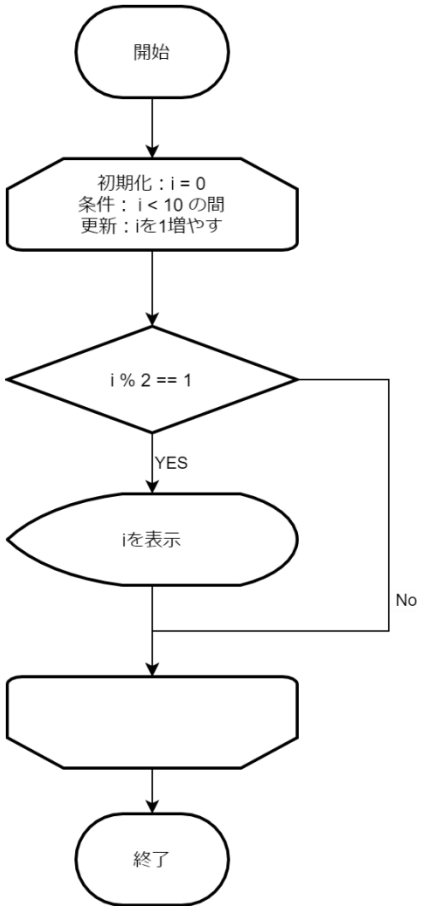
解説

- 今回のフローチャートをそのままソースコードに書き下すためには、任意の場所の処理にジャンプするような制御構造が必要です。
- 「順次」の考え方としては、任意の場所にジャンプされるのは好ましくありません。
- そのような構文として「goto文」が存在します
- goto文は自由度が高いため最近のプログラミング言語では搭載されていないものが多くなっています。
- JavaScriptはgoto文を持ちませんが、gotoに制限を掛けたような構文としてlabeled文があります。
- 左図のようなフローチャートをコードに起こす際は、for文やwhile文に置き換えて記述します。

Copyright © Asial Corporation. All Rights Reserved.

■ 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。

Copyright © Asial Corporation. All Rights Reserved.

【参考】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

Copyright © Asial Corporation. All Rights Reserved.

【参考】指導要領ではスコープの話はスコープ外

- 変数宣言時にvarやletを付けると参照可能な範囲(スコープ)が変わります
- 文科省教材のサンプルコードでは、何も付けていません
- プロ向けの技術書ではletやconstが推奨されています
- 教科『情報』では、どちらでも構わないと考えます

宣言	効果
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ます。 関数の中からも読み書きが行えます。 楽ですが長いプログラムでは事故の元。
var	スコープの狭い変数宣言を行えます。 関数の中で宣言した変数は、外から参照できません。
let	varより狭いスコープの変数宣言を行えます。 {}(ブロック)の中で宣言した変数は、外から参照できません。
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。

Copyright © Asial Corporation. All Rights Reserved.

【参考】JavaScriptのデータ型

Boolean	真偽値。trueとfalseの2値があります。 比較演算の結果などで用いられます。 例えば 10 < 100などの式を書くとtrueが返ります。 if (10 < 100)のような記述は内部的にはif (true)と評価されます。
Number	数値型。JavaScriptでは整数も負数も小数も全部Numberです。

String	文字列型。
	x = 10では数値の10が代入されますが、x = "10"の場合は文字列です。
	x = "10";
	y = "20";
	のような変数があったとして
	z = x + y;
	の結果zは文字列"1020"になります。
	これはJavaScriptの+記号は文字列連結も兼ねているためです。
	z = x * y;
	とした場合の結果zは数値の200となります。
	JavaScritpの*は掛け算のため、両辺が数値に型変換されます。

null	ナル値です。 数値の0や文字列""(空文字)とは異なります。
ArrayやObject	配列やオブジェクトは応用編以降で学習します

Copyright © Asial Corporation. All Rights Reserved.

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

Copyright © Asial Corporation. All Rights Reserved.

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

Copyright © Asial Corporation. All Rights Reserved.

■ 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);
    }
}
```

Copyright © Asial Corporation. All Rights Reserved.

■ もっと条件分岐について学ぶ

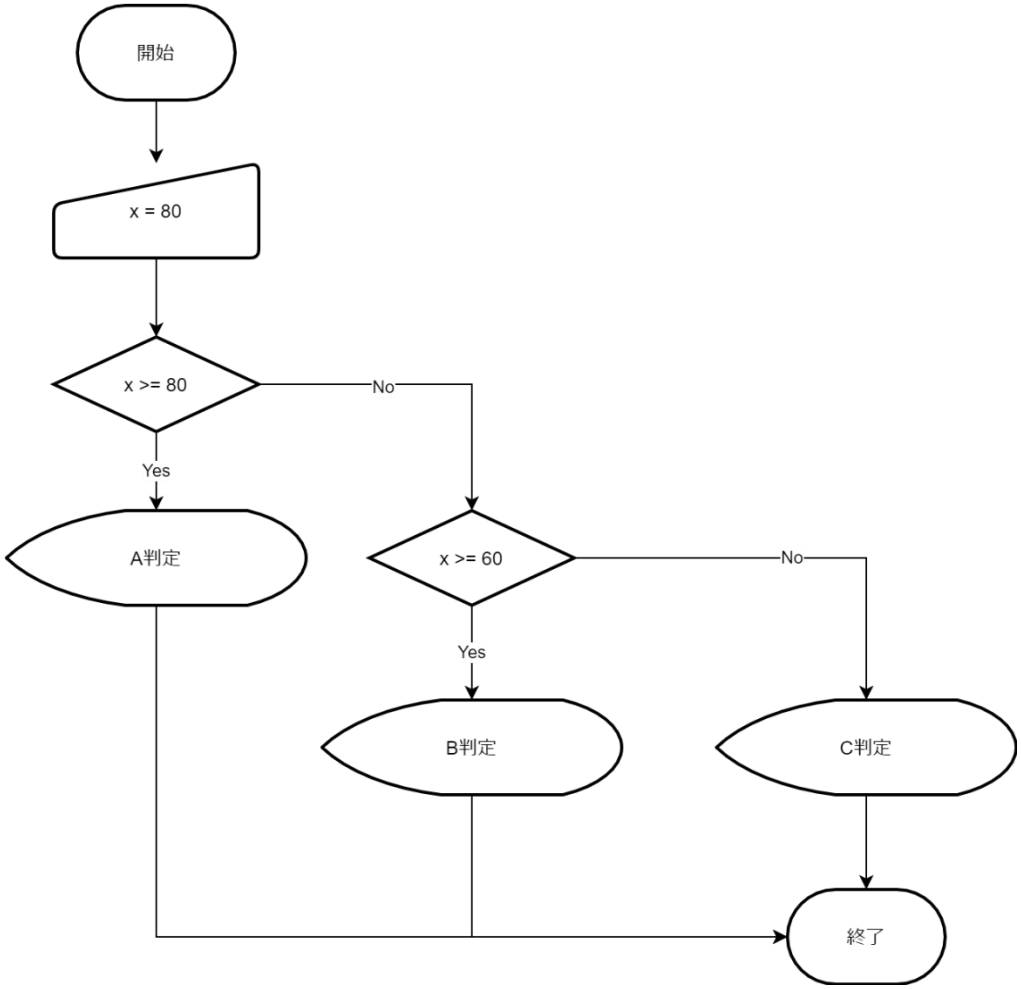
- 多方向分岐（else if 文）
  - 1つめの条件式にあてはまらない場合に、次の条件式を試みます

■ 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
}
```

Copyright © Asial Corporation. All Rights Reserved.

■ 実習：点数に応じてA～C判定を行うプログラムの作成



Copyright © Asial Corporation. All Rights Reserved.

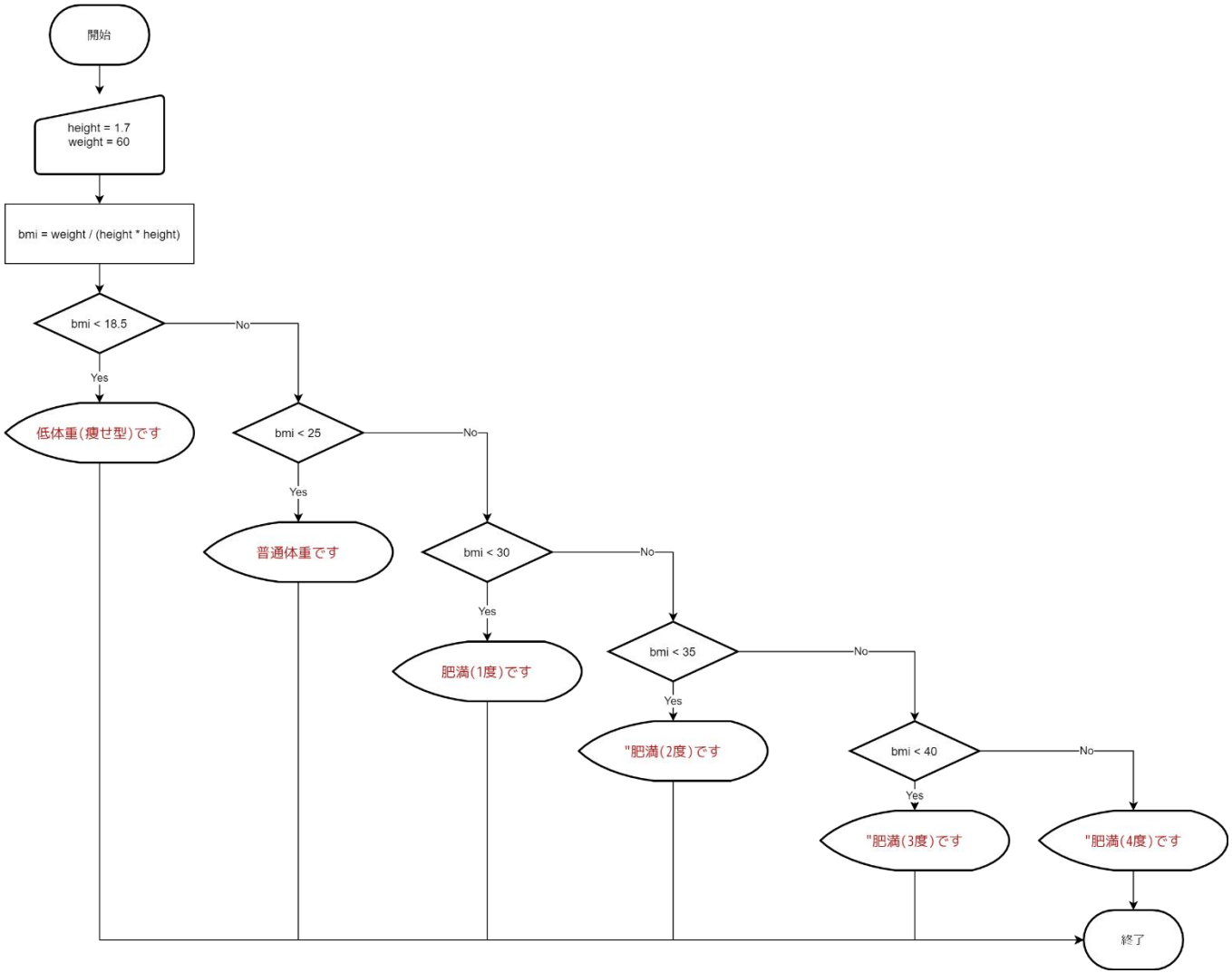
■ コード例

```
x = 80;

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

Copyright © Asial Corporation. All Rights Reserved.

【参考】身長と体重に応じてBMI判定を行うプログラムの作成



Copyright © Asial Corporation. All Rights Reserved.

■ 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。");
}
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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
}
```

Copyright © Asial Corporation. All Rights Reserved.

while文とfor文の比較

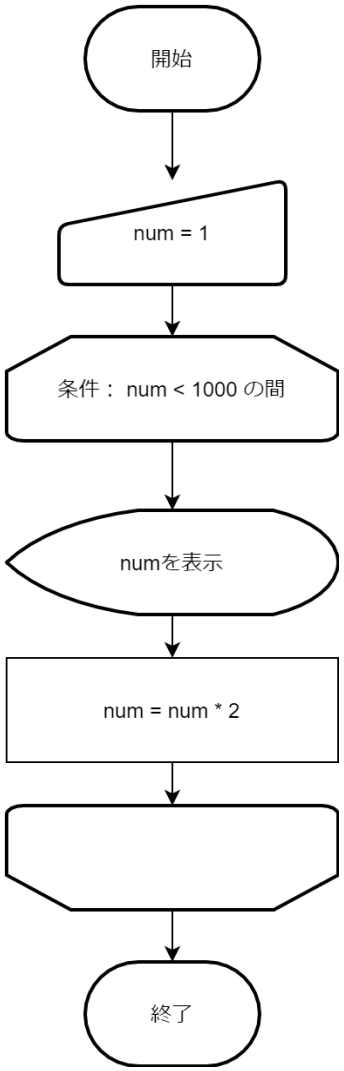
- 以下のfor文とwhile文は等価です
- 1行に初期化式と更新式をまとめられるのがfor文

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

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

Copyright © Asial Corporation. All Rights Reserved.

【参考】while文の例




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

Copyright © Asial Corporation. All Rights Reserved.

応用

Copyright © Asial Corporation. All Rights Reserved.

- 配列
- 関数
- 乱数
- WebAPI
  - 郵便番号WebAPIの取得

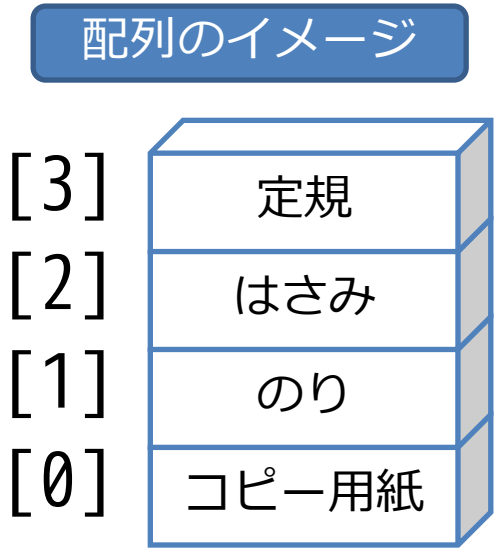
Copyright © Asial Corporation. All Rights Reserved.

■ 配列とは

└ 複数のデータを一括で管理するための仕組みです

└ レターケースのようなイメージ、各段に値(要素)が入ります

└ 各段にアクセスするための値を、「キー・添え字・インデックス」などと呼びます



Copyright © Asial Corporation. All Rights Reserved.

■ 配列の宣言と代入を行う文法

```
配列名 = [要素1, 要素2, 要素3, …];
```

■ 記述例(配列の宣言と代入)

```
letterCase = ["コピー用紙", "のり", "はさみ", "定規"];
```

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の参照を行う文法

```
配列名[添え字];
```

■ 記述例(配列の参照)

```
console.log(letterCase[0]); // コピー用紙がログに記録されます
console.log(letterCase);    // 全要素がログに記録されます
```

■ 配列の要素数を取得する文法

```
配列名.length
```

■ 記述例

```
letterCase.length;
```

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の要素を追加する文法

```
配列名.push("電池");
```

■ 記述例

```
letterCase.push("電池");
```

■ 配列の要素を削除する文法

```
配列名.pop();
```

■ 記述例

```
letterCase.pop();
```

Copyright © Asial Corporation. All Rights Reserved.

【参考】配列の要素を変更する命令一覧

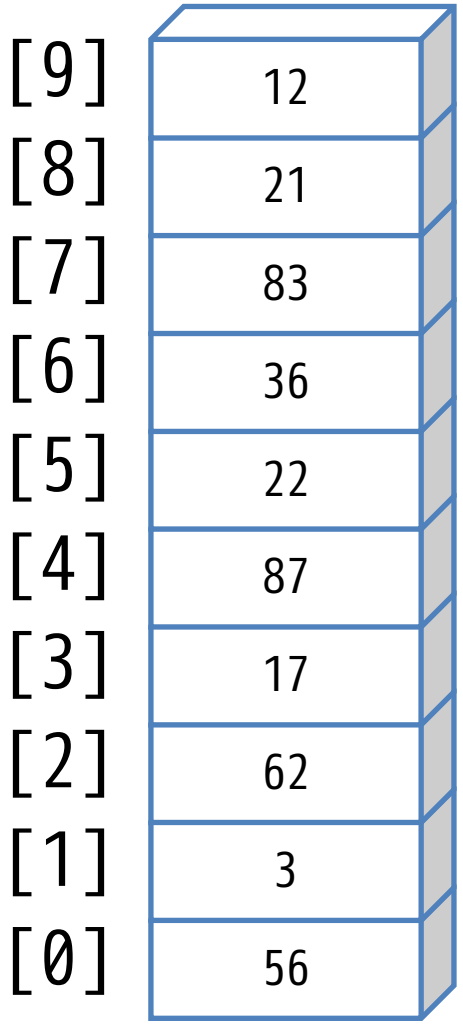
命令	効果
push()	配列の最後に要素を追加します。
pop()	配列の最後の要素を削除します。
shift()	配列の最初に要素を追加します。
unshift()	配列の最初の要素を削除します。
slice()	配列をコピーするときなどに使用します。
splice()	配列の任意の位置で要素の追加や削除を行えます。

※ slice()とsplice()は名前が間際らしいので注意。

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の計算

- 右図の配列を元に次の処理を行いたい
- [3]と[7]の合計値を求める
- 全ての合計値を求める
- 平均値を求める
- 最小値を求める



Copyright © Asial Corporation. All Rights Reserved.

■ 配列の足し算の例

```
a   = [56,3,62,17,87,22,36,83,21,12];
sum = a[3] + a[7];
console.log(sum);
```

■ 配列の合計値を求める計算の例

```
a   = [56,3,62,17,87,22,36,83,21,12];
sum = 0;

for (i = 0; i < a.length; i++) {
    sum += a[i]; // sum = sum + a[i];でも可。
}
console.log(sum);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の平均値を求める計算の例

```
a   = [56,3,62,17,87,22,36,83,21,12];
sum = 0;

for (i = 0; i < a.length; i++) {
    sum += a[i]; // sum = sum + a[i];でも可。
}

avg = sum / a.length;

console.log(avg);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の最小値を求める計算の例

```
a   = [56,3,62,17,87,22,36,83,21,12];
min = a[0]; // 暫定で0番目の要素を最小の値とする

for (i = 0; i < a.length; i++) {
    // minより小さい値があればそれを最小値とする
    if (a[i] < min) {
        min = a[i];
    }
}

console.log(min);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の最小値を求める関数の例

```
function min(a) {
    min = a[0];

    for (i = 0; i < a.length; i++) {
        // minより小さい値があればそれを最小値とする
        if (a[i] < min) {
            min = a[i];
        }
    }
    return min;
}

a   = [56,3,62,17,87,22,36,83,21,12];
min = min(a);
console.log(min);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 配列の合計値を求める関数の例

```
function sum(a) {
    sum = 0;

    for (i = 0; i < a.length; i++) {
        sum += a[i];
    }
    return min;
}

a   = [56,3,62,17,87,22,36,83,21,12];
sum = sum(a);
console.log(sum);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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。

Copyright © Asial Corporation. All Rights Reserved.

■ 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;
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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");
}
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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);
```

Copyright © Asial Corporation. All Rights Reserved.

■ 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);
```

Copyright © Asial Corporation. All Rights Reserved.

WebAPI （郵便番号）

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.

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。

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.

■ 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

Copyright © Asial Corporation. All Rights Reserved.