アルゴリズム・フローチャート・プログラム

Go.Ota

概要

1. プログラム構造とフローチャートの理解

2. Scratchを使用したプログラムの基本要素の組み込み

・各自のペースで進めてください。

注意: プログラムを作る場合は、作り始めに「新規」又は「コピーの保存」を使って別のプログラムにしてください。

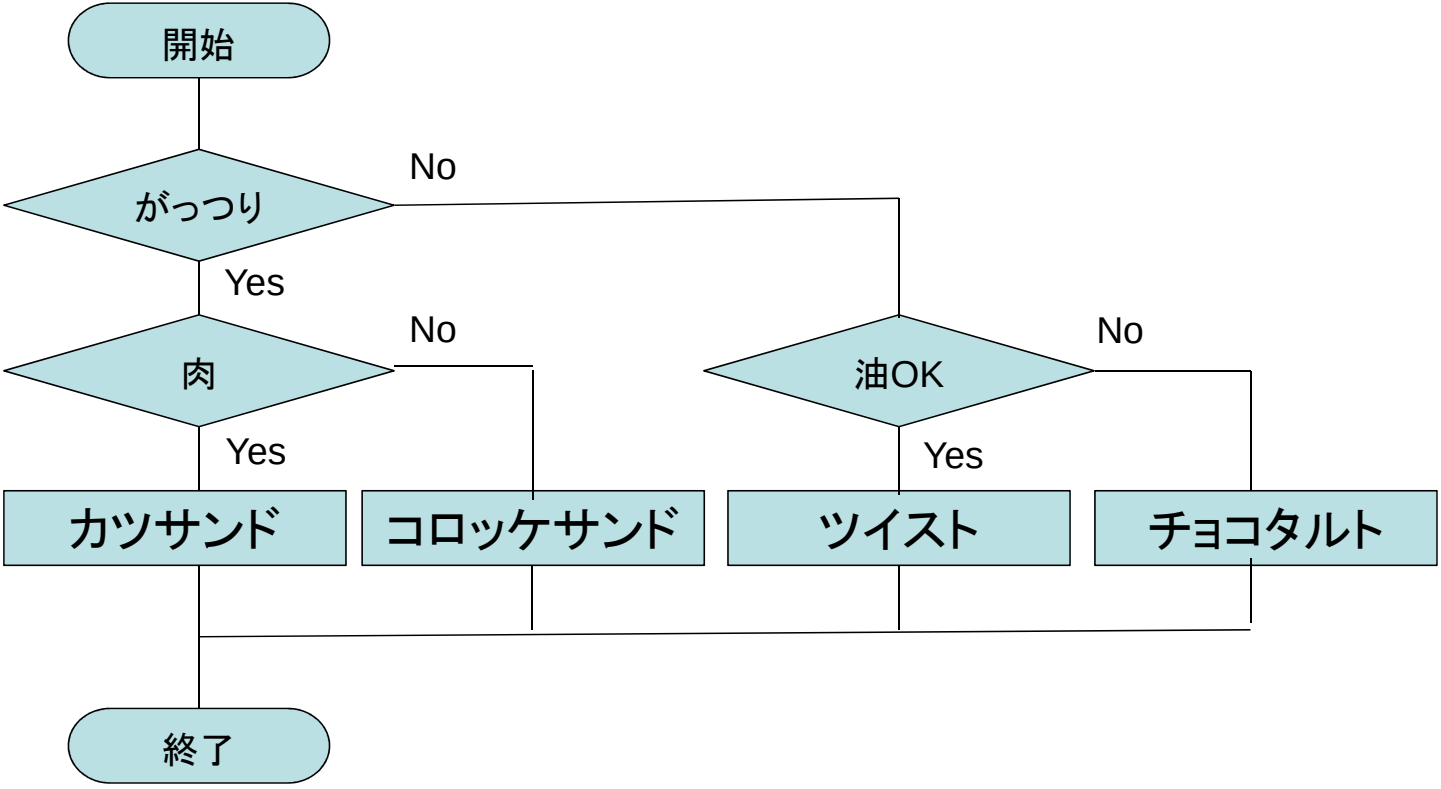
アルゴリズムは役に立つ

アルゴリズム = 問題解決や行動の手順

例: お昼のパンの選択

1. がっつり食べたい
   そうならば 揚げ物系のパン
   1.2 肉っぽいのがいいか
      そうならば　トンカツパン
      そうでなければ　コロッケサンド
   そうでないなら スィーツ系のパン
   1.3 油っこいのも平気
      そうならば　ツイスト
      そうだないならば　チョコタルト

フローチャートはアルゴリズムを視覚化する手法



・以下の作業を理解できた/作成できたを確認するチェックリストがあります。

◎スライドとサンプルプログラム

◎サーチプログラム

◎応用:素数プログラム

◎応用:並び替えのプログラム

チェックリストの使い方

内容	スライドNo	チェック			備考*
		(理解)	(打込み)	(開発)	
変数と Scratch での利用	7				
Scratch での変数への入力	8				
プログラムの構造/フローチャート	9				
変数(X=X+1)	10				
フローチャートと Scratch の対応	11-12				
一番簡単な自動販売機	13-14				P/F
チャレンジ:正三角形の判断	15		△		P
単純な 1 から n の合計(配列/リストは使っていません)	16				P/F

・スライドNo. スライド内の番号
・理解 :スライドの内容を見て自分なりにわかったらチェック
　　スタジオ内のプログラムの意味がわかったらチェック
・打ち込み：スライドのプログラムを入力して動作確認
・開発：スライドの課題のプログラムを作ったらチェック

変数とScratchでの利用

理解・打込み

a と b　の変数を作ってください。

普通の数学と演算記号が違います。

掛け算　*
割り算　/

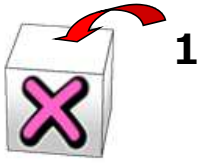
Scratchでの変数への入力

理解・打ち込み

aとbの値を入力してから計算しています。

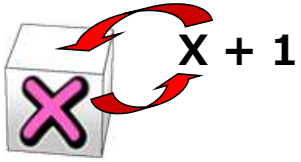
変数とScratchでの利用

理解



X ＝ 1

Xと名前をつけた箱(変数)に1を入れる



X ＝ X ＋1

(X ← X ＋ 1のイメージ)

初めにXの箱(変数)の中を取り出し+1する。
計算結果をXの箱(変数)に入れなおす。

◎Scratchでの X ＝ X ＋1

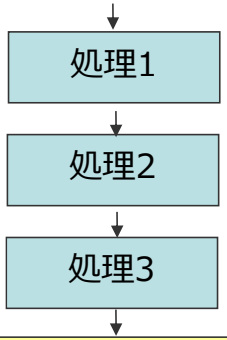
次の二つは同じ意味

変数への代入は普通の数学の=とは違う意味なのでイメージを示してみました。

プログラムの構造/フローチャート

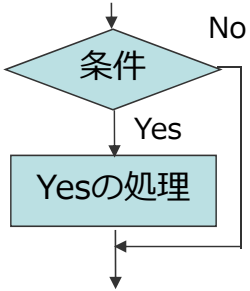
理解

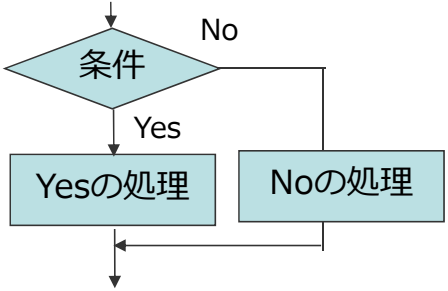
逐次構造(直線型)



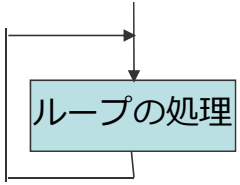
プログラムや人間の判断などのアルゴリズムは基本的に、逐次、選択:分岐、繰り返し(ループ)の組み合わせで表現できます。

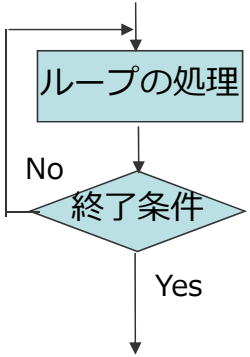
選択構造:分岐

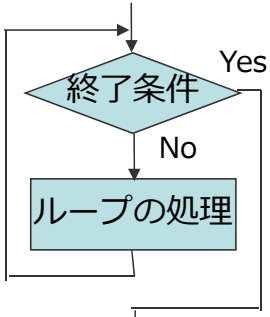




繰り返し構造(ループ)





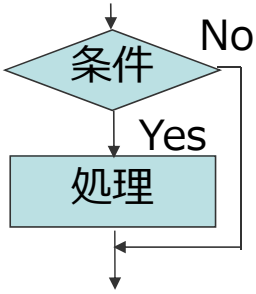


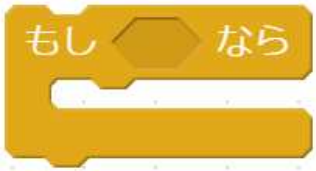
無限繰り返し型　後判定型　前判定型

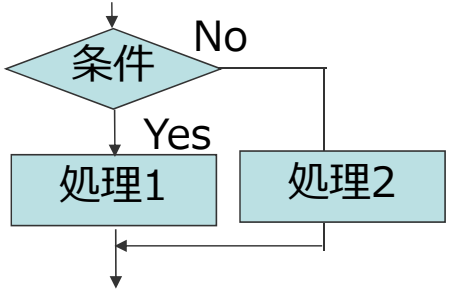
フローチャートとScratchの対応

理解

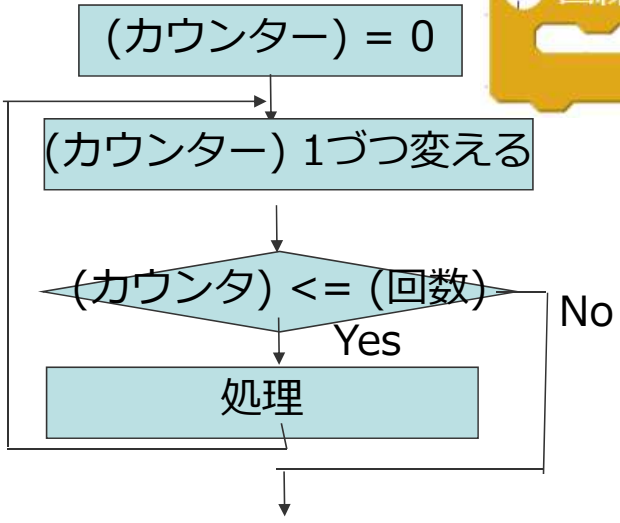
分岐



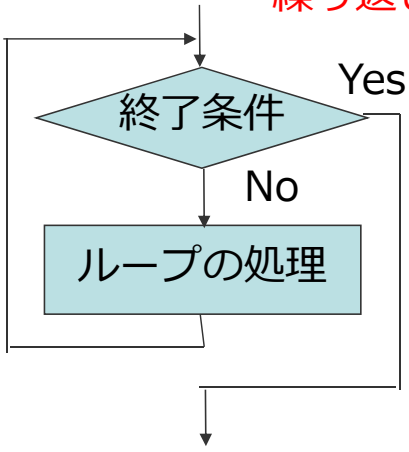




繰り返し:回数指定



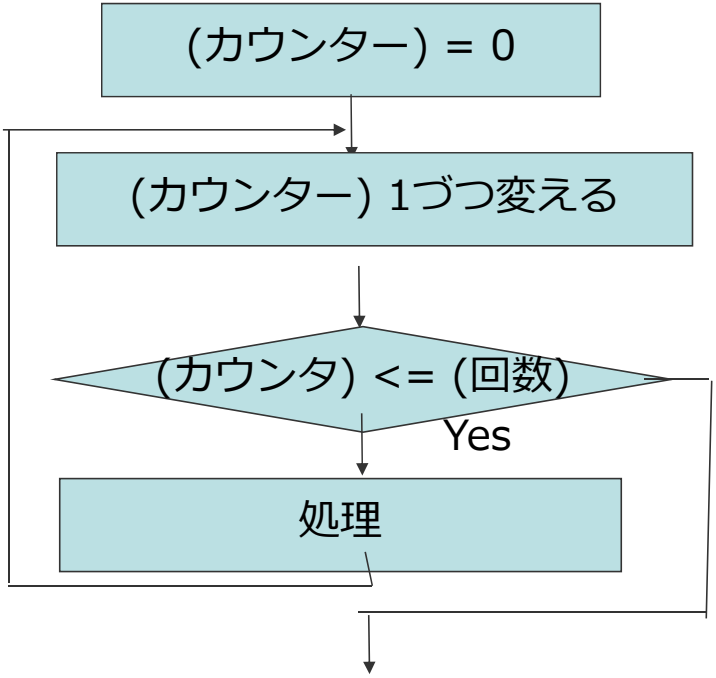
繰り返し:終了条件指定

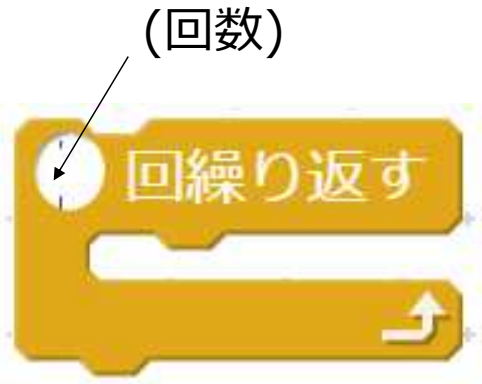


補足:フローチャートとScratchの対応

理解

繰り返し:回数指定





Scratchでは(カウンター)はシステムが使い見えません。
カウンターも変数です。

チャレンジ: 合格の判断

開発

aの変数に値を入力し

aが70未満だったら不合格

それ以外だったら合格

とネコに言わせてみよう。

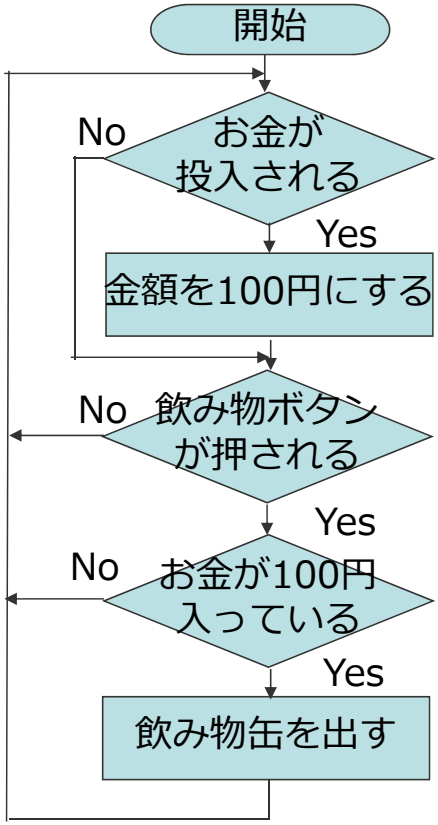
一番簡単な自動販売機

理解・打ち込み

簡単な仕組みの自動販売機
・100円玉を1枚だけ入れられる
・商品は1つだけ
・商品切れランプは無し
・お金返却ボタンは無し
・つり銭切れランプは無し
・お金を入れて一定時間たったら自動的にお金返却は無し

まず、始めの一番簡単な自動販売機について考えてみましょう。お金を入れることと製品のボタンを押すことしかできません。
このプログラムの動作をフローチャートとサンプルプログラムを次スライドに示します。

自動販売機01」


開始
お金が投入される
No
Yes
金額を100円にする
飲み物ボタンが押される
No
Yes
お金が100円入っている
No
Yes
飲み物缶を出す


がクリックされたとき
お金 を 0 にする
ずっと
100円入れる? yes/no と聞いて待つ
もし 答え = yes なら
お金 を 100 にする
飲みのもボタンを押す? yes/no と聞いて待つ
もし 答え = yes なら
もし お金 = 100 なら
お金 を 0 にする
缶を出すよ! と 2 秒言う

チャレンジ: 正三角形の判断

開発

a, b, cの3個の三辺の値を入力して、すべての値が同じ場合に、「正三角形」、そうでない場合は「正三角形じゃない」と表示するプログラムを作ってみよう。

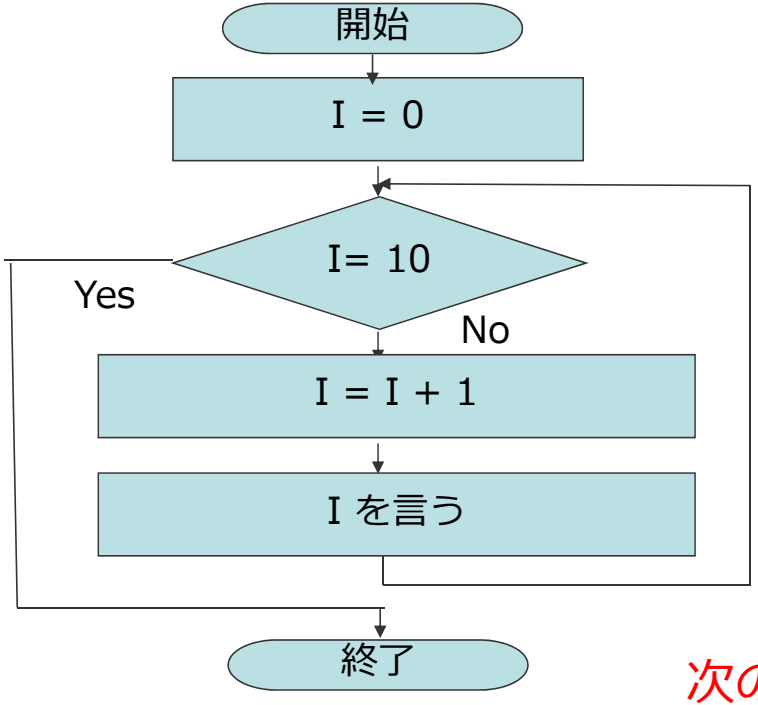
フローチャートは作っても、作らなくてもいいです。

・a,b,cは1以上の数を入力する前提で作っていいです。

1からnを言う

理解・打ち込み

1から10までの数を言うプログラムです



次のスライド
(スライド18)を先に見て

「1からnを言う」のイメージ

理解

1から10までの数を言うプログラムです

←変数I(アイ)のイメージ

ただし、このフローチャートは →
のような動作をしています。
カチカチしてみましょう

```
I = I + 1
```

これがIをカチカチしているところ。

カウンターが10になったら終了。

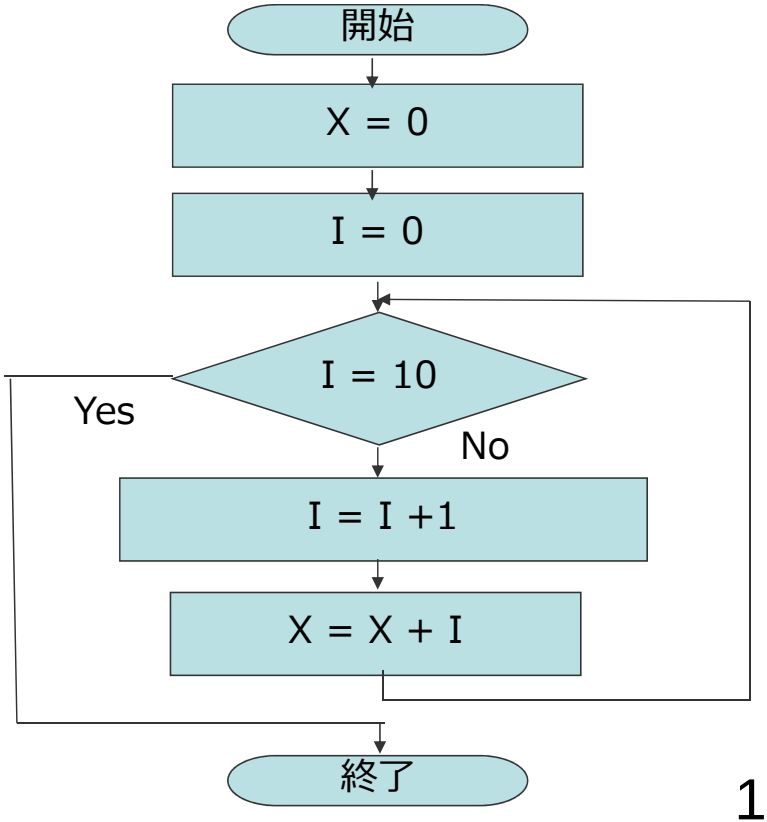
単純な1からnの合計

開発

1から10までの合計を計算するプログラムを作ってください
最期にネコに合計を言わせてね。

次のスライド
(スライド20)を
先に見て

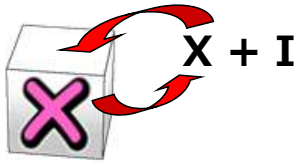
重要なヒント
プログラムは一つ前(スライド14)の「1からnを言う」を少しだけ改造して作ります。
スライド17のフローチャートと、右のフローチャートを見比べてどこが違うか考えてみよう。



「単純な1からnの合計」のイメージ

理解

1から10までの数を言うプログラムです



I = I + 1

1,2,3・・・と
数を作る

X = X + I

プログラムは、これがIをカチカチ
やって出てきた数をXの内容に足して
いくイメージです。
カチカチしながら、Xの中がどのよう
に変わるか紙に書いてみよう

チャレンジ:単純な2からAまでの偶数の合計

開発

Aの値を入力して、2からA以下の偶数の合計を求めるプログラムを作成してください。

例: A = 6の場合、 2+4+6
A = 9の場合、 2+4+6+8

ヒント:

Step1:

まず、一つ前(スライド19)の「1から10までの合計」を求めるプログラムを少し改造して、2から10までの偶数を認めるプログラムを作ります。次のスライド(スライド22)を先にみてね

Step2:

次に数をAに入力したあと、その数までの合計を求めるプログラムに改造します。Aを入力することは、スライド8の変数への入力を見て。「その数までの合計」は、スライド19のフローチャートを良く見て、どこで終わりを判断しているかな?。

「単純な1からnの合計」のイメージ

理解

I = I +1

1,2,3・・・と
数を作る場合

????????

2,4,6・・・と
数を作る場合

X = X + I

これは同じ。

プログラムは、これがIをカチカチやって出てきた
数をXの内容に足していくイメージは同じです。
Iで偶数を作る方法を考えてみてください。

チャレンジ:5個の数の合計

5個の数を入力して合計を求める
プログラムを作成してください。

開発

ヒント:
5つの数を入力するプログラムです。
これに合計する部分を追加してください。
A,B,C,D,Eの変数を使います。

たくさんの数の合計

次からたくさんの数を入力して合計を
求めるプログラムをつくっていきます。

今まで、数を入力するのに変数をつくっていきましたが、たくさんの数を入力するには、A,B,C,D,E・・・・と多くの変数を使うのは大変です。

こんな時に配列(リスト)使うと便利です。
次から配列(リスト)を学習していきます。

配列(リスト)を作る

理解・打ち込み

変数の中で[リストを作る]を指定。

リスト名を指定して作成

配列はDに番号がついているイメージ

おみくじをつくる

理解・打ち込み

配列(リスト)を使っておみくじを作ってみます。

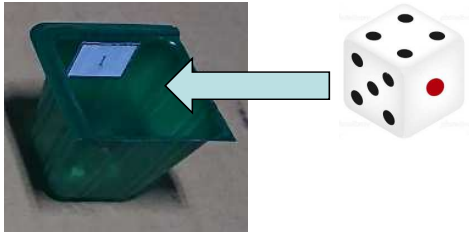
動作のイメージは次のスライド(スライド27)を見て

「おみくじを作る」のイメージ

理解・打ち込み

大吉　中吉　吉　凶　大凶

↓　↓　↓　↓　↓



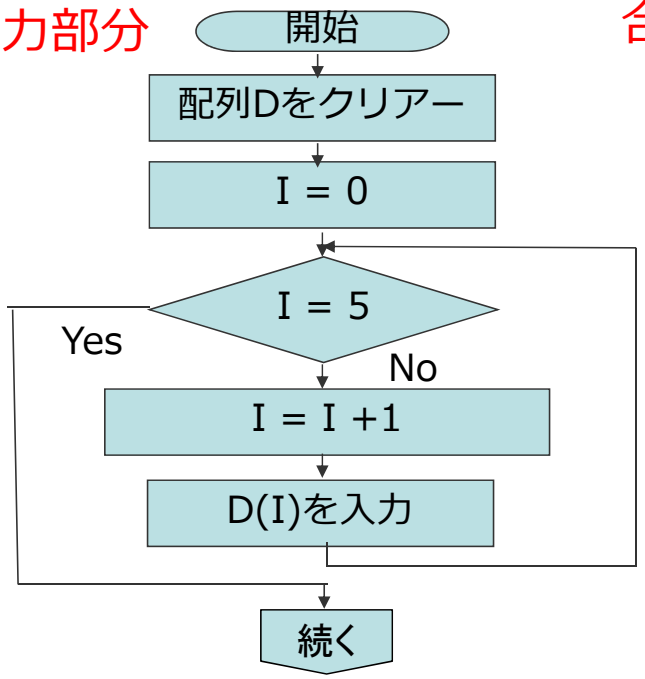
D[I] の内容を言う

チャレンジ: 5回数字を入力してその合計(配列利用)

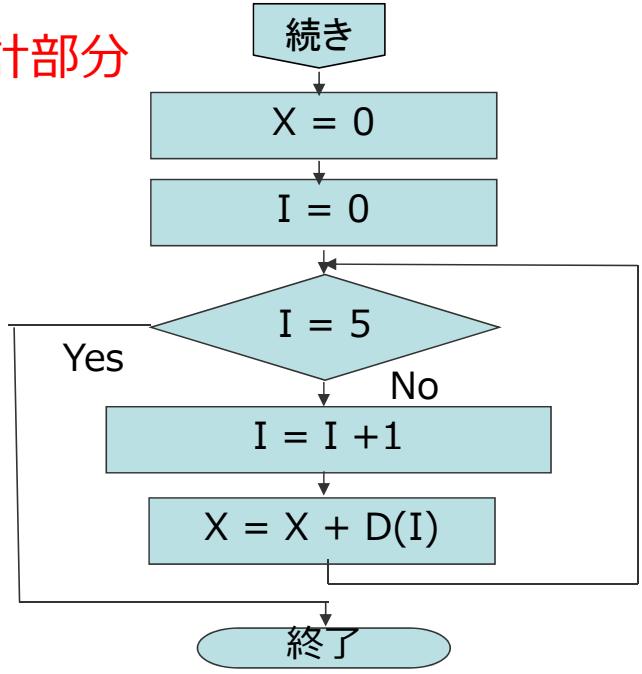
5個の数を入力して、合計を計算するプログラム
配列を利用します。

開発

入力部分



合計部分



次の2枚のスライドを先に見て

チャレンジ: 5回数字を入力してその合計(配列利用)

入力部分(スライド28の左側)のプログラムです。

開発

この下に、スライド28の右側のプログラムを追加して

カチカチやりながら、D[I] を5回入力しているイメージです。

合計のイメージの
次のスライドも見て

チャレンジ: 5回数字を入力してその合計(配列利用)

合計部分のプログラムのイメージです。

開発

カチカチやりながら、
D[Ｉ] を5回とりだしています。

おみくじで
使いました。

X = X + D[Ｉ]

D[Ｉ] を5回, Xに加えています。

配列に数をセットする方法(1)(2)

理解

配列(リスト)に数をセットする方法にはいくつかあります。ここでは、3つの方法を理解しましょう。

(1) 入力する。
スライド29で説明済み

(2) 配列(リスト)に直接、
数値を入力する。

次のスライドで(3)を説明しています。

配列に数をセットする方法(3)

理解・打ち込み

(3) ランダムな数を設定する。

カチカチしながら、1から100の目の出るサイコロを振って数値を入れているイメージです。

次から課題はこれを使って配列(リスト)に数値を設定します。

D[1:5]に数値設定

チャレンジ:配列の中から数を探す(検索)

開発

予め配列D[1]からD[5]まで数を入れておきます。

つぎに変数のAに数を入力して、

その数がD[1]からD[5]にあれば、何番目に入っていたか変数Xにその番号を入れます。

入っていなければ変数Xに0を入れます。

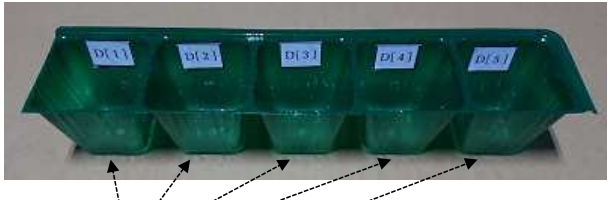
次のスライド(スライド34と35)にヒントがあります。

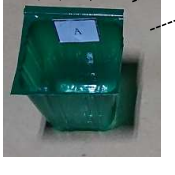
チャレンジ:配列の中から数を探す(ヒント)

1. あらかじめ配列(リスト)Dに5つの数を入れておきます。
   スライド32のプログラムを使います。
2. 変数Xに見つからなかった時のために0を初めに入れときます。
3. 変数Aに探す数を入力します。

⇦ 0

4. 変数Iをカチカチしながら,D[I]とAの内容を比較します。
もし等しければ、I(何番目かの番号)の値をXに入れます。

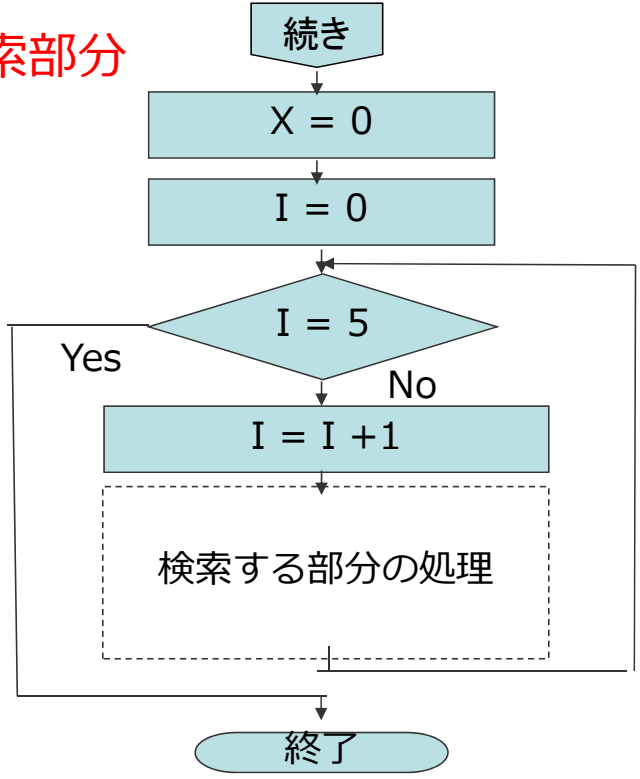




順番に比較していく。

チャレンジ:配列の中から数を探す(ヒント)

検索部分



検索する部分の処理は

数のAに数を入力して、AがD[I]と等しければ、何番目に入っていたか変数Xにその番号を入れます。

補足:
合計部分の前には、スライド31/32のどれから方法使って、配列に数値を入れておきます。

数の並び替えの作業の説明

予め配列D[1]からD[5]まで数を入れておきます。

この中の数を小さい順番に並び替えて配列D[1]からD[5]に入れなおしてください。

理解

これが最後の課題です。できると思う人は、いきなり開発してもいいです。
少し難しいと思う人は、次の課題を順番にやっていきましょう。

・配列の中の一番小さい数を見つける
・配列の中の一番小さい数を、配列の先頭に入れる。
・二重のループを使う
・並び替えの開発

配列の中の一番小さい数を見つける

開発

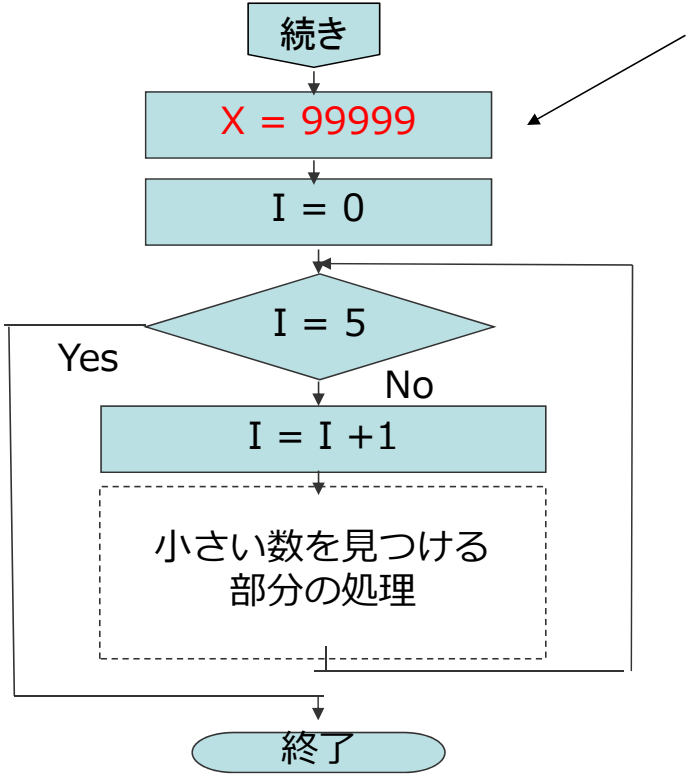
予め配列D[1]からD[5]まで数を入れておきます。

その中で一番小さな数をXに入れます。

次のスライド(スライド38)にヒントがあります。

配列の中の一番小さい数を見つける(ヒント)

見つける部分



ポイント

変数Xにあらかじめ、配列に無いような大きな数を入れておきます。

小さい数を見つける部分の処理は

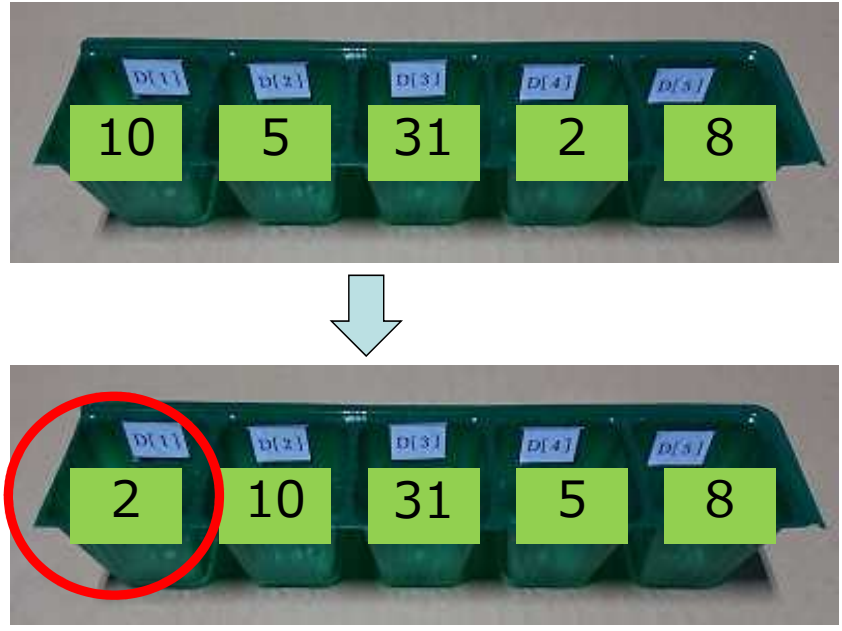
Xには常に一番小さい数が入っていると考えます。
Xの中の数とD[I]を比較して、D[I]の方が小さければXにそのD[I]を入れます。

配列の中の一番小さい数を配列の先頭に入れ替える

予め配列D[1]からD[5]まで数を入れておきます。その中で一番小さな数を配列の先頭(D[1])に入るように入れ替えます。

開発

次のスライド(スライド40/41/42)にヒントがあります。

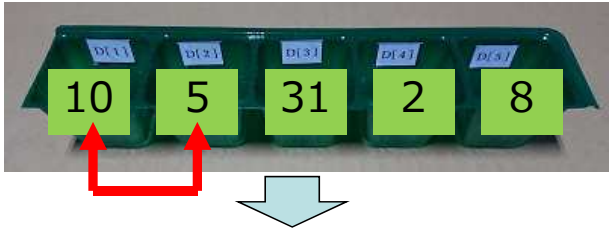


プログラムを実行した後は、

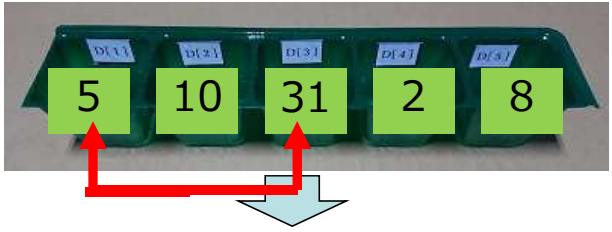
一番小さい数がD[1]に入ります。但し、配列には元の配列にあったすべての数が残っています。

配列の中の一番小さい数を配列の先頭に入れ替える

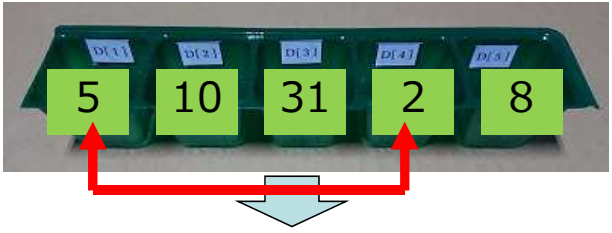
ヒント: プログラムの考え方　(実際に紙に数値を書いてやってみよう)



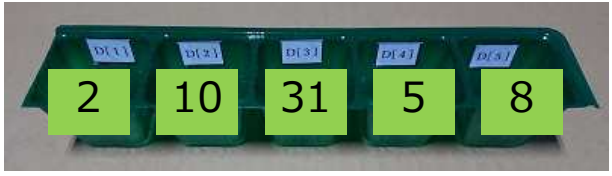
1番目と2番目を比較して5の方が小さいので入れ替え



1番目と3番目を比較して5の方が小さいので入れ替えない



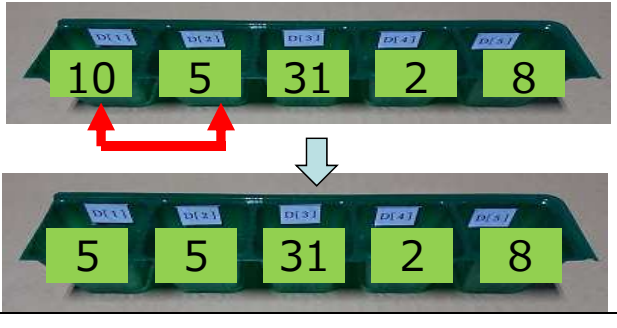
1番目と4番目を比較して2の方が小さいので入れ替え



1番目と5番目を比較して2の方が小さいので入れ替えない

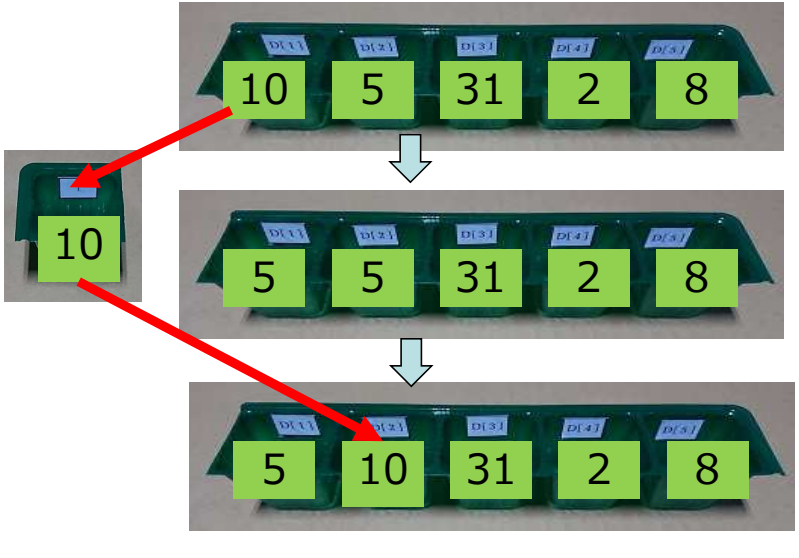
配列の中の一番小さい数を配列の先頭に入れ替える

ヒント: 数の入れ変え



1番目と2番目の入れ替え

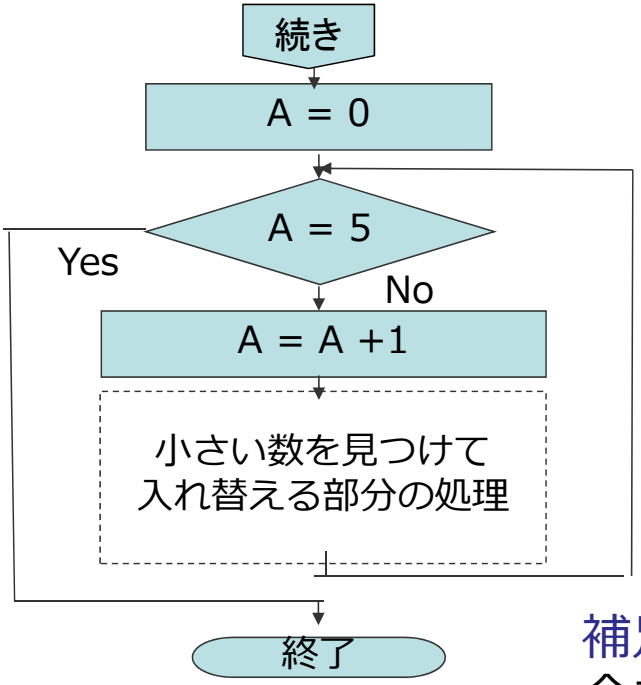
いきなり入れ替えると10が残らない



入れ替え先の数を別の変数に保存しておき、それを後で戻す

配列の中の一番小さい数を配列の先頭に入れ替える

入れ替える部分



小さい数を見つけて
入れ替える部分の処理は

D[1]には常に一番小さい数が入っていると考えます。
D[1]の中の数とD[I]を比較して、D[I]の方が小さければD[1]とD[I]を入れ替えます。

ポイント:
ちょっとおかしいけど初めはD[1]とD[1]比較します。

補足
今までは、配列の番号はIという変数を使っていましたが、ここではAという変数を使っています。

二重繰り返しに挑戦

理解・打ち込み

予め配列D[1]からD[5]まで数を入れておきます。
初めにD[1]からD[5]までの数
次にD[2]からD[5]までの数
その次にD[3]からD[5]までの数
その次の次にD[4]からD[5]までの数
最期にD[5]までの数を言うプログラムを作ります。

スライド44にプログラム
スライド45にフローチャート
スライド46に解説
があります。

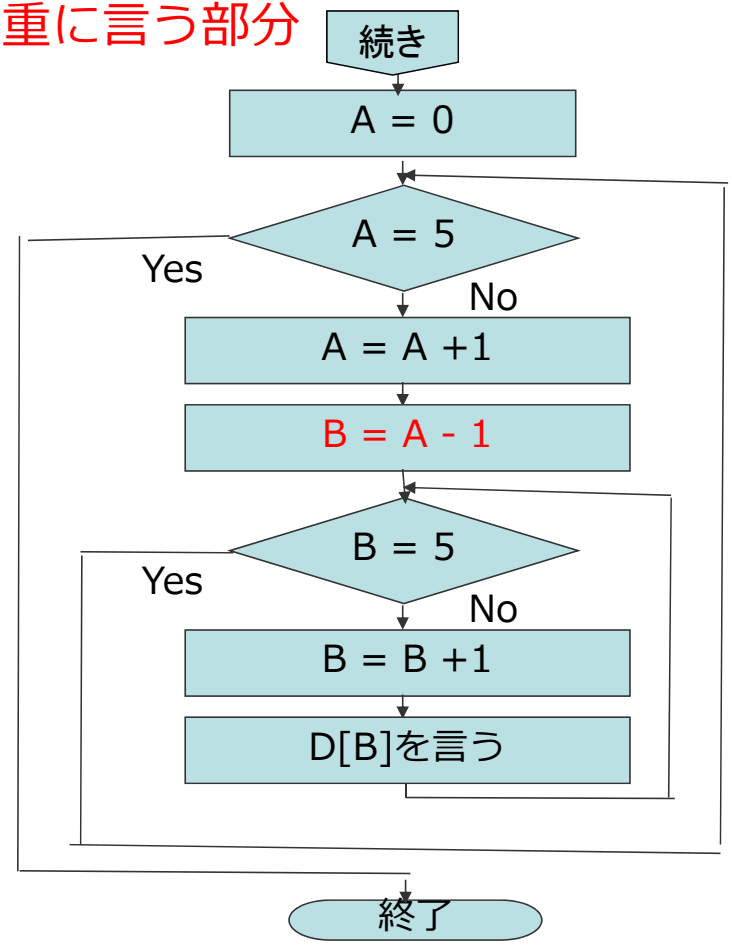
二重繰り返しに挑戦

理解・打ち込み

補足:二重に言う部分の前には、スライド31/32のどれかの方法使って、配列に数値を入れておきます。

二重に言う部分

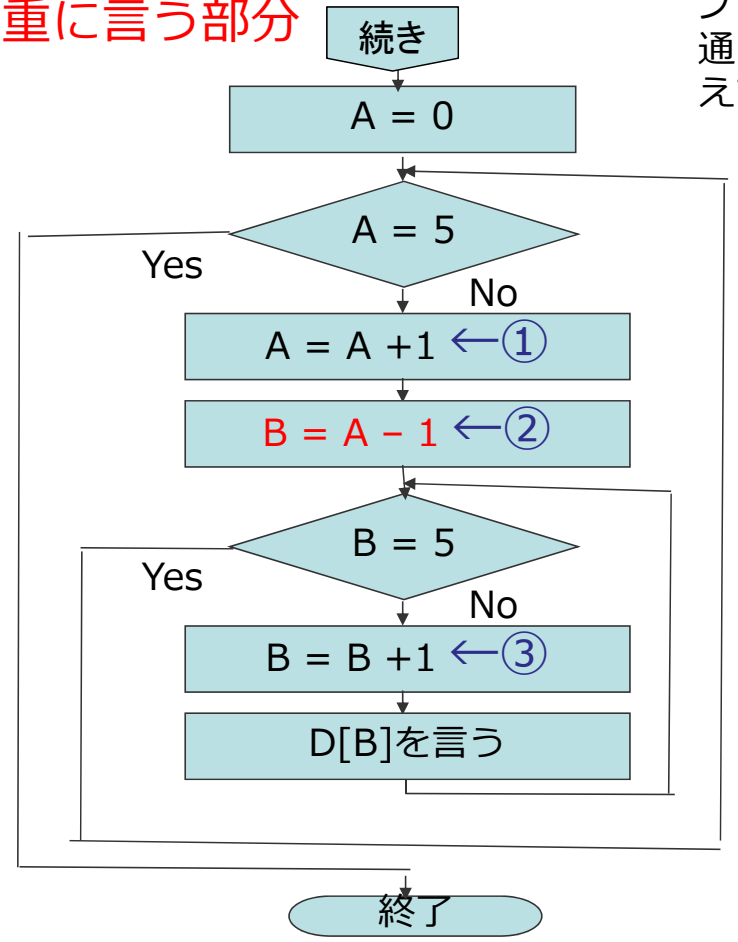
二重に言う部分



フローチャートはごちゃごちゃしていますが、プログラムで見ると、
Aを使った大きな繰り返しの中にBを使った繰り返しが入っている形になります。

二重繰り返しに挑戦

二重に言う部分



プログラムを動かすと①②③のどこを通って、AとBがどのように変わるか考えてみましょう。

通る場所	A	B
①	1	?
②	1	0
③	1	1
③	1	2
③	1	3
③	1	4
③	1	5
①	2	5
②	2	1
③	2	2
③	2	3
③	2	4
続いていきます		

最期のチャレンジ:数の並び替え

開発

予め配列D[1]からD[5]まで数を入れておきます。この中の数を小さい順番に並び替えて配列D[1]からD[5]に入れなおしてください。

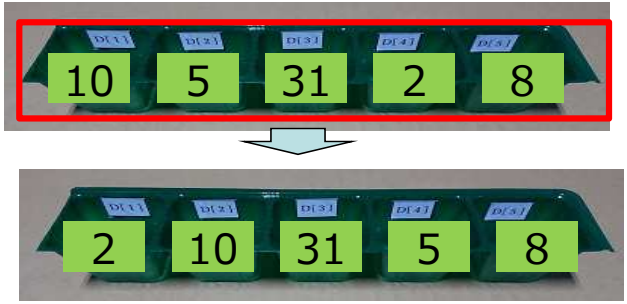
プログラムを実行した後は、数が大きくなるように入れ替えが行われています。

次のスライド(スライド48/49)にヒントがあります。

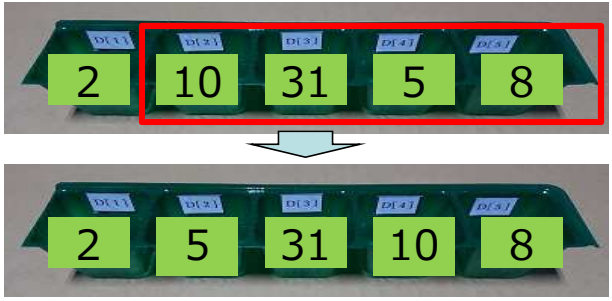
最期のチャレンジ:数の並び替え

ヒント: プログラムの考え方：「配列の中の一番小さい数を配列の先頭に入れ替える(スライド37)を繰り返し実施(実際に紙でやってみよう)

1回目
D[1]からD[5]で一番小さい数を配列の先頭D[1]に入れ替える



2回目
D[1]に一番小さい数がはいっているので、D[2]からD[5]で一番小さい数を配列のD[2]に入れ替える



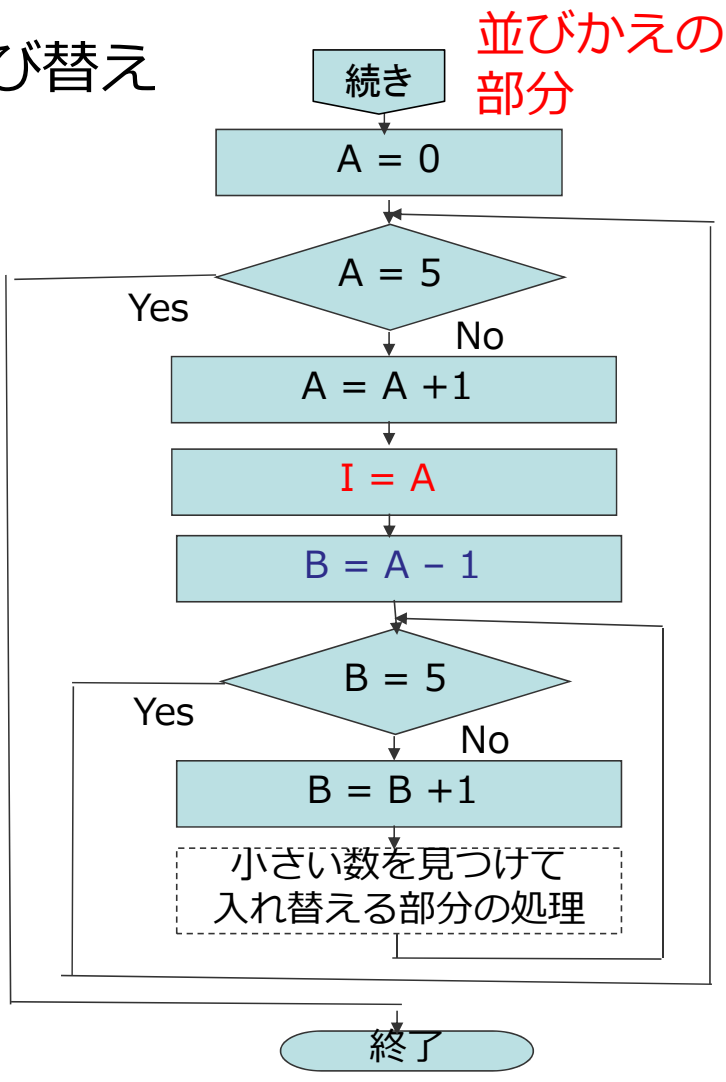
3回目
D[3]からD[5]で一番小さい数を配列のD[3]に入れ替える

3回目
D[4]からD[5]で一番小さい数を配列のD[4]に入れ替える

最期のチャレンジ:数の並び替え

プログラムは①「二重繰り返しに挑戦」の中に、②「配列の中の一番小さい数を配列の先頭に入れ替える」を組み込んだものになります。

ポイント:
②のプログラムの場合は、一番小さな数が入る配列はD[1]に固定されていました。並べ替えの場合は、これが変わっていきます。右のフローチャートでIを追加したので、これを利用して開発してみてください。



発展課題1：沢山の数の並び替え

開発

配列入れる数の個数Aを入力し、配列にA個のランダムな数を設定してから、この配列の中の数を小さい順番に並び替えてください。

ヒント:
スライド32の方法で、配列にA個のランダムな数を設定します。

発展課題2:並び替えの無駄を省く

開発

「最期のチャレンジ:数の並び替え」のプログラムは、「二重繰り返しに挑戦」のプログラムを改良したため、他えばA=5, B=5の時はすでに並び替えが終っているのに、無駄な処理をしているようなことがあります。

この無駄を省いたプログラムを考えてみましょう。

発展課題3:いろいろな並び替えの方法

開発

並び替えにはいろいろなソートの方法があります。今回開発した方法は、「選択ソート」という方法です。

他にいろいろな並べ替えの方法がありますが、Webで「バブルソート」を調べて、プログラムを開発してみてください。