第１回プログラミング入門

本演習履修にあたって

参考書：「C言語によるプログラミング入門」
吉村賢治著、昭晃堂

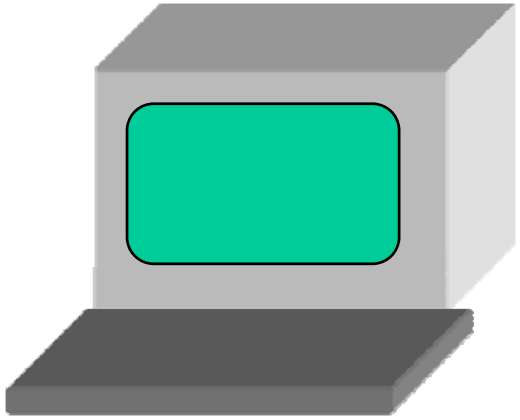
「プログラミング言語C」
カーニハン、リッチー著、共立出版

サポートページ：
http://www.ec.h.akita-pu.ac.jp/~programming/

プログラミング演習の目的

コンピュータを用いた問題解決ができるようになる。

そのために、プログラムの作成能力を身に付ける。



今回の目標

- 演習の遂行に必要なツールの使用方法を習得する。
- 課題の提出法を習得する。
- 現実の問題をプログラムにするまでの概要を理解する。

☆演習室から課題を提出する。

演習で利用する
Linux上プログラミング環境（1）

- GNOME 端末
  コンピュータをキーボードを使って操作する

- Emacs テキストエディタ
  プログラムを記述し、保存する

- Subversion バージョン管理システム
  記述したファイルを他の人と共有したり、
  過去の内容を調べたりする
  （本演習では課題の提出に利用）

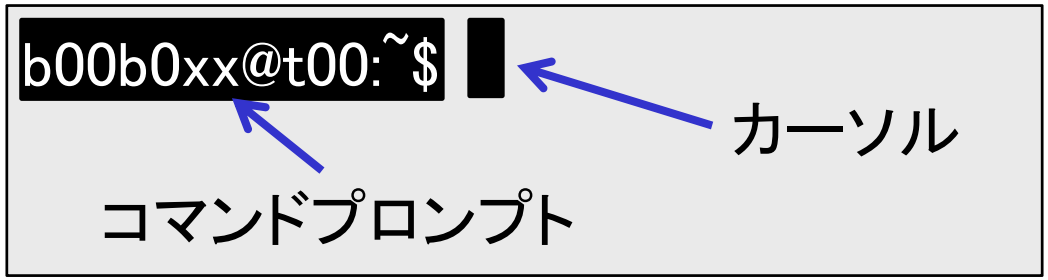
演習で利用する
Linux上プログラミング環境（2）

以下のツールは次回の演習で説明する

- GCC（ GNU C コンパイラ ）
  C言語で記述されたプログラムを
  コンピュータが実行できる形式（機械語）
  に変換する

- Make
  GCCなどを自動的に起動する

端末とコマンド

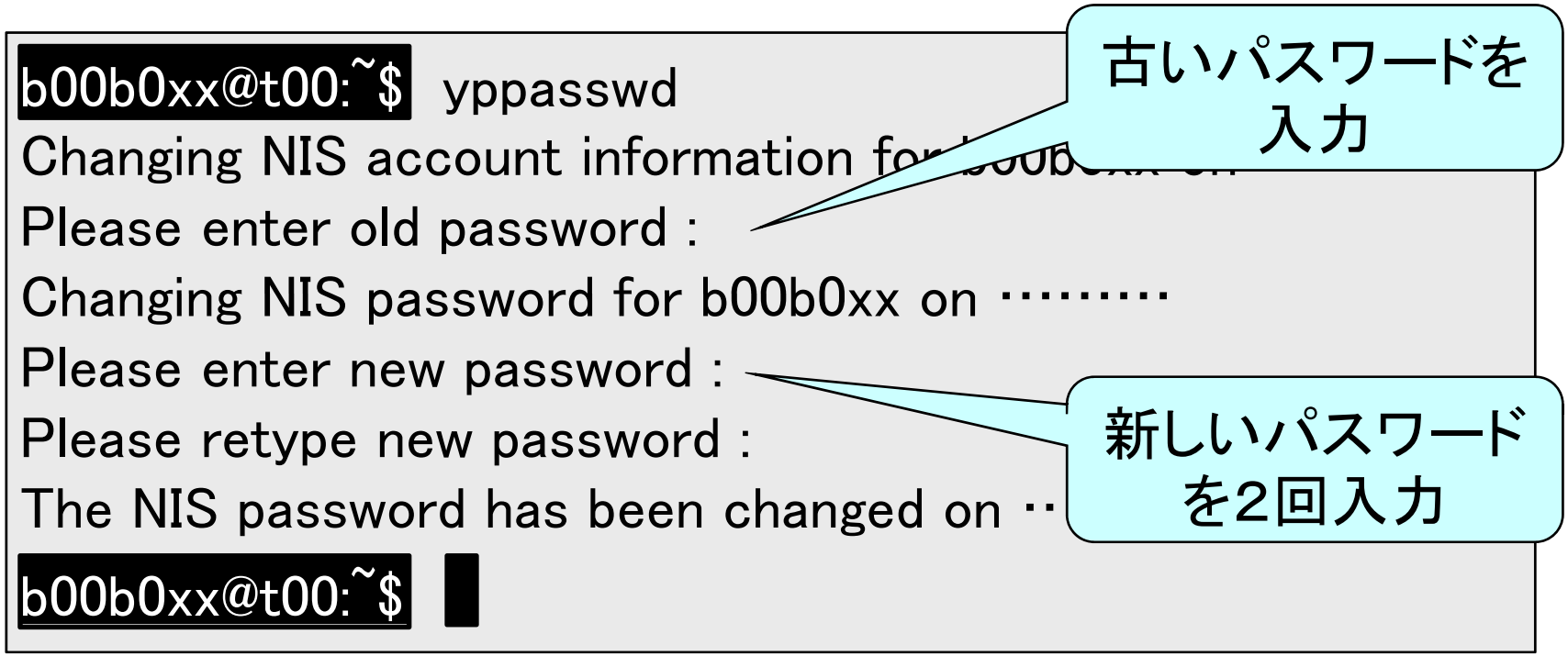


- コマンドプロンプトが出ている時に、
  コマンド（命令）をキーボードを使って入力
- カーソル位置に文字が入力される
- 矢印キーなどを使って編集できる

コマンドの実行

- 多くのコマンドはコマンドの引数を必要とする（スペースで区切って入力）
- コマンドを入力後、Enterキーで実行
- ウィンドウを開くコマンド（Emacsなど）を実行する際には、最後に「&」を付けること

パスワードの変更

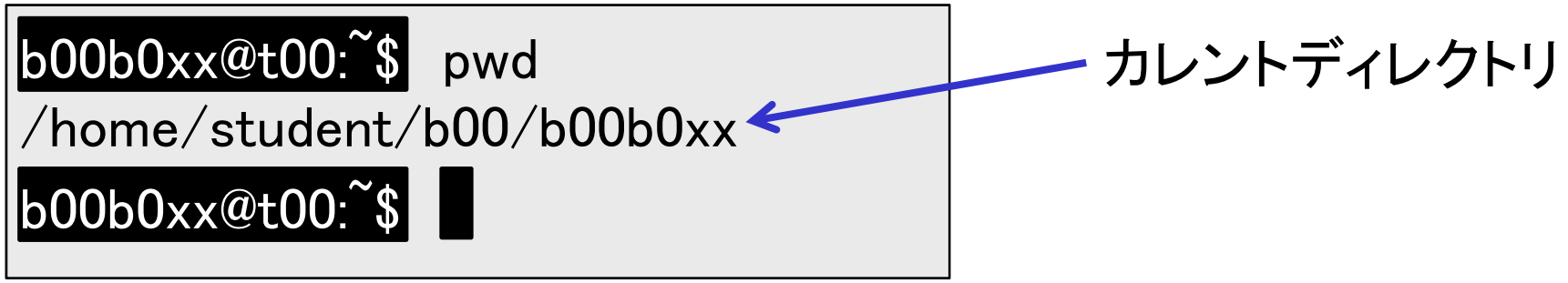


注意：
パスワード入力時は何も表示されないので
（「＊＊＊」も表示されない）、慎重に入力すること

パスワードの付け方

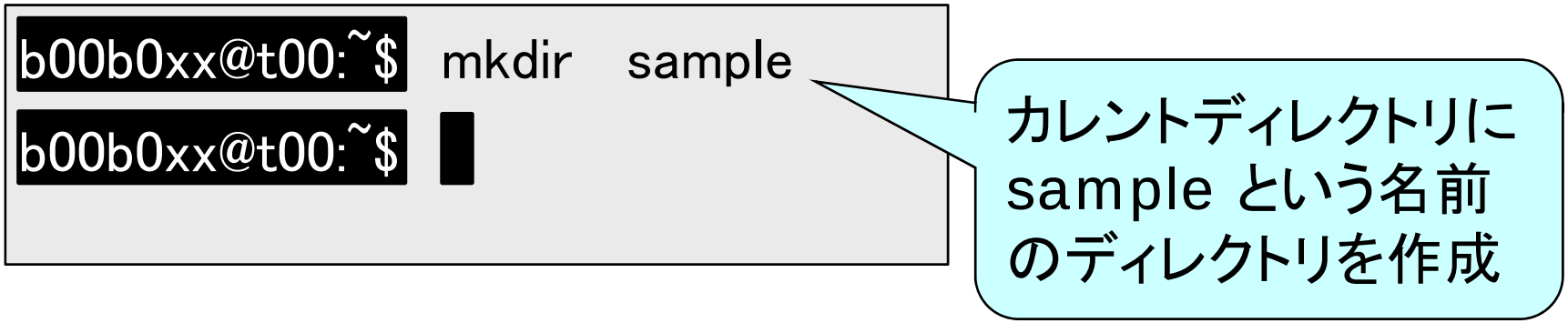
- 英文字の大文字・小文字・記号・数字を必ず混ぜて使うこと
- ６文字以上とすること
- ユーザ名（学籍番号）と同一の文字列を含んではならない
- 氏名、生年月日、車のナンバー、電話番号、誕生日などを含んでいてはならない

カレントディレクトリ



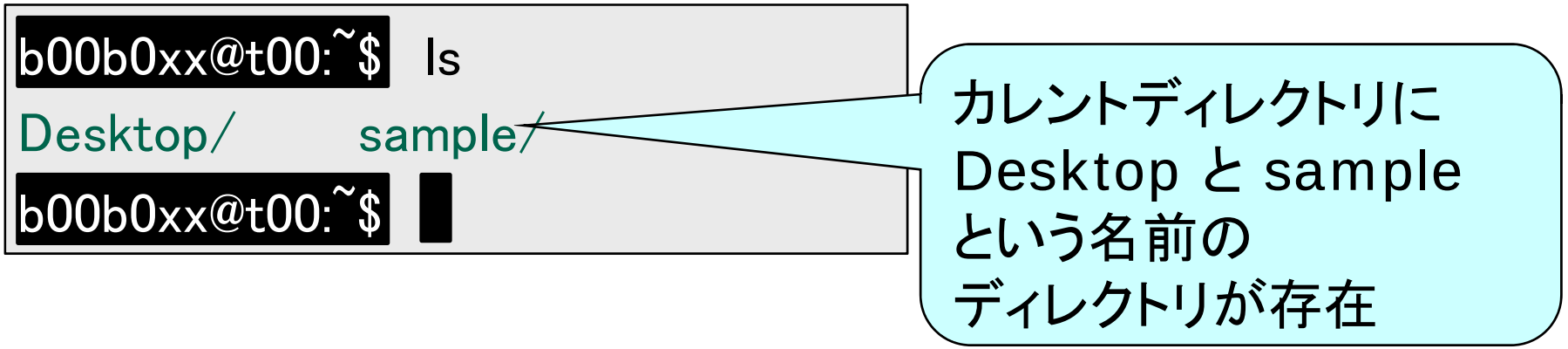
- カレントディレクトリ：今いるディレクトリ
- 特に指定しない限り、多くのコマンドは
  カレントディレクトリにあるファイルを操作
- pwd コマンドでカレントディレクトリを表示
- ウィンドウ毎に異なるので注意

ディレクトリの作成



- mkdir コマンドで新しいディレクトリを作成
- mkdir コマンドの引数に指定した名前のディレクトリを、カレントディレクトリの中に作成する

ディレクトリ内容の表示



- ls コマンドでカレントディレクトリにある
  ファイルやディレクトリなどの一覧を表示
- ファイルやディレクトリの種類を
  色や記号で区別
- 作業前後に実行する癖を付けておくと良い

カレントディレクトリの変更

```
b00b0xx@t00:~$ cd sample
b00b0xx@t00:~/sample$ pwd
/home/student/b00/b00b0xx/sample
b00b0xx@t00:~/sample$
```

カレントディレクトリが確かに変更されている

- cd コマンドの引数に指定したディレクトリにカレントディレクトリを変更
- カレントディレクトリの変更に伴い、コマンドプロンプトの表示も変わる

cd コマンドの特別な使い方

```
b00b0xx@t00:~/sample/test$ pwd
/home/student/b00/b00b0xx/sample/test
b00b0xx@t00:~/sample/test$ cd ..
b00b0xx@t00:~/sample$ pwd
/home/student/b00/b00b0xx/sample
```

cd .. で「ひとつ上のディレクトリ」に移動

```
b00b0xx@t00:~/sample/test$ pwd
/home/student/b00/b00b0xx/sample/test
b00b0xx@t00:~/sample/test$ cd
b00b0xx@t00:~$ pwd
/home/student/b00/b00b0xx
```

引数を付けずに cd を実行すると、いつでもホームディレクトリへ移動

その他の有用なコマンド（一例）

- lv　：　ファイルの内容を表示
- cp：ファイルのコピー
- mv：他のディレクトリへのファイルの移動、ファイル名の変更
- rm：ファイルの消去
- rmdir：ディレクトリの消去
- man：コマンド使用法（マニュアル）の表示

プログラミングにおけるバージョン管理

- 複数の開発者による共同作業で
  プログラムを作成できる
- 過去のファイルの内容を調べることが
  できる（以前の内容に戻せる）

作業ディレクトリの作成

```
b00b0xx@t00:~$ svn checkout http://...../svn/b00b0xx/prog
b00b0xx@t00:~$ ls
Desktop/      prog/      sample/
b00b0xx@t00:~$ cd prog
b00b0xx@t00:~/prog$ ls
01/  03/  05/  07/  09/  11/  13/  S2/
02/  04/  06/  08/  10/  12/  S1/
b00b0xx@t00:~/prog$ cd 01
b00b0xx@t00:~/prog/01$ ls
comment.c
```

作業ディレクトリと、
その中身が、
カレントディレクトリに
自動的に作られる

- 本演習で使うリポジトリのアドレスは
  http://dav.ec.h.akita-pu.ac.jp/svn/ユーザ名/prog

毎日の準備

```
b00b0xx@t00:~$ ls
Desktop/        prog/        sample/
b00b0xx@t00:~$ cd  prog
b00b0xx@t00:~/prog$ svn  update
リビジョン 1 です。
b00b0xx@t00:~/prog$ cd  01
b00b0xx@t00:~/prog/01$ ls
comment.c
```

作業ディレクトリの中身が
最新の状態に更新される

- その日の作業を始める前に
  作業ディレクトリで svn update を実行

Emacs の起動
（既存ファイルの編集）

```
b00b0xx@t00:~/prog/01$ ls
comment.c
b00b0xx@t00:~/prog/01$ emacs  comment.c  &
b00b0xx@t00:~/prog/01$ ls
comment.c
```

comment.cの内容を編集できる

- コマンドの引数として、ファイル名を指定する（各種プログラミング支援機能が利用できるようになる）
- 最後に「＆」を付けること

Emacs の起動
（新規ファイルの作成）

```
b00b0xx@t00:~/prog/01$ ls
comment.c
b00b0xx@t00:~/prog/01$ emacs  comment2.c  &
b00b0xx@t00:~/prog/01$ ls
comment.c  comment2.c
```

カレントディレクトリに
comment2.c が
自動的に作成される

- コマンドの引数として、ファイル名を指定する（各種プログラミング支援機能が利用できるようになる）
- 最後に「＆」を付けること

課題の提出

```
b00b0xx@t00:~/prog/01$ ls
comment.c    comment2.c
b00b0xx@t00:~/prog/01$ svn   add   comment2.c
A   comment2.c
```

提出すべきファイルが正しいディレクトリにあるか必ず確認

提出するファイルであることを示すマークを付ける

- 新しく作ったファイルを提出したいときは、svn add コマンドでマークを付ける
- すでにマークが付いているときは、そのままで良い

課題の提出（続き）

```
b00b0xx@t00:~/prog/01$ cd
b00b0xx@t00:~$ cd prog
b00b0xx@t00:~/prog$ svn commit
送信しています 01/comment.c
追加しています 01/comment2.c
ファイルのデータを送信しています . .
リビジョン 2 をコミットしました。
```

カレントディレクトリを作業ディレクトリに変更

エディタが起動するので、ログメッセージを入力して終了。

- svn commit コマンドで、マークが付いていて変更があったファイルを全て提出
- ログメッセージには、何を解決したかをわかりやすく書く（作業報告）
- 失敗したら再度 svn update してから

提出の確認

```
b00b0xx@t00:~/prog$ svn update
リビジョン 2 です。
b00b0xx@t00:~/prog$ svn log -r '{2009-04-10}'
------------------------------------------------------------------------
r2 | b10b0xx | 2009-04-09 17:40:00 +0900（木, 09 1月 2009）
第一回演習基本課題の提出
サンプルファイルの日付と名前、学籍番号などを修正した。
------------------------------------------------------------------------
```

作業ディレクトリの中身を
最新の状態にしてから

- svn log -r '{締切日}'
  で、指定した締切日より前に、最後に提出した提出日時とログメッセージを確認
- svn log
  で、全てのログメッセージを確認

提出内容の確認

```
b00b0xx@t00:~/prog$ cd 01
b00b0xx@t00:~/prog/01$ svn cat -r '{2009-04-10}' comment.c
/*
  作成日:2009年4月09日
  作成者:本荘　てまり
  学籍番号:b00b0xx
...
```

- svn cat -r '{締切日}' ファイル名
  で、提出された時点でのファイルの内容
  (採点対象になる内容)を確認

ヒントの確認

```
b00b0xx@t00:~/prog$ svn update
U  01/comment.c
A  01/README
リビジョン 3 に更新しました。
b00b0xx@t00:~/prog$ cd 01
b00b0xx@t00:~/prog/01$ lv README
◎　コメントを表す「/*」と「*/」は1バイト文字でなくてはいけません。
b00b0xx@t00:~/prog/01$ lv comment.c
/* TODO : 日付を最終更新日に直してください。*/
/*
...
```

svn update でリビジョンが更新されたときはヒントあり

- READMEというファイルやソースファイル上に教員からのヒントが記述される

コンピュータの2つの側面

ソフトウェアとハードウェア

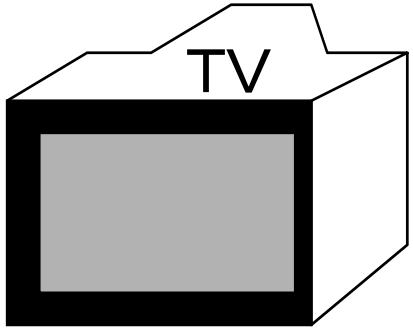
ソフトウェア

プログラム

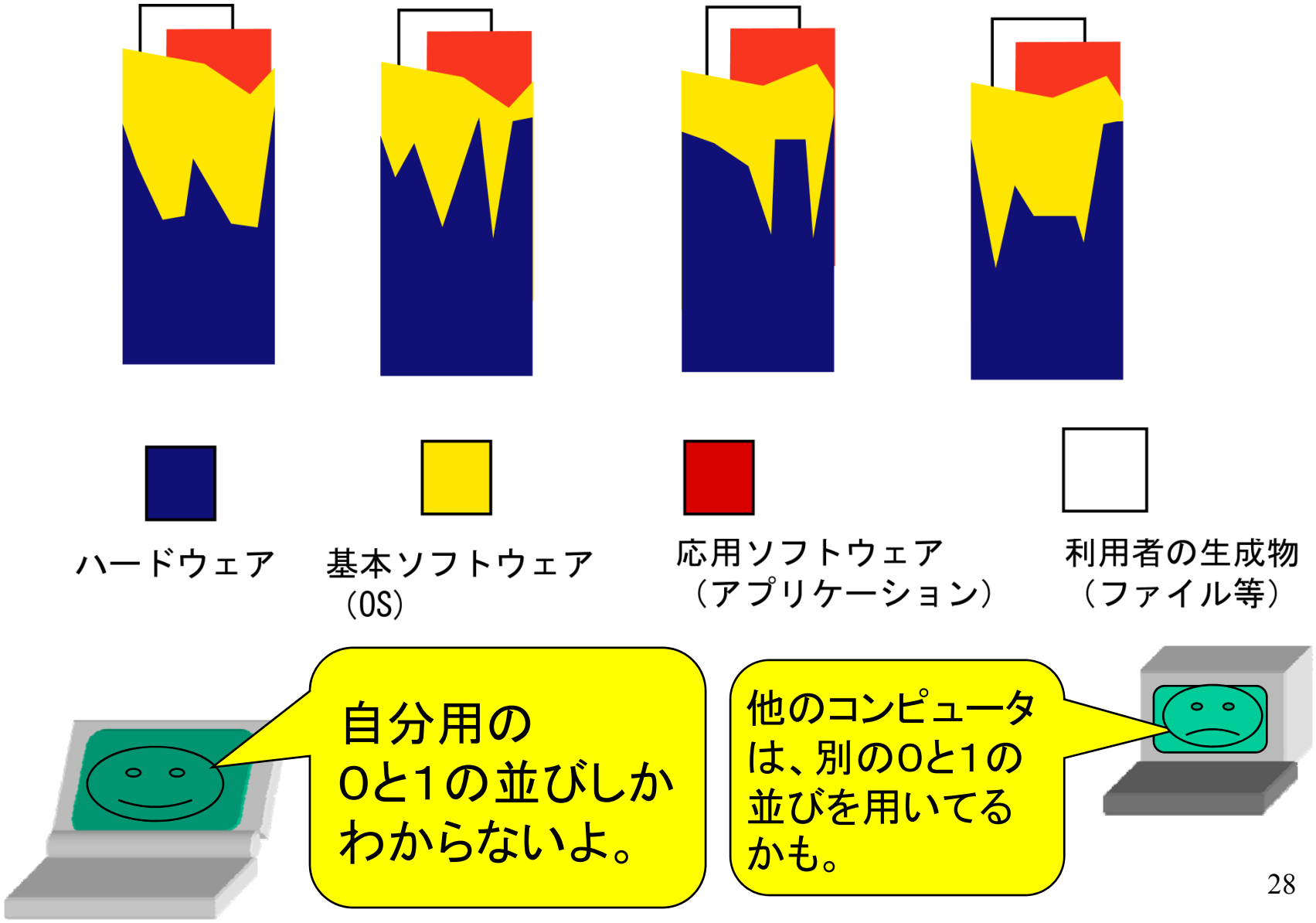
番組

思考、記憶

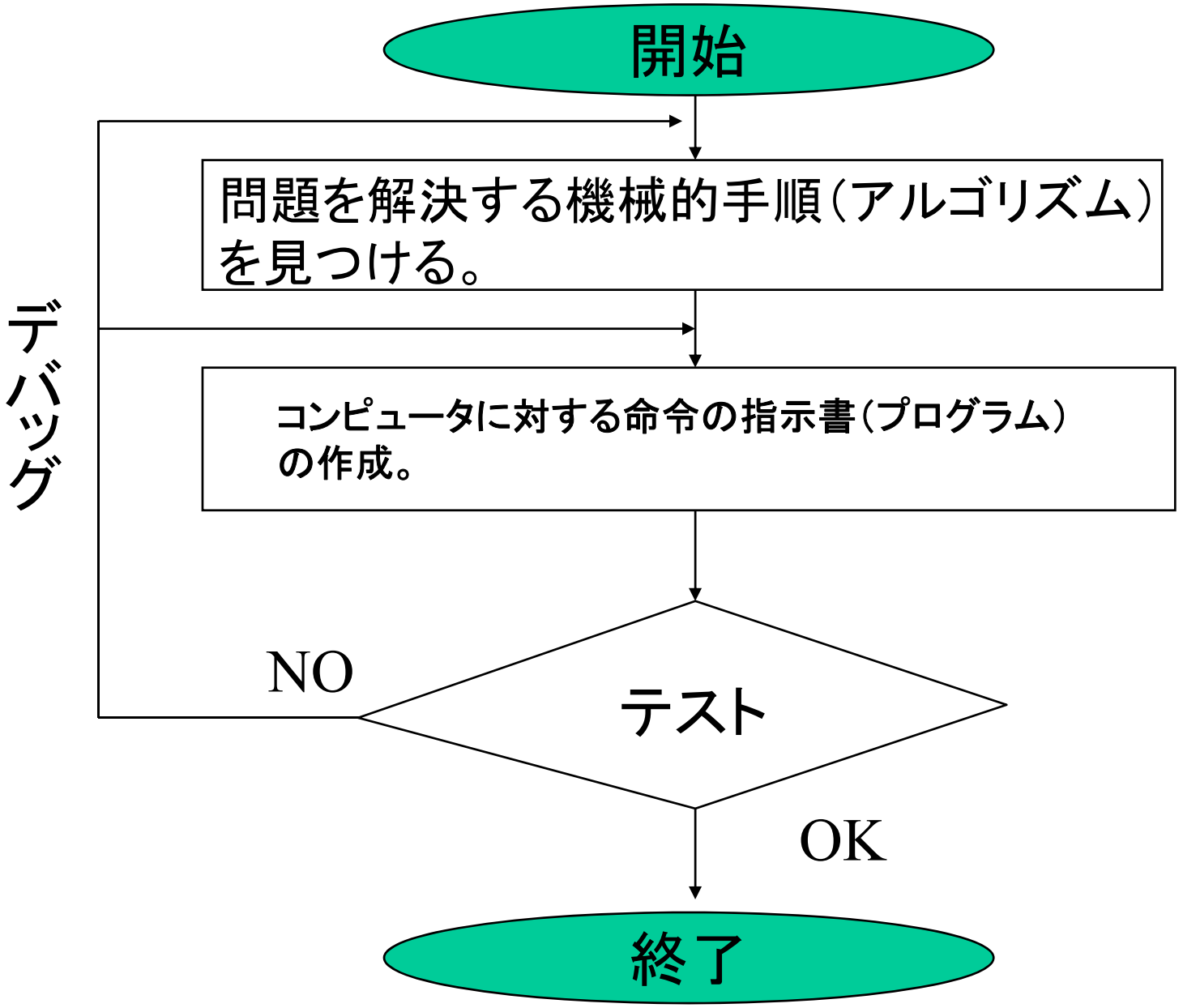
ハードウェア



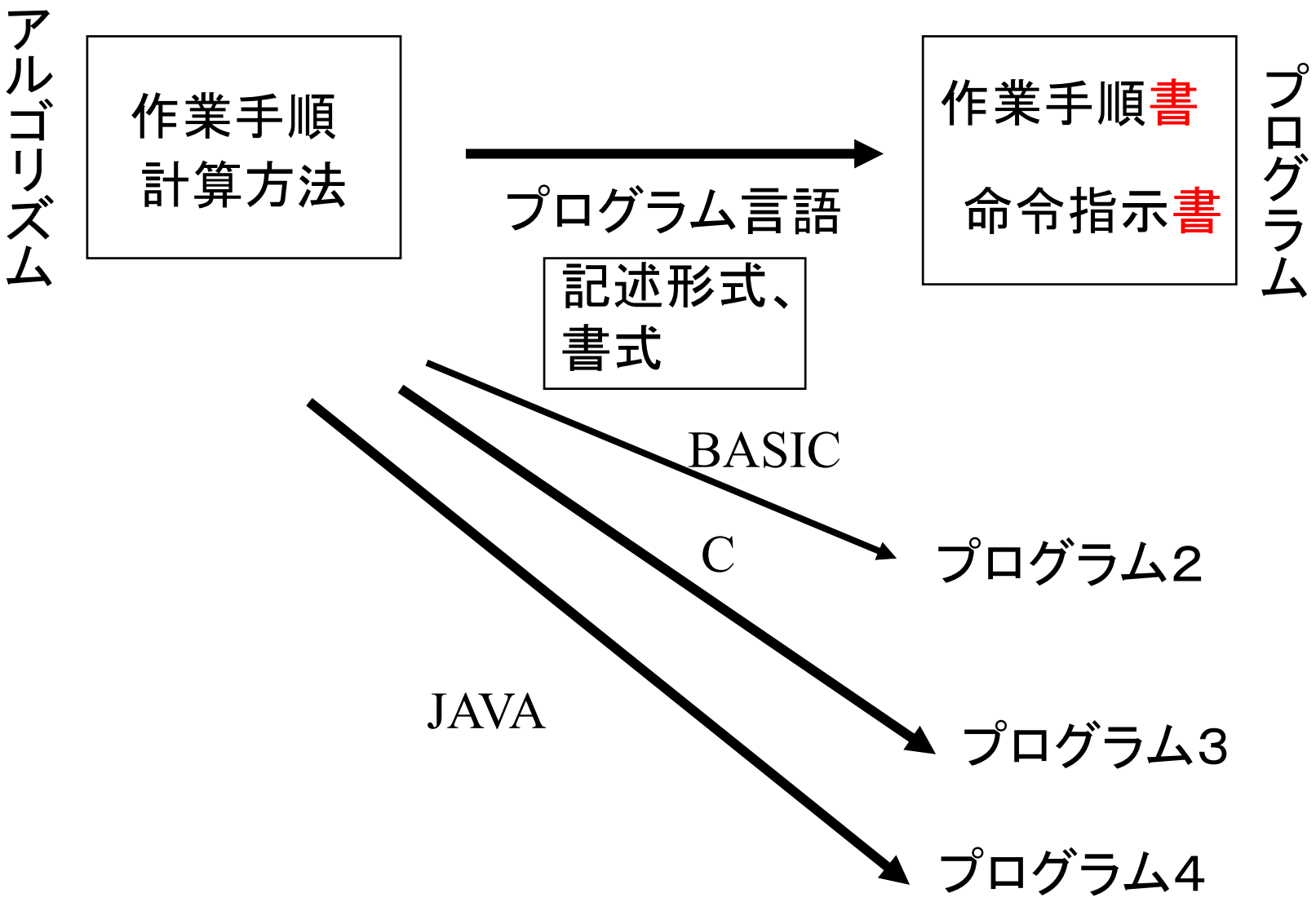
ハードウェアとソフトウェアの階層構造



問題解決のためのソフトウェアの開発

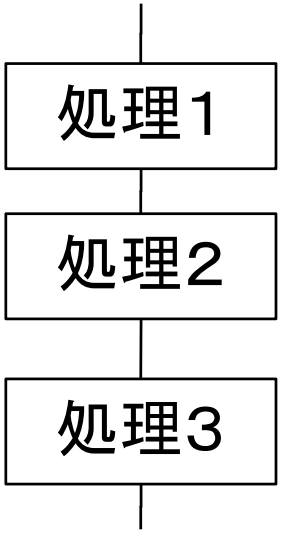


アルゴリズムとプログラム

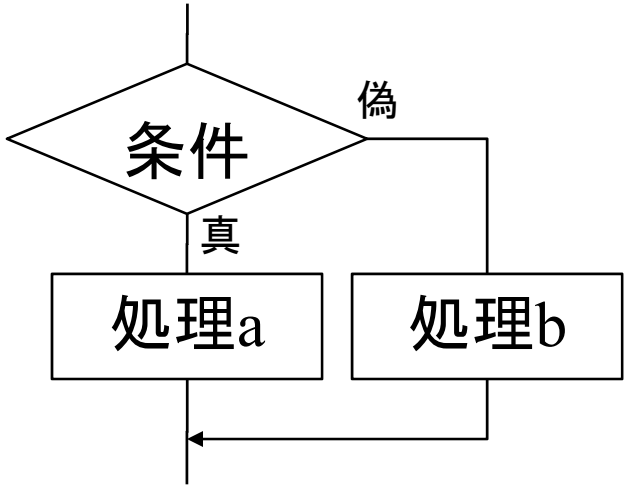


アルゴリズムの基本要素

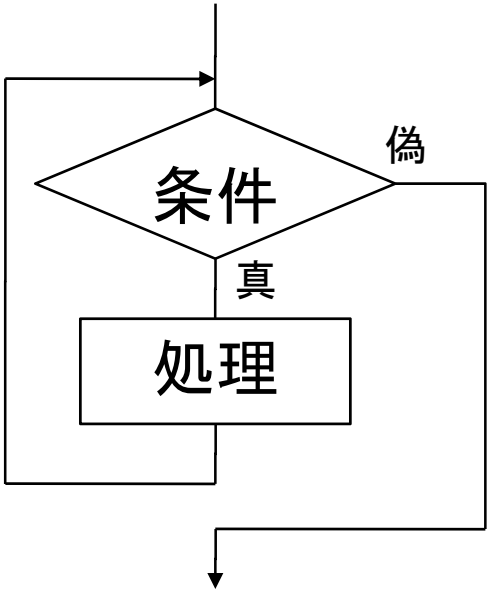
1. 決められた順番にいくつかの処理を行う （順次）
2. 条件判断により二つの処理のどちらかを行う （選択）
3. ある処理を繰り返し何度も行う （反復）



順次



選択



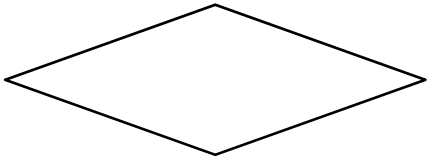
反復

フローチャートによる手順の記述

フローチャートの記号

端子（手順の開始・終了）

処理の基本単位



条件判断

他のフローチャートで
定義された手順

処理の流れ（通常は上から下へ）

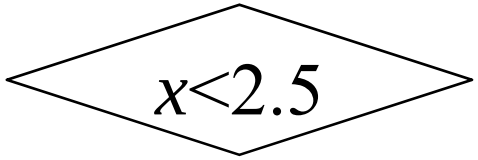
処理の基本単位

- １つの値（定数）を変数に代入
- １つの変数と1つの定数の二項演算（＋，－，×，÷）の結果を変数に代入
- ２つの変数の二項演算の結果を変数に代入

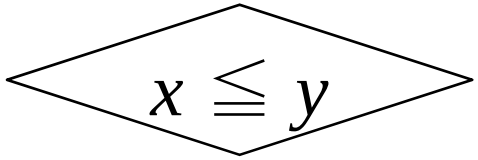
$x \leftarrow 2.5$	$x \leftarrow x+1$	$x \leftarrow y+z$
変数 x に 2.5を代入	変数 x の値をそれまでより１増やす	変数 x に $y+z$ を計算した値を代入

条件判断

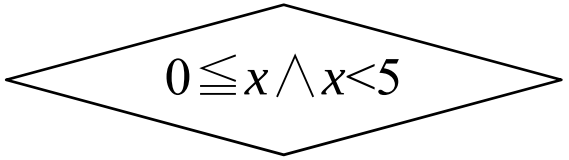
- １つの値（定数）と変数の一致・大小比較
- ２つの変数の値の一致・大小比較
- 上記の論理式（かつ「∧」、または「∨」、否定「¬」）による組み合わせ



変数 x の値が
2.5未満ならば真

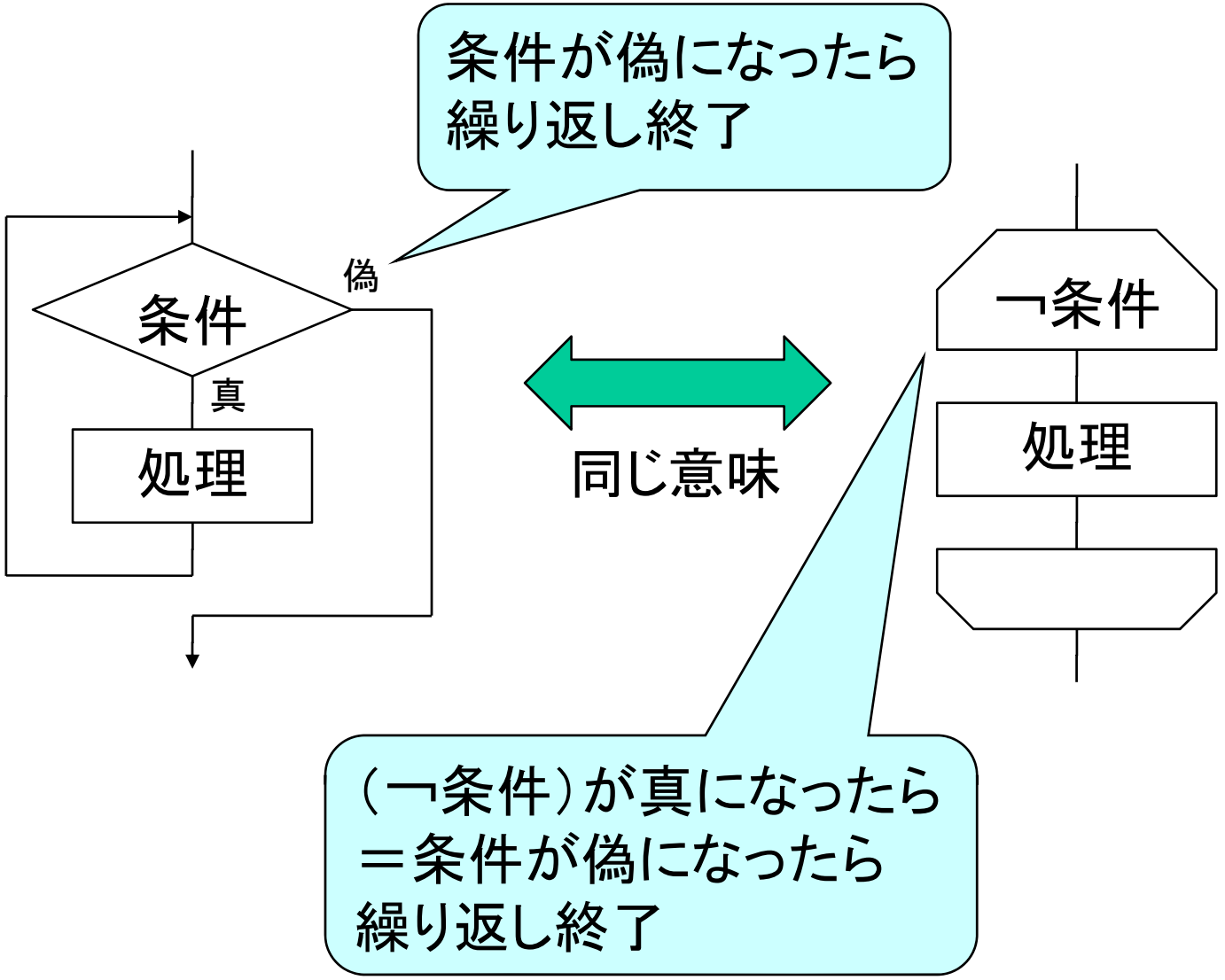


変数 y の値が
x 以上ならば真



変数 x の値が
0以上5未満
（0～4）ならば真

反復処理の省略記法

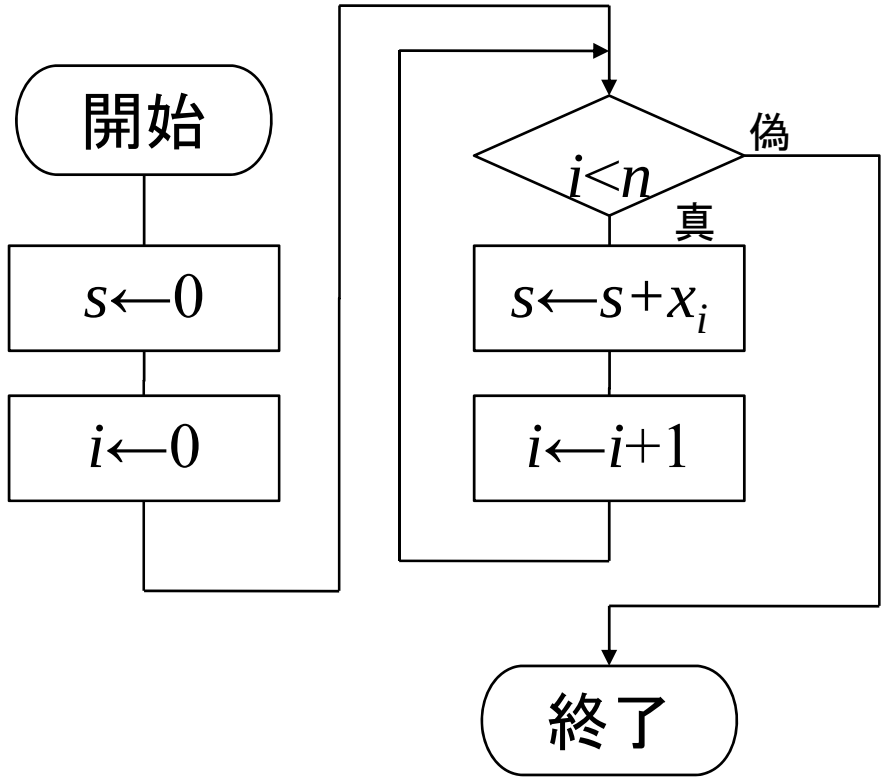

条件が偽になったら
繰り返し終了
偽
条件
真
処理
同じ意味
¬条件
処理
(¬条件)が真になったら
＝条件が偽になったら
繰り返し終了

フローチャートの例

- n 個の要素からなる数列 $X=x_0,x_1,\ldots,x_{n-1}$ の要素の総和を求めるアルゴリズム

入力:
要素数n
数列要素 $x_0,x_1,\ldots,x_{n-1}$

出力:
総和 $s=\sum_{i=0}^{n-1} x_i$



正しいアルゴリズム

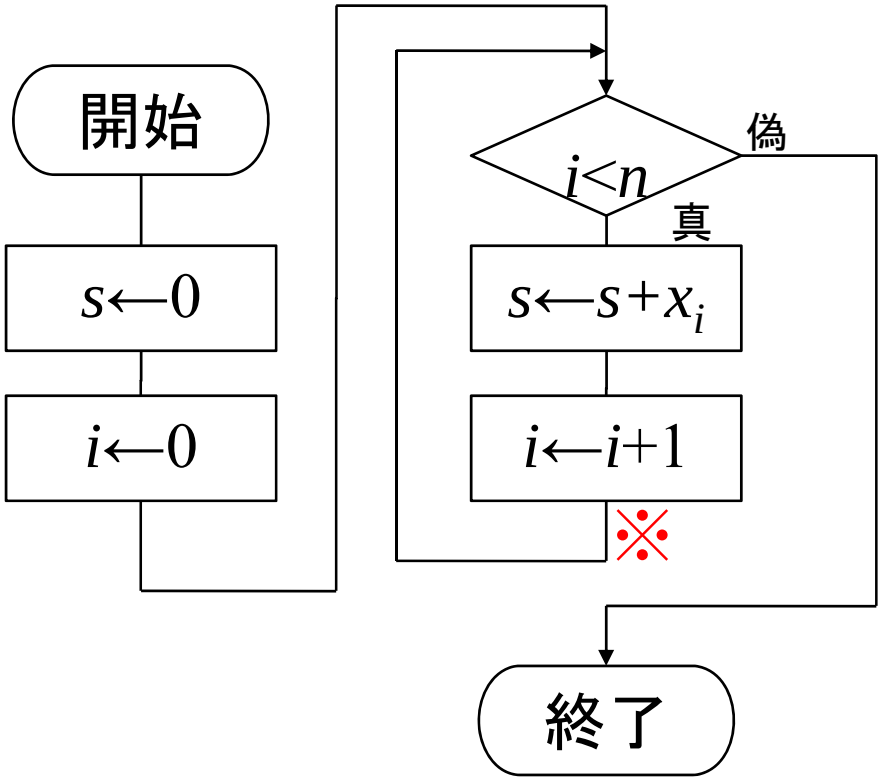
- アルゴリズムが正しいことを数学的に証明することができる

右図※の箇所で

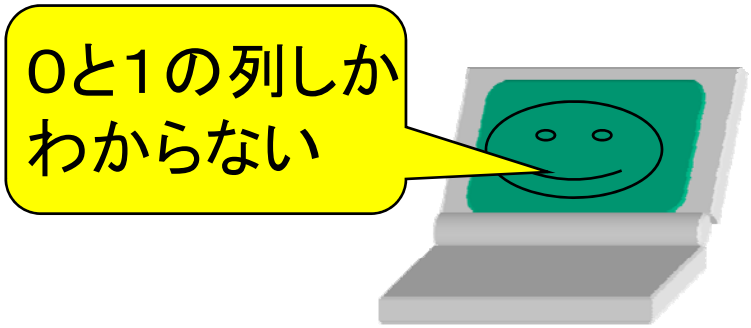
$$s = \sum_{j=0}^{i-1} x_j$$

が常に真。

ループ不変条件：
数学的帰納法を使って
証明してみよう



プログラミング言語の分類
（高級言語と低級言語）



コンピュータ側

機械語

アセンブリ言語

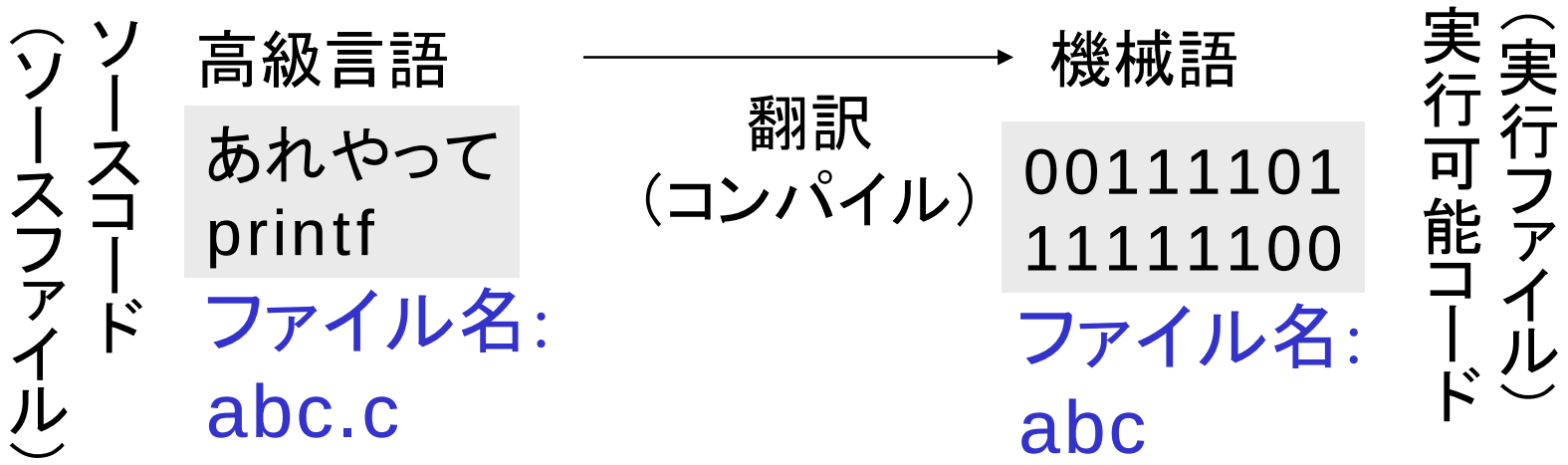
低級

Java
C言語
LISP
FORTRAN

人間側

高級

コンパイラ

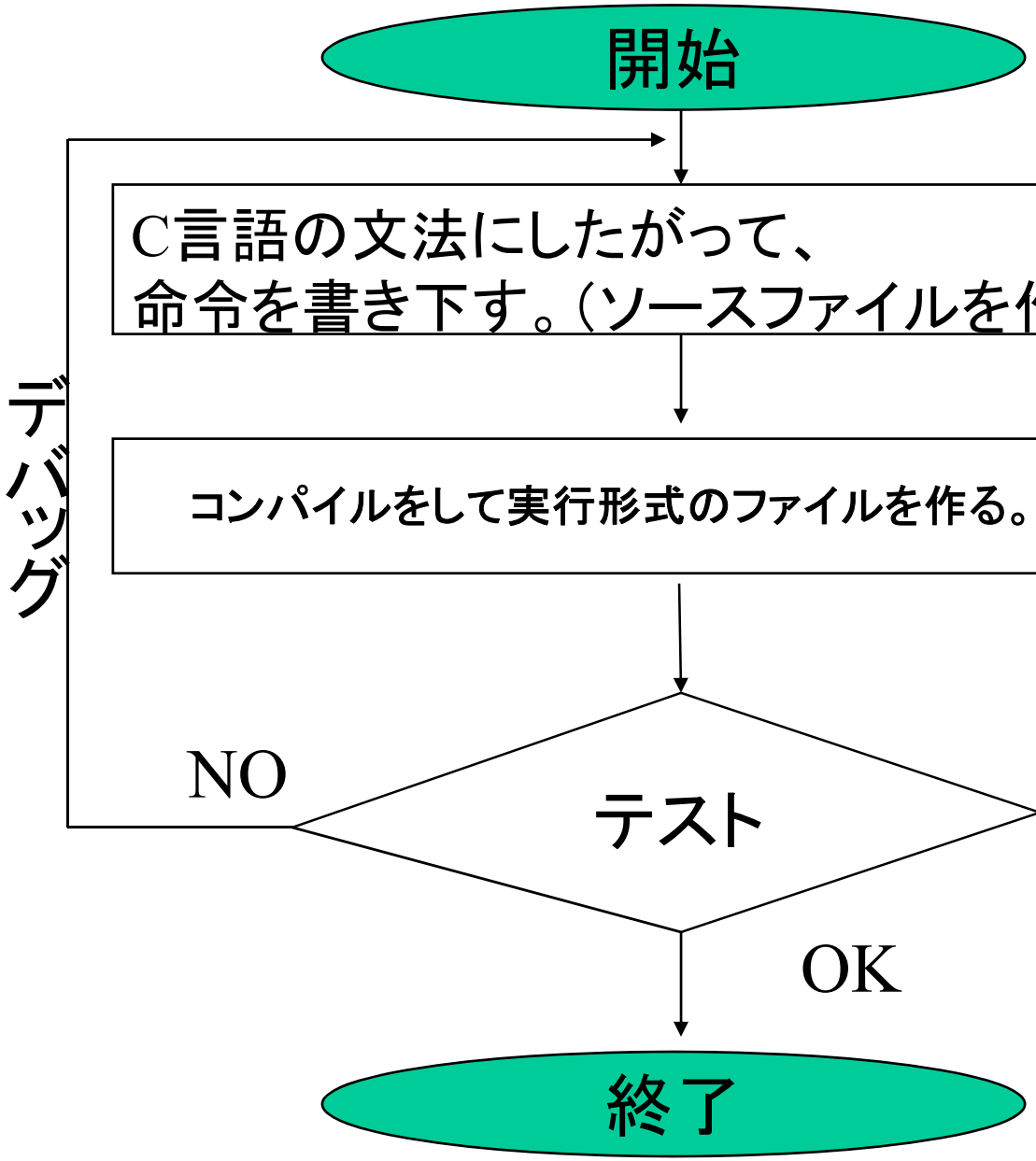


コンパイラ:

高級言語（例えばC言語）から、低級言語（機械語）へ翻訳（コンパイル）するソフトウェア。

注意：C言語では、ソースファイルは
＊＊＊＊＊.c
という名前にする。（拡張子が".c"のファイルにする。）

C言語でのプログラムの作り方



	使うもの	できるもの
C言語の文法にしたがって、命令を書き下す。（ソースファイルを作る。）	エディタ	ソース（abc.c）
コンパイルをして実行形式のファイルを作る。	コンパイラ	実行形式（abc）
テスト	シェル（OS）	いろいろ（プログラム次第）

良いソフトウェアの基準

- 速い

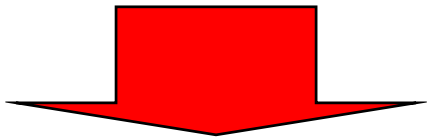
同じことするなら速い方がいいでしょ。

- 強い

どんな入力でもきちんと動作してほしいでしょ。

- 分かりやすい

誰がみても理解しやすいほうがいいでしょ。



本演習では、独自のスタイル規則に沿って
プログラミングしてもらいます。（ガイダンス資料参照）