

ICT リテラシー (情報技術論) A

-- 第 13 回 : アルゴリズム --

栗野 俊一

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く
禁じます

2022/12/12 ICT リテラシー (情報技術論) A

伝言

私語は慎むように !!

□ 席は自由です

- できるだけ前に詰めよう
- コロナ対策のために、ソーシャルディスタンスをたもとう

□ 色々なお知らせについて

- 栗野の Web Page に注意する事

<http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino>

- google で「kurino」で検索

今後の予定

ICT リテラシー (情報技術論) A

今後の予定

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

今後の予定

□ 今後の予定(後ろから)

○ 15 回目：試験を行う (試験は年明けになる)

- ▶ オンライン試験を予定している(自宅から受ける)
- ▶ 都合がわるい場合は連絡をすれば別の日時に行く(詳しくは次週説明)
- ▶ 栗野もオンラインで質問対応で待機するが教室には来ない予定

○ 14 回目(次回)：データベース (Text p.81, 6.3 節)

- ▶ 試験に関する説明行う

○ 13 回目(今回)：アルゴリズム (Text p.77, 6.2 節)

前回 (第 12 回) の復習

ICT リテラシー (情報技術論) A

前回 (第 12 回) の復習

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

前回 (第 12 回) の復習

□ 前回 (第 12 回) の復習 : オペレーティングシステム

- ソフトウェア : CPU の命令を記述したもの (狭義/広義にはハードウェアへの指示を記述したもの)

 - ▶ ソフトはハード上で動き、ハードが実現するサービスに柔軟性を与える

- OS (オペレーティングシステム)

 - ▶ ハードを管理、操作するためのソフト (cf. アプリはハードを使って何かをするソフト)

 - ▶ OS はソフトのプラットフォーム ($M*N \rightarrow M+N$) : アプリは OS に対して作られる

- OS の種類 : unix 系と MS-Windows 系がある

- OS の機能

 - ▶ ユーザインタフェース : コンピュータの操作性を定める

 - ▶ ソフトとハードの仲介 : ハードウェアの相違を吸収する (プラットフォーム)

 - ▶ 記憶管理 : キャッシュ/メモリ/ハードディスク等を管理 (速度の差)

 - ▶ プロセス管理 : CPU の管理

 - ▶ ユーザ管理 : 利用者の管理

- ビジネスにおける OS の位置づけ

 - ▶ MS の戦略 : アプリを握るには、プラットフォームの OS を握れ

 - ▶ デファクトスタンダード : 標準規格の重要性

今週 (第 13 回) の概要

ICT リテラシー (情報技術論) A

今週 (第 13 回) の概要

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

今週 (第 13 回) の予定

□ 今週 (第 13 回) の予定

○ 講義：アルゴリズム

- ▶ アルゴリズム
- ▶ プログラム (Text p.77 6.2 節)
- ▶ プログラミング言語とは (Text p.77 6.2.1 節)
- ▶ プログラムの内部動作 (Text p.78 6.2.2 節)
- ▶ 高級言語の基本処理 (Text p.79 6.2.3 節)

今週 (第 13 回) の目標

□ 今週 (第 13 回) の目標

○ コンピュータによる問題解決手法について学ぶ

- ▶ アルゴリズムとは何か
- ▶ プログラムとは何か

今週 (第 13 回)

- 前回 (第 12 回) の課題
 - Web Class「小テスト-12」
- 今週 (第 13 回) の課題
 - Web Class「小テスト-13」

アルゴリズム

ICT リテラシー (情報技術論) A

アルゴリズム

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

アルゴリズム

□ アルゴリズムとは

○ ある「問題」を解く アルゴリズム の定義

- ▶ 確定性：明確な手順の有限な列で表現されている
- ▶ 正当性：その「問題」を解く(解を求める)事ができる
- ▶ 停止性：有限時間で終了する

○ 問題に対して、アルゴリズムが与えられれば、

- ▶ アルゴリズム(の示す手順を適用する事)により、答を得る事ができる
- ▶ 例: 公式(計算の手順が与えられる)、ユークリッドの互除法(最大公約数の求め方)、筆算、10進2進変換

○ 数学的に問題を「解く」事

- ▶ 問題の「答えを得る」事ではなく、「答えを得る手段(アルゴリズム)を得る」事
- ▶ アルゴリズムがあれば、(答えを知らない)問題の答えが得られる

□ 知識の構造

○ 問題は What : 問題の答となるものを定義

○ アルゴリズムは How to: 問題の答となるもの求める手段

□ コンピュータに答を求めさせるには、アルゴリズムが必要

○ アルゴリズムが明確でない問題をコンピュータにやらせるのは難しい..

○ Deep Learning は、「(ある種の拡張された)アルゴリズムを求める」アルゴリズム

- ▶ Deep Learning は正当性(問題定義)の点で、課題(正確性)を抱えている

プログラム

ICT リテラシー (情報技術論) A

プログラム

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

プログラム

□ プログラム (Text p.77 6.2 節)

○ プログラムとは

- ▶ 定義：アルゴリズムをコンピュータに扱える(実行できる)ように表現した物
- ▶ プログラムをコンピュータに与えると、アルゴリズムを実行してくれる

○ ソフトウェア

- ▶ 特定なシステム(CPU/OS上)で、実行可能なプログラム

プログラミング言語

□ プログラミング言語とは (Text p.77 6.2.1 節)

○ プログラムを記述するために作られた人工的な言語

- ▶ ハード(電気回路)でも、アルゴリズムが表現可能
- ▶ 柔軟性を高めるために、ソフト(プログラム)で、機能を追加

□ プログラミング言語の分類

○ 手続型：何(What)を、どうするか(How to) という処理手順を記述する

- ▶ 例：C++, Java, Python, etc..
- ▶ 特徴：CPU の命令に対応する指示を直接指定できるので、効率が良い

○ 非手続型：手続型以外のプログラミング言語

- ▶ 特徴：手順の記述が不要なので、プログラム書き易いが、(手順がないので)非効率な事が多い
- ▶ 関数型：問題の解を求める関数の定義を行う(例：Lisp, ML)
- ▶ 論理型：問題の解が満す条件を指定し、解を求めさせる(例：Prolog, SQL)

色々なプログラミング言語

□ 低級言語 : CPU 依存する

- 機械語 : CPU への命令の並び (CPU 毎に異なる / 2 進表現)

 - ▶ アセンブリ言語 : 機械語の命令とほぼ 1 対 1 で表現可能 (文字列表現)

□ 高級言語 : CPU と独立 (言語処理系 [翻訳/通訳 を行うプログラム] が必要)

○ 手続型

 - ▶ Fortran : 最初の高級言語 (科学技術計算に利用) / Basic : Fortran 教育用簡易版

 - ▶ Cobol : 商業計算用

 - ▶ Algol (Pascal) : アルゴリズム記述用 (データ構造)

 - ▶ C 言語 : Unix OS を記述されるために、設計 (free な Unix と一緒に広く利用される)

 - ▶ C++ : Object 指向型 / Java : 仮想 CPU の実装 (OS から独立) / Python : ライブラリが多く、Deep Learning で利用

 - ▶ Javascript : Java とは名前が似て居るが、別物 (HTML と併用される)

- 関数型 Lisp : シンボル処理言語 (人工知能のアセンブラ / ラムダカリキュラス)

- 論理型 Prolog : 論理プログラミング言語 / SQL : 関係データベースを操作する言語 (DB 専用言語)

□ マークアップ言語 : プログラミング言語ではなく、Content を記述する言語

- HTML : Web Page 記述 (javascript と組み合わせる事により、機能を持つページが作

プログラムの内部動作

□プログラムの内部動作 (Text p.78 6.2.2 節)

○CPU が理解できるプログラムは機械語のみ

- ▶メモリ上に記録されている命令も機械語の命令 (2進数)
- ▶チューリングマイシと同じ(ノイマン型)なので、命令とデータの区別がない
- ▶CPU にとっては都合がよいが、人間には分かり難い (低級言語)

□言語処理系

○(機械語以外の) 言語で記述されたプログラムを実行できるようにするプログラム

- ▶コンパイラ(翻訳系) : 機械語に変換 (翻訳) する / 一度に変換 / 変換がおわれば不要
- ▶インタープリター(通訳) : 機械語に変換 (通訳) する / 逐次変換 / いつでも必要

高級言語の基本処理

□ 高級言語の基本処理 (Text p.79 6.2.3 節)

○ 手続き型言語(C++,Java,Python) によるプログラム記述

- ▶ 変数操作(入力,出力,代入,参照)を命令に基本として、その手順を記述
- ▶ 操作手順の記述(プログラム)がアルゴリズムを表現する

□ 基本操作

○ 変数：名前がついた記憶領域(データを記録できる)

- ▶ 名前を指定して、その値を取り出す(参照)や、値を記録させる(代入)が可能

○ 代入：変数に値を記録するように指示する事 (値は計算された結果になる)

○ 参照：変数名を指定して、その変数に記録されている値を取り出す事 (値は何度、参照しても変わらない)

○ 式 (四則計算)：代入する値を「計算式」の形で表現すると、その「式」の値が計算される

- ▶ 式で利用可能な演算子は言語によって異なるが、四則は (+, -, *, /) が共通で利用できる

□ 制御構造：命令の実行の可否や、繰返し等を指示する構文

○ if 文(条件判断)：条件によって、指定した命令を実行したりしなかったりを表現する

○ for 文(繰返し)：条件によって、指定した命令を繰り返すか止めるかを表現する

おしまい

ICT リテラシー (情報技術論) A

おしまい

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます