コンピュータ概論 A/B

-- TeX Macro --

(TeX Programming の基礎)

数学科 栗野 俊一 (TA: 浜津 翔 [院生 2 年])

2014/11/18 コンピュータ概論

伝言

私語は慎むように !!

- 席は自由です (出席パスワード : 20141118)
  - できるだけ前に詰めよう
  - 教室にきたら直ぐにやる事
    - PC の電源 On / ネットワーク接続 / Web を参照する / skype を起動する
- 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
- 次週(2014/11/24)は「補習」はありません
  - 2014/11/24(月)は前日の勤労感謝の日の振替休日です
    - しかし、「休日授業実施日」なので通常の講義はあります
    - それでも、「補習」はしません(ごめんなさい..)
- Local Server
    - 10.9.209.128 (VNC) : 画面の操作を見ることができます (PW : vnc-2014)
    - 10.9.209.88 (Web) : 普段のサーバより速いはずです

前回(2014/11/11)の内容

- 前回(2014/11/11)の内容
  - システム：(興味)の「対象」と対象の「操作」からなる (cf. 親と子の関係)
    - 「操作」を学ぶ事により「対象」を望みの形にする事が可能になる
  - メタシステム：「操作」を「操作の対象」にするシステム (cf. 祖父母と親の関係)
    - 「操作」を操作する事により、「間接的」に「対象」を操作する
    - 「間接的」なので「難しい」が、「とても強力」な仕組
  - 計算機に於けるメタシステム：対象はデータで、操作はソフト
    - プログラミング：ソフトを「作る」作業 (メタシステム)
  - Mathematica Programming (2)
    - 自分独自の「関数」が定義できる
    - 式で定義された関数：「f[x_] := x^2」とすると、関数「f(x)=x^2」が定義できる
    - 漸化式で定義された関数：「g[1]:=a」、「g[n_] := g[n-1]*r」とすれば、等比数列
  - Mathematica で「数」を作る
    - ペアノの公理に基く「自然数」と「四則」の実現

本日(2014/11/18)の予定

- 講義
  - 木構造
  - ファイルシステム
  - TeX のマクロ
- 実習
  - [演習 1] TeX の復習
  - [演習 2] TeX のマクロの作成
  - [演習 3] 課題の作成

本日の課題 (2014/11/18)

- 前回 (2014/11/11) の課題
  - 次のファイルを提出しなさい
    - 表題 : ペアノの方法による「有理数の差」の関数 qsub を定義しなさい
    - ファイル名 : 20141118-QQQQ.nb (QQQQ は学生番号)
    - 詳しくは、配布した nat.txt の内容を参照
- 今回 (2014/11/18) の課題
  - 次のファイルを提出しなさい
    - 表題 : 自分の名前のロゴマクロ
    - ファイル名 : 20141118-QQQQ.sty (QQQQ は学生番号)
    - 詳しくは、配布した sample-20141118.sty の内容を参照

木 (Tree)

- 構造(グラフ)
  - いくつかのノード(点/要素)をアーク(線/関係)で結んだもの
- 木(き)/木構造(もくこうぞう)
  - 階層を表す構造の一つで、次の様に再帰的に定義される
    - 単独のノードは木である(この木の根は、この単独のノード自身である)
    - 新しいノードと複数の木の根を結んだ物は木である(根は新しいノード)
    - 上記の二つの規則で作られた物だけが木である
  - cf.「自然数の定義」(ペアノの公理の一部)
    - 1 は自然数である
    - n が自然数ならば、n+1 も自然数である
    - 上記の二つの規則で作られた物だけが自然数である
  - 木に関連する用語
    - 祖先/子孫：根は他の要素の先祖になる、その逆の関係が子孫
    - 親/子：自分と直接接続している祖先は親、その逆の関係が子
    - 上/下：親が上、子が下 (計算機の「木」は根を上に書く)
    - 根/枝/葉：親を持たないノードが根、子を持たないノードが葉、その他が枝

木の例

(a)　(b)　(c)

- (a) : 根が一つだけの木
- (b) : 根が一つ、葉が一つで、枝のない木
- (c) : 葉が四つ、枝が三の木

木でない例

(A)

(B)

- (A)：継がってない部分がある(森の例)
- (B)：輪ができている(木には、輪ができない)

ファイルシステム

- ファイルシステム
  - 外部記憶上のデータの管理方法の事
    - ファイルをどのように指定するかの方式
  - Windows 7 のファイルシステムは NTFS
  - 情報の単位はファイル
- NTFS は階層型のファイル管理を行っている
  - 外部記憶装置の一番大きな単位は、ドライブ(外部記憶装置)
    - SSD, HD, DVD-Drive, USB Memory, etc ..
  - ドライブ内は木構造
    - ドライブは一文字 (ドライブ・レター) で表す
    - ハードディスクは「C」となっている (が、決まっているわけではない)
  - 要素(根以外は名前をもつ)は二種類ある
    - ディレクトリ(フォルダ) : 子を持つことができる
    - ファイル : 子は持てない(必ず葉になる)
  - 根の事を「ルート(root)」と呼ぶ事が多い

ファイルシステムの例

- ノードはディレクトリ(○)とファイル(□)の二種類
  - ファイルは葉にしかならない (子ができない)
    - 空のディレクトリも作れ、葉になる。
  - 個々のノードには名前がついている

パス(path)名

- □ パス(path)名
  - ○ ファイルを指定する名前
    - ▹ 絶対パス : どこに居ても同じ物を指すが長い
    - ▹ 相対パス : 居る場所(カレントディレクトリ)によって異なる物を指すが短い
- □ 絶対パス(path)名
  - ○ ディレクトリ/ファイルを一意に示す識別子
    - ▹ 根からディレクトリ名を「￥」で継げたもの (かならず「￥」で始まる)
    - ▹ 根からそのノードまでの経路(path)を示す
    - ▹ 根自身のパス名は「￥」
  - ○ 絶対パス名の例
    - ▹ サンプル TeX ファイル:「C:￥usr￥tex￥20141118￥sample-20141118.tex」

相対パス名

- □ カレントディレクトリとカレントドライブ
  - ○ 現在の注目位置を表現する(パスで表現される)
    - ▷ コマンドプロンプトで表示される文字列
    - ▷ アドレスバーに表示される文字列
  - ○ カレントディレクトリは、ドライブ毎に存在する
    - ▷ cd コマンドで変更(移動)可能 (cd 移動先のフォルダのパス名)
  - ○ カレントドライブの変更
    - ▷「ドライブレター」+「:」(例「D:」で、カレントドライブを D に変更)
- □ 相対パス
  - ○ カレントディレクトリからみた経路
    - ▷ かならず「￥」以外で始まる
  - ○ 基本：カレントディレクトリ + ￥ + 相対パス = 絶対パス
    - ▷ カレントディレクトリが異れば同じ相対パスでも異なるファイルを指す
  - ○ 応用
    - ▷「.」はカレントディレクトリを表す
    - ▷「..」は親ディレクトリを表す(根の親は根になっている)
    - ▷ 相対パスを利用して、従兄弟ファイルを指定することもできる

パス名の例

- 状況：TeX の利用中
  - 例 1 (子/子孫)
    - カレントディレクトリ: C:￥usr￥tex
    - 絶対パス: C:￥usr￥tex￥20141118￥sample-20141118.tex
    - 相対パス: 20141118￥sample-20141118.tex
  - 例 2 (親)
    - カレントディレクトリ: C:￥usr￥tex￥20141111
    - 絶対パス: C:￥usr￥tex￥hello.tex
    - 相対パス: ..￥hello.tex
  - 例 3 (兄弟/従兄弟)
    - カレントディレクトリ: C:￥usr￥tex￥20141118
    - 絶対パス: C:￥usr￥tex￥20141111￥sample-20141111.tex
    - 相対パス: ..￥20141111￥sample-20141111.tex
- パス名の利用
  - 異るファイルはパス名が異るのでファイル名が同一でも区別できる
  - パス名はファイルの位置も表現しているので、そこに行ける

TeX の復習(1)

- pLaTeX システムとは(What)
  - tex 形式の文章ファイル(*.tex)を typeset(整形) して dvi 形式ファイルを作るソフト
    - dvi 形式のファイルは更に dvipdfmx で pdf 形式に変換できる
    - 一度、pdf 形式にすれば、表示も印刷も綺麗にできる
- TeX を利用する理由(Why)
  - 印刷物(「内容」と「形式」の二つの情報から成る)の作成を「内容」だけに専念したい
    - 「形式」は、TeX が対処してくれる(形式は style ファイルの形で別に記述されているので、その御仕着せに頼る)
  - tex 形式は基本 Text 形式である
    - 「編集」を「Text Editor(サクラ)」で行いたい / 高速にかつ汎用で、しかも、手慣れている
    - pLaTeX がなくても、ある程度、中身が理解できる (e-mail での利用)
  - 色々なシステムで動く
    - MS-Windows は勿論、Linux (栗野愛用)や、MacOS でも利用可能
  - 数式を使いたい
    - 数式を扱いたければ、(残念な事に、未だに..) 一番便利

実習 2: pLaTeX の利用 (復習)

- [実習 2] pLaTeX の利用
  - tex ファイルの作成 (tex-000.tex) : Text エディタ(サクラエディタ)を使う
    - 拡張子は「tex」にする
    - sample file をダウンロードしてもよい
  - platex による typeset
    - コマンドプロンプトを利用して「platex tex-000.tex」とする
    - 問題がなければ、「tex-000.dvi」(dvi 形式ファイル)が作られる
  - dvipdfmx による pdf 形式への変換
    - コマンドプロンプトを利用して「dvipdfmx tex-000.dvi」とする
    - 問題がなければ、「tex-000.pdf」(pdf 形式ファイル)が作られる
  - pdf ファイルの表示
    - コマンドプロンプトを利用して「tex-000.pdf」とする
    - 普通に、tex-000.pdf

TeX のマクロ機能

- □ TeX のマクロ機能 (tex-001.tex)
  - ○ マクロ機能
    - ▹ TeX では、「『文字列』に『名前』を付けて、参照する」事ができる
  - ○ マクロ名：文字列につけらえた名前
    - ▹ 「名前」は習慣により「\(バックスラッシュ)」で始まる物にする
  - ○ マクロ定義：『文字列』に『名前』を付る事
    - ▹ マクロ定義の表現：\newcommand{新しい名前}{文字列}
  - ○ 例：\newcommand{\MyJapaneseName}{栗野俊一}
    - ▹ 文字列「栗野俊一」に、マクロ名「\MyJapaneseName」をつけた
    - ▹ マクロ「\MyJapaneseName」を、定義した(マクロ定義した)
    - ▹ 文字列「栗野俊一」は、マクロ「\MyJapaneseName」の定義内容
  - ○ マクロ展開(参照)
    - ▹ マクロ定義済のマクロ名を記述すると「その定義内容」に置き換わる
    - ▹ 例：「私の名前は\MyJapaneseName です」→「私の名前は栗野俊一です」

Style ファイルとマクロの分離

- マクロは何度も使い回す事が多い
  - 毎回同じ事を書くのは面倒 / 一度作成した内容を使い回す
    - マクロ定義だけを記述した Style ファイルを作成して読み込む (\input)
- マクロ集としての style ファイル
  - TeX の特徴の一つ：内容と形式が分離できる (内容だけに専念できる)
    - 内容は .tex ファイルに記述。では、形式は .. ? 実は style ファイルに
  - 色々な style ファイル
    - 中に、色々なマクロが定義されている
    - 内容で、同じ名前のマクロを利用しても、マクロの内容が異れば、異なる結果になる
    - 「形式」の情報を「マクロ」にしておくことにより、形式と内容を分離できる
  - \usepackege マクロ：実は、style ファイルを読み込んでいる

pLaTeX vs TeX

- pLaTeX vs TeX
  - 今迄、pLaTeX は TeX の一種と言っていたが..？
    - 実は、pLaTeX は TeX そのもの / 単に、pLaTeX 専用のマクロが事前に定義されて(読み込まれている..)だけ..
  - じゃあ、なぜ別の名前？
    - マクロの量が多く、また、利用形式が独特なので、まるで違ったようにみえる
    - 違う物は違う名前の方が混乱がすくない(cf. 役者はドーランを塗ったら別人)

色々な TeX マクロ

- 表示位置を変更するマクロ
  - 文字の位置を上に上げる
    - \raise<サイズ>\hbox{<文字列>}
    - 例: \raise.4ex\hbox{野} : 「野」の表示位置を x の高さ(ex)の 0.4 倍だけ上に
  - 文字の位置を下に下げる
    - \lower<サイズ>\hbox{<文字列>}
    - 例: \lower.2ex\hbox{俊} : 「俊」の表示位置を x の高さ(ex)の 0.2 倍だけ下に
  - 文字と文字の間隔を調整する
    - \kern<サイズ>
    - 例: 俊\kern.1em一 : 「俊」と「一」の間を m の幅(em)の 0.1 倍だけ広げる
    - サイズに負の数を指定すると幅を狭める事もできる

引数付きマクロの定義

- □ 引数付きマクロ
  - ○ マクロの内容を一部後から指定したい場合がある
    - ▷ ほとんど同じだが、一部だけ異なる文字列を何度も利用したい場合
    - ▷ 引数付きマクロによって対応可能
- □ 引数付きマクロの定義(三つのポイント)
  - ○ 引数付きマクロの場合は、引数の個数を「[n] (n は個数)」で指定する
    - ▷ 例 : \newcommand{\ThreeArgMacro}[3]{ .. } : 引数は三つある
  - ○ 内容の中に「#1 ～ #n」を含める
    - ▷ 例 : 上記で { 最初は #1, 次は #2, 最後が #3 } とする
  - ○ マクロを利用する時にマクロ名の後ろに n 個の引数を指定する
    - ▷ 例1 : \ThreeArgMacro{芋虫}{蛹}{蝶}
    - ▷ 例2 : \ThreeArgMacro{おたまじゃくし}{足が生えてきて}{蛙}
    - ▷ 例3 : \ThreeArgMacro{ちょろちょろ}{ぱっぱ}{子供が泣いても蓋とるな}

メタ, 関数定義, マクロ定義

- □ 計算機操作におけるメタ
  - ○ ファイル(対象),ソフト(操作)
    - ▹ メタは？: インストール / プログラム作成
- □ プログラミング
  - ○ 計算機操作におけるメタの典型
    - ▹ Mathematica の関数 / TeX のマクロ
- □ コンピュータ概論の*裏*目標
  - ○「計算機に使われる」のではなく「計算機が使える」人材の育成
    - ▹ ソフトを使うだけの人 (実は、ソフトに使われている人：×)
  - ○ プログラミングができる人
    - ▹ ソフトを作る人：つまり、「計算機が使える」人
  - ○ 講議では「概念」だけを述べている
    - ▹ 自分で色々試してみよう (結局、何時ものセリフ)

実習 3: TeX マクロ

- [実習 3.1] マクロを利用する
  - tex-001.tex の \MyJapaneseName の定義を変更して試してみる
- [実習 3.2] style ファイルの作成
  - tex-001.tex の \MyJapaneseName の定義を切取り、別のファイル(myinfo.sty) に保存
  - tex-001.tex のプリアンブルに \input{myinfo.sty} を追加
    - tex-002.tex を DL してもよい
- [実習 3.3] 名前ロゴの作成
  - tex-003.tex を参考にロゴを作るマクロを作成する
- [実習 3.4] 引数付きマクロの作成
  - tex-004.tex を参考に引数付きマクロを作ってみる
- [実習 3.5] 課題の作成
  - 自分ロゴのマクロを作る
    - sample-20141118.tex, sample-20141118.sty を参考に sample-20141118.sty を作る
    - sample-20141118.tex を typeset して、きちんと、pdf が自分の情報になっている事を確認