情報処理演習II

Literacy

情報処理演習II

- 1年生前期
- 目標：プログラム作成の基本的な実践力
  - プログラミング論Ⅰと連携
- 使用プログラミング言語：Scheme (後述)
- 単位取得：
  - 講義時の平常点
  - 全レポート提出
  - 必要に応じて試験を行うことがある

Literacy

なぜプログラミングを学習するか

- 今やコンピュータは社会基盤の一部
  - コンピュータソフトウェアに関する素養は必須
- コンピュータはデータを形式的(機械的)に処理
  - 特定の言語によらず、データおよびその処理手順(プログラム)を論理的に記述する能力は重要
- テクニックよりまず基本的な考え方を重視
  - 問題分析、問題解決の技術

Literacy

プログラム設計法(レシピ)

プログラム設計レシピ：問題解決過程のガイド

1. 問題の解析とデータの定義
2. 規約、目的、結果の記述とヘッダ作成
3. 例題を用いた振舞の例示
4. プログラムテンプレート、レイアウトの作成
5. テンプレートを完全な定義に変換
6. テストを通してエラーの発見

各ステップで中間生成物をチェック

Literacy

ソフトウェア開発と言語

- ソフトウェア開発(抽象レベルでは言語非依存)：
  - 物理システムの開発と同様に、きちんとした抽象化、設計法
  - 物理法則のような制約はない → 論理的正しさが特に重要
- 具体的実現：目的に適した言語(モデル)を選択
  - モデル：式、関数 ／ 手続き、副作用 ／ オブジェクト指向 等
  - 目的：科学技術計算、事務処理、システムソフト、DB など
- 本演習：基本的「式、関数」をベースとした言語
  - Scheme という言語を使う

Literacy

Scheme の特徴

- 関数型：数学体系λ計算が基礎
  - 式(関数)でプログラムを構成、式の簡約で計算が進む
    - 紙と鉛筆(頭の中)でもプログラム実行過程が追える
- インタプリタ型処理系
  - 初学者にも優しい(本質が埋もれがちな作業が少ない)
- 見た目としては演算子が前置され、括弧が多い
  - プログラムの基本単位である式を括弧で囲む
  - 式の構成要素にまた式を持ち得る(括弧の入れ子)

例 $4\times2+4\div2$ の表記とその計算過程：
(+ (* 4 2) (/ 4 2))→(+ 8 (/ 4 2))→(+ 8 2)→10

Literacy

Scheme についての参考資料

■参考書

– [この講義] M. Felleisen et al., “How to Design Programs”, MIT Press (Web版 http://www.htdp.org/)

– [Scheme言語仕様] ケント ディヴィグ (村上雅章訳), "プログラミング言語SCHEME", ピアソン・エデュケーション

– [言語として Scheme を用いる他の著名な講義用テキスト] サスマン他 (和田英一訳), "計算機プログラムの構造と解釈", ピアソン・エデュケーション

Literacy

情報処理基礎演習
(リテラシも含め合計約13回程度の予定)

■ 前半:データに着目したプログラム開発の基本
- 数と数式、プログラムと実行過程、エラー
- 変数と関数、条件、プログラム設計法によるプログラム作成
- 様々なデータ
- 任意長データと処理(リストと再帰処理プログラム)

■ 後半:プログラミングに関するより高度な話題
- 設計の抽象化
- 値としての関数を使った抽象化
- 再帰のバリエーション
- 知識の蓄積(アキュムレータ)

■ コンピュータリテラシ(メイル、ワープロ...)

Literacy

他のプログラミング関連科目との関係

– 1年前期
- プログラミング論 I (必修、箱崎開講)
  – プログラム設計の基本:講義中心
- 情報処理基礎演習(必修、六本松開講)
  – 端末を利用した演習中心

連携 内容的には言語非依存だが便宜上Schemeを使用

– 1年後期
- プログラミング演習 I (必修、箱崎開講)
  – C言語(手続型言語)でのプログラミング
- (高度プログラミング演習 (選択、六本松開講))
  – C言語の演習

– 2 年以降 → 各課程ごとの内容

Literacy

第1回 (リテラシ)
インターネット、電子メール、WWW

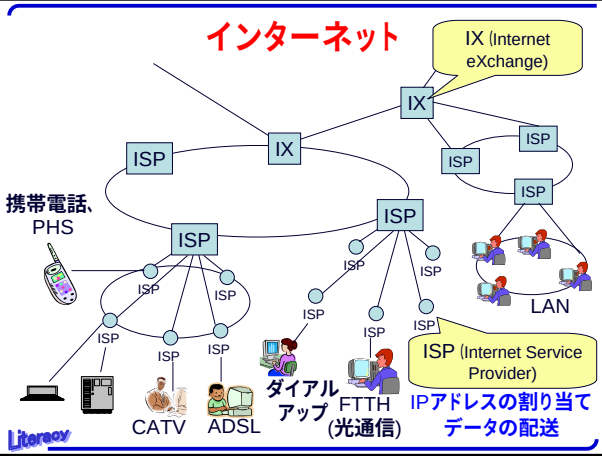
Literacy

今日の内容

1. インターネットとは？
2. 電子メールの利用
3. World Wide Web (WWW)
4. 検索エンジン
5. 今日の課題

Literacy

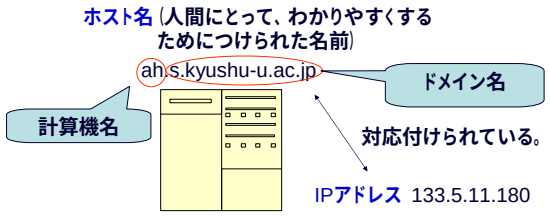
インターネット



Literacy

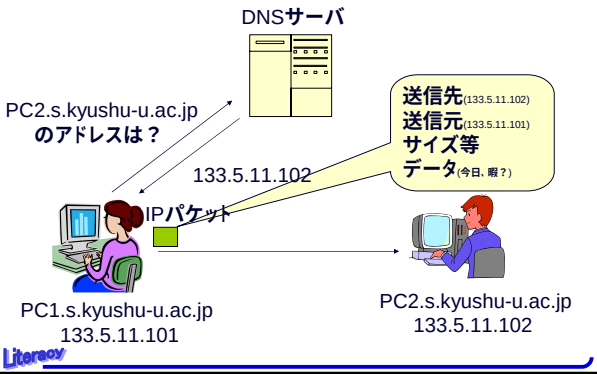
IPアドレス

- インターネット上に接続している機器を識別するためのもの



Literacy

ドメインネームシステム (DNS)



Literacy

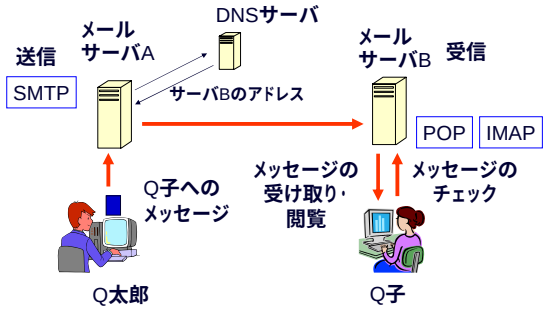
電子メールの利用

- クライアント (Mailer)
- メールサーバ
  - POP, IMAP (受信用プロトコル)
  - SMTP (送信用プロトコル)
- DNSサーバ
  - ドメイン名 → IPアドレス (変換)
  - ah.s.kyushu-u.ac.jp → 133.5.11.180
  - s.kyushu-u.ac.jp
    - (Mail eXchanger) -> vwall.nc.kyushu-u.ac.jp

プロトコル：あるサービスを実現するための、決められた手続き

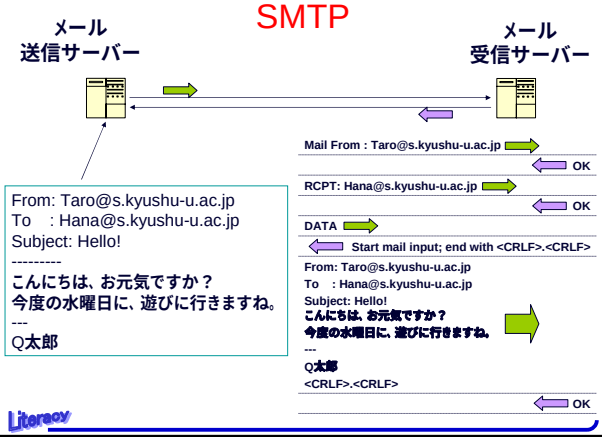
Literacy

電子メールの送受信



Literacy

SMTP



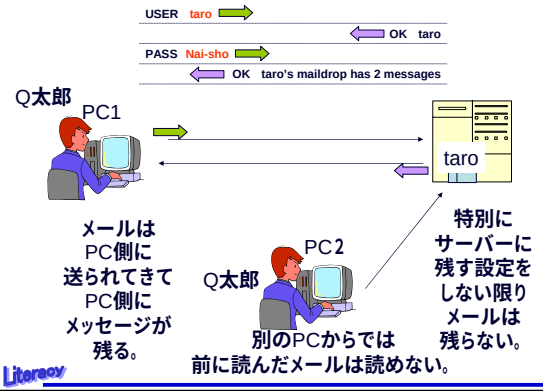
Literacy

SMTPで送信されるメッセージは？

- メッセージは平文 (そのままの表現) で送られる。
- 文字以外のデータを添付ファイルなどで送る場合は、MIME (Multipurpose Internet Mail Extension)という規格に従ってコード化して送る。
- 暗号化しない限り、大事な情報 (クレジットカードの番号、暗証番号など) は、メールで送らない。

Literacy

POP



Literacy

POPもメッセージは暗号化されていない

- POP
  - USER taro
  - PASS Nai-Sho!

  は暗号化されていない。

  APOPを利用すると、USERとPASSは暗号化
  - USER taro → kd3xqaP)
  - PASS Nai-Sho! → %Qk8#4qxkp1

  しかし、メッセージは暗号化されていない。

Literacy

経路の暗号化

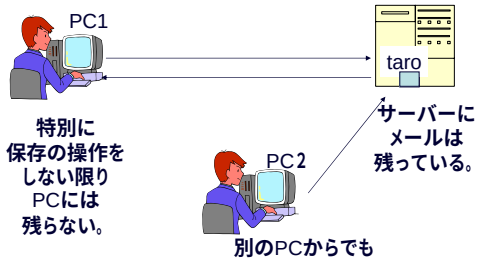
http (Webのプロトコル) + SSL = https

Grace Mailのページのアドレスは
https://secure.s.kyushu-u.ac.jp/mail/

この手法はメーラとメールサーバ間の通信を暗号化するだけで、すべての通信が暗号化されるわけではないので注意!!

Literacy

IMAP



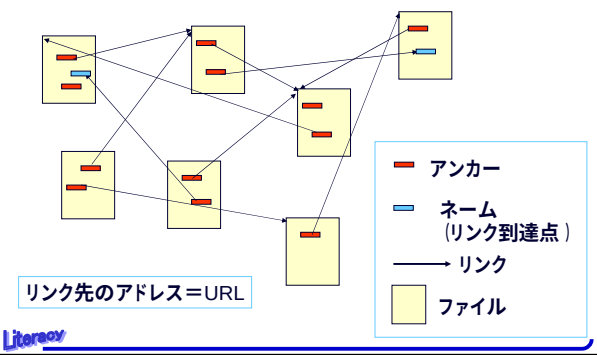
Literacy

World Wide Web (WWW)

- 1989. 3
  - Tim Berners-Leeが最初のプロジェクトプロポーザルを書く。
- 1990. 11
  - WWWの最初のプロトタイプ作成
- 1991. 3
  - Line mode browserの最初のリリース
- 1993. 1
  - Midas, Viola, Mosaicなどのbrowserがリリース
  - サーバ数は50
- 2005.11
  - サーバー数: 74,572,794 (http://news.netcraft.com/より)
  - Google: ページ数は87億以上、
    Yahoo: ページ数は113億以上

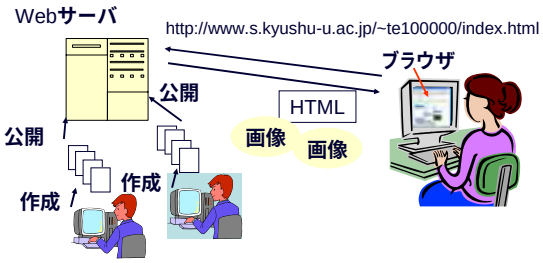
Literacy

World Wide Webの特徴



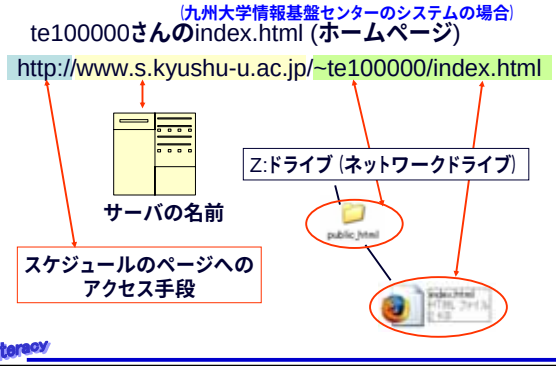
Literacy

Webサーバーとブラウザ



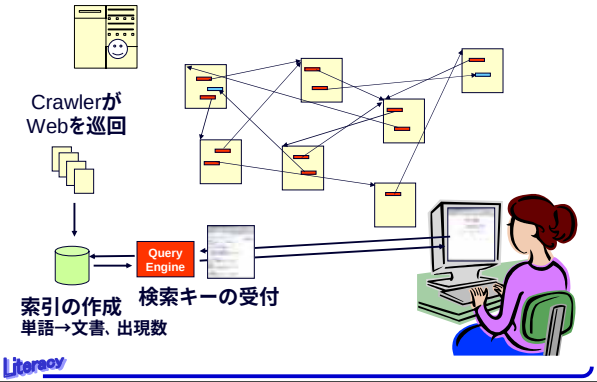
Literacy

URL(Uniform Resource Locator)とディレクトリ・ファイル



Literacy

検索エンジン



Literacy

1. Scheme の式とプログラム

数、式、プログラムと実行過程、
エラー

Literacy

例題

Literacy

例題1. 簡単な式

- 次の Scheme の式を DrScheme の実行用ウインドウに入力し、実行する

$$(2+2)*\frac{(3+5)*(30/10)}{2}$$

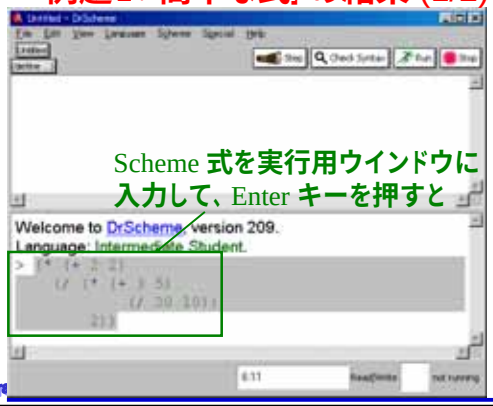
- Scheme で書くと:

```
(* (+ 2 2)
   (/ (* (+ 3 5)
         (/ 30 10))
      2))
```

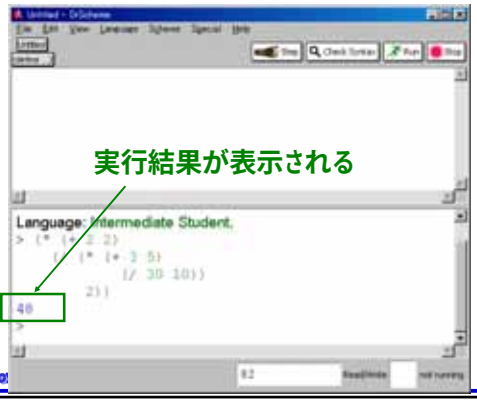
Scheme の式
演算子を前に置く
(前置記法)

Literacy

「例題1．簡単な式」の結果 (1/2)



「例題1．簡単な式」の結果 (2/2)



よくある間違い

- 「スペース（空白文字）」に意味がある

間違いの例 1

```
(* (+2 2)
   (/ (* (+ 3 5)
         (/ 30 10))
      2))
```

＋の後にスペースが無い

間違いの例 2

```
(*(+ 2 2)
   (/ (* (+ 3 5)
         (/ 30 10))
      2))
```

＊の後にスペースが無い

例題2．簡単な式

- 次の Scheme の式を DrScheme の実行用ウインドウに入力し、実行する

```
5 + 5
-5 + 5
3 * 4
8 / 12
3 + 4.5
(2 + 2) * (((3 + 5) * (30 / 10)) / 2)
```

簡単な式の実行例

```
Welcome to DrScheme, version 209.
Language: Intermediate Student.
> (+ 5 5)
10
> (+ -5 5)
0
> (* 3 4)
12
> (/ 8 12)
0.6
> (+ 3 4.5)
7.5
> (* (+ 2 2) (/ (* (+ 3 5) (/ 30 10)) 2))
48
```

6の上の線は 0.6666666… と6が続くことを意味する

例題3．円の面積

- 円の半径 r から面積を求める関数 area-of-disk を定義し、実行する

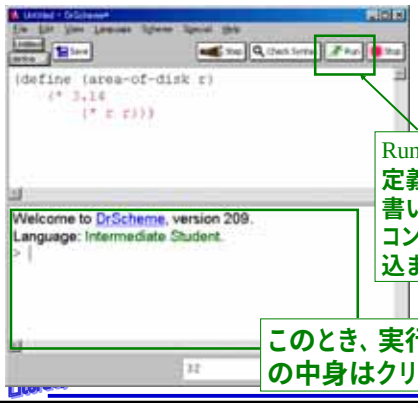
例） 5 → 78.5

- 関数の名前：area-of-disk
- パラメータ： r

「例題2．円の面積」の結果(1/4)

まず、定義用ウインドウで関数を定義している

「例題2．円の面積」の結果(2/4)



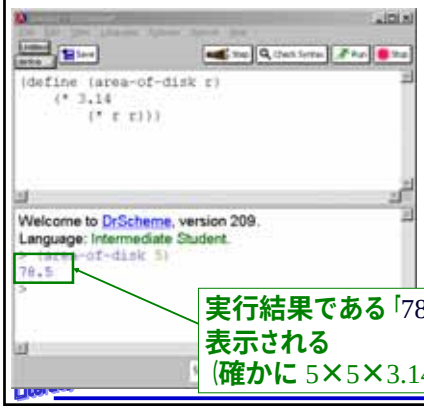
Run ボタンを押すと、定義用ウインドウに書いた関数定義が、コンピュータに読み込まれる

このとき、実行用ウインドウの中身はクリアされる

「例題2．円の面積」の結果(3/4)

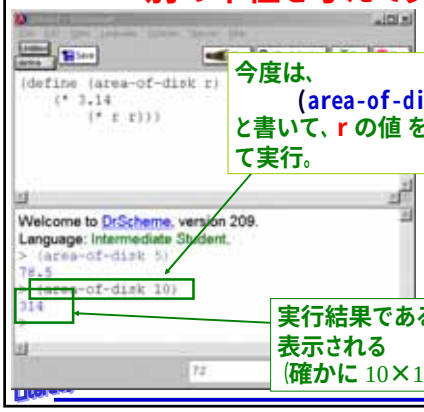
実行用ウインドウで、
(area-of-disk 5)
と入力して、Enter キーを押す
(これは、r の値 を 5 に設定しての実行)

「例題2．円の面積」の結果(4/4)



実行結果である「78.5」が表示される
(確かに 5×5×3.14 = 78.5)

別の半径を与えて実行



今度は、
(area-of-disk 10)
と書いて、r の値 を 10 に設定して実行。

実行結果である「314」が表示される
(確かに 10×10×3.14 = 314)

Scheme プログラミングの手順

Scheme の関数を定義し、インタプリタ(DrScheme)に読み込ませる

⇨

作った関数を実行して、実行結果を得る

Scheme の関数の定義　　Scheme の式の実行

Literacy

プログラム実行までの手順

① 関数を定義し、インタプリタ (DrScheme) に読み込ませる

例

```
(define (area-of-disk r)
    (* 3.14
       (* r r)))
```

Scheme の関数定義

円の面積を求める関数定義

② 読み込ませた関数を使った式を実行

例

```
(area-of-disk 5)
```

```
(area-of-disk 10)
```

関数 (定義した関数)を使ったScheme式の実行

実際に、半径 5、半径 10 の円の面積を求める

Literacy

プログラムでの関数定義の利用

■ 式の中に「関数名」を書く

⇒ このことを「関数適用」という

例: `(area-of-disk 5)` ←これも Scheme の式

実際に、半径 5 の円の面積を求める

Literacy

コンピュータが行っていること

Scheme の プログラム (関数)

例えば、関数定義

```
(define (area-of-disk r)
    (* 3.14
       (* r r)))
```

を読み込ませると

インタプリタ (DrScheme)

いったん、内部に記憶される

Literacy

コンピュータが行っていること

Scheme の式

例えば:

```
(area-of-disk 5)
```

を入力すると・・・

インタプリタ (DrScheme)

式の実行結果

78.5 が表示される

Literacy

実習

Literacy

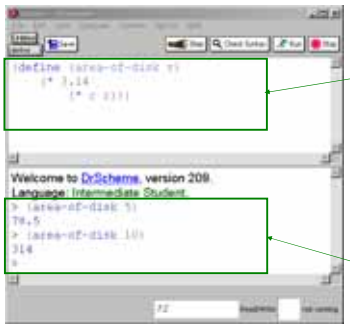
DrScheme の2つのウインドウ

定義用ウインドウ

実行用ウインドウ

Literacy

DrScheme の2つのウインドウ

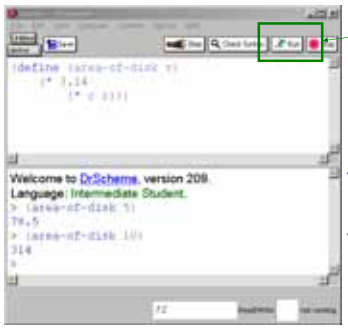


関数を定義し、コンピュータに読み込ませている

読み込んだ関数を実行させている

Literacy

DrScheme の Runボタン



Run ボタン

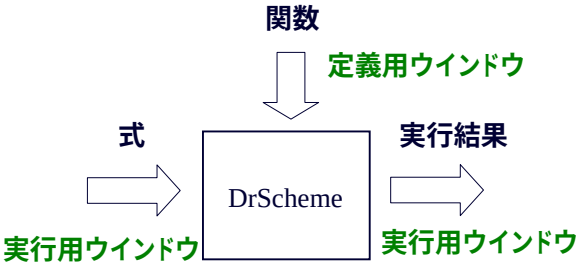
・定義を行ったら「Run ボタン」を押す

→ 関数定義の内容をコンピュータが読み込む

→「実行用ウインドウ」の中身はクリアされる

Literacy

DrScheme の2つのウインドウ



Literacy

実習の進め方

- 資料を見ながら、「実習」を行ってみる
- その後、各自「課題」に挑戦する
  - 各自で自習 ＋ 巡回指導
  - 遠慮なく質問してください
- 自分のペースで先に進んで構いません

Literacy

DrScheme の使用

- DrScheme の起動
  - プログラム → PLT Scheme → DrScheme
- 今日の実習では「Intermediate Student」に設定
  - Language
  - → Choose Language
  - → Intermediate Student
  - → OK ボタン

Literacy

「Intermediate Student」に設定

Language → Choose Language

Intermediate Student を選択し、「OK」をクリック

最後に Runボタン

Literacy

実習1．簡単なプログラム

- 次の関数を書き、実行する
  - f1: x と N から「x^N／N」を求める
  - f2: x と y から「x、y のうち大きいほう」を求める
  - f3: x から「x を100で割った余り」を求める
  - f4: x から「x を100で割った商」を求める

Literacy

「実習1．簡単なプログラム」の手順

1. 次を「定義用ウインドウ」で、実行しなさい
   - 入力した後に、Run ボタンを押す

```
(define (f1 x N)
    (/ (expt x N) N))
(define (f2 x y)
    (max x y))
(define (f3 x)
    (remainder x 100))
(define (f4 x)
    (quotient x 100))
```

2. その後、次を「実行用ウインドウ」で実行しなさい

```
(f1 2 5)
(f2 3 4)
(f3 123)
(f4 123)
```

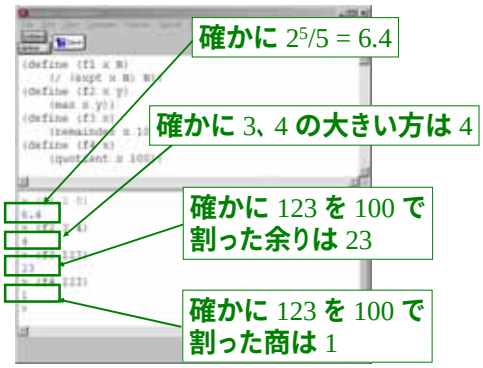
☆ 次は、課題に進んでください

Literacy

実習1の実行例(1/2)

Literacy

実習1の実行例(2/2)



Literacy

Scheme の式の例 (変数が登場するもの)

関数の本体には「変数を含む式」を書くことになる

- x^N／N

```
(/ (expt x N) N)
```
・・・ 2変数の式

- x、y のうち大きいほう

```
(max x y)
```
・・・ 2変数の式

- x を100で割った余り

```
(remainder x 100)
```
・・・ 1変数の式

- xを100で割った商

```
(quotient x 100)
```
・・・ 1変数の式

Literacy

課題

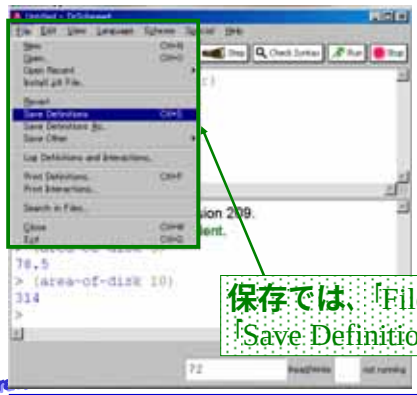
Literacy

DrScheme でのプログラム保存法

- 何日かかけてプログラム作成したいとき
  → プログラムを保存する必要あり
- DrScheme の「Save 機能」を活用すること
  – ファイル名は「英語」で付けることを勧める
- 九州大学のパソコンでは、「Zドライブ」に保存するのがよい
  – 「Dドライブ」に保存することもできるが、ログアウトする前に「Zドライブ」にコピーするのを忘れないこと。

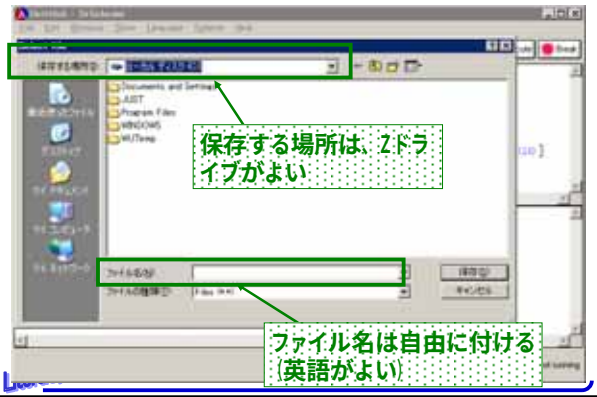
Literacy

プログラムのファイルへの保存(1/2)



Literacy

プログラムのファイルへの保存(2/2)



Literacy

課題1. ドルから円への変換

- ドル d から円を求める関数 d2y を定義し、実行例を示しなさい
  – 関数定義には define を使う
  – 1ドルは106.53円とする

Literacy

課題のヒント(解答例)

- x から「10 × x + 30」を求める関数 fooを定義し、実行例を示しなさい

解答の例:

```
(define (foo x)
   (+ (* 10 x) 30))
```

実行例は次の通り

```
(foo 10) は 130
(foo 20) は 230
```

実際に、DrScheme で実行して確認した

(あくまでも例です。各自、工夫を試みなさい。
独自の工夫を高く評価します)

Literacy

課題のヒント

- ここにあるのは「間違い」の例です。同じ間違いをしないこと

1. 「かっこ」の間違い

```
define (d2y d)
   (* 106.53 dollar)
```

⇒ 全体をかっこで囲むこと

2. 変数名の対応の間違い

```
(define (d2y dollar)
   (* 106.53 d))
```

⇒ 変数名dとdollarはどちらかに統一すること(どちらでも可)

3. 関数の書き方の間違い

```
(define (d2y)
   (* 106.53 d))
```

⇒ d2y の後に d が必要

4. 関数名の付け方の間違い

```
(define (d 2 y d)
   (* 106.53 d))
```

⇒ 「d 2 y」でなく「d2y」と書くこと

Literacy

課題2. 摂氏から華氏への変換

- 摂氏 (Celsius) 表現の温度 c から華氏 (Fahrenheit) 表現の温度 f を求める関数 c2f を定義し、実行例を示しなさい
  - 関数定義には define を使う
  - 摂氏と華氏の変換式： c＝5 ×(f - 32) / 9

Literacy

課題3. 元利の計算

- 元金と年利と年数を受け取り、元利を求める関数 interest を定義し、実行例を示しなさい
  - 関数定義には define を使う
  - 元利の計算式：
    「元利 = 元金 × (1＋年利)^{年数}」
  - 作成した関数を実行し、元金1000円、年利2%とし、50年後の元利を報告しなさい
  - x の y 乗の計算には expt を使う
    (expt x y)

Literacy

今日の課題 (リテラシ編)

- 電子メールのガイドライン
  http://www.cgh.ed.jp/netiquette/rfc1855j.html
  の中の電子メールのガイドラインを読んで、注意すべきことをまとめましょう。
- また、わからないことがあれば、その説明をWebから探してみましょう。
  - 検索エンジン (Google、MSNサーチ、Yahoo、gooなど)

Literacy