プログラミングAIII

2025年度講義資料 (2)

新潟大学　工学部工学科　知能情報システムプログラム

青戸等人

目次

目次

インタープリタ(対話型コンパイラ)との対話 (1)

```
$ smlsharp
...
# 12;
val it = 12 : int
# 1 + 1;
val it = 2 : int
# 1 + 2 * 3;
val it = 7 : int
# (1 + 2) * 3;
val it = 9 : int
# 2 - 3;
val it = ~1 : int
# ~2 * ~3;
val it = 6 : int
```

- インタープリタのループ
  式の入力
  ⇒値の表示
  ⇒プロンプトの表示
- 式の区切りは";"(セミコロン)
- 式の値と一緒に「型」も表示される
  - int ... 整数型
- 整数型の演算 (教p.12.7)
  - 加算，乗算，減算
  - マイナスは，~
  - 通常の結合順序

インタープリタ(対話型コンパイラ)との対話 (2)

```
# 11; 12;
val it = 11 : int
val it = 12 : int
# 11
> + 12
> ;
val it = 23 : int
# 2 + 3;
val it = 5 : int
# it;
val it = 5 : int
# it * 4;
val it = 20 : int
```

- ";"と改行の扱い
  - ";"は行の途中にも使用可.
  - 改行は式の終了とは見做されない.
  - ">" ... 式の続きを入力するプロンプト (教p12.10)
- itについて (教p.11.1)
  - "it"には，最後の入力の値が保持されている.
  - itを式に使ってよい.
- インタープリタの終了(教1.6節) ... ^D (Ctrlキーを押しながらDキー，ファイル終了文字)

インタープリタとの対話の基本形

インタープリタとの対話の基本形

```
1 # 式 ;                              (*ユーザの入力*)
2 val it = 式の値 : 式の型  (*インタープリタからの返答*)
3 #
```

- インタープリタは「式」から「値」を計算する. 「式」から「値」を計算することを「式を評価する」という.
  「式」=「プログラム」
  「値」=「計算結果」
- 式には型がついている.「式の値」とともに「式の型」にも注目すべし. 型は自動的に推論される.

基本的な型の例 (教p.11〜p.14)

整数型 int 以外の基本的な型いくつか. 詳細は第3回の講義で.

```
# 1.0; 1e2;
val it = 1.0 : real
val it = 100.0 : real
# true; false;
val it = true : bool
val it = false : bool
# #"A"; #"#"; #"\n";
val it = #"A" : char
val it = #"#" : char
val it = #"\n" : char
# "Programming AIII"; "";
val it = "Programming AIII" : string
val it = "" : string
```

- 実数型 real
  - 小数表示や指数表示で，整数型と区別.
- 真理値型 bool
  - true (真) と false (偽) (小文字)
- 文字型 char
  - ダブルクオート(")で囲んだ文字の前に#を付ける.
- 文字列型 string
  - 文字列は"で囲んで表す.

整数型 int と実数型 real

```
# 10 div 3; 10 mod 3; ~ 3;
val it = 3 : int
val it = 1 : int
val it = ~3 : int
# 3.0 * 1e~3;
val it = 0.003 : real
# 10.0 / 3.0;
val it = 3.33333333333 : real
# (real 10) / 3.0;
val it = 3.33333333333 : real
# (1.0 - 0.6) * (1.1 + 0.9) / ~4.0;
val it = ~0.2 : real
```

- 整数型の演算
  - 加乗減算
  - マイナス(~)
  - div ...商
  - mod ...剰余
- 実数型の演算
  - 四則演算(+,-,*,/)
  - マイナス(~)
  - real関数 (教p.16.1) ...整数を対応する実数に変換

真理値型 bool

```
# 10 = 2 * 5;
val it = true : bool
# 2 + 2 > 3;
val it = true : bool
# (1 < 2) = (2 > 3);
val it = false : bool
# if 2 < 3 then "y" else "n";
val it = "y" : string
# (if 2 < 3 then 1 else 0) * 3;
val it = 3 : int
# (if 2 < 3 then 1 else 0) = 1;
val it = true : bool
```

- 真理値型を返す演算
  - =... 値の同一性
  - <,>... 値の大小
- if c then exp_1 else exp_2
  - cが真なら，exp_1の値
  - cが偽なら，exp_2の値
  - if 文全体で1つの式になる．
  - 式なので，もちろん，他の式の中でも使ってよい．

文字型 char と文字列型 string

```
# ord #"A"; ord #"a";
val it = 65 : int
val it = 97 : int
# chr 66; chr 98;
val it = #"B" : char
val it = #"b" : char
# #"A" < #"B";
val it = true : bool
# (str #"A")  = "A";
val it = true : bool
# "aBc" < "abC";
val it = true : bool
# "Hello, " ^ "World!";
val it = "Hello, World!" : string
```

- ord,chr関数... 文字と文字コード(整数)の変換
- 文字の大小... 文字コードで比較
- str関数... 文字に対応する文字列を返す
- 文字列の大小... 辞書式順序で比較
- ^ 演算... 2つの文字列を連結した文字列を返す．

関数適用における括弧について (教p.13.-3)

- ord,chr,real,strなど，さまざまな関数を見てきた．
- 数学の通常の記法では，関数fに対して，(実)引数eを適用した形(関数適用)を，「$f(e)$」のように，引数を括弧を囲んで書く．
- 一方，SMLの記法では，「$f\ e$」のように，括弧を使わず，スペースで区切って書く．
- 前者のように書いてもよいが，後に学習する「高階関数」を含めて扱う場合には，後者の書き方が合理的．(実は，数学でも，場合によっては，後者の記法を使うことがよくある．)
- もっぱら，括弧は結合の順序を表わすのに用いる．

ファイルに作成したプログラムの利用 (教1.5節)

プログラムファイルの利用

```
1 # use "ファイル名" ;
```

- use 文は，指定されたファイルに書かれている式を実行する．後述する「変数定義」や「関数定義」が書かれていれば，それを実行して変数名や関数名を束縛する．
- 式の評価に失敗したときは，そのエラーを解析してエラーメッセージを表示する．
- ファイル名は，インタプリタを起動したディレクトリをカレントディレクトリとしたパス指定を文字列として与える．

プログラムファイルの利用とコメント

```
$ cat test.sml
if #"a" < #"A"
then (str #"a") ^ (str #"A")
else (str #"A") ^ (stu #"a");    (* スペルミス *)
$ cat test2.sml
if #"a" < #"A"
then (str #"a") ^ (str #"A")
else (str #"A") ^ (str #"a");
$ smlsharp
...
# use "test.sml";
test.sml:3.19-3.21 Error: (name evaluation "190") unbound variable: stu
# use "test2.sml";
val it = "Aa" : string
#
```

- (* と *) に囲まれた部分はコメントとしてインタープリタに無視される(教p.19)．

構文と型のエラー (教1.3節) (1)

エラー検出，および，これまで学習した概念で説明できる「構文エラー」と「型エラー」について説明する．

- SMLでは，コンパイルの時点で型の整合性チェックが行なわれる(静的型チェック)．「静的型チェック」はモダンな(関数型)プログラミング言語の大きな特徴．
- エラーレポートの最初の$n.x$–$m.y$の部分は，エラーの位置情報(n行目のx文字目〜m行目のy文字目)．
  インタープリタ場合は，インタープリタ起動後の位置．ファイルから読み込んだ場合は，ファイルの位置．
- エラー検出は完全でない．
  - 2番目以降に検出されたエラーは，最初のエラーが伝搬したものかもしれない．
  - 検出された位置より，手前の箇所にバグがあるかもしれない．エラーメッセージも注意深く見る．

構文と型のエラー (教1.3節) (2)

- 構文エラー：構文的にあり得ない箇所が検出された．

```
# 1..0;
(interactive):1.1-1.1 Error: syntax error
found at PERIOD
```

- 型エラー：演算子(operator)の型と引数(operand)の型の整合性がとれていない．

```
# chr #"a";
(interactive):2.0-2.7 Error:
  (type inference 026) operator and operand
don't agree
  operator domain: int
          operand: char
```

  - 演算子(chr)の定義域(domain)は，整数(int)．
  - 実際に与えられている引数(operand)は，文字(chr)．

目次

変数の束縛と利用

```
# val x = 2;                    (* 変数の束縛 *)
val x = 2 : int
# x + 3;
val it = 5 : int                (* xの値は2になっている *)
# val y = 1 + x;                (* 変数の束縛 *)
val y = 3 : int                 (* 変数の値は式の評価結果 *)
# val x = "aaa";                (* 変数の新しい束縛 *)
val x = "aaa" : string
# val y = "bbb";
val y = "bbb" : string
# x ^ y;                        (* 新しい束縛が使われる *)
val it = "aaabbb" : string
# z;                            (* 変数が未定義の場合 *)
(interactive):3.0-3.0 Error: (name evaluation "190")
unbound variable: z             (* 未定義エラー *)
#
```


変数の束縛の基本形

変数の束縛の基本形

```
1 # val 変数名 = 式 ;                (*ユーザの入力*)
2 val 変数名 = 式の値 : 式の型 (*インタープリタの返答*)
3 #
```

式中の変数の評価

式の中に表われる変数は，その値に評価される．

- 変数の値を定義することを，変数を束縛するという．
- 変数の値は，式を評価した値に束縛される．
- 変数の型は式の型と同じになる．
- 変数の値が束縛されていなければ，未定義エラーになる．

変数名の約束 (教p.16–17)

```
# val A31 = 11;
val A31 = 11 : int
# val Z' = 3;
val Z' = 3 : int
# val A_2 = 11;
val A_2 = 11 : int
# val abc_52' = 4;
val abc_52' = 4 : int
# val -- = 2;
val -- = 2 : int
# val ++ = 3;
val ++ = 3 : int
# -- + ++ ;
val it = 5 : int
#
```

- 変数を表わす識別子(変数に使える名前)は2種類のタイプがある．（ここでは，値を表わす変数のみを考える．）
- (1種類目) 先頭はアルファベットで，かつ，以降は，アルファベット，数字，_，'のどれか．
- (2種類目) 特殊記号の列(特殊記号は，例えば，+,-,<,など)
- ただし，いくつかの予約語は例外．

変数束縛のスコープ

```
#val x = 1; val y = 2; val x = x + y;
val x = 1 : int
val y = 2 : int
val x = 3 : int
#
```

変数束縛のスコープ(有効範囲)

- 変数の有効範囲は，宣言の後ろから，インタープリタが終了するか同じ名前の変数が宣言されるまで．
- 同じ名前の変数が宣言されると，元の変数は隠れて見えなくなる．　(⇒ 述語論理式における変数束縛と同じ)
- 変数宣言の右辺(bodyとよぶ)の評価では，元の変数が使われる．(右辺を評価して，その後で束縛されるため．)