プログラミング言語論

第5回
手続き呼出し

担当：犬塚

今日の講義

手続きや関数などの呼出しに関わること

- 手続きの宣言、手続きの定義
- 名前の束縛とスコープ
- ブロック構造
- スコープ規則
- 引数渡し
- 引数渡しと評価（計算）方式

手続きと関数
procedures and functions

- 手続きと関数はともにサブプログラム。
- 引数を渡して、何らか引数に依存した計算を行う
  ＝パラメータ化された計算（parameterized computation）
- 結果の返し方：
  1. 呼出し側とサブプログラムの両方から可視(visible)な変数に結果を反映させる。
  2. 実引数と結びついた仮引数の一つに値を与える。
  3. 関数として結果を返す。
- 関数は返す値以外に計算状態に全く影響を与えないのが理想的＝副作用（side-effect）無し

手続き／関数の宣言と定義
declaration and definition of procedures and functions

手続きの定義と宣言を分けて考える。

手続きの宣言

Procedure 手続き名（パラメータ）;

いろいろな呼び方がある：
インターフェース
プロトコル
パラメータプロファイル
プロトタイプ

手続きの定義

Procedure 手続き名（パラメータ）;

手続き本体

型宣言

```
type intarray =
  array[1..10]
  of integer
```

変数定義

```
var a : intrarray
```

名前，束縛　names, bindings

- プログラム中で各種の対象を表わすために名前を用いる
  - 変数名，手続き名，型名など
- 名前を何かそれが表わす対象に結びつけることを束縛という。
- 束縛時刻 binding time を念頭に置くことは重要
  - 言語の設計時
  - コンパイラの実装時
  - コンパイル時
  - プログラムのロード時
  - リンク時
  - 実行時

例：　var count:integer;
　　count := count +5;

integerの意味	＝言語設計時
5の表現	＝コンパイラ実装時
countの型	＝コンパイル時
変数としてcount	＝リンク時
countの値	＝実行時
+	＝コンパイル時 (両辺の型がきまってから)

特別な名前　special names

- プログラミング言語で、特別な名前によってプログラムの構造などを示す。

  if，procedure，integer，…

- 言語によってその扱いが異なる
  - キーワード　keywords　文脈によって特別な意味をもつ名前　他の文脈では、別の意味で使ってもよい
  - 予約語　reserved names　特別な意味を表わすために予約された名前　他の目的には使えない。

    FORTRAN　Integer Real　整数型のRealという変数
    　　Real　Integer　実数型のIntegerという変数

  - 事前定義語　predefined words　予め意味が定められているが、再定義して別の意味で使うこともできる。

有効範囲（スコープ）scope

- 名前（変数名、手続き名等）とその束縛に対し、その名前でその束縛が参照されるとき、その名前は可視(visible)という。
- ある名前が可視であるプログラムの範囲をスコープという。

  スコープ規則：

  1. ある手続きを実行中に変数xが表れたとき、xの宣言がその手続きにあるならば、この宣言がxの束縛を決める。
  2. その手続きにないなら、この手続きを定義している手続き（親手続き）の中に宣言を探す。そこにあれば、その宣言が束縛を決める。
  3. さらに、そこになければ…その親…

有効範囲（スコープ）scope

- 通常のスコープの規則

  ＝静的スコープ規則　static scope rule
- 静的スコープ規則は、プログラムによって予め決められる規則である。だから静的。
- プログラムの字面で決まるので、lexical scope rule ともいう。
- 逆に、プログラムのある箇所での束縛の全体を環境 environment という。

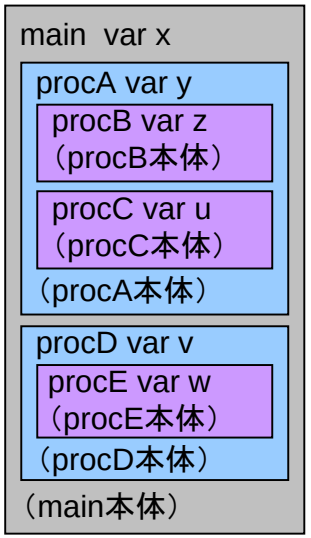
動的スコープ　dynamic scope

- 名前の束縛が実行時にきまる規則。
  1. ある名前xが手続きに現れたとき、xの宣言がその手続きにあれば、その宣言がxの束縛を決める。
  2. なければ、この手続きを呼び出した手続き内の、宣言をみる。そこにxの宣言があればそれがxの束縛を決める。
  3. さらになければ、それを呼び出した手続きを、…
- この方式を動的スコープ規則（dynamic scope rule）という。
- ある変数xの束縛、型が実行時にしか決められない。
- LISP他で使われてきたが、可読性、実行効率、安全性の問題で、最近のLISPや他の言語では使われていない。

ブロック構造 block structure

- 手続きの中に手続きを入れ子状に書いた場合、スコープが形成される。
- 手続きを作らずに、プログラムの一部をくくりだして、そこにスコープをつくる仕掛けをブロックという。
- ブロックは、
  - その中で変数など、名前(局所変数など)の宣言ができる。つまり、束縛できる。
  - 通常その変数はstack-dynamic。
  - その中がスコープになる。
- ブロックをつくれる言語：
  ブロック構造言語　block-structured languages

手続きのスコープ関係



各手続き本体で可視の名前

- main:　procA, procD, 変数x
- procA:　procA, procB, procC, 変数x, y
- procB:　procA, procB, 変数x, y, z
- procC:　procA, procB, procC, 変数x, y, u
- procD:　procA, procD, procE, 変数x, v
- procE:　procA, procD, procE, 変数x, v, w

その手続きの本体で、名前が可視であれば手続きは呼べる。自分よりも後ろのものは通常可視でない。

プログラム中での場所で、可視の変数を
参照環境 reference environment
という。

引数　parameters

- 仮引数 formal parameters と
- 実引数 actual parameters

- 仮引数と実引数の対応付け：
  - その位置によって対応付ける：positional parameters
  - 仮引数の名称を指定：keyword parameters
    sort(list => L, number => n);
- 引数に関するその他のしくみ
  - デフォルト値を持つ引数
  - 仮引数と実引数の個数が異なる
    実引数が足らない → 未定義を許す
    実引数があまる → 複数の実引数をリストにしてまとめて1つの仮引数に対応づける
    その個数を与える、など。

引数渡し parameter passing

- 実引数を仮引数に渡す方式(呼出しの方式)には複数ある。
  - 値渡しまたは値呼出し
    pass-by-value, call by value
  - 結果渡し
    pass-by-result
  - 入出力渡し
    pass-by-value-result
  - 参照渡しまたは参照呼出し
    pass-by-reference, call by reference
  - 名前渡しまたは名前呼出し
    pas-by-name, call by name
- 引数には、値を手続きに渡す目的(in mode)と受取る目的(out mode)のものがある。

値渡し、値呼出し

- in mode のためのモデル。
- 実引数の値(右辺値)をコピーして仮引数に渡す方式
  1. 実引数の値を求める。
  2. 仮引数は、手続き内の局所変数として生成する。
  3. 実引数の値をその変数の初期値とする。
  4. 手続きの本体を実行する。
- 手続き内で局所変数の内容が変化しても、呼出し側に影響しない。
- 局所変数の生成と値のコピーの必要がある。

結果渡し

- out-mode のモデル。計算結果を戻すために使う。
- 値渡しを逆方向で行う方式。
  1. 仮引数は単に局所変数として生成され、計算で利用される。
  2. 手続きでの計算が行われ結果がでる。
  3. 仮引数に入っている値を実引数にコピーする。
- 実引数は変数でなければならない。
- 変数生成、値のコピーが必要。

入出力渡し

- in/out-mode のモデル。
- 値渡しと結果渡しの組合せ。
- 値渡しを逆方向で行う方式。
  1. 仮引数を局所変数として生成する。
  2. in modeの仮引数には、実引数の値がコピーされる。
  3. 手続きでの計算が行われる。
  4. out-modeの仮引数に入っている値を実引数にコピーする。

参照渡し、参照呼出し

- 変数渡し、変数呼出し pass-by-variable, call by variable ともいう。
- in mode, out mode 両方のモデル。
- 値ではなく、実引数への参照（＝アクセス経路、通常は番地、左辺値）が仮引数に渡される。
- 仮引数への変更は、実引数に反映するので、結果をかえすため（out mode）として使える。
- 局所変数の生成、コピーの必要がない。
- 実引数は変数等であり、式は許されない。
- 変数の衝突に注意が必要。

名前呼出し

- 実引数に与えたテキスト、仮引数の箇所に置換する。
  ただし、名前の衝突が起きる場合はこれを回避する。
  1. 手続きの本体にある仮引数を、対応する実引数のテキストで置換する。
  2. ただし、これによって手続き内の局所変数と衝突する場合は、局所変数の名前の付け替えをする。
  3. できあがった手続き本体を、手続き呼出しのところに挿入する。
  4. ただし、この挿入によって手続き内の大域変数が呼出し側の局所変数と衝突する場合は呼出し側の局所変数を付け替える。
  5. そのまま実行する。

名前の衝突は2通り
- 実引数と手続き内の局所変数との衝突
- 手続き内の大域変数と呼出し側の局所変数の衝突

名前呼出しの例

呼出しプログラム
```
begin
  integer i, n;
  n:=n+1;
  p(A[i])
    ...
end;

proc p(x)
begin
  integer i;
  x:=x+i+n
end;
```

①手続きp本体の仮引数xを実引数A[i]で置換
```
begin
  integer i;
  A[i]:=A[i]+i+n
end;
```

②ｉが衝突するので局所変数をｊに付け替え。
```
begin
  integer j;
  A[i]:=A[i]+j+n
end;
```

①呼出し箇所に手続きを挿入
```
begin
  integer i, n;
  n:=n+1;
  begin
    integer j;
    A[i]:=A[i]+j+n
  end;
    ...
end;
```

①手続き内の大域変数nが呼出し側の局所変数と衝突するので、名前の付け替え。
```
begin
  integer i, m;
  m:=m+1;
  begin
    integer j;
    A[i]:=A[i]+j+n
  end;
    ...
end;
```


引数渡しと評価方式

- 引数渡しの方式は式の評価方式と関係する。
- 例
  ```
  p(X,Y,Z)
  begin if X>0 then Y else Z end;
  ```
  において、呼出しp(f(x), g(x), h(x))を考える。

- 参照渡しでは → こうした呼出しはできない。
- 値呼渡しでは → f(x), g(x), h(x)を計算し、その値を渡す。
- 名前渡しでは → f(x), g(x), h(x)をそのままテキストとして渡し、置換されたプログラムを実行する。
  f(x)の値によって、g(x)とh(x)の一方のみ計算される。

最内評価、最外評価、遅延計算

- 値呼出しは、式を計算するのにその最も内側から評価する方式に対応し、最内評価 inner-most evaluationという。
- 名前呼出しは最も外側から評価する最外評価 outer-most evaluation に関連する。
- 最外評価は、式の計算をなるべく後回しにする遅延計算（評価） lazy computation (evaluation) と関連する。
- 遅延計算は、並行プログラミングなどでの基礎技術。
- 式の一部に未定義の関数がある場合、遅延計算は最内評価よりも広い範囲で計算に成功する。

まとめ

- 手続きと手続き呼出しに関する基本的メカニズムを紹介した。
  - 手続き内での名前と対象の結びつき＝束縛、環境
  - 束縛を決めるタイミング
  - 束縛の範囲＝スコープ
  - スコープを決める規則＝スコープ規則
    - 静的スコープ、動的スコープ
  - 手続きに情報を渡す仕掛け＝引数
  - 引数渡しの方式
    - 値渡し、参照渡し、名前渡し
  - 引数渡しと評価方式、最内評価、遅延計算