Algol60 補足

プログラム言語論

Algol 60 interpreter

- ここに一つあります
  - http://www.bertnase.de/a60/

inout.a60

```
'Begin'
        'Real' x, y;
        'Integer' i, j;

        outstring (1, "Give me two real numbers:¥n");
        inreal (0, x);
        inreal (0, y);
        outstring (1, "Got: ");
        outreal (1, x);
        outstring (1, " and ");
        outreal (1, y);
        outstring (1, ".¥n");

        outstring (1, "Give me two integer numbers:¥n");
        inreal (0, i);
        inreal (0, j);
        outstring (1, "Got: ");
        outinteger (1, i);
        outstring (1, " and ");
        outinteger (1, j);
        outstring (1, ".¥n");

        outstring (1, "done.¥n")
'End'
```

a60 では ` はなくてよい
キーワード先頭は小文字も可
出力に vprint も利用可能
ex. vprint(x)

```
begin
  real procedure sum;
  begin real S, x; S:=0; x:=0;
      for  x:= x+0.01 while  x<=1  do
         S := S + f(x);
       sum := S/100
  end;
    ...
  ...
  begin
    real procedure f(x);
      value x; real x;
       f := x^2 + 1;
    sumf := sum
  end
  ...
end
```

rand.a60

```
'begin'
'comment'       create some random numbers, print them and
                print the average.
        ;

        'integer' NN;       NN := 20;

        'begin'
                'integer' i;
                'real' sum;

                vprint ("random numbers:");

                sum := 0;
                'for' i := 1 'step' 1 'until' NN 'do' 'begin'
                        'real' x;
                        x := rand;
                        sum := sum + x;
                        vprint (i, x)
                'end';

                vprint ("average is:", sum / NN)
        'end'
'end'
```

Jensen's device

- $x = \Sigma_{i=1,n}\ V_i$ x := Sum(i,1,n,V[i])

```
real procedure Sum(k,l,u,ak);
  value l,u;  integer k,l,u; real ak;
  begin real S; S:=0;
      for k:=l step 1 until u do
            S:= S+ak;
       Sum:=S
  end;
```

- $x = \Sigma_{i=1,n}\ 1/i$; n=100の時、5.1873ぐらい
- $x = \Sigma_{i=1,n}\ \Sigma_{j=1,n}\ 1/(j+n*(i-1))$

  Sum(i,1,n,Sum(j,1,n, 1.0/(j+n*(i-1))))
- 行列の積

注意事項

- 宣言文は、実行文の前にあるべし。
  - 例えば、begin integer i; i:=1; real x; x:=1 end; はコンパイルエラー
- call-by-name では実引数に不適当なもの（変数が期待されているときに、定数）を渡すとエラーになる
  - 例えば、Sum(1,1,4,x)

マクロ

（コンピュータ科学では）

1. 規則またはパターンであって、
2. ある入力列（しばしば、文字列）を出力列（当該入力列を置き換える）に、
3. ある定義済み変換方法に従い、写像する方法を記したもの

https://en.wikipedia.org/wiki/Macro_(computer_science)

C のマクロ

- プリプロセッサがコンパイル前に実行する
- 文字列の置き換え

```
#define STRING1 STRING2
```

- パラメータをとる関数形式マクロ (function-like macro)

```
#define SQUARE(val) ((val)*(val))
```

- マクロ特有の演算子を用いることもできる

```
#define TO_STRING(symbol) #symbol
#define PASTE_TOKEN(a,b)  a##b
```

https://ja.wikipedia.org/wiki/マクロ_(コンピュータ用語)

Lispのマクロ

- (補足) Lisp ではデータとプログラムは同じ形の対象(S式と呼ぶ)である

  `(x y)　　(add x y)
- S式からS式への変換であり、変換後コンパイル・実行される

```
(defmacro unless (cond then)
  `(if (not ,cond)
       ,then))

(if (not foo-p)        (unless foo-p
    bar)                   bar)
```

https://keens.github.io/blog/2015/07/04/makuronitsuiteseirishitemiru/