B6 KSIM －統計値の自動収集を考えたシミュレーション言語－

山本 喜一 （慶応義塾大学情報科学研究所）

はじめに

KSIMは統計値の自動収集を考え，特に使い易さに重点を置いてSIMSCRIPTを改良した言語である．

SIMSCRIPTは離散型システムのためのシミュレーション言語で，その最大の特色は表現力が非常に豊かで，ほとんどすべての離散型システムのモデルを記述できるといっても過言ではない．ところが言語の汎用性と使い易さという二面は，特にシミュレーション言語においては相反するように思われる．実際SIMSCRIPTを完全に使いこなすにはFORTRAN，SIMSCRIPTの知識とともにimplementされている計算機についての知識も必要である．

KSIMは特に使い易さを重点にして設計した，FORTRAN-likeな言語で，基本思想はSIMSCRIPTと同じである．

Ⅰ KSIMの構成

KSIMシステムはKSIM ソース プログラムからFORTRANⅣのソース プログラムおよびシミュレーション パッケージを生成するトランスレータである．

基本的にはSIMSCRIPTにもとずいて構成しているが，SIMSCRIPTにはdefinition formの複雑さ，変数のinitializeの問題という使い易さという面での 大きな障害がある．

機能としてはSIMSCRIPT 1.5，SIMSCRIPT Ⅱから取り入れた部分もあるが，使い易さを主眼にしてdefinition form，intialization form の簡略化をはかり，待ち行列および施設に関する標準的な統計値の自動収集と出力のためのステートメントの追加，setオペレーションに関する機能の追加，算術ステートメントの追加，random look-up法の拡張などを行なっている．

Ⅱ プログラムの構成

プログラムはできるだけFORTRAN-likeなステートメントで構成できるように考え，様式を細かく指定するような部分をなくすことが基本的な考えである．

プログラムの構成は図1に示すとうりで，DEFINITIONはentity，attribute，event notice，set，queue，equipment などを定義する部分，MAIN ルーチン，EVENTルーチン，サブルーチン，FUNCTIONはシミュレーション・モデルの状態変化

を記述する部分，INITIAL CONDITIONはDEFINITIONで定義した変数に初期値を与えるデータ部分，DATAはその他のデータである．

KSIMではイベントに関しては外生的イベントと内生的イベントの区別をなくし，シミュレーションはかならずMAINルーチンの先頭から実行するようになっている．一般にMAINルーチンには少なくとも1つのイベントを予定表に登録するCREATE，CAUSEステートメントが含まれる．

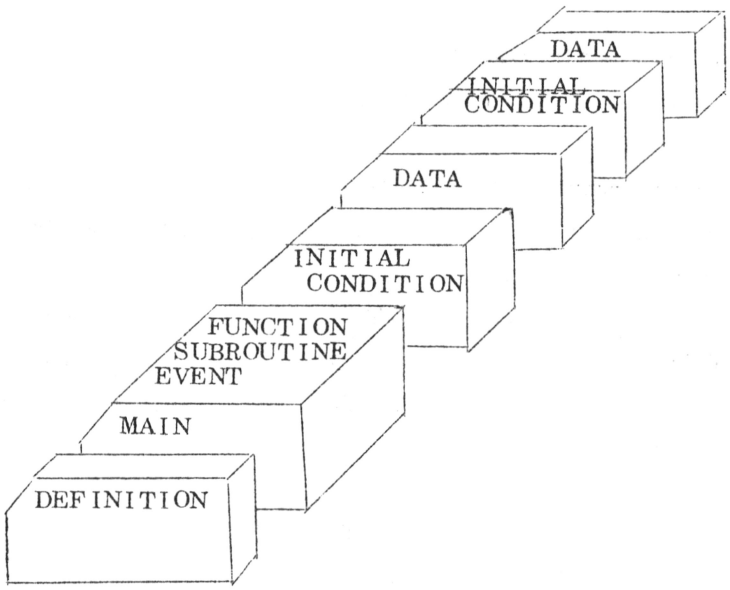


図1.　プログラムの構成

外生的イベント，内生的イベントの区別をなくした結果として，それぞれのイベントに対応してかならず同名のevent noticeをDEFINITIONで定義しなければならない．KSIMシステムはDEFINITIONを参照しながらタイミング ルーチンを生成する．

INITIAL CONDITIONでは配列名によるinitializeをやめ，変数名とデータを対にして与えるようになっている．またデータは型だけを指定しformatは指定しない，いわゆるfree format readを行なう．さらに初期値が必要な変数に対するデータだけが与えられれば，その他の変数の値は自動的に0にセットされる．

シミュレーションの実行をいくつかの初期値の組に対して連続して行なう場合には，2回目以降の実行に対する初期値は1回目と初期値が変った変数名とデータを与えれば，その他の変数については最初と同じ初期値に自動的にセットする．

Ⅲ　データ構造

KSIMではデータ構造をシステムが適当に決定するという点でSIMSCRIPT と大きく異っている．SIMSCRIPTにおけるdefinition formの難しさは，userが自分の用いるデータの構造まで定義しなければならず，equivalence を達成するための注意が必要であることに大きな原因がある．KSIMではDEFINITIONで変数名と型（IntegerまたはReal）およびPackingの有無だけを指定すれば，あとはシステムが適当なデータ構造を生成するようになっている．

Ⅲ．1　Temporary entityの構造

Temporary entity のためのデータ構造は4語ブロックを基本レコードとしたリスト構造である．

たとえばTemporary entity CAR に attributeとしてATT1，ATT2，……，ATT15 がすべてpackingされないで定義されていたとすると，CARがCREATE されるたびに図2に示すデータ構造が作られる．

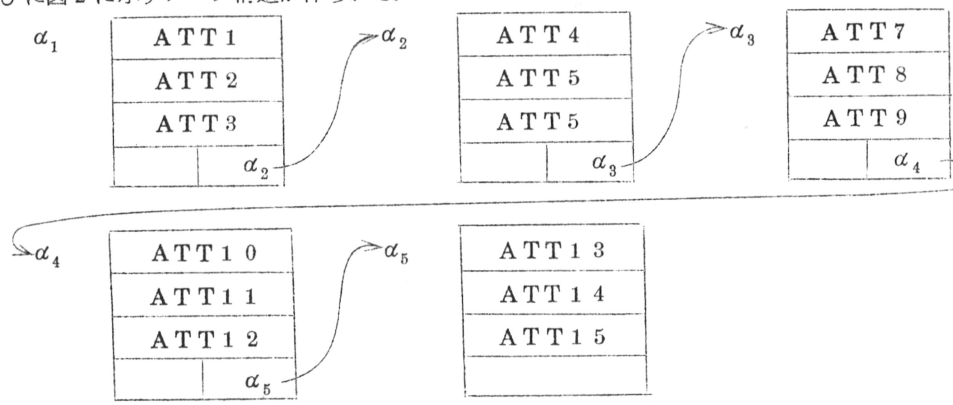


図2.　Temporary entity CARのデータ構造
（attribute の packing を行なわないとき）

また偶数番目のattributeすなわちATT2，ATT4，……，ATT14 だけにpackingが指定されている場合には図3に示すデータ構造が作られる．

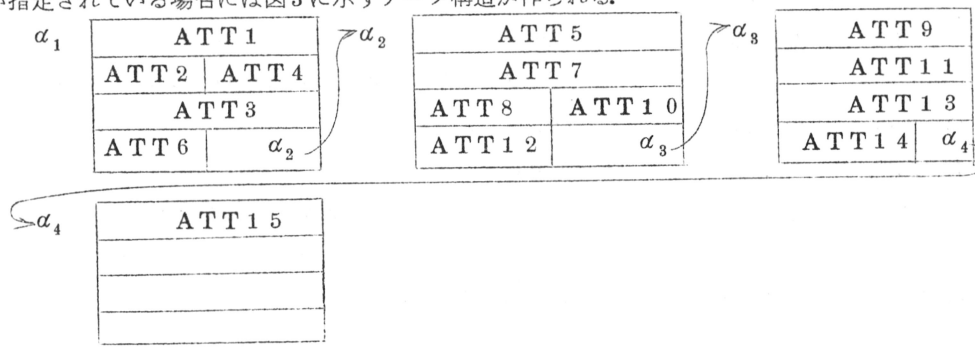


図3.　Temporary entity CARのデータ構造
（attributeのpackingを行なうとき）

図2から明らかなように各4語ブロックの4語目の前半分は無駄になる．けれども図3に示したようにattributeのpackingを用いればこの無駄をある程度防ぐことができる．

Temporary entityのattributeのpackingは，KSIMでは1／2語packingだけを許している．

このリスト構造はいくらでも長く続けることができるのでtemporary entityのattributeの個数は無制限である．

Ⅲ．2 Event noticeのデータ構造

Event noticeがDEFINITIONで定義されていると，CREATEのたびに4語から成るレコードが自動的に生成されるが，このレコードの最初から3語はシステムで用いるのでuserは関知しなくても良い．もしevent noticeにattributeを持たせたときにはtemporary entityのときと同様に必要に応じて4語ブロックからなるリスト構造が作られる．

一般的なevent noticeのデータ構造を図4に示す．図4でpriorityはイベントを発生するときの優先順位，ID番号は対応するイベントの番号，PCLOCK，SCLOCKはそれぞれインベントの予定表（CLOCK）のpredecessor，successorのポインタ，TIMEはそのイベントの発生時刻である．

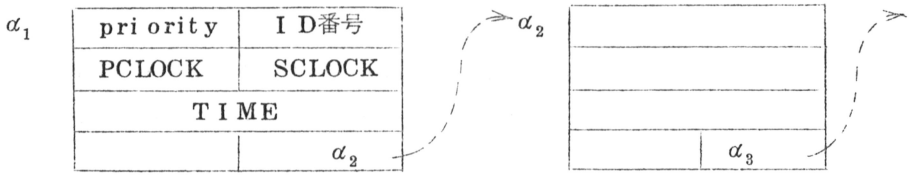


図4 一般的なevent noticeのデータ構造

Ⅲ．3 Permanent entityのデータ構造

Permanent entity，permanent attribute，system attributeについてはSIMSCRIPTとほとんど同じである．

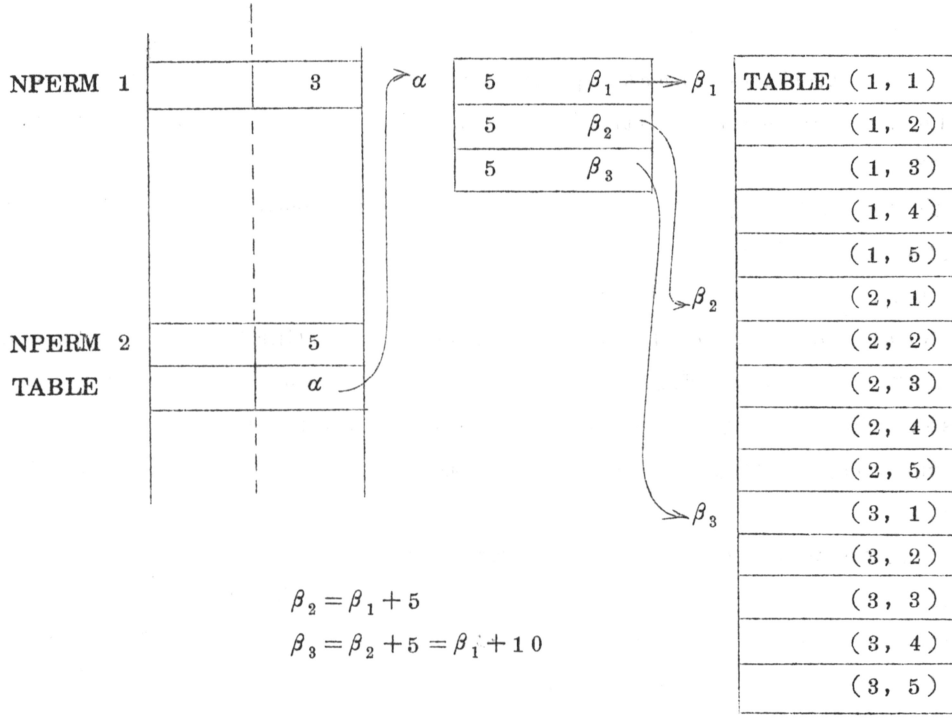
ただしpermanent attributeにはpackingを許し，添字を2つもつpermanent attribute（2次元配列）のinitializeのときには必ず行方向にデータを読み込むものとする．たとえばTABLEというpermanent attributeが添字としてpermanent entity PERM1，PERM2をもちそれぞれが3，5個としたときにTABLEに対して図5のようなデータ構造が作られる．

またpackingした場合のデータ構造を図6に示しておく．

Ⅲ．4 Attribute のequivalence

KSIMシステムが自動的にデータ構造を決定するときに最も大きな問題となるのが，attributeのequivalenceの達成である．

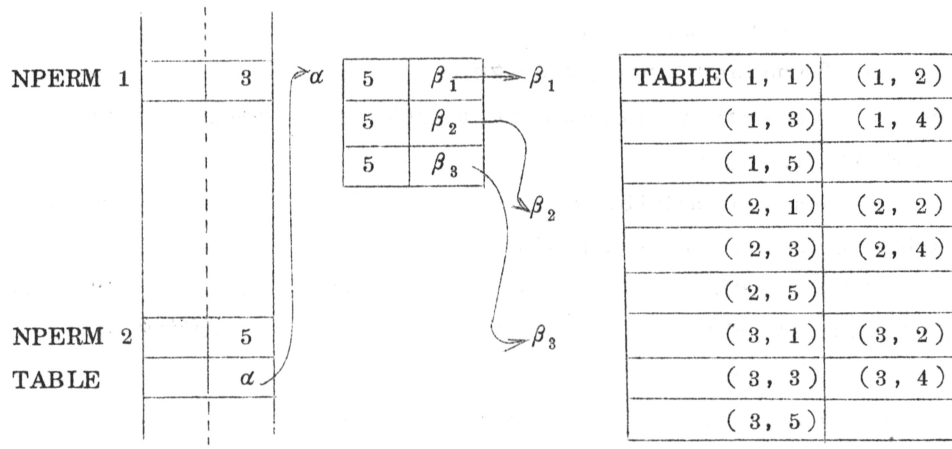
いまFAMLYというsetが定義され，owner entityがsystemでmember entityがtemporary entity JOHN，JIM，BETSYだとすると，KSIMはowner entity



$\beta_2 = \beta_1 + 5$

$\beta_3 = \beta_2 + 5 = \beta_1 + 10$

図5. Permanent attribute TABLEのデータ構造
（attributeのpackingを行なわないとき）



$\beta_2 = \beta_1 + 3$

$\beta_3 = \beta_2 + 3 = \beta_1 + 6$

図6. Permarent attribute TABLEのデータ構造
（attributeのpackingを行なうとき）

に対してFFAMLY，LFAMLYという２つのsystem attribute，member entity のそれぞれに対してPFAMLY，SFAMLYという２つのtemporary attributeを定義する.

FAMLYのmember entityが他のsetのmemberになっていなければ問題はないが，JOHNはまたset SONのmemberであり，BETSYはset WIFEのmemberであるとすると，それぞれのtemporary entityの構造は図７のようになるかもしれないし，ある場合には図８のようになるかもしれない.

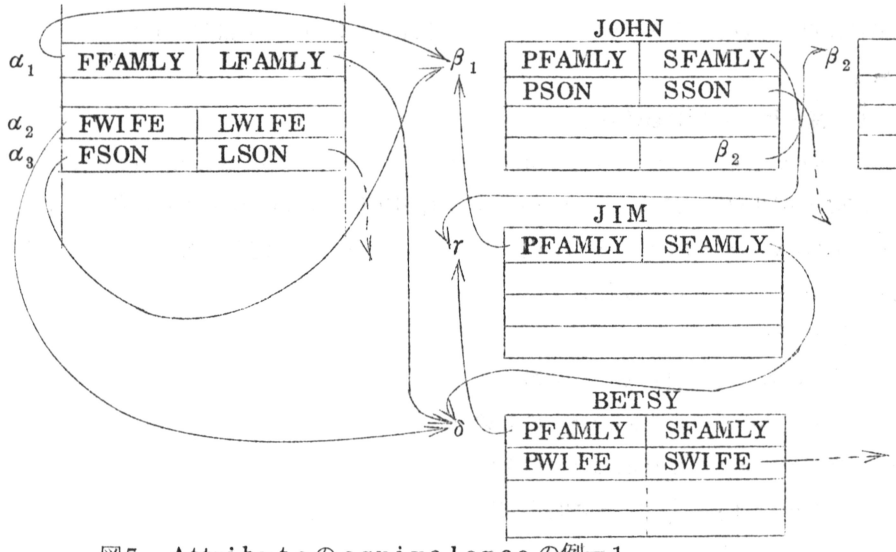


図7　Attributeのequivalenceの例－１

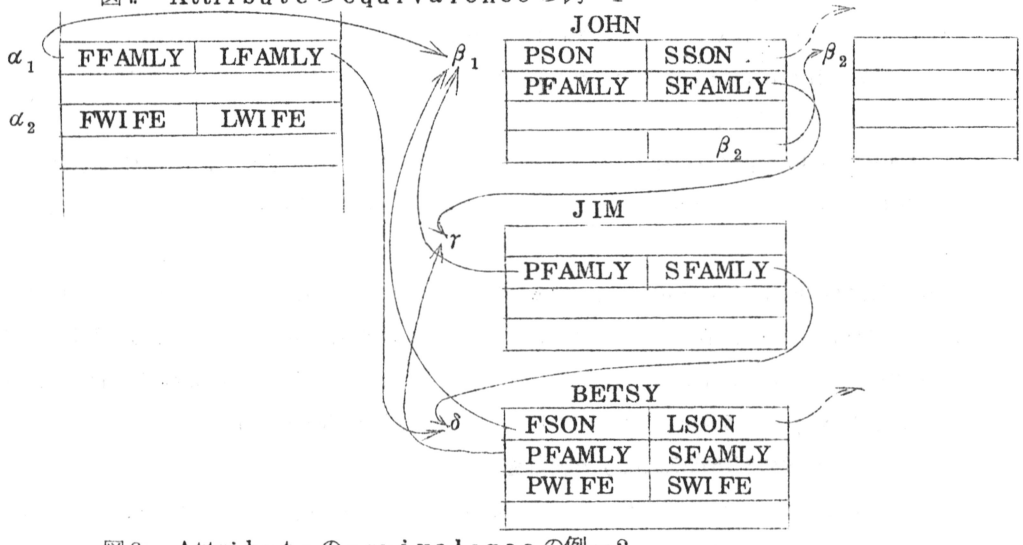


図8　Attributeのequivalenceの例－２
set SONのowner entityがBETSYのとき

このようなequivalenceが自動的に達成できるが，その反面ある条件のもとではメモリの使用効率が悪くなることがあることを注意しなければならない．

Ⅳ　統計値の自動収集機能

自動収集する統計値としてはGPSS程度を目標としたため，その対象となるのはあくまでも待ち行列型のシステムである．

その他のシステムのための統計値に対してはuserが自由に収集および計算ができるようにACCUMULATE，COMPUTEステートメントはSIMSCRIPTと同様に使用できるようになっている．

Ⅳ．1　待ち行列に関する統計値

待ち行列はsetと考えるが，統計値を求めるためには，いくつかのattributeが必要になるのでDEFINITIONで特別なステートメントQUEUEを用いて待ち行列の名前およびowner entity，member entity，必要ならばID番号を定義しておかねばならない．

自動的に収集され計算される統計値は

(1) 待ち行列の最大長さ
(2) 待ち行列に入ったentityの総個数
(3) 待ち行列で待たなかったentityの個数
(4) 待ち行列で待たなかったentityの全体に対する割合
(5) すべてのentityに対する平均待ち時間
(6) 待たされたentityに対する平均待ち時間
(7) すべてのentityに対する待ち時間の分散
(8) 待たされたentityに対する待ち時間の分散
(9) 現在の待ち行列長さ

で，LIST STATISTICSステートメントを実行するたびに出力する．またCLEAR STATISTICステートメントを実行することによりすべての値を0にセットすることができる．

KSIMはQUEUEのowner entityに対しては8個，member entityに対しては1個のattributeを自動的に生成する．これらのattributeの値の更新はFILEおよびREMOVEステートメントが実行されるたびに行なわれる．またこれらのattributeを参照するにはQXXijという名前を用いればよい．ここにjはQUEUEのID番号である．

Ⅳ．2　施設の利用状況に関する統計値

施設はpermanent entityでもtemporary entityでもよいが，施設であることを明記するためにDEFINITIONで特別なステートメントEQUIPMENTを用いて施設の名前，その施設を利用するentity（temporaryでもpermanentでもよい）の名前，必要ならばID番号を定義しておかねばならない．

自動的に収集され計算される統計値は

(1) 施設を利用したentityの個数

(2) 合計使用時間

(3) 平均使用時間

(4) 使用時間の分散

(5) 同時に施設を利用したentityの個数

(6) 現在施設を利用しているentityの平均個数

(7) 施設を利用したentityの平均個数

(8) 施設の平均利用率

で，LIST STATISTICSステートメントを実行するたびに出力し，CLEAR STATISTICSステートメントを実行するたびにすべての統計値が0にセットされる.

KSIMではEQUIPMENTとして定義されたentityに対し7個，member entityに対し1個のattributeを自動的に生成する．ただしEQUIPMENTには必ず容量(capacity)が定義されねばならないのでINITIAL CONDITIONでそれを指定しなければいけない.

EQUIPMENTに関する統計値はENTER，LEAVE ステートメントが実行されるたびに，自動的に定義された7個のattributeに累積される.

またこれらのattributeはEXXijという名前で自由に参照することができる．ここにjはEQUIPMENTのID番号である.

QXXij，EXXijというattributeの値をuser が勝手に変更することも可能であるが，その場合には得られた統計値は保証できないことは当然である.

V その他の機能

(1) Setオペレーションに関する機能

DEFINITIONでset を定義する段階では，setの構成(FIFO，LIFO，Ranked)を指定せず，プログラム中でステートメントを組合せることによって任意の構成をとることができる.

たとえばFILE LASTステートメントとREMOVE FIRSTステートメント，あるいはFILE FIRSTステートメントとREMOVE LAST ステートメントを組合せればFIFOとなる.

またranked setに関してはmember entityの特定のattribute の値に注目し，ORDERED フレイズを用いてその値の大きい順または小さい順にFILE することができる．さらにORDEREDフレイズを必要なだけ続けることによって複数個のattribute を用いたrankingを行なうことも可能である．ただしORDEREDフレイズが現われた順にset の先頭のメンバーから順に調べられるのでFILEする際にORDEREDフレイズを複数個使う場

合には注意を要する.

(2) イベントのスケジュールに関する機能

外生的イベント，内生的イベントの区別がなく，イベントに対しては必ずevent noticeが定義されているので，同時刻に起るイベントに対して発生する順位を決定するために，各event noticeにはattributeとして優先度が自動的に定義されている.

Event noticeに優先度を与えるにはDEFINITIONで，PRIORITY ORDER ステートメントを用いるか，event noticeのattribute PRORTY に値を与えればよい. 優先度は必ず0以上の整数値（≦262，143）で大きいものほど高い優先度をもつものと考える.

タイミング ルーチンが参照する予定表はCLOCKという名のsetとして自動的に定義されているので，KSIMの中味を熟知していればset オペレーションのために用意されているステートメントを用いてuserが自由にイベントのスケジュールをすることも可能である.

(3) Look-up table

DEFINITIONでpermanent attributeまたはsystem attributeをlook-up tableとして定義することもできる. この場合には同時に引数および，ステップ関数であるか線型補間するかを指定しなければならない.

look-up tableとして定義したattributeは参照するたびにそのときの引数の値にもとずいてtable look-upを行ない値を決定する.

その他にもいくつかのステートメントを追加して機能の拡張を図っているが，詳細は§Ⅵの Language dictionaryを参照されたい.

Ⅵ Language dictionary

ここで述べるステートメントの他に，報告書の出力のためにreport generatorが用意されており，またFORTRAN のステートメントを自由に挿入することができる.

以下で用いる記号とその意味は次のとうりである.

v	：変数
$i.v$	：整変数
$r.v$	：実変数
e	：算術式
$l.e$	：論理式
$i.e$	：整数型算術式
$r.e$	：実数型算術式
$s.n$	：ステートメント番号
i	：整定数または整変数
f	：ステートメント番号または配列名

{ } ：どれか1つを選択する.

〔 〕 ：optionalなステートメントまたは句

(1) Definition ステートメント

TEMPORARY ENTITY

PERMANENT ENTITY

SYSTEM ATTRIBUTE

SET

QUEUE

EQUIPMENT

EVENT NOTICE

PRIORITY ORDER

(2) Entityステートメント

CANCEL event notice名〔CALLED $i.e$〕

CAUSE event notice名〔CALLED $i.e$〕 AT $r.e$

CREATE { event notice名 / temporary entity名 } 〔CALLED v〕

DESTROY { event notice名 / temporary entity名 } 〔CALLED $i.e$〕

FILE $i.e$ { FIRST / LAST } IN { set名 / queue名〔, BY WEIGHT $i.e$〕 }

FILE $i.e$ { BEFORE / AFTER } $i.e$ IN { set名 / queue名〔, BY WEIGHT $i.e$〕 }

FILE $i.e$ IN { set名 / queue名〔, BY WEIGHT $i.e$〕 }, { HIGH / LOW } ORDERED

attribute名〔, 任意の数のORDERED句〕

REMOVE { FIRST / LAST } v FROM { set名 / queue名〔, BY WEIGHT $i.e$〕 }

〔, IF NONE, 任意のステートメント〕

REMOVE $i.e$ FROM { set名 / queue名〔, BY WEIGHT $i.e$〕 }

〔, IF NONE, 任意のステートメント〕

ENTER $i.e$ TO equipment名〔, BY WEIGHT $i.e$〕

〔, IF NOT, 任意のステートメント〕

LEAVE v FROM equipment名〔, BY WEIGHT $i.e$〕

(3) コントロール ステートメントとコントロール句

DO $s.n$ i , { FOR 句 / FOR EACH 句 / FOR EACH OF 句 }

⋮

⋮

i LOOP

……, FOR $i.v = i.e_1, i.e_2$〔, $i.e_3$, 修飾句〕

……, FOR EACH permanent entity名 $i.v$〔, 修飾句〕

——, FOR EACH $i.v$ OF { set名 / queue名 }〔, 修飾句〕

(4) 修飾句

……, WITH $l.e$

……, WHILE $l.e$

……, UNTIL $l.e$

(5) 算術ステートメント

ACCUMULATE a_i INTO b_i SINCE c_i

a_i, b_i, c_i はreal variable

LET $v = e$

ADD e TO $v_1, v_2, \cdots\cdots, v_n$

SET e TO $v_1, v_2, \cdots\cdots, v_n$

STORE v_1 TO v_2

COMPUTE $v_1, \cdots\cdots, v_{11}$ = function$_1$, ……, function$_{11}$, OF e〔, コントロール句〕

ただしfunctionは次のとうり

NUMBER

SUM

MEAN

SUN-SQUARES

MEAN-SQUARES

VARIANCE

STD-DEV

MAX

MIN

MAX(I)

MIN(I)

CLEAR STATISTICS OF { queue名 / equipment名 }〔, { queue名 / equipment名 }, ……〕

(6) 決定ステートメント

GO TO $s.n$

GO TO $(s.n_1, \cdots\cdots, s.n_i)$, $i.e$

IF $(l.e)$ 任意のステートメント

IF (e) $s.n_1, s.n_2, s.n_3$

IF { set名 / queue名 } { IS / IS NOT } EMPTY, 任意のステートメント

IF $i.e$ { IS / IS NOT } IN { set名 / queue名 }, 任意のステートメント

STOP

(7) 選択ステートメント

FIND { FIRST / LAST } v, { FOR句 / FOR EACH句 / FOR EACHOF句

FIND v = { MAX / MIN } OF e, { FOR句 / FOR EACH句 / FOR EACHOF句

(8) 宣言ステートメント

DIMENSION $v_1(i)$, $v_2(i)$, ……

INTEGER v_1, v_2, ……

REAL v_1, v_2, ……

(9) 入出力ステートメント

LIST STATISTICS OF { queue名 / equipment名 }〔, { queue名 / equipment名 }……〕

READ TIME $r.v$

FORMAT $(s_1, \cdots\cdots, s_i/s_{i+1}, \cdots\cdots, s_n)$

s_i は欄記述子

READ(i, f) list

READ(i) list

WRITE(i, f) list*

WRITE(i) list*

* expressionを含んでもよい

ENDFILE $i.e$

REWIND $i.e$

BACKSPACE $i.e$, $i.e$ {RECORD / FILE}

ADVANCE $i.e$, $i.e$ {RECORD / FILE}

(10) その他のステートメント

CALL subroutine名(arg_1, ……, arg_n)

END

EVENT event名

〔{INTEGER / REAL}〕 FUNCTION function名 (arg_1, ……, arg_n)

SUBROUTINE subroutine名(arg_1, ……, arg_n)

REPORT report名 (arg_1, ……, arg_n)

MAIN

RETURN

DEFINITION

INITIAL CONDITION

……, IF NONE, 任意のステートメント

……, IF NOT, 任意のステートメント

……, {HIGH / LOW} ORDERED attribute名

……, BY WEIGHT $i.e$

Ⅶ おわりに

KSIMの設計にあたり慶応義塾大学工学部の浦昭二教授，中西正和氏，慶応義塾大学情報科学研究所の原田賢一氏，土居範久氏に多くの助言をいただいたことを感謝いたします．

Ⅷ 参考文献

(1) H. M. Markowitz, B. Hausner, H. W. Karr;
"SIMSCRIPT-A Simulation Programming Language,"
RAND Corp., Santa Monica, 1963

(2) "GE－600 SIMSCRIPT Appication Reference Manual",
General Electric, 1969. 10

(3) P. J. Kiviat, R. Villanueva, H. M. Markowitz;
"The SIMSCRIPT Ⅱ Programming Language", Prentice-Hall,

Inc., Englewood Cliffs, New Jersey, 1968

(4) "General Purpose System Simulator Ⅲ User's Manual", IBM Application Program, H20-0163-0 IBM

本 PDF ファイルは 1972 年発行の「第 13 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html