



YBG 命令解説マニュアル

YBG2010X用 2012年8月版

(2012年1月13日改定)

(2012年8月 5日改定)

目次

1.	def 命令	ゲーム定義
2.	scon 命令	シリーズ定数
3.	gcon 命令	広域定数
4.	tvar 命令	チーム変数
5.	svar 命令	シリーズ変数
		<<<特殊変数について>>>
6.	ipage 命令	プレイヤー入力画面
7.	ivar 命令	プレイヤー入力変数
8.	ipre 命令	プレイヤー入力変数初期値設定
9.	cpage 命令	コントローラ入力画面
10.	cvar 命令	コントローラ入力変数
11.	cpre 命令	コントローラ入力変数初期値設定
12.	tlet 命令	チーム変数の計算
13.	slet 命令	シリーズ変数の計算
14.	prop 命令	比例配分
15.	pinv 命令	逆比例配分
16.	dist 命令	入札形式配分
17.	ooption 命令	表示形式指定
18.	opage 命令	出力ページ定義
19.	begintable 命令	表の開始
20.	endtable 命令	表の終了
21.	out 命令	表の内容定義
22.	putpage 命令	表の挿入
23.	getv 命令	変数値の取得
24.	rank 命令	変数値の順位の取得
25.	topv 命令	特定順位の変数値の取得
26.	topt 命令	変数の値が特定順位にあるチーム番号の取得

付録. 主要なエラーメッセージ

【参考文献】久野靖, 「ビジネスゲーム生成システム第7版解説/参照マニュアル」, 筑波大学大学院経営システム科学専攻, 2001 (YBGの命令体系は本文献を原型としている.)

=== ゲームの構成に関する命令 ===

1. def 命令 ゲーム定義

ゲーム構成を定義するもので、必須である。

def max-team ** :チーム数 (半角数字)

def max-round ** : 最大ラウンド数(半角数字) ここで設定した数より 1 だけ
少ないラウンドまで実行可能。

注) 実行したラウンドの結果表示を行うためには, ラウンド数を
1 だけ進める必要があるため.

=== 定数に関する命令 ===

以下では、モデル内で利用する定数について説明する。

定数は、全角文字とする。(英数字も全角を使う。半角を使うと結果は保証されない。)

2. scon 命令 シリーズ定数

第0ラウンドから、第1、第2、・・・第nラウンドまで各チーム共通の定数の値を設定する。ゲーム中に値の変更はできない。

scon 変数名 [初期値]

例) scon 需要 100 120 135 150 166 180 196 : 定数名は日本語使用可。

この例では、第0ラウンド需要=100、第1ラウンド需要=120、第2ラウンド需要=135、(中略)第6ラウンド需要=196となる。

3. gcon 命令 広域定数

ゲーム中の全ラウンドを通じて各チームに共通な定数を設定する。ゲーム中に値の変更はできない。

gcon 変数名 [初期値]

例) gcon 金利 0.01

この例では、ゲーム中の金利は、1%で固定となる。

=== 変数に関する命令 ===

以下では、モデル内で利用する変数について説明する。

変数は、全角文字とする。(英数字も全角を使う。半角を使うと結果は保証されない。)

4. tvar 命令 チーム変数

ゲーム中に計算される、各チームの状態を表すチーム変数の名前をあらかじめ定義する。値はゲーム中の計算により自由に変更できる。文字列を代入することも可能である。

tvar 変数名 [初期値]

例) tvar 発注数 100 120 : 第0ラウンドの発注数は100, -1ラウンドの発注数は120となる。-5ラウンドまで設定可能。

チーム変数は、後述する tlet 命令により計算が行なわれると、その結果が入る。

第0ラウンドに初期値が設定されている場合に、第1ラウンドでその値を継承して使いたい時は、下記のように tlet 命令 (後述) を実行して値を移す必要がある。

例) tvar 現金残高 1000 : チーム変数として「現金残高」を定義し、第0ラウンドの値を1000とする。

tlet 現金残高 = 現金残高@1; : 1つ前のラウンドの値を移す。

tlet 現金残高 = 現金残高@1 + 今期現金収入; : 今期の計算を行なう。

ゲーム開始時点 (第1ラウンド) の「現金残高」は下表のように設定されているが、tlet 命令を実行することにより、新しい値が代入される。

現金残高	ラウンド0	ラウンド1	ラウンド2	ラウンド3	ラウンド4
チーム1	1000	未定義	未定義	未定義	未定義
チーム2	1000	未定義	未定義	未定義	未定義
チーム3	1000	未定義	未定義	未定義	未定義
チーム4	1000	未定義	未定義	未定義	未定義

<<文字変数>>

tlet 命令により、tvar に文字を代入することも可能である。

```
例)   tvar 経営状況
       tlet 経営状況 = “良好”
```

この例では、チーム変数「経営状況」に、「良好」という文字列が代入される。
opage 命令で表示すると、文字がそのまま表示される。

<<相対ラウンド指定>> @指定：@が一つ

変数名の後に、@1、@2、@3・・・をつけると、その変数の1ラウンド前、2ラウンド前、3ラウンド前・・・の値を参照することができる。

例) 発注数@1 ：現在のラウンドより1つ前のラウンドの発注数を示す。

<<絶対ラウンド指定>> @@指定：@が二つ

変数名の後に、@@1、@@2、@@3・・・をつけると、その変数の第1ラウンド、第2ラウンド、第3ラウンド・・・の値を参照することができる。

例) 発注数@@1 ：第1ラウンドの発注数を示す。

5. svar 命令 シリーズ変数

各ラウンドで全チーム共通の1つの変数名を設定する。値の設定は、slet 命令（後述）により各ラウンドで行う必要がある。（行わないと0が入る。）

```
svar 変数名 [初期値]
```

例) svar 実需要 100 200 300

この例では第0ラウンドの実需要は100、-1ラウンドの実需要は200、-2ラウンドの値は300となる。-5ラウンドまで設定可能。初期値は省略可能(ゼロとなる。)。また、初期値と初期値の間は半角スペースとする。

<<< 特殊変数について >>>

次の変数名は、システムの特変数として定義済みである。

(変数名)	(内容)
チーム	チーム番号
ラウンド	現在のラウンド数
チーム数	max-team で定義されているチーム数

例) tlet if(チーム = 1 && ラウンド = 1) {資本金 = 999999}
第1ラウンドで、特定のチームの変数の初期値を変更する。

=== 入力画面の作成に関する命令 ===

ここでは各ラウンドで、プレーヤが意思決定変数を入力するための画面について説明する。

6. ipage 命令 プレーヤ入力画面

1つの入力ページの開始を示す。ivar 命令 (必須)、ipre 命令 (必要に応じて) とセットになる。HTMLを記述できる。

ipage 英語名	日本語名 [if(条件式)]	: 日本語名はP C画面上に表示される。
HTML 記述		: そのままP C画面上に表示される。 (始めに半角スペースを入れる.) : 条件式を書くと、条件を満足する時 だけ、入力画面が表示される。

例) ipage price 価格の入力 if(チーム = 4)
<H1>価格入力</H1>
<P>販売価格を入力して下さい。</P>

: チーム4だけこの入力画面が表示されるので、プレーヤによって役割を変えることができる。ただし、Y B Gの構造上、ここでivarで指定した変数は、他チームにも同一の変数が設定されるが、入力画面が表示されないため、エージェント入力により初期値を強制的に輸入してゲームを進める必要がある。また、この変数をチーム4に適用する場合、tlet if(チーム = 4){ 数式 }の処理を行う必要がある。

7. ivar 命令 プレーヤ入力変数

プレーヤの入力変数と入力種別を定義する。一旦入力した値は、ゲーム中に変更はできない。(tlet命令でivarに値を書き込もうとするとエラーが表示される。ivarの値を変更して利用したい場合は、別名のtvar変数にコピーすることで変更できる。)

(1) 数値入力型 range

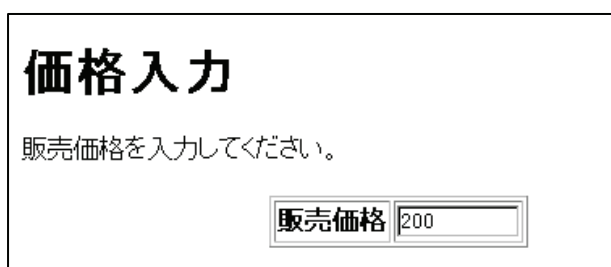
ivar 変数名 range 最小 最大 [初期値]

例) ipage price 価格の入力 : 入り口ページに「価格の入力」と表示。

<H1>価格入力</H1> : 下記のように画面に表示。

<P>販売価格を入力してください。</P> :

ivar 販売価格 range 0 1000 200 : 下限 0 上限 1000 初期値 200



(2) 項目選択型 select

ivar 変数名 select 初期数値 数値 名前 数値 名前 . . . : 数値と名前は対。

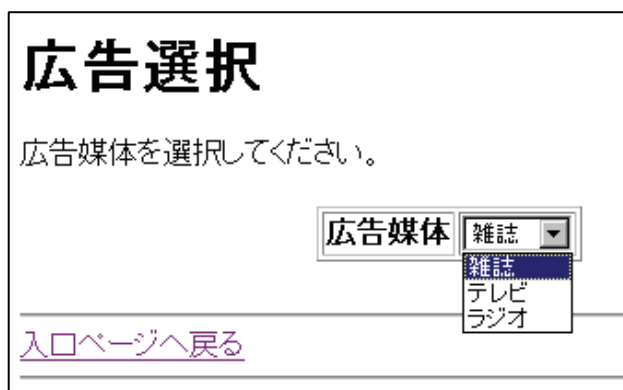
例) ipage advertisement 広告の選択 : 入り口ページに「広告の選択」と表示。

<H1>広告選択</H1> : 下記のように画面に表示。

<P>広告媒体を選択してください。</P> :

ivar 広告媒体 select 1 1 雑誌 2 テレビ 3 ラジオ : 初期値は1が設定されている。

↑
初期値

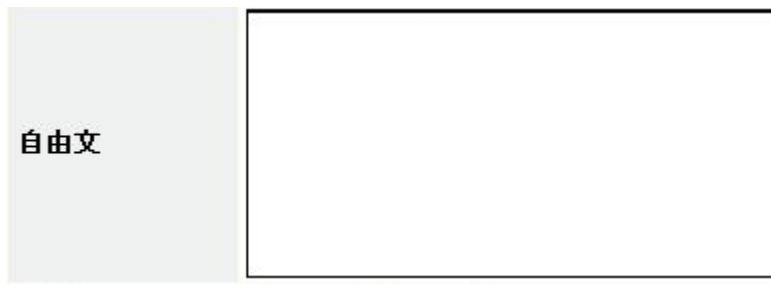


(3) 自由記述型 textarea

ivar 変数名 textarea 文字数 行数 注) 文字数は半角文字でカウント

入力パラメータに自由記述が可能。最大 256Byte (全角文字で 128 文字)

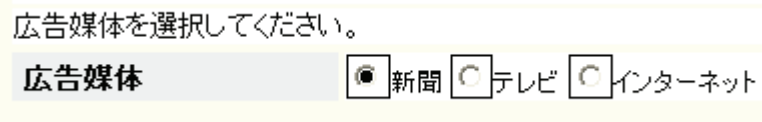
例) ivar 自由文 textarea 40 8



(4) ラジオボタン型 radio

ivar 変数名 radio 初期数値 数値 名前 数値 名前 . . . : 数値と名前は対。

例) ivar 広告媒体 radio 1 1 新聞 2 テレビ 3 インターネット
↑
初期値



8. ivar の付属命令

- (1) ipre 命令 入力変数初期値設定
ivar 変数の第 0 ラウンド以前の値を初期設定することができる。
- 5 ラウンドまで設定可能.

例) ipre 金利 0.02 0.03 0.04

この例では、ivar 変数で別途定義された「金利」の初期値として、第 0 ラウンドは 2%、
- 1 ラウンドは 3%、- 2 ラウンドは 4% が設定されている。

- (2) ichk 命令 入力変数の条件チェック
複数の ivar 変数の合計値などをチェックすることができる.

例) ichk “予算オーバーです.” if((広告費 + 販促費) > 予算残@1)

注: “ ” は半角文字.

この例では、広告費と販促費の入力の合計が、前期の予算残以上の金額を越えると、「予算オーバーです。」というメッセージが出るので、プレーヤは入力をやり直す.

=== コントローラの入力に関する命令 ===

ここでは各ラウンドでコントローラが入力するための画面について説明する。(これによって、ゲーム中にコントローラがゲーム進行を調整することも可能になる.)

9. cpage 命令 コントローラ入力画面

コントローラ用の入力ページの開始を示す。cvar 命令、cpre 命令とセットになる。HTML も記述可能。

[書式]

書式 cpage 英語名 日本語名
日本語名はそのまま HTML 出力される。

例) cpage price 価格の入力
<H1>価格入力</H1>
<P>販売価格を入力して下さい。</P>

10. cvar 命令 コントローラ入力変数

コントローラの入力変数と入力識別を定義する。

(1) 数値入力型 range

cvar 変数名 range 最小 最大 [初期値]

(2) 項目選択型 select

cvar 変数名 select 初期数値 数値 名前 数値 名前 . . . : 数値と名前は対。

※ 初期値は、cpage を表示したときに、デフォルトで表示される値として使用される。また、未入力状態でモデル計算を行った場合に、自動的に入力値として使用される値である。

※ 初期値を省略した場合、cpage ではデフォルトでは空白 (select の場合は最初に定義されている値) が表示される。未入力状態でモデル計算を行うと、未入力の cvar を含む計算は実行されない。

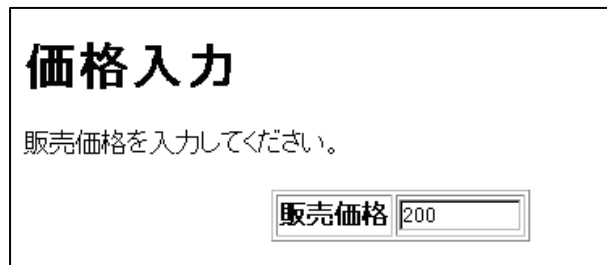
例 1)

cpage price 価格の入力 : 入り口ページに「価格の入力」と表示。

<H1>価格入力</H1> : 下記のように画面に表示。

<P>販売価格を入力してください。</P> :

cvar 販売価格 range 0 1000 200 : 下限 0 上限 1000 初期値 200



例 2)

cpage advertisement 広告の選択 : 入り口ページに「広告の選択」と表示。

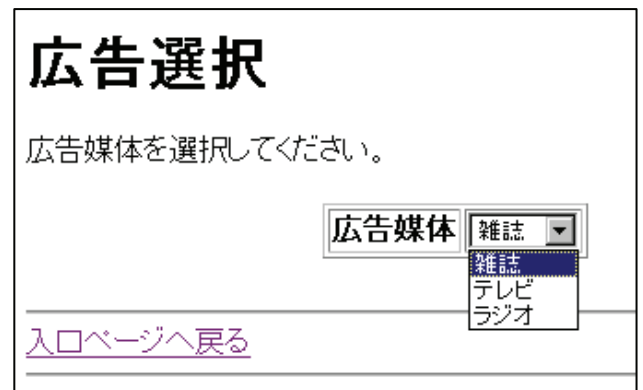
<H1>広告選択</H1> : 下記のように画面に表示。

<P>広告媒体を選択してください。</P> :

cvar 広告媒体 select 1 1 雑誌 2 テレビ 3 ラジオ

↑

初期値は 1 が設定されている。



1 1. cpre 命令 コントローラ入力変数初期値設定

cvar 変数の第 0 ラウンド以前の値を初期設定することができる。

cpre 変数名 初期値

例) cpre 金利 0.01 0.02 0.03

この例では、cvar 変数で別途定義された「金利」の初期値として、第 0 ラウンドは 1%、-1 ラウンドは 2%、-2 ラウンドは 3% が設定されている。

=== 計算式の作成に関する命令 ===

12. tlet 命令 チーム変数の計算

各チームの変数の値を計算するための数式を定義する。数式内の全ての変数や定数はあらかじめ定義されていなければならない。計算はソースコードに記述された順番に上から実行されるので、計算の順序を考えておくことが必要である。

tlet 数式 : 数式は右辺を計算した結果を左辺に代入する。
数式の記号 (+ - * / など) は半角。
記号の前後には半角スペースを入れる。

<<加減乗除>>

```
tlet 費用 = 人件費 + 広告費
tlet 売上高 = 販売価格 * 販売数
tlet 粗利益 = 売上高 - 費用
tlet 一人当売上高 = 売上高 / 人員数
```

<<条件分岐>>

if then else 文を用いて、条件により異なった計算を行うことができる。

```
tlet if(条件文) {実行文}
```

例)

①イコールを判定する場合

```
tlet if(受注数 = 在庫数){出荷数 = 在庫数}
```

実行文が複数あるときは、行の1文字目を空白とし、各行の最後に区切りとして、半角セミコロンの「;」をつける。

```
tlet if(受注数 = 在庫数){  
  出荷数 = 在庫数;  
  在庫数 = 0;  
}
```

②イコール or 小 を判定する場合は、条件文には <= と書く。(=< ではない。)

```
tlet if(入力価格 <= 最低有効価格){販売価格 = 最低有効価格}
```

③イコール or 大 を判定する場合は、条件文には >= と書く。(=> ではない。)

```
tlet if(入力価格 >= 最高価格){販売価格 = 最高価格}
```

④条件を満足する場合と、しない場合で実行内容を変えたい場合は、else を使う。

```
tlet if(入力価格 <= 最低有効価格){販売価格 = 最低有効価格}  
  else{販売価格 = 入力価格}
```

⑤条件を二重に書く場合

チーム1の現金貯金を第1ラウンドに10000に設定する場合は、

```
tlet if(チーム =1 && ラウンド = 1){現金貯金 = 10000}
```

注) && は and 条件を示す. or 条件は || で表す.

または次のようにも書ける。

```
tlet if(ラウンド = 1) {  
  if(チーム = 1){現金貯金 = 10000}  
  else{現金貯金 = 20000}          *この場合、他のチームは20000になる。  
}
```

⑥ not イコールを判定する場合は、条件文には != と書く。

```
tlet if(ラウンド != 1){剰余金 = 剰余金@1 + 経常利益}
```


<<数学関数>>

以下の数学関数を利用することができる。

tlet 変数名 = 数学関数 ;

int(x) : 小数点以下の切捨て

例) tlet 生産可能数 = int(部品在庫数 / 所要量)

rint(x) : 小数点以下の四捨五入

min2(a, b) : a と b のうちの小さいほうの値

例) tlet 生産数 = min2(生産可能数, 生産指示数)

max2(a, b) : a と b のうちの大きなほうの値

sqrt(x) : x のルート

pow(x, y) : x の y 乗

log(x) : 自然対数

exp(x) : e の x 乗

rand(x) : 0 以上 X 未満の一様乱数

記述例) svar 乱数

slet 乱数 = rand(10) # 0 以上 10 未満の整数

注 1) rand(x) の x は, 1 以上の値を書く。

注 2) 乱数の seed は指定しないとゼロであり, 常に同じ乱数系列となる。

指定する場合は以下のように書く。 svar seed 100

使用例) 商品需要を乱数で変化させる方法

以下のようなソースコードを書くと、商品需要を、商品基礎需要からプラスマイナス最大 50 まで変化させることができる。

```
scon 商品基礎需要 300 300 300 300 300 300 300 300 300 300
```

```
svar 商品需要
```

```
svar 乱数 1
```

```
svar 乱数 2
```

```
svar seed 99 # 変数 seed の初期値に任意の数を書くと乱数の列を変えられる。
```

```
slet 乱数 1 = rand(50) # 乱数 1 は 0 から 50
```

```
slet 乱数 2 = rand(50) # 乱数 2 は 0 から 50
```

```
slet 商品需要 = int(商品基礎需要 + 乱数 1 * 乱数 2)
```

実行結果

Round:	00	01	02	03	04	05	06	07	08	09
商品基礎需要	300	300	300	300	300	300	300	300	300	300
商品需要	0	293	307	269	295	311	323	319	331	287
乱数1	0	38	28	2	28	14	32	46	42	22
乱数2	0	45	21	33	33	3	9	27	11	35
seed	99	16745	11871	40133	32283	16353	11159	50877	13011	36185

<<メッセージ出力>>

tmsg/gmsg 命令

プレイヤーに対してメッセージを通知する。ゲーム中に何かのイベントが発生した場合などに、その内容をメッセージとして、メッセージ表示画面に通知できる。

tlet tmsg(“資金が残り少なくなりました。”) :各チーム別のメッセージ
tlet gmsg(“景気は上昇に転じそうです。”) :全チーム共通のメッセージ

注： (“ ”) は半角文字

tmsg()は、当該チーム宛のメッセージを生成する。(他のチームにはそのメッセージは見られません。) gmsg()は全チームに同じメッセージを生成する。

特定の条件に該当するチームにメッセージを出力したり、特定の条件のときに全チームに共通メッセージを出力したりするために、if文の中で利用するのがよい。

メッセージの中に、\${変数名}を書くと、その部分の変数の値が表示される。

例) tlet if(販売価格 < 仕入価格){gmsg(“チーム\${チーム}が原価割れで販売しています”)}

*原価割れで販売しているチームがあると、それを全チームに知らせる。

注：このメッセージ出力命令の代わりに、変数に文字を代入して、出力の表(opage)に表示することもできる。次ページ参照

*出力ページ (opage) へのメッセージ表示

(例1)

```
tvar 経営状況
tlet 経営状況 = “順調”
#
opage joukyo 経営の状況 public
begintable
out teams
out teams-vars 経営状況
endtable
```

Team:	01	02	03
経営状況	川順調	川順調	川順調

(例2)

```
tvar 景気
tlet if(ラウンド = 1){景気 = "景気上昇中"}
#
opage joukyo 経営の状況 public
  <p>第${ラウンド}ラウンド,${景気}</P>
begintable
out teams
out teams-vars 受注数
endtable
```

第01ラウンド,景気上昇中

Team:	01	02	03
受注数	33	33	33

<<メッセージの結合>> concatenate 命令で2つのメッセージを結合することができる。
メッセージが増えていく場合などに用いる。

```
tlet メッセージ = concatenate(メッセージ1,メッセージ2)
```

13. slet 命令 シリーズ変数の計算

シリーズ変数 `svar` の値を計算する為の数式を定義する。数式内の全ての変数や定数は予め定義されていなければならない。slet 式の中で使用できるのは、`svar`, `scon`, `gcon`, 数値および下記の集約関数であり、`tvar`, `ivar` は使用できない。

slet式では、tlet式と同様の記述が可能である。

(集約関数)

slet式では次の集約関数を使用可能である。

最大値：`maxt`(チーム変数)

最小値：`mint`(チーム変数)

合計値：`sumt`(チーム変数)

平均値：`avgt`(チーム変数)

例) 各チームの販売価格の平均値を求める。

```
svar 業界平均価格 #業界に一つだけ存在する
```

```
tvar 販売価格 #チーム毎の販売価格
```

```
slet 業界平均価格 = avgt (販売価格)
```

=== 配分のための計算に関する命令 ===

14. prop 命令 比例配分

ある変数の値を、他の変数に比例して各チームに配分する。シェアの配分などに用いることができる。

prop チーム変数名 = 式1 by 式2 : tlet は不要。

式2の値をチームごとに計算し、その値に比例して式1を配分した結果をチーム変数に入れる。式1や式2は数式を書けるが空白を入れてはならない。

例) prop 受注数 = 総需要 by 広告費

各チームの受注数は、総需要を各チームの広告費に比例配分した値となる。広告費の多いほうが受注数が多くなる。

注1 式2がゼロだと配分結果は0になるので注意。

15. pinv 命令 負の比例配分

ある変数の値を、他の変数の逆数に比例して各チームに配分する。シェアの配分などに用いることができる。

pinv チーム変数名 = 式1 by 式2 : tlet は不要。

式2の逆数をチームごとに計算し、その値に比例して式1を配分した結果をチーム変数に入れる。式1や式2は数式を書けるが空白を入れてはならない。

例) pinv 受注数 = 総需要 by 販売価格

各チームの受注数は、総需要を各チームの販売価格の逆数に比例配分した値となる。販売価格が低いほうが受注数が多くなる。

注1 式2がゼロだと配分結果は0になるので注意。

16. dist 命令 入札形式配分

ある変数の値を、他の変数が最大または最小のチームから順に配分する。この時、同じ値の複数のチームが存在した場合は、乱数を発生させて特定のチームを決定する。

`dist` チーム変数名 = 式1 `by` 式2 [`limit` 式3] [`max|min`] : `tlet` は不要。

`max or min` 式2の大きい順または小さい順の指定。省略時は `min` とする。

`limit` 式3の値だけ配分する。省略時は 1 とする。

式1、式2、式3には数式、数学関数、集約関数の記述が可能。

例) `dist` 受注数 = 総発注数 `by` 入札価格 `limit` 100 `max`

各チームの受注数は、入札価格の高いチームから順番に 100 ずつ配分される。

各チームに配分する毎に総発注数が減らされていき、ゼロになった時点で残りのチームには配分されない。

また、入札価格が同じチームが複数存在した場合は、乱数でチームを特定する。

注1 式2の値がゼロだと配分結果は0になるので注意..

=== 出力画面の作成に関する命令 ===

ここでは各ラウンドの計算結果を表示する出力画面について説明する。

17. ooption 命令 表示形式指定

出力に関する表示形式を指定する。

option fmt %0.*[c] : 0.*は、小数点以下の表示桁数を示す。
C はカンマ付数字の指定。

例) option fmt %0.3 : 小数点以下3桁まで表示 1.234, 3.141 など
option fmt %0.0 c : 整数部のみ表示, カンマ付 12,345 など

指定すると、その後にある opage 命令（後述）から表示形式が変更される。
opage 毎に指定しても良い。

小数点以下を省略すると、小数第1位が四捨五入されて整数で表示されるので、実際は違う数字でも同じに見える場合があるので注意が必要。

18. opage 命令 出力ページ定義

1つの出力ページの開始を示す。HTMLを記述できる。

opage 英語名 日本語名 [public/teamspec/control] [if(条件式)]
HTML 記述

<日本語名> PC画面上に表示される。

<HTML 記述> そのままPC画面上に表示される。

HTML 記述の中で、\${変数名} を指定すると、その部分は変数の値に置き換えられて表示される。変数として「ラウンド」、「チーム」という指定をすると、ラウンド番号、チーム番号が入る。変数名には、@1 や式は使えない。

<ページのアクセス制限指定>

public : このページは全チームが参照できる。

teamspec : このページは各チームが自チームの分だけを参照できる。

control : このページはコントローラ（審判）だけが参照できる。

<条件式> 条件を満足する時だけ、出力画面が表示される。

19. begintable 命令 表の開始

出力ページの表の開始を指定する。

```
begintable
```

この命令から endtable 命令までが一つの表になる。

20. endtable 命令 表の終了

出力ページの表の終了を指定する。

```
endtable
```

HTML 記述 : 表の後に表示される。\${変数名} の指定も可能。

21. out 命令 表の内容定義

表の内容を出力する。以下に示す形式が選べ、1行分の出力の場合も、複数行出力の場合もある。out 命令は、begintable と endtable の間に置かなければならない。

① out teams と out rounds

out teams はチーム番号、out rounds はラウンド番号を横一列に出力する。見出し用に使う。

③の例に、out teams を示す。

⑤の例に、out rounds を示す。

② out values 出力指定

values では出力指定の各項目を1セルずつ横に出力するような、表の1行を指定する。出力指定は次のどれかである。

' 文字列 : その文字列がそのまま表示される。見出し用。

- : そのセルは空欄となる。

計算式 : tlet 命令と同様に任意の計算式が記述でき、その値が表示される。

例)

```
option fmt %1.0lf
```

```
opage balance 収支の状況 teamspec
```

```
<H1>収支の状況</H1>
```

```
<P>第${ラウンド}期、チーム: ${チーム}、総需要: ${商品需要}</P>
```

```
begintable
```

```
out values '項目' '残高' '収入' '支出'
```

```
out values '前期繰越 現金貯金@1 - -
```

```
out values '売上高 - 売上金額 -
```

```
out values '調達費用 - - 調達費用
```

```
out values '生産費用 - - 生産費用
```

```
out values '今期繰越 現金貯金 - -
```

```
endtable
```

収支の状況			
第1期、チーム: 1、総需要: 300			
項目	残高	収入	支出
前期繰越	5000000		
売上高		2700	
調達費用			900000
生産費用			900
今期繰越	-898200		

③ out teams-vars 出力指定

teams-vars では出力指定はそれぞれが1つの計算式を表し、1つの変数につき指定したラウンドにおける各チームの値を横に並べて1行ずつ表示する。1つの出力指定は「見出し: 計算式: 書式」の形をしていて、見出しは左端の列に表示され、その右にチームごとに計算式の値を指定した書式で出力したセルが並ぶ。見出しを省略した場合、計算式がそのまま見出しとして表示される。

書式の指定方法は option fmt での指定と同じである。「: 書式」を省略した場合、option fmt で指定した書式が使われる。

例)

```
option fmt %1.0lf
```

```
opage sales 販売状況 public
```

```
<H1>販売状況</H1>
```

```
<P>第${ラウンド}期: 需要: ${商品需要}</P>
```

```
begintable
```

```
out teams
```

```
out teams-vars 販売価格 出荷数 売上金額
```

```
endtable
```

販売状況				
第1期: 需要: 300				
Team:	01	02	03	04
販売価格	300	200	200	200
出荷数	9	97	97	97
売上金額	2700	19400	19400	19400

④ out vars-teams 出力指定

vars-teams は teams-vars と同様であるが、ただし、横方向に変数が並び、縦方向にチーム 1 からチーム N のデータが並ぶ。

例)

```
option fmt %1.0lf
opage sales 販売状況 public
  <H1>販売状況</H1>
  <P>第${ラウンド}期: 需要: ${商品需要}</P>
begintable
out values 'チーム' '販売価格' '出荷数' '売上金額'
out vars-teams 販売価格 出荷数 売上金額
endtable
```

販売状況			
第1期: 需要: 300			
チーム	販売価格	出荷数	売上金額
Team 01	300	9	2700
Team 02	200	97	19400
Team 03	200	97	19400
Team 04	200	97	19400

⑤ out rounds-vars 出力指定

rounds-vars は teams-vars と同様であるが、ただし、チーム横断ではなく、特定チームについてのラウンド横断となる。

例)

```
option fmt %1.0lf
opage sales 販売状況 teamspec
  <H1>販売状況</H1>
  <P>第${チーム}チーム: 第${ラウンド}期: 需要: ${商品需要}</P>
begintable
out rounds
out rounds-vars 販売価格 出荷数 売上金額
endtable
```

販売状況					
第1チーム: 第1期: 需要: 300					
Round:	-2	-1	0	1	2
販売価格	0	0	0	300	0
出荷数	0	0	0	9	0
売上金額	0	0	0	2700	0

⑥ out vars-rounds 出力指定

vars-rounds は rounds-vars と同様であるが、ただし、横方向に変数が並び、縦方向に各ラウンドのデータが並ぶ。

例)

```
option fmt %1.0lf
```

```
opage sales 販売状況 teamspec
```

```
<H1>販売状況</H1>
```

```
<P>第${ラウンド}期: 需要: ${商品需要}</P>
```

```
begintable
```

```
out values - 'ラウンド' '販売価格' '出荷数' '売上金額'
```

```
out vars-rounds ラウンド 販売価格 出荷数 売上金額
```

```
endtable
```

販売状況

第2期: 需要: 300

	ラウンド	販売価格	出荷数	売上金額
Team 01	-2	0	0	0
Team 01	-1	0	0	0
Team 01	0	0	0	0
Team 01	1	300	9	2700
Team 01	2	300	75	22500
Team 01	3	0	0	0

⑦out rounds-teams 出力指定

EXCEL 等でグラフを書くために，指定した変数の全チーム・全ラウンドの値を表にする．
これをコピー&ペイストすると簡単にグラフが書ける．

例)

```
opage graph グラフ用データ control
```

```
begintable
```

```
out rounds-teams 売上高
```

```
endtable
```

Round/Team:	T01	T02	T03
売上高			
R00	0	0	0
R01	2000000	2000000	2000000
R02	2000000	2000000	2000000
R03	2000000	2000000	2000000

=== デバッグ用画面の作成に関する命令 ===

開発したゲームが正しく動作しているかどうかを確認するための画面である。
以下の例をそのまま記述して、コントローラ用の表示として利用すること。

⑦ out teams-allvars

teams-allvars は teams-vars と同様である。指定したラウンドの全チームの変数をゲームのソースコードに現れた順に表示する。

例)

```
opage allvteam 全変数チーム横断 control
  <H1>第${ラウンド}ラウンド: 全変数チーム横断</H1>
begintable
out teams
out teams-allvars
endtable
```

⑧ out rounds-allvars

rounds-allvars は rounds-vars と同様である。指定したチームの全ラウンドの変数をゲームのソースコードに現れた順に表示する。

例)

```
opage allvround 全変数ラウンド横断 control
  <H1>チーム${チーム}: 全変数ラウンド横断</H1>
begintable
out rounds
out rounds-allvars
endtable
```

2.2. putpage 命令 表の挿入

条件によりプレーヤの表示項目を変更する。

`putpage` は、条件式に参照先の `opage` を呼び出す事を可能とする。
但し、条件式から `ipage` や `cpage` 等の入力形式のページを呼び出す事はできない。

`<putpage opage 名 [if(条件)]>` : `opage` 名は英語名

本命令は、`<putpage XXX if(XX)>`までを1行で記述し、途中で改行することはできない。

`opage` 名は、必ず `putpage` と `if` の間にスペースを入れる。

本命令は、`opage` 及び `ipage` 及び `cpage` 命令以降の行であれば、HTML タグ中に記述する事が可能である。

条件には、チーム変数 `tvar`、シリーズ変数 `svar`、シリーズ定数 `scon` 及び数字が指定可能とするが、条件は単一の条件のみとし、代入式や数式を記述することはできない。

本命令に指定できる `opage` 名は1個までとし、2つ以上指定する場合は、新しい命令として指定する。

条件式は省略可能とし、省略された場合は本命令中に指定された `opage` を無条件に表示する。

書式を間違えて書くと、実行時に何も表示されない。

例1) `ipage` に `putpage` を追加し、`opage` を呼び出す記述例

この例では、チーム1の場合だけ、入力ページ中に出力名 `sales` を持つ出力ページを表示する。

(`ipage` の指定)

`ipage price` 意思決定の入力

```
<table><th><tr>
```

```
<putpage sales if(チーム = 1)>
```

```
</tr></th></table>
```

```
<P>製品販売価格を入力してください。</p>
```

```
<P>製品製造指示数を入力してください。</p>
```

```
<P>材料発注数を入力してください。</p>
```

```
ivar 販売価格 range 0 1200 700
```

(opage の指定)

```
opage sales 販売の状況 public
  <H1>販売の状況</h1>
  <P>第${ラウンド}日、総需要: ${商品需要}</p>
  begintable
  out teams
  out teams-vars 販売価格 受注数
  endtable
```

例 2) opage に putpage を追加し、別の opage を呼び出す記述例

この例では、全てのチームの出力ページ中に出力名 teamstatus を持つ別の出力ページを表示する。

(opage の指定)

```
opage sales 販売の状況 public
  <H1>販売の状況</h1>
  <P>第${ラウンド}日、総需要: ${商品需要}</p>
  <table><th><tr>
  <putpage teamstatus >
  </tr></th></table>
  begintable
  out teams
  out teams-vars 販売価格 受注数
  endtable
```

(opage の指定)

```
opage teamstatus 自社の状況 teamspec
  <H1>自社の状況</h1>
  <P>第${ラウンド}日、チーム: ${チーム}、総需要: ${商品需要}</p>
  begintable
  out values '項目' '単価' '数量' '金額'
  out values '製品販売価格' '販売価格' --
  endtable
```


===== エージェント機能を実現する命令 =====

2.3. getv 命令 変数値の取得

チーム番号と相対的なラウンド番号を指定して、変数の値を取得します。tlet 命令や slet 命令の計算式の中で用います。

[書式] getv(変数, チーム番号, 相対ラウンド番号, エラーコード)

- 「変数」には、値を取得したい変数名を指定します。変数には、tvar で定義されたチーム変数、ivar で定義された入力変数、svar で定義されたシリーズ変数、cvar で定義されたコントロール変数を指定することができます。現時点の仕様では geon や scon で定義された定数の値は指定できません。また、「変数」のあとに相対参照演算子@をつけてはいけません。変数名そのものを記述してください。
- 「チーム番号」には、値を取得したいチーム番号 (1, 2, ...) を指定します。「変数」に指定した変数がシリーズ変数のときには、チーム番号に 0 を指定します。「チーム番号」には、変数や計算式を書くこともできます。その際は変数の値や計算式の評価結果がチーム番号として取り扱われます。
- 「相対ラウンド番号」には、計算が行なわれる時点のラウンドから何ラウンド前の値を取得するかを指定します。0 以上の整数を指定します。「チーム番号」と同様に、ここには変数や計算式を書くこともできます。相対ラウンド番号が 0 の場合は、計算が行なわれる時点のラウンドを意味します。なお、通常シリーズ変数の過去の値は、相対参照演算子@を使って取得できますので、シリーズ変数に対して getv 関数を使って値を取得する場面はほとんどありません。しかしながら相対ラウンド番号を計算によって変化させながら値を取得する場合には getv 関数を利用する必要があります。
- 「エラーコード」には、取得しようとする変数の値が（変数データベース上に存在しないという理由によって）値が取得できなかったときに、代わりに使われる値を指定します。数値または文字列、あるいは計算式を指定することができます。通常、エラーコードには、取得したい変数が取りえない値を指定します。たとえば、チーム変数「販売価格」は、負の値を取りませんが、このような変数に対して、「エラーコード」に -1 を指定し、getv 関数の値を if 文で評価することによって、変数の値が正しく取得できたかどうかを判断することができます。エラーコードを活用することで、モデル計算における計算の確実性と正確性を向上させることができます。なお、getv 関数が「エラーコード」を返すのは、典型的には 0 ラウンドよりも前の値を取得しようとする場合に起こります。

使用例

```
tlet 前期営業利益最大チーム販売価格
    = getv(販売価格, 営業利益最大チーム@1, 1, -1)
tlet if(前期営業利益最大チーム販売価格!= -1) {
    メッセージ
        =concatenate(concatenate("営業利益最大のチームの販売価格は",
            前期営業利益最大チーム販売価格), "円です。")
} else {
    メッセージ
        ="前期営業利益最大チーム販売価格が取得できませんでした"
}
```

解説

1期前に営業利益が最大であったチーム（「営業利益最大チーム@1」）の、1期前の「販売価格」の値を取得し、「前期営業利益最大チーム販売価格」に代入します。前期営業利益最大チーム販売価格が正しく取得できた場合には、その価格を表すメッセージ（文字列）を変数「メッセージ」に代入しますが、そうでない場合には、値が取得できなかったというメッセージ（文字列）が「メッセージ」に代入されます。

2.4. rank 命令 変数値の順位の取得

チーム番号と相対ラウンド番号を指定して、変数の値の順位を取得します。tlet 命令の計算式で用います。同順のチームが存在する場合は、同じ順位が返されます。次の順位は、変数値を昇順（小さい順）あるいは降順（大きい順）に並べたときに、先頭から数えた順位となります。したがって、順位の間隔は必ずしも 1 とはなりません（注意事項参照）。

[書式] rank(変数, チーム番号, 相対ラウンド番号, 順序コード)

- 「変数」には、順位を取得したい変数名を指定します。変数には、tvar または ivar で指定した変数を指定します。「変数」のあとに相対参照演算子@をつけてはいけません。変数名そのものを記述してください。
- 「チーム番号」には、値を取得したいチーム番号 (1, 2, ...) を指定します。「チーム番号」には、変数や計算式を書くこともできます。その際は変数の値や計算式の評価結果がチーム番号として取り扱われます。
- 「相対ラウンド番号」には、計算が行なわれる時点のラウンドから何ラウンド前の値を取得するかを指定します。0 以上の整数を指定します。ここには変数や計算式を書くこともできます。相対ラウンド番号が 0 の場合は、計算が行なわれる時点のラウンドを意味します。
- 「順序コード」には 0 または 1 を指定します。0 は昇順（「変数」の値が小さいほど順位の数値が小さい）を表し、1 は降順（「変数」の値が大きいほど順位の数値が小さい）を表します。

注意事項

順位比較をする変数を V とし、チーム番号が t であるようなチームの変数 V の値 $V(t)$ と表すことにします。さらに値の集合 $\{V(t) \mid t=1, \dots, n\}$ の要素を $v_1 < v_2 < \dots < v_m$ のように（昇順に）並べ、チーム番号の集合 $T_k = \{t \mid V(t) = v_k\}$ を生成します。チーム番号 t に対して、 $t \in T_k$ を満たす集合の添え字 k は一意に定まるので、これを $k(t)$ と表すことにします。このとき rank 関数は、 $|T_1| + |T_2| + \dots + |T_{k(t)-1}| + 1$ を返します。ここで、 $|T_i|$ は集合 T_i の要素数を表し、 $|T_0| = 0$ とします。なお、降順の場合には、 $v_1 > v_2 > \dots > v_m$ のように並べて同様の処理をします。

具体的な説明のために、いま、チーム 1、チーム 2...、チーム 10 のある変数の値が、それぞれ 6、1、4、7、7、3、5、8、3、3 であるとし、この両者の列を“変数値（チーム番号）”の列として 6(1), 1(2), 4(3), 7(4), 7(5), 3(6), 5(7), 8(8), 3(9), 3(10) のように書き表すことにします。これを変数値の小さい順にならべると 1(2), 3(6), 3(9), 3(10), 4(3), 5(7), 6(1), 7(4), 7(5), 8(8) を得ます。この並びの括弧内（すなわちチーム番号）を同じ変数値をとるチーム番号の集合の列として並べると $T_1 = \{2\}$, $T_2 = \{6, 9, 10\}$, $T_3 = \{3\}$, $T_4 = \{7\}$, $T_5 = \{1\}$, $T_6 = \{4, 5\}$, $T_7 = \{8\}$ が得られます。したがって、チーム 2 は順位は $|T_0| + 1 = 1$ 位、チーム 6, 9, 10 の順位は、 $|T_0| + |T_1| + 1 = 0 + 1 + 1 = 2$ 位、チーム 3 の順位は、 $|T_0| + |T_1| + |T_2| + 1 = 0 + 1 + 3 + 1 = 5$ 位、チーム 7 の順位は、 $|T_0| + |T_1| + |T_2| + |T_3| + 1 = 0 + 1 + 3 + 1 + 1 = 6$ （以下同様）となります。

使用例

tlet 順位（昇順） = rank(順位比較対象変数, チーム, 0, 0)
tlet 順位（降順） = rank(順位比較対象変数, チーム, 0, 1)
tlet 順位（昇順） @1 = rank(順位比較対象変数, チーム, 1, 0)

tlet 順位（降順）@1 = rank(順位比較対象変数, チーム, 1, 1)

tlet 順位（昇順）@2 = rank(順位比較対象変数, チーム, 2, 0)

tlet 順位（降順）@2 = rank(順位比較対象変数, チーム, 2, 1)

解説

上の例では、順位比較対象変数の当該期、1 期前、2 期前のそれぞれに対して、自チームのその変数値の順位をそれぞれ昇順、降順で求めています。

表 実行結果の表示例

Team:	01	02	03	04	05	06	07	08	09	10
順位比較対象変数	4	2	5	8	2	5	1	2	1	7
順位比較対象変数@1	7	8	1	2	1	6	6	1	3	4
順位比較対象変数@2	4	1	4	1	5	4	7	4	7	2
順位（昇順）	6	3	7	10	3	7	1	3	1	9
順位（昇順）@1	9	10	1	4	1	7	7	1	5	6
順位（昇順）@2	4	1	4	1	8	4	9	4	9	3
順位（降順）	5	6	3	1	6	3	9	6	9	2
順位（降順）@1	2	1	8	7	8	3	3	8	6	5
順位（降順）@2	4	9	4	9	3	4	1	4	1	8

2 5. topv 命令 特定順位の変数値の取得

指定した変数の特定の順位にある値を取得します。tlet 命令の計算式で用います。同順のチームが複数存在する場合には、それらのチームの順番は全て同じものとして扱われ、次の順位は、先頭から数えた順番になります。したがって、指定した順位を取る変数値が存在しない場合があります。この場合には、topv 関数は変数値として 0 を返すので、注意が必要です（詳しくは注意事項参照）。

[書式] topv(変数, 順位, 相対ラウンド番号, 順序コード)

- 「変数」には、値を取得したい変数名を指定します。変数には、tvar または ivar で指定した変数を指定します。「変数」のあとに相対参照演算子@をつけてはいけません。変数名そのものを記述してください。
- 「順位」には、値を取得したい変数値の順位を指定します。「順位」には、変数や計算式を書くこともできます。その際は変数の値や計算式の評価結果が順位として取り扱われます。「順位」は整数、かつ 1 以上チーム数以下でなければなりません。
- 「相対ラウンド番号」には、計算が行なわれる時点のラウンドから何ラウンド前の値を取得するかを指定します。0 以上の整数を指定します。ここには変数や計算式を書くこともできます。相対ラウンド番号が 0 の場合は、計算が行なわれる時点のラウンドを意味します。
- 「順序コード」には 0 または 1 を指定します。0 は昇順（「変数」の値が小さいほど順位の数値が小さい）を表し、1 は降順（「変数」の値が大きいほど順位の数値が小さい）を表します。

注意事項

いま、チーム 1、チーム 2…、チーム 10 のある変数の値が、それぞれ 6、1、4、7、7、3、5、8、3、3 であるとします。これを小さい順に並べると、1, 3, 3, 3, 4, 5, 6, 7, 7, 8 となります。このとき、変数値 1, 3, 4, 5, 6, 7, 8 のそれぞれの順位は、1, 2, 5, 6, 7, 8, 9 となります。topv 関数は、昇順の順位に 1, 2, 5, 6, 7, 8, 9 を指定した場合には、それぞれ 1, 3, 4, 5, 6, 7, 8 を返しますが、順位に 3, 4, 10 を指定した場合には、いずれも 0 が返ります。したがって、変数値に 0 が存在している場合には、指定した順位が、変数値が存在しない順位なのか、変数値が存在する順位であってかつ、その変数値が 0 であるかの判断は出来ません。その場合には、変数値が 0 をとらないように補正してから当該順位の変数値を取得してください。

使用例

tlet 一位チーム（昇順）変数値 = topv(順位比較対象変数, 1, 0, 0)

tlet 二位チーム（昇順）変数値 = topv(順位比較対象変数, 2, 0, 0)

tlet 三位チーム（昇順）変数値 = topv(順位比較対象変数, 3, 0, 0)

tlet 一位チーム（降順）変数値 = topv(順位比較対象変数, 1, 0, 1)

tlet 二位チーム（降順）変数値 = topv(順位比較対象変数, 2, 0, 1)

tlet 三位チーム（降順）変数値 = topv(順位比較対象変数, 3, 0, 1)

tlet 一位チーム（昇順）変数値@1 = topv(順位比較対象変数, 1, 1, 0)
 tlet 二位チーム（昇順）変数値@1 = topv(順位比較対象変数, 2, 1, 0)
 tlet 三位チーム（昇順）変数値@1 = topv(順位比較対象変数, 3, 1, 0)
 tlet 一位チーム（降順）変数値@1 = topv(順位比較対象変数, 1, 1, 1)
 tlet 二位チーム（降順）変数値@1 = topv(順位比較対象変数, 2, 1, 1)
 tlet 三位チーム（降順）変数値@1 = topv(順位比較対象変数, 3, 1, 1)

解説

上の例では、順位比較対象変数の値（当該期、および1期前）について、それぞれ昇順、降順の順番に関して、1位、2位、3位の変数値を取得しています。

表 実行結果の表示例

Team:	01	02	03	04	05	06	07	08	09	10
順位比較対象変数	4	2	5	8	2	5	1	2	1	7
順位比較対象変数@1	7	8	1	2	1	6	6	1	3	4
一位チーム（昇順）変数値	1	1	1	1	1	1	1	1	1	1
二位チーム（昇順）変数値	0	0	0	0	0	0	0	0	0	0
三位チーム（昇順）変数値	2	2	2	2	2	2	2	2	2	2
一位チーム（降順）変数値	8	8	8	8	8	8	8	8	8	8
二位チーム（降順）変数値	7	7	7	7	7	7	7	7	7	7
三位チーム（降順）変数値	5	5	5	5	5	5	5	5	5	5
一位チーム（昇順）変数値@1	1	1	1	1	1	1	1	1	1	1
二位チーム（昇順）変数値@1	0	0	0	0	0	0	0	0	0	0
三位チーム（昇順）変数値@1	0	0	0	0	0	0	0	0	0	0
一位チーム（降順）変数値@1	8	8	8	8	8	8	8	8	8	8
二位チーム（降順）変数値@1	7	7	7	7	7	7	7	7	7	7
三位チーム（降順）変数値@1	6	6	6	6	6	6	6	6	6	6

2 6. topt 命令 変数の値が特定順位にあるチーム番号の取得

指定した変数の値が、特定の順位にあるようなチームの番号を取得します。tlet 命令の計算式で用います。同順のチームが存在する場合でも、同順のチーム番号はランダムに並べられた上で、上位から指定された順番に位置するチーム番号が返されます（詳しくは注意事項参照 1）。したがって topt は必ず有効なチーム番号を返しますが、変数値が同じである複数のチームが存在する場合、それらの順番はすべて異なるため、厳密な順位が必要な場合には注意が必要です（注意事項 2 参照）。

[書式] topt(変数, 順位, 相対ラウンド番号, 順序コード)

- 「変数」には、比較対象の変数名を指定します。変数には、tvar または ivar で指定した変数を指定します。「変数」のあとに相対参照演算子@をつけてはいけません。変数名そのものを記述してください。
- 「順位」には、比較対象の変数値の順位を指定します。「順位」には、変数や計算式を書くこともできます。その際は変数の値や計算式の評価結果が順位として取り扱われます。「順位」は整数、かつ 1 以上チーム数以下でなければなりません。
- 「相対ラウンド番号」には、計算が行なわれる時点のラウンドから何ラウンド前の値を取得するかを指定します。0 以上の整数を指定します。ここには変数や計算式を書くこともできます。相対ラウンド番号が 0 の場合は、計算が行なわれる時点のラウンドを意味します。
- 「順序コード」には 0 または 1 を指定します。0 は昇順（「変数」の値が小さいほど順位の数値が小さい）を表し、1 は降順（「変数」の値が大きいほど順位の数値が小さい）を表します。

注意事項 1

チーム番号を取得する対象の変数を V とし、チーム番号が t であるようなチームの変数 V の値 $V(t)$ と表すことにします。このとき、topt 関数は、 $V(t_1) \leq V(t_2) \leq \dots \leq V(t_{n-1}) \leq V(t_n)$ をみたすようなチーム番号列 t_1, t_2, \dots, t_n をランダムに生成し、この列の先頭（または末尾）から数えて指定した順位（これを k とする）にあるチーム番号 t_k （または t_{n-k+1} ）を返します。

具体的な説明のために、いま、チーム 1、チーム 2…、チーム 10 のある変数の値が、それぞれ 6、1、4、7、7、3、5、8、3、3 であるとし、この両者の列を“変数値（チーム番号）”の列として 6(1), 1(2), 4(3), 7(4), 7(5), 3(6), 5(7), 8(8), 3(9), 3(10) のように書き表すことにします。これを変数値の小さい順にならべると 1(2), 3(6), 3(9), 3(10), 4(3), 5(7), 6(1), 7(4), 7(5), 8(8) を得ます。この並びの括弧内（すなわちチーム番号）を同じ変数値をとるチーム番号の集合の列として並べると {2}, {6, 9, 10}, {3}, {7}, {1}, {4, 5}, {8} が得られます。このとき topt 関数は 2、9、10、6、3、7、1、4、5、8 のように（ランダムに）チーム番号を並べます。ここで、集合 {6, 9, 10} および {4, 5} からは、ランダムな番号列 9, 10, 6 および 4, 5 が生成されています。topt 関数はこのようにして得られたチーム番号列から順位を計算します。昇順（比較対象の変数値の小さい順）の場合には、左側から順位を数え、降順の場合には右から数えます。例えば、昇順 2 位のチーム番号は 9、降順 3 位のチーム番号は 4 です。

注意事項 2

ある変数値が 1 位のチーム番号を topt 関数で取得するとき、実際には 2 位のチームの変数値、あるいは 3 位のチームの変数値なども同じ値である場合があります。ある変数値が 1 位を取る複数のチームが存在する場合、

たとえば、それぞれのチームの別の変数を参照して評価を行い、目的の処理に移ることが必要な場合があります。この場合には、`topt` 関数で、1 位、2 位、3 位などのチーム番号を取得したのち `getv` 関数を使用して、それぞれのチームの特定の変数値を参照することで、同順の場合の処理を行なうことができます。

使用例

```
tlet 一位チーム (昇順) = topt(順位比較対象変数, 1,0,0)
tlet 二位チーム (昇順) = topt(順位比較対象変数, 2,0,0)
tlet 一位チーム (降順) = topt(順位比較対象変数, 1,0,1)
tlet 二位チーム (降順) = topt(順位比較対象変数, 2,0,1)
tlet 一位チーム (昇順) @1= topt(順位比較対象変数, 1,1,0)
tlet 二位チーム (昇順) @1= topt(順位比較対象変数, 2,1,0)
tlet 一位チーム (降順) @1= topt(順位比較対象変数, 1,1,1)
```

解説

上記では、順位比較対象変数の値の順位を比較し、それぞれその値が 1 位、2 位であるようなチームのチーム番号を求めています。

表 実行結果の表示例

Team:	01	02	03	04	05	06	07	08	09	10
順位比較対象変数	4	2	5	8	2	5	1	2	1	7
順位比較対象変数@1	7	8	1	2	1	6	6	1	3	4
順位比較対象変数@2	4	1	4	1	5	4	7	4	7	2
一位チーム (昇順)	9	9	9	9	9	9	9	9	9	9
二位チーム (昇順)	7	7	7	7	7	7	7	7	7	7
一位チーム (降順)	4	4	4	4	4	4	4	4	4	4
二位チーム (降順)	10	10	10	10	10	10	10	10	10	10
一位チーム (昇順) @1	3	3	3	3	3	3	3	3	3	3
二位チーム (昇順) @1	5	5	5	5	5	5	5	5	5	5
一位チーム (降順) @1	2	2	2	2	2	2	2	2	2	2
二位チーム (降順) @1	1	1	1	1	1	1	1	1	1	1

(参考) rank, topv, topt の使用例

```
#
# rank, topv, topt テスト 2010.8.17
# by Motonari Tanabu
#
# ゲームの規模
def max-team 10
def max-round 5
#
# シリーズ定数
# 広域定数
scon 定数 A1      8      1      2      7      6      8      4      7      4
                5      2      7      4      6      3
scon 定数 A2      4      5      1      3      2      2      1      8      2
                3      1      4      8      1      2
scon 定数 A3      8      1      7      7      7      5      4      1      5
                3      1      2      7      4      8
scon 定数 A4      2      8      7      8      5      1      1      2      8
                8      3      4      5      7      1
scon 定数 A5      4      8      1      4      5      4      5      1      2
                5      1      2      1      7      8
scon 定数 A6      1      7      8      2      1      5      4      6      5
                4      8      8      1      3      2
scon 定数 A7      4      3      4      3      4      8      7      6      1
                2      6      2      7      5      2
scon 定数 A8      1      3      3      4      8      4      4      1      2
                6      2      3      6      8      3
scon 定数 A9      8      4      1      5      7      3      7      3      1
                4      4      4      8      3      2
scon 定数 A10     4      3      8      6      1      2      2      4      7
                1      3      8      1      3      2

#
tvar 順位比較対象変数
tvar 順位比較対象変数@1
tvar 順位比較対象変数@2
tvar 順位 (昇順)
tvar 順位 (昇順) @1
tvar 順位 (昇順) @2
tvar 順位 (降順)
tvar 順位 (降順) @1
tvar 順位 (降順) @2
#
tvar 一位チーム (昇順)
tvar 二位チーム (昇順)
tvar 三位チーム (昇順)
tvar 四位チーム (昇順)
tvar 五位チーム (昇順)
tvar 六位チーム (昇順)
tvar 七位チーム (昇順)
tvar 八位チーム (昇順)
tvar 九位チーム (昇順)
tvar 十位チーム (昇順)
#
```

tvar 一位チーム (降順)
tvar 二位チーム (降順)
tvar 三位チーム (降順)
tvar 四位チーム (降順)
tvar 五位チーム (降順)
tvar 六位チーム (降順)
tvar 七位チーム (降順)
tvar 八位チーム (降順)
tvar 九位チーム (降順)
tvar 十位チーム (降順)

tvar 一位チーム (昇順) @1
tvar 二位チーム (昇順) @1
tvar 三位チーム (昇順) @1
tvar 四位チーム (昇順) @1
tvar 五位チーム (昇順) @1
tvar 六位チーム (昇順) @1
tvar 七位チーム (昇順) @1
tvar 八位チーム (昇順) @1
tvar 九位チーム (昇順) @1
tvar 十位チーム (昇順) @1

tvar 一位チーム (降順) @1
tvar 二位チーム (降順) @1
tvar 三位チーム (降順) @1
tvar 四位チーム (降順) @1
tvar 五位チーム (降順) @1
tvar 六位チーム (降順) @1
tvar 七位チーム (降順) @1
tvar 八位チーム (降順) @1
tvar 九位チーム (降順) @1
tvar 十位チーム (降順) @1

tvar 一位チーム (昇順) @2
tvar 二位チーム (昇順) @2
tvar 三位チーム (昇順) @2
tvar 四位チーム (昇順) @2
tvar 五位チーム (昇順) @2
tvar 六位チーム (昇順) @2
tvar 七位チーム (昇順) @2
tvar 八位チーム (昇順) @2
tvar 九位チーム (昇順) @2
tvar 十位チーム (昇順) @2

tvar 一位チーム (降順) @2
tvar 二位チーム (降順) @2
tvar 三位チーム (降順) @2
tvar 四位チーム (降順) @2
tvar 五位チーム (降順) @2
tvar 六位チーム (降順) @2
tvar 七位チーム (降順) @2
tvar 八位チーム (降順) @2
tvar 九位チーム (降順) @2
tvar 十位チーム (降順) @2

#

tvar 一位チーム (昇順) 変数値
tvar 二位チーム (昇順) 変数値
tvar 三位チーム (昇順) 変数値
tvar 四位チーム (昇順) 変数値
tvar 五位チーム (昇順) 変数値
tvar 六位チーム (昇順) 変数値
tvar 七位チーム (昇順) 変数値
tvar 八位チーム (昇順) 変数値
tvar 九位チーム (昇順) 変数値
tvar 十位チーム (昇順) 変数値

tvar 一位チーム (降順) 変数値
tvar 二位チーム (降順) 変数値
tvar 三位チーム (降順) 変数値
tvar 四位チーム (降順) 変数値
tvar 五位チーム (降順) 変数値
tvar 六位チーム (降順) 変数値
tvar 七位チーム (降順) 変数値
tvar 八位チーム (降順) 変数値
tvar 九位チーム (降順) 変数値
tvar 十位チーム (降順) 変数値

tvar 一位チーム (昇順) 変数値@1
tvar 二位チーム (昇順) 変数値@1
tvar 三位チーム (昇順) 変数値@1
tvar 四位チーム (昇順) 変数値@1
tvar 五位チーム (昇順) 変数値@1
tvar 六位チーム (昇順) 変数値@1
tvar 七位チーム (昇順) 変数値@1
tvar 八位チーム (昇順) 変数値@1
tvar 九位チーム (昇順) 変数値@1
tvar 十位チーム (昇順) 変数値@1

tvar 一位チーム (降順) 変数値@1
tvar 二位チーム (降順) 変数値@1
tvar 三位チーム (降順) 変数値@1
tvar 四位チーム (降順) 変数値@1
tvar 五位チーム (降順) 変数値@1
tvar 六位チーム (降順) 変数値@1
tvar 七位チーム (降順) 変数値@1
tvar 八位チーム (降順) 変数値@1
tvar 九位チーム (降順) 変数値@1
tvar 十位チーム (降順) 変数値@1

tvar 一位チーム (昇順) 変数値@2
tvar 二位チーム (昇順) 変数値@2
tvar 三位チーム (昇順) 変数値@2
tvar 四位チーム (昇順) 変数値@2
tvar 五位チーム (昇順) 変数値@2
tvar 六位チーム (昇順) 変数値@2
tvar 七位チーム (昇順) 変数値@2
tvar 八位チーム (昇順) 変数値@2
tvar 九位チーム (昇順) 変数値@2

```

tvar 十位チーム (昇順) 変数值@2

tvar 一位チーム (降順) 変数值@2
tvar 二位チーム (降順) 変数值@2
tvar 三位チーム (降順) 変数值@2
tvar 四位チーム (降順) 変数值@2
tvar 五位チーム (降順) 変数值@2
tvar 六位チーム (降順) 変数值@2
tvar 七位チーム (降順) 変数值@2
tvar 八位チーム (降順) 変数值@2
tvar 九位チーム (降順) 変数值@2
tvar 十位チーム (降順) 変数值@2

#
# 入力変数と入力ページ
ipage price 意思決定の入力
<P>販売価格を入力してください。</P>
ivar 販売価格 range 0 100000 20000
#
# 計算モデル
#
tlet if(チーム=1){順位比較対象変数=定数 A1;}
tlet if(チーム=2){順位比較対象変数=定数 A2;}
tlet if(チーム=3){順位比較対象変数=定数 A3;}
tlet if(チーム=4){順位比較対象変数=定数 A4;}
tlet if(チーム=5){順位比較対象変数=定数 A5;}
tlet if(チーム=6){順位比較対象変数=定数 A6;}
tlet if(チーム=7){順位比較対象変数=定数 A7;}
tlet if(チーム=8){順位比較対象変数=定数 A8;}
tlet if(チーム=9){順位比較対象変数=定数 A9;}
tlet if(チーム=10){順位比較対象変数=定数 A10;}

tlet 順位比較対象変数@1 = 順位比較対象変数@1;
tlet 順位比較対象変数@2 = 順位比較対象変数@2;

tlet 順位 (昇順) = rank(順位比較対象変数, チーム, 0, 0)
tlet 順位 (降順) = rank(順位比較対象変数, チーム, 0, 1)
tlet 順位 (昇順) @1 = rank(順位比較対象変数, チーム, 1, 0)
tlet 順位 (降順) @1 = rank(順位比較対象変数, チーム, 1, 1)
tlet 順位 (昇順) @2 = rank(順位比較対象変数, チーム, 2, 0)
tlet 順位 (降順) @2 = rank(順位比較対象変数, チーム, 2, 1)

tlet 一位チーム (昇順) = topt(順位比較対象変数, 1,0,0)
tlet 二位チーム (昇順) = topt(順位比較対象変数, 2,0,0)
tlet 三位チーム (昇順) = topt(順位比較対象変数, 3,0,0)
tlet 四位チーム (昇順) = topt(順位比較対象変数, 4,0,0)
tlet 五位チーム (昇順) = topt(順位比較対象変数, 5,0,0)
tlet 六位チーム (昇順) = topt(順位比較対象変数, 6,0,0)
tlet 七位チーム (昇順) = topt(順位比較対象変数, 7,0,0)
tlet 八位チーム (昇順) = topt(順位比較対象変数, 8,0,0)
tlet 九位チーム (昇順) = topt(順位比較対象変数, 9,0,0)
tlet 十位チーム (昇順) = topt(順位比較対象変数, 10,0,0)

```

tlet 一位チーム (降順) = topt(順位比較対象変数, 1,0,1)
tlet 二位チーム (降順) = topt(順位比較対象変数, 2,0,1)
tlet 三位チーム (降順) = topt(順位比較対象変数, 3,0,1)
tlet 四位チーム (降順) = topt(順位比較対象変数, 4,0,1)
tlet 五位チーム (降順) = topt(順位比較対象変数, 5,0,1)
tlet 六位チーム (降順) = topt(順位比較対象変数, 6,0,1)
tlet 七位チーム (降順) = topt(順位比較対象変数, 7,0,1)
tlet 八位チーム (降順) = topt(順位比較対象変数, 8,0,1)
tlet 九位チーム (降順) = topt(順位比較対象変数, 9,0,1)
tlet 十位チーム (降順) = topt(順位比較対象変数, 10,0,1)

tlet 一位チーム (昇順) @ 1= topt(順位比較対象変数, 1,1,0)
tlet 二位チーム (昇順) @ 1= topt(順位比較対象変数, 2,1,0)
tlet 三位チーム (昇順) @ 1= topt(順位比較対象変数, 3,1,0)
tlet 四位チーム (昇順) @ 1= topt(順位比較対象変数, 4,1,0)
tlet 五位チーム (昇順) @ 1= topt(順位比較対象変数, 5,1,0)
tlet 六位チーム (昇順) @ 1= topt(順位比較対象変数, 6,1,0)
tlet 七位チーム (昇順) @ 1= topt(順位比較対象変数, 7,1,0)
tlet 八位チーム (昇順) @ 1= topt(順位比較対象変数, 8,1,0)
tlet 九位チーム (昇順) @ 1= topt(順位比較対象変数, 9,1,0)
tlet 十位チーム (昇順) @ 1= topt(順位比較対象変数, 10,1,0)

tlet 一位チーム (降順) @ 1= topt(順位比較対象変数, 1,1,1)
tlet 二位チーム (降順) @ 1= topt(順位比較対象変数, 2,1,1)
tlet 三位チーム (降順) @ 1= topt(順位比較対象変数, 3,1,1)
tlet 四位チーム (降順) @ 1= topt(順位比較対象変数, 4,1,1)
tlet 五位チーム (降順) @ 1= topt(順位比較対象変数, 5,1,1)
tlet 六位チーム (降順) @ 1= topt(順位比較対象変数, 6,1,1)
tlet 七位チーム (降順) @ 1= topt(順位比較対象変数, 7,1,1)
tlet 八位チーム (降順) @ 1= topt(順位比較対象変数, 8,1,1)
tlet 九位チーム (降順) @ 1= topt(順位比較対象変数, 9,1,1)
tlet 十位チーム (降順) @ 1= topt(順位比較対象変数, 10,1,1)

tlet 一位チーム (昇順) @ 2= topt(順位比較対象変数, 1,2,0)
tlet 二位チーム (昇順) @ 2= topt(順位比較対象変数, 2,2,0)
tlet 三位チーム (昇順) @ 2= topt(順位比較対象変数, 3,2,0)
tlet 四位チーム (昇順) @ 2= topt(順位比較対象変数, 4,2,0)
tlet 五位チーム (昇順) @ 2= topt(順位比較対象変数, 5,2,0)
tlet 六位チーム (昇順) @ 2= topt(順位比較対象変数, 6,2,0)
tlet 七位チーム (昇順) @ 2= topt(順位比較対象変数, 7,2,0)
tlet 八位チーム (昇順) @ 2= topt(順位比較対象変数, 8,2,0)
tlet 九位チーム (昇順) @ 2= topt(順位比較対象変数, 9,2,0)
tlet 十位チーム (昇順) @ 2= topt(順位比較対象変数, 10,2,0)

tlet 一位チーム (降順) @ 2= topt(順位比較対象変数, 1,2,1)
tlet 二位チーム (降順) @ 2= topt(順位比較対象変数, 2,2,1)
tlet 三位チーム (降順) @ 2= topt(順位比較対象変数, 3,2,1)
tlet 四位チーム (降順) @ 2= topt(順位比較対象変数, 4,2,1)
tlet 五位チーム (降順) @ 2= topt(順位比較対象変数, 5,2,1)
tlet 六位チーム (降順) @ 2= topt(順位比較対象変数, 6,2,1)
tlet 七位チーム (降順) @ 2= topt(順位比較対象変数, 7,2,1)
tlet 八位チーム (降順) @ 2= topt(順位比較対象変数, 8,2,1)
tlet 九位チーム (降順) @ 2= topt(順位比較対象変数, 9,2,1)
tlet 十位チーム (降順) @ 2= topt(順位比較対象変数, 10,2,1)


```

tlet 十位チーム (昇順) 変数値@2 = topv(順位比較対象変数, 10, 2, 0)

tlet 一位チーム (降順) 変数値@2 = topv(順位比較対象変数, 1, 2, 1)
tlet 二位チーム (降順) 変数値@2 = topv(順位比較対象変数, 2, 2, 1)
tlet 三位チーム (降順) 変数値@2 = topv(順位比較対象変数, 3, 2, 1)
tlet 四位チーム (降順) 変数値@2 = topv(順位比較対象変数, 4, 2, 1)
tlet 五位チーム (降順) 変数値@2 = topv(順位比較対象変数, 5, 2, 1)
tlet 六位チーム (降順) 変数値@2 = topv(順位比較対象変数, 6, 2, 1)
tlet 七位チーム (降順) 変数値@2 = topv(順位比較対象変数, 7, 2, 1)
tlet 八位チーム (降順) 変数値@2 = topv(順位比較対象変数, 8, 2, 1)
tlet 九位チーム (降順) 変数値@2 = topv(順位比較対象変数, 9, 2, 1)
tlet 十位チーム (降順) 変数値@2 = topv(順位比較対象変数, 10, 2, 1)

#
# 出力指定
ooption fmt %1.0lf
#
opage allvteam 全変数チーム横断 control
<H1>第${ラウンド}ラウンド: 全変数チーム横断</H1>
begintable
out teams
out teams-allvars
endtable
#
opage allvround 全変数ラウンド横断 control
<H1>チーム${チーム}: 全変数ラウンド横断</H1>
begintable
out rounds
out rounds-allvars
endtable
#
# end

```

付録. 主要なエラーメッセージ

(ソースコード修正時)

line(n):メッセージ ソースコードの n 行目にエラーがあることを示す.

- ○○は不正です. : ○○が, tvar, ivar, scon, gcon, svar で定義されてない.
- n 行目からの構文中にエラーがあります. : ソースコードの記述形式に誤りがある.
()や{ }が抜けている場合などが多い.
前述の「○○が不正です」の○○を使用している場合にも出る.
ivar に計算値を代入した場合にも出る.

(ゲーム実行時)

- **ERROR Argument "M-#M-5" isn't numeric in numeric gt (>) at** 以下省略

: 入力データが半角ではなく, 全角で入力されている.

- **ERROR Illegal division by zero at** 以下省略

: 計算式の割り算の分母の値がゼロのため計算できない.

- **opage 命令の表示画面が空白で何も表示されない**

: opage 命令の中で使用している、\${変数名}の変数が未定義

: 表示しようとする文字メッセージの中身が入っていない

- **ERROR Useless use of a constant in void context**

: tmsg, gmsg 命令の書式が正しくない