ビジネスゲーム生成システム第2版解説/参照マニュアル

久野　靖

1997.12.3

1　はじめに

「ビジネスゲーム生成システム」(以下単に生成システム) は、その名前の通りビジネスゲームを行うソフトウェア群を生成するためのソフトウェアです。その特徴としては次のことがらが挙げられます。

- 生成されるゲームは、Web サーバとブラウザを使用して複数のチームによりプレイできる。パソコンで動くゲームプログラムでよく見られるように、入力や結果を紙などでやりとりする必要がない。
- ゲームの制御もプレイと同様ブラウザからすべて行うようになっている。このため、コントローラの座る位置などの制約がない。
- 1 つの記述ファイルから、ゲームに必要なすべてのファイル群を一括して生成する。ゲームを修正する時もこの記述ファイルだけを変更すれば済む。
- ゲームのモデルは計算式の集まりとして記述するが、その変数名として日本語を使うことができるため、(日本人には) 記述しやすく理解しやすい。
- ビジネスゲームに必要とされるような計算手順を専用のコマンドとして提供することにより、プログラミングの苦手な人でも細かいプログラム制御に煩わされることがない。

本文書では、生成システムの概要、使い方、システムが提供しているゲームの概要、記述ファイルの記述形式などについてひととおり説明しています。

2　生成システムの概観

「はじめに」で述べたように、生成システムが生成するゲームはすべて Web サーバ上で動きます。このため、ゲーム開発者は Web サーバ上にそのゲーム用のディレクトリを用意し、ここに記述ファイルを置きます。続いてそのディレクトリで「gg2　記述ファイル」コマンドを動かすと、生成システムが必要なファイルを生成し、環境設定をすべて行います。

あとは、ブラウザからサーバのディレクトリの URL を指定すると、入口ページが表示され、そのままゲームを始めることができます。入口ページには基本的に次の 3 種類のものが入っています。

- 入力ページへのリンク — 各チームが各種データを投入するページへ移動できます。
- 表示フォーム — ゲーム進行に際して、その各種状況を Web ページとして表示するためのボタンがあります。情報の種類によってはチームパスワードを正しく入力しないと見られないものまります (当然、自チームの情報しか見られないわけです)。
- コントローラページへのリンク — コントローラがゲームを制御するためのページへ移動できます。

通常は、各チームは入力ページ (複数ある場合が多い) に移動して、そこで自チームの決定事項を入力します。ゲームはラウンド単位に分かれていて、あるラウンドについて全チームの入力情報が揃ったら、その情報に基づいて計算を行い、次のラウンドに進むことになります。

入力ページにはデータ入力欄のほかに、チーム番号選択メニュー、パスワード欄、提出ボタンがあります。当然、自チームの番号を選択して正しいパスワードを打ち込まなければデータは提出できません。データは一度提出すると変更はできなくなります。

最後に、コントローラページにはラウンド選択メニュー、チーム選択メニュー、および次のようなボタンが備わっています。多くのボタンはそのボタンが押された時に選択されているラウンドやチームに対する操作や表示を行います。そのほかに、多くのコマンドは「カレントラウンド番号」を参照しますが、これはこのページには表示されていません。

- 計算実行 — 指定されたラウンドの計算を行います。
- 入力状況 — どのチームがどのデータを入力済みか表示します。
- 残存入力完了 — カレントラウンドのまだ提出されていないデータをすべて標準値に強制設定します。
- ラウンド番号設定 — カレントラウンドを指定されたラウンド番号に設定します。
- 各種表示ボタン — ボタンごとに各種の情報表示を行います。ゲームによって異なります。

コントローラページで操作を行うには、常にコントローラパスワードを入力する必要があります。

3 ゲームの計算モデル

ゲームは非常に多数の数値を「変数」という形で扱います。これらの変数は次のように分類されています。

- 入力変数 — 上述のように、各ラウンドごとに入力が行われ、さらに各チームがそれぞれ入力するので 2 次元の行列になります (つまり $I_{round,team}$ のように 2 つの添字を取る)。たとえば「商品価格」は毎ラウンドごと、各チーム個別に判断してつけるのでこれに相当します。
- チーム変数 — 入力変数と同じく 2 次元ですが (たとえば「各チームの手持ち金」はチームごとに別で、さらにラウンドごとに別の値になるので、2 次元になるわけです)、値が入力によってではなく内部の計算によって決まることが違います。
- シリーズ変数 — ラウンドごとに別ですが各チームの別はない変数を表します。たとえば「全チーム売上合計」などがこれに当たります。1 次元の変数で S_{round} のように 1 つ添字を取ります。★注意: シリーズ変数は未実装です。★
- シリーズ定数 — シリーズ変数とまったく同じですが、計算によって値を求める代わりに初期値として固定的に値が設定されています。たとえば経済成長による需要増などは計算式でモデル化するよりデータとして与えてしまった方が簡単です。
- コントローラ入力変数 — これもシリーズ定数と同じですが、コントローラがゲームの進行に応じて値を入力します。★注意: コントローラ入力変数は未実装です。★
- 広域定数 — ラウンドごとの変化もないような単一の定数値を表します。たとえば「製品最低価格」などはゲーム中には変化させないでしょうから、広域定数で表すのが普通でしょう。

これらの変数群に基づいて、各ラウンドでの計算は次のようにモデル化できます。

ラウンド番号 R の計算とは、$R-1$ ラウンドまでのすべての変数値、および R ラウンドの入力変数値をもとに、R ラウンドのチーム変数とシリーズ変数の値を定めることである。

実際の計算内容はこれを複数の計算式で書き表して行きます。その際、「現在のラウンド」の変数をもっとも多く参照するので、単に変数名を書くと現在のラウンドを表すこととし、1 つ前、2 つ前、…のラウンドの値を参照したければ「変数名@番号」のように書きます。たとえば次のようにするわけです。

```
tlet 利息額 = 預金残高@1 * 標準利率;
tlet 預金残高 = 預金残高@1 + 利息額 + 入金額;
```

tlet はチームごとに個別に値を計算する命令で、モデルの計算の大部分は tlet を使うことになるでしょう。ここで「標準利率」は広域定数、「入金額」はチーム入力変数、他はすべてチーム変数になるのが普通でしょうが、それらの情報は別のところで指定するのでここには書きません。

4 ゲーム記述の実例

ごたくばかり並べてもつまらないので、とりあえず非常に簡単なゲームの記述ファイルを見て頂きましょう。箇所ごとに区切って説明します。

```
# Case: Sample1
# ・商品の値付けのみを入力。
# ・価格が安いほどよく売れる。
# ・商品は注文があったらあっただけ売れる。
# ・売れた分だけ補充する。仕入れ価格は一定。
```

「#」で始まる行はすべてコメント (注釈) です。何のゲームか分かるようにコメントはきちんと書きましょう。

```
#
# ゲームの規模
#
def game-name Price Only
def max-team 3
def max-round 10
password ctr t1 t2 t3
```

ゲーム記述ファイルは、行ごとに「命令 指定 …」という形をしていて、命令ごとに機能が違っています。命令 def はゲームの主要なパラメタを設定するものです。具体的には、game-name(ゲームの名前)、max-team(チーム数)、max-round(最大ラウンド数) を必ず指定する必要があります。

password 命令はその名の通りパスワードを設定するもので、最初の要素がコントローラパスワード、その後がチーム 1、2、3、…のパスワードになります。

```
#
# シリーズ定数
#
scon 商品需要 497 1195 2447 4037 5406 6626 8177 9451 9945 10713
```

scon 命令はシリーズ定数を定義します。ここでは「商品需要」というのがシリーズ定数名で、その後の数値が 1 期、2 期、…に対応する定数値です。

```
#
# 広域定数
#
gcon 仕入価格   90
gcon 最低有効価格 50
```

gcon 命令は広域定数を定義します。scon と似ていますが、広域定数なので数値は 1 つだけ指定します。

```
#
# 入力変数と入力ページ
#
ipage price 価格の入力
  <H1>価格入力</H1>
  <P>販売価格を入力してください。</P>
ivar 販売価格 range 0 1000 120
```

ipage 命令は 1 つ現われるごとに入力ページが 1 つ作られます。そのパラメタとしては、ページの英字名、日本語名を指定します。その後 2 行はこの命令の「継続行」で、一般に先頭が空白で始まる行のはすべてその前にある命令の継続行になります。ipage 命令の場合は、継続行は HTML コードとしてそのページの先頭にそのまま差し込まれますから、入力変数に関する説明を記してください。

続いて ivar 命令で入力変数を指定します。複数の入力変数がある場合は ivar 命令も複数必要です。ここでは range 指定で数値を自由に打ち込む入力欄を持つ入力変数を 1 つだけ作っています。他に select 指定も可能です (マニュアル参照)。

```
#
# チーム毎モデル変数と初期値
#
tvar 販売数
tvar 売上金額
tvar 仕入金額
tvar 差引損益
tvar 現金貯金    150000
```

tvar 命令はチーム変数を定義します。初期値が最大 5 個まで書け、書いた順に 0 期、−1 期、−2 期、…のその変数の値になります (1 期以降についてはチーム変数の値は計算で求めるので、初期値を与えることはできません)。

```
#
# 計算モデル
#
pinv 販売数 = 商品需要 by 販売価格-最低有効価格;
tlet 販売数 = rint(販売数);
tlet 売上金額 = 販売数 * 販売価格;
tlet 仕入金額 = 販売数 * 仕入価格;
tlet 差引損益 = 売上金額 - 仕入金額;
tlet 現金貯金 = 現金貯金@1 + 差引損益;
```

tlet 命令は、指定した計算をチームごとに個別に実行することを意味します。ここで「@1」「@2」…は「1 期前」「2 期前」…を意味しています。継続行は何行でも可能で、実は任意の C の構文が使えます (単にそのまま C のコードとして埋め込まれるだけ)。

pinv 命令は特別な計算をするもので、pinv X = Y by Z; という形をしていて、チーム変数 X は、値 Y を Z の逆数に従って比例配分する、という意味です。この場合は商品需要を「$frac1$販売価格 – 最低有効価格に比例させて各チームに配分し、それを各チームの販売数とするわけです。逆数を取らずに比例配分する prop 命令もあります (あとの書き方は同じ)。

注意! 入力変数に対して値を設定した場合は、変更された値はそのラウンドの間だけは有効だが、次のラウンドにはまた元の値 (入力値) に戻ってしまう。これは混乱の元になるので、入力変数に値を設定するのは避けた方がよい。

```
#
# 出力指定
#
ooption 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 販売数 売上金額 仕入金額 差引損益
endtable
```

ooption 命令は出力に関するオプションを設定します。fmt オプションは数値出力用のフォーマット文字列を指定するもので、「%1.0lf」は少数点以下四捨五入して最少の文字数出力を意味します。標準値は「%5.3lf」で、これは少数点以下 3 桁常に表示した上で最少の文字数出力です。

opage 命令は 1 つごとに出力ページを 1 つ定義します。英字名、日本語名は ipage と同じですが、その後に public(誰にでも公開)、teamspec(自チームの状況のみ見える)、control(コントローラのみ見える) のいずれかを指定します。その後に継続行で HTML を記述できるのは ipage と同じです。

出力は表を多用するので、表の区切りを表すため個々の表は begintable、endtable で囲みます。endtable には継続行として表の後に記したい HTML コードを書くことができます。

out 命令は begintable〜endtable の中でのみ使用可能で、各種のデータ出力を 2 番目のパラメタで指定します。teams は、単にチーム番号を順次出力するもので、見出し用です。teams-vars は、その後に指定した変数の値を 1 行ずつ、各行についてチーム横断で出力します。

```
#
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 ’収入 ’支出
out values ’前期繰越 現金貯金@1 -
out values ’収入金額 売上金額 -
out values ’支出金額 - 仕入金額
```

```
out values ’差引損益 - 差引損益
out values ’今期繰越 - 現金貯金
endtable
```

values はもっとも一般的な出力指定で、表の 1 つの行を指定します。その後に各セルに入る内容を指定しますが、「’」で始まる文字列はそのまま見出しとして表示され、「-」はそのセルが空欄のまま残され、それ以外であればその変数値が埋め込まれます。

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

この部分はコントローラ用に全変数を表示するためのもので、rounds はラウンド横断の見出行し出力、teams-allvars、teams-allrounds はそれぞれ全変数をチーム横断、ラウンド横断で表示します。

5 gg2 の使い方

gg2 を使ってゲームを生成する手順を説明しておきます。

1. 「cd WWW」で自分の WWW ディレクトリへ行く。
2. ゲームのディレクトリ名を決めて作る。ここでは仮に「gamex」だとすると、「mkdir gamex」を実行する。
3. 「cd gamex」でそのディレクトリへ行く。
4. ゲーム記述ファイルをそのディレクトリに置く。名前は何でもよいが、「gamex.gg」のように最後を「.gg」としておくこと。
5. 「gg2 gamex.gg」により、記述ファイルを処理し、ゲームを生成する。
6. 何らかのエラーがあれば記述ファイルを手直しして 5 からやり直す。
7. すべて OK であれば、Netscape を起動して

```
http://smm-atm/~user/gamex/     --- E428、G431 の場合
http://smm/~user/gamex/         --- その他
```

を開くと、ゲームの入口画面になっているはず (なお「~user」はもちろん、自分のユーザ名で適宜置き換えること)。

なお、tlet などの記述内容はすべて変数部分のみ変換した上で C 言語に埋め込みますから、構文誤りがあれば C コンパイラからエラーメッセージが出ます。その場合は compute.c、display.c などの C ソースファイルを見てエラー内容を調べ、元の変換記述ファイルを修正する必要があります。

ただし、C ソースでは変数名が漢字を含まないものに機械的に変換されているので、そのままでは何が何だか分からないかも知れません。その場合は、次のように ggmore コマンドを使って変換すれば元の漢字まじりの変数名として読めるようになります。

```
ggmore compute.c | less   --- 単に見る場合
ggmore compute.c | cat -n | less  --- 行番号つきで見る場合
```

よくある間違いとしては次のものがあります。

- 変数名の打ち間違い (gg2 でもエラーメッセージが出ているはずだが、見落としている?)。
- 文末の「;」が抜けている (C の代入文は必ず最後に「;」が必要です。
- かっこの不対応。

6 gg2 記述ファイル参照マニュアル

6.1 def 命令

ゲーム構成を規定する値を設定する。

```
def game-name ゲームの名称
def max-team チーム数
def max-round 最大ラウンド数
```

これらの値は必ず設定されていなければならない。

6.2 password 命令

パスワードを設定する。

```
password コントローラパスワード チームパスワード…
```

命令の次の語がコントローラパスワード、その後の語がチーム 1、2、…のパスワードになる。

6.3 scon 命令

シリーズ定数を定義する。

scon 変数名 $値_1$ $値_2$ …

シリーズ定数名を定義するとともに、その一連の値を設定する。順に第 1 ラウンド、第 2 ラウンド、…の値となる。

6.4 gcon 命令

広域定数を定義する。

```
gcon 変数名 値
```

広域定数を定義するとともに、その値を設定する。

6.5 ipage 命令

1 つの入力ページの開始を表す

```
ipage 英語名 日本語名
   HTML 記述
```

英語名は ASCII の英数字のみから成る名前で、ファイル名に使われる。日本語名は入口ページからのリンクに使われる。HTML 記述は何行に渡ってもよく、ページの先頭部分に差し込まれる (入力するものに関する説明等を記述することをおもに想定している)。

6.6 ivar 命令

入力変数と入力種別を定義する

```
ivar 変数名 range 最小 最大 [デフォルト]
ivar 変数名 select デフォルト 数値 名前 数値 名前 …
```

いずれも入力変数を定義する点は同じだが、入力のされ方が違う。

range では自由に文字を打ち込める入力欄が現われ、そこにユーザが数値を打ち込む。値は最小と最大の範囲でなければならない。デフォルトを書くとその値が入力欄に初期値として表示され、またエージェント機能 (後述) を使った時の最終デフォルト (他に指定がまったく無かった時使われる値) にもなる。デフォルトを指定しないと「最小」の値がデフォルトになる。★注記: 入力値が最大と最小の範囲内であるかどうかチェックする機能は未実装である★

select では画面に複数選択メニューが現われ、一連の名前のうちから 1 つが選べるようになる。ある名前を選んだ場合、その直前に書かれた数値が入力される。デフォルトに指定した数値に等しい名前が初期値となり、またこのデフォルト値がエージェント機能の最終デフォルトになる。

6.7 tvar 命令

チーム変数を定義する。

```
tvar 変数名 [初期値 …]
```

初期値を指定した場合は、それはラウンド 0、ラウンド −1、ラウンド −2、…におけるその変数の値になる。ラウンド 1 以降は計算によって求めるので初期値は指定できない。

6.8 ipre 命令

入力変数に初期値を設定する。

```
ipre 変数名 [初期値 …]
```

ivar 命令には初期値を設定する機能がないので、初期値を設定したい場合はこの命令を使う。変数は入力変数でなければならない。指定した値は順にラウンド 0、ラウンド −1、ラウンド −2、…におけるその変数の値になる。

6.9 tlet 命令

チーム個別に変数計算を行う。

tlet 計算動作

指定した「計算動作」はすべて C 言語の文として C 言語に埋め込まれて実行される。ただしチーム個別なので

for(t = 1; t < MAXT; ++t) { 計算動作 }

という形のループになり、各チームごとの値が計算できる。継続行を書いた場合はそれらもひとまとまりで埋め込まれる。

動作内には任意のゲーム変数を書くことができ、これらの変数名はすべて C の変数名 (漢字を含まない) に変換された形で埋め込まれる。また、変数名の後に「@1」「@2」…をつけることで「1 期前」「2 期前」…の値を参照できる。C の文なので、if 文等を書くことは自由に行える。また数学関数を呼ぶことも自由である。典型的には次のものがある。

sin(x) --- sin。
cos(x) --- cos。
log(x) --- 自然対数。
log10(x) --- 10 の対数。
exp(x) --- e^x。
pow(x, y) --- x^y。
sqrt(x) --- $\sqrt{x}$。
rint(x) --- 整数への丸め。

なお、すべての計算は倍精度実数で行われる。rint(x) も、丸めは行うが結果は丸めた値の実数である。

6.10 prop 命令

各チームへの比例配分を計算する。

prop チーム変数名 = 式$_1$ by 式$_2$;

式$_2$ の値をチームごとに計算し、その値に比例して 式$_1$ を配分して結果をチーム変数に入れる。たとえば 式$_2$ の値がすべてのチームにおいて等しければすべてのチームのチーム変数は $\frac{\text{式}_1}{T}$ になる (T はチームの数とする)。式$_1$ や 式$_2$ は複雑な式でもよいが、途中に空白を入れてはならない。

6.11 pinv 命令

各チームへの比例配分を計算する。

prop チーム変数名 = 式$_1$ by 式$_2$;

式$_2$ の値の逆数をチームごとに計算し、その値に比例して 式$_1$ を配分して結果をチーム変数に入れる。式$_2$ が 0 以下のチームには配分されない。式$_1$ や 式$_2$ は複雑な式でもよいが、途中に空白を入れてはならない。

6.12 ooption 命令

出力に関する各種指定を行う。★現在は設定項目は fmt のみ★

```
ooption fmt 書式文字列
```

変数を表示する際の書式を C 言語の printf の倍精度実数型用の書式指定として指定する。たとえば次のような指定ができる。

```
%5.3lf  --- 最小 5 桁、少数点以下 3 桁。3.141、0.234 など。
%1.0lf  --- 最小 1 桁、少数点以下は丸める。3、0 など。
```

fmt を設定するとその後にある out 命令から書式が変更される。途中で何回変更してもよい。

6.13 opage 命令

出力ページの開始を指定する。

```
opage 英語名 日本語名 [public|teamspec|control]
   HTML 記述
```

英語名は ASCII の英数字のみから成る名前で、ファイル名に使われる。日本語名は入口ページ等からのリンクに使われる。3 番目の引数はページのアクセス制限で、次の意味を持つ。

- public — このページのデータは全チームが自由に参照できる。
- teamspec — このページのデータは各チームが自チームの分だけを参照できる (チームパスワードが必要)。
- control — このページのデータはコントローラのみが参照できる (コントローラページからのみアクセス可能で、なおかつコントローラパスワードが必要)。

HTML 記述は何行に渡ってもよく、ページの先頭部分に差し込まれる (出力するものに関する説明等を記述することをおもに想定している)。

HTML 記述の中に${変数名}という形のものがあった場合、その部分は変数の対応する値で置き換えられて表示される。変数としてはすべてのゲーム変数に加えて「ラウンド」「チーム」という特別な変数名も使え、表示時に指定したラウンド番号やチーム番号がそこに入る。

6.14 begintable 命令

出力ページ中の表の開始を指定する。

```
begintable
```

出力ページ中のデータは表の形で表されるものが多い。この命令から endtable 命令までの間が 1 つの表になることを指定する。

6.15 endtable 命令

出力ページ中の表の終了を指定する。

```
endtable
   HTML 記述
```

begintable の項で説明したように、表の範囲の終わりを示す。この後に何行でも HTML 記述を置くことができ、これらは表の後に出力される。この部分でも opage 命令と同様の変数の埋め込みが行える。

6.16 out 命令

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

```
out teams
out rounds
out values 出力指定 …
out teams-vars 出力指定 …
out vars-teams 出力指定 …
out rounds-vars 出力指定 …
out vars-rounds 出力指定 …
out teams-allvars
out rounds-allvars
```

teams、rounds はそれぞれチーム番号、ラウンド番号を横一列に出力する。見出し用である。

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 等と同様の任意の計算式が記述でき、その値が表示される。普通は単に「変数名」「変数名@ 番号」を書くことが多いと思われるが、もっと複雑な式を書いてもよい。

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。書式の指定方法は ooption fmt での指定と同じである。「見出し:」を省略した場合、計算式がそのまま見出しとして現われる。「:書式」を省略した場合、ooption fmt で指定した書式が使われる。

vars-teams は teams-vars と同様だが、ただし横方向に変数が並び、縦方向にチーム 1〜チーム N のデータが並ぶ。

rounds-vars は teams-vars と同様だが、ただしチーム横断でなく特定チームについてのラウンド横断となる。

vars-rounds は rounds-vars と同様だが、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。

teams-allvars は teams-vars と同様だが、ただしすべての変数を記述ファイルに現われた順に指定したものとみなす。

rounds-allvars は rounds-vars と同様だが、ただしすべての変数を記述ファイルに現われた順に指定したものとみなす。

7 エージェント機能

「エージェント機能」とは、あるラウンドにおいてデータがまだ提出されていないチームの変数値を自動的に計算して適当な値に設定してしまう機能を言います。エージェントによって値が設定されてしまうと、そのチームはもはやそのデータを提出することはできなくなります (「既にデータがある」と言われて拒否される)。

現在の gg2 では、エージェント機能は「どのチームのどのラウンドのどの入力値をいくつに設定する」という単純なものに限られています。これを指定するために、ゲームのディレクトリに AGENT という名前のファイルを作成し (gg2 がからっぽのファイルを用意してくれているはずです)、そこに次のような内容を書きます。

```
r01:t01:変数名=値
all:t01:変数名=値
r01:all:変数名=値
all:all:変数名=値
```

これらはそれぞれ、特定ラウンド特定チームに対する指定、特定チームに対する全ラウンドにおける指定、特定ラウンドにおける全チームに対する指定、全ラウンドにおける全チームに対する指定になっています。

1 つのチーム、変数、ラウンドにおいてあてはまるものが複数ある場合は、次の優先順位に従って値が決まります。

1. 特定チーム特定ラウンド
2. 全チーム特定ラウンド
3. 特定チーム全ラウンド
4. 全チーム全ラウンド
5. ゲーム記述ファイルに記されたデフォルト値

なお、AGENT ファイルの各行は先頭に空白があってはいけません。また、その漢字コードは EUC である必要があります。最初に打ち込んだ後で

```
toeuc AGENT
```

というコマンドを実行すれば EUC に変換されます。いちど EUC になれば編集してもそのままのはずですが、Mule で編集するときに「[あ]」の隣に「E」と表示されていることを確認し、「J」や「S」が表示されているようなら再度 toeuc で変換してください。

A 生成ファイル一覧

生成システムが生成するファイル群とその役割についてまとめておきます。

- index.html — ゲームの入口となる Web ページ。
- ig-○○.cgi — 各入力ページの内容を生成する CGI スクリプト。
- ir-○○.cgi — 各入力ページのデータを受け取り、値をチェックした上でデータファイルに書き出す CGI スクリプト。
- control.html — コントローラの制御フォームの Web ページ。
- control.cgi — コントローラの各種操作を実行するための CGI スクリプト。
- Makefile — C 言語のコードをコンパイルするための指示ファイル。
- variables.c — C 言語の変数定義と読み書き用関数のコード。
- compute.c — モデルの計算を実行する C 言語コード。
- compute — compute.c をコンパイルした実行形式。

- display.c — 変数の表示出力を行う C 言語コード。
- display — display.c をコンパイルした実行形式。
- display.cgi — 入口ページの表示ボタンを扱う CGI スクリプト。
- PASSWORD — 各チーム、コントローラのパスワードを格納したファイル。
- ROUND — 現在のラウンド番号を格納したファイル。
- DATA — ゲームのデータファイル群を格納するディレクトリ。
- DATA/iv-○○-RRtTT.dat — 入力ページ○○、ラウンド番号 RR、チーム番号 TT に対応する入力変数データファイル。
- DATA/tv-RRtTT.dat — ラウンド番号 RR、チーム番号 TT に対応するチーム毎変数データファイル。

B sample1.ggの全リスト

```
# Case: Sample1
# ・商品の値付けのみを入力。
# ・価格が安いほどよく売れる。
# ・商品は注文があったらあっただけ売れる。
# ・売れた分だけ補充する。仕入れ価格は一定。
#
# ゲームの規模
#
def game-name Price Only
def max-team 3
def max-round 10
password ctr t1 t2 t3
#
# シリーズ定数
#
scon 商品需要 497 1195 2447 4037 5406 6626 8177 9451 9945 10713
#
# 広域定数
#
gcon 仕入価格  90
gcon 最低有効価格 50
#
# 入力変数と入力ページ
#
ipage price 価格の入力
  <H1>価格入力</H1>
  <P>販売価格を入力してください。</P>
ivar 販売価格 range 0 1000 120
#
# チーム毎モデル変数と初期値
#
```

```
tvar 販売数
tvar 売上金額
tvar 仕入金額
tvar 差引損益
tvar 現金貯金    150000
#
# 計算モデル
#
pinv 販売数 = 商品需要 by 販売価格-最低有効価格;
tlet 販売数 = rint(販売数);
tlet 売上金額 = 販売数 * 販売価格;
tlet 仕入金額 = 販売数 * 仕入価格;
tlet 差引損益 = 売上金額 - 仕入金額;
tlet 現金貯金 = 現金貯金@1 + 差引損益;
#
# 出力指定
#
opage sales 販売状況
  <H1>販売状況</H1>
  <P>第${ラウンド}期: 需要: ${商品需要}</P>
begintable
out teams
out teams-vars 販売価格 販売数 売上金額 仕入金額 差引損益
endtable
#
opage balance 収支の状況
  <H1>収支の状況</H1>
  <P>第${ラウンド}期、チーム: ${チーム}、総需要: ${商品需要}</P>
begintable
out values ’項目 ’収入 ’支出
out values ’前期繰越 現金貯金@1 -
out values ’収入金額 売上金額 -
out values ’支出金額 - 仕入金額
out values ’差引損益 - 差引損益
out values ’今期繰越 - 現金貯金
endtable
#
opage allvteam 全変数チーム横断
  <H1>第${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
  <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
```