37. 汎用マークシート処理システム

慶応大学　近藤頌子・原田賢一
大野義夫・山本喜一

はじめに

慶応義塾大学では昭和33年以来，入学試験の採点や集計に何らかの形で計算機を利用してきた。ここではその現状を，特にマークシートによる答案用紙の採点の部分を中心に御紹介したい。

このシステムは，履習申告，健康診断や，各種アンケートの集計なども行なえるような汎用性をもつものとして設計されている。

1．マークシートの構成

このシステムでは，図1のようなマークシートを取り扱う。

マークシート上で1つのグループとして扱うと便利なマークポジションの集まりを「欄」と呼ぶ。欄は F_1, F_2, ‥‥ という名前で識別する。受験番号マーク欄も各桁別に5個の欄として扱う。このシステムで扱うマークシートは，このようなキーとなる5桁の番号を持っていなければならない。

1つの欄の中の各マークポジションには名前をつける。名前は英字1字または数字1字とする。

2．システムの構成

このマークシート処理システムの構成を図2に示す。各ルーチンの主な機能は次のとおり。

◇欄定義　マークシート上に欄を定義する。
◇名前定義　各欄に属するすべてのマークポジションに名前を与えるとともに，各欄の最大許容マーク数を設定する。
◇採点手続き定義　「各欄のどのマークポジションにどうマークされていたら何点与えるか」を決める採点手続きを定義する。
◇欠席者リスト作成　受験生の受験番号の範囲および試験当日の欠席者のリストを読みこむ。
◇変　換　答案シートを読んで，順序のチェック，マーク数のチェックなどを行なって，チェックリストを印刷する。また，答案シートのイメージを以後の処理に適した形に変換する。

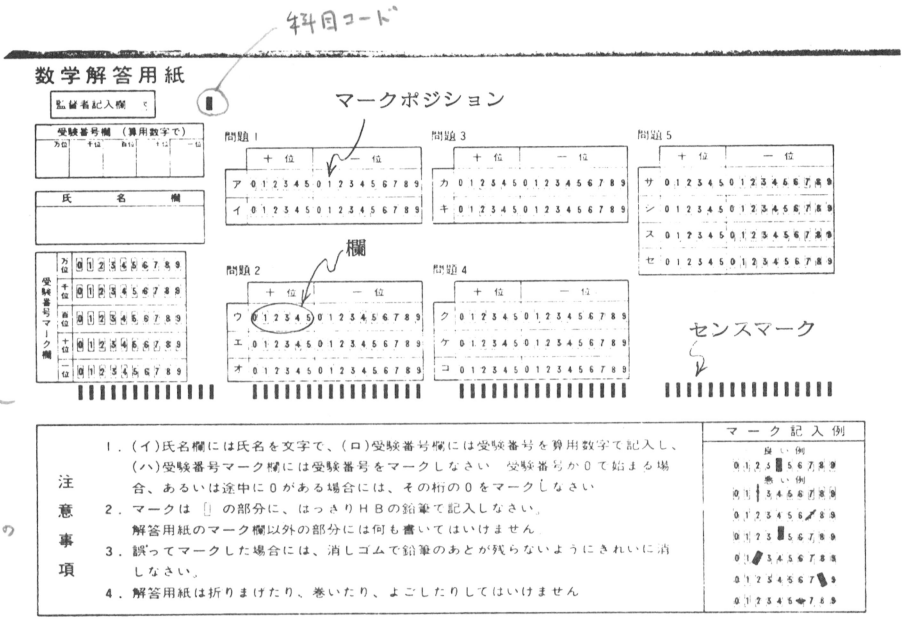

図1　マークシートの構成

◇修　正　　チェックリストにもとづき，必要に応じて答案ファイルの修正を行なう。

◇採　点　　採点手続きによって採点を行ない，問題ごとの得点を算出する。

こうして作られた採点ファイルが入力ルーチンに与えられる。採点までを人手で行なった場合には，受験生ごとに各問題の得点をカードにパンチし，これを入力ルーチンに与える。したがって，以後の処理は，マークシートを使う場合も使わない場合も同じである。

上記のうち，欄定義，欠席者リスト作成の各ルーチン以外は，すべて TSS 端末から対話形式で実行する。

以下の各節では，これらのルーチンのうち，マークシート処理として特徴的な，欄定義(3)，名前定義(4)，採点手続き定義(5)，変換(6) の各ルーチンについて述べる。

3．欄定義ルーチン

マークシート上に欄を定義するためのルーチンである。n 個の欄 F_1, F_2, ‥‥, F_n を定義したい場合には，n 枚のマークシートを用意し，i 枚目のシートには，欄 F_i を構成するマークポジション全部にマークをしておく。この n 枚のシートをオフライン OMR によって磁気テープに変換し，欄定義ルーチンに与えると，欄定義ファイルに次の情報が書きこまれる。

◇欄の個数とセンスマークの個数
◇各欄に属するマークポジションの個数とその位置

最後に，確認用の図がラインプリンタに出される。

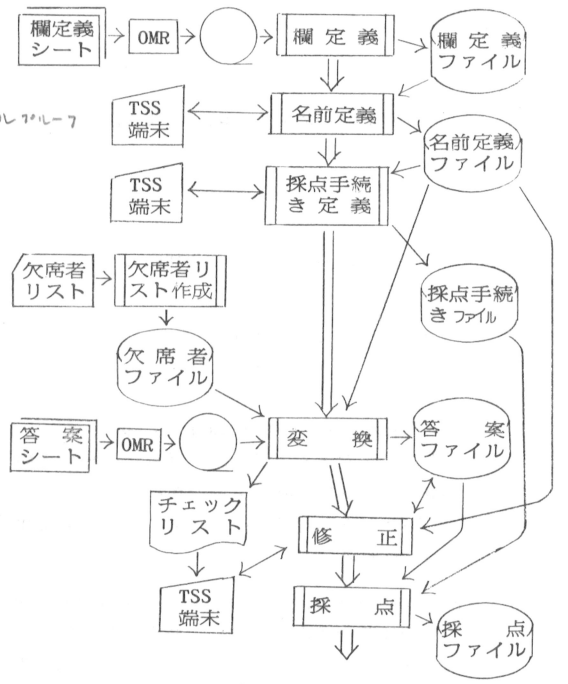

図2　マークシート処理システムの構成

4．名前定義ルーチン

TSS 端末を介した対話によって，各欄に属するすべてのマークポジションに名前を与え，受験番号を構成する欄を指定するとともに，各欄の許容マーク数を設定する。マークポジションの名前は，英字1字または数字1字とする。

処理が完了すると，欄定義ファイルの内容に次の情報をつけ加えたものを名前定義ファイルに書きこむ。

◇各欄の許容マーク数
◇各マークポジションの名前
◇受験番号を構成する5個の欄の名前

また，確認用として，各マークポジションの位置に名前を書きこんだ図を TSS 端末に出力する。

5．採点手続き定義ルーチン

マークシート上の欄や各欄のマークポジションに対して定義した名前を用いて，採点をするための手続きを TSS 端末から入力するためのルーチンである。採点手続きの定義に関する作業のステップは以下のとおりである。

ステップ1．採点手続きの入力
まず，1つの科目について，その問題がどのような副問題から構成されているかを定義する。次に，その各問題に対してどのようにして得点を計算するのか（採点手続き）を入力する。

ステップ2．修　正
ステップ1で作成した採点手続きに誤りがあったときには，専用の編集プログラムを用いて修正を行なう。

ステップ3．翻　訳
採点手続きに誤りがなかったときには，翻訳プログラムを呼んで，採点手続きを実行可能なプログラム（リロケータブル・プログラム）に変換する。

5.1　問題の構造の定義

通常，1つの科目はいくつかの問題からなり，その各問題はさらにいくつかの問題からなっている。そこで問題のレベルを考える。ある問題がいくつかの細かい問題を含んでいるとき，前者を後者の上位レベルの問題といい，後者を前者の下位レベルの問題という。下位レベルの問題を含まないものを基本問題という。

最上位の問題を大問題，それよりも直接的に下位レベルの問題を中問題，さらに下位レベルの問題を小問題と

いう。採点手続き定義ルーチンでは，このように最高3レベルまでの構造をもった問題が取り扱えるようになっている。

問題の構造の例：

```
［1］
    (1)
        〈1〉  ……
        〈2〉  ……
    (2)   ……
［2］
    (1)   ……
    (2)   ……
    (3)   ……
```

各レベルにおいて問題の番号は1から始まるものとする。各レベルの問題の番号は次のような記号によって表示される。

［n］： 大問題 n
(m) ： 中問題 m
〈k〉： 小問題 k

問題の構造の定義の例を図3に示す。

5.2 採点手続きの記述

次に，各問題に対する得点の計算方法を考え，以下に述べる記法を使ってそれを表現する。いま，ある基本問題に対する採点方法について考えることにする。説明のために以下の記号を用いる。

F_i　　マークシート上の i 番目の欄
mark　マークポジション。'$C_1C_2\cdots C_n$' と書くことがある。C_i は英字または数字である。
p　　得点。整数型の式で表わす。

(1) もっとも簡単なものは，ある欄の特定の位置だけにマークがあったときに何点，そうでなかったら0点という形式の採点方法である。このときには次のように書く。

F_i=mark ： p ；

(2) 上の形式の組み合わせで，いくつかの条件を全部満たしていたときだけ点を与えるようにするには次のように書く。

F_i=$mark_1$ & F_j=$mark_2$ & ‥‥
& F_n=$mark_m$ ： p ；

(3) いくつかの条件の中の少なくとも1つを満たしていれば得点を与え，どれも満たしていなければ0点とするときには次のように書く。

F_i=$mark_1$! F_j=$mark_2$! ‥‥
! F_n=$mark_m$ ： p ；

コロン（：）の左側の部分は通常のプログラミング言語にみられる論理式と同じであり，上述の説明を一般化すると次のようになる。

◇F_i=mark は論理式である。
◇e_1, e_2, ‥‥ を論理式としたとき，次のものも論理式である。

e_1 & e_2 & ‥‥ & e_n　　(& は and)
e_1 ! e_2 ! ‥‥ ! e_n　　(! は or)
△ e_1　　(△ は not)

◇F_i の mark のところにマークがなければ真，マークがあるとき偽とするには

F_i △= mark　　(not equal)

と書く。
◇ここに述べた採点方法の記法は

〈論理式〉 ： p ；

という形で表わされる。

条件 e_1 のときは p_1 点，条件 e_2 のときは p_2 点，‥‥ という採点方法は次のように表現する。

e_1 ： p_1 , e_2 ： p_2 , ‥‥ , e_n ： p_n
[else p_{n+1}] ；

この場合，もっとも左側の条件からテストを始め，最初に満足した条件に対応した式の値が得点となる。すべての条件が満足されなかったとき，else の指定があれば p_{n+1} が得点となる。また，else の指定がなければ0点となる。

特定の形式をした採点方法については，以下に示すような簡便な入力形式が用意してある。

(1) F_i=mark ： p ；
は
i $C_1C_2\cdots C_n$ p
と書くだけでよい。ただし，mark='$C_1C_2\cdots C_n$'。

(2) F_i=$mark_1$! F_i=$mark_2$! ‥‥
! F_i=$mark_n$
は
F_i= [$mark_1$, $mark_2$, ‥‥, $mark_n$]
と表わすことができる。

(3) F_i=$mark_0$ & F_{i+1}=$mark_1$ & ‥‥
& F_{i+n}=$mark_n$ ： p ；
は
i —> $mark_0$ $mark_1$ ‥‥ $mark_n$ ： p
と表わすことができる。これは，番号が連続している欄を調べ，すべての欄のマークが指定したものと一致していたときにだけ得点を与えることを表現するのに用いられる。

基本問題に対する得点は多くの場合，上述の表現方法のいずれかで記述できると思われる。さらに複雑な採点方法をとるときには，これから述べるような文を用いて記述しなければならない。基本問題よりも上位レベルの問題の得点は，特別な指定がない限り，次のようにして

計算されるものと仮定している。

◇科目全体の得点は大問題の得点の合計
◇大問題の得点はその中の中問題の得点の合計
◇中問題の得点はその中の小問題の得点の合計

これ以外の方法で得点を求めたかったら，下位レベルの問題に対する採点手続きを入力し終ったあとで，得点を操作するための文を入力すればよい（図3）。

5.3 採点手続きを記述するための文

前節ではもっとも簡単な文を説明したが，実際には以下に述べるように，代入文，if 文，および case 文が用意されている。上に示したものは if 文の一種であり，それらは，内部的には，採点手続き定義ルーチンの入力処理ルーチンによって，if 文に変換されている。

（1）代入文

ある計算結果を1つの変数に代入するための文で，次のような形をしている。

〈変数〉：＝〈式〉

式には整数型のものと文字列型のものとがある。文字列型の値はいくつかの英字または数字の列から構成されていて，主としてマークを照合するのに用いられる。変数として使用できるものはあらかじめ定められている。等号の左辺に書くことができる変数には次の3種類がある。

◇得点変数　Pi, Pi.j, Pi.j.k
　i, j, k はそれぞれ整定数である。問題 i, i.j, または i.j.k の得点を表わすのに用いる。1枚の答案用紙を採点するとき，各得点変数には初期値として0が代入される。
◇A_i　ただし i＝0～70。
　作業用の整数型の変数で，中間結果を保持するのに用いる。
◇S_i　ただし i＝0～70。
　作業用の文字列型の変数である。

もっとも簡単な形の式は，変数や定数だけからなるものである。式を構成する変数は上記のもの以外に，次に示すものが使える。

F_i または VF_i	欄 F_i のマークの値（文字列型）
CF_i	欄 F_i の実際のマーク数（整数型）
PF_i	欄 F_i のマークポジションの個数（整数型）
NF_i	欄 F_i の許容マーク数（整数型）

定数にも整定数と文字列定数とがある。文字列定数には次の2種類がある。

◇'$C_1C_2\cdots C_n$'
　C_1, C_2, ‥‥, C_n は英字または数字。このような定数を基本文字列定数と呼ぶ。
◇S_1, S_2, ‥‥, S_m を基本文字列定数としたとき
　[S_1, S_2, ‥‥, S_m]
と書いて，文字列の集合を表わすのに使う。

算術式は整数型の変数や定数，あるいはそれらを算術演算子（＋，－，＊，／）で結んだものである。

算術式の中で関数 SUM が使える。この関数はそれが用いられた問題に直接含まれる下位レベルの問題の得点を合計した値を求めるものである。

文字列型の式は文字列型の定数や変数，あるいはそれを連結子 !!（concatenation）で結んだものである。

代入文として，右辺の算術式だけを書くことが許される。そのときには，

〈得点変数〉：＝〈算術式〉

として解釈され，その〈算術式〉が書かれた問題に対する得点変数が左辺に生成される。

（2）If 文

If 文の一般形は次のとおりである。

if 〈論理式〉 then 〈文の列$_1$〉
　[else 〈文の列$_2$〉] end

論理式の中では次の2つの関数を使用することができる。

◇SOME(e_1,e_2)　文字列 e_1 が文字列 e_2 の各文字を含んでいれば真，そうでなければ偽。
◇ALL(e_1,e_2)　文字列 e_1 がすべて文字列 e_2 で構成されていれば真，そうでなければ偽。

（3）Case 文

式の値によってそれぞれ異なった処理をするために case 文がある。Case 文は次のように書く。

```
case  e
  of
    \c1\  b1
    \c2\  b2
       :
       :
    \cn\  bn
  [ else  bn+1 ]
  end
```

ここで，e は式，c_1, c_2, ‥‥, c_n は定数，b_1, b_2, ‥‥, b_n は文の列である。式の型と定数 c_i の型は一致していなければならない。この文を実行すると，まず e が評価され，その値が c_i と一致したときには b_i が実行される。もし，一致するものがなく，else の指定があれば，そこで指定された文の列 b_{n+1} が実行される。

5.4 採点手続きの入力と例

問題の構造の定義が正しく終了すると，引き続いて，採点手続きを入力するようにメッセージが出力される。これまでの作業によって問題の構造は明らかになってい

```
   Type in "I", "E", "T", or "END" as an answer to
the next question.  where,
 I means input, i.e. file creation of a rating proc
 E   : Editing of a rating procedure
 T   : Translation of a proc
 END : End of edit and trans. on a particular proc
Input, Edit, Translate, or END? >I
```

問題の構造の定義を指定

問題レベルの最大深さ

```
Now, let's start definition of number of problems!
Type in max depth of the problem: 1, 2, or 3? >3
Are there three levels? >Y
Do all minor problems have sub-problems? >Y
How many major problems do you have? >2
You have 2 major problems.  Don't you? >Y
Ok!  Fine!
Do you need echo check as shown above, hereafter too? >N
I see, Be careful of your typing from now!
How many minor problems are there in Prob.1 ? >2
How many sub-prob's are there in Prob.1.1? >2
How many sub-prob's are there in Prob.1.2? >4
How many minor problems are there in Prob.2 ? >3
How many sub-prob's are there in Prob.2.1? >3
How many sub-prob's are there in Prob.2.2? >5
How many sub-prob's are there in Prob.2.3? >6
Would you like to have the definition list? >Y
```

折返しチェック

大問題の個数

折返しチェックの有無

問1の中問題数

問1.1の小問題数

問2の指定

```
Your definition of problem structure is following:

[1]  (1)  <1>  <2>
     (2)  <1> ... <4>
[2]  (1)  <1> ... <3>
     (2)  <1> ... <5>
     (3)  <1> ... <6>
```

問題の構造

図3　問題の構造の指定例

```
Rating Procedure Definition
 [1]
    (1)  >6 A 2
    (2)  >7 D 3,
         >7 E 4
    (3)  >8 -> 3 5 2 6 : 10,
         >8 -> 3 5 3 6 : 8
         Is the score of Prob. 1 defined to be sum of
      the minor prob. scores? >Y
 [2]
    (1)  >SOME(F12,['2','4']) : 4 ;
    (2)  >PF13=2 & (F13!!F14='74') : 5 ;
         Is the score of Prob. 2 defined to be sum of
      the minor prob. scores? >N
Prob 2 >IF P2.1>P2.2 THEN P2.1 ELSE P2.2 END ;

   Is the total score defined to be sum of scores for Prob.
1 to 2 ? >Y
Your current version of the program has been written on a file.
End of rating procedure input

Would you like to have listing of the input? >Y


List of the Rating Procedure

 [1]
     (1) F6='A' : 2 ;
     (2) F7='D' : 3 ,
         F7='E' : 4 ;
     (3) F8='3' & F9='5' & F10='2' & F11='6' : 10 ,
         F8='3' & F9='5' & F10='3' & F11='6' : 8 ;

 [2]
     (1) SOME(F12,['2','4']) : 4 ;
     (2) PF13=2 & (F13!!F14='74') : 5 ;
     IF P2.1>P2.2 THEN P2.1 ELSE P2.2 END ;
```

問1の得点はその中問題の得点の合計かという問合せ

入力した手続きのリスト

図4　採点手続きの指定例

check にひっかかる回答カードの率
入試のとき 0.5～1%

るので，採点手続き定義ルーチンからは，問題の番号を順に指定して，その問題に対する採点方法を入力するように要求される。1つの問題に対する採点手続きは，文の列にセミコロン（;）が続いたものである。したがって，セミコロンを入力すると，自動的に次の問題へ進むようになっている。最初は基本問題に対する採点手続きの入力が要求される。1つの問題に直接含まれるすべての基本問題についての処理が終ると，このルーチンからは，「この問題の得点はこの各基本問題の得点の合計でよいか」という問合せが出される。もし特別な処理が必要ならば，そこで採点方法を指示するための文の列を入力すればよい。このような入力手順を繰り返し，最後にいま定義している科目全体の得点の計算方法を指示することによって，採点手続きの定義は終了する。

1つの科目に対する採点手続きの入力が終了すると，入力されたテキストは，大問題，中問題，あるいは小問題を表わす記号と問題の番号とが内部的に各文の列につけられ，ファイルに書き出される。したがって，このセッションが終了しても，入力テキストはほぼそのままの形で保存されているので，別の機会に修正を行なうこともできる。

採点手続きの入力時には，チェックはほとんど行なわれない。詳細な構文チェックおよび意味のチェックは，翻訳のステップで行なわれる。

採点手続きの入力の例を図4に示す。

5.5 翻　訳

採点手続き定義ルーチンに対して翻訳の指示が出されると，指定された科目の採点手続きのファイルをさがし出し，次のようなステップによって，リロケータブル・オブジェクト・プログラムを作り出し，それをファイルに書き出す。

（1）採点手続きのテキストをファイルから読みこんで，コア内に中間形を作成する。このとき，構文および意味のチェックを行なう。
（2）採点手続きの記述に誤りがなければ，中間形を走査し，オブジェクト・プログラムとして，SIMPL に書きなおした採点手続きを生成し，ファイルに書く。
（3）SIMPL コンパイラを呼び出して，そのプログラムをコンパイルする。
（4）こうして作り出されたリロケータブル・プログラムを採点手続きファイルに複写する。

採点手続きを SIMPL に変換した理由は，採点処理では文字列の処理が大部分であり，SIMPL がそれに適していたからである。

6．変換ルーチン

受験生の答案シートをもとにオフライン OMR で作成した磁気テープを読みこんで，各種のチェックを行なうとともに，シートのイメージを以後の処理に適した形に変換する。

答案シートを磁気テープに変換する際，フィードミスなどにより，センスマークの個数のチェックにひっかかって正しく読まれなかったシートは，その時点で OMR が一度停止するので，別にまとめておき，その磁気テープの最後で読みなおしを行なう。受験生の数が多いので，何本かの磁気テープに分割してこの作業を行なう。

変換ルーチンは，この磁気テープを読んで，次のようなチェックを行なう。

◇センスマークの個数
◇受験番号欄が正しくマークされていること（マーク忘れ，多重マークのチェック）
◇シートの順序（受験番号の小さい順に並べられていること）
◇受験番号が正規の範囲にあること
◇欠席者リストに載っている受験番号のシートには欠席マークがついていること および この逆
◇各欄のマーク数が許容マーク数をこえていないこと

これらのチェックにひっかかったシートは，該当する部分にエラーフラグをつけて答案ファイルに書きこむ。また，受験番号の順序が正しくなかったり，受験番号欄が正しくマークされていない場合には，前後関係から判断して，正しい受験番号が推定できる場合には，その番号に書き換える。推定ができないシートは，答案ファイル中の特別な領域に書きこむ。なお，許容マーク数をこえるマークがなされた欄があっても，（許容マーク数 ＋ 1）個のマークまでは読みとっておく。

以上のチェックの結果は，すべてチェックリストとしてラインプリンタに出す。