ALGOL—入出力の手続き*

井　上　謙　蔵**

1. ALGOL の標準化について

今年5月，ニューヨークで行なわれた ISO/TC-97 の会議で，Revised ALGOL 60[1] とそのサブセット (IFIP)[2] を，国際標準プログラム言語とすることが決められた．本来 Revised ALGOL 60 にも，そのサブ・セット (IFIP) にも，入出力の表現方法が全然与えられていないのであるから，入出力の規定をつくらなければ標準語として採用することができない．そこで ACM の委員会から，すでに提案されている入出力の表現方法[3] と，IFIP-WG 2.1 で作成された標準手続きを文法に対する補足として同時に採用することとなった．終局的には若干の改訂が行なわれるようであるが，現在のところわれわれが利用できる文献は3と4のみであるから，これらにもとづいて解説を進めることにする．

のちに ACM と IFIP の入出力手続きの関連を論じるために，まず Revised ALGOL 60 とサブセット (IFIP) の関係を説明しておこう．今後一々 Revised ALGOL 60 や，サブセット (IFIP) と書くのはめんどうであるから，それぞれ，ALGOL，サブセットと略称することにする．特に必要ある場合にのみ正式の名称をあげることにする．また ACM や IFIP で定めた入出力手続きの全体をそれぞれ ACM-I/O, IFIP-I/O と略称することにする．さて，サブセットの意味は，ALGOL の文法で書かれたプログラムを，サブセットのコムパイラーで処理することはできないことはありうるけれども，サブセットの文法で書かれたプログラムはいずれのコムパイラーでも同様に処理できることを意味している．ここで同様に処理できるということは，数値解析的に同じ結果が得られることを意味する．ISO, TC-97 でこれらを標準プログラム言語と定めたのは，個々のプログラマによって作製されるコムパイラーが，Revised ALGOL 60 のサブセットであり，かつ，サブセット (IFIP) をサブセットとするようなものであることを期待しているからであろう．Revised ALGOL 60 の文法の解釈については，完全な一致の得られない若干の点があるが，一応これは，個々のコムパイラー製作者に文法の上限を与え，サブセット (IFIP) のほうは厳密に下限を規定して，国際標準言語としての役割を果たすわけである．ところで ACM-I/O と IFIP-I/O の両者を標準化のために採用したのは，入出力の表現方法に ALGOL とサブセットの関係に対応する関係づけを想定したものと考えられるが，のちに述べるように現在の形のままでは両者はうまく整合していないので，二，三の標準函数の追加または変更が必要と思われる．

* The Procedures for the Input and Output of ALGOL 60, by Kenzo Inoue (the Institute for Solid States Physics, the University of Tokyo)

** 東京大学物性研究所

2. IFIP-I/O

IFIP-I/O については文献を入手しにくいと考えられるので，ていねいな解説を行なうことにしよう．

標準入出力手続きとして，IFIP-I/O で定められたものは，

inreal, outreal; inarray, outarray;

insymbol, outsymobl; length;

の7個である．

inreal と outreal は，1個の実数型の変数に数値を読みこむ場合と，1個の数値を実数型の表現型式で印刷する場合に使われる．今後，読みこむものというのは，入力機器の種類に関係しない入力の機能であり，印刷ということも印刷，穿孔，等々，すべての出力の機能をさすことにする．inreal と outreal の宣言を書きあらわすと，

```
procedure inreal (channel, destination);
  value channel; integer channel; real
  destination; <procedure body>
procedure outreal (channel, source);
  value channel, source; integer channel;
  real source; <procedure body>
```

である．ここで channel はそれぞれ，入力と出力の機器番号をあらわす正の整数である．destination は入力機器をとおして読まれた実数値が割り当てられる変数の型式的パラメタの名前であり，source は印刷されるべき数値を与える arithmetic expression の型式的パラメタ名である．これらの数値の，計算機の

外側での形，たとえば紙テープやカード上の形は，outreal で出されたものが inreal で読みこめるようなものでなければいけない．したがって通常は指数を伴う正規化された 10 進形式となるであろう．もちろん，このような計算機の外側での数の形式が，コムパイラーごとに異なって決められたとしても，プログラム言語のほうは同じだから，IFIP-I/O を国際標準として採用する障害とはならない．

次に

```
procedure inarray (channel, destination);
  value channel; integer channel;
  array destination; <procedure body>
procedure outarray (channel, source);
  value channel; integer channel;
  array source; <procedure body>
```

がある．array の全要素の値を，行の順に読みこみ，また印刷する手続きである．個々の要素の計算機の外側での形式は，inreal, outreal の場合と同じである．

以上4個の標準手続きのみで，数値計算のためのすべての情報の入出力ができるわけであるが，実際には整数形式の入出力や文字の印刷といったようなぜいたくもほしくなる．これらの作業を自由に，かつ ALGOL の文法と矛盾することなく行なわしめるために，残りの3個の標準手続きが用意された．

まず

```
procedure insymbol (channel, string,
  destination); value channel; integer
  channel, destination; string string;
  <procedure body>
```

を例題によって説明する．insymbol を用いて整数読みこみの手続きを作ると

Example 1

```
procedure in integer (channel, integer);
  value channel; integer channel, integer;
  begin integer n, k; Boolean b;
        integer:=0; b:=true;
        for k:=1, k+1 while n=0 do insymbol
              (channel, '0123456789−+', n);
        if n=11 then b:=false else if n≠12
          then integer:=n−1;
        for k:=1, k+1 while n≠13 do
          begin integer:=10×integer+n−1;
              insymbol
                (channel, '0123456789−+',n)
          end 1;
        if ¬ b then integer:=−integer
  end
```

これはたとえば member=354, などというデータを 354 と変数 integer に読みこむ手続きである．問題は insymbol の使い方であるが，この手続きの2番目のパラメタ，string の中にはもちろん ALGOL の basic symbol しか書くことを許されない．そして string 中にあらわれた順序に basic symbol に番号をつける．たとえば

```
insymbol (channel, '0123456789−+',n)
```

では，basic symbol 0,1,2,…,9,−,+の順に 1,2,3,……, 10, 11, 12 と番号づけがされる．insymbol は外部より一つの記号を読んで，それが上記の basic symbol の一つならば，その番号を3番目のパラメタ n に与える．読んだ basic symbol が string 中になければ，0を与える．読んだ記号が ALGOL の basic symbol でなければ，その記号に該当する負の整数を与える．最後の場合は，記号と負の整数との対応は適当な形でプログラムに与えられなければならない．

最後に outsymbol と length の説明をする．

```
procedure outsymbol (channel, string,
  source); vulue channel, source; integer
  channel, source; string string;
  <procedure body>
integer procedure length (string);
  string string; <procedure body>
```

これらを用いて文字の印刷を行なう procedure を作ると

Example 2

```
procedure outstring (channel, string);
  value channel; integer channel; string
  string; begin integer i; for i:=1 step 1 uutil
    length (string)
      do outsymbol (channel, string, i)
  end
```

である．length (string) は，そのパラメタ，string 中の basic symbol の個数を求める整数型函数である．outsymbol は insymbol の逆の手続きで，3番目のパラメタの値で指定された番号の，2番目のパラメタ中の basic symbol を印刷する．もし3番目のパラメタの値が負の整数なら，このプログラムに対して定められた特別な記号が印刷される．このパラメタの値が 0 の場合は，無意味になることに注意された

い.

以上で IFIP-I/O の標準入出力手続きをすべて説明したが，行変えや頁送りの記号または手続きが定められていないのは困ったことである．もちろん上記の標準手続きはすべて，1個の数値の印刷途中に行かえがはいってしまうというようなみっともないことはしないものであるとしても，計算結果の印刷様式に対する要求はプログラムごとにかなり異なるものなので，最低限の標準出力手続きの中にはせめて行かえや頁送りの記号を含むか，標準手続きを与えるべきである．ACM-I/O のように string quote の中が，数の format と title とわけられるようにはなっていないので，行変えや頁送りを，/ や↑であらわすことはできないし，また新しく basic symbol を定めることも問題であろう．outsymbol (channel, '',−1) と，outsymbol (channel, '',−2) が行かえと頁送りをあらわすと定めてしまうのが，最も簡単な方法である．

3. ACM-I/O

ACM-I/O については補足的な説明にとどめたいので，読者は文献を参照していただきたい．

印刷の様式を柔軟に制御するために，ACM-I/O では多数の標準入出力手続きが規定されている．まず名前をあげると

直接情報の入出力のために

in list, input n, out list, output n
(ただし n=0, 1, 2, ……)
in item, out item

入出力の情報の様式を制御するために

format, format n (ただし n=0, 1, 2, ……)
h end, v end, h limit, v limit, no data

入出力の様式制御について当面の状況を知るために

in control, out control,

中間結果の入出力のために

get, put

などが用意される．そのうえ，磁気テープの操作や，誤入力の処置などのための標準函数を作ることも許されている．in item と out item はプログラマが直接利用できる手続きではないが，その説明はのちにゆずることにする．

まず，IFIP-I/O の outreal に対応するものとして output n がある．たとえば整数 I の値 (≥0) 5桁を，空白3個に続いて，最初の4桁は有効数字があらわれるまでは空白 (いわゆる zero suppress) で印刷するとすれば

output 1 (6,'BBBZZZZD',I)

なるステートメントを書けばよい．1番目のパラメタの値6は，入出力機器のプログラム上の番号 (logical number) であって，実際の機器の番号 (physical number) との対応は，プログラムの外でオペレータによってつけられるものとする．実用上からはもちろんのこと，入出力言語の標準化の見地からみて，これも当然な配慮というべきである．上記のステートメントによってたとえば

3 6 0 5 は ␣␣␣␣3 6 0 5
4 7 1 は ␣␣␣␣␣4 7 1

と印刷される．2番目のパラメタである string は，印刷すべき数値の形式を定める format string であるが，もしここに null string を与えれば，

output 1 (6, '',I)

は，I の値を標準形で印刷するものとする．この標準形は IFIP-I/O の inreal, outreal の数値の様式と同じように指数を伴う正規化された 10 進数である．format string の細かい書き方は文献 3) の part A に文法とともに示されているから，ここでは 2, 3 の例題を示すにとどめる．

Example 3

output 2 (6,'3 B,+D.6 D_{10}+3 D,3 B,
−Z.4 DB 4D', X[M],cos(t))

output 2 の2は印刷すべきパラメタが2個 (X[M] と cos(t)) あることを示す．format string 中+は，正負の符号を印刷する，−は正符号を空白で置きかえることを示す．3B は BBB, 4D は DDDD と同じであるから

X[M]=1234, cos(t)=−0.30456

の印刷形式は

␣␣␣+1.234000$_{10}$+003␣␣␣␣−.3045␣6000

となるはずである．このようにして任意個のパラメタを任意の様式で印刷する手続き output n (n=0, 1, 2, ……) を，標準手続きとして使用することができる．

標準手続き output n の個々の機能を総括的に果たす重要な標準手続きとして out list がある．たとえば上記の例題3では

Example 4

```
procedure LAYOUT;
  format ('3 B,+D.6 D10+3 D,3 B,
  −Z.4 DB 4 D');
```

```
procedure LIST (ITEM);
  begin ITEM(X[M]);
  ITEM(cos(t)) end;
```

なる手続きを宣言しておけば，ステートメント

out list (6, LAYOUT, LIST)

は，上記の output 2 と全く同じことを行なう．ここで format は標準函数で，手続き out list が LAYOUT を呼び出したとき，上例では出力機器6に対して format のパラメタに示された format string を与えるものである．ところで手続き LIST は，プログラマによって作られるにもかかわらず，見掛け上実際のパラメタで置きかえられないような型式的パラメタ ITEM を書かなければならない．これはたいへん奇妙な話であるが，実は out list の本体には，out item なる印刷の標準手続きが局所的に宣言されていて，手続き out list がその本体の中で LIST を呼び出すとき，ITEM を out item によって置きかえることによって LIST に定められた順序に印刷すべきパラメタを取り出して，LAYOUT の様式で印刷を行なうのである．すなわち LIST の型式的パラメタ ITEM は，標準手続き out list のメカニズムから要求されるパラメタなのである．

out list に対応する入出力手続きに in list がある．それはたとえば宣言

Example 5

```
procedure FORM;
  format ('3B,3ZD,3B,+Z6D');
DATA (ELEM);
  begin ELEM(N); ELEM(A[N])end;
```

の下にステートメント

in list (1, FORM, DATA)

を働かせれば，入力機器1に対して手続き FORM に指定された様式に従って2個のデータが読み込まれ，N と A[N] に与えられる．この際，標準手続き in list は，その本体に局所的な読込みの標準手続き in item で，プログラマによって宣言されて手続き DATA の型式的パラメタ ELEM を置きかえて，DATA を働かすことになる．in list の場合にも，読み込むべきデータの様式が定義されていなければ，当然コムパイラーによって定められた標準的な様式をもっているものとされる．

Format string の中には，行かえや頁送りの記号，/ や ↑ を適宜にはさむことができるし，文字を印刷する title format もおくことができるので，output n, input n (n=0, 1, 2, ……), out list, in list によって，簡単な表現で自由な型式をもって入出力を行なうことができる．

以上によって，output n, input n, out list, in list が実質的に IFIP-I/O の outreal, inreal を含んでいることがわかるし，IFIP-I/O の場合には insymbol や ontsymbol を用いて複雑なプログラムによって作らなければならない種々の入出力の機能をも含んでいることも推測できるであろう．そこで ACM-I/O では insymbol, outsymbol に対応する機能を果たす alpha format の役割はきわめて矮小化されている．insymbol や outsymbol のように記号と整数の対応をプログラムによって定める手段は与えられていないから，この対応はコムパイラーによって決められたものと考えなければならない．たとえば，データが ALPHA で，上記の対応が，A=1, H=2, L=3, P=4 と与えられていれば

Example 6

```
begin integer I;
    procedure LAYOUT;
      format ('AAAAA');
    procedure LIST (ITEM); ITEM(I);
    ......
    in list (1, LATOUT, LIST);
    ......
end
```

によって，整数 13421 が I に与えられる．このようにして読まれた値に対しては，＝や≠以外の比較，算術演算などを施してはならない．これは文法に対するちょっとした制約となるが，実質的には問題となるまい．それはともかくとして，標準函数の名前や，パラメタの数や specification が IFIP-I/O と，ACM-I/O で一致するようにできたとしても，記号に対する取扱い方が異なる結果として，IFIP-I/O が ACM-I/O のサブセットになっていないことになる．そのうえこのくいちがいのおもな源である insymbol と outsymbol が IFIP-I/O の場合にはたいへん重要なものであるということは何としても工合の悪い話しである．

次に出力の標準手続きによって実際に印刷や穿孔などが行なわれるわけであるが，これらの機能を印刷用紙やカードの仕様などの多様性にもかかわらず，一般的な方法で表現している点を説明する．

まず，ライン・プリンターの用紙を考えて，その1頁に印刷可能なすべての行に上より順に 1,2,3.… と

番号づけをし，その最大番号を P′ とする．また各行の最左端の印刷位置より，1, 2, 3,… と順に番号づけをして，その最右端の番号をPとする．Pは各行に印刷できる最大桁数であり，P′ は各頁に印刷できる最大行数であって，これらはその出力機器ごとに定まる数値である(**文献** Fig. 2 参照)．プログラムは，このほかに1行中の印刷の最左端 L と，最右端 R を指定することができる．たとえば標準函数 h lim を用いて

h lim (10, 30)

と書けば，プログラマがいじることのできないかくされた変数 H2 に L=10 と R=30 の値が記録される．実際に印刷が行なわれるときは，印刷すべき最初(左)の桁が 10 字目より左にあれば，10 字目までずらされてから印刷がはじまるし，印刷すべき最後(右)の桁が30字目より右になるならば行かえが行なわれ，さらにL字目までずらされて印刷される．もし，このような指定を行なわなければ L=1, R=∞ と指定されたと同じこととする．このときには印刷すべき最後の桁が紙の幅，すなわちPを越えてしまう場合には，上に説明したRを越えた場合と同じように取り扱う．

行方向でも同じようにして，

v lim(20, 60)

として，L′=20, R′=60 と指定できる．この場合 L′ は，頁ごとに最初に印刷のはじまる最上段の行，R′ は最下段の行の番号である．もちろんこの場合も指定がなければ L′=1, R′=∞ とする．

出力が紙テープの穿孔であったとしても全く同様に，P′=∞, P は一時に読まれる字数と考えればよい．カードや，磁気テープの場合にも，この方法で一般的に表現できているわけである．

プログラムの間違いで，R や P を越える印刷がなされるときは，上記のように行かえが生ずるのであるが，それにしてもなぜ行かえがおこったか，その理由を知りたい場合がある．したがって行かえの際に，プログラマが書いた手続きが呼ばれるようになっていれば便利である．これは標準手続き h end によって行なわれる．たとえば，記号 / や，Rの制限や，またはPの制限にひっかかって行かえが生じたときにとるべき処理を定めた．プログラマによって作られた手続きの名前をそれぞれ P_N, P_R, および P_P とすれば

h end (P_N, P_R, P_P)

と書けば，隠された変数 H_4 にこれらの名前が記録されて，それぞれの行かえのさいに，いずれかの手続きが呼ばれる．行の方向に関する同様な標準手続きが，v end である．もちろん h end や v end が使われなければ，だまって行かえ，頁送りをするだけである．

出力のための印刷様式制御の一例をあげる．

```
Example 8
procedure LAYOUT;
  begin format 1 ('↑,(X(2 B-4 Z.2 D),//)',n);
        h lim (11, 110); h end (K, L, L) end;
procedure K; h lim (11, 110);
procedure L; h lim (16, 105);
```

標準函数 format 1 は説明をはぶいたが，頁送りが行なわれた後に，n の数だけ

2 B-4 Z.2 D

が用いられ，次に行かえが2つはいる印刷様式を指定する．行の幅は左より 11 桁目から 110 桁目まで，/ による行かえがおこったときは手続きKによって，次の行から 11 桁目～110 桁目に印刷する．R または Pの範囲を越すときは手続きLによって，次の行から 16～105 桁目に印刷する．再び / によって 11～110 桁目の印刷に戻る．**例8**の LAYOUT はこのような形式を規定するものである．

例8は，紙幅が異なりRが違う出力機器に全く同じ手続きで印刷を行なことができることを示し，したがって，ACM-I/O によって書かれたプログラムの，各計算機ごとの互換性を保証することになる．

最後にちょっと特異な性格の標準手続き get と put について述べておく．

これは中間結果の入出に使用できるもので，LIST の手続き(**例 4,5**)を作って，適当な番号たとえば 100 をつけて

put (100, LIST)

と書くと，LIST で定められた各要素がその順に磁気テープ，ディスク，またはドラムなどの 100 と番号づけられたブロックに格納され

get (100, LIST)

ではそれが取り出されることになる．前者と後者の手続き LIST は，その要素の個数さえあっていれば，もちろん異なるものでよいから，これらを

```
begin begin……put (100, LIST 1)…… end;
……begin……get (100, LIST 2)……end end
```

と書いたときに，二つの解釈が存在する．一つは，100 という identification number は局所的な意味しかもたないとすることである，他は非局所的なものと考えることである．後者は非局所的な変数や array

を宣言したと同じ効果を与え，ブロックに対する一種の抜け道となるが，実用上はたいへん便利となるであろう．**文献4**では，後者の印象が強いが，この解釈は確定しておかないと困る．

以上で ACM-I/O を読むための手びきとしての紹介を終えることにする．

4. 国際標準言語の入出力としての問題

国際標準という観点で，いくつかの問題を述べた．国際標準言語の入出力手続きを二つ選ぶとすれば，やはり一方は他方のサブセットになっているべきである．ところが IFIP-I/O と ACM-I/O とは，セットとサブセットの関係になっていない．おもな問題は insymbol と outsymbol にあるけれども，これを IFIP-I/O から除くわけにはいかない．したがって，ACM-I/O に，これらを含ませる必要がある．その結果として alphaformat などの変更が要求されるであろう．また行かえや頁送りの手段の標準化を IFIP-I/O には必要とするし，したがって ACM-I/O にも同じものを，alignment mark ↑ や / との関係が不体裁にならないように含ませなければならない．

次に IFIP-I/O の表現力という問題がある．IFIP-I/O は ACM-I/O のサブセットとして作られたものではなく，かえってできるだけ一般的に入出力の機能をあらわそうとする考えのうえに作られている．それにもかかわらず，あるいはそれだからといったほうが適当であるかも知れないが，実用上は入出力に対してきわめて貧弱な手段しか提供していない．これはあとで例題によって示そう．プログラム言語には問題の演算過程と，計算機によるその実行の二面を記述する性格があり，IFIP では第一の側面に特に注意がはらわれている．しかし，入出力というものは第一の側面にはたいしたかかわりをもたないものである．入出力手続きが貧弱である結果として，プログラム中のその部分が大きくなって，演算過程の記述言語としての性格が圧迫をうけるということにもなりかねない．IFIP では，入出力手続きに対する見方を，ALGOL 60 の文法とはすこし変える必要があったのではなかろうか．

最後に，サブセット（IFIP）との関係にちょっとふれておく．サブセットの範囲では，たとえば inlist によって添字付変数に直接データを読みこむことはできない．これは名前によって呼ばれるパラメタには添字付変数を実際のパラメタとして与えてはいけないからである[2]．さらに工合の悪いことには ACM-I/O を現在の形のままでは利用できない．なぜならサブセット（IFIP）の範囲では，output 等の標準手続きのパラメタは実数型か整数型に specify されていなければならないし，それに与えられる実際のパラメタもいずれかの型にきめられてしまうからである[2]．サブセットと ACM-I/O の実用性を考えると，これらの点を調和させてもらいたいものである．

問題　n行 n列の行列 A の要素を行の順に印刷する．各要素の印刷様式は 2B-4Z.2D とし，行列の一行ごとに行かえをする．行列の一行が用紙の一行にはいらないときは残りの部分は次行に左より9字の余白をもって印刷をする．印刷はすべて1行おきとする．P=110 とする．

Example 9, ACM-I/O;

```
begin……; procedure K; h limit (1, 110);
    procedure L; h limit (10, 110);
    …… h end (K, L, L); output 0 (6,'↑');
    for i:=1 step 1 until n do begin
      for j:=1 step 1 until n do output 1
                          (6,'2B-4Z.2D',A[i, j]);
        output 0 (6,'//')end; ……end
```

Example 10, IFIP-I/O;

```
begin……;
  procedure OUTPUT (channel, source);
  value channel, source; integer channel;
  begin integer S, i, j;
    procedure D part (n); value n; integer n;
      begin integer i; for i:=1 step 1 until
                                    n do begin
          outsymbol (channel,'0123456789',
                              entier (source)+1);
          source:=10×(source −entier(source))
                                end end D part;
    S:=if source≥0 then 1 else 2;
    source:=(abs(source)+0.005)/1000;
    if rho+10>P then begin CR (channel);
                           CR (channel) end;
    outsymbol (channel, '␣', 1); outsymbol
                           (channel, '␣', 1);
    for i:=1 step 1 until 4 do
      if source<1 then begin outsymbol
                           (channel,'␣',1);
      source:=10×source end else go to L; i:=5;
L: outsymbol (channel, '+−',S);
```

```
                    D aprt(5−i);
  outsymbol (channel, '.',1);
                    D part (2) end OUTPUT;
procedure PS (channel); value channel; integer
  channel; begin if rho>0 then CR (channel);
      outsymbol (channel, ' ',−2);
      rho dash:=0 end PS;
  procedure CR (channel); value channel;
    integer channel; begin integer i; if
    rhodash≥P dash then PS (channel);
      outsymbol (channel, ' ',−1);
      for i:=1 step 1 until L−1 do outsymbol
                              (channel,'␣',1);
      rho:=L−1; rho dash:=rho dash+1
                                  end CR;
……; L:=1; PS(6);
for i:=1 step 1 until n do begin
  L:=10; for j:=1 step 1 until n do
                          OUTPUT(6,A[i,j]);
  L:=1; CR(6); CR (6) end; …… end
```

outsymbol (channel,' ',−1)は行かえ，outsymbol (channel,' ',−2) は頁送りを行なうこととする．

例10は．例9ほどに一般的な機能はもっていない．たとえば，桁数が固定しているとか，A [i, j] の値が指定された印刷様式をはみでるときの処理とか，印刷の列の方向の制御とかいろいろな点で単純に作られている．

参考文献

1) Peter Naur (Editor): Revised Report on the Algorithmic Language, Comm. ACM 6 (1963). 1～17.
2) W.L. van der Poel (Chairman): Report on SUBSET ALGOL 60 (IFIP). March, 1964.
3) D.E. Knuth et al: Aproposal for Input-Output conventions in ALGOL 60. Comm. ACM 7 (1964), 273～283.
4) W.L. van der Poel (Chairman): Report from W.G. 2.1 on Input-Output procedures. March, 1964.

(昭和39年9月25日受付)