SAS プログラムで関数とサブルーティンを作成する方法

周防　節雄
兵庫県立大学・名誉教授

How to Make Your Own SAS Functions and Subroutines

Setsuo Suoh
Professor Emeritus of the University of Hyogo

要旨

これまでSASの初心者が戸惑うことの一つとして、SASプログラムでは自作の関数とサブルーティンを作成する機能がないことが挙げられる。筆者は、最近まで LINK-RETURN 文やマクロ言語などを使うことで対処してきたが、数年前(SAS9.2 以降)から PROC FCMP を使って関数とサブルーティンの作成が可能となり、大変便利になった。本論文では、同じ処理をする SAS プログラムを、①初心者レベル、②ARRAY 文使用、③LINK-RETURN 文使用、④マクロ言語使用、⑤PROC FCMP を使って関数作成、⑥PROC FCMP を使ってサブルーティン作成の、6通りで作成して解説する。

キーワード　PROC FCMP、関数、サブルーティン、LINK-RETURN 文、マクロ言語、ARRAY 文

1. はじめに

筆者は SAS を使い始めて 30 数年になる。それまでは、FORTRAN や Algol でプログラミングをしてきた者にとって、SAS を使い始めた頃は戸惑うことが多かった。当時は今のように、分からないことはネット検索をすれば大抵のことが分かる時代ではなく、SAS のマニュアルも BASE と STAT の薄い 2 冊があるだけで、周りに SAS をよく知っている人もほとんどいない状況だった。そういう手探りの中で SAS のプログラミングを始めたのだが、当然困ることがいくつも出てきた。その一つが、SAS プログラムでは自作の関数とサブルーティンを作成する機能がないことであった。だんだん SAS を使っていくうちに、LINK-RETURN 文、ARRAY 文、マクロ言語を憶え、それらを使って関数やサブルーティンの代用品として対処してきたが、数年前(SAS9.2 以降)から PROC FCMP を使って関数とサブルーティンの作成が可能となり、大変便利になった。本論文では、簡単な例題を取り上げて、SAS を使い始めたユーザーにも分かるように、同じ処理をする SAS プログラムを、①初心者レベル、②ARRAY 文使用、③LINK-RETURN 文使用、④マクロ言語使用、⑤PROC FCMP を使って関数作成、⑥PROC FCMP を使ってサブルーティン作成の、6通りで作成して解説する。

2. 例題

表1に示す架空の 5 名の個人情報のエクセルファイルを取り上げる。同表には各自の①ID 番号、②生年、③結婚した年の三つの情報が含まれている。②や③には同表に示すように、西暦と和暦が混在している。和暦は、例えば、平成3年なら「平 3」、明治 44 年なら「明 44」のように表記されている。③から②を引き算して、結婚した時の年齢を計算する SAS プログラムを作成する。引き算をするためには、和暦を全て西暦に換算しなければならない。最終的には、その換算をする関数

表 1　婚姻年齢.xlsx

① idno	② birthyr	③ wedyr
A1001	昭25	1983
A1002	平3	平26
A1003	大14	昭30
A1004	1989	2014
A1005	明44	昭21

とサブルーティンを作成するのが目的である。次節以降で6種類の処理方法を解説するが、プログラムそのものは、便宜上、一つの SAS ファイル(婚姻年齢.sas)に納めている(付録参照)。

3. エクセル表を SAS データセットに変換

右のプログラムを実行すると、表 1 のエクセル表が図1に示すパーマネント SAS データセット wedage に変換される。

data=wedage

OBS	idno	birthyr	wedyr
1	A1001	昭25	1983
2	A1002	平3	平26
3	A1003	大14	昭30
4	A1004	1989	2014
5	A1005	明44	昭21

図 1　データセット wedage

```
/* 婚姻年齢.sas */ options nocenter;
libname FCMP "G:¥sasFCMP¥sasds";

proc import out=FCMP.wedage
    datafile="G:¥sasFCMP¥婚姻年齢.xlsx"
    dbms=excel replace;
    getnames=yes;
    sheet="sheet1";
run;

proc print; title "data=wedage"; run;
```

4. 和暦を西暦に換算するアルゴリズム

表 1 の和暦表現は明治、大正、昭和、平成の先頭の漢字一文字とそれに続く1～2桁の半角数字から成るので、文字変数としては、長さ4バイトになる。和暦か西暦かの区別は、先頭の2バイトが「明大昭平」か「19」か「20」のいずれかであるかによって判定できる。SELECT-WHEN 文を使って和暦ならそれぞれの「元年の西暦年－1」を変数 startyear に設定しておき、後で3～4バイト目の数字を startyear に加算すれば、西暦に換算できる。また、初めから西暦なら startyear に一旦0を設定しておき、後で元の西暦年の値を加算すればよい。プログラムでは、欠損値または表記エラーのケースも想定し、othrewise として startyear に数値型の欠損値「半角ピリオド」を設定しておく。ちなみに、生年の換算だけを取り出すと、このようなプログラムになる。

```
select(substr(birthyr,1,2)); *★★生年を西暦変換★★;
  when("明") startyear=1867;
  when("大") startyear=1911;
  when("昭") startyear=1925;
  when("平") startyear=1988;
  when("19") startyear=0;
  when("20") startyear=0;
  otherwise  startyear=.; *欠損値又はエラー;
end;
if startyear > 0 then birthyr=startyear+substr(birthyr,3,2);
                 else birthyr=startyear+birthyr;
```

5. 初心者レベルのプログラム

前節で生年の西暦換算のアルゴリズムを示したが、結婚した年でも同じ換算処理が必要になる。取りあえず、そのアルゴリズムをコピー・ペーストした後、変更が必要な箇所を修正して出来たのが以下に示すプログラムで、実行結果も示す(図2)。これはこれで今は問題ないが、仮に、両親の生年もデータに含まれていて、本人が生まれた年の両親の年齢も計算する場合を考えると、コピー・ペースト作業が更に 2 回必要になり、当然その都度変数名等を修正することになる。こういうコピー・ペースト作業を繰り返していると、うっかり変数名の修正をやり損なう箇所が出てきてもおかしくないし、プログラムを書いた経験のある人なら、そのようなミスをしたことは必ずあると思う。とりあえず

正常に作動すれさえすれば良いと思ってプログラムを書いているユーザーにこのようなプログラムが多いが、高度なプログラムを作成するには限界がある。

```
*★★★　①初心者のプログラム　★★★;
data beginner (drop=startyear); set FCMP.wedage;
  select(substr(birthyr,1,2)); *★★生年を西暦変換★★;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then birthyr=startyear+substr(birthyr,3,2);
                   else birthyr=startyear+birthyr;

  select(substr(wedyr,1,2)); *★★結婚した年を西暦変換★★;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then wedyr=startyear+substr(wedyr,3,2);
                     else wedyr=startyear+wedyr;

  wedage=wedyr-birthyr; *結婚した時の年齢計算;
run;

proc print data=beginner; title "①初心者のプログラム"; run;
```

①初心者のプログラム

OBS	idno	birthyr	wedyr	wedage
1	A1001	1950	1983	33
2	A1002	1991	2014	23
3	A1003	1925	1955	30
4	A1004	1989	2014	25
5	A1005	1911	1946	35

図２　実行結果

6. ARRAY 文使用のプログラム

同じアルゴリズムのコピー・ペーストの回避策として、array 文を使う方法がある。つまり、同じアルゴリズムで処理をする対象の全ての変数をひとつの array の中に並べてから、do ループで回せばよい。具体的には、「array arg $ birthyr wedyr;」とすれば、arg{1}と arg{2}はそれぞれ変数 birthyr と wedyr に対応しているので、do ループから出てきた時には、両方の変数とも西暦換算済みにある。

（六通りのプログラムの実行結果は、title 以外は図 2 と全く同じなので省略する。）

```
*★★★　②ARRAY 文使用のプログラム　★★★;
data array_used (drop=i startyear); set FCMP.wedage;
  array arg $ birthyr wedyr;
  do i=1 to 2;
    select(substr(arg{i},1,2)); *★★年を西暦変換★★;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then arg{i}=startyear+substr(arg{i},3,2);
                     else arg{i}=startyear+arg{i};
  end;
  wedage=wedyr-birthyr; *結婚した時の年齢計算;
run;

proc print data=array_used; title "②ARRAY 文使用"; run;
```

7. LINK-RETURN 文使用のプログラム

SAS にはこれまで自作の関数やサブルーティンを定義して利用する機能がなかったが、筆者はこの対策の一つとして使ってきたのが LINK-RETURN 文である。以下に示す data ステップの先頭部がいわゆるメインプログラムに相当し、それに続く箇所が「開いたサブルーティン」に相当する。

LINK-RETURN 文の仕組みは以下の①～④で解説している。

① 実引数に相当する変数 birthyr を仮の引数に相当する変数 YEAR に代入する。この時の「引数」の個数には制限はない。

② LINK 文で「開いたサブルーティン」の先頭行の行ラベルを指定する。ここで指定した行ラベル CONVERT から RETURN 文までが開いたサブルーティンに相当する。

③ この LINK 文が実行されると、そこで指定されたラベル行にプログラムの制御が移る。GOTO 文と同じように見えるが、ここでは GOTO 文は使えない。理由は、GOTO 文では後で自動的に元の箇所に制御を戻すことができないからである。

④ 指定された行ラベルに制御が移った後は、通常のプログラムと同じ動作をするが、RETURN 文まで来ると、先に実行された LINK 文の「次の命令文」に制御が自動的に戻される。

このプログラムでは、まず変数 YEAR に変数 birthyr を代入後、LINK 文で行ラベル CONVERT に制御を飛ばす。飛ばされた先で初めて出会う RETURN 文によってもと来た所に戻されて、次の命令文を実行することになり、換算結果の値を持つ変数 YEAR を変数 birthyr に代入する。こうすることで、何度でも開いたサブルーティンに飛んで、計算が終われば「自動的に」もと来た場所に戻ってくることが出来るので、前節のように煩わしいコピー・ペースト作業をする必要はない。同じことを変数 wedyr にも施した後、引き算をして結婚した年齢 wedage が計算できる。つまり、変数 YEAR は「仮引数」の役割を果たしている。

実は、ここで極めて重要なことがある。メインプログラムに相当する部分の最後に return 文がある。この return 文と「開いたサブルーティン」部の最後にある RETURN 文は共に「明示的 return 文」(explicit return)と呼ばれるが、役割は同じではない。「メインプログラムに相当」にある return 文は、これが実行されると現在のオブザベーション(正確には、PDV(プログラム・データベクトル)をデータセットに保存した後、次のオブザベーションに処理が移る。仮に、この return 文がないとすると、すぐ下の行ラベル CONVERT にある select 文を実行されてしまう。わかりやすく言えば、この明示的 return 文が関所の役割を果たして、その下の命令文には制御が行かないように阻止している。

```
*★★★  ③LINK-RETURN 文使用のプログラム  ★★★;
data link_return (drop=startyear YEAR); set FCMP.wedage;
  YEAR=birthyr; LINK CONVERT; birthyr=YEAR;
  YEAR=wedyr;   LINK CONVERT; wedyr  =YEAR;
  wedage=wedyr-birthyr;
  return; *明示的 return 文;
  CONVERT: *行ラベル;
  select(substr(YEAR,1,2)); *年を西暦変換;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then YEAR=startyear+substr(YEAR,3,2);
                   else YEAR=startyear+YEAR;
  RETURN;
run;

proc print data=link_return; title "③LINK-RETURN 文使用"; run;
```

メインプログラムに相当

開いたサブルーティン

8. SAS マクロ使用のプログラム

第４節で示した和暦を西暦に換算するアルゴリズムをSASのマクロ言語で表現したコードを右に示す。このマクロを実行するプログラムはその下に示す。

```
*★★★　④SAS マクロ使用のプログラム　★★★;
%macro convert(YEAR);
  select(substr(&YEAR,1,2)); *年を西暦変換;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then &YEAR=startyear+substr(&YEAR,3,2);
                   else &YEAR=startyear+&YEAR;
%mend convert;
```

実際の和暦・西暦換算は、data ステップにある

%convert(birthyr);

%convert(wedyr);

で行われるが、SAS が実際に実行するのは、このマクロを展開した結果、自動作成されたSASコードである。「options macrogen;」を指定しておけばその展開結果が以下のように、LOG 画面に表示される。

```
options macrogen;

data macro_use (drop=startyear); set FCMP.wedage;
  %convert(birthyr);
  %convert(wedyr  );
  wedage=wedyr-birthyr;
run;

proc print data=macro_use; title "③SAS マクロ使用"; run;
```

この LOG 画面に表示されたプログラムは第5節で示した初心者レベルのプログラムそのものである。つまり、マクロで表現しておけば、実行時に(正確にはコンパイル時に)SAS システムが自動でコピー・ペースト作業をしてくれるので、マニュアル作業で行う際のうっかりミスはまったくなく安全である。ユーザーにしてみれば、convert というマクロを一度定義しておけば、あたかもサブルーティンのように使え、メインプログラムに相当する data ステップは極めて簡潔で読みやすくなる。

LOG 画面の
マクロ展開結果

```
237     %convert(birthyr);
MACROGEN(CONVERT):   select(substr(birthyr,1,2));
MACROGEN(CONVERT):   *年を西暦変換;
MACROGEN(CONVERT):   when("明") startyear=1867;
MACROGEN(CONVERT):   when("大") startyear=1911;
MACROGEN(CONVERT):   when("昭") startyear=1925;
MACROGEN(CONVERT):   when("平") startyear=1988;
MACROGEN(CONVERT):   when("19") startyear=0;
MACROGEN(CONVERT):   when("20") startyear=0;
MACROGEN(CONVERT):   otherwise startyear=.;
MACROGEN(CONVERT):   *欠損値又はエラー;
MACROGEN(CONVERT):   end;
MACROGEN(CONVERT):   if startyear > 0 then
                                 birthyr=startyear+substr(birthyr,3,2);
MACROGEN(CONVERT):   else birthyr=startyear+birthyr;
238     %convert(wedyr  );
MACROGEN(CONVERT):   select(substr(wedyr,1,2));
MACROGEN(CONVERT):   *年を西暦変換;
MACROGEN(CONVERT):   when("明") startyear=1867;
MACROGEN(CONVERT):   when("大") startyear=1911;
MACROGEN(CONVERT):   when("昭") startyear=1925;
MACROGEN(CONVERT):   when("平") startyear=1988;
MACROGEN(CONVERT):   when("19") startyear=0;
MACROGEN(CONVERT):   when("20") startyear=0;
MACROGEN(CONVERT):   otherwise startyear=.;
MACROGEN(CONVERT):   *欠損値又はエラー;
MACROGEN(CONVERT):   end;
MACROGEN(CONVERT):   if startyear > 0 then
                                   wedyr=startyear+substr(wedyr,3,2);
MACROGEN(CONVERT):   else wedyr=startyear+wedyr;
```

9. PROC FCMP で関数の作成

本節では SAS9.2 から追加された新機能の proc fcmp を使って和暦・西暦換算をする関数 nengo を作成する。この関数を定義するコードを以下に示す。関数作成時の注意事項としては、最初に libname 文で3層構造のフォルダを定義しておかなければならない。次に、proc fcmp 文の中で outlib= オプションを使って、そのフォルダに対応する**3次**のライブラリ参照名を指定する。そうすることで最終的に、この例では、「F:¥sasFCMP¥functions¥conversion」のフォルダの中に二つの SAS ファイルが作成される。ひとつは functions.sas7bdat という SAS データセットが作成されており、ダブルクリックして開いてみると、その中に「value」という名前の変数があり、この関数の SAS コードがテキスト形式でそっくり保存されているのが分かる。もうひとつは functions.sas7bndx という SAS ファイルで、その中にはオブジェクトコード化された関数が保存され、この関数の使用時にはこれが直接使われると思われる。

```
*★★★  ⑤PROC FCMP で function 作成  ★★★;
libname sasFCMP "F:¥sasFCMP¥functions¥conversion";

proc fcmp outlib=sasFCMP.functions.conversion;
  FUNCTION nengo(YEAR $); ********************************;
    select(substr(YEAR,1,2)); *生年を西暦変換;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then xyear=startyear+substr(YEAR,3,2);
                     else xyear=startyear+YEAR;
      return(xyear);
  ENDSUB; ************************************************;
run;
```

関数を定義する本体部分は、FUNCTION 文から ENDSUB 文までである。FUNCTION 文で関数名および仮の引数とその変数型を定義する。デフォルトの変数型は数値型なので、その場合には変数の型を指定しなくて済む。文字型なら「$」を指定する。計算処理をする部分は、data ステップで使える命令文を使って記述する。この関数の計算結果を保存している変数 xyear を return 文で指定すると、この値が関数値となる。

実際にこの関数を使う data ステップを以下に示す。ここで注意しておくことは、data ステップで使用する前に options 文で cmplib=を使って、先に proc fcmp outlib=で指定した参照名の先頭の二つからなる**2次名**を指定することである。これは以下に示すコードのコンパイル時に必要な関数の保存場所であり、この例では、フォルダ sasFCMP の中にあるフォルダ functions の中に当該関数があることを示している。data ステップで使用する際は、通常の関数を使用するのと同じようにすればよい。

```
options cmplib=(sasFCMP.functions);

data function_use; set FCMP.wedage;
  wedage=nengo(wedyr)-nengo(birthyr);
run;

proc print data=function_use; title "⑤PROC FCMP で function 作成"; run;
```

10. PROC FCMP でサブルーティンの作成

和暦・西暦換算をするサブルーティン nengo_conv を proc fcmp で作成するのは、先の関数を作成するのとほぼ同じ手順でできる。異なる点は三点有り、①SUBROUTINE 文でサブルーティン名を定義して、かつ、必要なら、メインプログラムに相当する data ステップへ引き渡す仮引数と、data ステップに戻す仮引数を指定すること、②outargs 文で data ステップに戻す仮引数を定義しておくこと、③計算結果は実引数に持たせるので、関数のような return 命令は要らないことである。

このサブルーティンをdataステップで使用するには、普通にcall文を使い、必要に応じて実引数を指定する。以下の例では、生年の変数(birthyr)と結婚した年の変数(wedyr)からそれぞれ西暦換算した新しい変数xbirthyrとxwedyrが作成される。ここで気になるのは、この二つの新変数に事前に欠損値を与えて初期化していることである。実はこれがなくても変数 wedage は正しく計算されるが、LOG 画面には、「NOTE: 変数 xbirthyr は初期化されていません。NOTE: 変数 xwedyr は初期化されていません。」というメッセージが表示される。確かに、「xbirthyr=.;」がなければ、変数xbirthyrは初めて現れた変数なので初期化されていないのは当たり前なのだが、サブルーティン実行後には値が正しく与えられるから問題ないはずである。エラーメッセージでも警告メッセージでもないので、放っておいて問題はないと判断するが、現段階ではとりあえずすべての実引数に欠損値を call 直前で与えておくことにするが、いずれ将来このLOGにある余計なメッセージは出なくなると思う。

```
*★★★  ⑥PROC FCMP で subroutine 作成  ★★★;
libname sasFCMP "G:¥sasFCMP¥functions¥conversion";

proc fcmp outlib=sasFCMP.functions.conversion;
  SUBROUTINE nengo_conv(YEAR $,xyear); ********************;
    outargs xyear; *メインプログラムに引き渡す仮引数;
    select(substr(YEAR,1,2)); *生年を西暦変換;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then xyear=startyear+substr(YEAR,3,2);
                     else xyear=startyear+YEAR;
  ENDSUB; ************************************************;
run;
```

```
options cmplib=(sasFCMP.functions);

data subroutine_use (drop=xbirthyr xwedyr); set FCMP.wedage;
  xbirthyr=.; call nengo_conv(birthyr,xbirthyr);
  xwedyr  =.; call nengo_conv(wedyr  ,xwedyr  );
  wedage=xwedyr-xbirthyr;
run;
```

11. まとめ

SAS では、これまで自前の関数とサブルーティンを作成する機能がなく、それに代わる方法をいろいろ工夫しながら考えて切り抜けてきた。それらの技法をまず解説した後、SAS9.2 から追加された新機能の proc fcmp を使って、和暦・西暦換算をする関数 nengo とサブルーティン nengo_conv を作成し、自前の関数とサブルーティンの作り方と使用法を解説した。

そもそも proc fcmp が出現したことを当初は知らなかった。2012 年に米国フロリダ州で開催された SAS の世界大会(SAS Global Forum 2012)で数独を解く SAS プログラムを招待論文として筆者が報告した際、その準備段階でカナダ人の世話人(参考文献1)から、自分たちは proc fcmp を使って数独プログラムを作る旨のメールを受け取り、その存在を知った次第である。

本稿で示したように、自前の関数とサブルーティンが手軽に作れるようになると、data ステップが格段に簡潔になり、見やすく、しかも信頼度の高いプログラムになる。今後ユーザーの皆さんにもぜひ活用されることを大いに望んでいる。参考文献2は分かりやすく proc fcmp の解説をしている。

参考文献

1. Tabachneck, A. S. & Kastin, M.(2012) Yet Another Sudoku Solver: PROC FCMP, SAS Global Forum 2012, http://support.sas.com/resources/papers/proceedings12/433-2012.pdf
2. Carpenter, A. L.(2013) Using PROC FCMP to the Fullest: Getting Started and Doing More, SAS Global Forum 2013, http://support.sas.com/resources/papers/proceedings13/139-2013.pdf

謝辞

本稿は学術振興会科学研究費(課題番号 24530232:研究代表者・周防節雄)による研究成果の一部である。記して謝意に代えたい。

付録　SAS プログラム

```
/* 婚姻年齢.sas */ options nocenter;
libname FCMP "G:¥sasFCMP¥sasds";

proc import out=FCMP.wedage
    datafile="G:¥sasFCMP¥婚姻年齢.xlsx"
    dbms=excel replace;
    getnames=yes;
    sheet="sheet1";
run;

proc print; title "data=wedage"; run;

*★★★　①初心者のプログラム　★★★;
data beginner (drop=startyear); set FCMP.wedage;
  select(substr(birthyr,1,2)); *★★生年を西暦変換★★;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then birthyr=startyear+substr(birthyr,3,2);
                   else birthyr=startyear+birthyr;

  select(substr(wedyr,1,2));  *★★結婚した年を西暦変換★★;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then wedyr=startyear+substr(wedyr,3,2);
                   else wedyr=startyear+wedyr;

  wedage=wedyr-birthyr; *結婚した時の年齢計算;
run;

proc print data=beginner; title "①初心者のプログラム"; run;
```

```
*★★★　②ARRAY 文使用のプログラム　★★★;
data array_used (drop=i startyear); set FCMP.wedage;
  array arg $ birthyr wedyr;
  do i=1 to 2;
    select(substr(arg{i},1,2)); *★★年を西暦変換★★;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then arg{i}=startyear+substr(arg{i},3,2);
                     else arg{i}=startyear+arg{i};
  end;
  wedage=wedyr-birthyr; *結婚した時の年齢計算;
run;

proc print data=array_used; title "②ARRAY 文使用"; run;

*★★★　③LINK-RETURN 文使用のプログラム　★★★;
data link_return (drop=startyear YEAR); set FCMP.wedage;
  YEAR=birthyr; LINK CONVERT; birthyr=YEAR;
  YEAR=wedyr;   LINK CONVERT; wedyr  =YEAR;
  wedage=wedyr-birthyr;
  return;

  CONVERT: *行ラベル;
  select(substr(YEAR,1,2)); *年を西暦変換;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then YEAR=startyear+substr(YEAR,3,2);
                   else YEAR=startyear+YEAR;
  RETURN;
run;

proc print data=link_return; title "③LINK-RETURN 文使用"; run;

*★★★　④SAS マクロ使用のプログラム　★★★;
%macro convert(YEAR);
  select(substr(&YEAR,1,2)); *年を西暦変換;
    when("明") startyear=1867;
    when("大") startyear=1911;
    when("昭") startyear=1925;
    when("平") startyear=1988;
    when("19") startyear=0;
    when("20") startyear=0;
    otherwise  startyear=.; *欠損値又はエラー;
  end;
  if startyear > 0 then &YEAR=startyear+substr(&YEAR,3,2);
                   else &YEAR=startyear+&YEAR;
%mend convert;

options macrogen;

data macro_use (drop=startyear); set FCMP.wedage;
  %convert(birthyr);
  %convert(wedyr  );
  wedage=wedyr-birthyr;
run;
```

```
proc print data=macro_use; title "④SAS マクロ使用"; run;

*★★★  ⑤PROC FCMP で function 作成  ★★★;
libname sasFCMP "G:¥sasFCMP¥functions¥conversion";

proc fcmp outlib=sasFCMP.functions.conversion;
  FUNCTION nengo(YEAR $); ********************************;
    select(substr(YEAR,1,2)); *年を西暦変換;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then xyear=startyear+substr(YEAR,3,2);
                     else xyear=startyear+YEAR;
      return(xyear);
  ENDSUB; ************************************************;
run;

options cmplib=(sasFCMP.functions);

data function_use; set FCMP.wedage;
  wedage=nengo(wedyr)-nengo(birthyr);
run;

proc print data=function_use; title "⑤PROC FCMP で function 作成"; run;

*★★★  ⑥PROC FCMP で subroutine 作成  ★★★;
libname sasFCMP "G:¥sasFCMP¥functions¥conversion";

proc fcmp outlib=sasFCMP.functions.conversion;
  SUBROUTINE nengo_conv(YEAR $,xyear); ********************;
    outargs xyear; *メインプログラムに引き渡す仮引数;
    select(substr(YEAR,1,2)); *年を西暦変換;
      when("明") startyear=1867;
      when("大") startyear=1911;
      when("昭") startyear=1925;
      when("平") startyear=1988;
      when("19") startyear=0;
      when("20") startyear=0;
      otherwise  startyear=.; *欠損値又はエラー;
    end;
    if startyear > 0 then xyear=startyear+substr(YEAR,3,2);
                     else xyear=startyear+YEAR;
  ENDSUB; ************************************************;
run;

options cmplib=(sasFCMP.functions);

data subroutine_use (drop=xbirthyr xwedyr); set FCMP.wedage;
  xbirthyr=.; call nengo_conv(birthyr,xbirthyr);
  xwedyr  =.; call nengo_conv(wedyr  ,xwedyr  );
  wedage=xwedyr-xbirthyr;
run;

proc print data=subroutine_use; title "⑥PROC FCMP で subroutine 作成"; run;
```