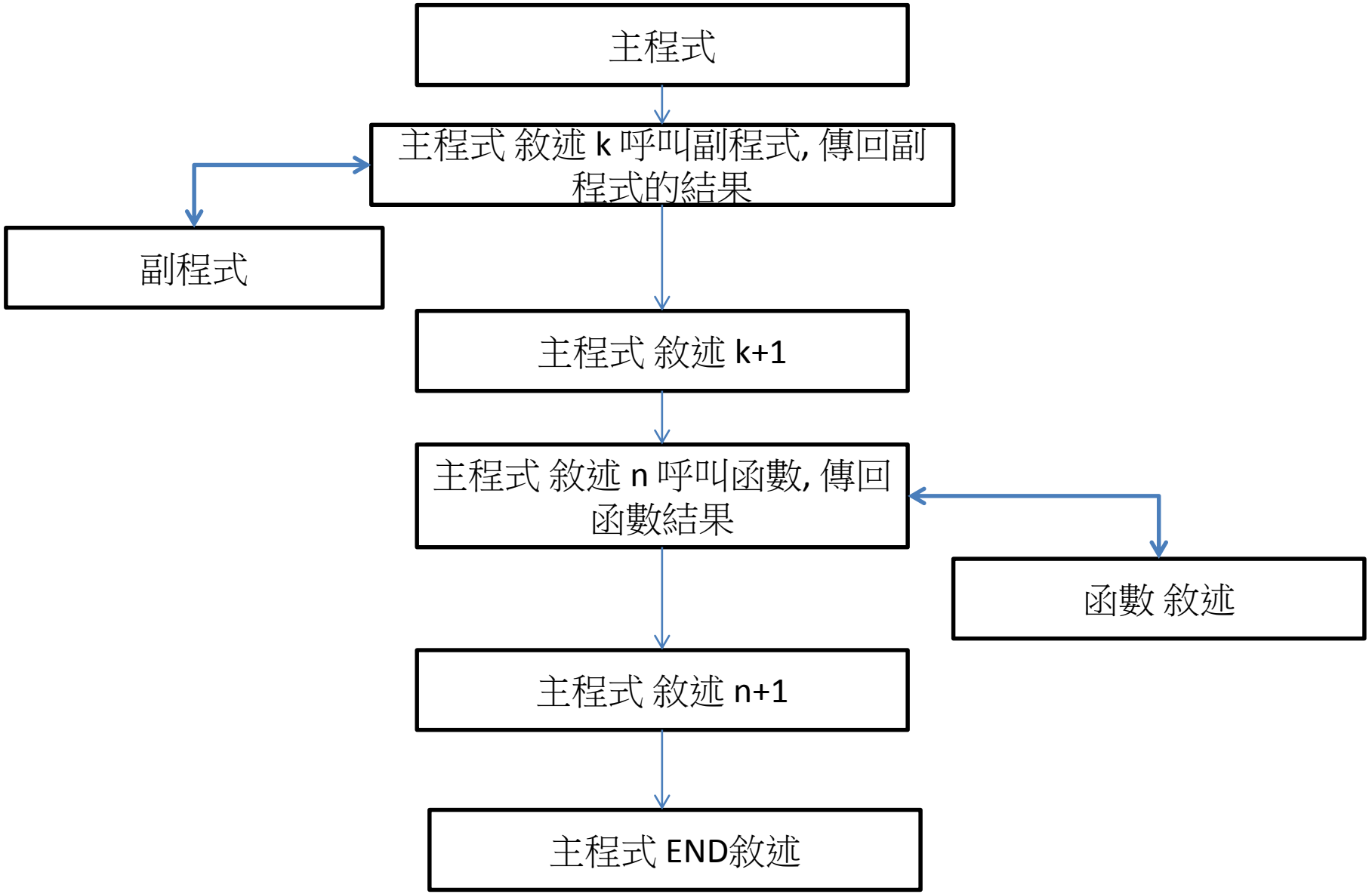
Ch. 15
副程式的實施

當程式(program)中的一段指令敘述需要常被使用時，可考慮把這段指令敘述獨立出來，發展成一個副程式(subroutine)


主程式
主程式 敘述 k 呼叫副程式, 傳回副程式的結果
副程式
主程式 敘述 k+1
主程式 敘述 n 呼叫函數, 傳回函數結果
函數 敘述
主程式 敘述 n+1
主程式 END敘述

IDL程式的種類

種類	語法
主程式 (Main Program)	敘述 1 … 敘述 n END
副程式函數 (Function)	Function NAME, 引數1, …,引數N 敘述 1 … 敘述 n RETURN, 表示式 END
副程式程序 (Procedure)	Pro NAME, 引數1,…,引數N 敘述 1 … 敘述 n RETURN (可省略) END
批次檔 (Batch File)	敘述 1 … 敘述 n

主程式 IDL_Ch15_main_1.pro

```
a=indgen(5)+1
print,max(a),min(a)
print,'mean=',mean(a),' std=',stddev(a)
help,a

ave_f = mean_fun1(a)
print,'ave_f=',ave_f
mean_p1,a,ave_p
print,'ave_p=',ave_p
end
```

副程式函數 mean_fun1.pro

```
FUNCTION mean_fun1,array
   sum1 = TOTAL(array)
   no1 = N_ELEMENTS(array)
   ave = sum1 / no1
RETURN, ave
END
```

副程式程序**mean_p1.pro**

```
PRO mean_p1,array,ave
   sum2 = TOTAL(array)
   no2 = N_ELEMENTS(array)
   ave = sum2 / no2
RETURN
END
```

批次檔 batch_1.pro

.RUN mean_fun1

.RUN mean_p1

.RUN IDL_Ch15_main_1

```
@batch_1
% Compiled module: MEAN_FUN1.
% Compiled module: MEAN_P1.
% Compiled module: $MAIN$.
      5       1
mean=      3.00000 std=      1.58114
A               INT       = Array[5]
ave_f=      3.00000
ave_p=      3.00000
IDL>
```

IDL執行指令的輸入位置

輸入位置	方式
視窗界面上的選單	點選視窗界面的編譯(compile)和執行(跑 run)按鈕
指令列	鍵入編譯和執行的指令

IDL的編譯和執行指令

指令	說明
.COMPILE 主程式	編譯主程式
.COMPILE 副程式	編譯副程式
.RUN 主程式	編譯執行主程式
@批次檔	執行批次檔
.RNEW 主程式	與.RUN指令功能類似，但會先清除以前留下的變數
.RESET_SESSION	不需要跳出，即可重新啟動IDL

COMPILE_OPT指令的語法

語法	說明
COMPILE_OPT	改變系統的編譯規則

IDL編譯主副程式的預設規則是”使用二每位元組的整數”，”不區分中括號和小括號的用法”，及”顯示編譯後的主副程式名稱”
若要回復預設的編譯選項，則需離開IDL系統再重新啟動，或鍵入 .RESET_SESSION指令重新啟動IDL

COMPILE_OPT指令的選項

選項	說明
DEFINT32	使用四個位元組來表示一個整數
STRICTARR	嚴格區分中括號和小括號的用法
HIDDEN	宣告此程式在編譯後隱藏
IDL2	是選項DEFINT32和STRICTARR的簡稱

```
IDL> compile_opt defint32
IDL> .run idl_ch15_main_1
% Compiled module: $MAIN$.
           5           1
mean=      3.00000 std=      1.58114
A               LONG      = Array[5]
ave_f=      3.00000
ave_p=      3.00000
IDL>
```

IDL主程式呼叫副程式的方式

副程式種類	主程式呼叫方式
函數(Function)	Result = NAME(引數1,…,引數N)
程序(Procedure)	NAME,引數1,…,引數N

練習 1

試寫一主程式，產生一維亂數常態分布的浮點數(實數)向量(長度自定)，再分別以呼叫先前所建立的副程式函數和副程式程序，再分別顯示副程式函數和副程式程序的輸出結果。

練習 1

```
a=RANDOMN(seed,100)
ave1=mean_fun1(a)
print,'ave1=',ave1
mean_p1,a,ave2
print,'ave2=',ave2
end
```

IDL執行字串指令

指令	說明
CALL_FUNCTION	以字串呼叫函數
CALL_PROCEDURE	以字串呼叫程序
EXCUTE	執行指令字串 (在Virtual Machine模式上無法使用)

```
a=[5,11,8,9,3]
d='mean_fun1'
ave1 = CALL_FUNCTION(d,a)
print,'ave1=',ave1
d='mean'
ave2 = CALL_FUNCTION(d,a)
print,'ave2=',ave2
e='mean_p1'
CALL_PROCEDURE,e,a,ave3
print,'ave3=',ave3
f='mean_p1,a,ave4'
g = EXECUTE(f)
print,'ave4=',ave4
print,'g=',g
end
```

```
% Compiled module: $MAIN$.
ave1=      7.20000
ave2=      7.20000
ave3=      7.20000
ave4=      7.20000
g=       1
IDL>
```

IDL副程式中處理訊息傳遞的函數

函數	功能
ARG_PRESENT	檢查引數的引用狀況
KEYWORD_SET	檢查關鍵字的引用狀況
N_PARAMS	檢查引數被引用的數目

```
FUNCTION mean_fun2,array,DOUBLE=DOUBLE
IF KEYWORD_SET(DOUBLE) THEN BEGIN
  PRINT,'keyword_set=','Double'
ENDIF
PRINT,'n_params=',N_PARAMS()
IF ARG_PRESENT(array) EQ 0 THEN BEGIN
  RETURN, !VALUES.F_NAN
ENDIF
sum1 = TOTAL(array)
no1 = N_ELEMENTS(array)
ave1 = sum1 / no1
RETURN,ave1
END
```

主程式

```
a=[5,3,8,7,6]
b = mean_fun2(a)
print,'b=',b
c = mean_fun2()
print,'c=',c
d=mean_fun2(a,/DOUBLE)
print,'d=',d
END
```

```
% Compiled module: $MAIN$.
n_params=       1
b=     5.80000
n_params=       0
c=       NaN
keyword_set=Double
n_params=       1
d=     5.80000
IDL>
```

COMMON資料區塊的語法

語法	說明
COMMON Name, Var1, …,VarN	定義程式之間的共用變數 Var1,…,VarN

```
Pro mean_p2,array
COMMON block1, ave
sum2 = TOTAL(array)
no2 = N_ELEMENTS(array)
ave = sum2 / no2
RETURN
END
```

主程式

```
COMMON block1,ave5
a = [5,3,8,9,6]
MEAN_P2,a
print,'ave5=',ave5
END
```

```
% Compiled module:
$MAIN$.
ave5=      6.20000
IDL>
```

IDL處理錯誤訊息的程序

程序	說明
CATCH [,Variable] [,/CANCEL]	當程式執行發生錯誤時，錯誤處理程序會啟動，使得整個程式不會因為此錯誤的發生而停頓
ON_ERROR,N	當副程式的執行發生錯誤時，此程序指示系統處理的方式
ON_IOERROR,Label	當I/O的執行發生錯誤時，此程序將跳至標號Label的位置

```
pro catch_ex
  CATCH, variable
    print,'variable='
  IF variable NE 0 THEN BEGIN
     print,'err_mag'
     PRINT, !ERROR_STATE.MSG
     a=1
     print,'variable=',variable
     CATCH,/CANCEL
     print,'a=',a
  ENDIF
  PRINT,a
  print,'2..',!error_state.msg
RETURN
END
```

```
IDL> catch_ex
% Compiled module: CATCH_EX.
variable=
variable=
err_mag
PRINT: Variable is undefined: A.
variable=       -178
a=       1
      1
2..PRINT: Variable is undefined: A.
IDL>
```

```
PRO CATCH_EXAMPLE
  ; Define variable A:
  A = FLTARR(10)
  help,A
  ; Establish error handler. When errors occur, the index of the
  ; error is returned in the variable Error_status:
  CATCH, Error_status
  ;This statement begins the error handler:
  IF Error_status NE 0 THEN BEGIN
    PRINT, 'Error index: ', Error_status
    PRINT, 'Error message: ', !ERROR_STATE.MSG
    ; Handle the error by extending A:
    A=FLTARR(12)
    CATCH, /CANCEL
  ENDIF
  ; Cause an error:
  A[11]=12
  ; Even though an error occurs in the line above, program
  ; execution continues to this point because the event handler
  ; extended the definition of A so that the statement can be
  ; re-executed.
  HELP, A
END
```

```
IDL> .reset_session
IDL> catch_example
% Compiled module: CATCH_EXAMPLE.
A               FLOAT     = Array[10]
Error index:        -144
Error message: Attempt to subscript A
with <INT      (      11)> is out of range.
A               FLOAT     = Array[12]
IDL>
```

ON_ERROR程序引數N的選項

選項	說明
0	讓程式停留在錯誤發生的地方，且列印主副程式目前的堆疊，此為預設值
1	當錯誤發生時，讓程式停留在主程式的位置，且列印副程式目前的堆疊
2	當錯誤發生時，讓程式停留在主程式的位置，且列印從錯誤發生之副程式至主程式的堆疊
3	讓程式停留在錯誤發生的地方，且列印從錯誤發生之副程式至主程式的堆疊

```
print,'main_1'
on_error_ex,0
print,'main_2
end
```

n=0
讓程式停留在錯誤發生的地方，且列印主副程式目前的堆疊，此為預設值

```
pro on_error_ex,n
  print,'1..n='
  ON_ERROR,n
  print,'a=',a
  print,'n=',n
RETURN
END
```

```
IDL> .reset_session  ;先重新啟動IDL
IDL> .go
% Compiled module: $MAIN$.
main_1
% Compiled module: ON_ERROR_EX.
1..n=
% PRINT: Variable is undefined: A.
% Execution halted at: ON_ERROR_EX       4
H:\300GB138_F\Courses\1061_2017Sept\1061
_IDL\ on_error_ex.pro
%                $MAIN$          2
H:\300GB138_F\Courses\1061_2017Sept\1061
_IDL\ IDL_CH15_on_error_1.pro
IDL>
```

```
print,'main_1'
on_error_ex
print,'main_2
end
```

n未給表示(使用預設值n=0)
讓程式停留在錯誤發生的地方，且列印主副程式目前的堆疊，此為預設值

```
IDL> .reset_session
IDL> .go
% Compiled module: $MAIN$.
main_1
% Compiled module: ON_ERROR_EX.
1..n=
% ON_ERROR: Variable is undefined: N.
% Execution halted at: ON_ERROR_EX       3
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  on_error_ex.pro
%                $MAIN$          2
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  IDL_CH15_on_error_1.pro
IDL>
```

n=1

當錯誤發生時，讓程式停留在主程式的位置，且列印副程式目前的堆疊

```
IDL> .RESet_session
IDL> .go
% Compiled module: $MAIN$.
main_1
% Compiled module: ON_ERROR_EX.
1..n=
% PRINT: Variable is undefined: A.
% Error occurred at: ON_ERROR_EX    4
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\on _error_ex.pro
%   $MAIN$   2 H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
IDL_CH15_on_error_1.pro
% Execution halted at: $MAIN$          2
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  IDL_CH15_on_error_1.pro
IDL>
```

n=2

當錯誤發生時，讓程式停留在主程式的位置，且列印從錯誤發生之副程式至主程式的堆疊

```
IDL> .reset_SESSION
IDL> .go
% Compiled module: $MAIN$.
main_1
% Compiled module: ON_ERROR_EX.
1..n=
% PRINT: Variable is undefined: A.
% Execution halted at: $MAIN$          2 H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  IDL_CH15_on_error_1.pro
IDL>
```

n=3

讓程式停留在錯誤發生的地方，且列印從錯誤發生之副程式至主程式的堆疊

```
.reset_session
IDL> .go
% Compiled module: $MAIN$.
main_1
% Compiled module: ON_ERROR_EX.
1..n=
% PRINT: Variable is undefined: A.
% Execution halted at: ON_ERROR_EX       4
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  on_error_ex.pro
%                $MAIN$          2
H:\300GB138_F\Courses\1061_2017Sept\1061_IDL\
  IDL_CH15_on_error_1.pro
IDL>
```

```
pro on_ioerror_ex
flag=0
print,'1..flag=',flag
WHILE flag EQ 0 DO BEGIN
  ON_IOERROR, ERR
  a=1
  READ,"Enter a number:",a
  flag=1
 ERR:
  IF flag Eq 0 THEN PRINT,'You entered a character'
ENDWHILE
print,'2..flag=',flag
PRINT,a
RETURN
END
```

```
IDL> .reset_session
IDL> on_ioerror_ex
% Compiled module: ON_IOERROR_EX.
1..flag=       0
Enter a number:IDL> a
You entered a character
Enter a number:IDL> b
You entered a character
Enter a number:IDL> 3
2..flag=       1
       3
IDL>
```

```
IDL> .reset_session
IDL> on_ioerror_ex
% Compiled module: ON_IOERROR_EX.
1..flag=       0
Enter a number:IDL> 3
2..flag=       1
       3
IDL>
```

IDL錯誤訊息的發送和查詢

程序或系統變數	說明
MESSAGE	發送特定錯誤訊息
!ERROR_STATE.MSG	此結構欄位記錄系統發生錯誤的訊息

```
IDL> .reset_session
IDL> print,‘1..’,!error_state.msg  ;目前没發生錯誤，所以是空白

1..
IDL> print,var1
% PRINT: Variable is undefined: VAR1.
% Execution halted at: $MAIN$
IDL> print,'2..',!error_state.msg
2..PRINT: Variable is undefined: VAR1.
IDL>
```

```
IDL> message,'error message 1'
% $MAIN$: error message 1
% Execution halted at: $MAIN$
IDL>
```

```
pro message1_ex,n
 CATCH,var1
 IF var1 NE 0 THEN BEGIN
   ;message,'not enough argument'
   PRINT,!ERROR_STATE.msg
   GOTO, err3
 ENDIF
 PRINT,'n=',n
 GOTO,Label1
ERR3:
 print,'err3...'
LABEL1:
END
```

```
IDL> .reset_session
IDL> message1_ex,3
% Compiled module: MESSAGE1_EX.
n=       3
IDL> message1_ex
PRINT: Variable is undefined: N.
err3...
IDL>
```

轉變執行階層的指令

指令	說明
RETURN	回到上一階層
RETALL	回到主程式的階層

```
pro subroutine_1,c

print,'subrountine_1'
print,'c=',c
print,'test...1'
subroutine_2,c,sum0
print,'aft subr_2'
print,'sum0=',sum0
avg_0=nan_average(c)
print,'avg_0=',avg_0
return
end
```

```
pro subroutine_2,c2,sum_a
sum_a=total(c2,/nan)
print,'sum_a=',sum_a
return
end
```

```
a=findgen(5)
subroutine_1,a
print,'after sub_1'
subroutine_3,a,avg_0
print,'after sub_3'
print,'avg_0=',avg_0
aa=nan_average_2(a)
print,'aa=',aa
end
```

```
FUNCTION nan_avgerage,c

newc = c[where(finite(c))]
nn=N_ELEMENTS(newc)
total1=total(newc)
avg0=total1/nn
print,total1,nn
return,avg0
end
```

```
FUNCTION nan_average_2,c

newc = c[where(finite(c))]
nn=N_ELEMENTS(newc)
total1=total(newc)
avg0=total1/nn
print,'total1=',total1,nn
;retall,avg0
return,avg0
end
```

```
IDL> .RESET_SESSION
IDL> .go
% Compiled module: $MAIN$.
% Compiled module: SUBROUTINE_1.
subrountine_1
c=     0.000000      1.00000      2.00000      3.00000
4.00000
test...1
% Compiled module: SUBROUTINE_2.
sum_a=      10.0000
aft subr_2
sum0=      10.0000
% Compiled module: NAN_AVERAGE.
      10.0000           5
avg_0=      2.00000
after sub_1
% Compiled module: SUBROUTINE_3.
subr_3, c=     0.000000      1.00000      2.00000      3.00000
4.00000
% Compiled module: NAN_AVERAGE_2.
total1=      10.0000           5
avg_1=      2.00000
after sub_3
avg_0=      2.00000
total1=      10.0000           5
aa=      2.00000
IDL>
```

```
FUNCTION nan_average_2,c

newc = c[where(finite(c))]
nn=N_ELEMENTS(newc)
total1=total(newc)
avg0=total1/nn
print,'total1=',total1,nn
retall,avg0
;return,avg0
end
```

```
IDL> .reset_session
IDL> .go
% Compiled module: $MAIN$.
% Compiled module: SUBROUTINE_1.
subrountine_1
c=    0.000000     1.00000     2.00000     3.00000     4.00000
test...1
% Compiled module: SUBROUTINE_2.
sum_a=      10.0000
aft subr_2
sum0=      10.0000
% Compiled module: NAN_AVERAGE.
      10.0000           5
avg_0=      2.00000
after sub_1
% Compiled module: SUBROUTINE_3.
subr_3, c=    0.000000     1.00000     2.00000     3.00000     4
% Compiled module: NAN_AVERAGE_2.
total1=      10.0000           5
IDL>
```

```
function average1,c
nn=N_ELEMENTS(c)
total1=total(c)
avg0=total1/nn
print,total1,nn
return,avg0
end
```

```
FUNCTION nan_average,c

newc = c[where(finite(c))]
nn=N_ELEMENTS(newc)
total1=total(newc)
avg0=total1/nn
print,total1,nn
return,avg0
end
```

```
a=findgen(5)
print,'a=',a
avg_0=average1(a)
print,'avg_0=',avg_0
print,'sum_0=',total(a)
print,'mean_0=',mean(a)
b=a
b[0]='nan'
print,'b=',b
avg_1=average1(b)
print,'avg_1=',avg_1
print,'sum_1=',total(b)
avg_2=NAN_AVERAGE(b)
sum_2=total(b,/nan)
print,'avg_2=',avg_2
print,'sum_2=',sum_2
print,'nan_mean_2=',mean(b,/nan)
end
```