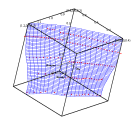
XploRe Course - Day 4

Uwe Ziegenhagen
Sigbert Klinke

Institut für Statistik and Ökonometrie
Humboldt-Universität zu Berlin
http://ise.wiwi.hu-berlin.de

Schedule for Today

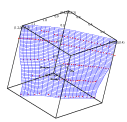
1. XploRe Function Basics
2. Local and Global Variables
3. Branching and Looping
4. Errors and Warnings
5. Optional Input and parameters
6. APSS templates



XploRe Function Basics

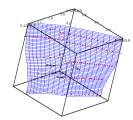
```
1 proc() = myquant()
2           ; empty program
3 endp
4 myquant()
```

```
1 proc(result) = myquant(input1,input2)
2     ; store as myquant.xpl
3     result = input1 + input2
4     ; just calculate something
5 endp
6 myquant(2,2) ; call the function
```

Loading Quantlets with `func()`

- to make a quantlet available from other quantlets it has to be loaded/executed first
- manually: load quantlet and execute it
- automatically: use `func("quantletname")`

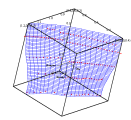


`func()` Exercise

1. write one quantlet which multiplies two numbers
2. write another quantlet which adds two numbers that have been squared with quantlet (1)
3. use `func()` to call quantlet 1 from quantlet 2

Alternative to func()

Alternatively a text-file can be created with the quantlet names and stored with the extension *.lib in the lib directory of the XploRe installation, see the examples there.

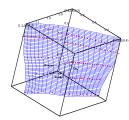


Local and Global Variables

XploRe strictly distinguishes between globally and locally defined variables:

- **global** variables are defined in the main program
- **local** variables defined in a quantlet

```
1 proc() = glcheck() ; define "glcheck"
2          localvar = 5
3 endp                ; end of the procedure
4 glcheck()
5 localvar ; try to access => failure
```

Branching and Looping

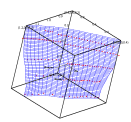
- if condition is fulfilled do something
- if condition is fulfilled do something or do something else
- repeat something until condition is fulfilled
- while condition do something

for-Loop

```
for (int i=1;i<=5;i++){
 // do something
}
```

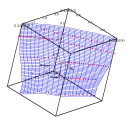
while-Loop

```
while(j<=x)
// do something
j=j+1
endo
```

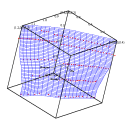
If and Else I

```
1 proc() = ifcheck(x) ; define "ifcheck"
2     if (x>0)
3                   sqrt(x)
4     endif
5 endp                ; end of the procedure
6 ifcheck (5)
7 ifcheck(-5)
```

If and Else II

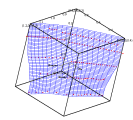
```
1 proc() = ifcheck(x) ; define "ifcheck"
2     if (x>0)
3             sqrt(x)
4     else
5             sqrt(abs(x))
6     endif
7 endp                ; end of the procedure
8 ifcheck (5)
9 ifcheck(-5)
```

IF Exercise

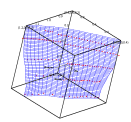
Write one quantlet which accepts one parameter k and gives

- "negative" for negative input
- "is zero" for $p = 0$
- "smaller than 5" for $p < 5$
- "smaller than 10" for $p < 10$
- "bigger than 10" for $p > 10$



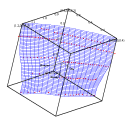
IF Exercise

```
1  proc () = ifcheck (x) ; define " ifcheck "
2   if (x<0) "negative"
3   endif
4   if (x==0) "is zero"
5   endif
6   if (x<5) "smaller 5"
7   endif
8   if (x<10) "smaller 10"
9   endif
10  if (x>10) "bigger 10"
11  endif
12  endp ; end of the procedure
13  ifcheck (5) ; try different values!
```

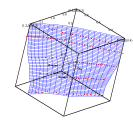
The While Loop

```
1  proc(j) = factorial(x) ; define "factorial"
2    j = 1                ; define variable j as 1
3    while (x >= 2)       ; as long as this condition
4                         ; is fulfilled,
5                         ; XploRe executes the following
                            commands
6    j = j * x            ; computes j as product of j and x
7    x = x - 1            ; reduces x by 1
8    endo                 ; end of the while loop
9  endp                   ; end of the procedure
10 factorial(5)
```

`While` Exercise I

1. write one quantlet which accepts one positive number n
2. outputs the numbers 1 to n as scalars

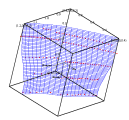


While Exercise I

```
1 ; working
2 proc()=loopi(x)
3 j=1
4      while(j<=x)
5      j
6      j=j+1
7      endo
8 endp
9 loopi(5)
```

```
1 ; not working
2 proc(j)=loopi2(x)
3 j=1
4 while(j<=x)
5 j
6 j=j+1
7 endo
8 endp
9 loopi2(5)
```

The right version gives also number $x + 1$. In the final `while` loop j is printed and increased afterwards. This increased j then has the value $x + 1$ and is printed because j is also used as return parameter.

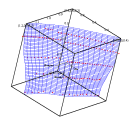


`While` Loop Example

Problem

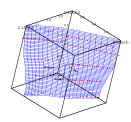
A man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair gets a new pair which from the second month on becomes productive?

- Leonardo Pisano (Fibonacci), 1170-1240
- $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$
- Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
- made arabic numbers popular in Europe
- ratio of succeding numbers approaches the *Golden Ratio*
- leafs on a stem, etc.



While Loop Example

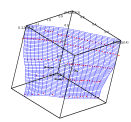
```
1  proc(a)=Fibonacchi(x)
2       a=0
3       b=1
4       while (x>=2)
5            c=a+b
6            a=b
7            b=c
8            x=x-1
9       endo
10      a=b
11 endp
12 Fibonacchi(3)
13 Fibonacchi(25)
```

Fibonacci Recursive Version

$$\text{fibo}(n) = \begin{cases} 1 & n \leq 2 \\ \text{fibo}(n-1) + \text{fibo}(n-2) & \text{else} \end{cases}$$

```
1 proc(a)= fibo(n)
2 if (n<=2)
3       a= 1
4 else
5       a = fibo(n-1)+fibo(n-2)
6 endif
7 endp
8 fibo(15)
```

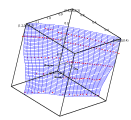



`While` Exercise II

Write a quantlet which:

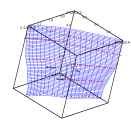
1. takes a number n as parameter,
2. generates a $n \times n$ matrix of zeros
3. fills the columns of the matrix with values from 1 to n

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$



While Exercise II

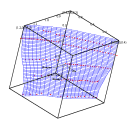
```
1  proc() = dosome(x)
2  data = zeros(x,x)
3  i = 0
4  j = 1
5  while(i<x)
6     i=i+1
7     while(j<=x)
8        data[i,j]=i
9              j=j+1
10       endo
11      j=1
12    endo
13 data
14 endp
15 dosome(5)
```

Alternative Solutions

```
proc()=scal(x)
; using .* operator
     matrix(x,x) .* aseq(1,x,1)
endp
scal(7)
```

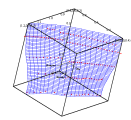
```
proc()=scal(x) ; L. Rohrschneider
     b=vec(1:x)
     c=unit(x)
     d=(b+c)-unit(x)
     d
endp
scal(20)
```

`While` Exercise II

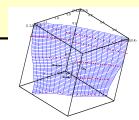
Write a quantlet which generates a triangular $n \times n$ matrix for a given n:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



Solution with two `While` Loops

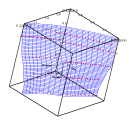
```
1  proc() = dotriag(x)
2  data = unit(x)
3  i=2
4  j=2
5  while (i<=x)
6           data[i,i-1] = 1
7           i = i+ 1
8  endo
9  i=1
10 while (i<x)
11          data[i,i+1] = 1
12          i = i+ 1
13 endo
14 data
15 endp
16 dotriag(10)
```



A Nice One-Loop-Solution (O. Chochola)

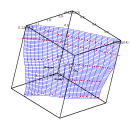
```
1  proc()=triag(n)
2       v=zeros(n,n+2)
3       i=1
4       while(i<=n)
5            v[i,i:i+2]=1
6            i=i+1
7       endo
8       v=v[,2:n+1]
9  endp
10 triag(10)
```

```
1 1 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 1 1
```

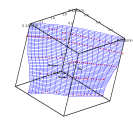
The Do-Until Loop

```
1   proc(j) = factorial(x)  ; define "factorial"
2     j = 1                 ; define variable j as 1
3     do                    ; opens the do loop
4       j = j *x            ; computes j as the product
          of j and x
5       x = x -1            ; reduces x by 1
6     until (x < 2)         ; if the condition is not
        fulfilled,
7                           ; the loop will be run
                              again
8   endp                    ; end of the procedure
9 factorial(5)
```

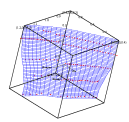
Difference of `While` and `Do-Until`

- **while** checks the condition first, then does something (or not)
- **do-until** does first, then checks the condition



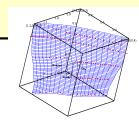
The Switch-Statement

```
1     switch              ; opens the switch branch
2       case  <some boolean expression>
3                         ; do something
4         break           ; finish branch
5       case  <some boolean expression>
6                         ; do something else
7         break
8       default
9                         ; do the default commands
10        break
11    endsw               ; end switch
```

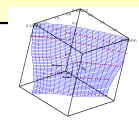
Switch-Statement Example

```
1  proc() = signum(x)   ; define procedure "signum"
2      switch           ; opens the switch branch
3        case (x > 0)
4          "positive number" ; output if x > 0
5          break
6        case (x < 0)
7          "negative number" ; output if x < 0
8          break
9        default
10         "number is zero"  ; output if x = 0
11         break
12     endsw                 ; end of switch
13   endp                    ; end of procedure
14 signum(-2)
15 signum(0)
16 signum(2)
```



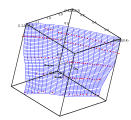
Alternative Solution

```
1  proc(res)=iff(k)
2    if(k<0)          "negative"
3    else
4      if(k==0)       "zero"
5      else
6        if(k<5)      "smaller 5"
7        else
8          if(k<10)   "smaller 10"
9          else
10                    "bigger or equal 10"
11         endif
12       endif
13     endif
14   endif
15 endp
16 iff(4)
```



Optional Inputs with `exist()`

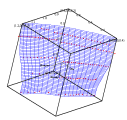
```
exist("x")
; -1 if x is undefined
;  0 if empty
;  1 if existing and numeric (scalar or vector)
;  4 for display, 9 for list,
; 10 for quantlet, >10 for exists as
; quantlet and variable
```

Errors and Warnings

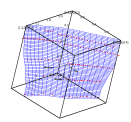
```
x=matrix(2,2)
x
warning(rows(x)<5, "rows of x less than 5")
2
error(rows(x)<5, "rows of x less than 5")
3
```

```
proc() = quad(x)
    error((rows(x)>1)||(cols(x)>1),"not a scalar")
    x^2
endp
quad(5)
```

Optional Inputs with `exist()`

```
1 proc(result) = addiere(x,y)
2     error((exist("x")<>1 || exist("y")<>1),"At least
         one arg not numeric")
3     x+y
4 endp
5 addiere(3,2) ; try addiere("abc",2)
```

APSS Templates

```
; Library     => name of the library
; See_also    => which other commands are relevant for this command
; Macro       aaa => name of the quantlet
; Description => what does it do
; Usage       aaa(a1,a2)
; Input
; Parameter   a1
; Definition
; Parameter   a2
; Definition
; Output      => a vector or matrix?
; Notes       => some hints for usage
; Example     => always provide an example
; Result      => and the result for the example
; Keywords    => what should appear in the list of keywords
; Reference   => a book or article
; Link        => some URL you may refer to
; Author      => your name
```