```
library("metrics")
randomize(10178)
z=(uniform(n).>0.5)~(normal(n).<0.5)
```

```
proc() = myquant()
; empty program
endp
myquant()
```

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

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

```
proc() = ifcheck(x) ; define "ifcheck"
    if (x>0)
                sqrt(x)
    endif
endp              ; end of the procedure
ifcheck (5)
ifcheck(-5)
```

```
proc() = ifcheck(x) ; define "ifcheck"
    if (x>0)
          sqrt(x)
    else
          sqrt(abs(x))
    endif
endp              ; end of the procedure
ifcheck (5)
ifcheck(-5)
```

```
proc () = ifcheck (x) ; define " ifcheck "
 if (x<0) "negative"
 endif
 if (x==0) "is zero"
 endif
 if (x<5) "smaller 5"
 endif
 if (x<10) "smaller 10"
 endif
 if (x>10) "bigger 10"
 endif
 endp ; end of the procedure
 ifcheck (5) ; try different values!
```

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

```
; not working
proc(j)=loopi2(x)
j=1
while(j<=x)
j
j=j+1
endo
endp
loopi2(5)
```

```
proc(a)=Fibonacchi(x)
    a=0
    b=1
    while (x>=2)
        c=a+b
        a=b
        b=c
        x=x-1
    endo
    a=b
endp
Fibonacchi(3)
Fibonacchi(25)
```

```
proc(a)= fibo(n)
if (n<=2)
      a= 1
else
      a = fibo(n-1)+fibo(n-2)
endif
endp
fibo(15)
```

```
proc() = dosome(x)
data = zeros(x,x)
i = 0
j = 1
while(i<x)
   i=i+1
   while(j<=x)
      data[i,j]=i
            j=j+1
      endo
```

```
      j=1
  endo
data
endp
dosome(5)
```

```
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)
```

```
proc() = dotriag(x)
data = unit(x)
i=2
j=2
while (i<=x)
        data[i,i-1] = 1
        i = i+ 1
endo
i=1
while (i<x)
        data[i,i+1] = 1
        i = i+ 1
endo
data
endp
dotriag(10)
```

```
proc()=triag(n)
    v=zeros(n,n+2)
    i=1
    while(i<=n)
        v[i,i:i+2]=1
        i=i+1
    endo
    v=v[,2:n+1]
endp
triag(10)
```

```
  proc(j) = factorial(x) ; define "factorial"
    j = 1                ; define variable j as 1
    do                   ; opens the do loop
      j = j *x           ; computes j as the product of j and x
      x = x -1           ; reduces x by 1
    until (x < 2)        ; if the condition is not fulfilled,
```

```
                            ; the loop will be run again
  endp                      ; end of the procedure
factorial(5)
```