```
> maxi := proc( )
    local r, i;
    r := args[1];
    for i from 2 to nargs do
        if r < args[i] then r := args[i] fi
    od;
    r;
    end
```

```
maxi := proc( )
    local r, i;
    r := args[1]; for i from 2 to nargs do if r < args[i] then r := args[i] end if end do; r
end proc
```

(1)

```
> maxi(1, 4, 2, 3, 19, 2, 15)
```

19 (2)

```
> divise := proc(a, b)
    local q;
    q := 0;
    while a - b·q ≥ b do
        q := q + 1
    od;
    q, a - b·q
    end
```

```
divise := proc(a, b)
    local q; q := 0; while b <= a - b*q do q := q + 1 end do; q, a - b*q
end proc
```

(3)

```
> divise(101, 3)
```

33, 2 (4)

```
> modulo := proc(a, b)
    local q, r;
    q, r := divise(a, b);
    r
    end
```

modulo := **proc**(*a*, *b*) **local** *q*, *r*; *q*, *r* := *divise*(*a*, *b*); *r* **end proc** (5)

```
> modulo(101, 9)
```

2 (6)

```
> with(ArrayTools)
```

[*AddAlongDimension*, *Alias*, *AllNonZero*, *AnyNonZeros*, *Append*, *BlockCopy*, *CircularShift*, *ComplexAsFloat*, *Compress*, *Concatenate*, *Copy*, *DataTranspose*, *Diagonal*, *Dimensions*, *ElementDivide*, *ElementMultiply*, *ElementPower*, *Extend*, *Fill*, *FlipDimension*, *GeneralInnerProduct*, *GeneralOuterProduct*, *HasNonZero*, *HasZero*, *Insert*, *IsEqual*, *IsMonotonic*, *IsZero*, *Lookup*, *LowerTriangle*, *MultiplyAlongDimension*, *NumElems*, *Partition*, *Permute*, *PermuteInverse*, *RandomArray*, *ReduceAlongDimension*, *RegularArray*, *Remove*, *RemoveSingletonDimensions*, *Replicate*, *Reshape*, *Reverse*, *ScanAlongDimension*, *SearchArray*, (7)

```
    Size, SuggestedDatatype, SuggestedOrder, SuggestedSubtype, Uncompress, UpperTriangle ]
> init_rand := proc(n)
    local i, T;
    T := Array(1..n);
    for i from 1 to n do
        T[i] := rand(0..100)( )
    od;
    T
    end
```

$$\mathit{init\_rand} := \mathbf{proc}(n) \quad \mathbf{local}\ i, T;\ T := \mathit{Array}(1..n);\ \mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}\ T[i] := \mathit{rand}(0..100)()\ \mathbf{end\ do};\ T\ \mathbf{end\ proc} \tag{8}$$

```
> init_rand(10)
```

$$\left[\begin{array}{cccccccccc} 92 & 44 & 95 & 5 & 97 & 58 & 43 & 99 & 37 & 68 \end{array}\right] \tag{9}$$

```
> moy := proc(T)
    local n, S, i;
    n := Size(T)[2];
    S := 0 :
    for i from 1 to n do
        S := S + T[i]
    od;
    S/n
    end
```

$$\mathit{moy} := \mathbf{proc}(T) \quad \mathbf{local}\ n, S, i;\ n := \mathit{ArrayTools}\text{:-}\mathit{Size}(T)[2];\ S := 0;\ \mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}\ S := S + T[i]\ \mathbf{end\ do};\ S/n\ \mathbf{end\ proc} \tag{10}$$

```
> T := init_rand(10)
```

$$T := \left[\begin{array}{cccccccccc} 58 & 15 & 0 & 69 & 76 & 38 & 91 & 70 & 66 & 77 \end{array}\right] \tag{11}$$

```
> moy(T)
```

$$56 \tag{12}$$

```
> max_tab := proc(T)
    local n, i, r;
    n := Size(T)[2];
    r := T[1];
    for i from 2 to n do
        if r < T[i] then r := T[i] fi
    od;
    r
    end
```

$$\mathit{max\_tab} := \mathbf{proc}(T) \quad \mathbf{local}\ n, i, r; \tag{13}$$

```
    n := ArrayTools:-Size(T)[2];
    r := T[1];
    for i from 2 to n do if r < T[i] then r := T[i] end if end do;
    r
end proc
```

```
> T := init_rand(10)
```

$$T := \left[\begin{array}{cccccccccc} 72 & 84 & 40 & 3 & 43 & 12 & 18 & 9 & 14 & 63 \end{array}\right] \quad (14)$$

```
> max_tab(T)
```

$$84 \quad (15)$$

```
> somme :=proc(T1, T2)
    # On suppose les deux tableaux de même taille
    local T, n, i;
    n := Size(T1)[2];
    T := Array(1..n);
    for i from 1 to n do
        T[i] := T1[i] + T2[i]
    od;
    T
    end
somme := proc(T1, T2)                                                    (16)
    local T, n, i;
    n := ArrayTools:-Size(T1)[2];
    T := Array(1..n);
    for i to n do T[i] := T1[i] + T2[i] end do;
    T
end proc
```

```
> T1 := init_rand(10); T2 := init_rand(10)
```

$$T1 := \left[\begin{array}{cccccccccc} 8 & 11 & 51 & 24 & 71 & 89 & 17 & 42 & 54 & 39 \end{array}\right]$$

$$T2 := \left[\begin{array}{cccccccccc} 16 & 69 & 51 & 80 & 86 & 33 & 84 & 8 & 67 & 82 \end{array}\right] \quad (17)$$

```
> somme(T1, T2)
```

$$\left[\begin{array}{cccccccccc} 24 & 80 & 102 & 104 & 157 & 122 & 101 & 50 & 121 & 121 \end{array}\right] \quad (18)$$

```
> tri_bulle :=proc(T)
    local n, i, j, tmp;
    n := Size(T)[2];
    print(T);
    for i from n by -1 to 1 do
        for j from 1 to i - 1 do
            if T[j + 1] < T[j] then
                tmp := T[j];
                T[j] := T[j + 1];
                T[j + 1] := tmp;
```

```
                    print(T)
                fi;
            od
        od
        end
```

```
tri_bulle := proc(T)                                                    (19)
    local n, i, j, tmp;
    n := ArrayTools:-Size(T)[2];
    print(T);
    for i from n by -1 to 1 do
        for j to i - 1 do
            if T[j + 1] < T[j] then
                tmp := T[j]; T[j] := T[j + 1]; T[j + 1] := tmp; print(T)
            end if
        end do
    end do
end proc
```

```
> T := init_rand(5)
```

$$T := \begin{bmatrix} 77 & 22 & 72 & 21 & 31 \end{bmatrix} \quad (20)$$

```
> tri_bulle(T)
```

$$\begin{bmatrix} 77 & 22 & 72 & 21 & 31 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 77 & 72 & 21 & 31 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 72 & 77 & 21 & 31 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 72 & 21 & 77 & 31 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 72 & 21 & 31 & 77 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 21 & 72 & 31 & 77 \end{bmatrix}$$
$$\begin{bmatrix} 22 & 21 & 31 & 72 & 77 \end{bmatrix}$$
$$\begin{bmatrix} 21 & 22 & 31 & 72 & 77 \end{bmatrix} \quad (21)$$

```
>
```