



Haute École de Bruxelles  
École Supérieure d'Informatique

Rue Royale 67 – 1000 Bruxelles  
02/219.15.46 – esi@heb.be

# Logique & Techniques de programmation

Bachelor en Informatique – 1<sup>ère</sup> année

Cours enseigné par :

<i>L. Beeckmans</i>	<i>M. Codutti</i>	<i>G. Cuvelier</i>	<i>J. Dossogne</i>	<i>A. Hallal</i>
<i>C. Leruste</i>	<i>E. Levy</i>	<i>N. Pettiaux</i>	<i>F. Servais</i>	<i>W. Willame</i>

Ce syllabus a été écrit à l'origine par M. Monbaliu. Il a ensuite été adapté par Mme Leruste, M. Beeckmans et M. Codutti. Qu'ils en soient tous remerciés. Nous remercions également tout ceux qui ont contribué à son amélioration grâce à leur lecture attentive et leurs remarques.

Document produit avec L<sup>A</sup>T<sub>E</sub>X.  
Version du 16 octobre 2013.

**Ce syllabus couvre la matière du premier quadrimestre  
(jusque fin décembre).  
La suite sera diffusée en janvier.**



Ce document est distribué sous licence  
Creative Commons Paternité - Partage à l'Identique 2.0 Belgique  
(<http://creativecommons.org/licenses/by-sa/2.0/be/>).  
Les autorisations au-delà du champ de cette licence  
peuvent être obtenues à [www.heb.be/esi](http://www.heb.be/esi) - [mcodutti@heb.be](mailto:mcodutti@heb.be).

# Table des matières

# Correction des exercices 4.4

## Simplification d'algorithme

```
si ok alors
  afficher nombre
fin si
```

ok  $\leftarrow$  condition

```
si NON ok alors
  afficher nombre
fin si
```

ok  $\leftarrow a \geq b$

```
si ok1 ET ok2 alors
  afficher x
fin si
```

## Exercice 3 – Maximum de 2 nombres

```
module max2Nb()
  nb1, nb2 : entiers
  max : entier
  lire nb1, nb2
  si nb2  $\geq$  nb1 alors
    max  $\leftarrow$  nb2
  sinon
    max  $\leftarrow$  nb1
  fin si
  afficher max
fin module
```

## Exercice 4 – Maximum de 3 nombres

```
module max3Nb()
  nb1, nb2, nb3 : entiers
  max : entier
  lire nb1, nb2, nb3
  si nb2  $\geq$  nb1 alors
    max  $\leftarrow$  nb2
  sinon
    max  $\leftarrow$  nb1
  fin si
  si nb3  $\geq$  max alors
    max  $\leftarrow$  nb3
  fin si
  afficher max
fin module
```

### Exercice 5 – Signe

```
module signe()
  nb : entier
  lire nb
  selon que
    nb > 0 : afficher "positif"
    nb < 0 : afficher "négatif"
    autre : afficher "nul"
  fin selon que
fin module
```

### Exercice 6 – La fourchette

```
module fourchette()
  nb1, nb2, nb3, petit, grand : entiers
  ok : booléen
  lire nb1, nb2, nb3
  si nb2 > nb3 alors
    petit ← nb3
    grand ← nb2
  sinon
    petit ← nb2
    grand ← nb3
  fin si
  afficher ← nb1 > petit ET nb1 < grand
fin module
```

### Exercice 7 – Équation du second degré

```
module racinesÉquation()
  coeffCarré, coeff, termeIndé : entiers
  delta : entier
  lire coeffCarré, coeff, termeIndé
  delta ← (coeff)2 - 4 * coeffCarré * termeIndé
  selon que
    delta > 0 : afficher (-coeff ± √delta)/(2 * coeffCarré)
    delta = 0 : afficher -coeff/(2 * coeffCarré)
    autres : afficher "pas de racine"
  fin selon que
fin module
```

### Exercice 8 – Une petite minute

```
module plusUneMin()
    heure, minute : entiers
    lire heure, minute
    si minute = 59 alors
        minute ← 0
        heure ← heure + 1
    sinon
        minute ← minute + 1
    fin si
    afficher heure, minute
fin module
```

### Exercice 9 – Calcul de salaire

```
module salaireNet()
    salaireBrut : entier
    constante RETENUE : 15
    salaireNet : entier
    lire salaire
    si salaire > 1200 alors
        salaireNet ← salaire - (salaire * RETENUE) / 100
        afficher salaireNet
    sinon
        afficher salaireBrut
    fin si
fin module
```

### Exercice 10 – Nombres de jours dans un mois

```
module nbJours()
    mois : chaine
    jours : entier
    lire mois
    selon que mois vaut
        "JANVIER", "MARS", "MAI", "JUILLET", "AOÛT", "OCTOBRE", "DÉCEMBRE":
            afficher 31
        "AVRIL", "JUIN", "SEPTEMBRE", "NOVEMBRE":
            afficher 30
        "FÉVRIER":
            afficher 28
    fin selon que
fin module
```

### Exercice 11 – Année bissextile

```
module estBissextile()  
  annee : entier  
  lire annee  
  afficher annee MOD 4 = 0 ET NON(annee MOD 100 = 0) OU annee MOD 400 = 0  
fin module
```

### Exercice 12 – Valider une date

```
module dateValide()  
  annee, mois, jour, jourMois : entiers  
  bissextile : booléen  
  lire jour, mois, annee  
  bissextile  $\leftarrow$  annee MOD 4 = 0 ET annee MOD 100  $\neq$  0 OU annee MOD 400 = 0  
  selon que mois vaut  
    1, 3, 5, 7, 8, 10, 12:  
      jourMois  $\leftarrow$  31  
    4, 6, 9, 11:  
      jourMois  $\leftarrow$  30  
    2:  
      si bissextile alors  
        jourMois  $\leftarrow$  29  
      sinon  
        jourMois  $\leftarrow$  28  
      fin si  
    autres :  
      afficher "mois inconnu"  
  fin selon que  
  afficher  $1 \leq \text{jour} \leq \text{jourMois}$   
fin module
```

### Exercice 13 – Le jour de la semaine

```
module jourSemaine()  
  dateMois : entier  
  lire dateMois  
  selon que dateMois MOD 7 vaut  
    0 : afficher "vendredi"  
    1 : afficher "samedi"  
    2 : afficher "dimanche"  
    3 : afficher "lundi"  
    4 : afficher "mardi"  
    5 : afficher "mercredi"  
    6 : afficher "jeudi"  
  fin selon que  
fin module
```

#### Exercice 14 – Quel jour serons-nous ?

```
module jourFutur()  
  jour : chaîne  
  n, jourFutur : entier  
  lire jour, n  
  selon que jour vaut  
    "lundi" : jourFutur ← 1  
    "mardi" : jourFutur ← 2  
    "mercredi" : jourFutur ← 3  
    "jeudi" : jourFutur ← 4  
    "vendredi" : jourFutur ← 5  
    "samedi" : jourFutur ← 6  
    "dimanche" : jourFutur ← 7  
  fin selon que  
  selon que (jourFutur + n) MOD 7 vaut  
    0 : afficher "lundi"  
    1 : afficher "mardi"  
    2 : afficher "mercredi"  
    3 : afficher "jeudi"  
    4 : afficher "vendredi"  
    5 : afficher "samedi"  
    6 : afficher "dimanche"  
  fin selon que  
fin module
```

#### Exercice 15 – Un peu de trigono

```
module cosinus()  
  n : entier  
  cosinus : entier  
  lire n  
  si NON(n MOD 2 = 0) alors  
    cosinus ← 0  
  sinon  
    si (n/2) MOD 2 = 0 alors  
      cosinus ← 1  
    sinon  
      cosinus ← -1  
    fin si  
  fin si  
  afficher cosinus  
fin module
```



---

**Exercice 16 – Le stationnement alternatif**

```
module bienStationné()  
  dateJour, numMaison : entiers  
  si  $1 \geq \text{dateJour} \geq 15$  alors  
    afficher NON(numMaison MOD 2 = 0)  
  sinon  
    afficher numMaison MOD 2 = 0  
  fin si  
fin module
```

## Correction des exercices 5.8

### Exercice 4 – Échange de variables

```
module swap(a↓↑,b↓↑ : entiers)
  a ← a + b
  b ← a - b
  a ← a - b
fin module
```

### Exercice 5 – Valeur absolue

```
module abs(a : réel) → réel
  si a < 0 alors
    a ← -a
  fin si
  retourner a
fin module
```

### Exercice 6 – Maximum de 4 nombres

```
module max4Nb(a↓,b↓,c↓,d↓ : entiers) → entier
  max : entier
  max ← max2(max3(a,b,c), d)
  retourner max
fin module
```

### Exercice 7 – Validité d'une date

```
module dateValide(jour, mois, annee : entiers) → booléen
| retourner 1 ≤ jour ≤ nbJours(mois, annee)
fin module
module
module estBissextile(annee : entier) → booléen
| retourner annee MOD 4 = 0 ET NON(annee MOD 100 = 0) OU annee MOD 400 = 0
fin module
module nbJours(mois, annee : entiers) → entier
| selon que mois vaut
| 1, 3, 5, 7, 8, 10, 12:
| | retourner 31
| 4, 6, 9, 11:
| | retourner 30
| 2:
| | si estBissextile(annee) alors
| | | retourner 29
| | sinon
| | | retourner 28
| | fin si
| autres:
| | erreur "mois inconnu"
| fin selon que
fin module
```

### Exercice 7 – Valider une date

```
module dateValide(jour, mois, annee : entiers) → booléen
| jourMois : entier
| bissextile : booléen
| selon que mois vaut
| 1, 3, 5, 7, 8, 10, 12:
| | jourMois ← 31
| 4, 6, 9, 11:
| | jourMois ← 30
| 2:
| | bissextile ← annee MOD 4 = 0 ET annee MOD 100 <> 0 OU annee MOD 400 = 0
| | si bissextile alors
| | | jourMois ← 29
| | sinon
| | | jourMois ← 28
| | fin si
| autres :
| | erreur "mois inconnu"
| fin selon que
| retourner 1 ≤ jour ≤ jourMois
fin module
```

## Correction des exercices 6.6

### Structure – Moment

```
structure Moment
| heure : entier
| minute : entier
| seconde : entier
fin structure
```

### Exercice 1 – Conversion moment-secondes

```
module secondeMinuit(moment↓ : Moment) → entier
| retourner moment.heure * 3600 + moment.minute * 60 + moment.seconde
fin module
```

### Exercice 2 – Conversion secondes-moment

```
module secondesVersMoments(secondes↓ : entier) → Moment
| moment : Moment
| moment ←  $\left\{ \frac{\text{secondes}}{3600}, \frac{(\text{secondes} \bmod 3600)}{60}, \text{secondes} \bmod 60 \right\}$ 
| retourner moment
fin module
```

### Exercice 3 – Temps écoulé entre 2 moments

```
module écartEntreMoments(moment1, moment2 : Moments) → entier
| retourner SecondesMinuit(moment1) - SecondesMinuit(moment2)
fin module
```

#### Exercice 4 – Milieu de deux points

```
module MilieuSegment(a, b : Points) → Point
|   retourner  $\left\{ \frac{a.x + b.x}{2}, \frac{a.y + b.y}{2} \right\}$ 
fin module
```

#### Exercice 5 – Distance entre deux points

```
module LongueurSegment(a, b : Points) → entier
|   retourner  $\sqrt{(b.x - a.x)^2 + (b.y - a.y)^2}$ 
fin module
```

#### Exercice 6 – Un cercle

```
structure Cercle
|   centre : Point
|   rayon : réel
fin structure

module SurfaceCercle(cercle : Cercle) → réel
|   retourner  $\pi * (\text{cercle.rayon})^2$ 
fin module

module CréeCercle(a, b : Points) → Cercle
|   cercle : Cercle
|   cercle.centre ← a
|   cercle.rayon ← LongueurSegment(a,b)
|   retourner cercle
fin module

module pointDansCercle(point : Point, cercle : Cercle) → booléen
|   retourner  $\sqrt{(\text{point.x} - \text{cercle.centre.x})^2 + (\text{cercle.centre.y} - \text{point.y})^2} < \text{cercle.rayon}$ 
fin module

module IntersectionCercle(cercle1, cercle2 : Cercles) → booléen
|   distanCentre : réel
|   distanceCentre ← LongueurSegment(cercle1.centre, cercle2.centre)
|   retourner distanceCentre - (cercle1.rayon * 2 - cercle2.rayon * 2) ≤ 0
fin module
```

## Exercice 7 – Un rectangle

```
structure Rectangle
|   bg : Point
|   hd : Point
fin structure

module périmètreRectangle(rect : Rectangle) → réel
|   base, hauteur : réels
|   base ← LongueurSegment(rect.bg.x, rect.hd.x)
|   hauteur ← LongueurSegment(rect.bg.y, rect.hd.y)
|   retourner 2 * base + 2 * hauteur
fin module

module aireRectangle(rect : Rectangle) → réel
|   base, hauteur : réels
|   base ← LongueurSegment(rect.bg.x, rect.hd.x)
|   hauteur ← LongueurSegment(rect.bg.y, rect.hd.y)
|   retourner base * hauteur
fin module

module pointDansRectangle(rect : Rectangle, point : Point) → booléen
|   retourner (rect.bg.x ≤ point.x ≤ rect.hd.x) ET (rect.bg.y ≤ point.y ≤ rect.hd.y)
fin module

module pointSurSegment(rect : Rectangle, point : Point) → booléen
|   si rect.bg.x ≤ point.x ≤ rect.hd.x ET rect.bg.y = point.y alors
|       retourner vrai rect.bg.y ≤ point.y ≤ rect.hd.y ET rect.bg.x = point.x
|   fin si
fin module

module pointSurBordRectangle(rect : Rectangle, point : Point) → booléen
|   surBaseBas, surBaseHaut, surHauteurGauche, surHauteurDroit : booléens
|   surBaseBas ← rect.bg.x ≤ point.x ≤ rect.hd.x ET (rect.bg.y = point.y)
|   surBaseHaut ← rect.bd.x ≤ point.x ≤ rect.hd.x ET (rect.hd.y = point.y)
|   surCôtéGauche ← rect.bg.y ≤ point.y ≤ rect.hd.y ET (rect.bg.x = point.x)
|   surCôtéDroit ← rect.bg.y ≤ point.y ≤ rect.hd.y ET (rect.hd.x = point.x)
|   retourner surBaseBas OU surBaseHaut OU surCôtéGauche OU surCôtéDroit
fin module
```

## Correction des exercices 7.4

### Exercice 3 – Afficher les $n$ premiers

```
module strictementPositifs()
  n, i : entiers
  lire n
  pour i de 0 à n faire
    afficher i
  fin pour
fin module

module strictementPositifsDécroissants()
  n, i : entiers
  lire n
  pour i de n à 0 par -1 faire
    afficher i
  fin pour
fin module

module carrésParfaits()
  n, i : entiers
  lire n
  pour i de 0 à n faire
    afficher  $i^2$ 
  fin pour
fin module

module naturelsImpairs()
  n, i : entiers
  lire n
  pour i de 1 à n * 2 par 2 faire
    afficher i
  fin pour
fin module

module naturelsImpairsInférieurs()
  n, i : entiers
  lire n
  pour i de 1 à n/2 par 2 faire
    afficher i
  fin pour
```

#### Exercice 4 – Maximum de nombres

```
module maxCote()  
  cote, max : entier  
  max ← 0  
  faire  
    lire cote  
    si cote > max alors  
      max ← cote  
    fin si  
  jusqu'à ce que cote = -1  
  afficher max  
fin module
```

#### Exercice 5 – Afficher les multiples de 3

```
module multiplesDe3()  
  nombre, multiple3 : entier  
  faire  
    lire nombre  
    si nombre MOD 3 = 0 alors  
      afficher nombre  
      multiple3 ← multiple3 + 1  
    fin si  
  jusqu'à ce que nombre = 0  
  afficher multiple3  
fin module
```

#### Exercice 6 – Placement d'un capital

```
module placementCapital()  
  capitalDépart, nbAnnées, tauxPlacement, capitalIntérêt : entiers  
  lire capitalDépart, nbAnnées, tauxPlacement  
  pour année de 2013 à n faire  
    capitalIntérêt ← capitalDépart + (tauxPlacement * capitalDépart)/100  
    afficher année, capitalDépart, capitalIntérêt - capitalDépart  
  fin pour  
fin module
```



### Exercice 7 – Produit de 2 nombres

```
module produit2Nb()  
  nb1, nb2 : entiers  
  lire nb1, nb2  
  retourner nb1 / (1/nb2)  
fin module
```

### Exercice 8 – Génération de suites (1/2)

```
module pasCroissant()  
  pas : entier  
  pas ← 1  
  pour i de 1 à n faire  
    afficher i + pas - 1  
    pas ← i  
  fin pour  
fin module  
  
module boiteuse()  
  pour i de 1 à n faire  
    si i MOD 3 = 1 alors  
      afficher i  
    fin si  
  fin pour  
fin module  
  
module suiteDeFibonacci()  
  i1, i2 : entiers  
  i1 ← 0  
  i2 ← 1  
  si n = 0 alors  
    afficher i1  
  sinon  
    si n = 1 alors  
      afficher i1,i2  
    sinon  
      afficher i1,i2  
      tant que 1 < n faire  
        i ← i1 + i2  
        i2 ← i1  
        i1 ← i  
        afficher i  
      fin tant que  
    fin si  
  fin si  
fin module
```

### Exercice 8 – Génération de suites (2/2)

```
module processionEchternach()
  pour i de 1 à n faire
    afficher i, i + 1, i + 2, i + 3, i + 2
  fin pour
fin module

module combinaison2Suites()
  pour i de 1 à n faire
    afficher i + (i - 1), i + 1
  fin pour
fin module

module capricieuse()
  pour i de 1 à n faire
    si i MOD 2 = 1 alors
      pour j de 10 * i - 9 à 10 * i faire
        afficher j
      fin pour
    sinon
      pour j de i * 10 à i * 10 - 9 par -1 faire
        afficher j
      fin pour
    fin si
  fin pour
fin module
```

### Exercice 9 – Factorielle

```
module factorielle()
  n, fact : entiers
  fact ← 1
  lire n
  pour i de n à 1 par -1 faire
    fact ← fact * i
  fin pour
  afficher fact
fin module
```

### Exercice 10 – Somme de chiffres

```
module sommeChiffre()
  chiffre, nombre : entiers
  lire nombre
  tant que nombre > 0 faire
    chiffre ← nombre MOD 10
    nombre ← nombre / 10
  fin tant que
  afficher chiffre
fin module
```