Calcul scientifique et programmation.

Introduction aux langages de programmation

Alexei Stoukov

Septembre 2017

Sommaire

Problème physique à résoudre : équations

Soit un écosystème dynamique composé de deux espèces, une de prédateurs (P=Patelles) et l'autre de proies (G=Goëmons), dont l'évolution des populations est gouvernée par le système d'équations différentielles suivant :

$$\begin{cases} \frac{dG}{dt} = \underbrace{k_1 G}_{(1)} - \underbrace{k_2 G^2}_{(2)} - \underbrace{k_3 PG}_{(3)} \\ \frac{dP}{dt} = \underbrace{\beta k_3 PG}_{(4)} - \underbrace{k_4 P}_{(5)} - \underbrace{k_5 P^2}_{(6)} \end{cases}$$

Problème physique à résoudre : constantes

k_1, k_2, k_3, k_4, k_5 et β sont des paramètres constants et positifs. Les termes des seconds membres modélisent :

(1) le bilan pour G entre les vitesses de reproduction et de décès naturels,

(2) la limitation des ressources vitales pour G,

(3) la diminution de la population G par prédation par l'espèce P,

(4) la vitesse de reproduction de P considérée comme proportionnelle au nombre de proies G,

(5) la vitesse de décès naturels de l'espèce P,

(6) la limitation des ressources vitales pour P autres que l'espèce G.

Problème physique à résoudre : objectifs

- Effectuer l'analyse qualitative et quantitative de l'évolution de ce système en fonction des conditions initiales $G\,(t=0)$ et $P\,(t=0)$.

Constantes (empiriques ?) :
$k_1 = 1.1,\ k_2 = 10^{-5},\ k_3 = 10^{-3},\ k_4 = 0.9,\ k_5 = 10^{-5},\ \beta = 0.02$
l'unité de temps est le jour.

Discrétisation du problème physique

Equation du premier ordre

$$\frac{dy}{dt} = f(t, y)$$
$$(t_0, y_0)$$

Schéma de discrétisation temporelle

Schéma de Runge-Kutta d'ordre 1 (schéma d'Euler)

$$y'_0 = f(t_0, y_0)$$
$$y_1 = y_0 + hf(t_0, y_0)$$

$$k_1 = hf(t_0, y_0)$$
$$y_1 = y_0 + k_1$$
avec $h = \Delta t$

Application du schéma de Runge-Kutta au problème

n - indice temporel, forme discrétisée de $G(t), P(t)$ est G^n, P^n
De la même façon : $G(t+\Delta t), P(t+\Delta t) \rightarrow G^{n+1}, P^{n+1}$

Modèle discrétisé

$$G^{n+1} = G^n + \Delta t(k_1 G^n - k_2 G^{n^2} - k_3 P^n G^n)$$

$$P^{n+1} = P^n + \Delta t(\beta k_3 P^n G^n - k_4 P^n - k_5 P^{n^2})$$

Résolution du problème discrétisé

Posons

- Temps d'observation (temps final) $t_f = 10$ jours
- Pas de temps $\Delta t = 0.1$ jours

Nombre des pas de temps $N = 100$
Nombre d'opérations arithmétiques nécessaires :

- (6 multiplications+2 soustractions+1 addition)*100=900 pour G
- (7 multiplications+ 2 soustractions+ 1 addition)*100=1000 pour P

Comment effectuer ces opérations ?

- Calcul mental
- Papier,crayon, gomme
- Boulier
- Calculette
- Ordinateur !

Langages machine

Chaque famille de processeur possède un langage binaire qui lui est propre : le langage machine

00000010101111001010
00000010111111001000
00000011001110101000

Pour chaque langage machine, on peut avoir une variante symbolique, dite langage d'assembleur

LOAD I ADD J STORE K

... alors qu'il est tellement plus facile et lisible d'utiliser :

k = i + j (mais il a fallu plus de 10 ans pour y arriver !)
Abstraction = élimination d'erreurs, augmentation de la productivité, de l'efficacité et de la portabilité

Langage informatique : définition

Langage informatique

Ensemble de mots-clés et de règles de syntaxe permettant de créer des programmes informatiques

Deux grandes familles :

- Langages interprétés
- Langages compilés

Code source

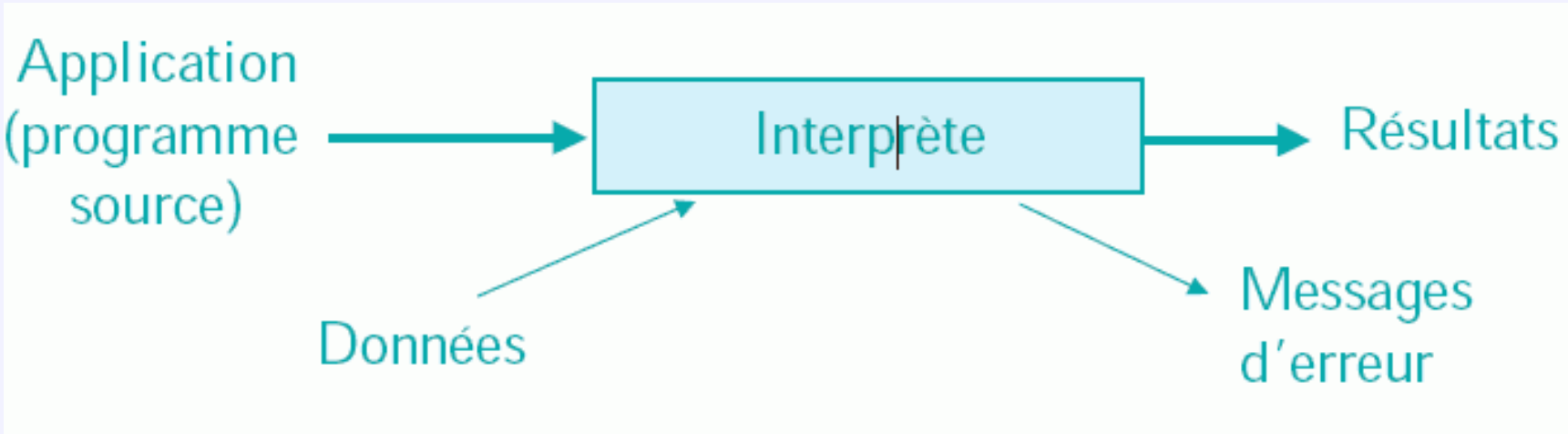
Code (texte) source - définition *selon Erwan Esnault*

La représentation dans un langage humainement compréhensible du fonctionnement d'une oeuvre.
Le langage est choisi initialement par l'auteur.
Ce langage peut-être également standardisé, normalisé ou tout au moins reconnu et utilisé de la même manière par un ensemble de personnes.
Le code source peut être complété de commentaires et de documentation en langage naturel.
Le but du code source est d'être utilisé par un dispositif de transformation en langage compréhensible (processeur, compilateur, interpréteur) par une machine numérique (un ordinateur) qui donnera le code machine.
L'utilisation de ce code sur la machine donnera l'oeuvre.

Langages interprété

Un interprète (ou interpréteur)

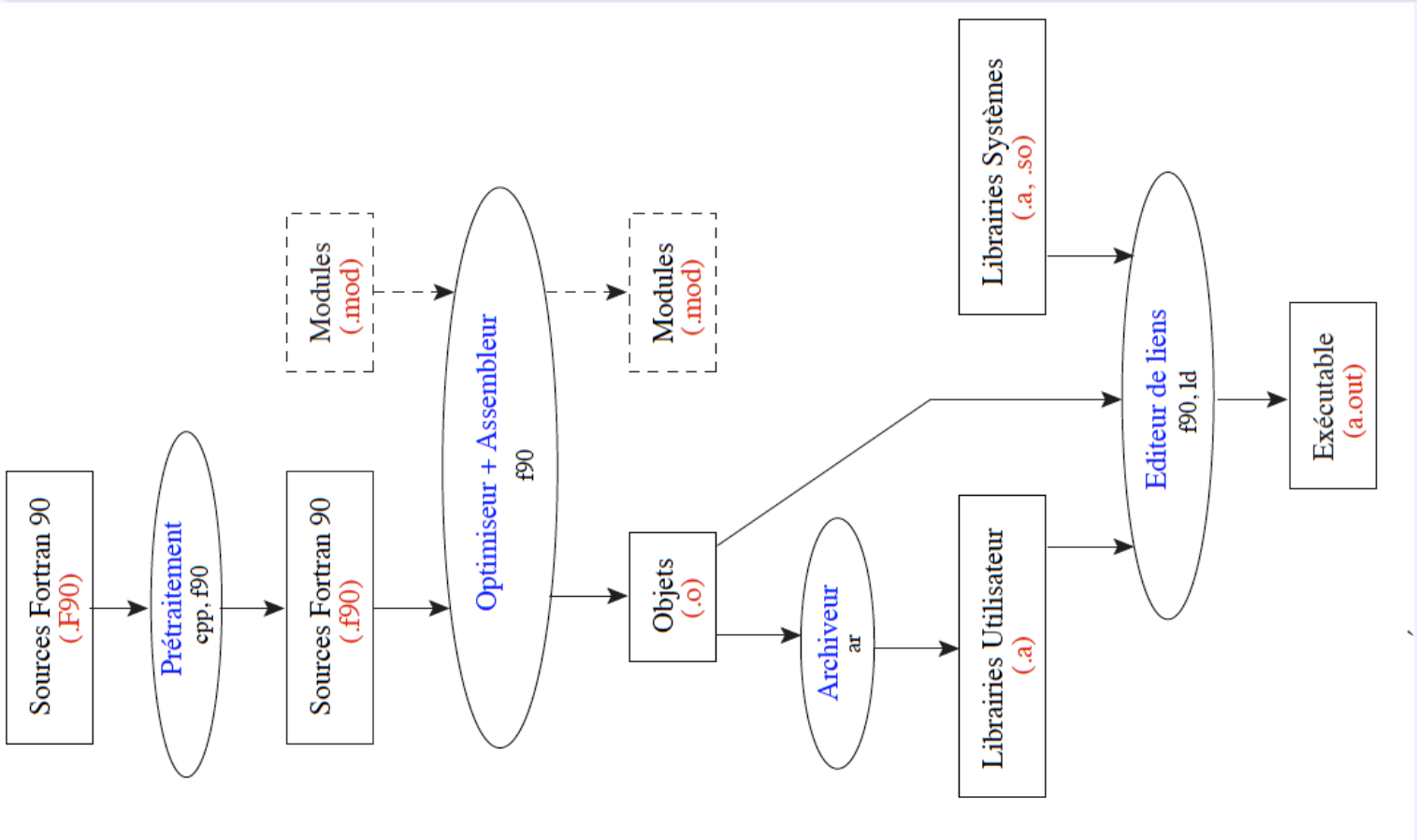
Lit, puis exécute immédiatement le texte source



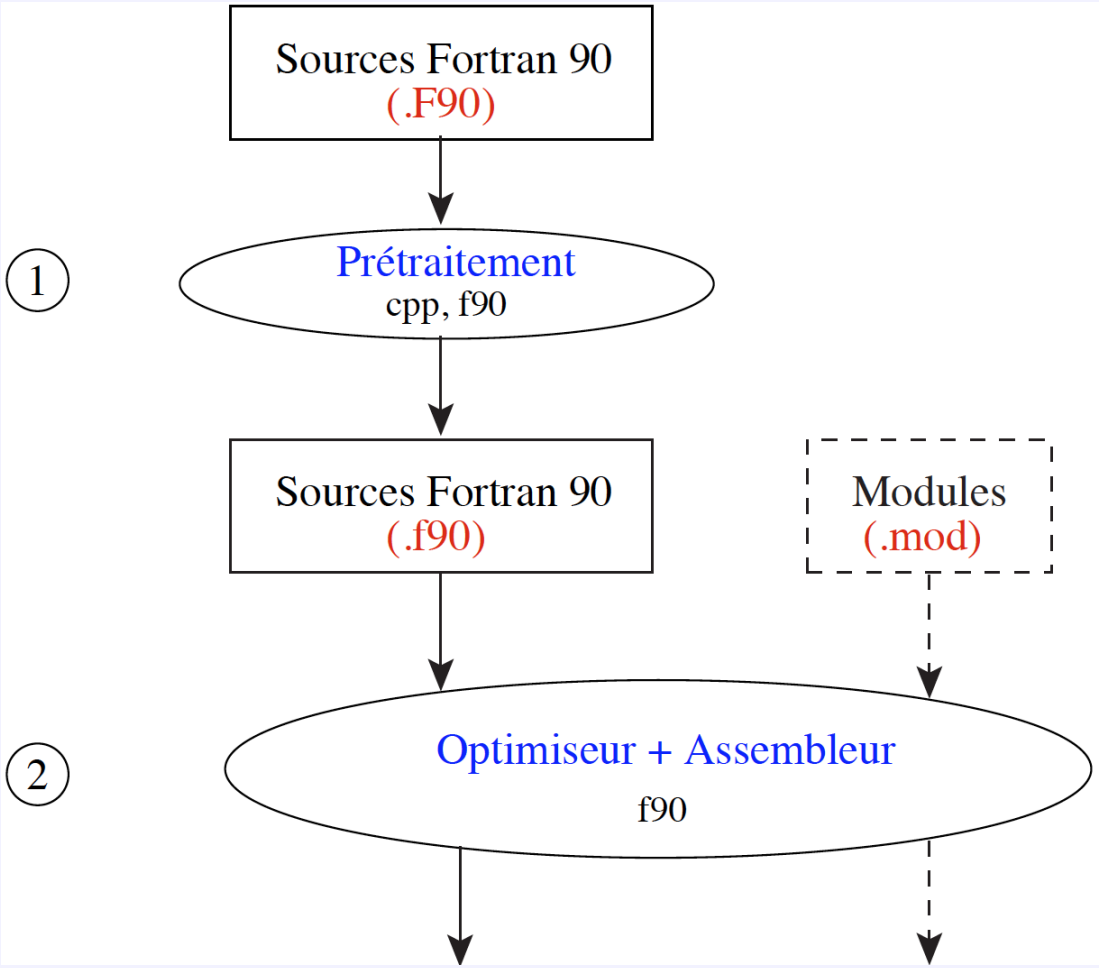
Langages compilés

Un compilateur

Prend un texte source et le traduit dans un langage cible ; le résultat est un fichier objet. Le(s) fichier(s) objet est lié aux autres objets, le fichier exécutable est créé. L'exécution est remise à plus tard.

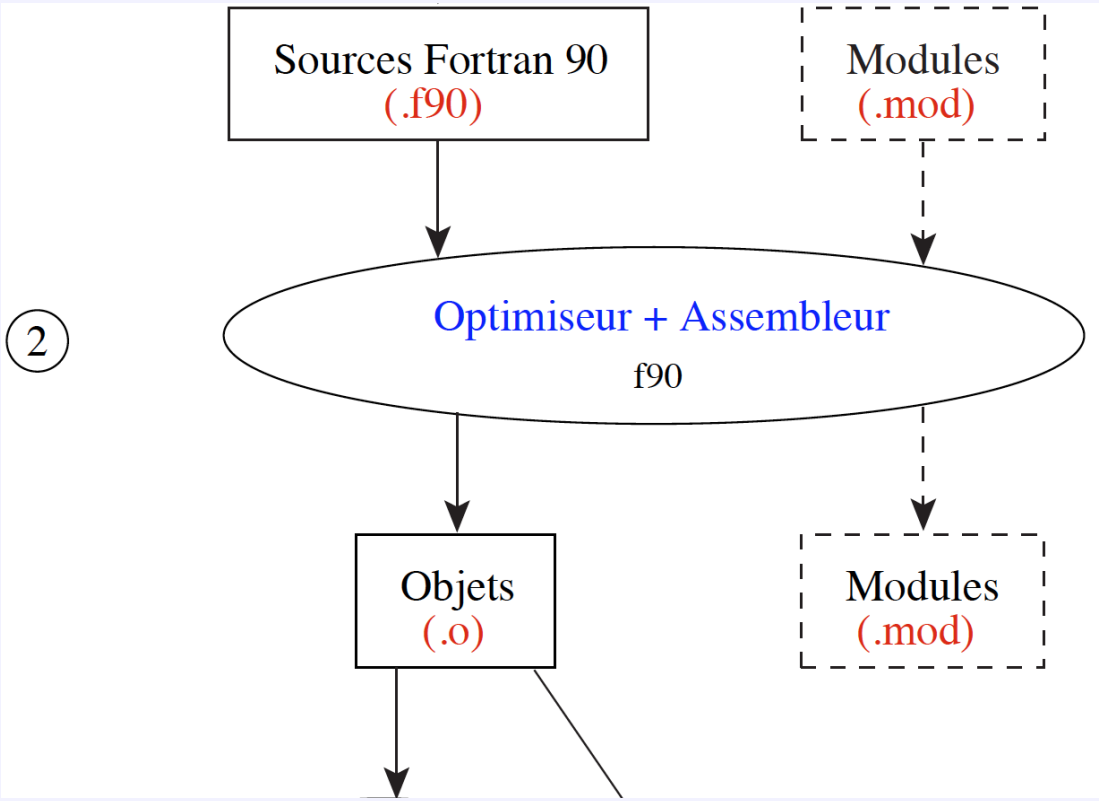


Etapes de compilation



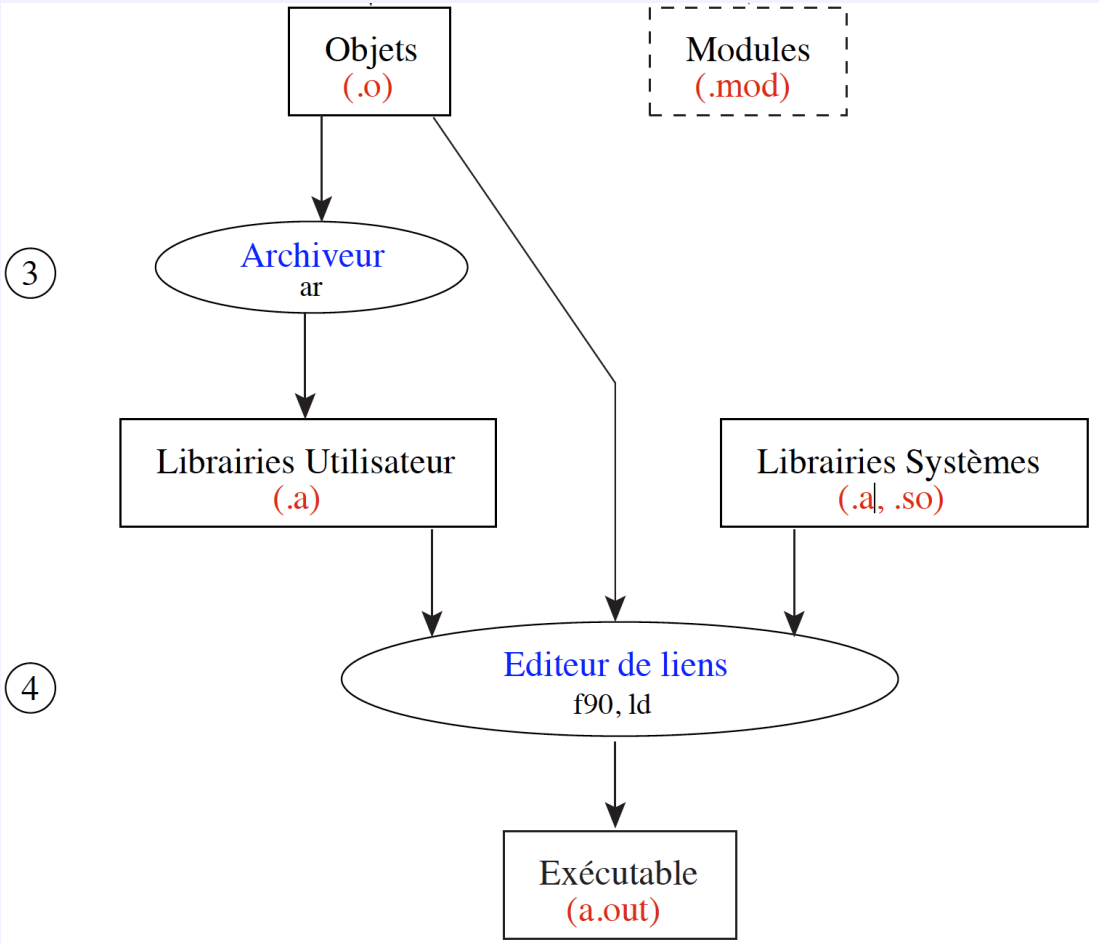
1. Le pré-traitement des sources remplace des parties de texte par d'autres textes grâce à un système de macros. Cette opération **non-obligatoire** est réalisée par un préprocesseur (cpp, fpp)

Etapes de compilation



2. L'optimisation et l'assemblage du code transforme le programme source en instruction assembleur après une étape d'optimisation puis effectue la traduction vers les instructions machines. Les fichiers binaires générés sont appelés fichiers objets (extension .o).

Etapes de compilation



4. L'édition de liens permet d'ajouter les bibliothèques systèmes et de rassembler tous les objets (sous forme de librairie ou individuellement) qui composent l'application pour former l'exécutable.

Exemples de langages

ADA	Le temps réel	compilé
BASIC	Programmation basique à but éducatif	interprété
C	Programmation système	compilé
C++	Programmation système objet	compilé
Cobol	Gestion	compilé
Fortran	Calcul	Langage compilé
Java	Programmation orientée internet	intermédiaire
MATLAB	Calcul mathématique	interprété
Mathematica	Calcul mathématique	interprété
LISP	Intelligence artificielle	intermédiaire
Pascal	Enseignement	compilé
PHP	Développement de sites web dynamiques	interprété
Prolog	Intelligence artificielle	interprété
Perl	Traitement de chaînes de caractères	interprété
Python	Calcul mathématique, Enseignement	interprété

Exemple d'un langage interprété : Python

Modèle discrétisé

$$G^{n+1} = G^n + \Delta t(k_1 G^n - k_2 G^{n^2} - k_3 P^n G^n)$$

$$P^{n+1} = P^n + \Delta t(\beta k_3 P^n G^n - k_4 P^n - k_5 P^{n^2})$$

```
k1 = 1.1
k2 = 1.e-05
k3 = 1.e-03
k4 = 0.9
k5 = 1.e-05
beta = 0.02
P = [10.]
G = [100.]
dt = 0.2
t = 0.
```

```
tf = 100.
it = 0
while t <= tf :
    t = t+dt
    G.append (G[it] + dt*(k1*G[it] - k2*G[it]**2
                          - k3*P[it]*G[it]))
    P.append (P[it] + dt*(beta*k3*P[it]*G[it] -
                          k4*P[it] - k5*P[it]**2))
    it = it+1
print t, G[it], P[it]
```

Exemple d'un langage interprété : Python

A noter

- Programmation impérative structurée
- Orientée objet
- Typage dynamique fort
- Gestion automatique de la mémoire par ramasse-miettes
- Besoin d'interpréteur mais ne nécessite pas la compilation

Exemple d'un langage compilé : Fortran 90

```
program eco_0
implicit none
integer :: it,npas
real :: k1,k2,k3,k4,k5,beta,dt,t,tf
real,allocatable,dimension(:) :: P,G

k1 = 1.1
k2 = 1.e-05
k3 = 1.e-03
k4 = 0.9
k5 = 1.e-05
beta = 0.02
dt = 0.2
t = 0.
tf = 100.
npas = int(tf/dt)+1
```

Exemple d'un langage compilé : Fortran 90

```
allocate (P(0:npas+1),G(0:npas+1))
P(0) = 10.
G(0) = 100.
it = 0
do while (t <= tf)
   t = t+dt
   G(it+1) =  G(it) + dt*(k1*G(it) - k2*G(it)**2 - k3*P(it)*G(it))
   P(it+1) =  P(it) + dt*(beta*k3*P(it)*G(it) - k4*P(it) - k5*P(it)**2)
   it = it +1
end do
print*, 'Tfinal=',t, 'Goemons = ',G(it), 'Pattelles = ',P(it)

stop
end
```

Exemple d'un langage compilé : Fortran 90

A noter

- Programmation structurée
- Pas de typage dynamique, déclaration de toutes les variables
- Gestion de la mémoire par l'allocation
- Besoin d'un compilateur, mais le code compilé peut être exécuté sur des plates-formes similaires.