# Paradigmes et langages de programmation

Exercices sur les chapitres 1 et 2

6 septembre 2007

## Chapitre 1 : Introduction et historique

### 1.1. Lecture : une introduction historique

« Historical survey », chapitre 2 du livre *Comparative programming languages* par Leslie B. Wilson et Robert G. Clark, Addison-Wesley 1988.

**Questions :**

1. Ce texte a 18 ans. Comment imagineriez-vous de le compléter ?
2. Situez dans ce résumé historique les langages de programmation qu'on vous a appris jusqu'à maintenant dans vos études.
3. Même question pour les langages que vous connaissez ou dont vous avez entendu parler mais qui n'ont pas fait partie des enseignements suivis.
4. Étudiez les questions 2.2, 2.4 et 2.5 dans les exercices de ce chapitre.

### 1.2. Lecture : Sur la conception des langages

« Why Pascal is not my favorite programming language », par Brian W. Kernighan, texte non publié de 1981.

**Questions :**

1. Ce texte a 26 ans. Dans les points étudiés et la manière de le faire, qu'est-ce qui vous paraît toujours d'actualité, et qu'est-ce qui peut vous paraître éventuellement dépassé ?
2. L'auteur confond en général trois aspects : la définition d'un langage, l'utilisation qu'on peut en faire, et l'implémentation particulière qu'il a pu utiliser. Notez quelques points significatifs qui ne devraient pas être considérés comme faisant partie du langage.
3. *À faire pour le prochain TD* : Le langage « Pascal étendu » (*Extended Pascal*) a été défini en 1990. Une petite recherche sur le Web vous permettra d'en connaître suffisamment sur ce langage pour vous permettre de déterminer dans quelle mesure ses auteurs ont pris en compte les critiques de Kernighan.

### 1.3. Sur l'implémentation des langages

**Phases du compilateur C** À faire en-dehors des travaux dirigés : faites compiler un programme en C quelconque par `gcc`, en ajoutant l'option `-v`. Déterminez le rôle dans le processus de compilation de chacun des programmes appelés.

**Erreurs** On peut classer les erreurs dans un programme d'après l'endroit du processus de compilation où elles sont détectées. En prenant des exemples dans votre langage de programmation favori, donnez un exemple de chacune des erreurs suivantes :

1. erreur lexicale, détectée par l'analyseur lexical ;
2. erreur syntaxique, détectée par l'analyseur syntaxique ;
3. erreur de sémantique statique, détectée par l'analyseur sémantique ;
4. erreur de sémantique dynamique, détectée par code produit par le générateur ;
5. erreur qui ne peut pas être détectée par le compilateur, et qui soit une violation de la définition du langage.

# Chapitre 2 : Noms

## 2.1. À quel moment se prend la décision

Pour chacun des exemples ci-dessous, indiquez à quel moment se prend la décision pour deux langages différents que vous connaissez bien :

1. nombre de fonctions prédéfinies
2. déclaration de la variable correspondant à une utilisation de variable particulière
3. taille maximale allouée à une chaîne de caractères constante
4. environnement de référence d'un sous-programme passé en paramètre
5. adresse en mémoire d'un sous-programme de bibliothèque particulier
6. taille totale de l'espace alloué au code du programme et à ses données

## 2.2. Portée statique ou dynamique

Soit l'exemple suivant :

```
var x : integer

procedure aff_x(n : integer)
   x := n

procedure impr_x
   write_integer(x)

procedure un
   aff_x(1)
   impr_x

procedure deux
   var x : integer
   aff_x(2)
   impr_x

aff_x(0)
un
impr_x
deux
impr_x
```

1. Indiquez ce que ce programme imprime si la portée des noms est statique.
2. Même question si elle est dynamique.
3. Dans le cas d'une portée dynamique, montrez l'évolution du mécanisme de gestion des portées pendant l'exécution du programme.

## 2.3. Liaison de surface et liaison profonde

Soit l'exemple suivant :

```
var x : integer

procedure aff_x(n : integer)
   x := n

procedure impr_x
   write_integer(x)

procedure toto(A, I : procedure; n : integer)
   var x : integer
   if n in {1, 3} then
      aff_x(n)
   else
      A(n)
   endif
   if n in {1, 2} then
      impr_x
   else
      I
   endif

aff_x(0); toto(aff_x, impr_x, 1); impr_x
aff_x(0); toto(aff_x, impr_x, 2); impr_x
aff_x(0); toto(aff_x, impr_x, 3); impr_x
aff_x(0); toto(aff_x, impr_x, 4); impr_x
```

On suppose que la portée est dynamique.

1. Montrez ce qu'imprime le programme si la liaison est de surface.
2. Même question si elle est profonde.