Langages et paradigmes

Olivier Lecarme

Master d'Informatique, première année

2007–2008

Première partie I

Introduction

Organisation de cet enseignement

Il est organisé en plusieurs périodes :

1. Semaines 39 à 43 : Introduction générale
   - 2h de cours par semaine :
   - 2h de travaux dirigés par semaine : lectures dirigées et commentées, exercices

Organisation

2. Étude de trois langages de programmation (C++, Caml et Icon) :
   - Semaines 39 à 42 : C++
     - 2h de cours par semaine
     - 2h de travaux pratiques par semaine
   - Semaines 43 à 46 : Caml et Icon :
     - 1h30 de cours par semaine : deux séances par langage
     - 1h30 de travaux pratiques par semaine : deux séances par langage

Organisation

3. Semaines 47 à 51 : Mini-projet
   - groupes de 5
   - un sujet différent par groupe
   - le sujet est à traiter dans les trois langages
4. Semaine 3 : Conclusion
   - présentation des mini-projets (écrite et orale)
   - cours de synthèse et conclusion (par courrier)

Pourquoi tant de langages de programmation ?

- Nombre de langages

  Il existe plusieurs milliers de langages de programmation, et on continue d'en créer. Pourquoi ?

Nombre de langages

- Évolution de la discipline :
  - passage du langage machine au langage d'assemblage puis aux langages de haut niveau
  - progrès dans les formalismes de description
  - progrès dans les techniques de compilation
  - courants nouveaux en méthodologie : programmation structurée, programmation modulaire, programmation par objets, etc.

Nombre de langages

- Variété des problèmes à résoudre :
  - manipulation de données symboliques
  - structures de données très complexes
  - opérations complexes sur chaînes de caractères
  - programmation de bas niveau pour les systèmes d'exploitation
  - représentation du raisonnement logique
  - nouveaux problèmes spécifiques
  - applications hautement sécuritaires

Nombre de langages

- Préférences personnelles :
  - questions de goûts personnels principalement
  - écrire bref ou écrire clair ?
  - récursivité ou itération ?
  - pointeurs explicites ou non ?
  - sécurité ou liberté ?
  - un seul langage « universel » ou plusieurs langages spécialisés ?
  - l'interactivité est-elle importante ?

Succès des langages

Parmi les milliers de langages qui ont été définis et implémentés, quelques centaines seulement sont encore utilisés, et quelques dizaines le sont de manière importante.

Succès des langages

Qu'est-ce qui fait qu'un langage a du succès ?

- Puissance d'expression :
  - tout langage permet de tout exprimer, mais plus ou moins facilement
  - certaines caractéristiques ont une grande importance sur la facilité de programmation
  - les possibilités d'*abstraction* sont les plus importantes

Succès des langages

- Facilité d'utilisation :
  - facilité d'apprentissage pour un débutant
  - rapidité des progrès
  - absence de pièges et de surprises
- Facilité d'implémentation :
  - sur des machines peu coûteuses
  - rapidité de construction d'une nouvelle implémentation
  - transportabilité de l'implémentation
  - absence de freins légaux ou commerciaux

Succès des langages

- Qualité de l'implémentation :
  - bonnes performances dès le début
  - restrictions faites au langage pour le permettre
  - convivialité des outils annexes
- Facteurs économiques et autres :
  - support par un industriel puissant : Fortran, Cobol, PL/I
  - support par un organisme public : Ada
  - campagne publicitaire : Java
  - occupation d'une niche spécifique : Prolog
  - difficulté du changement
  - inertie

Catégories de langages de programmation

- Comment les définir

  Les catégories ont des limites floues. On ne s'intéressera de toutes manières ici qu'aux « vrais » langages de programmation, en excluant les langages d'assemblage, les macro-processeurs, etc.

Catégories de langages

- Langages « impératifs » ou *procéduraux*.
  - Modèle de von Neumann :
    - le calcul est décrit par les changements d'état des variables
    - Fortran, Algol, Pascal, Basic, C, Ada, Icon, etc.
    - c'est le modèle le plus répandu, en particulier parce qu'il est conforme à la structure des ordinateurs.

Catégories de langages

- Modèle par objets :
  - variante ou spécialisation du modèle de von Neumann
  - le calcul est décrit par les interactions entre objets
  - Simula 67, Smalltalk, Eiffel, C^{++}, Java, etc.
  - c'est le modèle le plus à la mode.

Catégories de langages

- Langages déclaratifs
  - Modèle fonctionnel :
    - fondé sur le *lambda-calcul* : un programme est une fonction qui fournit un résultat à partir d'arguments
    - Lisp, Scheme, ML, etc.
    - c'est le modèle préféré des mathématiciens.
  - Modèle par flot de données :
    - le calcul se fait par les données qui traversent des nœuds fonctionnels
    - Id, Val, Sisal
    - modèle d'importance actuelle marginale.

Catégories de langages

- Modèle logique :
  - le calcul est la tentative de trouver des données qui satisfont des équations logiques
  - Prolog presque uniquement
  - les tableurs ordinaires peuvent se rattacher à ce modèle
  - modèle résistant dans sa niche.

En fait, la plupart des langages fonctionnels incluent des mécanismes procéduraux, et on peut programmer de manière fonctionnelle dans la plupart des langages procéduraux.

Pourquoi étudier les langages de programmation ?

- Quelques raisons fondamentales :
  - position centrale dans l'informatique professionnelle
  - permettre de choisir le langage le plus approprié au problème
  - faciliter l'apprentissage de nouveaux langages
  - découvrir les mécanismes et principes sous-jacents
  - améliorer votre esprit critique

Pourquoi étudier

- Comprendre les décisions de conception et d'implémentation pour mieux utiliser les langages :
  - Comprendre les caractéristiques obscures : on peut les ramener aux grands principes si on les connaît.
  - Choisir parmi les différentes manières de faire la même chose, en particulier en tenant compte des coûts :
    - passage par valeur de variables encombrantes
    - passage par nom
    - utilisation d'astuces pour améliorer les performances
    - évaluation des compromis entre encombrement et rapidité

Pourquoi étudier

- Simuler des caractéristiques manquantes :
  - programmation structurée par discipline et commentaires
  - conventions de nommage pour imiter la modularité
  - simulation des générateurs à l'aide de fonctions et de variables statiques
  - simulation de la récursivité

Pourquoi étudier

- Appliquer les connaissances acquises à d'autres situations similaires :
  - langages de commande, shells
  - générateurs de rapports
  - outils programmables (Emacs)
  - fichiers de configuration et options de commandes
  - langages de description (HTML, XML)

Formalismes de définition

- Aspects lexicaux :
  - en général non séparés de la syntaxe
  - distinction entre symboles et représentation en Algol 60 et Algol 68
  - pas de formalisme spécifique
- Aspects syntaxiques :
  - forme normale de Backus à partir d'Algol 60
  - notation originale mais sans descendance pour Cobol
  - extensions variées à BNF, sans uniformisation

Formalismes

- Aspects sémantiques :
  - utilisation de la langue naturelle la plus fréquente
  - expression stéréotypée dans les documents de normalisation
  - Vienna Definition Language pour PL/I
  - grammaires de van Wijngaarden (grammaires à deux niveaux) pour Algol 68 (effet néfaste)
  - définition axiomatique pour Pascal
  - Vienna Definition Method pour Modula-2
  - à part Algol 68, la formalisation apparaît tardivement
  - peu d'influence sur la définition du langage

Normalisation des langages

- Organismes de normalisation
  - Organismes nationaux :
    - AFNOR en France
    - DIN en Allemagne
    - BSI au Royaume-Uni
    - ANSI et FBS aux États-Unis

Organismes

- Organismes influents :
  - IEEE aux États-Unis
  - Grandes associations (ACM)
- Organismes internationaux :
  - ISO
  - CCITT
  - IFIP

Rôles de la normalisation

- rassembler des définitions divergentes
- clarifier la définition
- faire évoluer le langage
- définir un nouveau langage
- clairement ces buts sont contradictoires

Difficultés de la normalisation

- lourdeur générale des mécanismes
- intérêts contradictoires des participants
- non-représentativité des participants
- conflits entre précision et clarté
- trop grande lenteur

Quelques exemples de normalisations

- Fortran
  - le premier exemple de normalisation de langage de programmation
  - seul moyen d'évolution du langage après Fortran IV
  - dominante : conserver la compatibilité avec les programmes existants
- PL/I
  - normalisation inutile
  - simple officialisation de la définition du langage par IBM

Exemples

- Algol 68 : la norme ISO est un pointeur sur le document de l'IFIP
- Pascal
  - première normalisation d'un langage n'émanant pas d'une puissance économique
  - conflit entre l'ISO et l'ANSI
  - tentative de ne faire que clarifier
  - deux extensions indispensables
  - ensuite, définition d'un nouveau langage

Exemples

- C
  - une grande partie du document sert à préciser ce qui n'est pas précisé
  - l'important est de ne pas invalider les compilateurs des fabricants de logiciel représentés dans le comité
  - travail interne à l'ANSI

Exemples

- Modula-2
  - travail similaire à celui pour Pascal : propre à l'ISO, pas d'intervention de l'auteur du langage
  - choix malheureux de VDM comme formalisme
  - norme ayant « tué » le langage

Tableau généalogique

- Commentaires sur le tableau
  - dates approximatives
  - filiations schématiques et parfois discutables
  - choix de langages supposés représentatifs : les plus importants devraient être présents
  - certains langages très anciens sont encore très utilisés : Cobol, Lisp
  - certains langages ne font plus parler d'eux, mais sont utilisés par des communautés fidèles : APL, Simula 67, Snobol4, Algol 68

Généalogie

- certains langages sont mentionnés mais n'ont pas d'utilisation importante
- parmi les implémentations libres ou gratuites :
  - FSF : C, C^{++}, Fortran 77, Pascal, Ada 95, Prolog, Java, Eiffel, Cobol (en cours) : suite de compilation avec langage intermédiaire unique
  - autres : Oberon, Scheme, Snobol4, Icon, Perl, APL, etc.

Quelques mots sur quelques langages

- Fortran : « traducteur de formules » plus efficace qu'un bon programmeur (Backus, IBM)
- Algol 60 : langage d'expression des algorithmes pour la communauté scientifique (universitaires européens et américains, Backus, Naur, McCarthy, Bauer, Rutishauser, Woodger, Perlis, Vauquois, etc.)
- Cobol : la langue américaine pour dire ce qui doit être fait dans les affaires (Codasyl, IBM)
- Lisp : λ-calcul, auto-définition, traitement de listes (McCarthy, MIT)

Quelques langages

- APL : tout s'exprime par combinaison d'opérateurs et élargissement aux tableaux (Iverson)
- PL/I : applications numériques et commerciales dans le même langage, « tout marche » (IBM)
- Algol 68 : groupe WG 2.1 de l'IFIP, principes de base poussés au paroxysme (van Wijngaarden)
- Snobol4 : modèle de Markov, traitement de chaînes de caractères (Griswold, Bell Labs)
- Pascal : contre Algol 68, simple, systématique, fait pour enseigner les bases de la programmation (Wirth)

Quelques langages

- C : programmation de bas niveau la plus concise possible, pas d'avenir prévu (Ritchie, Bell Labs)
- Prolog : moteur d'évaluation d'équations logiques, pris comme emblème du projet d'ordinateurs de 5e génération (Colmerauer et Roussel)
- Ada : appel d'offres du Ministère de la Défense américain, évaluation internationale (Ichbiah)
- Java : projet initial sans avenir, marketing et publicité pour contrer Microsoft (Gosling, Sun)