I.U.T MONTPELLIER
DÉPARTEMENT INFORMATIQUE

Programmation 1

Introduction à l'algorithmique et à la programmation

(7 semaines)

1

Volume horaire

- Prog.1 (jusqu'à Toussaint)
  - Cours magistral : 1h (Amphi)
  - TD / TP : 6h (TD + TP Machines)

2

Programme du premier demi-semestre

- Structures de base
- Procédures, passage de paramètres
- Fonctions
- Tableaux à une dimension
- Tableaux à plusieurs dimensions
- (Enregistrements / Classes) => Prog1-bis

3

Langages utilisés

- TD : Langage algorithmique
- Salle machines : Langage de programmation

4

Contrôles

Contrôle continu : deux petits contrôles

Contrôle continu & Rendus de TD/TP → Note TD

Partiel de mi-semestre → Note Partiel

(si absence justifiée, contrôle de rattrapage sous forme orale ou écrite)

Note de l'UV = moyenne (Note TD , Note Partiel)

5

Développement d'un logiciel

- Analyse
- Spécifications
- Algorithme
- Les variables
- Les instructions
- Programmation
- Tests programmeur
- … Tests utilisateur

6

Introduction

Un problème peut être traité avec un ordinateur si :

- on sait parfaitement définir les données et les résultats
- on peut décomposer le passage des données vers ces résultats en une suite d'opérations élémentaires dont chacune peut être exécutée par un ordinateur

7

Introduction

L'algorithmique consiste donc à manier la structure logique d'un programme informatique,

8

Introduction

Algorithme → Programme source → Programme objet

(pseudo-code) (langage de programmation) (langage machine)

9

Introduction

Suivant le langage de programmation utilisé, le programme source sera ensuite :

- soit **compilé**,
- soit **directement interprété**
- soit **semi interprété**

10

Algorithme

Un algorithme est un énoncé, dans un langage bien défini, d'une suite d'opérations permettant de résoudre par calcul un problème donné bien spécifié.

Un algorithme doit répondre aux 5 caractéristiques suivantes :

- la finitude
- la non-ambiguïté
- le déterminisme
- le domaine des données
- le domaine des résultats

11

Les variables

Dans un programme on aura besoin de stocker en permanence des **valeurs**.

12

Les variables

Avant d'être utilisée, une variable doit être déclarée :

- identificateur de la variable
- type des valeurs utilisées
- valeur initiale
- signification / rôle : *sous forme de commentaire*

13

Les variables : quelques remarques

Quelques remarques :

14

Les variables : quelques remarques

Il existe des opérateurs spécifiques pour chaque type.

Attention à ne pas confondre « contenant » et « contenu »

15

Les instructions

5 types d'instructions :

- Instruction vide
- Affectation
- Itération
- Alternative (ou sélective) et choix multiple
- Appel à une procédure

16

Les instructions

Instruction vide

17

Les instructions

Affectation :

Variable ← Expression

Attention : En partie gauche, c'est bien une Variable,
et non une fonction d'une variable.

18

Les instructions

Affectation :

Attention :

Ce n'est pas une instruction symétrique

19

Les instructions

Affectation :

- Langage algorithmique :

  Variable ← Expression

- Langage ADA :

  Variable := Expression

- Langage JAVA :

  Variable = Expression

20

Les instructions

Itération : répétition d'une action

1°/ Instruction **TANT QUE**

2°/ Instruction **POUR**

tant que *Condition*
faire
instructions
fin tant que

Les instructions

Itération Tant Que :

⇒ **Attention** : Il faut que *Condition* puisse devenir Fausse, sinon …

```
ex1:  X, Y: entier
      Début
        X ← 5
        Tant Que X > 0
        Faire  Y ← X
        Fin TQ
      Fin
```

```
ex2:  X: entier
      Début
        X ← 2
        Tant Que X > 0
        Faire  X ← X – 1/X
        Fin TQ
      Fin
```

Les instructions

Itération Tant Que :

Saisie d'une suite de valeurs terminée par une valeur sentinelle

ex.: moyenne des ages des étudiants, saisie terminée par 0

```
som ← 0                 -- somme des ages des étudiants
cpt ← 0                 -- compteur du nombre d'étudiants
Lire (age)              -- saisie age 1er étudiant
Tant Que age ≠ 0
Faire  som ← som + age
       cpt ← cpt + 1
       Lire (age)       -- saisie age étudiant suivant
Fin TQ
Ecrire (som / cpt)      -- Attention : si cpt = 0 …
```

23

Les instructions

Remarque sur l'expression de la condition d'égalité :

24

Les instructions

Itération Tant Que :

- Langage ADA : While *condition*
  loop *instructions*
  end loop ;

- Langage JAVA : while (*condition*)
  { *instructions* } *;*

25

Les instructions

Itération POUR :

Pour *VariableDeContrôle* dans *Intervalle*
faire
instructions
fin pour

26

Les instructions

Itération POUR :

Attention : L'exécution de *Instructions* ne doit pas modifier la valeur de *VariableDeContrôle*, ni les bornes de *Intervalle*.

27

Les instructions

Itération POUR :

Pour *VariableDeContrôle* dans *Intervalle*
faire
instructions
fin pour

- Langage ADA : For *VarDeContrôle* in *Intervalle*
  loop *instructions*
  end loop ;
- Langage JAVA : for (*initialisation* ; *condition* ; *incrémentation*)
  { *instructions* } ;

28

Les instructions

Alternative :

si ***Condition***
alors ***instructions_A***
sinon ***instructions_B***
fin si

Les instructions

Alternative :

si *Condition*
alors *instructions_A*
sinon *instructions_B*
fin si

- Langage ADA : If *condition*
  then *instructions_A*
  else *instructions_B*
  end if ;
- Langage JAVA : if (*condition*)
  { *instructions_A* }
  else *{ instructions_B } ;*

Les instructions

Appel à une procédure

Rmq.: Les procédures permettent de décomposer un problème.

Rmq.: L'appel à fonction ne constitue pas une instruction.

31

Chronologie de création d'un programme

(partie Programmation)

Ecriture de l'algorithme (langage algorithmique) => logique

Traduction en langage de programmation => syntaxe

Compilation

Correction éventuelle des erreurs de syntaxe

Edition de liens

Exécution (et Tests programmeur)

Correction éventuelle des erreurs logiques (debugging)

32

Structure d'un algorithme

Un algorithme est composé de quatre parties :
(simplifiées ici)

- En-tête
- Spécifications
- Déclarations
- Instructions

33

Structure d'un programme (écrit en ADA ou JAVA)

De plus, en langage de programmation (ADA ou JAVA), il faudra préciser les **clauses de contexte** .

En ADA :

```
with ada.text_io;  use ada.text_io;
with ada.integer_text_io;  use ada.integer_text_io;      -- pour les entiers
with ada.float_text_io;  use ada.float_text_io;          -- pour les réels
```

En JAVA, cela dépendra de l'environnement utilisé en TD :

```
import Utile.class;
```

En JAVA, il faudra également déclarer une classe principale dans laquelle sera écrite votre procédure (méthode)

34

Exemple

programme PGCD

pré-requis : aucun

action : calcule le plus grand diviseur commun à deux nombres M et N qui sont saisis au clavier

stratégie : on calcule le reste de la division de M par N;

- si le reste est égal à 0, N est diviseur de M et donc N est le PGCD cherché.
- Si le reste n'est pas nul, on recommence l'opération en divisant N par le reste que l'on vient de calculer
- PGCD (M,N) = PGCD (N, M mod N)
  = PGCD (M mod N, N mod (M mod N)) = …….
  = PGCD (valeur, 0) = valeur

Exemple : PGCD (120,45) = PGCD (45,30) = PGCD (30,15) = PGCD (15,0) = 15

variables

35

Exemple : variables

identificateur	type	Valeur initiale	signification
M	entier	saisi au clavier	La plus grande des 2 valeurs dont on veut calculer le PGCD
N	entier	saisi au clavier	La plus petite des 2 valeurs dont on veut calculer le PGCD
R	entier	reste M/N	Reste de la division entière de M par N

36

Exemple : l'algorithme

```
début
Afficher (« taper deux nombres, le plus grand en premier »)
Saisir (M)
Saisir (N)
R ← M mod N                    -- calcul du reste
tant que R ≠ 0                 -- TQ le reste est ≠ de 0
   faire  M ← N                -- PGCD (M,N) = PGCD (N, M mod N)
          N ← R
          R ← M mod N          -- calcul du nouveau reste
 fin tant que
 Afficher (″ Le PGCD est égal à ″)
 Afficher ( N )
fin
```

37