Algorithmique et structure de données

Chapitre 1 : Introduction

Objectif :

Résoudre un problème, indépendamment du langage

* Le programmeur indique ce que la machine doit faire (≠intelligence artificielle)
* Pas une notion seulement informatique
  + Maths : méthode d’addition
  + Instruction d’un GPS exemples d’algorithmes
  + Recette de cuisine

Fonctionnement :

* Exécution d’ordres de manière séquentielle (méthode systématique)
* Pas de réflexion de la part de l’utilisateur

Conclusion :

Un algorithme dépend de ce que sait faire l’ordinateur

2 types de problème algorithmique :

* Encadrer la racine de 10 par deux entiers successifs
  + Une seule solution : pas très intéressant.
* Encadrer la racine de x par deux entiers successifs
  + Plus intéressant car changement de variables (elles ont chacune des contraintes)
  + Plus intéressant économiquement car permet un plus grand nombre de possibilités pour le même algorithme.

Spécifier un problème

Ensemble d’éléments :

Paramètres d’entrées : un paramètre nommé qui caractérise une instance du problème : élément variable avec l’énoncé

Précondition : condition que doit vérifier les paramètres d’entrée : en dehors de cette condition, le problème n’a pas de sens.

Paramètres de sortie : caractérise la solution à une instance du problème : réponse au problème.

Post-condition : conditions que doivent vérifier les paramètres d’entrée, sortie. Si elle ne correspond pas : la solution au problème est fausse.

Spécification complète : instance du problème décrite par les paramètres d’entrée

Exemple :

:entrée x : réel

:pré-cond : x≥0

:sortie inf : entiers

:sortie sup :

:post-cond : inf ≤ ≤

:post-cond : |sup-inf| :1

Transmission entre algorithme :

:entrée n : entier (affectée précédemment) passé en paramètre

:pré-cond : n≥0 -> condition de validité de l’algorithme contrat d’une fonction

:sortie f : entier ; (affécté par la suite) retourné

:port-cond : f=n !=1x2x3x…n

Données transmises entre algorithmes

Dans ce cas on va écrire un contrat un peu différent :

:entrée n : entier AFFECTE précédemment

:pré-cond : n≥0

:sortie f : entier AFFECTE pour la suite

:post-inf : f=n !=1x2x3x…n