UNIVERSITÉ GRENOBLE ALPES

UFR IM²AG

DÉPARTEMENT LICENCE SCIENCES ET TECHNOLOGIE

LICENCE SCIENCES & TECHNOLOGIES, 1re ANNÉE

UE INF201/231
ALGORITHMIQUE ET PROGRAMMATION FONCTIONNELLE

2023-2024

AIDE-MÉMOIRE

Au partiel de mi-semestre et à l'examen final, aucun document n'est autorisé ; cet aide-mémoire vous sera fourni avec l'énoncé.

Pour ne pas perdre de temps pendant ces épreuves, lisez-le dès aujourd'hui et familiarisez-vous avec sa structure et les informations qu'il contient (ou ne contient pas !).

NB. Toutes les notions résumées ici n'ont pas encore été vues ; en cas de doute, n'hésitez pas à demander à vos enseignants.

T.S.V.P

[git] • Branche master @ 115928d • Version v2024.0.0-partiel (23/02/2024) par François Puitg

SPÉCIFICATION profil ; sémantique ; exemples ou propriétés

RÉALISATION
NON RÉCURSIVE : algorithme ; implémentation
RÉCURSIVE : équations de récurrence ; mesure et terminaison ; implémentation

NOM ↔ VALEUR
GLOBALE : `let...=...`
LOCALE : `let...=...in...`
MULTIPLE : `let...=...and...=...`

COMPOSITION
FONCTIONNELLE : `g(f(...))` ou `g (f ...)`
CONDITIONNELLE : `if ... then ... else ...`
`match ... with ... -> ... | ...` (filtrage)

DÉFINITION DE TYPE : `type ... = ...`

CONSTRUCTEURS

DE TYPE { produit : ***** ; somme : **|** ; fonctionnel : **→** }
DE VALEURS { `C` (sans param.) ; `C of...` (avec param.) } , `C` : constante symbolique

TYPES ET OPÉRATIONS ASSOCIÉES

- BOOLÉENS — $\mathbb{B}$ (`bool`)
  `true`, **`false`** : $\mathbb{B}$; **`not`** : $\mathbb{B} \to \mathbb{B}$; **`(&&)`**, **`(||)`** : $\mathbb{B} \to \mathbb{B} \to \mathbb{B}$; **`string_of_bool`** : $\mathbb{B} \to$ *string*
- ENTIERS — $\mathbb{Z}$ (`int`)
  `(+)`, **`(-)`**, **`( * )`** : $\mathbb{Z} \to \mathbb{Z} \to \mathbb{Z}$; **`(/)`**, **`(mod)`** : $\mathbb{Z} \to \mathbb{Z}^* \to \mathbb{Z}$; **`min_int`**, **`max_int`** : $\mathbb{Z}$; **`abs`** : $\mathbb{Z} \to \mathbb{N}$
  `char_of_int` : $\{0..127\} \to$ *char* : **`float_of_int`** : $\mathbb{Z} \to \mathbb{R}$; **`string_of_int`** : $\mathbb{Z} \to$ *string*
- RÉELS — $\mathbb{R}$ (`float`)
  `(+.)`, **`(-.)`**, **`( *. )`**, **`(/.)`**, **`mod_float`**, **`( ** )`** : $\mathbb{R} \to \mathbb{R} \to \mathbb{R}$
  `Float.pi`, **`epsilon_float`**, **`neg_infinity`**, **`infinity`**, **`nan`** : $\mathbb{R}$; **`abs_float`** : $\mathbb{R} \to \mathbb{R}^+$
  `int_of_float` : $\mathbb{R} \to \mathbb{Z}$; **`string_of_float`** : $\mathbb{R} \to$ *string*
- CARACTÈRES (`char`)
  `int_of_char` : *char* $\to \{0..127\}$; `String.`**`make`** : $\mathbb{N} \to$ *char* $\to$ *string*
- CHAÎNES DE CARACTÈRES (`string`)
  $\wedge$: *string* $\to$ *string* $\to$ *string*
  `bool_of_string` : *string* $\to \mathbb{B}$; **`int_of_string`** : *string* $\to \mathbb{Z}$; **`float_of_string`** : *string* $\to \mathbb{R}$
- SÉQUENCES — *séq* (α) (`'a seq` ou `'a list`)
  `type` **`'a seq`** `= Vide | Cons of 'a * 'a seq`
  `type` **`'a list`** `= [] | (::) of 'a * 'a list`
  `(@)` : *séq* $(\alpha) \to$ *séq* $(\alpha) \to$ *séq* (α)
  `List.`**`hd`** : *séq* $(\alpha)\backslash\{[]\} \to \alpha$; `List.`**`tl`** : *séq* $(\alpha)\backslash\{[]\} \to$ *séq* (α)
- ARBRES BINAIRES — *abin*(α) :
  `type` **`'a abin`** `= Av | Ab of 'a abin * 'a * 'a abin`
  `abS` : $\alpha \to$ *abin*(α) ; **`abUNg`** : *abin*$(\alpha) * \alpha \to$ *abin*(α) ; **`abUNd`** : $\alpha *$ *abin*$(\alpha) \to$ *abin*(α)
  `rac` : *abin*$(\alpha)\backslash\{Av\} \to \alpha$; **`gauche`** : *abin*$(\alpha)\backslash\{Av\} \to$ *abin*(α) ; **`droit`** : *abin*$(\alpha)\backslash\{Av\} \to$ *abin*(α)

COMPARAISON **`(=)`**, **`(<>)`**, **`(<)`**, **`(<=)`**, **`(>)`**, **`(>=)`** : $\alpha \to \alpha \to \mathbb{B}$

ORDRE SUPÉRIEUR
`List.`**`map`** : $(\alpha \to \beta) \to$ *séq* $(\alpha) \to$ *séq* (β)
`(map f [`e_0 `; ... ; `e_n`]) = [f `e_0 `; ... ; f `e_n`]`
`List.`**`fold_left`** : $(\alpha \to \beta \to \alpha) \to \alpha \to$ *séq* $(\beta) \to \alpha$
`(fold_left co init [`e_0`;...;`e_n`]) = (co...(co init `e_0`)...`e_n`)`
`List.`**`fold_right`** : $(\alpha \to \beta \to \beta) \to$ *séq* $(\alpha) \to \beta \to \beta$
`(fold_right init co [`e_0`;...;`e_n`]) = (co `e_0`...(co `e_n` init)...)`
fonction anonyme : `fun ... -> ...`

COMMENTAIRES : `(* ... (* ... *) ... *)` (emboîtables)