



1ALGO Algorithmique et logique

Gaël Roustan (Argonautes)

2025

Abstract

TD 4



TD 4

CONSTRUCTION D'UNE PILE

Écrire une variable globale décrivant une structure de données de type PILE construite comme un tableau de 5 éléments entiers.

ALGORITHME : pile

VARIABLE:

TABLEAU: pile[0..4] : REELS

tete <-- 0 : ENTIER

taille <-- 5 : ENTIER

ALGORITHME POUR UNE PILE

Avec la structure de données de l'exercice précédent, écrire les algorithmiques des primitives EMPILER, DEPILER, EST_VIDE et EST_PLEINE associées à la structure de données non linéaires nommée pile.

ALGORITHME: empiler

VARIABLES:

TABLEAU: pile[0..4] <-- {1, 2, 3, 4, 5} : REELS

tete <-- 5 : ENTIER

taille <-- 5 : ENTIER

element : ENTIER

DEBUT

element <-- LIRE()

SI (tete >= taille) ALORS

 ECRIRE("Il n'y a plus de place")

SINON

 pile[tete] <-- element

 tete <-- tete + 1

FINSI

FIN



ALGORITHME: dépiler

VARIABLES:

pile, tete, taille

DEBUT

SI (tete <= 0) ALORS

 ECRIRE("Pile vide")

SINON

 tete <-- tete - 1

 ECRIRE(pile[tete])

FINSI

FIN

ALGORITHME: est_vide

VARIABLES:

pile, tete, taille

DEBUT

SI (tete <= 0) ALORS

 ECRIRE("Pile vide")

SINON

 ECRIRE("Il y a des éléments")

FINSI

FIN

ALGORITHME: est_pleine

VARIABLES:

pile, tete, taille

DEBUT

SI (tete >= taille) ALORS

 ECRIRE("Pile pleine")

SINON

 ECRIRE("Il reste de la place")

FINSI

FIN



CONSTRUCTION D'UNE LISTE

Écrire un enregistrement nommé `maillonListe` décrivant un maillon pour une structure de données de type LISTE doublement chaînée. La liste contient des entiers.

ENREGISTREMENT: `maillonListe`

DEBUTENREGISTREMENT

* `ptr_precedent` <-- NULL : `maillonListe`

* `ptr_suivant` <-- NULL : `maillonListe`

`element` : ENTIER

FINENREGISTREMENT

ALGORITHME POUR UNE LISTE

Avec la structure de données de l'exercice précédent, écrire les algorithmiques des primitives, CONSTRUIRELISTE avec une première valeur à 10, AJOUTERFIN, et ESTVIDE associées à la structure de données non linéaires nommée liste.

ALGORITHME: `construireliste`

VARIABLES:

*`ptr_liste` <-- NULL : `maillonListe`

DEBUT

`ptr_liste` <-- NOUVEAU : `maillonListe`

(`*ptr_liste`).`element` <-- 10

FIN

ALGORITHME: `estvide`

VARIABLES:

*`ptr_liste`...

DEBUT

SI (`ptr_liste` = NULL) ALORS

ÉCRIRE("Liste vide")

FINSI

FIN

ALGORITHME: `ajouterfin`

VARIABLES:

*`ptr_liste` ...

*`ptr_ajout` <-- NULL : `maillonListe`

*`ptr_parours` <-- NULL : `maillonListe`

DEBUT

`ptr_ajout` <-- NOUVEAU : `maillonListe`

(`*ptr_ajout`).`element` <-- LIRE()

`ptr_parours` <-- `ptr_liste`

TANTQUE ((`*ptr_parours`).`ptr_suivant` != NULL) FAIRE

`ptr_parours` <-- (`*ptr_parours`).`ptr_suivant`

FINTANTQUE

(`*ptr_parours`).`ptr_suivant` <-- `ptr_ajout`

(`*ptr_ajout`).`ptr_precedent` <-- `ptr_parours`

FIN



```
ALGORITHME: estvide
FONCTION : estvide(*ptr_liste : maillonListe) : BOOLEEN
DEBUT
    SI (ptr_liste = NULL) ALORS
        RETOURNER: Vrai
    FINSI
FIN
```



```
ALGORITHME: estvide
FONCTION : estvide(*ptr_liste : maillonListe) : BOOLEEN
DEBUT
    RETOURNER: ptr_liste = NULL
FIN
```