

Cours Algorithmes

28
05 2023

Pointeur
L'adresse.

alloquer ()
liberer ()

P : pointeur
 P^1 : contenu de la variable.

insérer élément dans une liste au début, au milieu, à la fin

Def type

élément = struct{
 info : Type.
 suivant : P^1 élément}

Fin élément

Def var

Tête, P^1 élément

Algorithme Insertion - début

Def type

Def var

Tête, New = P^1 élément

debut

si ($\text{Tete} = \text{Nil}$) alors

 afficher (Nouv)

 écrire ($\text{Nouv}^{\wedge} \cdot \text{info}$)

$\text{nouv}^{\wedge} \cdot \text{suivant} \leftarrow \text{Nil}$

sinon

 afficher (Nouv)

 écrire ($\text{Nouv}^{\wedge} \cdot \text{info}$)

$\text{Nouv}^{\wedge} \cdot \text{suivant} \leftarrow \text{Tête}$

$\text{Tête} \leftarrow \text{Nouv}$

fin si

Fin

Procédure Insertion -> debut ($\text{Tete}^{\wedge} \text{Element}$, $\text{val}^{\wedge} \text{Type}$)

~~debut~~

defvar

$\text{Nouv}^{\wedge} \text{Element}$

debut

 afficher (Nouv)

$\text{nouv}^{\wedge} \cdot \text{info} \leftarrow \text{Val}$

$\text{Nouv}^{\wedge} \cdot \text{suivant} \leftarrow \text{Tête}$

$\text{Tête} \leftarrow \text{Nouv}$

Fin

Insertion élément à la fin de la liste.

Procédure InserFin ($\text{Tete}^{\wedge} \text{Element}$, $\text{val}^{\wedge} \text{Type}$)

defvar

$\text{Nouv}^{\wedge} \text{Element}$

Debut

in den α -tête.

Tand que ~~while~~ ($\text{inden} \neq \text{Nil}$) faire

in den α — in den α . suivant

Procédure Insertion-Fn (tête : \wedge élément, val : Type)

defvar

P, Nouv : \wedge élément

Debut

si ($\text{tête} = \text{Nil}$) alors

insérer début (tête, val)

sinon

$P \leftarrow \text{tête}$

??

Tand que ($P^1\text{. suivant} \neq \text{Nil}$) faire

$P \leftarrow P^1\text{. suivant}$

fin tand que

allouer (Nouv)

Nouv $^1\text{. info} \leftarrow \text{val}$

Nouv $^1\text{. suivant} \leftarrow \text{Nil}$

$P^1\text{. suivant} \leftarrow \text{Nouv}$

Fin Si

$\frac{P}{P'}$

insertion au milieu d'une liste après élément cherché

Def var:

Tête = $\text{^t}\ell\text{ement}$,

ValP = Type

Tas

Algorithme insertion-milieu

Def var:

p, tête, Nouv : ℓ lement

valP : Type

debut

si ($\text{tête} = \text{nil}$) alors $\mathcal{E}\text{crire}$ ("la liste est vide")

sinon

$\mathcal{E}\text{crire}$ ("Donner la valeur à chercher")

Lire (valP)

P = tête

trouve = Faux

Tant que ($P \neq \text{nil}$ et trouve = Faux) faire

si ($P^1.\text{info} = \text{valP}$) alors

trouve = ~~Faux~~ vrai°

sinon

P = $P^1.\text{suivant}$

Fin Tant que

Fin Si

Si (trouve = vrai°) alors

appeler (Nouv)

Lire (Nouv. $^1.\text{info}$)

P1.suivant \leftarrow Nouv

Nouv. $^1.\text{suivant} \leftarrow P^1.\text{suivant}$

P1.suivant \leftarrow Nouv

sinon

lire ("le valP n'existe pas")

Fin

Suppression élément de la liste

Supprimer 1^{er} élément

Algorithme supp-debut

début

P : 1^{er} élément

debut

Si (tête = nil) alors Écrire ("la liste est vide")

sinon Pa tête

tête \leftarrow tête 1^e. suivant
Libérer (P)

Fin Si

Supprimer dernier élément

Algorithme supp-fin (tête : 1^{er} élément)

début

Si (tête = nil) alors Écrire ("Liste vide")

sinon

(Pa tête.

Pa tête 1^e. suivant

Tandis que (P 1^e. suivant \neq nil) faire

{ Pa P

Pa P 1^e. suivant

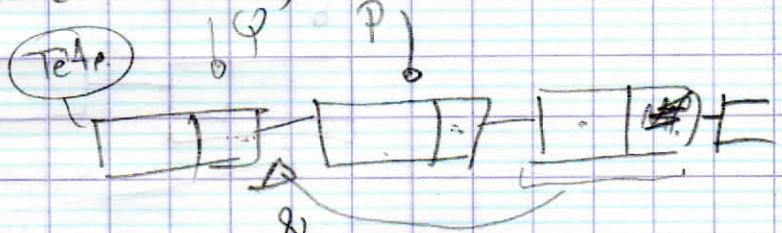
Fin Tandis que

Libérer (P)

Q 1^e. suivant \leftarrow nil.

Supprimer un élément chercher.

algorithme supp. (tête : ^ Element)



Algorithme supp. milieu (tête : ^ Element, Val : T)

debut

si (tête = nil) alors écrire ("Liste vide")

sinon

trouve \leftarrow Faux

P \leftarrow tête.

tant que ($P \neq \text{nil}$ et trouve = Faux) faire

si ($P^{\circ}.\text{info} \neq \text{val}$) alors

trouve \leftarrow Vrai

P $\leftarrow P^{\circ}.\text{suivant}$

P $\leftarrow P^{\circ}.\text{suivant}$

sinon

trouve \leftarrow Vrai

$P^{\circ}.\text{suivant} \leftarrow P^{\circ}.\text{suivant}.$

liberer (P)

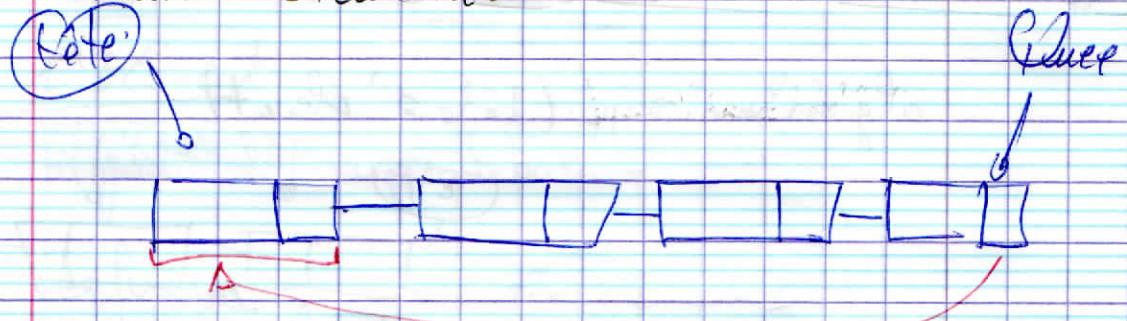
plus

fin Tant que.

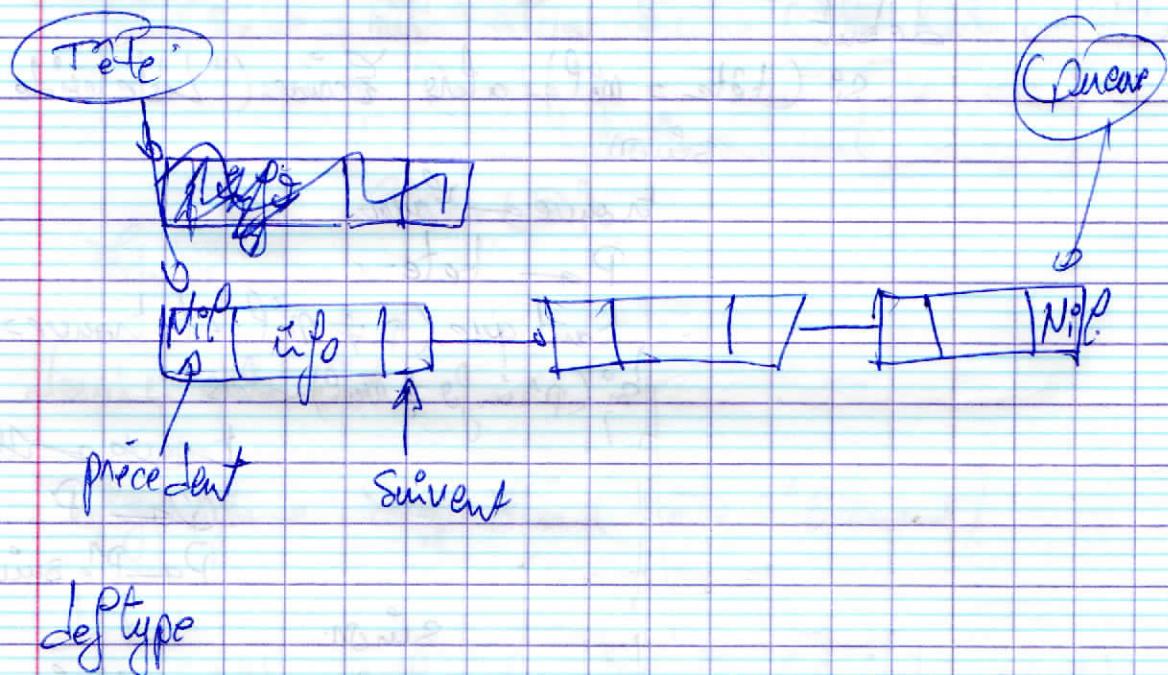
si (trouve = Faux) alors écrire ("L'élément n'existe pas")

fin

Liste Chaînée Circulaire



Liste Doublement Chaînée



def type

ElémentBi = structure

| avant : ^ ElémentBi

| info : T

| après : ^ ElémentBi

Fin structure

def var

tête, queue : ^ ElémentBi

Créer DoubList chain, 20 élément par insertion de but

Tête = afficher()

debut

afficher (tête)

lire (têteⁿ.info)

têteⁿ.avant \leftarrow Nil

têteⁿ.après \leftarrow Nil

queue \leftarrow tête

pour i de 1 à 19 faire

afficher (P)

lire (Pⁿ.info)

Pⁿ.après \leftarrow tête

Pⁿ.avant \leftarrow Nil

L₂₀ \leftarrow P

insertion élément à la fin (ma position queue)

Algorithme insertion-en-queue

Défvar:

debut

si (tête = Nil) alors

afficher (P)

lire (Pⁿ.info)

Pⁿ.avant \leftarrow Nil

Pⁿ.après \leftarrow Nil

L₂₀ \leftarrow P

queue \leftarrow P

Simon

A flouter (P)

($l \in (P^1.info)$)

$P^1.$ avant \leftarrow Queue

$P^1.$ après \leftarrow $N.C.$

Queue $^1.$ après $\leftarrow P$

Queue $\leftarrow P$

Suppression éléments — Cherché

Procédure suppression-milieu ($Tete^1$, élément, Queue, valeur, T)

Début

Si ($tete = N.C.$) alors écrire ("liste vide")

Simon

$P \leftarrow tete$

trouve \leftarrow False

Tant que ($P \neq N.C$ et trouve = False) faire

Si ($P^1.info = val$) alors

Si ($P = tete$) alors

$P \leftarrow tete$

$tete \leftarrow tete.suivant$
liberer (Q)

$tete^1.$ avant $\leftarrow N.C.$

Simon

Si $P = Queue$ alors

$P \leftarrow Queue$

Queue $^1.$ \leftarrow Queue $^1.$ avant

libérer (CP)

Queen: après \Leftarrow NiP

Simon

~~Disposer~~

(P¹-avant) Pⁿ-après \Leftarrow Pⁿ-après
libérer (P)

Simon

pa Pⁿ-après

Pisi?

Pisi?