**Exercice sur les listes chainée : Algorithme**

LEFAKONG TSOMELOU Vignol Dilane

Matricule : 22T2966 Niveau : L2 informatique

Définition des structures de donnée utilise dans les algorithmes qui vont suivre.

type liste = ^cellue ;

cellue = enregistrement

info : int ;

suivant : liste ;

fin

type d\_liste = ^d\_cellule;

d\_cellule = enregistrement

precedent : d\_liste ;

info : int ;

suivant : d\_liste ;

fin

1. Création d’une liste

a. Algorithme qui crée une liste avec deux entiers :

Algorithme : liste\_deux\_entiers

var : l : liste;

a,b:entier

debut :

écrire(“Entrer le premier entire: “) ;

lire(a) ;

écrire(“Entrer le deuxieme entire: “) ;

lire(b) ;

nouveau(l) ;

l->info = a ;

nouveau(l->suivant) ;

l->suivant->info = a ;

l->suivant->suivant = NULL ;

fin

b. Algorithme qui crée n entiers entres par l’utilisateur

Algorithme : liste\_n\_entiers

Var : l , iterateur :liste ;

a ,n ,i :entiers ;

debut :

l = NULL ;

ecrire(‘entrer la valeur de n : ‘) ;

lire(n) ;

pour i allant de 1 a n  faire

ecrire(‘Entrer un entier’) ;

lire(a) ;

si l = NULL alors

nouveau(l) ;

l->info = a ;

l->suivant = NULL ;

sinon

iterateur = l ;

tantque iterateur->suivant != NULL faire

iterateur = iterateur->suivant ;

fintantque

nouveau(iterateur->suivant)

iterateur->suivant->info = a ;

iterateur->suivant->suivant = NULL ;

finsi

finsi

fin

c. Algorithme qui s’achevé lorsque l’utilisateur saisir -1

Algorithme : liste\_saisir

Var l, iterateur : liste

a :entier

debut

ecrire(‘pour arrêter la saisie veuillez taper -1’) ;

repeter

ecrire(‘Entrer un entier : ‘);

lire(a) ;

si a != -1 alors

si l = NULL alors

nouveau(l) ;

l->info = a ;

l->suivant = NULL ;

sinon

iterateur = l ;

tantque iterateur->suivant != NULL faire

iterateur = iterateur->suivant ;

fintantque

nouveau(iterateur->suivant)

iterateur->suivant->info = a ;

iterateur->suivant->suivant = NULL ;

finsi

finsi

jusqu’à(a == -1) ;

fin

2. Recherche d’un élément dans une liste chainée

a. procédure qui recherche une valeur dans une liste chainée

procedure rechercher\_valeur(l :liste, valeur :entier)

var  iterateur : liste ;

tr :entier

debut

iterateur = l ;

tr = 0 ;

tantque tr != 1 et iterateur != NULL  faire

si iterateur->info = valeur alors

tr = 1

finsi

iterateur = iterateur->suivant ;

fintantque

si tr = 1 alors

ecrire(‘L’élément a été trouvé.’) ;

sinon

ecrire(‘L’élément n’a pas été trouvé.’)

finsi

fin

b. Procédure qui recherche la dernière occurrence

procedure rechercher\_dernier\_occurrence(l :liste, valeur :entier)

var iterator : liste ; pos,i :entier ;

debut

iterateur = l ;

pos = -1 ;

i = 1 ;

tantque iterateur != NULL faire

si iterateur->info = valeur alors

pos = i ;

finsi

i = i+1 ;

iterateur = iterateur->suivant ;

fintantque

si pos = -1  alors

écrire(‘l’élément n’a été trouvé.’) ;

sinon

écrire(‘l’élément a été trouvé et la dernière occurrence est la position’, pos ) ;

finsi

fin

3. Suppression d’élément dan une liste chainée

a. Procédure qui supprime le premier élément de la liste

procedure supprimer\_premier(l :^liste)

var t : liste ;

debut

si ^l != NULL alors

t = ^l ;

(^l) = t->suivant ;

t->suivant = NULL ;

liberer(t) ;

finsi

fin

b. Procédure qui supprime la première occurrence d’une valeur donnée

procedure supprimer\_premiere\_occurrence(l :^liste, valeur : entier)

var t ,prec: liste ;

debut

si ^l != NULL alors

t = ^l ;

si (^l)->info = valeur alors

(^l) = (^l)->suivant ;

liberer(t)

Sinon

prec = (^l) ;

t = prec->suivant ;

tantque t != NULL et t->info != valeur faire

prec = t ;

t = t->suivant ;

fintantque

si t != NULL alors

prec->suivant = t->suivant ;

t->suivant = NULL ;

liberer(t) ;

finsi

finsi

finsi

fin

c. Procédure qui supprime toutes les occurrences d’une valeur donnée

procedure supprime\_toutes\_occurrence(l :^liste, valeur :entier)

var itérateur, prec, element : liste ;

debut

si ^l != NULL alors

itérateur = ^l ;

prec = ^l ;

tantque iterateur != NULL faire

si itérateur->info = valeur alors

si itérateur ! = ^l alors

element = itérateur ;

prec->suivant = itérateur->suivant ;

element->suivant = NULL ;

liberer(element) ;

iterator = prec->suivant ;

sinon

element = ^l ;

^l = (^l)->suivant ;

element->suivant =NULL ;

liberer(element) ;

iterateur = ^l ;

prec = ^ l ;

finsi

sinon

prec = iterateur ;

itérateur = itérateur->suivant ;

finsi

fintantque

finsi

fin

4. Liste doublement chainée

a. procédure qui affiche une liste doublement chainee dans l’ordre d’enregistrement

procedure affiche\_ordre\_enregistrement(l : d\_liste)

var iterateur : d\_liste ;

début

si l != NULL alors

itérateur = l ;

tantque itérateur != NULL faire

ecrire(‘[’, itérateur->info ,’]’) ;

iterateur = itérateur->suivant ;

fintantque

finsi

fin

b. procédure qui affiche une liste doublement chainee dans l’ordre inverse d’enregistrement

procedure affiche\_ordre\_inverse\_enregistrement(l : d\_liste )

var iterateur : d\_ liste ;

debut

si l != NULL alors

iterateur = l ;

tantque iterateur->suivant != NULL faire

iterateur = iterateur->suivant ;

fintantque

tantque iterateur != NULL faire

ecrire(‘[’, iterateur->info ,’]’) ;

iterateur = iterateur->precedent ;

fintantque

finsi

fin