**Structures de données**

**Corrigé Examen2  // 2009 2010**

**Arbres**

1. Les prochains préordre ( N T1 T2) de chaque élément de l’arbre

|  |  |  |
| --- | --- | --- |
|  | Prochain Préordre |  |
| 34 | 22 |
| 67 | 56 |
| 87 | 90 |
| 22 | 15 |
| 15 | 5 |
| 56 | 60 |
| 5 | 67 |
| 60 | 58 |
| 58 | 87 |
| 90 | - |

1. Algorithme de recherche qui construit en même temps la pile des nœuds traversés.

P := Arbre ; Trouv := faux ;

Creerpile(Pile) ; // Initialisation

Tq P <> Nil et Non Trouv

Si info(P) = val

Trouv := vrai

Sinon

Empiler(Pile, P)

Si val < Info(P)

P := Fg(P)

Sinon

P := Fd(P)

Fsi

Fsi

Ftq

1. Algorithme qui donne le prochain Préordre d’un nœud donné en utilisant la pile.

APPEL Recherche ( Arbre , Val , P , Trouv, Pile ) ; // P adresse du nœud trouvé et Pile la pile

SI NON Trouv

ECRIRE ( 'Valeur inéxistante ' )

SINON

SI FG ( P ) <> NIL

ECRIRE ( 'Suivant préordre =' , INFO ( FG ( P ) ) ) ;

SINON

SI FD ( P ) <> NIL

ECRIRE ( ' Suivant préordre =' , INFO ( FD ( P ) ) ) ;

SINON

Depiler(Pile, Q) ;

Continue := VRAI ;

TANTQUE ( Q <> NIL ) ET Continue

SI P = FD ( Q )

P := Q ;

Depiler(Pile, Q)

SINON

SI FD ( Q ) <> NIL

Continue := FAUX

SINON

P := Q ;

Depiler(Pile, Q) ;

FSI

FSI

FTQ ;

SI Q <> NIL

ECRIRE ( ' Suivant préordre =' , INFO ( FD ( Q ) ) )

SINON

ECRIRE ( 'Pas de Suivant Préordre' )

FSI

FSI

FSI

FSI

**Graphes**

1. Séquence des opérations permettant de créer le graphe ci-après :

|  |  |
| --- | --- |
| CréerNoeud(G, u1) ; CréerNoeud(G, u2) ; CréerNoeud(G, u3) ; CréerNoeud(G, u4) ; Aff\_info(u1,A) ; Aff\_info(u2,B) ; Aff\_info(u3,D) ;  Aff\_info(u4,E) ;  Aff\_Arc(u1, u2, v1) ; Aff\_Arc(u1, u3, v3) ; Aff\_Arc(u3, u2, v2) ; |  |

1. Parcours d’un graphe quelconque G en DFS (Depht First Search).

Utilise un tableau MARK initialisé à ‘Non Visité’.

Initialisation : Pour i :=1, *NbrGraphe(G)* : Mark[i] := ‘Non Visité’ Fpour

Module principal :

Pour i :=1 to *NbrGraphe(G)*

Si Mark[i]= ‘Non Visité’ : DFS(*NoeudGraphe ( G, i )*) Fsi

Fpour

Module DFS pour un nœud donné :

DFS(v)

Mark[i] := ‘Visité’

Pour i := 1, *Degré(v) :* DFS(*Adjacent(v, i)*) Fpour

**Implémentation du graphe en C**

/\* Graphes \*/

struct Typegraphe

{

struct Typeliste \*Prem;

struct Typeliste \*Dernier ;

int Nbr;

}

/\* Listes \*/

struct Typeliste

{

int Valnoeud ; /\* Valeur rattachée au noeud \*/

int Degre ; /\* Degré du noeud \*/

struct Typelistearc \*List; /\* Liste des noeuds adjacents contenant les valeurs des arcs \*/

struct Typeliste \*Suivant;

} ;

struct Typelistearc

{

int Valarc ; /\* Valeur rattachée à l'arc \*/

struct Typenoeud \*Noeud;

struct Typelistearc \*Suivant ;

}

**Implémentation de quelques opérations sur les graphes**

int Nbrgraphe ( struct Typegraphe G )

{ return ( G.Nbr) ; }

struct Typenoeud \*Noeudgraphe(struct Typegraphe G, int I )

{

struct Typeliste \*N;

int K;

N = G.Prem;

K =1;

while ( (K<I) && (N != NULL) )

{

K = K+1;

N=N->Suivant ;

};

return N->Element;

}

int Degre( struct Typenoeud \*U )

{

return ( U->Degre );

}

struct Typenoeud \* Adjacent(struct Typenoeud \*U, int I )

{

struct Typelistearc \*Ptarc;

int K ;

Ptarc = U->List;

K=1;

while (K<I && Ptarc != NULL )

{

K++;

Ptarc = Ptarc->Suivant;

}

return Ptarc->Noeud;

}