Université de Bejaia
Faculté des sciences exactes
Département M/MI
1ère année MI
Module PSSD

Nom :
Prénom :
Groupe :
Emargement:
Lundi 19/05/2014, Durée : 2 heures

Exercice 01 :

```
algorithme exo_01 ;
    Type
        client = enregistrement
                    numc : entier ;
                    nom, pre : chaine[25];
                    solde : reel;
                    adr : chaine[25];
                    Fin ;
       Tabc=tableau[1..200] de client ;
        typet=(credit,debit,consu) ;
        trans = enregistrement
                    numt : entier ;
                    numcl : entier ;
                    type : typet;
                    date : chaine[10];
                    mont: reel;
                    Fin ;
        Tabt=tableau[1..200] de trans ;

    Var     tac :tabc ; tat : tabt;
     Procedure      Cremplir (var TC: tabc ; nc: entier) ;
          Var               i: entier ;
            Debut
            |    Pour  i  allant de   1  jusqu'à   nc   faire
            |    |  avec TC[i] faire
            |    |  |       lire ( nom, pre, solde, adr);
            |    |  |       numc ←i ;
            |    |  finavec ;
            |    |finpour ;
            |fin ;
     Procedure      Tremplir (var TT: tabt ; nt: entier) ;
          Var               i: entier ;
            Debut
            |    Pour  i  allant de   1  jusqu'à   nt   faire
            |    |  avec TT[i] faire
            |    |  |       lire ( numcl, type, date, mont);
            |    |  |       numt ←i ;
            |    |  finavec ;
            |    |finpour ;
            |fin ;
```

```
Procedure     crediter (var sol :reel ; mnt :reel) ;
        Debut
        |       sol ←sol+mnt ;
        |fin ;
Procedure     debiter (var sol :reel ; mnt :reel) ;
        Debut
        |       sol ←sol-mnt ;
        |fin ;
fonction    consulter  (TC:tabc ; nc :entier ; numclt :entier) :reel ;
       Var  i :entier ; trouve:booleen ;
        Debut
        |    i← 1;
        |    trouv ← faux  ;
        |    tantque    ((i≤nc) et non trouv )  faire
        |    |        avec TC[i] faire
        |    |        |  Si    (numc= numclt )  alors
        |    |        | trouv← vrai ;
        |    |        |sinon
        |    |        |       i ← i+1 ;
        |    |        |finsi ;
        |    |fintantque ;
        |    si (trouve=vrai) alors consulter ←TC[i].solde;
        |    |sinon      consulter ←-1;
        |    |finsi;
        |fin ;

fonction    existe  (TC:tabc ; nc :entier ; noc,prc :chaine[25]) :booleen;
       Var  i :entier ; trouve:booleen ;
        Debut
        |    i← 1;
        |    trouv ← faux  ;
        |    tantque    ((i≤nc) et non trouv )  faire
        |    |        avec TC[i] faire
        |    |        |  Si    (nom= noc) et  (pre= prc)  alors
        |    |        | trouv← vrai ;
        |    |        |sinon
        |    |        |       i ← i+1 ;
        |    |        |finsi ;
        |    |fintantque ;
        |     existe ← trouve;
        |fin ;
```

```
Procedure      ajouter (var TC: tabc ; nc: entier ; noc,prc :chaine[25])) ;
      Var              i: entier ;  rep :booleen;
        Debut
        |    rep ←existe  (TC, nc, noc,prc);
        |    si (rep= vrai) alors      ecrire('client existe déjà') ;
        |    |sinon
        |    |  si (nc=200) alors      ecrire('on ne peut pas rajouter le client') ;
        |    |  |sinon
        |    |  |   avec TC[nc+1] faire
        |    |  |   |    lire (solde, adr);
        |    |  |   |    numc ←nc+1 ;
        |    |  |   |    nom ←noc ;
        |    |  |   |    pre ←prc ;
        |    |  |   |finavec ;
        |    |  | finsi;
        |    |finsi;
        |fin ;

fonction    comparer  (D1,D2 :chaine[25]) :booleen;
       Var  i :entier ; trouve:booleen ;
        Debut
        |    j1←copier(D1,1,2);
        |    j2←copier(D2,1,2);
        |    m1←copier(D1,4,2);
        |    m2←copier(D2,4,2);
        |    a1←copier(D1,7,4);
        |    a2←copier(D2,7,4);
        |   //comparer deux date, on retourne vrai si c'est supérieur ou égal sinon faux.
        |fin ;

Procedure      afficher ( TC: tabc ; nc: entier ; TT :tabt ; nt :entier ; D1,D2 :chaine[10])) ;
      Var              i: entier ;  rep1,rep2 :booleen;
        Debut
        |    rep1 ← comparer  (D1,D2);
        |    Pour  i  allant de    1  jusqu'à    nt   faire
        |    |  avec TT[i] faire
        |    |  | rep1 ← comparer  (date,D1); rep2 ← comparer  (D2, date);
        |    |  |   si (rep1 et rep2) alors
        |    |  |   |  ecrire ( numt,numcl, type, date, mont);
        |    |  |   | j← 1;
        |    |  |   | trouv ← faux  ;
        |    |  |   | tantque   ((i≤nc) et non trouv )  faire
        |    |  |   | |         avec TC[i] faire
        |    |  |   | |         |  Si   (numc=numcl) alors ecrire ( non,pre); trouve ← vrai finsi ;
        |    |  |   | |         |sinon   i ← i+1 ;
        |    |  |   | |         |finsi ;
        |    |  |   | |fintantque ;
        |    |  |   |finsi ; finavec ; finpour ;fin ;
```

```
Procedure     Ccrediter ( TC: tabc ; nc: entier ; TT :tabt ; nt :entier ; numclt :entier) ;
    Var              i: entier ;  rep1,rep2 :booleen;
      Debut
      |       i← 1;
      |      trouv ← faux  ;
      |      tantque    ((i≤nc) et non trouv )  faire
      |      |    avec TC[i] faire
      |      |  |     |Si    (numc=numclt) alors lire ( mnt); crediter (solde, mnt) ; trouve ← vrai
      |      |  |     |                              avec TT[nt+1] faire
      |      |  |     |                              | numt ← nt+1 ; numcl ← numclt ;
      |      |  |     |                              | lire(date,mont,type) ;
      |      |  |     |                              |finavec ;
      |      |  |     |
      |      |  |     |sinon    i ← i+1 ;
      |      |  |     |finsi ; finavec ;
      |      |fintantque ;
      |fin ;

      Debut
      |       lire(nbc,nbt);
      |      Cremplir (tac,nbc) ; Tremplir (tat,nbt) ;
      |      lire (nocl,prcl) ;
      |     ajouter (tac, nbc, nocl,prcl) ;
      |     lire (Da1,Da2) ;
      |     afficher (tac, nbc, tat, nbt,Da1,Da2) ;
      |     lire(numclt) ;
      |     Ccrediter ( tac, nbc,tat,nbt,numclt) ;
      | fin .
```

Exercice 2:

```
algorithme exo_02;
    Type   pointeur = ↑log ; categ :(eco,soc,nor) ;
        Tlog = enregistrement
                    ident: entier ;
                    cat:categ;
                    pri: reel ;
                    loc: chaine[50] ;
                    Suivant : pointeur ;
                    Fin ;
    Var     L :pointeur ;

    Procedure     creation (var liste : pointeur) ;
        Var             P   : pointeur ; N :entier ;

          Debut
          |
          |    Lire (N) ;
          |    liste← Nil ;
          |    Pour  i  allant de   1  jusqu'à   N   faire
          |    |
          |    |        allouer (P)   ;
          |    |  avec P faire
          |    |  |      lire (ident,cat,pri,loc)   ;
          |    |  |      suivant ←liste ;
          |    |  finavec ;
          |    |       liste←  P ;
          |    |finpour ;
          |finfin ;

        procedure    afficher (liste : pointeur) ;
        Var             P   : pointeur ;
          Debut
          |    P← liste;
          |    tantque   (P<> Nil)   faire
          |    |        Si   (P↑.cat= eco )  alors
          |    |        | ecrire (P↑.ident, P↑.cat, P↑.pri, P↑.loc)   ;
          |    |        |sinon
          |    |        |       P ← P↑.suivant ;
          |    |        |finsi ;
          |    |fintantque ;
          |finfin ;
```

```
procedure     supprimer (var liste : pointeur ; id :entier) ;
              Var              P, R   : pointeur ;
                Debut
                |      P ← liste;
                |   si (P↑.ident =id)  alors
                |   |                                liste ← liste↑.suivant ;
                |   |                                liberer (P) ;
                |   |sinon
                |   |  R← P↑.suivant ;
                |   |  Tantque   (R <> NiL)    faire
                |   |  |    si (R↑.ident =id)  alors
                |   |  |    |                          si (R↑.suivant<>Nil) alors
                |   |  |    |                          |                  P↑.suivant ← R↑.suivant;
                |   |  |    |                          |                  liberer (R ) ;
                |   |  |    |                          |sinon
                |   |  |    |                          |       P↑.suivant ← Nil;
                |   |  |    |                          |       liberer (R ) ;
                |   |  |    |                          finsi ;
                |   |  |    |finsi ;
                |   |  |         P ← R;
                |   |  |         R ← R↑.suivant;
                |   |  |fintantque ; finsi ; fin ;

            procedure     creer ( liste,Le,Ls,Ln : pointeur ) ;
              Var              Pe, Ps, Pn   : pointeur ;
                Debut
                |      Li ← liste;Le ← Nil;  Ls ← Nil; Ln ← Nil;
                |    Tantque   (Li <> NiL)    faire
                |    |    si (Li↑.cat=eco) alors
                |    |    |                                        Pe ← Li;
                |    |    |                                        Pe↑.suivant ← Le;
                |    |    |                                        Le←Pe ;
                |    |    sinon
                |    |    |   si (Li↑.cat=soc) alors
                |    |    |   |                                    Ps ← Li;
                |    |    |   |                                    Ps↑.suivant ← Ls;
                |    |    |   |                                    Ls←Ps ;
                |    |    |    sinon
                |    |    |    |    Pn ← Li;
                |    |    |    |    Pn↑.suivant ← Ln;
                |    |    |    |    Ln←Pn ;
                |    |    |    |finsi ;
                |    |    ||finsi ;
                |    |    Li ← Li↑.suivant;
                |    |fintantque ;
                |      Li ← Li↑.suivant;
                |fin ;
```

```
procedure     ajouter (var liste : pointeur ; id :entier) ;
Var           P, Q   : pointeur ; inter :entier ;
  Debut
  |     P ← liste;  lire(v)
  |   si (P↑.ident > id)  alors  allouer (Q) ;
  |   |                          lire (Q↑.ident, Q↑.cat, Q↑.pri, Q↑.loc )  ;
  |   |                          Q↑.suivant ← P↑.suivant ;
  |   |                          liste ← Q ;
  |   |sinon
  |   |  R← P↑.suivant ; trouve←faux ;
  |   |  Tantque   (R <> NiL) et non trouve    faire
  |   |  |   R←P↑.suivant ;
  |   |  |   si (P↑.ident <id) et (R↑.ident > id) alors      allouer (Q) ;
  |   |  |   |                                              lire (Q↑.ident, Q↑.cat, Q↑.pri, Q↑.loc )  ;
  |   |  |   |                                                    Q↑.suivant ← P↑.suivant;
  |   |  |   |                                                    P↑.suivant ← Q↑.suivant;
  |   |  |   |                                                    trouve←vrai ;
  |   |  |   sinon
  |   |  |   |      R ← R↑.suivant;
  |   |  |   |      P ← P↑.suivant;
  |   |  |   |finsi ;
  |   |  |fintantque ;
  |   |  si (trouve =faux) alors    lire (Q↑.ident, P↑.cat, P↑.pri, P↑.loc )  ;
  |   |  |                          Q↑.suivant ← Nil;
  |   |  |                          P↑.suivant ← Q;
  |   |  | finsi ;
  |   |finsi ; fin ;
debut
|     creation (L) ;  lire (iden) ;
|     afficher (L) ;
|     supprimer (L, iden) ;
|     créer (L,Le,Ls,Ln);
|     lire(cate) ;
|     si (cate=eco)  alors lire (iden) ; ajouter (Le, iden) ; finsi ;
|     si (cate=eco)  alors lire (iden) ; ajouter (Ls, iden) ; finsi ;
|     si (cate=eco)  alors lire (iden) ; ajouter (Ln, iden) ; finsi ;
|fin.
```