SOLUTION de TD N°4 : LES LISTES LINEAIRES CHAINEES

Exercice 1 : Création et parcours

1. Ecrire un algorithme qui construit la liste L à partir de n données lues, ensuite affiche ses éléments.

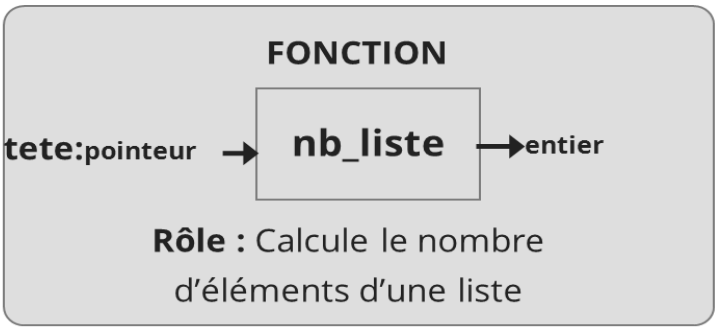
Solution :

```
Algorithme CreerListe;
Type maillon = Enregistement
               Val : entier ;
               Adr : Pointeur (maillon)
               Fin ;

Var  Tete, P, Q : Pointeur(maillon) ;
     i, N, Val : entier ;

Début
   Tete ← Nil;
   P ← Nil;
   Ecrire('Donner le nombre N :');
   Lire (N);

   //1. Remplir la liste

   Pour i de 1 à N Faire
      Lire(Val) ;
      Allouer(Q) ;
      Aff_val(Q, val) ;          //ou bien : Q^.Val ← Val
      Aff_adr(Q, NIL) ;          //ou bien : Q^.Adr ← NIL
      Si (Tete <> Nil) Alors
            Aff_adr(P, Q)        //Chainer le nouveau élément Q avec le dernier élément P
      Sinon
            Tete ← Q             //Q est le premier élément donc changer la Tete seulement
      FSi;
      P←Q;                       //sauvegarder le dernier élément dans P
   Fpour

   //2. Afficher les éléments de la liste

   P ⟵ Tete;
   TQ (P <> Nil) Faire
      Ecrire(Valeur(P)) ;        //ou bien : Ecrire(P^.Val);
      P ⟵ Suivant(P);            //ou bien : P ⟵ P^.Adr;
   Fpour;

FIN.
```

Exercice 2 : Nombre d'éléments

1. Ecrire une **fonction** qui calcule le nombre d'éléments de la liste L.

Solution :



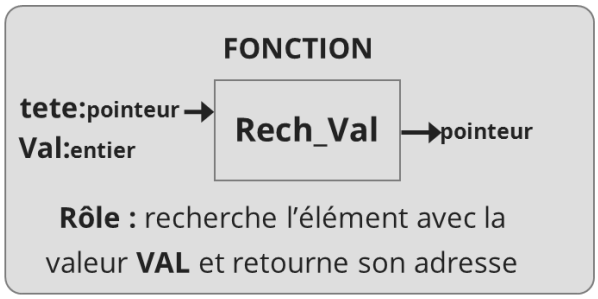
Fonction nb_liste:

```
Fonction  nb_liste (tete: Pointeur(maillon)): entier;
Var   nbr: entier;
      P: pointeur(maillon)
Début
   P ← tete;
   nbr ← 0;
   TQ P <> Nil Faire
      nbr ← nbr +1;
      P ← Suivant (P);     //ou bien : P ← P^.Adr;
   FTQ
nb_liste ← nbr ;
Fin;
```

Exercice 3 : Rechercher Par valeur

1. Ecrire une **fonction** qui retourne l'adresse (pointeur) de l'éléments avec la valeur VAL s'il existe sinon, retourne Nil.

Solution :



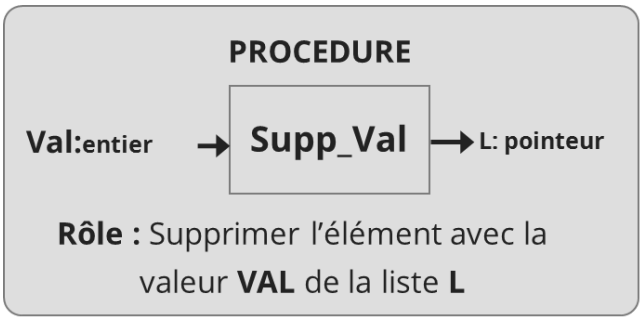
Fonction Rech_Val:

```
Fonction  Rech_Val (tete: Pointeur(maillon) , VAL:entier): Pointeur;
Var
      P: pointeur(maillon)
Début
   P ← tete;
   TQ P <> Nil et Valeur(P) <> VAL Faire
      P ← Suivant (P);     //ou bien : P ← P^.Adr;
   FTQ

   Si P = Nil Alors Rech_Val ← Nil;
   Sinon            Rech_Val ← P;

Fin;
```

Exercice 4 : Supprimer par Valeur

1. Ecrire une **procédure** qui supprime un élément qui a la valeur **Val** dans la liste **L** s'il existe.

Solution :



Fonction Supp_Val:

```
Procédure  Supp_Val (VAL:entier; VAR L: Pointeur(maillon));
Var   P, Q: pointeur(maillon) ;      // P : pointeur courant
                                     // Q : pointeur du précédant
Début
   P ← L;
   Q ← Nil;
   TQ P <> Nil et Valeur(P) <> VAL Faire
      Q ← P;                         //Sauvegarder le précédant de P
      P ← Suivant (P);               //Avancer P à l'élément suivant
   FTQ

   Si P <> Nil Alors                 //L'élément avec la valeur VAL existe !
      Si Q <> Nil Alors              //Supprimer un élément au milieu
         Aff_Adr(Q, Suivant(P)); //Chainer le précédant avec le suivant de p
      Sinon                          //Supprimer le 1er élément (la tete)
         L ← Suivant(P);             //Avancer la tete au suivant de P
      Fsi

      Liberer(P);                    //Supprimer le maillon pointé par P
   Fsi
Fin;
```

Exercice 5 : (Examen PSD 2013-2014)

Soit L une liste linéaire chainée qui est utilisée pour stocker les moyennes des étudiants de 1ère année du département de Mathématique et Informatique.

1. Ecrire un algorithme qui permet d'afficher le nombre des étudiants admis en deuxième année.

Solution :

Algorithme : Nombre d'étudiants admis

```
Algorithme nbr_admis ;
Type mailon = Enregistement
              Val : réel ;
              Suivant : Pointeur (maillon)
              Fin ;

Var  L, P : Pointeur ;
     nbr : entier ;

Debut
    // Remplir la liste L
    nbr := 0 ;
    P := L ;
    TQ (P <> Nil) Faire
       DTQ
           Si (Valeur(P) >= 10) Alors nbr := nbr  + 1;
           P := Suivant (P) ;
       FTQ
    Ecrire ('Le nombre des étudiants admis est ',nbr ) ;
Fin.
```

2. Ecrire un algorithme qui supprime tous les étudiants abondants (qui ont une moyenne égale à zéro).

Solution :

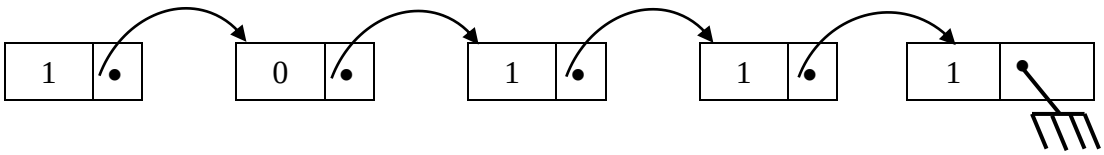
Algorithme : Suppression des étudiants abondants

```
Algorithme supp_abonds ;
Type mailon = Enregistement
              Val : réel ;
              Suivant : Pointeur (maillon)
              Fin ;

Var  L, P, Q, R : Pointeur ;

Debut
    // Remplir la liste L
    P := L ;
    Q := Nil ;
    TQ (P <> Nil) Faire
       DTQ
           Si (Valeur(P) = 0) Alors
              DSi
                     Si (Q = Nil) Alors
                            L := Suivant (P) ;
                     Sinon
                            Aff_Adr(Q, Suivant(P)) ;
                     FSi
                     R := P ;
              FSi
           Q := P ;
           P := Suivant (P) ;
           Libérer(R) ;
       FTQ
Fin.
```

Exercice 6 : (Examen PSD 2015-2016)

Soit N un nombre entier écrit sous la base 10.

1. Ecrire un algorithme qui permet de convertir N en binaire et le stocker dans une liste linéaire chainée chiffre par chiffre.
   Exemple : Si N = 29 ; alors $(29)_{10} = (11101)_2$



1. Ecrire un algorithme qui permet d'afficher le nombre des étudiants admis en deuxième année.

Solution :

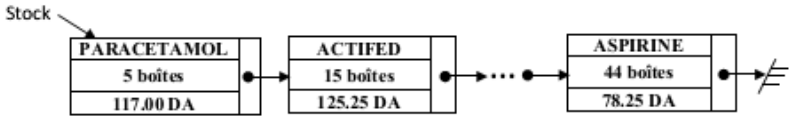
L'Algorithme Principale

```
Algorithme list_binaire ;
   Type maillon = Enregistement
              val : entier ;
              suiv : Pointeur (etudiant)
         Fin ;

   Var  L, P, Q : Pointeur (etudiant);
        N, R : entier ;

    Fonctions     Moy_section;
    Procédures    Ordonner, Affiche_admis;
Debut
    // 1. Convertir N et remplir la liste
    Lire(N) ;
    L := Nil ;
    TQ (N <> 0) faire
       DTQ
           R := N MOD 2 ;
           N := N DIV 2 ;
           Allouer(Q) ;
           Q^.val := R ;
           Q^.suiv := Nil;

           Si (L = Nil) Alors  L := Q ;
           Sinon               P^.suiv := Q;
           P := Q;
       FTQ

Fin.
```

Exercice 7 : Le pharmacien

Un pharmacien souhaite traiter les informations concernant son stock de médicaments par ordinateur. On vous propose de représenter ces informations sous forme de liste linéaire chainée ou chaque maillon contient le libellé d'un médicament, la quantité disponible (nombre de boites) et le prix unitaire.

Exemple :



1. Donner les structures de données nécessaires à la représentation de ce stock.

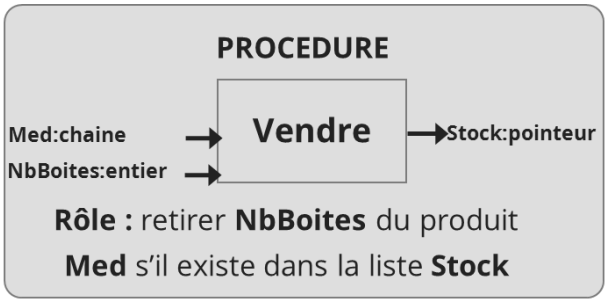
Solution 01: Déclaration des Structures de données et remplissage de la liste des produits

```
Algorithme  CreerListeParmacie;
    Type     produit=enregistrement
                 libelle:chaine;
                 Qte:entier;
                 Prix:réel;
                 Adr:pointeur(produit)
             Fin;
  Var   Tete,P,Q: pointeur(produit);
        i, N : entier ;
Début
    Tete ← Nil;
    P ← Nil;
    Ecrire('Donner le nbr de produits N :');
    Lire (N);
```

```
//Remplir  la liste
Pour i de1 à N faire
     Allouer(Q) ;
     Lire (Q^.libelle);
     Lire (Q^.Qte);
     Lire (Q^.Prix);
      Q^.Adr ← NIL;
     Si (Tete <> Nil) Alors
       P^.Adr ← Q;
     Sinon
       Tete ← Q
     FSi;
     P←Q;
  FPour;
Fin.
```

2. Ecrire la procédure **Vendre (Stock, Med, NbBoites)** permettant de retirer, si possible, 'Nb-Boites' du médicament 'Med' du stock. **Attention** : Il faut supprimer du stock le médicament dont la quantité atteint 0.

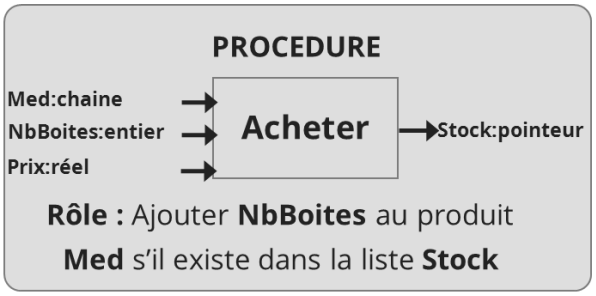
Solution 02: procédure Vendre (Stock, Med, NbBoites)




```
Procédure  Vendre (Med:chaine;NbBoites:entier;VAR Stock: Pointeur);
Var    P, Q: pointeur(maillon)
Début
    P ← Stock; Q ← Nil;
    TQ P <> Nil et P^.libelle <> Med Faire
         Q ← P;
         P ← Suivant (P);
    FTQ
    Si  P <> Nil Alors
         Si P^.Qte > NbBoites Alors
             P^.Qte ← P^.Qte – NbBoites;
         Sinon
             Si Q <> Nil Alors   Q^.Adr ← P^.Adr;
             Sinon               Stock ← P^.Adr;
         Fsi
         Liberer(P);
    Fsi
Fin;
```

3. Ecrire la procédure **Acheter (Stock, Med, NbBoites, Prix)** permettant au pharmacien d'alimenter son stock par 'NbBoites' du médicament 'Med' ayant le prix unitaire 'Prix' DA.

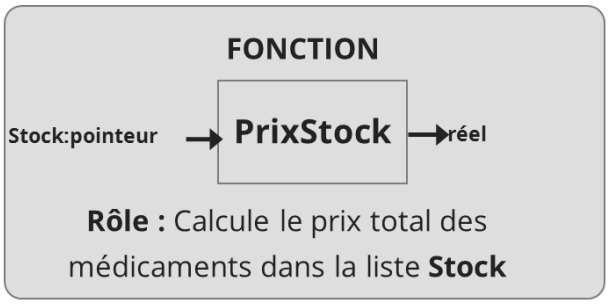
Solution 03: procédure Acheter (Stock, Med, NbBoites, Prix)



```
Procédure  Acheter(Med:chaine; NbBoites:entier; Prix : Reel;
             VAR Stock: Pointeur(produit) );
Var    P, Q, R: pointeur(maillon)
Début
    P ← Stock; Q ← Nil;
    TQ P <> Nil et P^.libelle <> Med Faire
         Q ← P;
         P ← Suivant (P);
    FTQ
    Si  P <> Nil Alors
         P^.Qte ← P^.Qte + NbBoites;
    Sinon
         Allouer(R);
         R^.libelle ← Med; R^.Qte ← NbBoites; R^.Prix ← Prix;
         Si Stock = Nil Alors    Stock ← R;
         Sinon                   Q^.Adr ← R;
         Fsi
    Fsi
Fin;
```

4. Ecrire la fonction **PrixStock(Stock)** permettant de calculer le prix total des médicaments dans le stock.

Solution 04: fonction PrixStock (Stock)




```
Fonction PrixStock(Stock: Pointeur(produit) ): réel;
Var    P : pointeur(maillon);
       S: Réel;
Début
    P ← Stock;
    S ← 0;
    TQ P <> Nil Faire
         S ← S + P^.Qte * P^.Prix;
         P ← Suivant (P);
    FTQ
    PrixStock ← S;
Fin;
```

Exercice 8 : La bibliothèque (Examen PSD 2016-2017)

Une bibliothèque souhaite stocker les informations de ces ouvrages (livres) dans une liste linéaire chainée. Chaque livre est représenté par son titre, son auteur, son prix et le nombre d'exemplaires de ce livre.

1. Ecrire un algorithme qui permet de ***construire la liste*** « ***biblio*** » à partir de N livre donnés.

Un algorithme qui permet de construire la liste « biblio » à partir de N livre donnés.

```
Algorithme bibliotheque ;
   Type livre = Enregistement
              titre : chaine[30];
              auteur : chaine;
              prix : réel ;
              nbr : entier ;
              adr : Pointeur (livre)
         Fin ;

   Var  biblio, P, Q : Pointeur (livre);
        N : entier ;
```

```
Debut
    // 1. Remplir la bibliothèque par des livres
    Lire(N) ;
    biblio := Nil ;
    Pour (i := 1 à N) faire
       DPour
           Allouer(Q) ;
           Lire (Q^.titre);
           Lire (Q^.auteur);
           Lire (Q^.prix);
           Lire (Q^.nbr);
           Q^.suiv := Nil;

           Si (biblio = Nil) Alors    biblio := Q ;
           Sinon                      P^.adr := Q;
           P := Q;
       FPour
Fin.
```

2. Ecrire la fonction **nbr_livres(biblio)** qui permet de calculer le nombre d'exemplaires total;

La fonction nbr_livres(biblio) qui permet de calculer le nombre d'exemplaires total;

```
Fonction nbr_livres (VAR biblio : Pointeur) : entier ;
    Var P : Pointeur (etudiant);
        nb: entier ;

Debut
    nb:= 0;
    P := biblio ;
    TQ (P <> Nil) faire
       DTQ
            nb := nb + P^.nbr;
            P := P^.adr;
       FTQ
    nbr_livres := nb;
Fin.
```

3. Ecrire la procédure **ajout_livre(biblio, t, aut, p, nb)** qui permet d'ajouter « **nb** » exemplaires au livre qui a le titre « **t** » (l'auteur « **aut** » et le prix « **p** ») s'il existe. Dans le cas où le livre n'existe pas, on l'ajoute à la fin de la liste ;

La procédure : ajout_livre(biblio, t, aut, p, nb)

```
Procedure ajout_livre (VAR biblio : Pointeur;t,aut:chaine;p:réel;nb:entier);
    Var P, Q, Prec : Pointeur (Livre);
        trouve : booleen;

Debut
    trouve := faux ;
    P := biblio ; Prec := Nil;
    TQ (P <> Nil et trouve = faux) faire
        DTQ
           Si (P^.titre = t)  Alors trouve = vrai ;
           Sinon
              Prec := P;
              P := P^.adr;
           FSi
        FTQ

       Si (trouve = vrai) Alors
           P^.nbr := P^.nbr + nb;
       Sinon
           Allouer(Q);
           Q^.titre := t;         Q^.auteur := aut;
           Q^.prix := p;           Q^.nbr := nb;
           Q^.suiv := Nil;

           Si (biblio = Nil) Alors    biblio := Q ;
           Sinon                      P^.adr := Q;
       FSi
Fin;
```

4. Ecrire une procédure **nettoyage(biblio)** qui permet de supprimer tous les livres qui n'ont aucuns exemplaires.

La procédure : nettoyage (biblio)

```
Procedure nettoyage (VAR biblo : Pointeur)
    Var P, Prec : Pointeur (Livre);

Debut
    P := biblio ; Prec := Nil;
    TQ (P <> Nil) faire
       DTQ
          Si (P^.nbr = 0) Alors
               Q := P;
               Si (P = biblio) Alors biblio := P^.adr ;
               Sinon                Prec^.adr := P^.adr ;
               P := P^.adr;
               Liberer(Q);
          Sinon
              Prec := P;
              P := P^.adr ;
          FSi
       FTQ
Fin.
```

Exercice 9 : La société (Examen PSD 2017-2018)

Une société souhaite stocker les informations de son personnel (travailleurs) dans une liste linéaire chainée. Chaque personne est représentée par son nom et prénom, son type (dirigeant, administrateur ou employé), sa date de naissance et son année de recrutement.

1. Donner les structures de données nécessaires à la représentation de cette liste.

Construire la liste

```
Algorithme societe ;
   Type personne = Enregistement
               nom : chaine[30];
               prenom : chaine[30];
               type : chaine;
               date_naissance : chaine ;
               annee : entier ;
               adr : Pointeur (personne)
         Fin ;

   Var  L : Pointeur (personne);
```

2. Ecrire la fonction **nbr_travailleurs(L, type)** qui permet de calculer le nombre d'employeur du type « ***type*** » dans la liste ***L***;

La fonction nbr_travailleurs(L, type)

```
Fonction nbr_travailleurs (VAR L : Pointeur, type : chaine) : entier ;
    Var P : Pointeur (etudiant);
        nb: entier ;

Debut
    nb:= 0;
    P := L ;
    TQ (P <> Nil) faire
       DTQ
          Si (P^.type = type) Alors
             nb := nb + 1;
          Fsi
          P := P^.adr;
       FTQ
    nbr_travailleurs:= nb;
Fin.
```

3. Ecrire une procédure **supprime_ancien(L)** qui permet de supprimer tous employeurs qui ont dépassés 25 ans de travail cette année (2018).

La procédure : supprime_ancien (L)

```
Procedure supprime_ancien (VAR L : Pointeur)
    Var P, Prec : Pointeur (Livre);

Debut
    P := L ; Prec := Nil;
    TQ (P <> Nil) faire
       DTQ
          Si (2018 - P^.annee > 25) Alors
               Q := P;
               Si (P = L) Alors       L := P^.adr ;
               Sinon                  Prec^.adr := P^.adr ;
               P := P^.adr;
               Liberer(Q);
          Sinon
              Prec := P;
              P := P^.adr ;
          FSi
       FTQ
Fin.
```

4. Ecrire un algorithme qui permet d'éclater la liste L en trois : une liste –L1- contient que les dirigeants, une liste –L2- contient que les administrateurs et la dernière liste –L3- contient que les employés).

Algorithme : Eclater

```
Algorithme societe ;
   Type personne = Enregistement
              ...

   Var  L,L1,L2,L3,P1,P2,P3,P,Q : Pointeur (personne);

Debut
    P := L ;L1=Nil; L2=Nil; L3=Nil; P1=Nil; P2=Nil; P3=Nil;
    TQ (P <> Nil) faire
       Allouer(Q);
       Q^.nom := P^.nom ;         Q^.prenom := P^.prenom ;
       Q^.type := P^.type ;       Q^.date_naissance := P^.date_naissance ;
       Q^.annee := P^.annee ;     Q^.suiv := Nil;

       Si (type = 'dirigeant') Alors
           Si (L1 = Nil) Alors        L1 := Q ;
           Sinon                      P1^.adr := Q;
           P1 :=Q ;
       Sinon
           Si (type = 'administrateur') Alors
               Si (L2 = Nil) Alors    L2 := Q ;
               Sinon                  P2^.adr := Q;
               P2 :=Q ;
           Sinon
               Si (L3 = Nil) Alors    L3 := Q ;
               Sinon                  P3^.adr := Q;
               P3 :=Q ;
           Fsi
       FSi
   Ftq;

Fin.
```

Exercice 10 : Inverser une liste

Soit L1 une liste linéaire chainées. Ecrire une procédure qui inverse les éléments de la liste L1 en créant une nouvelle liste L.

La procédure : Inverser (L1, L)

```
Procédure Inverser (L1 : pointeur(maillon); VAR L: Pointeur(maillon));
Var   P, Q: pointeur(maillon) ;     // P : pointeur courant
                                    // Q : nouveau maillon créé
Début
    P ← L1;
    L ← Nil;
    TQ P <> Nil Faire
         Allouer(Q) ;
         Q^.Val ← P^.Val ;      //le nouveau élément Q prend la valeur de P
         Q^.Adr ← L ;           //le nouveau élément Q pointe la tete L
         L ← Q;                 //Pointer la tête L au nouvel élément
         P ← P^.Adr ;           //Avancer P à l'élément suivant
    FTQ

Fin;
```