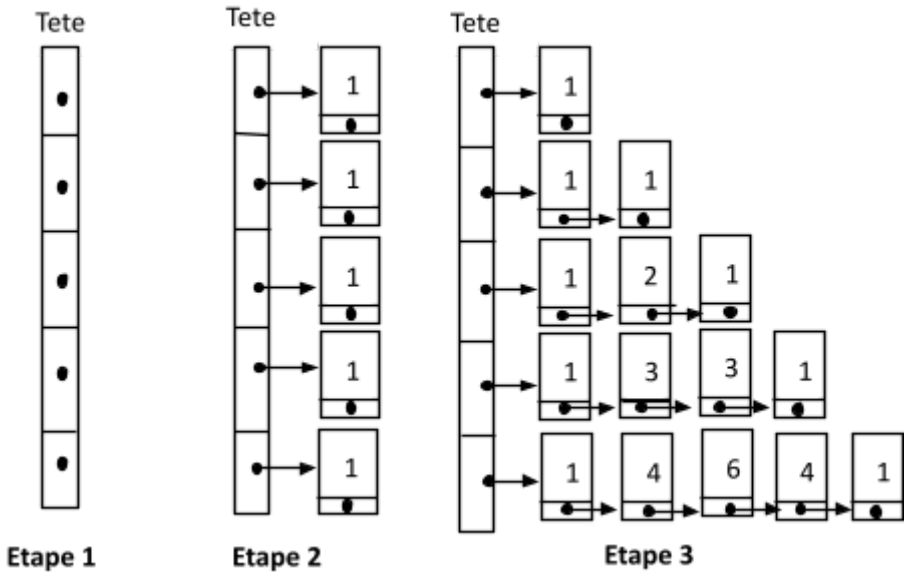
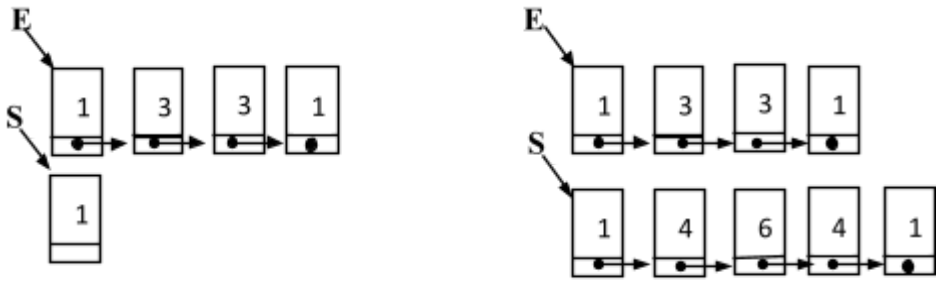
Exercice 1 :

On veut construire le triangle de Pascal pour un nombre de lignes donné en utilisant un tableau Tete et plusieurs listes chainées. Les étapes à suivre sont données comme suit :



1. Initialisez le tableau Tete[100] et donnez sont type (voir Etape 1).
2. Créez la première colonne du triangle avec la valeur 1. Mettez l'adresse de cet élément dans le tableau Tete (voir Etape 2). Le prototype de cette fonction est le suivant :
3. Écrire une procedure qui, à partir de la ligne *n* du triangle, calcule les valeurs de la ligne *n+1*(voir la figure ci-dessous. Le prototype de cette procedure est le suivant :

   Procedure Const_ligne_svt(?? E, ?? S)



· Avant l'appel de la fonction · Après l'appel de la fonction

4. Écrire une Action qui affiche les éléments d'une ligne du triangle.

   ?? Affiche_ligne (??)
5. On souhaite imprimer, parmi les 100 premières lignes du triangle, toutes celles qui ne contiennent que des valeurs strictement inférieures à un nombre strictement positif donné *nb* (nb ≥1). On vous demande d'écrire une Action **F( )** qui utilise les actions **Affiche_ligne( )** et **Const_ligne_svt ( )** pour calculer et afficher une à une les lignes du triangle jusqu'à la première ligne contenant une valeur supérieure ou égale au nombre *nb* donné (nb ≥1). Cette ligne ne sera pas affichée. Si on atteint la ligne 100 sans avoir trouvé, on s'arrête.
6. Ecrire une action **Liberer ( )** qui libère l'espace mémoire alloué au triangle dans le but de retrouver la configuration donnée par l'étape 2.
7. Écrire le programme C qui construit et affiche le triangle pour un nombre de lignes donné (nb_ligne ≤ 100) en utilisant les deux méthodes :

- Const_ligne_svt( ), Affiche_ligne( )
- Liberer ( ), F ( )

Solution

1/

```
Type  liste=^Cellule ;
Type Cellule=enregistrement
                    info : entier;
                    svt :liste ;
              fing ;

tete : tab[100] liste ;
i : entier ;
pour (i<- 1 à 100)
  faire  tete[i]<- nil ;
fait ;
```

2/

```
Procédure  init_un (E/ t : tab[100] Liste)
Début
   i : entier ;
   pour (i<- 1 à nb)  faire  t[i]<- ajout_tete(t[i], 1) ;
fait ;
Fin ;
```

3/

```
Procédure Const_Ligne_svt(E/  S, E : Liste)
Début
   Som : entier ;
   prd : Liste
Prd <- S
Tant que(E ≠  nil)
Faire
   Som <- E^.info ;
   E<- E^.svt ;
   Si (E ≠ nil) alors som <- som + E^.info ;
Fsi ;
   Ajout_apres (prd, som) ;
Fait ;
Fin ;
```

4/

```
Procédure  Affiche_ligne(E/ t : Liste)
 Début
      Tantque (t ≠ nil) faire
         Ecrire(t^.info) ;
          t   t^.svt;
       fait ;
Fin ;
```

5/

```
Procédure F (ES/  E, S : Liste, E/ nb : entier)
Début
    i : entier ;
    arret : booléen ;
    p, ptr, s1 : Liste

Affiche_ligne(E) ;
arret <- faux ;
i <- 2 ;

Tant que (i < 100 et arret=faux)
Faire
    Const_Ligne_svt(E,  S) ;
    Ptr <- S ;
    TQ (ptr t ≠ nil et ptr ^.info < nb) faire ptr <- ptr^.svt ; fait
    Si(ptr=nil) alors Affiche_ligne(S) ;
                Sinon arret <- vari ;
    Fsi ;
    i <- i+1 ;
    TQ (ptr^.svt ≠ nil)
     Faire
       p<- ptr^.svt ;
       ptr^.svt = p^.svt;
       Liberer(p) ;
     Fait ;
    S1 <- S ;
    S <- E ;
    E <- S1 ;
Fait ;
 Liberer(S) ; S <- nil ;
 TQ (E^.svt ≠ nil)
     Faire
       p<- E^.svt ;
       E^.svt = p^.svt;
       Liberer(p) ;
     Fait ;
  Liberer(E) ; E <- nil ;
Fin ;
```

6/

```
Procédure Libérer(E/ T tab : [100] Liste)
 Début
   i : entier ;
   prd : Liste ;

  Pour (i<- 1 à 100)
     Faire
        prd <-t[i];
        TQ(prd^.svt ≠ nil) faire
supp_apres(prd) ; fait
     Fait ;
  Fait ;
Fin ;
```

7/

```
Debut
  tete : tab[100] liste ;
  i, nb : entier ;
  Pour (i<- 1 à 100)  faire  tete[i]<- nil ;
fait ;
  Lire( nb) ;
  Init_un(tete) ;
  Pour (i<- 2 à 100)  faire  Cons_Ligne_svt
(tete[i-1], tete[i]);  fait ;
  pour (i<- 1 à 100)  faire  Affiche_ligne
(tete[i]);  fait ;
  Libérer (tete) ;
  F (tete[1], tete[2],nb) ;
Fin ;
```

Solution exercice 4 (liste circulaire) :

1/

```
Type  listeC=^Cellule ;
Type Cellule=enregistrement
                    info : chaine;
                    svt :listeC ;
             fing ;

Fonction créer_liste(E/ ch : chaine) : listeC
Début
  P, Q : listeC ;

   Q <- nil ;
   Pour (i<- 1 à longueur(ch))
    Faire
      Si (Q = nil) alors  Allouer (Q) ;
                          Q^.info <-ch[i] ;
                          Q^.svt <- Q ;
                     Sinon Allouer (P) ;
                           P^.info <-ch[i] ;
                           P^.svt <- Q^.svt ;
                           Q^.svt <- P;
                           Q <- P ;
      Fsi ;
   Fait ;
Fin ;
```

2/

```
Type  liste=^Cellule1 ;
Type Cellule1=enregistrement
                    info : chaine;
                    svt :liste ;
fing ;

Fonction créer_chaine(E/ q : listeC, ch : chaine) : liste
Début
  p, Q1 : listeC ;
  tete, nouv : liste ;
  ch1 : chaine ;
  tete <- nil ;
  Q1 <- Q ;

  Répéter
    p <-Q1 ;
    ch1<- "" ;// cette instruction vide le contenu de la chaine
    Répéter
      ch1<-ch1+p^.info ;
      p<- p^.svt ;
    Jusqu'à (p=Q1) ;
    Si (ch1 ≠ ch et Exist(tete, ch1)=Faux)
                        alors  Allouer(nouv) ;
```

```
                        nouv ^.info <- ch1 ;
                        nouv ^.svt <- tete ;
                        tete <- nouv ;
    Fsi ;
    Q1 <- Q1^.svt ;
 Jusqu'à (Q1=Q) ;
Fin ;

Fonction Exist (E/ p : liste, ch : chaine) : booléen
Début
  Tant que (p ≠ nil et p^.info ≠ ch) faire p <- p^.svt ; fait ;
   Si (p ≠ nil) alors retourne vrai ;
               sinon retourne faux ;
   Fsi ;
Fin ;
```

3/

```
Type  listeP=^Cellule2 ;
Type Cellule2=enregistrement
                info : chaine;
                svt1 : listeP ;
                svt2 : liste ;
fing ;

Début
   lc : listeC ;
   t : liste ;
   p, teteP, cteteP : listeP ;
   N, i : entier ;
   p<- nil ; teteP<- nil ;

// Phase de création
   Lire(N) ;
   Pour (i<- 1 à N)
     Faire
       Lire (ch) ;
       Allouer(p) ;
       p^.info <- ch ;
       p^.svt <- teteP ;
       teteP <- p ;
    Fait ;

  cteteP <- teteP ;
  Tant que (cteteP ≠ nil)
      faire
         ch <-  cteteP^.info;
         lc <- créer_liste(ch) ;
         cteteP^.svt2 <- créer_chaine(lc, ch) ;
         Liberer_ListC(lc) ;
```

```
        cteteP <- cteteP^.svt1 ;
    fait ;


  // Phase d'affichage
  cteteP <- teteP ;
  Tant que (cteteP ≠ nil)
     Faire
        Ecrire (cteteP^.info) ;
        t <- cteteP^.svt2 ;
        Tant que (t ≠ nil)
            Faire
              Ecrire (t^.info) ;
              t <- t^.svt ;
            Fait ;
        cteteP <- cteteP^.svt1 ;
     Fait ;
Fin ;

Procédure  Liberer_ListC(ES/ ^Q : listeC)
 Début
   p, prd : listeC ;

    prd <- Q^.svt ;
    Tant que (prd ≠ nil)
      faire
        p <- prd ;
         prd <- prd^.svt ;
         Q^.svt <-prd ;
         Liberer(p) ;
     Fait ;
  Liberer(Q) ;
  Q <- nil ;
Fin ;
```