Programmation et Structure de Données

Structure de données de Base
Les listes chainées Partie 2

ENS KATEB

Ajout d'un élément à une liste

Cas d'une liste simple :

C – En milieu d'une liste dans une position précise :

Les positions possibles sont : première pos=1, Deuxième pos=2, … ou dernière pos = nbr(L)+1 où nbr(L) retourne le nombre des éléments de la liste L

Voici les Sous programmes récursifs qui permettent de retourner le nombre des éléments dans une liste simple (nbr_s) et doublement chainée (nbr_d)

```
int nbr_s (elem_s *L)
{
 if(L==NULL)return 0;
 else return 1+nbr(L->suiv);
}
```

```
int nbr_d (elem_d *L)
{
 if(L==NULL)return 0;
 else return 1+nbr(L->suiv);
}
```

Exemple : si notre liste contient 3 éléments, donc les positions possibles sont : 1, 2, 3 ou 4 les autres sont des positions incorrectes

Ajout d'un élément à une liste

Cas d'une liste simple :
C – En Milieu de liste dans une position précise :

```
elem_s * ajoutM_s (elem_s *L, int pos)
{  if(pos >= 1 && pos <=nbr_s(L)+1)
            {  elem_s *E = new(elem_s);
               cout <<"\nDonner la valeur du nouvel element : "; cin >> E->D;
               if(pos==1)
                  {  E->suiv = L;
                     return E;
                  }
               else
                {   int i=1;        elem_s *T = L;
                    while (i<pos-1) {T = T->suiv; i++;}
                    E->suiv = T->suiv;
                    T->suiv = E;
                    return L;
                 }
             }
    else  {cout <<"\nPosition Incorrecte!!!";return L;}
}
```

Ajout d'un élément à une liste

Cas d'une liste doublement chainée :
C - En Milieu de liste dans une position précise:

```
elem_d * ajoutM_d (elem_d *L, int pos)
{
  if (pos >= 1 && pos <=nbr_d(L)+1)
  {
     elem_d *E = new(elem_d);
     cout <<"\nDonner la valeur du nouvel element : "; cin >> E->D;
     if(L==NULL)    {E->suiv = L;       E->prec = NULL;       L=E;}
     else
         if(pos==1)
            {  E->suiv = L;      E->prec = NULL;      L->prec = E;      L = E;  }
         else
           {
              int i=1; elem_d* T=L;
              while (i<pos-1) {T = T->suiv;  i++;}
              E->suiv = T->suiv ;     E->prec = T;
              if(T->suiv != NULL) T->suiv->prec = E;
              T->suiv = E;
            }
   }
  else  {cout <<"\nPosition Incorrecte!!!";}
  return L;
}
```