C++

Partie I – Overview

50 Final
25 Mid
20 Projects
5 Participation
Partiel – 16 Novembre
Final – 11/18 Janvier

1/ Structures

1. Une structure regroupe des variables liées de type différents

   struct nomDeStruct
   {
   type nomDeVariable;
   }

   Exemple : struct point
   {
   float x;
   float y;
   };

2. Access de la struct :
   struct nomDeStruct nomDeVariable;
   nomDeVariable.nomdeVariable prédéfinie dans la structe de base;

Exemple :

```
struct point p1;
p1.x = 1;
p2.y = 2;
```

3. Structure incluant un pointeur :
   struct nomDeStruct *nomDeVariable;
   nomDeVariable = nomDeStruct;

Exemple :

```
struct point *pp1;
pp1 = &p1;
```

Acces :

(*pp1).x = 3; // Pas très pratique
Alternative : pp1 →x=3;

4. Pour éviter la duplicatiion des struct : on utilise typedef int entier;
   entier x = 1;

```
typedef  struct point
{
int x;
int y;
} point;
point A;
```

Partie II – Listes chainées

1/Création d'une liste chainée et insertion d'un element

5. Utilisation des listes pour insérer et retirer des éléments facilement
6. Dans les structures des LC on a un pointeur qui pointe vers l'élément suivant
7. Une liste chainée contient un pointeur qui pointe à sa propre structure car une LC est une structure qui s'auto-reference
8. Déclaration de la structure de liste :

Structure de liste

```
typedef struct _un_element
  {
   int data;
   struct _un_element *suivant;
  } Un_element;
```

ou bien :

```
typedef struct _un_element *P_un_element;

typedef struct _un_element
  {
   int data;
   P_un_element suivant;
  } Un_element;
```

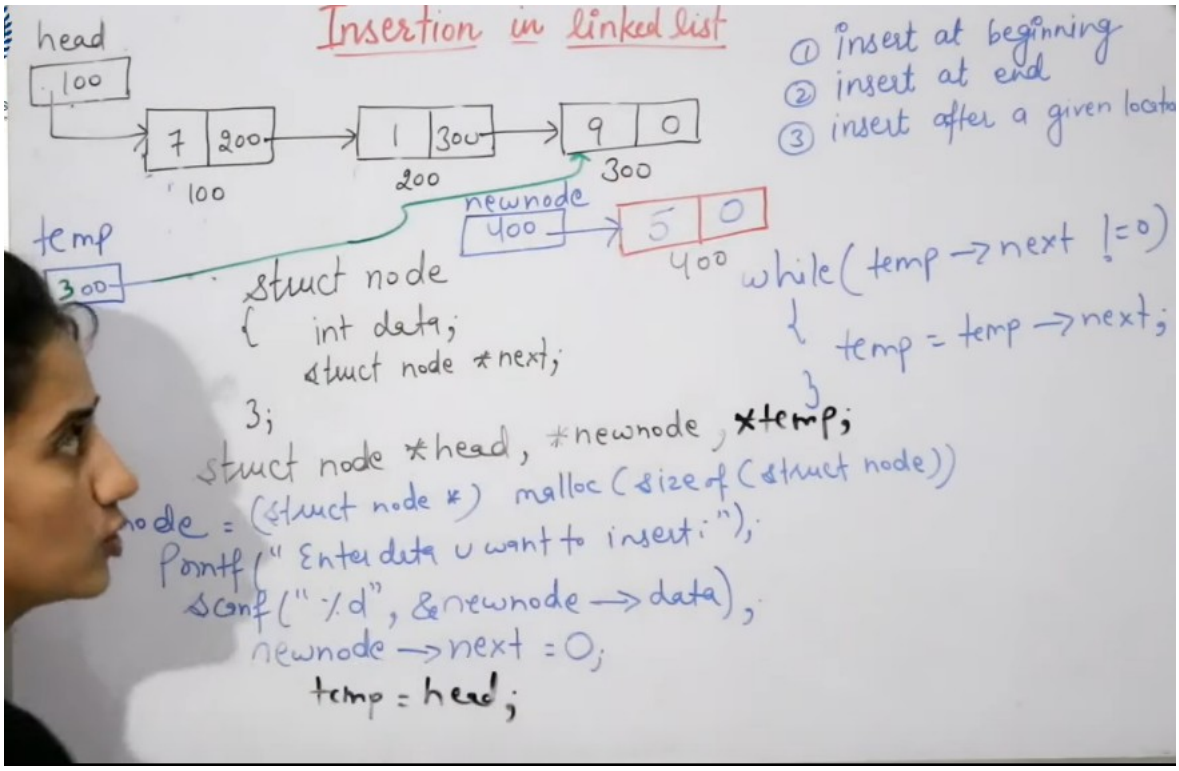
9. Ensuite on déclare les fonctions qui nous serviront – insertion en fin de list

```
noeud * creerListeChainee (int n) {
  int x = 0;
  Noeud * debut = NULL;  il n'y a qu'une adresse ici sans data
  Noeud * p = NULL;
  Noeuf  * temp = NULL; pointer qui parcourera ma chaine

  Création de noeuds individuels
  for (x = 0; x < n; x++){

  p = (noeud *)malloc(sizeof(node)); allocation de mémoire pour le noeud
  printf("Entrer la valeur du noeud %d", x +1);
  scanf("%d",&(p → data)); on utilise & meme si p est un pointer car on ne
  stocke pas qu'un pointer mais l'adresse de (p → valeur)
  p  → suivant = NULL;

  if  (debut == NULL) { Si la liste est vide, rendre temp comme premier noeud
    debut = p;
  } else {
    temp  = debut;
    while (temp → suivant != NULL) { on écrit pas p != NULL sinon on perd
  notre 'parcoureur' et on peut plus compareur
     temp= temp → suivant; On pointe p aux adresses jusqu'à ce qu'on arrive à
  NULL
  }
    temp → suivant = p; On insère le dernier noeud
```



10.

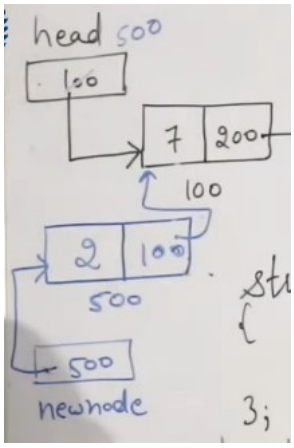
11. Afficher la liste chainée :

```
void afficherListe (noeud * debut) {
  Noeud * p = debut;
  while (p → suivant != NULL)
  { printf("\t%d→", p → data);
    p = p → next;
}}
```

12. Insertion en début de liste :

```
 noeud * debut; noeud * nouveaunoeud;
…
scanf("%d", &nouveaunoeud → data);
nouveaunoeud → suivant = debut;
début = nouveaunoeud;
```

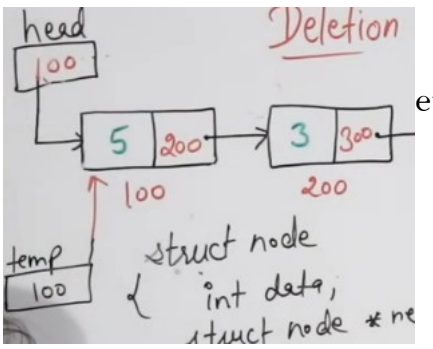
On change en premier l'adresse du nouveau noeud car si on changeait d'abord celle du début, on perdrait son adresse et le nouveau noeud ne pourra pas etre attaché au reste de la liste (il est rattaché à la meme valeur GRACE à l'adresse du début : si on éliminer l'adresse du début, on eliminerai l'adresse de rattachement aussi ...)



2/Suppression d'un element d'une liste

13. On declare une structure element

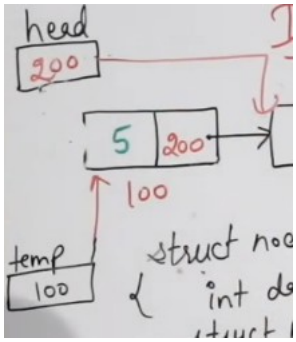
14. Pour supprimer un élément, on raccorde le debut à l'élément qui suit celui qu'on veut supprimer. Ensuite, on supprime la mémoire allouée de l'élément concernée en lui accordant un pointer et en le libérant ensuite.



15. Avant le raccordement et avant la suppresion

et

16. Après le raccordement de début à l'élément qui pointe après l'element à inserer



17. Après la typedef struct :

```
Noeud * temp;
DeleteDepuisDebut () {
if (debut == NULL){ system.out.println("La liste est vide");
debut = debut → next;
free(temp);
temp = NULL;
}
```

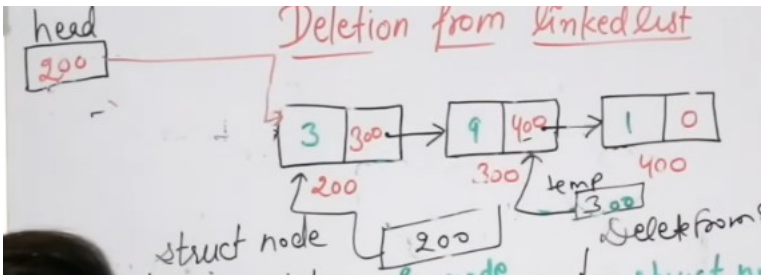
18. Supprimer depuis la fin :

```
Noeud * debut, * temp;
DeleteDepuisFin(){
Noeud * p;
if (debut == NULL){ system.out.println("La liste est vide");
temp = head;
while (temp → next != 0) {
  p = temp;
  temp = temp → next; }
if (temp == head) { head = 0; free(temp);
} else {
  p → next = 0;
  free(temp);
} free(temp);
```

En gros on n'a pas besoin d'écrire 2 fois free(temp), on peut juste le mettre à la fin du if-else

19.

20. L'adresse de p avant la dernière itération ou temp → next == 0



}

21. Arret de la boucle while et p pointe bien à l'element precedant celui qu'on veut supprimer. On supprimera ce dernier en liberant le pointeur temp