Centre Universitaire de Naâma
Institut des Sciences et Technologies
Département Mathématiques et Informatique

Algorithmique et structure de données

- Introduction
- Les algorithmes séquentiels simples
- Les structures conditionnelles
- Les boucles
- Les tableaux et les chaînes de caractères
- Les sous-programmes – Procédures et Fonctions
- Les types personnalisés
- Les fichiers
- Les listes chaînées
  - Introduction

Djelloul BOUCHIHA
bouchiha.dj@gmail.com
2020-2021

Tableau vs Liste chaînée

- **Un tableau :**
  - ☞ un ensemble d'éléments de même type.
  - ☞ nombre d'éléments fixé lors de la déclaration.
  - ☞ ne peut pas être étendu pendant l'exécution.

- **Une liste chaînée :**
  - ☞ un ensemble d'éléments de même type.
  - ☞ étendue pendant l'exécution par l'allocation (réservation) d'un nouvel espace mémoire.
  - ☞ réduite par une désallocation de l'espace non utile.
  - ☞ ses éléments sont liés par des pointeurs.

Les pointeurs

- **Un pointeur :**
  - ☞ une variable qui contient une adresse mémoire, et non pas la valeur.
  - ☞ une variable qui au lieu de contenir la donnée, contient son adresse en mémoire.

- **Format général :**

  <type de données> *Nom_pointeur ;

- **Exemple :**

  Variables
  Entier *Y ;

- **En C :**

  int *Y ;

Les pointeurs

- **Illustration :**
  - ☞ Soit le programme :

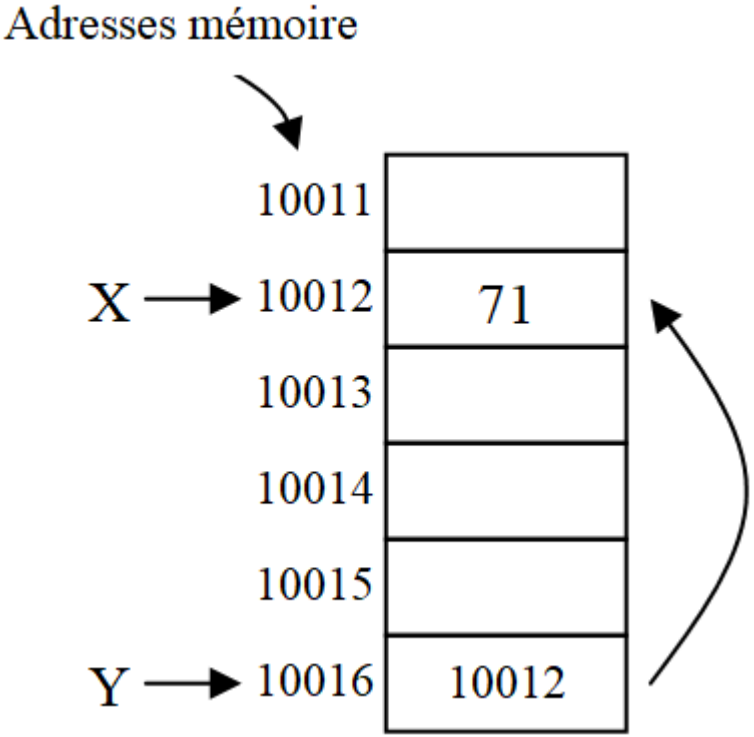
```
#include<stdio.h>
main()
{ int X;
  int *Y;  // Y est un pointeur vers un entier
  X = 7;
  Y = &X ;  // Y reçoit l'adresse de la variable X
  printf("*Y = %d\n", *Y) ; // afficher la valeur pointée par Y
}
```

  - ☞ Le programme affiche :

***Y = 7**

Les pointeurs

- **Illustration :**
  - ☞ X, Y peuvent êtres représentées en mémoire :



Les pointeurs

- **Propriétés :**
  - ☞ Un pointeur peut recevoir la valeur d'un autre pointeur par une opération d'affectation.
  - ☞ On peut aussi comparer deux pointeurs pointant vers des objets de même type par les opérateurs de comparaison (=, >, <, >=, <=, !=).
  - ☞ un pointeur peut être incrémenté ou décrémenté.
  - ☞ On peut faire la différence de deux pointeurs ; on aura un entier.
  - ☞ Les autres opérations arithmétiques (somme, produit…) ne sont pas acceptées.
  - ☞ L'espace mémoire réservé à une variable de type pointeur peut varier en fonction du matériel et du système d'exploitation. Ça peut être 2 octets, 4 ou même 8 octets.
  - ☞ Les pointeurs mènent à une gestion dynamique de la mémoire.

Gestion dynamique

- **Gestion dynamique vs Gestion statique :**
  - ☞ La gestion statique permet la réservation de l'espace mémoire lors de la déclaration. Exemple : les tableaux.
  - ☞ La gestion dynamique permet la réservation (allocation) de l'espace pendant l'exécution du programme grâce à des variables de type pointeur.
- **Une variable de type pointeur :**
  - ☞ contient l'adresse d'une variable dite "variable pointée".

Gestion dynamique

- **Procédures de la gestion dynamique :**
  - ☞ Allouer(p) : réserver un espace mémoire pour une variable pointée par le pointeur p. En C, on écrit : p=(<type_données>*)malloc(sizeof(<type_données>));.
  - ☞ Désallouer(p) : libérer l'espace mémoire réservé à une variable pointée par le pointeur p. En C, c'est la fonction free(p);.

- **Remarques :**
  - ☞ malloc et free, appartiennent à la bibliothèque stdlib.h;.
  - ☞ La valeur NULL de bibliothèque stdlib.h indique qu'un pointeur ne pointe vers aucun élément.
  - ☞ malloc et free utilisent une zone mémoire appelée TAS (en anglais Heap).

Gestion dynamique

- **Exemple :**
  - ☞ Soit l'algorithme :

```
Algorithme pointeurs
Variables
  Entier *Y;
Début
  Allouer(Y) ;
  *Y ← 7 ;
  Ecrire("La valeur est ", *Y) ;
  Désallouer(Y) ;
Fin
```

Gestion dynamique

- **Exemple :**
  - ☞ En C :

```
#include<stdio.h>
#include<stdlib.h>
main()
{ int *Y ;
  Y = (int*)malloc(sizeof(int));
  *Y = 7 ;
  printf("La valeur est %d\n", *Y) ;
  free(Y) ;
}
```

  - ☞ Le programme affiche :

La valeur est 7

Merci