Université Batna 2
Faculté des Mathématique et Informatique
Département d'Informatique
Master ISIDS - Semestre 1

Programmation Avancée

Préparé par Dr O.MESSAOUDI

2022/2023

Chapitre I
Rappels

- Généralité sur l'algorithmique
- Algorithmique et Programmation
- Langage algorithmique
- Tableaux
- Pointeurs

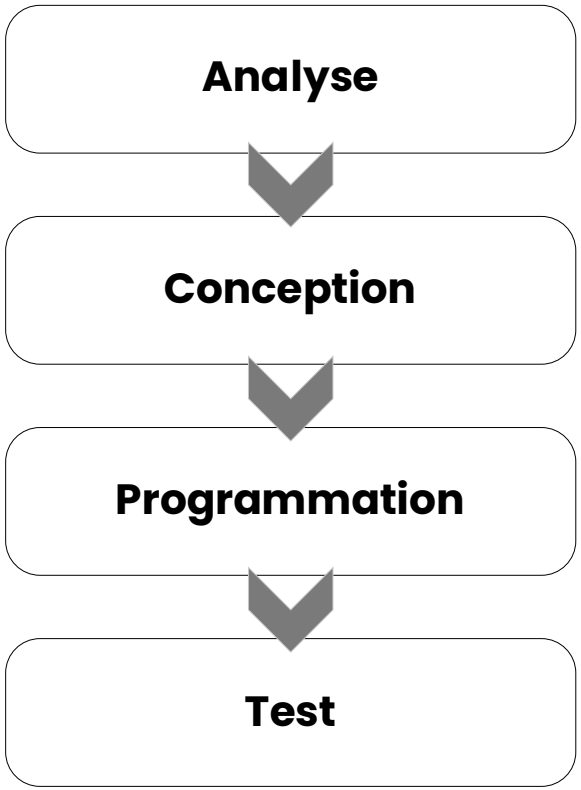
Généralité sur l'algorithmique

- **Historique:** L'algorithmique est un terme d'origine arabe, vient de Al Khawarizmi, un mathématicien persan du 9ième siècle.

- **Définition:** Un algorithme est une suite finie d'opérations (étapes) élémentaires constituant un schéma de calcul ou de résolution d'un problème.

Généralité sur l'algorithmique

Étapes de conception d'un algorithme:



Analyse des besoins définis dans un cahier de charges et définition d'une spécification claire de ce que doit faire le programme

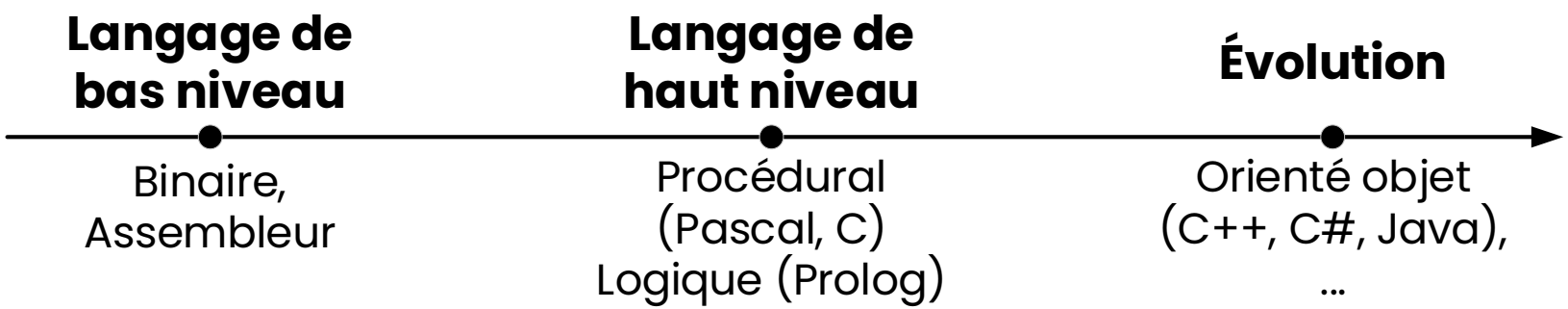
Conception de l'architecture du programme, ainsi la conception détaillée de chaque composant (sa contribution)

Réalisation et implémentation des composants définits pendant la conception, assemblage des différents composants

Test unitaire, test d'intégration, test système, et test d'acceptation

Algorithmique et Programmation

- **Définition:** Un programme est la description (traduction) d'un algorithme dans un langage de programmation.



Algorithmique et Programmation

Qualité d'un bon algorithme/programme

- **Correcte:** Il faut que le programme exécute correctement ses tâches.
- **Complet:** Il faut que le programme considère tous les cas possibles.
- **Efficace:** Il faut que le programme exécute sa tâche avec efficacité qui se mesure sur:
  - Temps (complexité temporelle),
  - Ressources (complexité spatiale),

Langage algorithmique

Structure d'un algorithme

```
Algorithme NomDeLAlgorithme ;
Const   c1= ...  c2=... ;
Var     var1 : Type ;  var2 : Type2 ;
        Procedure P1(....)
           Debut
            .........
           fin
        fonction F1(....) : Type ;
           Debut
           .........
           fin
debut
  ...
  P1(...) ;
  Var1 ← F1(...) ;
  ...
fin.
```

Langage algorithmique

Structures de données

- **Simples:** entier, réel, booléen, caractère, tableau, etc.
- **Complexes:** incluent:
  - Structures séquentielles: Listes, Piles, Files
  - Structures hiérarchiques: Arbres

Tableaux

- C’est un objet décomposé en plusieurs **éléments** de même type,
- Chaque élément est repéré par un **indice** (index),
- Le nombre d’éléments constitue sa **taille**,
- Le nombre d’indices qui permet de désigner un élément est appelée **dimension** du tableau,
- Le type de l’indice est un **intervalle** [0..taille -1]

Tableaux

Déclaration: se fait en précisant le mot **TABLEAU**, suivi par sa **taille** et par le **type** de ses éléments.

Tableau nom_tableau[**taille**]:**type**

- L'accès à un élément s'effectue en précisant le nom du tableau suivi par l'indice entre crochets: **Tab[1]**

Tableaux

Algorithmique	Langage C
Algorithm exemple; **Var** Tableau tab[10]:entier; **Début** tab[0] ← 0; affichier(tab[0]); affichier(*tab); affichier(tab); **Fin**	**int main()** **{** int tab[4] = {10, 23,505,8}; tab[0] = 10; printf("%d", tab[0]); printf("%d", *tab); printf("%p", tab); **}**

Tableaux

Tableaux multidimensionnels

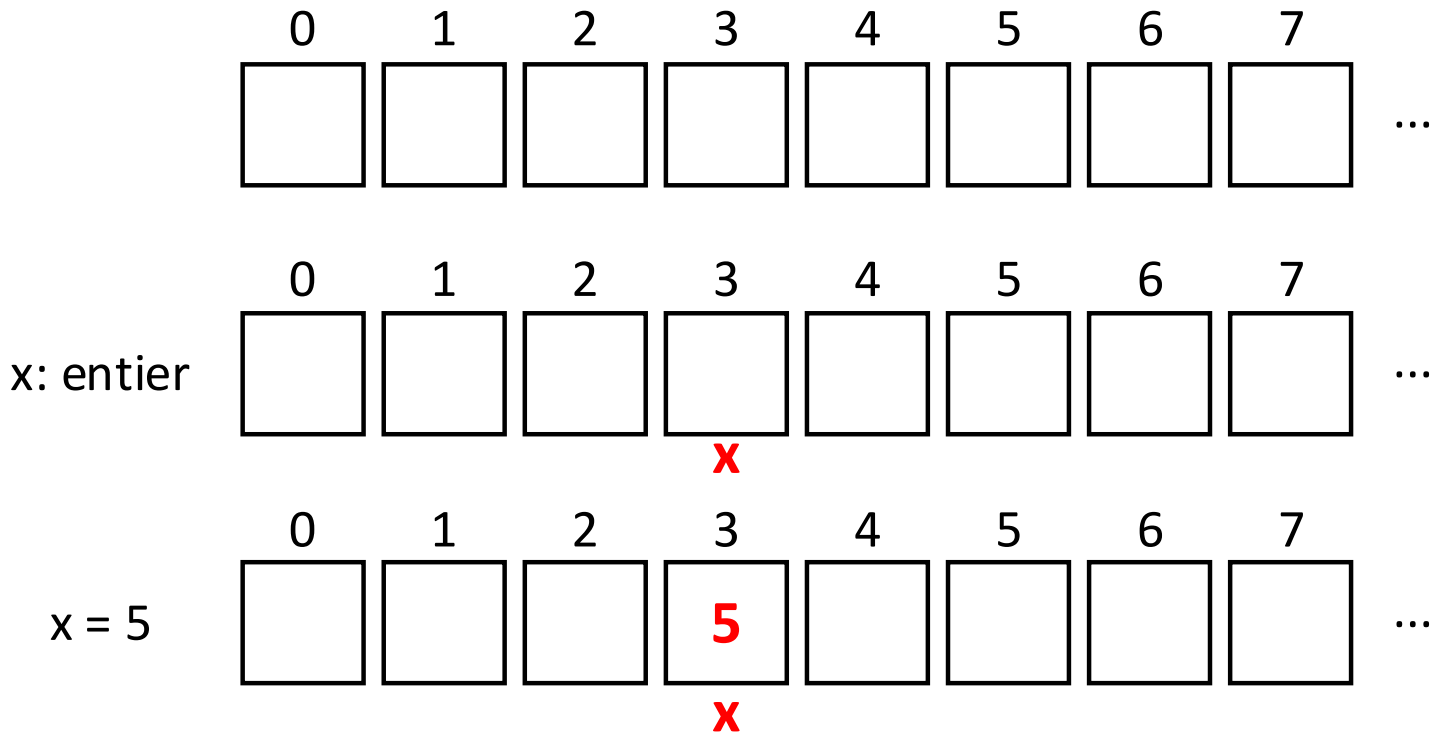
Un tableau multidimensionnel est considéré comme un tableau dont les éléments sont eux mêmes des tableaux.

Tableau nom [**taille_dim_1, taille_dim_2, …**]: **type**

- L'accès à un élément s'effectue en précisant le nom suivi par l'indice des dimensions: **Tab[1,2]**

Les pointeurs

Variable: est destinée à contenir une valeur du type avec laquelle elle est déclarée. Physiquement cette valeur se situe en mémoire.



Les pointeurs

Un pointeur est une variable destinée à contenir une ***adresse mémoire***, c-à-d une valeur identifiant un emplacement en mémoire.

Tout pointeur est associé à un type d'objet.

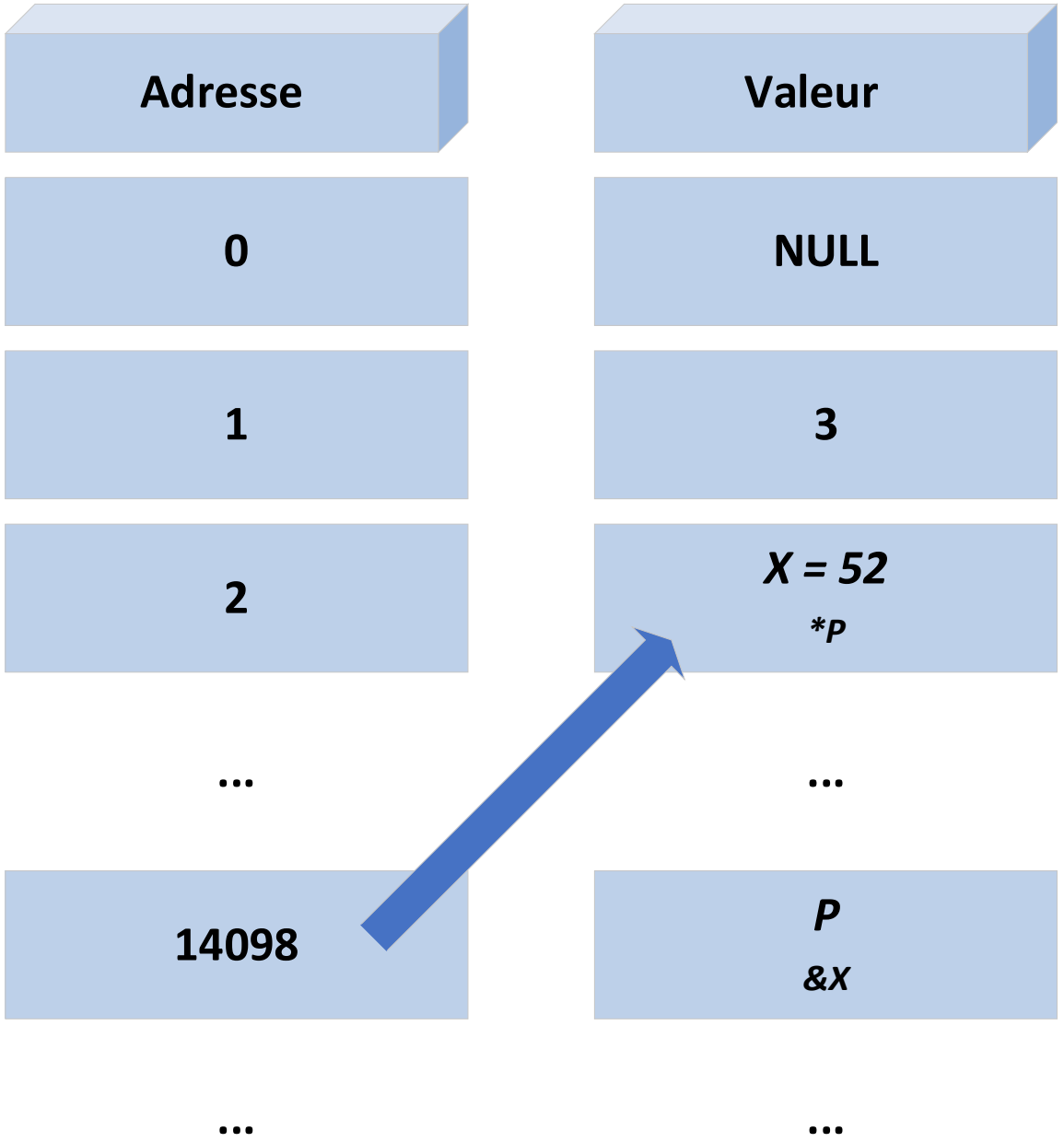
Opération sur les pointeurs:

- ***Affectation*** d'une adresse au pointeur
- ***Utilisation*** du pointeur pour accéder à l'objet dont il contient l'adresse

Les pointeurs

Exemples

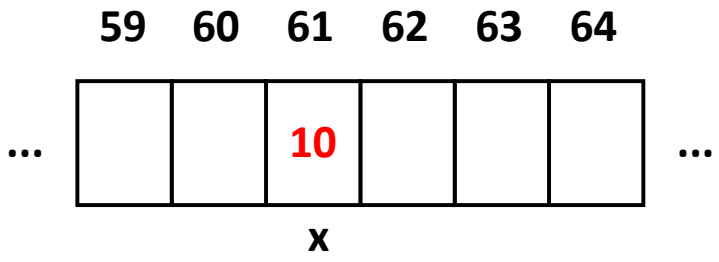
```
Var X:entier;
      P:*entier;
...
x=52;
P=&X;
...
```



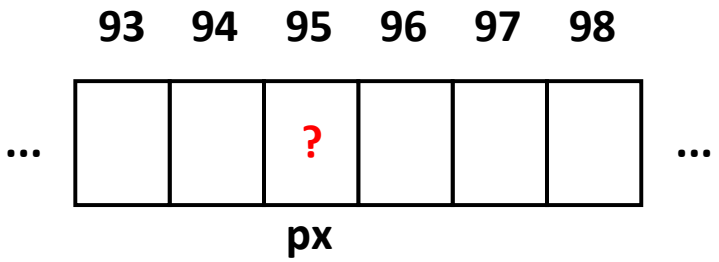
Les pointeurs

Exemples

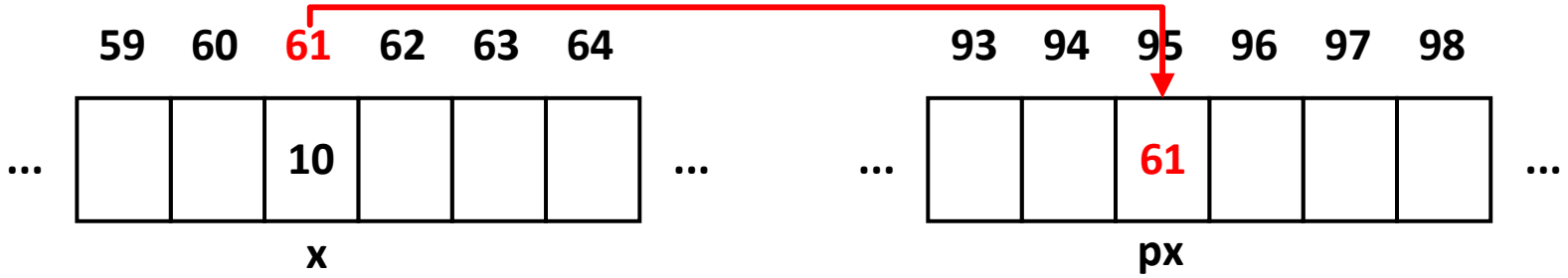
x:enter; x = 10;



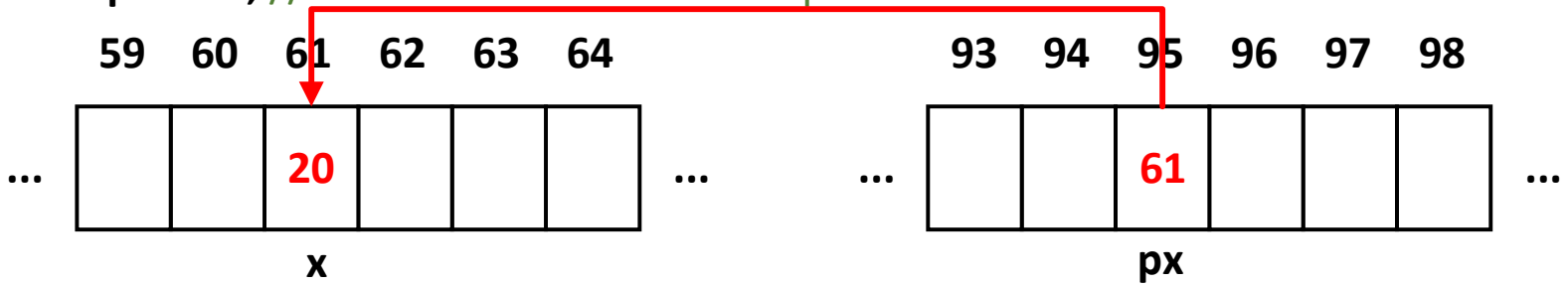
px:*enter;



px = &x; // affectation de @ de x au pointeur



***px = 20;** // affectation en utilisant le pointeur



Les pointeurs

Allocation dynamique du mémoire

- **Allocation statique:** la déclaration des variables réserve de l'espace mémoire pour ces variable.
- *Limitation:* connaitre au début l'espace nécessaire au stockage des variables.
- **Allocation dynamique:** l'espace nécessaire (ex. les tableaux) peut varier d'une exécution à une autre.

Les pointeurs

Algorithmique

- La **déclaration**:

$$ptr{:}*\,type;$$
$$ptr \leftarrow Nil;$$

- **Réserver** un espace mémoire/retourner un pointeur vers type:

$$ptr \leftarrow (*\,type)\ allouer()$$

- **Libérer**:

$$Liberer(ptr)$$

Les pointeurs

Langage C

- La **déclaration**:

$$type \ * ptr = NULL;$$

- **Réserver** un espace mémoire/retourner un pointeur vers type:

$$ptr = (* type)\boldsymbol{malloc}\big(\boldsymbol{sizeof}(type)\big);$$

- **Libérer**:

$$\boldsymbol{free}(ptr)$$

Les pointeurs

Les pointeurs et les tableaux

- Par défaut, le tableau est de **grandeur statique**, c-à-d qu'il impossible de les changer de taille après la compilation.
- Cependant, il est possible de changer la taille des **tableaux dynamiques** après la compilation.
- Pour faire des tableaux dynamique, il faut **réserver** un espace mémoire d'une taille donnée, puis d'assigner un **pointeur** à cet espace mémoire.

Les pointeurs

Les pointeurs et les tableaux (Algorithmique)

- La **déclaration**:

$$TAB{:}*\,type;$$
$$TAB \leftarrow Nil;$$

- **Réserver** un espace mémoire/retourner un pointeur vers type:

$$TAB \leftarrow (*\,type)\ \boldsymbol{allouerTab}(N)$$

- **Libérer**:

$$\boldsymbol{Liberer}(TAB)$$

Les pointeurs

Les pointeurs et les tableaux (Langage C)

- La **déclaration**:

$$type\ *TAB = NULL;$$

- **Réserver** un espace mémoire/retourner un pointeur vers type:

$$TAB = (*type)\boldsymbol{malloc}\big(\boldsymbol{N} * \boldsymbol{sizeof}(type)\big);$$

$$TAB = (*type)\boldsymbol{calloc}\big(\boldsymbol{N}, \boldsymbol{sizeof}(type)\big);$$

- **Libérer**:

$$\boldsymbol{free}(ptr)$$

Les pointeurs

Les pointeurs et le passage par variable

- Une autre utilité des pointeurs dans le langage C est de permettre le passage par variable des paramètres dans les procédures.

Algorithmique	Langage C
Procedure permuter**(**var x,y:**entier)** **Var** temp:entier; **Début** temp←x; x←y; y←temp; **Fin**	**void** permuter(**int** *px,**int** *py) { int temp; temp = *px; *px=py; *py=*temp; }
x←5;y←80;permuter(x,y);	x=5;y=80;permuter(&x,&y);

Les pointeurs

Les pointeurs et les autoréférences

- **Un autoréférence** est une structure dont un de ces membres est un pointeur vers une autre structure du même modèle.
- Cette représentation permet de construire des listes chainées et des arbres.

Algorithmique	Langage C
module:structure { moy:reel; **suiv:*module;** **}**	**struct module** { float moy; **struct module** *suiv; }