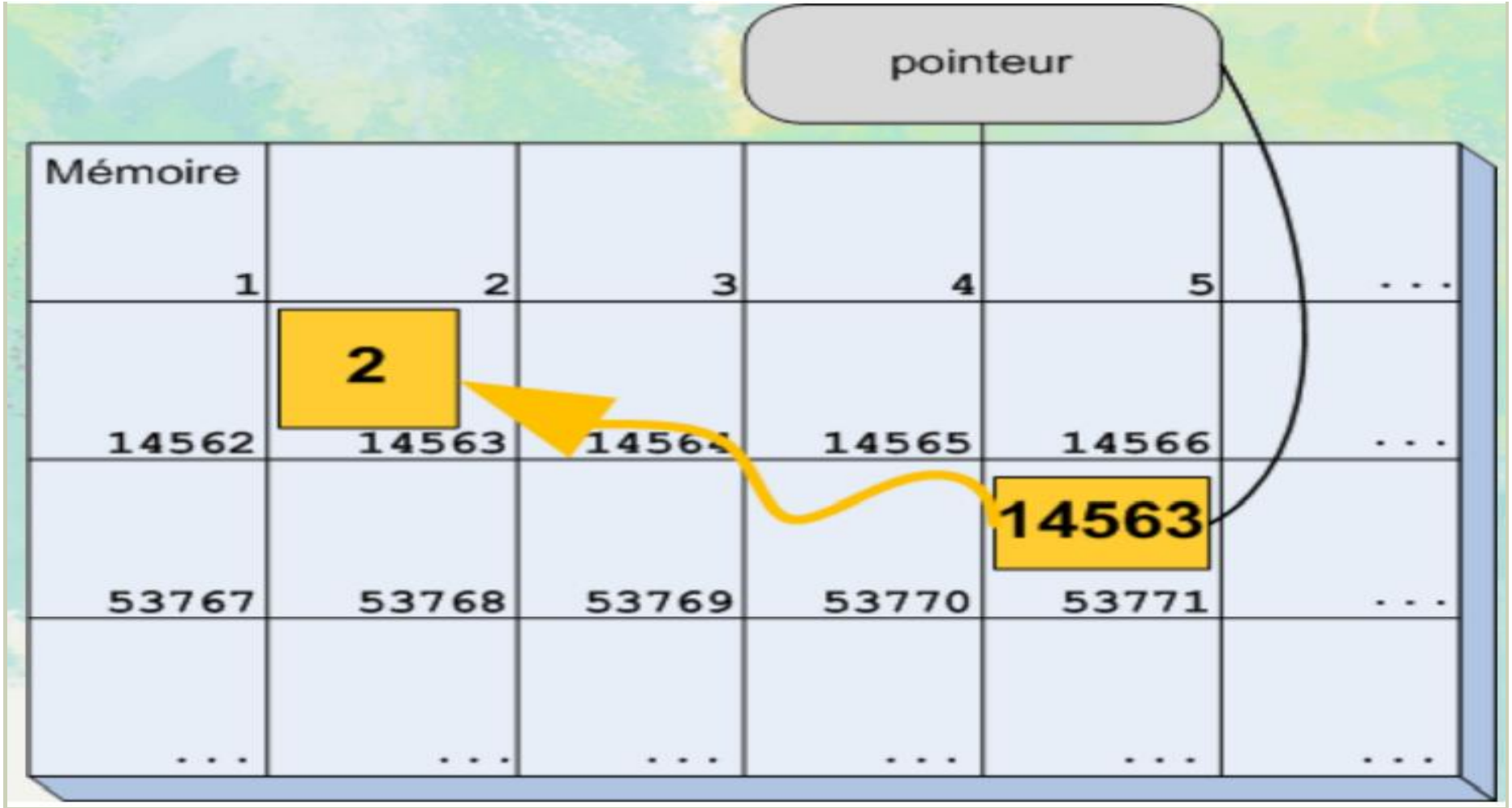
CHAPITRE 1
POINTEURS ET ALLOCATION DYNAMIQUE

QU'EST CE QU'UN POINTEUR?

Un pointeur en programmation est une variable qui contient l'adresse mémoire d'une autre variable.

ILLUSTRATION D'UN POINTEUR



QUELS SONT LES TYPES DE POINTEURS?

- A chaque type de variable correspond un type de pointeur (Même si en réalité un pointeur contiendra toujours une adresse sous forme d'entier).

- On peut même envisager des pointeurs sur des fonctions…

POURQUOI UTILISER LES POINTEURS?

- Il est souvent plus efficace en terme de temps d'exécution et de consommation mémoire de passer les données entre les différentes partie (fonctions) d'un programme en utilisant les pointeurs.

- Les pointeurs permettent de faire le passage par variable (ou par adresse) des paramètres d'une fonction.

POURQUOI UTILISER LES POINTEURS?

- Une fonction peut retourner plusieurs valeurs en utilisant les pointeurs.

- Les pointeurs permettent l'implémentation de types de données avancés comme les listes chaînées et les piles.

OPÉRATIONS SUR LES POINTEURS
DÉCLARATION D'UN POINTEUR

Pour déclarer un pointeur, il suffit d'écrire de précéder son nom par *. Par exemple:

- *nom_pointeur:type
- *ptr1, *ptr2 : type

OPÉRATIONS SUR LES POINTEURS
RÉFÉRENCEMENT

Le référencement ou affectation d'une adresse à un pointeur peut se faire de plusieurs façons:

- nom_pointeur <- &variable (l'opérateur & se lit adresse de)
- nom_pointeur1 <- nom_pointeur2
- nom_pointeur <- NULL
- Allocation dynamique

OPÉRATIONS SUR LES POINTEURS
DÉRÉFÉRENCEMENT

- Le déréférencement permet l'accès, à partir d'un pointeur, à la valeur de la variable sur laquelle il pointe.

- Le déréférencement se fait principalement à travers l'utilisation de l'opérateur *. Par exemple:

  val <- *nom_pointeur

OPÉRATIONS SUR LES POINTEURS
INCRÉMENTATION DE POINTEURS

Attention:

Incrémenter un pointeur par une valeur « i » revient à incrémenter sa valeur (qui est l'adresse d'une autre variable) par i fois la taille du type de la variable sur laquelle il pointe.

OPÉRATIONS SUR LES POINTEURS
INITIALISATION DES POINTEURS

- Lorsqu'un pointeur est déclaré pour la première fois, il contiendra une valeur entière et pointera très probablement vers une zone mémoire qui n'appartient pas à votre programme.

- Il est donc impératif d'initialiser un pointeur avant de l'utiliser.

- A défaut d'initialisation en temps de compilation, il vaut mieux initialiser le pointeur à NULL (nulle part) et lui affecter une adresse mémoire dynamiquement en temps d'exécution.

ALLOCATION DYNAMIQUE

Pour allouer une variable dynamiquement en temps d'exécution il faut:

- Déclarer un pointeur qui recevra l'adresse de cette variable:

  ***ptr : type**

- Réserver dynamiquement une zone mémoire et affecter son adresse au pointeur en question:

  ptr <- allouer(taille en octet)

  RQ. la fonction taille(TYPE) retourne le nombre d'octets nécessaires à l'allocation mémoire d'une variable de type TYPE.

- Contrairement à l'allocation statique il faut penser à libérer l'espace mémoire alloué une fois qu'on en a plus besoin:

  libérer(ptr)

POINTEURS ET TABLEAUX

Les tableaux peuvent être manipulés en utilisant les pointeurs. Soit **tab** un tableau de **type**, on aura dans ce cas:

- **tab** est un pointeur sur son premier élément.
- **taille(tab)** retourne la taille de tout le tableau.
- **tab <- allouer(taille)** est valide pour toutes les tailles multiple de la taille de type.
- **tab <- réallouer(taille)** permet de modifier dynamiquement la taille du tableau en plus ou en moins.