POINTEURS

POINTEURS
DÉFINITION

Un pointeur est une variable spéciale qui peut contenir l'adresse d'une autre variable.

- Si un pointeur P contient l'adresse d'une variable N, on dit que *'P pointe sur N*'.
- Les pointeurs et les noms de variables ont presque le même rôle (à exception prés):
  - Ils donnent accès à un espace mémoire.
  - Un pointeur peut 'pointer' sur différentes adresses tant disque le nom d'une variable reste toujours lié à la même adresse.

POINTEURS

DÉCLARATION, AFFECTATION & MANIPULATION

- **Déclaration**
  - Syntaxe: <pointeur>:^<type>
  - **Exemple**: Variable monPointeur: ^entier
- **Affectation**
  - Syntaxe: <pointeur> ←<adresse(variable)>
  - Ici on considère que adresse est une fonction. Elle nous renvoie l'adresse mémoire d'une variable
- **Manipulation**
  - pointeur^←valeur
  - Lire(pointeur^)
  - Ecrire(pointeur^)

POINTEURS EN C/C++

DÉCLARATION, AFFECTATION & MANIPULATION

- **Déclaration**
  - Syntaxe: <type> *<nom>
  - **Exemple**: int *p;
- **Affectation**
  - Syntaxe: <pointeur> = <&(variable)>
  - **Exemple**:
    - Int n=10;
    - Int *p(0);
    - p=&n;
- **Manipulation**
  - cout << "entrer une valeur";
  - cin >> *p; // écrire dans la case mémoire pointée par p
  - cout << "La valeur est : " << *p<< endl;
  - cout << "L'adresse est : " << p<< endl;

POINTEURS EN C

ALLOCATION DYNAMIQUE, LIBÉRATION DE LA MÉMOIRE

- Pour demander manuellement une case mémoire, on utilise l'opérateur malloc qui signifie « Memory ALLOCation ».
- malloc est une fonction ne retournant aucune valeur (void)(on n'en reviendra plus tard) :
  - void* malloc(size_t nombreOctetsNecessaires);
    - **Exemple**:
      - int* p= NULL;
      - p = malloc(sizeof(int));
- On peut libérer la ressource après usage via l'opérateur *free*
- *Free également est une fonction ne revoyant aucune valeur.*
  - *void free(void* p);*
    - **Exemple**: *free*(p);

POINTEURS EN C/C++

ALLOCATION DYNAMIQUE D'UN TABLEAU

- L'allocation dynamique d'un tableau est un mécanisme très utile. Elle permet de demander à créer un tableau ayant exactement la taille nécessaire (pas plus, ni moins).
- Si on veut créer un tableau de n élément de type int (par exemple), on fera appel à malloc.
  - **Exemple**:
    - int *t = NULL;
    - t = (int *)malloc(n*sizeof(int));
- Autres fonctions
  - calloc: identique à malloc mais avec initialisation des cases réservées à 0.
    - void* calloc(size_t taille, size_t nombreOctetsNecessaires);
  - realloc : permet d'agrandir une zone mémoire déjà réservée
    - void* realloc(void* tableau, size_t nombreOctetsNecessaires);
    - **Exemple**:
      - t = (int *) calloc (taille, sizeof(int));
      - taille = taille+10;
      - t =(int *) realloc(t, taille*sizeof(int));

POINTEURS EN C++

ALLOCATION DYNAMIQUE, LIBÉRATION DE LA MÉMOIRE

- Pour demander manuellement une case mémoire, on utilise l'opérateur new.
- Syntaxe: <pointeur> = new type
  - **Exemple**:
    - int *p(0);
    - p = new int;
- On peut accéder à la case et modifier sa valeur
  - **Exemple**: *p = 10;
- On peut libérer la ressource après usage via l'opérateur *delete*
  - **Exemple**: *delete* p;

POINTEURS EN C++

ALLOCATION DYNAMIQUE D'UN TABLEAU

- Un tableau dynamique est un tableau dont le nombre de cases peut varier au cours de l'exécution du programme. Il permet d'ajuster la taille du tableau au besoin du programmeur.
- L'utilisation de delete[] permet de détruire un tableau précédemment alloué grâce à new [].
- **Tableau unidimensionnel**
  - **Exemple**:
    - int i, taille;
    - …
    - cout << « Entrez la taille du tableau: ";
    - cin >> taille;
    - int *t;
    - t = new int[taille];
    - …
    - delete[] t;

POINTEURS EN C++

ALLOCATION DYNAMIQUE D'UN TABLEAU

- **Tableau à deux dimensions**
  - **Exemple**:
    - `int **t;`
    - `int nColonnes;`
    - `int nLignes;`
    - `…`
    - `t = new int* [nLignes];`
    - `for (int i=0; i < nLignes; i++)`

      `t[i] = new int[nColonnes];`
    - `…`
    - `delete[] t;`

EXERCICES D'APPLICATIONS

Application 12 :

Ecrivez un programme déclarant une variable i de type int et une variable p de type pointeur sur int. Affichez les dix premiers nombres entiers en :

- n'incrémentant que *p
- n'affichant que i

Application 13 :

Écrire un programme qui lit un entier n au clavier, alloue un tableau de n entiers initialisés à 0, remplir le tableau par des valeurs saisies au claviers et affiche le tableau.

Application 14 :

Ecrire un programme qui place dans un tableau T les N premiers nombres impairs, puis qui affiche le tableau. Vous accèderez à l'élément d'indice i de t avec l'expression *(t + i).

Application 15 :

Ecrivez un programme qui demande à l'utilisateur de saisir un nombre n et qui crée une matrice T de dimensions n*n avec un tableau de n tableaux de chacun n éléments. Nous noterons tij=0 j-ème élément du i-ème tableau. Vous initialiserez T de la sorte : pour tous i, j, tij=1 si i=j (les éléments de la diagonale) et tij=0 si i≠j (les autres éléments). Puis vous afficherez T.