Leçon 5 : Les Pointeurs en C++

UVCI

2018

Table des matières

1- Généralité sur les pointeurs

I

1.1- Définition

Comme en langage C, le langage C++ permet d'utiliser des pointeurs pour manipuler des données, mais il introduit aussi le concept de référence, très pratique pour permettre la modification d'une donnée passée en paramètre d'une fonction.

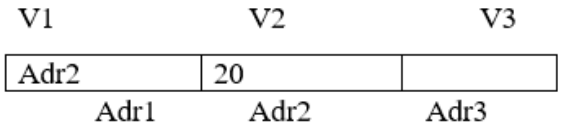
Un *pointeur* est une variable qui contient l'adresse d'une autre variable. L'adresse contenue dans un pointeur est celle d'une variable qu'on appelle *variable pointée*.

On dit que le pointeur pointe sur la variable dont il contient l'adresse. Un pointeur est associé à un type de variable sur lequel il peut pointer.

Remarque

- Un pointeur sur entier ne peut pointer que sur des variables entières, pareille pour les autres types de données.
- Un pointeur est lui-même une variable qui possède une adresse.
- Il ne faut pas confondre l'adresse de la variable pointeur et l'adresse contenue dans le pointeur.

Exemple :



Explication :

• V2 est placé à l'adresse Adr2

• V1 contient l'adresse de la variable V2 alors on dit V1 pointe sur V2

Application 1 :

II

Exercice

Comment appelle t-on chaque numéro qui identifie un espace en mémoire ?

- Pointeur
- Variable pointée
- Adresse

Exercice

Un pointeur est :

- Un espace qui conserve une valeur
- Un espace mémoire qui conserve une adresse
- Un espace mémoire qui conserve une valeur

Exercice

Un pointeur pointe sur :

- une valeur en mémoire
- une variable entière
- la variable dont il contient l'adresse

2- Traitement sur les pointeurs en C++

III

Syntaxe : 2.1- Déclaration d'un pointeur

*Type * variablepointeur ;*

Exemple :

int *Nbre; //Un pointeur qui peut contenir l'adresse d'un nombre entier

double *Nbre; //Un pointeur qui peut contenir l'adresse d'un nombre à virgule

string *val;//Un pointeur qui peut contenir l'adresse d'une chaîne de caractères

Remarque :

- *Il ne faut donc jamais déclarer un pointeur sans lui donner d'adresse car il peut pointer sur n'importe quel élément par défaut, pour éviter cela il faut toujours déclarer un pointeur en lui donnant la valeur 0.*
- *L'adresse 0 n'existe pas , du coup lorsque nous créons un pointeur contenant l'adresse 0, cela signifie qu'il ne contient l'adresse d'aucune case.*

2.4- Affectation et manipulation des pointeurs

Les pointeurs sont des variables particulières, puisque leurs valeurs sont des adresses mémoires. Ils peuvent néanmoins être impliqués dans des affectations au cours desquelles des adresses sont assignées aux pointeurs.

- Pour vider un pointeur, c'est à dire annuler l'adresse qu'il contient, on lui affecte une valeur prédéfinie nommée 0.

Syntaxe :

*Type *variable(0) ;*

Exemple :

int *Nbre(0);

- Pour récupérer l'adresse d'une variable il faut :

Syntaxe :

variablepointeur ← &variable;

Exemple :

int *p(0); *// déclaration du pointeur p qui ne pointe sur aucun élément.*

int nb; *// déclaration de la variable nb.*

nb=10; *//affectation de la valeur 10 à nb.*

p=&nb; *//affecte l'adresse de la variable nb au pointeur p*

- On peut affecter un pointeur avec tout autre pointeur de même type. Après cette affectation, les deux pointeurs désigneront la même zone de mémoire

Syntaxe :

variablepointeur1 = variablepointeur2;

Exemple :

int *p1(0),*p2(0);*// déclaration des pointeurs p1 et p2 qui ne pointent sur aucun élément.*

int nb; *// déclaration de la variable nb.*

nb=10;*//affectation de la valeur 10 à nb.*

p2 = &nb; *//affecte l'adresse de la variable nb au pointeur p*2

p1 = p2; *//le pointeur p1 pointe sur la même zone mémoire que p2*

Syntaxe : 2.2- Allocation dynamique d'une zone de mémoire

Lorsque l'on déclare un pointeur int *Nbre, le système réserve un entier pour stocker l'adresse vers laquelle pointe Nbre , il ne réserve pas de place pour la variable pointée par Nbre . Le fait de réserver un espace-mémoire pour stocker la variable pointée par Nbre s'appelle allocation dynamique.

Syntaxe :

variablepointeur = new type;

NB :

- Pour accéder à la valeur d'une variable pointée il faut : **variablepoinnteur*;
- Affecter une valeur à la variable pointée : **variablepointeur = valeur*;

Exemple :

int *Nbre(0);*//déclaration du pointeur nbre*

Nbre = new int;*// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre = 35;//Affecter 35 à la variable pointée*

cout<<*Nbre ;*//accéder à la valeur de la variable pointée c'est à dire 35*

Syntaxe : 2.3- Libération d'une zone de mémoire

Elle permet la libération de la place occupée par une variable et laisse la valeur du pointeur en l'état (n'efface pas l'adresse qui est dans la variable pointeur).

Syntaxe :

delete variablepointeur;

Remarque :

Si on fait appel au pointeur libéré , il renvoie une information qui n'a aucun sens.

Exemple :

int *Nbre(0);*//déclaration du pointeur nbre*

Nbre = new int;*// réserver un espace-mémoire pour stocker la variable pointée par Nbre*

Nbre = 35;//Affecter 35 à la variable pointée*

cout<<*Nbre ;*//accéder à la valeur de la variable pointée c'est à dire 35*

delete Nbre *;//on libère la zone mémoire*

Remarque :

- Attention, le fait de mettre 0 dans un pointeur ne libère pas l'emplacement sur lequel il pointait. L'emplacement devient irrécupérable car le lien vers cet emplacement a été coupé par la valeur 0. Il faut le libéré avant d'affecter 0 au pointeur.
- Lorsqu'on libère un pointeur, il faut lui affecter la valeur 0, pour qu'il ne conserve pas une adresse mémoire qui n'a plus d'existence physique.

Application 2 :

IV

Exercice

Partie 1 :

En respectant les instructions,précédent chaque trou, veuillez ajouter les valeurs manquantes

NB : Toutes les réponses doivent être en minuscule et sans espace

```
#include <iostream>
using namespace std;
main()
{
int nb1,nb2,nb3;
int *p1(0),*p2(0);
nb1 =5;
// avec le pointeur p1 réserve de l'espace pour stocker un entier
                = 
// stocker la valeur de nb1 dans la zone mémoire réservée ci-dessus et par la suite p1 doit récupérer l'adresse de nb1
                = ;
                = ;
nb2 = 3;
nb3 = 2;
//Faites que le pointeur p2 pointe sur la même zone mémoire que p1
                = ;
// Modifier la valeur de la zone mémoire pointée par p2  par la valeur de nb2
                = ;
// Donner l'adresse de nb2 à p2
                = ;
// en utilisant uniquement p1 augmenter la valeur de la zone mémoire réservée de 3
                += ;
// afficher la valeur de la variable pointée de p1
                "p1 ="<<                ;
// affiche la valeur de la variable pointée de p2
                "p2 ="<<                ;
}
```

Partie 2 : Veuillez remplir le tableau avec les valeurs qui conviennent après l'exécution du programme

Application 2 :

variable	adresse	valeur
nb1	adr1	
nb2	adr2	
nb3	adr3	
p1	adr4	
p2	adr5	

3 - Travaux dirigés

V

Application :

Ce exercice a pour but de comprendre l'utilisation et la manipulation des pointeurs.

```
1 #include <iostream>
2 using namespace std;
3 main()
4 {
5 int nb1,nb2;
6 int *p1(0),*p2(0);
7 nb1 =15;
8 p1 = new int; // on réserve de l'espace pour stocker un entier
9 *p1 = nb1; // stocke 15 dans la zone mémoire pointée par p1
10 cout << "La valeur de p1 est :"<<*p1<<endl;  // affichera La valeur de p1 est : 15
11 nb1 =12;
12 cout << "La valeur de p1 est :"<<*p1<<endl;  // affichera La valeur de p1 est : 15
13 p2 = p1; // p2 va pointer la zone mémoire pointée par p1
14 cout << "La valeur de p2 est :"<<*p2<<endl;  // affichera La valeur de p2 est : 15
15 p2^=nb2; // on place la valeur de nb2 qui est 9 dans la zone mémoire pointée par
   p2
16 cout << "La valeur de p1 est :"<<*p1<<endl;  // affichera La valeur de p2 est : 9
17 cout << "La valeur de p2 est :"<<*p2<<endl;  // affichera La valeur de p2 est : 9
18 *p1 = 14 ; // stocke 14 dans la zone mémoire pointée par p1
19 *p2 +=1 ; // stocke 15 dans la zone mémoire pointée par p2
20 cout << "La valeur de p2 est :"<<*p2<<endl;  // affichera La valeur de p2 est : 15
21 delete p1; // on libère la zone mémoire pointée par p1 et p2
22 p1 = 0; //efface l'adresse mémoire qui n'a plus d'existence physique après avoir
   libéré
23 p2 =0; //efface l'adresse mémoire qui n'a plus d'existence physique après avoir
   libéré
24 }
25
```