eBook Gratuit

APPRENEZ pointers

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#pointers

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pointers

It is an unofficial and free pointers ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pointers.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec des pointeurs

Remarques

Cette section fournit une vue d'ensemble de ce que sont les pointeurs et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans les indicateurs, et établir un lien avec les sujets connexes. La documentation des pointeurs étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Premiers pas avec les pointeurs

Les pointeurs sont des variables qui stockent l'adresse d'une autre variable.En tant que fonctionnalité de langue, ils sont disponibles dans plusieurs langages de programmation tels que, mais sans s'y limiter:

- Aller
- C / C ++
- Ada
- Pascal
- C # (disponible sous certaines contraintes)
- COBOL
- FORTRAN

Pour commencer à utiliser les pointeurs C / C ++, procédez comme suit:

1. Installez le compilateur comme Minimalistic GNU pour Windows, http://www.mingw.org/wiki/Getting_Started

2. Accédez au dossier d'installation de g ++ binary via la ligne de commande, par exemple:

```
C:\MinGW\bin>
```

3.Créez un fichier texte et écrivez ce programme C ++

```
#include <iostream>

int main () {

   int  pointed=0;
   int* ptr = & pointed;

   std::cout<<"Address of pointed variable is: "<<ptr<<std::endl;
```

```
   return 0;
}
```

4. Enregistrer sous pointer.cpp 5.Sur l'invite de commande, exécutez la commande

   g ++ -o pointer.exe -c pointer.cpp

6. Dans le répertoire de travail, vous obtiendrez un exécutable en tant que pointer.exe, cet exe exécutant donnera des résultats comme

   L'adresse de la variable pointée est: 0x7e892dac0a0c

Si vous recevez la sortie ci-dessus, vous avez écrit votre premier programme de pointeur

Qu'est-ce qu'un pointeur?

C'est essentiellement l'adresse d'une variable en mémoire. Cela nous permet d'accéder indirectement à une variable. Donc, en utilisant des pointeurs, nous pouvons parler de l'adresse d'une variable (ainsi que de sa valeur en supprimant le pointeur). Ils sont utiles lorsque nous voulons traiter l'adresse d'un emplacement de mémoire plutôt que sa valeur.

Considérons la fonction d'échange simple suivante dans C:

```
void Swap(int firstVal, int secondVal)
{
 int tempVal = firstVal;
 firstVal = secondVal;
 secondVal = tempVal;
}
```

maintenant dans main si nous avons le code suivant:

```
.
.
int a = 9,b = 100;
swap(a,b);
//print a and b
.
.
```

Les valeurs de a et b resteraient inchangées comme cela serait clair en imprimant leurs valeurs dans la fonction principale. Pour implémenter correctement la fonction swap, au lieu de passer les valeurs des variables `a` et `b` , nous passons l'adresse des variables a et b comme:

```
swap(&a,&b);
```

L'opérateur `&` , renvoie l'adresse de la variable. Son utilisé comme ceci:

```
int *address_of_a = &a;
```

`int *address_of_a` indique que la variable `address_of_a` pointe vers (stocke l'adresse de) une variable entière.

Maintenant, notre fonction de swap correcte serait:

```
void Swap(int *firstaddress, int *secondaddress)
{
 int tempVal = *firstaddress;
 *firsaddress = *secondaddress;
 *secondaddress = tempVal;
}
```

Maintenant, les valeurs échangées se refléteraient dans la fonction principale:

```
int a = 9,b = 100;
swap(&a,&b);
//print
```

Vous pouvez toujours déréférencer le pointeur avec `*` si vous ne possédez pas la variable d'origine. Supposons que, dans une fonction, vous ne disposez pas de la variable d'origine mais que son adresse se trouve dans une variable de pointeur `int *x` . Nous pouvons simplement accéder à la valeur de l'adresse mémoire en tant que `value = *x` ;

Si nous n'avions pas de pointeurs, nous ne pourrions jamais émuler le **passage par référence** dans `C` , car `C` est un **passage par valeur** . Mais rappelez-vous que nous ne pouvons **émuler** que parce que, même lorsque nous utilisons des pointeurs, l' `int *firstaddress, int *secondaddress` ne sont que des variables de pointeur locales créées, qui ont l'adresse des variables `a` et `b` .

Lire Démarrer avec des pointeurs en ligne: https://riptutorial.com/fr/pointers/topic/9295/demarrer-avec-des-pointeurs

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec des pointeurs	Community, endTunnel, Sumeet Singh