C/C++: Les pointeurs.

Vincent Gaudillière

Polytech Nancy

2017 / 2018

Adresse d'une variable.

Que se passe-t-il lorsqu'on déclare une variable?

Adresse d'une variable.

Que se passe-t-il lorsqu'on déclare une variable?

Réponse: On réserve un **emplacement en mémoire** pour stocker notre variable. Cet emplacement est identifié de manière unique par son **adresse**.

Adresse d'une variable.

Que se passe-t-il lorsqu'on déclare une variable?

Réponse: On réserve un **emplacement en mémoire** pour stocker notre variable. Cet emplacement est identifié de manière unique par son **adresse**.

```c
int a = 10;

printf("Valeur de a: %d\n",a); // 10
printf("Adresse de a: %p\n",&a); // 0x7ffda03ad294
```

Pointeurs et variables.

Qu'est-ce qu'un pointeur?

Réponse: Un pointeur est une variable qui contient l'**adresse** d'une variable d'un type donné. Un pointeur est **typé**.

Pointeurs et variables.

Qu'est-ce qu'un pointeur?

Réponse: Un pointeur est une variable qui contient l'**adresse** d'une variable d'un type donné. Un pointeur est **typé**.

```c
int a = 10; // a est une variable de type 'int'.
int * p_a = &a; // p_a est un pointeur sur un 'int'.

printf("Valeur de a: %d\n",a); // 10
printf("Valeur de a: %d\n",*p_a); // 10

printf("Adresse de a: %p\n",p_a); // 0x7ffda03ad294
printf("Adresse de a: %p\n",&a); // 0x7ffda03ad294
```

Question.

Pourquoi le code suivant a-t-il de grandes chances d'entraîner une faute de segmentation?

```
int * p_a;
*p_a = 10;
```

Question.

Pourquoi le code suivant a-t-il de grandes chances d'entraîner une faute de segmentation?

```
int * p_a;
*p_a = 10;
```

Réponse:

```
double x;
printf("Valeur de x: %lf\n",x); // ???
```

Question.

Pourquoi le code suivant a-t-il de grandes chances d'entraîner une faute de segmentation?

```
int * p_a;
*p_a = 10;
```

Réponse:

```
double x;
printf("Valeur de x: %lf\n",x); // ???
```

```
int * p_a; // p_a prend ici une valeur aléatoire
*p_a = 10; // on tente d'écrire à une adresse aléatoire!
```

Pointeurs et tableaux.

- L'adresse d'un tableau est identifiée à l'adresse de son premier élément,
- Il existe une conversion du nom du tableau en pointeur vers sa première case.

```
int mon_tableau[5] = {1,2,3,4,5};
printf("%p\n",&mon_tableau[0]); // 0x7ffd47dbf700
printf("%p\n",mon_tableau); // 0x7ffd47dbf700
```

```
/* Arithmétique des pointeurs (+,- avec entiers) */
printf("%d\n",mon_tableau[0]); // 1
printf("%d\n",*mon_tableau); // 1
printf("%d\n",mon_tableau[1]); // 2
printf("%d\n",*(mon_tableau+1)); // 2
```

Pointeurs et fonctions.

- Lorsqu'une variable est passée en paramètre d'une fonction, une copie de cette variable est créée.
- C'est cette copie qui sera éventuellement modifiée dans la fonction.
- Cette copie, à l'instar de toutes les variables déclarées dans la fonction, sera supprimée à la fin de l'appel.

```
int a = 1,b;
b = triple(a);
```

```
int triple(int n) // une copie n de a est créée.
{
    return 3*n;
} // n est supprimée
```

Passage par valeur.

```c
int main(){
    int a = 1;
    printf("%d\n",a); // 1
    triple(a);
    printf("%d\n",a); // 1
}
```

```c
void triple(int n)
{
    n = 3*n;
}
```

Passage par adresse.

```c
int main(){
    int a = 1;
    int * p_a = &a;
    printf("%d\n",a); // 1
    triple_pointeur(p_a);
    printf("%d\n",a); // 3
}
```

```c
void triple_pointeur(int * p)
{
    *p = 3*(*p);
}
```