INF155 - Pointeurs

Hugo Leblanc

Cours 4

Les variables

Rappel des variables

- Une variable est un identificateur, une valeur, et un type.
- Le type de la variable réserve un **espace mémoire** spécifique à la taille requise du type.
- L'espace mémoire réservé a aussi une adresse dans la mémoire de l'ordinateur. Habituellement représenté sous forme hexadécimale (0x43AF).
- Nous verrons aujourd'hui comment utiliser les adresses mémoires pour venir agir sur les informations stockées en mémoire.

L'adresse d'une variable

- L'opérateur & permets d'obtenir l'adresse d'une variable.
- L'opérateur de formatage %p est utilisé avec printf pour afficher les adresses mémoires en hexadécimale.

Les pointeurs

Déclaration de variables pointeurs

- Un pointeur est le nom désigné à la valeur d'une adresse en mémoire qui contient une information. La valeur "pointe" l'adresse mémoire.
- Une variable peut être déclarée comme étant un pointeur à l'adresse d'une information en mémoire.

- L'opérateur * est utilisé à la déclaration pour indiquer que la variable créée est un pointeur.

Exemple de déclaration de pointeurs

```
int * un_pointeur_int;
```

- Remarquons qu'il faut quand même donner le type de données "pointé" à la déclaration. Si le type est inconnu, le type void est aussi disponible.

Le déréférencement

La déréférence

- L'opérateur * est aussi utilisé dans les expression et assignation pour faire référence au contenu de l'adresse pointé.
- Faire attention à quand le * est utilisé. L'utilisation à la déclaration et dans un expression est très différente.

```
int x;
int * un_pointeur;
un_pointeur = 0x34AF; // Adresse en mémoire arbitraire
x = *un_pointeur + 12; // Ajoute le contenu de l'adresse à 12
*x = 15; // La déréférence est aussi possible en assignation
```

Passage par référence

- L'utilisation des pointeurs et des déréférences permettent de faire la modification de valeur en dehors des contextes des fonctions.
- Cela nous permet de faire une modification à une variable qui sera permanente après l'appel de la fonction.
- Une fonction qui a besoin de plus qu'un retour peut aussi utiliser ce mécanisme.

Exemple de passage par référence

```
void changer_x(int * x, int nouv_valeur){
    *x = nouv_valeur;
```

```
}
```