

INF3135

Construction et maintenance de logiciels

Cours 5: Pointeurs

Alexandre Blondin Massé

Université du Québec à Montréal
Département d'informatique

Été 2020

Table des matières

1 Quiz 1

2 Travail pratique 1

3 Résumé du chapitre 2

4 Résumé du chapitre 3

Quiz 1

Retour sur le quiz 1

Statistiques

- **3** questions courtes ou à choix multiples
- **1** question longue
- **Moyenne:** 76.3%
- Questions courtes **bien réussies**
- Question longue **plus variable**

Problèmes rencontrés

- Envoi automatique du quiz
- Pas possible de déposer un fichier
- Pas le temps de joindre le fichier
- Copier-coller ne fonctionnait pas
- Autres?

Travail pratique 1

Travail pratique 1

- Date de remise: **18 juin**
- **20%** de la note totale
- Doit être fait **seul**
- **Dépôt GitLab**: doit être *forké*, sa visibilité mise à *privé*, puis l'accès en mode *Developer* doit être donné à blondin_al
- **Description du travail**: dans le fichier `sujet.md`
- À **compléter**: `canvascii.c`, `Makefile`, `.gitignore`, `README.md`
- À **modifier**: dans `check.bats`, supprimer `skip` pour activer le test

Résumé du chapitre 2

Chapitre 2: Outils de développement logiciel

Style de programmation

Présentation du code, syntaxe, nomenclature, organisation des fichiers

Documentation

Docstrings, Javadoc, Markdown

Bats

Installation, tests unitaires, variables spéciales (status, output, lines), invocation, skip, protocole TAP

Git

log, gr, init, clone, status, checkout, diff, show, add, commit, reset, remote, fetch, pull, push, etc.

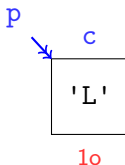
GitLab-CI

Fichier `.gitlab-ci.yml`, lancement de tests, lecture de *logs*

Résumé du chapitre 3

Adresse et pointeur

- **Adresse**: indique emplacement d'une donnée
- **Opérateur &**: retourne l'adresse d'une *left-value*
- **Pointeur**: *left-value* qui contient une adresse
- Pointeurs **typés**: `int*`, `double**`, `struct point*`, etc.
- Plusieurs **types**: nul, constant (`const`), générique (`void*`), de fonction



Deux « sortes » de pointeurs constants

- `const <type> *p`: pointeur en lecture seule
- `<type> *const q`: pointeur constant

Opérations sur les pointeurs

Déférencement

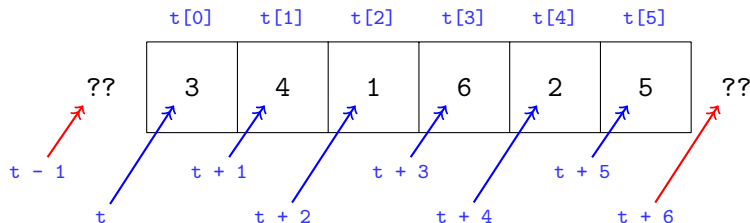
- *: accès à la donnée « pointée »
- ->: déréférencement du champ d'une structure

Autres opérations

- =: affectation
- (type*): conversion
- ==/!=: égalité et différences d'adresses
- <=, >=, <, >: comparaison d'adresses
- +/-: retourne un pointeur décalé
- ++/--: incrémentation et décrémentation

Tableaux et arithmétique des pointeurs

```
int t[] = {3, 4, 1, 6, 2, 5};
```



- On a `t + i == &t[i]`
- De façon équivalente, `*(t + i) == t[i]`

Chaînes de caractères

```
char s1[10];      // Tableau de caractères de taille fixe
char *s2;         // Pointeur vers début d'une chaîne
const char *s2;   // Chaîne en lecture seule
```

Bibliothèque `string.h`:

- `strcat`: concatène une chaîne à la suite d'une autre
- `strncat`: concatène une chaîne à une autre en tronquant
- `strcpy`: copie une chaîne dans une autre
- `strncpy`: copie une chaîne dans une autre en tronquant
- `strlen`: longueur d'une chaîne
- `strcmp`: compare deux chaînes
- `strncmp`: compare deux chaînes en tronquant
- `strchr`: cherche un caractère dans une chaîne de gauche à droite
- `strrchr`: cherche un caractère dans une chaîne de droite à gauche
- `strstr`: cherche une chaîne dans une autre de gauche à droite
- `strrstr`: cherche une chaîne dans une autre de droite à gauche
- `strtok`: segmente une chaîne en morceaux (*tokens*)

Pointeurs de fonctions

```
// Pointeur de fonction de type int -> int
int (*f)(int x);
// Pointeur de fonction de type (int, int) -> int
int (*g)(int x, int y);
```

Bibliothèque stdlib.h:

```
// Trie les valeurs d'un tableau
void qsort(void *base,
           size_t nmemb,
           size_t size,
           int (*compar)(const void *, const void *));

// Effectue une fouille binaire dans un tableau
void *bsearch(const void *key,
              const void *base,
              size_t nmemb,
              size_t size,
              int (*compar)(const void *, const void *));
```

Exercice

1. Écrire une fonction `strlower` qui transforme une chaîne en minuscules
2. Écrire un programme qui trie un ensemble de points 2D selon l'ordre **polaire**, c'est-à-dire d'abord selon la distance par rapport à l'origine et, quand il y a égalité, selon l'angle

