Programmation 1: Laboratoires

Éric Taillard

Filière informatique et sciences de la communication

eric(point)taillard(arrobase)heig-vd.ch

16 septembre 2021

Laboratoire 1

Donnée

- **Buts**
  - Prise en main de l'environnement informatique de la HEIG-VD
  - Installation d'un compilateur, si nécessaire
  - Maîtriser les bases de la console de commandes
- **Travail à réaliser**
  - Saisir le premier programme « Hello world ! » à l'aide d'un éditeur de texte (Notepad, Notepad++, Wordpad, gedit, etc.) et le sauver sous : `votre_nom_labo1.cpp`
  - Pour les utilisateurs du système d'exploitation Windows qui n'ont pas de compilateur g++ installé :
    - Voir page `https://sourceforge.net/projects/mingw-w64/files/latest/download`
    - Ajouter le chemin `... mingw64\bin` aux variables d'environnement du système (taper `env` dans la loupe)
    - Ouvrir une console (lancer cmd.exe)
  - Se déplacer dans la console dans le bon sous-répertoire, compiler et exécuter le programme :

    ```
    g++ votre_nom_labo1.cpp -o votre_nom_labo1.exe
    votre_nom_labo1.exe
    ```
  - Imprimer le code source du programme (s'assurer que votre nom, celui du fichier, la date et le numéro de page figurent sur l'en-tête ou le pied de page)

À rendre

- Le code assembleur de votre programme (obtenu avec `g++ -S votre_nom_labo1.cpp`)
- À déposer dans Moodle

Trousse de secours des commandes de la console

Commande unix	Commande DOS	Exemple	Effet
`cd`	`cd`	`cd xyz`	Descendre dans le sous-répertoire `xyz`
		`cd ..`	Remonter d'un répertoire
`cp`	`copy`	`cp toto.txt xyz`	Copier le fichier `toto.txt` dans le répertoire `xyz`
		`cp aa.txt bb.txt`	Copier le fichier `aa.txt` dans le fichier `bb.txt`
`ls`	`dir`	`ls xyz`	Lister le contenu du sous-répertoire `xyz`
`man`	`help`	`man cp`	Manuel de la commande `cp`
`mv`	`move`	`mv toto.txt xyz`	Déplacer le fichier `toto.txt` dans le répertoire `xyz`
`more`	`type`	`more toto.txt`	Afficher le contenu du fichier `toto.txt`

Laboratoire 2

Écrire un programme C++ qui :

1. Demande à l'utilisateur d'entrer son prénom
2. Lit la réponse de l'utilisateur (on supposera celle-ci correcte) et la stocke dans une variable prenom de type
3. Demande à l'utilisateur d'entrer son âge
4. Lit la réponse de l'utilisateur (on supposera celle-ci correcte) et la stocke dans une variable age de type entier int
5. Calcule l'année de naissance (à un an près) de l'utilisateur et l'enregistre dans une variable annee_naissance de type entier int
6. Affiche à l'écran le message suivant :
   `Bonjour <prenom>,`
   `Vous avez <age> ans et vous êtes ne(e) en <annee_naissance>.`

Indications :

- Pour lire le prénom :
  - ajouter la ligne `#include <string>` avant le `main`
  - insérer les 2 lignes de code suivantes dans le `main` :
    ```
    string prenom;
    getline(cin, prenom);
    ```
- Pour lire l'âge, utiliser l'instruction : `cin » age;`

Laboratoire 3

- **But**
  - Écrire par imitation un programme de transformation d'unité
- **Donnée**
  - Compiler et tester le fonctionnement du programme de transformation d'une longueur exprimée en mètres vers des pieds et des pouces.
  - S'inspirer de ce programme pour en écrire un qui transforme une durée, exprimée en secondes uniquement, vers des semaines, jours, heures et secondes
    - Exemple : si on introduit : 4075382 secondes, le programme doit répondre : 4075382 s = 6 semaines, 5 jours, 4 heures, 3 minutes et 2 secondes
- **Qualité du code**
  - Comme pour tous les programmes à rendre, un commentaire au début du code doit indiquer quelle est la fonction du programme, qui en est l'auteur, la date et le numéro du laboratoire.
  - Utiliser des constantes symboliques nommées plutôt que des nombres « magiques »
  - Commenter le code pour en expliquer les principales étapes, sans le charger inutilement
  - Choisir des noms d'identificateurs parlants
  - Compiler avec les options `g++ -O2 -Wall -Wextra -Wconversion -Wsign-conversion -Wvla -pedantic`. Ainsi, le compilateur nous aide à découvrir des erreurs sémantiques
- **Délai**
  - Fin de la séance