A mettre apres les chapitre sur les fonctions et les tests ?

Ce chapitre est encore en cours de rédaction, voire à l'état d'ébauche. N'hésitez pas a faire des remarques ou poser des questions sur le forum de Zeste de Savoir ou de OpenClassroom.

Entrée console

qu'est ce que la console/terminale? Comment la lancer?

On connait déjà cout, qui signifie "output C console", permet d'envoyer des messages sur la console. Egalement possible d'entrée des valeurs depuis la console avec `std::cin`

Note : non testable sur Coliru.com, besoin de compiler en local.

Permet d'attendre des instructions de l'utilisateur. Mode interactif : on lance le programme, celui ci s'arrete, le temps de saisir des infos.

Different de ligne de commande et argc/argv ⇒ info envoyee lors du lancement du programme, pas besoin de mettre en pause ou interaction dynamique avec l'utilisateur. (Note : creation de scripts automatique, lancee par des robots, plus pratiques d'utiliser la ligne de commande. Certain outils propose les 2, comme gdb -i pour lancer en mode interactif)

Les 2 sont possible.

Exemple : (ligne commencant par `>` = ce que l'utilisateur tape sur une console/terminal)

```
> dosomething unfichier.txt
Vous voulez ouvrir le fichier "unfichier.txt". Confirmer
(o/n).
> o
Lecture du fichier "unfichier.txt" en cours...
```

```
Fin de lecture du fichier.
```

Premiere ligne = lance le programme “dosomething” avec un argument “unfichier.txt”. Seconde ligne : le programme demande confirmation et attend une reponse (taper sur les touches “o” ou “n” puis entrer). Troisieme ligne: le programme continue.

Ici, 2 modes d'entree d'inso : via argument lors du lancement, et en interactif. Le programme attend la confirmation avant de poursuivre son travail.

Saisie simple

Creation d'une variable, puis utilisation de `»` avec `std::cin`. Typage fort du C++ fait que `std::cin` lit correctement le type de valeur selon le type.

main.cpp

```cpp
#include <iostream>

int main() {
    int i {};
    std::cin >> i;
    std::cout << "input: " << i << std::endl;
}
```

Pour lancer le programme :

```
> a.out 123
input: 123
```

Entrer une ligne de texte

Pour entrer une chaine. Essayons avec :

main.cpp

```cpp
#include <iostream>
#include <string>
```

```
int main() {
    std::string s {};
    std::cin >> s;
    std::cout << "input: " << s << std::endl;
}
```

Pour lancer le programme :

```
> a.out hello
input: hello
```

Pas de probleme. Autre test :

```
> a.out hello world
input: hello
```

Oups. Les espaces sont utilise pour indiquer une autre entree (comme pour les lignes de commande. Rappel : “a.out hello world” argc = 3)

Pour ligne une ligne complete (ie tout ce qu'il y a jusqu'a “Enter”).

main.cpp

```
#include <iostream>
#include <string>

int main() {
    std::string s {};
    std::getline(std::cin, s);
    std::cout << "input: " << s << std::endl;
}
```

Avec

```
> a.out hello world
input: hello world
```

Ok, ca lit correctement.

Lire plusieurs entree

Probleme de \n restant, ajouter ignore.

Valider les entrees

Taper du texte ou autre alors que entier attendu. Comment regler cela ? Tester le retour de cin, clear + ignore.

Autre solution ? Utiliser string + conversion stoi/stof, ou regex, ou autre.

Attendre une touche quelconque

Avec ignore.

Chapitre précédent	Sommaire principal	Chapitre suivant