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.

La ligne de commande

Lorsque l'on lance une application il est possible de lui envoyer des informations :

```
myapp.exe myfile.txt
```

Comment lire ces informations ?

Retour de la fonction main

La fonction `main` est obligatoire et doit respecter une signature imposée par la norme C++. Pour le moment, on utilise :

```
int main() {
}
```

Elle correspond à une fonction qui ne reçoit aucun paramètre et retourne un entier. Souvent, on ne mets pas l'entier (0 par défaut) ou on retourne un code d'erreur :

```
int main() {
    return 123456;
}
```

Retourner 0 ou EXIT_SUCCESS si pas d'erreur, EXIT_FAILURE ou un code d'erreur.

argc

Il existe une autre signature, avec des paramètres :

```
int main(int argc, char* argv[]) {
}
```

`argc` contient le nombre de paramètres passé lors de l'appel de la fonction. Chaque paramètre est séparé par une espace et le premier paramètre est le nom du programme. Le code suivant permet par exemple d'afficher le nombre de paramètres lors de l'appel du programme.

main.cpp

```
#include <iostream>

int main(int argc, char* argv[]) {
    std::cout << argc << std::endl;
}
```

Selon la ligne de commande utilisée :

```
./a.out
```

affiche :

```
1
```

La ligne de commande suivante :

```
./a.out hello world
```

affiche :

```
3
```

Il y a toujours au moins un argument, qui correspond au nom du programme. (Cela sera vu dans quelques instants).

argv

`argv` permet d'accéder aux chaînes de caractères correspondant à chaque paramètre. On reconnaît dans la déclaration le type `char*` correspondant aux littérales chaînes de caractères. Les crochets droits `[]` permettent de déclarer un tableau de type C. Le type de `argv` peut donc se lire comme étant un tableau de chaînes de caractères.

Les tableaux du style C

Moins safe que les collections du C++, besoin de conserver la taille dans une variable séparée.

Toujours vérifier les accès et préférer les tableaux du C++

Pour accéder aux éléments d'un tableau, on utilise également les crochets droits, en indiquant la position dans le tableau (en commençant à 0).

main.cpp

```cpp
#include <iostream>

int main(int argc, char* argv[]) {
    std::cout << argc << std::endl;
    std::cout << argv[0] << std::endl;
    std::cout << argv[1] << std::endl;
    std::cout << argv[2] << std::endl;
}
```

affiche si on l'appelle avec `./a.out in.txt out.txt` :

```
3
./a.out
in.txt
out.txt
```

Si on essaie d'afficher une valeur en dehors du tableau, le comportement est indéterminé. Pensez à toujours vérifier les accès aux tableaux.

Comme vous le voyez, le premier argument `argv[0]`, correspond au nom du programme.

Notez que même si `argv` n'est pas `const`, ce tableau ne doit pas être modifié.

convertir en vector

Possible de copier dans un `std::vector<std::string>` pour utiliser avec les algos standard.

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

int main(int argc, char* argv[]) {
    std::vector<std::string> args (argv, argv+argc);
    for (auto s: args) {
        std::cout << s << std::endl;
    }
}
```

affiche :

```
./a.out
hello
world
```

En pratique, il n'est pas nécessaire de copier dans un tableau `vector`, surtout que cela a un cout. Il est possible d'utiliser la paire `argv` et 'argv+argc'' avec les algos standard directement.

```
std::find(argv, argv+argc, "hello");
```

Mais, pour débuter, préférer les fonctionnalités du C++ plutôt que les anciennes syntaxes héritées du C.