

TP Arguments à la fonction main – Les structures - SN1

1. Passage d'arguments à la fonction main()

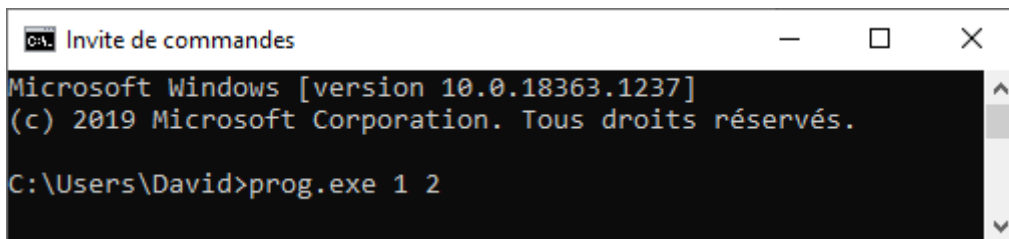
Rappel : la fonction *main()* est le point d'entrée du programme.

On peut donner en argument des valeurs au programme. Pour ce faire, il faut lancer le programme dans une console DOS (cmd) en faisant suivre le nom du programme par les arguments séparés par des espaces. Chaque argument est vu par la fonction *main()* comme étant des chaînes de caractères.

Un autre argument confié à la fonction *main()* est le nombre d'arguments. Attention, ce nombre tient compte du nom du programme. Ce nombre n'est pas saisi, c'est l'interpréteur de commandes qui compte le nombre d'arguments (programme compris) et le confie à la fonction *main()*

Explications :

Par exemple, notre programme s'appelle « prog.exe ». on veut lui « passer » les valeurs 1 et 2. On tapera alors :



```
Microsoft Windows [version 10.0.18363.1237]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\David>prog.exe 1 2
```

La fonction *main()* du programme *prog.exe* est définie comme telle :

```
#include <iostream>

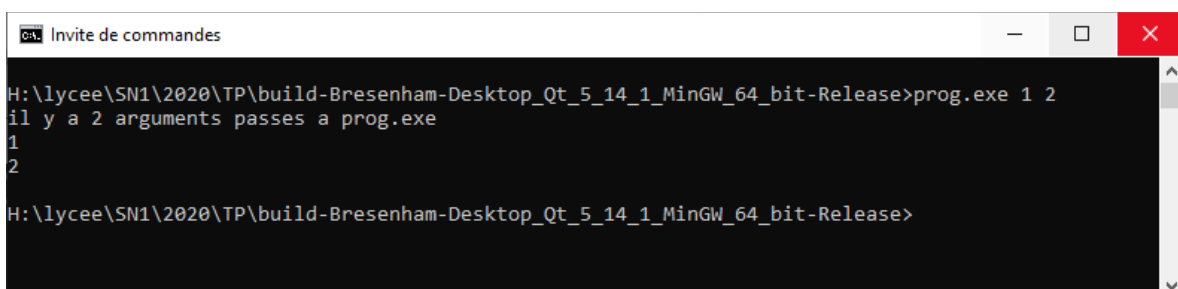
using namespace std;

int main(int argc, char *argv[])
{
    cout << "il y a " << argc-1 << " arguments passes a " << argv[0] <<
endl;

    for (int i = 1 ; i < argc ; i++)
    {
        cout << argv[i] << endl;
    }

    return 0;
}
```

⇒ Résultat de l'exécution :



```
H:\lycee\SN1\2020\TP\build-Bresenham-Desktop_Qt_5_14_1_MinGW_64_bit-Release>prog.exe 1 2
il y a 2 arguments passes a prog.exe
1
2
H:\lycee\SN1\2020\TP\build-Bresenham-Desktop_Qt_5_14_1_MinGW_64_bit-Release>
```

On peut constater que les valeurs 1 et 2 sont des chaînes de caractères tout comme le nom du programme.

Pourquoi déclare-t-on la variable contenant les arguments sous la forme : `char *argv[]` ?

Finalement, `argv[]` est équivalent à `*argv`

Donc `char *argv[]` est équivalent à `char **argv`

C'est donc un tableau de chaîne de caractères !

<i>argv</i> ->	0	1	2
0	'p'	'1'	'2'
1	'r'	'\0'	'\0'
2	'o'		
3	'g'		
4	'.'		
5	'e'		
6	'x'		
7	'e'		
8	'\0'		

Si on accède à la case `argv[0][1]` par exemple, on accède en fait au caractère 'r' de « prog.exe »

Si on accède à la case `argv[2][0]`, on accède au caractère '2' de l'argument « 2 »

TRAVAIL DEMANDE

Réaliser un programme *add* prenant en argument au moins deux entiers et affichant la somme de ces entiers.

Vous pourrez utiliser la fonction *atoi()* prenant en argument une chaîne de caractères représentant un entier et renvoyant l'entier correspondant. Par exemple :

`atoi("123")` renvoie la valeur 123.

2. Retour de la fonction *main()*

La valeur entière renvoyée par la fonction *main* peut être récupérée dans la console DOS. Après avoir exécuté votre programme, vous pouvez savoir comment il s'est terminé. Il suffit d'afficher la variable d'environnement `ERRORLEVEL` :

Taper `echo %ERRORLEVEL%`

Vous obtenez la valeur renvoyée par *main()*.

Exemple : dans le programme précédent, ajouter un test permettant de détecter que l'utilisateur n'a pas saisi suffisamment d'entiers. Par exemple, s'il tape : `add 1` alors la fonction *main()* se termine en renvoyant -1 (par exemple). Afficher l'`ERRORLEVEL` et vous devriez avoir -1 affiché

3. Les structures

Comme on a pu le voir en cours, les structures permettent de créer un nouveau type de données regroupant plusieurs données de types différents.

Par exemple, on peut créer le type *Eleve* regroupant les données suivantes :

- Nom : char => chaîne de caractères
- Prénom : char * => chaîne de caractères
- Année de naissance : int => un entier
- Notes : double * => Tableau de réels

En C++ :

```
struct Eleve
{
    char* Nom;
    char* Prenom;
    int AnneeDeNaissance;
    double* Notes;
};
```

Déclaration d'une variable E1 de type Eleve :

```
Eleve E1;
```

Si dans la fonction *main()*, on déclare cette variable E1 de type *Eleve*, on ne pourra affecter une valeur qu'au champ « AnneeDeNaissance ». Les autres champs sont des pointeurs. Autrement dit, aucune réservation mémoire n'a été faite pour ces pointeurs => ils pointent nulle part !!

Il faut réserver de l'espace mémoire pour y mettre les chaînes de caractères ou la liste des notes obtenues.

L'opérateur *new* nous permet de réserver un certain nombre d'octets en mémoire. Par exemple :

Si on veut saisir au clavier un nom d'élève et affecter le champ Nom de la variable E1 déclarée précédemment, on pourra écrire :

```
struct Eleve
{
    char* Nom;
    char* Prenom;
    int AnneeDeNaissance;
    double* Notes;
};

int main(int argc, char *argv[])
{
    Eleve E1;
    char *E;
    cin>>E;

    E1.Nom = new char[strlen(E)+1];
    strcpy(E1.Nom, E);

    .....
    .....
}
```

TRAVAIL DEMANDE

Créer un programme permettant de saisir les noms, prénoms, année de naissance et 5 notes d'un élève. Vous utiliserez une boucle pour saisir les 5 notes. Il faudra, évidemment, réserver l'espace mémoire nécessaire à l'aide de *new*

- Tableau d'élèves

On désire désormais enregistrer les informations et notes des élèves d'une classe. Pour cela, on décide de créer un tableau d'élèves. On se contentera d'un tableau déclaré en statique de 35 élèves :

```
Eleve Classe[35];
```

Comment accéder au nom du deuxième élève de notre tableau ?

Comment saisir la 5^{ème} note du 10^{ème} élève de notre tableau ?

TRAVAIL DEMANDE

Modifier le programme précédent de façon à permettre la saisie des trois premiers élèves de la classe.

Indication : il va falloir mettre en place une boucle imbriquée dans une autre