# Programme et module :

## Introduction :

À ce stade les notions de programme et module s'imposent. Nous allons répondre aux questions suivantes :

- Qu'est-ce qu'un programme ?
- A quoi sert un programme ?
- Qu'est-ce qu'un module ?

**Qu’est-ce qu’un programme ?**

Un programme correspond au code d’un exécutable dans lequel on peut faire un traitement sur les données, les afficher, les lire en entrée etc..

## Déclaration d’un programme :

Pour indiquer qu’un fichier EZ est un programme, il faut le déclarer comme suit :

```
program [program_name]

[…]

procedure [program_name]

[instructions..]

end procedure
```

La procédure portant le nom du programme est considérée comme étant le main en c. Le but est de définir le comportement du programme.

## Comment passer des arguments en ligne de commande ?

Il existe deux méthodes pour passer des arguments comme paramètres :

1. Envoi de valeurs en ligne de commandes :
   Commande : `./program_name.exe --var1=arg1 --var2=arg2`

2. Envoi d'un fichier de données en ligne de commandes :
   `ezc program_name.ez -o program_name.exe`
3. Pour l'envoi d'un fichier data, il faut passer le fichier comme option lors de la compilation.
   Commande : `./program_name.exe --data [data_file_path.dez]`

Exemple :

```
program hello_arg

global max_i is integer = 5

procedure showData (x is integer, c is string)
     print x, " fois ", c
end procedure

procedure hello_arg
     i is integer =0

     for i in 0.. max_i
     do
          showData(x,c)
     end for

end procedure
```

récupération des arguments :
`./hello_arg.exe --data data.dez`

Fichier data.dez :

x=5
c=20

## Comment récupérer la valeurs des arguments passés comme paramètre ?

La récupération des valeurs se fait automatiquement à partir de la ligne de commande. Pour les arguments passés en paramètre la syntaxe est la suivante :

Exemple :

```
program hello_arg

arguments
     x is integer as "--x"
     c is string as "--c"
end arguments

global max_i is integer = 5

procedure showData (x is integer, c is string)
     print x, " fois ", c
end procedure

procedure hello_arg
     i is integer =0

     for i in 0.. max_i
     do
          showData(x,c)
     end for

end procedure
```

récupération des arguments :
./hello_arg.exe --x=2 --c= hello

**Contraintes à respecter :**

Les argument passés doivent être uniquement de l'un des types suivants :

- Entier
- Réel
- String.

**A quoi sert un programme ?**

Un programme utilise les modules existants qui utilisent des classes, des fonctions etc...

## Comment inclure un module dans un programme ?

Pour inclure un module EZ dans un programme il suffit d'écrire la ligne suivante :

```
import person
```

Pour inclure une bibliothèque cpp :

```
import "math.h"
```

## Comment inclure du code C++ dans un programme EZ ?

Il est possible d'écrire du code C++ dans les fichiers EZ, il suffit de préciser qu'il s'agit du code C++ pour que le compilateur le prend en compte.
Exemple :

```
total is real
code(cpp, total=sum)

     double sum= 0.0;

     for (int i=0 ; i<10;i++)
          sum+= i*2;
end code
```

Traduction du code en c++ :

```
void code1(double &sum){
     sum=0,0 ;
     for(int i=0 ; i<10 ; ++i) {
          sum+=i*2 ;
     }
}
int main(){
     double total ;
     code1(total) ;
     return 0;
}
```

cppcode est une fonction particulière qui permet de faire l'interface entre du code EZ et du c++.

**Attention :** En cas de code assembleur il faut juste remplacer cpp par asm.

## Qu’est-ce qu’un module ?

Un module permet de définir les fonctions et les classes à réutiliser.

```
module person

class person

name is string
id is integer
age is integer

function isOlderThan(old is integer) return boolean
     return old<age
end function

end class
```

## Inclusion des modules EZ et C++

Il existe plusieurs types de bibliothèques :
On distingue les modules écrits par l'utilisateur, les bibliothèques C++ qu'on réutilise en tant que module EZ et les bibliothèques c++ externes que l'utilisateur voudrait utiliser.

### 1- Les modules écrits par l'utilisateur :

Un module représente un ensemble de classes et méthodes. L'utilisateur peut également faire appel à des fonctions C++ qu'il faudra prendre en compte par le compilateur EZ.

Exemple :

```
function cos(x is real) is extern cos
```

**Dans ce cas il faut prendre en considération la bibliothèque math du C++. Attention il faut vérifier que le type de la variable(s) passé(es) en paramètre(s) est le même que la signature de la méthode cpp. La surcharge des méthodes est possible uniquement en C.**

### 2- Les bibliothèques C++ :

Les bibliothèques C++ sont déduites à partir du programme directement. Dans le cas où l'utilisateur voudrait utiliser une bibliothèque C++ particulière (non supportée par g++ ou gcc), l'utilisateur doit définir le nom de la bibliothèque et le chemin vers le fichier comme suit :

```
include "usr/include/math.h"
library "/usr/lib/src -lm"
```