Program declaration

Syntax

```
program [program_name]
[...]
procedure [program_name]
      [instructions..]
end procedure
```

The procedure that has the same name as the program is considered as the main, whose goal is to define the behaviour of the program. If there is no such procedure, the program is considered as a library which can be included in another program that will be able to use all functions, procedures, classes and variables defined in the library.

Warning : each EZ file must begin with program [program_name] as it defines the logical name of the file.

Importing libraries

Syntax

- Import an EZ library : `import [program_name]`
- Import a C++ library covered by g++ : `include "[library_name]"`
- Import an external C++ library : `library [path_to_l ibrary]`

Example 1

```
program person

class person
      id is integer
      name is string
      age is integer
      function isOlderThan(old is integer) return boolean
            return old<age
      end function
end class
```

```
program person_main
import person

// This is a program using the « person » module
procedure person_main
      driss is person(1,"Driss", 40)
      hossam is person(2, "Hossam", 20)
      if driss.isOlderThan(hossam.age)
            print driss.name," is older than ", hossam.name
      else
            print hossam.name," is older than ", driss.name
      end
end procedure
```

Example 2

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

Calling C++ functions

C++ functions can be called from an EZ program.

Example

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

In this case, the math library from C++ should be used. The parameters' types must match those in the signature of the C++ function.

Including C++ code

C++ code can be written inside an EZ program using the keyword **code** followed by the name of the language (cpp for C++, asm for Assembly, ...).

Example :

EZ	Translation in C++

`total is real`
`code(cpp, total=sum)`
 	
`double sum=0.0 ;`
`   for (int i=0; i<10;i++)`
`        sum+=i*2 ;`
`end code`
	`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;`
	`}`

Arguments via the command line

Send arguments

There are two ways to send arguments when executing the program with the command line :

- Send directly the values
  ```
  ./program_name.exe --var1=arg1 --var2=arg2
  ```
- Send a data file
  ```
  ./program_name.exe --data [data_file_path.dez]
  ```

Example of data file data.dez :

```
x=5
c=20
```

The arguments to send must be of one of the following types :

- integer
- real
- string

Get arguments

Example :

```
./hello_arg.exe --x=2 --c= hello
```

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