Variables et types primitifs

Identifiants

Un **identifiant** valide est une suite d'au moins une lettre pouvant comporter des chiffres et/ou des Under scores.

Par convention il est recommandé de le faire débuter par une lettre en minuscule et de ne pas faire succéder deux Under scores.

Un identifiant ne peut pas contenir d'espaces, de caractères spéciaux ou des mot-clés du langage.

Variables :

Les **variables** sont des symboles qui associent un identifiant à une valeur.

En d’autres termes, on peut définir une variable comme une zone mémoire permettant de stocker l'information accessible via son identifiant. Cependant, la zone de mémoire à réserver n’est pas la même selon le type de la variable.

La portée des variables (Scope) : variable locale

Il y a trois endroits où une variable peut être déclarée :

Une variable locale : à l'intérieur d'une fonction ou d'un bloc, chaque variable est considérée comme locale même si on précise pas le mot-clé `local`.

Si on déclare à l'intérieur d'une fonction une variable `global`, une erreur sera déclenchée.

Syntaxe pour une variable locale :

- **name** `is` **Type**
- `local` **name** `is` **Type**

Exemple :

EZ Language	C++
////////////////// Bloc //////////////////////	////////////////// Bloc //////////////////////
// Begin Function	// Begin Function

//Déclaration d'une variable local
a,b,c are integer
// Initialisation
a = 10
b = 20
c = a + b
print "C = "+ c
	//Déclaration d'une variable local
	int a,b,c;
	// Initialisation
	a = 10;
	b = 20;
	c = a + b;
	cout<<"C = "<<c;

//End Function	//End Function
Resultat : C = 30	Resultat : C = 30

La portée des variables (Scope) : paramètre et variable globale

Paramètre : Dans la définition d'une fonction.

Une variable globale : Déclarée en dehors des fonctions.

Syntaxe pour une variable globale :

- `global` **name** `is` **Type**

Exemple :

EZ Language	C++

// Déclaration d'une variable globale
global g is integer
 	
// Begin Function
 	
a,b are integer
// Initialisation
a = 10
b = 20
g = a + b
print "g = "+g
//End Function
 	
Resultat :
g = 30
	// Déclaration d'une variable globale
	int g;
	
	// Begin Function
	
	int a,b;
	// Initialisation
	a = 10;
	b = 20;
	g = a + b;
	cout<<"g = "<<g;
	//End Function
	
	Resultat :
	g = 30

Deux variables une locale et l'autre globale dans le même programme peuvent avoir le même nom.

Exemple :

EZ Language	C++

// Global variable declaration:
global g is integer =1
 	
// Begin Function
 	
g is integer
g=10
 	
print "g = "+g
 	
//End Function
 	
Resultat :
g = 10
	// Global variable declaration:
	int g=1;
	
	// Begin Function
	
	int g;
	g=10;
	
	cout<<"g = "<<g;
	
	//End Function
	
	Resultat :
	g = 10

Types primitifs :

Nous pouvons distinguer différentes catégories de types :

Entier : représente les valeurs entières positives ou négatives stockées de différentes manières selon les valeurs maximales pouvant être prises.

Réel : les réels représentent les nombres à virgules, avec une précision pouvant varier.

Chaine de caractères : ce type permet de stocker l'ensemble des caractères existants en se basant sur la classe string du C++ et dispose de fonctions particulières héritées de ce dernier.

Booléen : est un type de variable à deux états. Les variables de ce type sont soit à l'état vrai soit à l'état faux (true/false).

Type	Syntaxe	Taille de stockage en octets	Exemples
Entier	integer	2/4 octets, valeur de -32 768 à 32 767	1- Number is integer 2- number1, number2 are integer
Réel	real	Généralement 4 octets en mémoire , valeur de 1.2e-308 à 3.4e-38.	
Chaine de caractères	string	Identique au stockage du type composé string du C++	name is string
Booléan	boolean		flag is boolean

Constantes :

Une **Constante** est une expression à valeur fixe et fait en sorte que le compilateur empêche le programmeur de la modifier.

Syntaxe pour une constante :

`const` scope Name `is` type

Exemple :

`const local` number `is` integer

Exemple :

EZ Language	C++

// Begin Function
// constant declaration:
const c is integer = 5
c = 10
c++
//End Function
Resultat :
error: assignment of read-only variable ‘c’
	// Begin Function
	// constant declaration:
	const int c = 5;
	c = 10;
	c++;
	//End Function
	Resultat :
	error: assignment of read-only variable ‘c’

Afin de faciliter la lecture des littéraux numériques en EZ on peut utiliser les underscore.

Exemple : number `is integer` = 1_000_000