

## 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 débiter par une lettre en minuscule et de ne pas faire succéder deux Under scores.

`[a-zA-Z_][a-zA-Z0-9_]*`

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

### Les variables :

Syntaxe : **variable** name **is** type

EZ Language	C++	Result
<pre> program p  procedure p() begin  //Déclaration des variables locales variable a,b,c are integer  // Initialisation a = 10 b = 20 c = a + b print "C = "+ c  end procedure </pre>	<pre> #include &lt;iostream&gt;  int main () {  //Déclaration des variables locales int a,b,c;  // Initialisation a = 10; b = 20; c = a + b; std::cout&lt;&lt;"C = "&lt;&lt;c;  } </pre>	C = 30

EZ Language	C++	Result
<pre> program p  // Déclaration d'une variable globale   global  g is integer = 0  procedure p() begin  variable a,b are integer  // Initialisation   a = 10   b = 20   g = a + b   print "G = "+g  end procedure </pre>	<pre> #include &lt;iostream&gt;  //Déclaration d'une variable globale   int g = 0;  int main () {  int a,b;  // Initialisation   a = 10;   b = 20;   g = a + b;   std::cout&lt;&lt;"G = "&lt;&lt;g;  } </pre>	<p style="text-align: center;">G = 30</p>

## Les expressions régulières

Une expression régulière ou normale ou rationnelle est une chaîne de caractères que l'on appelle parfois un motif qui décrit selon une syntaxe précise un ensemble de chaînes de caractères possibles.

Les expressions régulières sont une fonctionnalité que l'on trouve dans beaucoup de langages de programmation modernes.

Une séquence cible (target sequence) est la chaîne de caractères sur laquelle est appliquée l'expression régulière.

Un motif (pattern) est la séquence de caractères représentant ce que l'on cherche à identifier.

Une correspondance (match) est une sous-chaîne de la séquence cible qui correspond au motif.

## Les opérations regex :

- `regex.match (target sequence, pattern)`: retourne true si une séquence correspond à une expression régulière, sinon retourne false.
- `regex.search (target sequence, pattern, match)` : retourne true si une sous-séquence est retournée dans match où le pattern correspond à cette sous-séquence dans le target sequence, false sinon.
- `regex.replace (target sequence, pattern, The Replacement)` : retourne un string du résultat.

## Exemple :

EZ Language	C++	Result
<pre> program p procedure p() begin variable s is string = "subject" variable e is regex = "(sub)(.*)" if e.match(s) then print " String object matched \n"; endif end procedure </pre>	<pre> #include &lt;iostream&gt; #include &lt;string&gt; #include &lt;regex&gt;  int main () {     std::string s ("subject") ;     std::regex e ("(sub)(.*)");      if(std::regex_match (s,e))         std::cout &lt;&lt; " String object matched \n";      return 0; } </pre>	<p>String object matched</p>
<pre> program p procedure p() begin variable s is string = " this subject has a submarine as a subsequence"  variable e is regex = "\\b(sub)([^\n]*)"  variable m is smatch  while e.search (s,m) do foreach() print " \n"; end while  end procedure </pre>	<pre> #include &lt;iostream&gt; #include &lt;string&gt; #include &lt;regex&gt;  int main () {     std::string s ("this subject has a submarine as a subsequence") ;     std::regex e ("\\b(sub)([^\n]*)" ); // matches words beginning by "sub"     std::smatch m;      while (std::regex_search (s,m,e))     {         for (auto x:m)  std::cout &lt;&lt; x &lt;&lt; " ";     }      return 0; } </pre>	<p>subject submarine subsequence</p>

<pre> program p procedure p() begin      variable s is string = "there is a subsequence in the string"      variable e is regex = "\\b(sub)([^\ ]*)"      print e.replace(s, " sub-\$2")  end procedure </pre>	<pre> #include &lt;iostream&gt; #include &lt;string&gt; #include &lt;regex&gt;  int main () {     std::string s ("there is a subsequence in the string") ;      std::regex e ("\\b(sub)([^\ ]*)" ); // matches words beginning by "sub"      std::cout &lt;&lt; std::regex_replace(s,e," sub-\$2");  } </pre>	<p>There is a sub-sequence in the string</p>
--	---	--

### 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.

**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	4 octets valeur de -2147483648 to 2147483647	variable number is integer
Réel	real	8octets +/- 1.7e +/- 308 (~15 digits)	variable number is real

Chaine de caractères	string	Identique au stockage du type composé string du c++	variable name is string
Boolean	boolean		variable 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 :

**constant** Name **is** type = value

#### Exemple :

EZ Language	C++	Result
<pre> program p procedure p() begin // constant declaration: constant c is integer = 5 c = 10 end procedure </pre>	<pre> int main() { // constant declaration: const int c = 5; c = 10; } </pre>	<p>error: assignment of read-only variable 'c'</p>

### Les littéraux numériques :

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

**Exemple:** **variable** number **is** integer = 1\_000\_000

En EZ Language les nombres en hexadécimal (base 16) sont précédés par 0x

**Exemple :** 4b#16 = 75 (décimal)

Les nombres en octal (base 8) sont précédés par un 0

**Exemple :** 113#8 = 75 (décimal)

Les nombre en binaire (base 2) sont précédés par 0b

Exemple : 01001011#2 = 75 (décimal)

EZ Language	C++	Result
<pre> program p  procedure p() begin  variable x is integer = 01001011#2;  variable byte1 is integer = 113#8;  variable byte2 is integer = 4b#16;  variable total is integer = 256 * byte1 + byte2;  print "x = ",x, "\n" print "byte1 = ",byte1, "\n" print "byte2 = ",byte2, "\n" print "total = ",total, "\n"  end procedure </pre>	<pre> #include &lt;iostream&gt;  int main(){  int x = 0b01001011 ; int byte1 = 0113 ; int byte2 = 0x4b; int total = 256 * byte1 + byte2;  std::cout&lt;&lt;"x = "&lt;&lt;x &lt;&lt; std::endl; std::cout&lt;&lt;"byte1 = "&lt;&lt; byte1 &lt;&lt; std::endl; std::cout&lt;&lt; "byte2 = "&lt;&lt; byte2 &lt;&lt; std::endl; std::cout&lt;&lt; "total = "&lt;&lt; total &lt;&lt; std::endl;  return 0; } </pre>	<pre> X = 75 Byte1 = 75 Byte2 = 75 Total = 19275 </pre>