Operators

Logical operators

EZ	C++	Description
and	&&	a and b return true if a is true and b is true
or	||	a or b return true if a is true or b is true or if both are true
xor	^	a or b return true if a is true or b is true, but return false if both are true
not	!	not a return true if a is false, but return false if a is true

Arithmetic operators

The division operator / returns an integer if both operands are integer : the decimal part will be truncated. To obtain a decimal result, at least one of the operand must be cast in real.

The result of a mod operation has the sames sign as the first operand. Therefore, the result of a mod b will have the sign of a.

EZ	C++	Description
-a	-a	unary minus
a + b	a + b	addition
a - b	a - b	substraction
a * b	a * b	multiplication
a / b	a / b	division
a mod b	a % b	modulo
a pow b	pow(a, b)	returns a raised to the power b
abs a	abs(a)	returns the absolute value of a

Assignment operators

EZ	C++	Description
=	=	simple assignment
+=	+=	addition assignment
-=	-=	subtraction assignment
*=	*=	multiplication assignment
/=	/=	division assignment

Comparison operators

The two operands must be of the same type, otherwise an error will be thrown.
In order to compare two operands of different types, a cast operation must be realised.

EZ	C++	Description
a == b	a == b	equal to
a != b	a != b	not equal to
a < b	a < b	less than
a > b	a > b	greater than
a <= b	a <= b	less than or equal to
a >= b	a >= b	greater than or equal to

Increment / decrement operators

EZ	C++	Description
a++	a++	post-increment
++a	++a	pre-increment

a--	a--	post-decrement
--a	--a	pre-decrement

String operators

The following operators are available for strings :

Operator	Description
a = b	assignment operator
a[x]	return the character at the position x of the string a
a . b	return the concatenation of the strings a and b
a .= b	append the string b to the string a
a == b	return true if a is equal to b, false otherwise
a != b	return true if a is not equal to b, false otherwise

Operator precedence

The following table lists the precedence and associativity of EZ operators. Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description
1	++ -- - . [ ] ( ) not	postfix / prefix increment postfix / prefix decrement unary minus member access subscript function call logical not
2	* / mod pow abs	multiplication division remainder power absolute value

3	+ -	addition subtraction
4	< <= > >=	relational operator < relational operator <= relational operator > relational operator >=
5	== !=	relational operator == relational operator !=
6	and or xor	logical and logical inclusive or logical exclusive or
7	= += -= *= /=	direct assignment compound assignment by sum compound assignment by difference compound assignment by product compound assignment by quotient

Operators that have the same precedence are bound to their arguments in the direction of their associativity. For example, the expression a = b = c is parsed as a = (b = c), and not as (a = b) = c because of right-to-left associativity of assignment, but a + b - c is parsed (a + b) - c and not a + (b - c) because of left-to-right associativity of addition and subtraction.

Operators with the same precedence that are not associative can't be used together. For example, the expression 1 < 2 > 1 is forbidden. However, the expression 1 <= 1 == 1 is allowed because the operator == has a lower precedence than the operator <=.

Cast operation

The syntax used to realize a cast is the following :

A is initial_type
B is new_type
B = (new_type) A