Relational And Logical Operators and their Precedence

# Relational Operators

| **Operator function** | **Symbol** |
| --- | --- |
| greater than | > |
| lesser than | < |
| greater than or equal to | >= |
| lesser than or equal to | <= |
| equal to | =:= |
| not-equal to | =/= |
| comparing over different data types | .operator |

# Logical Operators

| **Operator** | **Symbol** |
| --- | --- |
| NOT | ! |
| AND | & |
| OR | | |
| short circuit AND | && |
| short circuit OR | || |

## Rules and Examples

pitch a\_4 = 440;

pitch a\_5 = 880;

int frequencies = 440;

Note Note1 = (440, 2);

Note Note2 = (880, 1);

Measure Measure1 = new Measure[4];

Measure1.add[Note1, 0];

Measure1.add[Note2, 2];

1. The first six symbols, when used as they are can only make comparisons between two entities of the same datatype (e.g. between two pitches or durations).

Correct

a\_4 <= a\_5

(evaluates to true)

Note2.pitch =/= a\_5

(evaluates to false)

Wrong

a\_4 =:= frequencies

(results in an error because they are of different datatypes)

1. The . operator is added as a predecessor to the other relational operators when one wants to compare just the values of the operands regardless of the type.

Correct

a\_4 .=:= frequencies

a\_5 .> frequencies

(evaluates to true despite being different types)

Wrong

a\_4 .(=:=) frequencies

(incorrect usage of the operator, no brackets are required)

**NOTE:** a\_4 .<= a\_5 will NOT result in an error because they are already the same type, the . operator makes no difference

1. Precedence of the operators:

| **Operator** | **Associativity** |
| --- | --- |
| < <= >= > | left to right |
| =:= =/= | left to right |
| short circuit AND | && |
| short circuit OR | || |
| AND | & |
| OR | | |

**NOTE:** Anything that’s not zero or not null evaluates to true. Default value of true: 1.

Correct

Note2.pitch =:= a\_5 .>= frequencies

(evaluated as:  
 Note2.pitch =:= (a\_5 .>= frequencies) which finally evaluates to true)

Note2.pitch >= a\_5 && Note1.duration =/= Note2.duration || frequencies =:= a\_5

(despite comparing frequencies with a\_5 without the . operator, the above statement will evaluate to true due to short circuiting)

Wrong

&& Note1.duration =:= Note2.duration

(results in an error because the logical operators need two operands, this just has one)

Note2.pitch >= a\_5 & Note1.duration =/= Note2.duration | frequencies =:= a\_5

(results in an error since frequencies =:= a\_5 is illegal and the regular OR operator (|) will evaluate both operands)

Loops and Conditional Statemments

# Loops

Beats++ has supports two kinds of loops:

* For loops
* While loops

The syntax for these loops are similar to C-like languages so there isn’t much of a learning curve for new users of Beats++.

## While Loops

Correct:  
while (condition) {

<--statements-->   
}

Examples Are you retardd while loop is simple just use brain. :) Error results in a one-day compiler ban :)And agraj is jaffa hi nandz HELLOOOOFuck you agraj you ignored

For Loops

Correct

for(int i = 0; i < 10; i++)

{

<-- Insert loop body here -->

}

Incorrect

for(int i = 0, i < 10, i++;)

{

<-- Insert loop body here -->

}

Reason: commas used to separate loop statements instead of semicolons

Correct

int length;

for(length = 2; length < 10; length++)

{

<-- Insert loop body here -->

}

Incorrect

int length = 2;

for(int length = 2; length < 10 ; length++);

{

<-- Insert loop body here -->

}

Reason:the integer length has already been declared outside of the declaration for loop and hence cannot be declared twice in the same scope. Additionally, a semicolon cannot be put at the end of the for loop declaration.

Nandz figure out third one pls

Thanks you ze besht

We’ll write about while aa?

While Loops

Correct

int length = 2;

while(length > 0)

{

<-- Insert loop body here -->

length--;

}

Incorrect

while(int length = 2; length > 0; length--)

{

<-- Insert loop body here -->

}

Reason: while loop declaration only takes a single conditional statement.

Conditional

int a = 5;

Correct

int a = 5;

if(a =;= 5)

{

a = 2;

}

else

{

a = 3;

}

Incorrect

int a = 5;

else

{

a = 3;

}

Correct

int a = 5;

if(a =;= 5)

{

a = 2;

}

else

{

a = 3;

}

Incorrect

if(a =;= 5) else (a >=5)

{

a = 3;

}

Reason: The if condition has no body and else immediately follows it. And the condition is after else (instead of else if)

Correct

if(a =;= 5)

{

a = 2;

}

elif(a >=5)

{

a = 3;

}