**Lexical Conventions**

**Comments**

A comment, whether single or mult-line, goes between <- characters, which indicates the start of it and ->, which indicate the end. The comment can be placed anywhere in the program as long as it is between these two characters.

Ex:

1. <- create a chord with

three notes lengthened by 1/8 ->

Chord cr = ([(C, quarter); (Bb, half); (D, whole)], eight)

**Identifiers**

In Cb language, an identifier, is a sequence of letters, digits, and underscores (\_). Note that an identifier must always starts with a lower case letter. There is no limit on how long an identifiers can be.

Below is the list of characters allowed in creating an identifier.

a b c d e f g h i j k l m

n o p q r s t u v w x y z

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z \_

0 1 2 3 4 5 6 7 8 9

**Ex:** nice\_note, nICE\_note, and n\_NOTE2 are acceptable identifiers. However,

\_nicenote, Nice\_note, and 2nicenote are not acceptable identifiers

**Keywords**

They are identifiers used to specify the types of expressions, for retrieving/including methods from an external packages. These keywords listed below are reserved for Cb, which means that they cannot be used as normal identifiers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Int | use | method | sixteen |  |  |
| Note | method | end | eight |  |  |
| Chord | if | return | quarter |  |  |
| Scale | else | compose | half |  |  |
| Stanza | while |  | whole |  |  |
| Score | foreach |  |  |  |  |
| String | in |  |  |  |  |

Ex: Chord cm = [(C, *half*), (G, *half*), (E, *half*];

Chord cmh = []; <- create an empty chord ->

Note x; <- declare an empty note ->

for x in cm { <- x is assigned to eah note of cm chord ->

cmh.put(x^+);

}

**Constants**

**Integer constant.**

Cb has a set on Integer constants that are used to represent basic notes and known durations of notes.

Ex:

**Special character.**

Special characters are represented using escape sequences ac in C language.

Ex: newline is represented by \n

**String Literals**

A string literal is written inside double quotation. To include a double quotation inside a string, use a backlash \.

Ex:

**Operators**

An operator specifies an operation to be performed. Some of operators used in Cb language are shown below:

Cb language takes advantage of existing arithmetic operators to manipulate notes and chords.

|  |  |
| --- | --- |
| ++ | finds all notes within object type and increase the pitch by a half step  **<note/chord/stanza/score>++**  Note that ++ increments by 1 if applied to **Int** |
| -- | finds all notes within object type and decrease the pitch by a half step  **<note/chord/stanza/score>--** |
| ^+ | finds all notes within object type and increase by one octave  **<note/chord/stanza/score>^+** |
| ^- | finds all notes within object type and decrease by one octave  **<note/chord/stanza/score>^-** |
| == | evaluates left hand side and right hand side for strict equality  **<object> == <object>** |
| >, <,  <=, >= | Arithmetic comparison, mostly for comparing note pitches  **same use as ==** |
| % | performs modulo for int arithmetic  **<int> % <int>** |

Ex: Chord cm = [(C, *half*), (G, *half*), (E, *half*];

cm++ <- make the duration of cm, whole ->

cm-- <- make the duration of cm 0, **should return an error because it doesn’t make sense** or we can make it stay at the same duration if it can be reduced by half step->

**Punctuators**

Cb uses punctuators to enhance its semantic. These punctuators are used in declaration and assignment of variables. Below is a list

|  |  |
| --- | --- |
| { } | Used as a pair, curly brackets are used to specify blocks in  methods construction |
| [ ] | Used to create a list of notes, chords and/or stanzas |
| | | Boolean separator Or |
| & | Boolean separator And |
| ! | Boolean separator Not |
| ; | Used to specify the end of a statement |
| = | Assignment symbol |
| . | Used when calling a **method**  **<Score/ chord/stanza/score identifier> = .<method>(<parameters>)**  It is also used for special methods (compose, put, append …) which are called by types  **<chord/stanza/score>.<method>(<parameters>)** |
| ( ) | Used as a parameters holder |

Ex:

1. **Note asharp = (A, *quarter*);**
2. <- do re mi song ->

method Stanzas doremi(Int duration){

Note do = (C, duration);

Note re = (D, duration);

Note mi = (E, duration);

Stanzas s = [do, re, mi, (F, duration), (G, duration), (A, duration), (B, duration), (C, duration)];

return s;

}