Martin Jabůrek xjabur02@stud.fit.vutbr.cz

Jabukód programming **Cheatsheet**

Jabukód language

Jabukód is a structured imperative programming language based on C. Programming in Jabukód is very similar to programming in C.

Principles from C

- Statements end with `;`
  Standalone `;` are not allowed!
- Statement blocks are delimited by `{` and `}`
- Single line comments start with `//`
- Block comments are delimited by `/*` and `*/`
- Identifier syntax
- Local and global variables
- `int` and `float` data types
- Type `enum`:
```
enum COLOR {
    R = 1, G, B = -5
}
```
- `const` and `static` specifiers
- Variable declarations:
```
int x;
```
- Variable definitions:
```
int x = 25;
```
- Function definitions, entry point function `main`:
```
int main() return 0;

int foo(float arg) {
    return arg;
}
```
- Expressions:
```
int x = 14 + 5.2;
float y = (x << 2) + -10;
```
- Function calls:
```
foo(8 * foo(!0));
```
- Structured control flow:
```
if, if-else, while, for
```
  Including falsey / truthy condition evaluation!
- Statements:
```
return, continue, break
```

Adapted features

Features taken from C, but changed slightly.

- `bool` data type
```
bool x = true;
```
  Same as in C, but in the base language!
- Arrays
```
float arr1[3];
int arr2[3] = {3.14, 10, false};
```
  Arrays in Jabukód can only be one-dimensional!!!
- Literal syntax:
  - Integer literals:
    `0, -12, -0xff, 0XF00D`
  - Decimal literals:
    `0.01, -12.5, 5e3, 3.14E-10`
  - Logical literals `true` a `false`
  - String literals delimited by `"` from both sides
  - Escape sequences
    `\n, \t, \\, \"`

  Non-zero decadic literals must not be preceeded by leading zeros!
- Function definitions with only a single statement may not be delimited by `{` and `}`.
- Enum definitions are not termited with a semicolon.
- Chosen operators
```
= ( ) + - * / % == != > < <= >=
&& || ! >> << & | ^ ~
```
  unary `+` is not defined!

Omitted features

- Multi-dimensional and static arrays
- Structs and unions
- Function declarations
  All functions are available globally, regardless of definition location!!!
- Stray block statements
  Block statements may only be used with control flow structures and functions.

New capabilities

Jabukód offers new constructs, not available in C.

- `foreach` loop
```
foreach (int x : array) {
    x = x + 1;
}
```
  Applies its body to on each item in an array! They cannot be mutualy nested.
- `redo` jump statement
  Repeats the current loop iteration without condition evaluation and an update!
- `restart` jump statement
  Repeats a loop again, including initialization!
- `string` data type
  Current implementation only allows use with string literals.
- `write` statement
  Prints a literal or variable of type `string`!
- `exit` statement
```
exit 1;
```
  Terminates a program with given exit code.