downloadable link: R++

§1: R++

Custom language created in 2 months with breaks. Now it is in active development and syntax changes all the time.

§2 Syntaxes

There are specific syntaxes.
{{ }} → we're using for functions, statements and loops
{[ ]} → we're using for classes

```
float a = 0.5421; // ';' → at the end of line
```

§3 Functions

§3.1 Variable types

int ⇒ just a number e.g: 5
float ⇒ decimal e.g: 5.55
string ⇒ text e.g: string a = "ttt" or string a = ttt
bool ⇒ true or false

§3.2 Basic functions

input() ⇒ to write everything you want that corresponds to variable type
print() ⇒ output

§3.3 Basic math functions

sqrt() ⇒ e.g.: sqrt(9) or print(sqrt(9)) ⇒ 3
module() ⇒ e.g: module(-9) or print(module(-9)); ⇒ 9
average() ⇒ e.g: average(2,3) or print(average(2,3)); ⇒ 2.5
round() ⇒ e.g: round(0.3245,2) or print(round(0.3245,2)); ⇒ 0.32
rand{type:range} ⇒ e.g: rand{int/float:30} or print(rand{int/float:30}); ⇒ random number from 0 to 30

§3.4 How to use libraries?

There are two libraries: Math and Date.
include <library_name>; ⇒ e.g: include <Math>;

exclude <library_name>;, is useful if you don't need a library after a moment.
⇒ e.g: exclude <Math>;

§3.5: Math-library functions

log(x) ⇒ log10(100) → 2
loga(x) ⇒ usable as: log2(4) → 2. You may choose the base
sin(), cos() and tan()

§3.6 Date-library functions

now() ⇒ time and date at the moment of usage
yesterday() ⇒ yesterday's date
today() ⇒ today's date

tomorrow() ⇒ tomorrow's date
year() ⇒ current year
month() ⇒ current month number
day() ⇒ current day number
hour() ⇒ current hour
minute() ⇒ current minute
second() ⇒ current second
time() ⇒ current time
day_of_week() ⇒ name of current day
month_name() ⇒ name of current month

§3.7 Statements
if (condition):
{{
    //code…
}}

elif (condition):
{{
    //code…
}}

else:
{{
    //code…
}}

§3.8 Loops
for (i/j/.. in (x → n); i++/i+t; target=y):
{{
    //code…
}}

§3.9 Other
or, and, not ⇒ usable in conditions

§4.0 Hello World!

Hello World! is the first program that is written by beginners in several languages. In R++ that has 3 options:

Option 1:

```
string h = input();
print(h);
```

Input: Hello World!
Output: “Hello World!”

Option 2:

```
print(“Hello World!”);
```

Output: Hello World!

Option 3:

```
hw();
```

Output: Hello World!

Option 4:

```
func main()
{{
    hw();
}}

main();
```

Option 5:

```
func main()
{{
    print(“Hello World!”);
}}

main();
```

§5.0 Complex information

Here's time to understand how the R++ works. We're going to start with creating variables and we will end with creating functions. There will be examples how you do that.

§5.1 Creating variable

R++ supports the following variable types: int, float, bool and string.

```
1   float a = 0.789;
2   int b = 3;
3   bool c = true;
4   bool c2 = false;
5   string name1 = Justin;
6   string name2 = "Justin";
```

Important—
float needs only point (.) in its value.

§5.2 Inputtable variables

You may also use input() to give value on your variable. This even works with bool.

```
1   int i = input("Pick a number: ");
```

§5.3 Built-in math

There are a few built-in math-functions. That are: sqrt(), module(), rand{type:range}, average() and round().

Usage with examples:

---

sqrt():

```
1   int a = 16;
2   float result = sqrt(a);
3   print(result);
```

module(): output → 16

```
1   int a = -16;
2   float result = module(a);
3   print(result);
```

rand{type:range}:

```
1   int range = 4;
2   int num = rand{int:range}; //or float:diap
3   print(num);
```

round(variable,digits):

```
1   float b = 0.98319;
2   print(round(b,2));
```

average():

```
1   int y = 5;
2   int x = 10;
3
4   float result = average(x,y);
5   print(result);
```

These functions can also be used inside of print();

§5.4 Statements and loops

In R++ you can work with if, elif and else. Other operands “and, or, not” are also available.

Statements:

In R++ we use these 3 statements that were inspired by C# and Python. Maybe also with C++.

But there is a difference between R++ and the other programming languages. The “structures” as break and continue are allowed as in loops as well in statements like if, elif and also else.

You can see it on the example on the right side →

Some operators like > and <, have an alternative:
> has >>, and < has <<. They’re equivalent to each other.

```
int x = 5;

if (x > 2):
{{
    print("OK");
}}
elif (x == 2):
{{
    print("Nice");
}}
else:
{{
    break;
}}
```

Loops:

```
int x = 5;

for (i in (-2 --> x); i++; target=3):
{{
    print(i);
}}
```

This is an example of a for-loop that increases (goes from small to high/bigger value).

You can set custom jumps with i++. In order to do that you should change i++ to something like that: i++ → i+2

—Changed loop—

```
int x = 5;

for (i in (-2 --> x); i+2; target=2):
{{
    print(i);
}}
```

New output →

```
-2
0
```

VS

```
-2
-1
0
1
2
```

Old output → → → → → →

—Reverse loop—

```
int x = 5;

for (i in (x <-- -2); i--; target=-1):
{{
    print(i);
}}
```

This is kind of a basic example with a loop that goes from bigger to smaller values.
It's also possible to change jumps in this code.

—Changed loop—

```
int x = 5;

for (i in (x <-- -2); i-2; target=0):
{{
    print(i);
}}
```

Also the same way of changing jumps between values is exactly the same as for the first loop, but make sure to replace + with -.

New output →

```
5
3
1
-1
```

VS

Old output →

```
5
4
3
2
1
0
```

—Role of the parameter “target” —
Target sets the endvalue where code of loop will stop executing if the limit has been reached. If you set target=x then your code will be completely executed. Use the target in the right way and it will be fine.

—Warnings—

1. You can’t mix the direction ‘ ←’ and i++ in the same loop because that won’t work correctly. The same is for ‘ →’ and i--.
2. When you change the direction you need also switch the place of x and a number or other variable as you can see on the examples.

---

§5.5 The libraries
Coming soon ….