
Introduction

Welcome to the **NDP**, in the following pdf file you will learn everything you need to know about **custom language fclang**. A language built just for you intending to provide you a language with a **syntax** that is a mix of C-family languages and modern ones. So that when you finish learning basics, you can choose from anything you want.

NDP team encourages you to write all example scripts in **playground** as you follow and test some of your scripts.

Content

Hello world

As any good programmer, you have to start by writing a Hello World script. The way you do it is simply by using a print statement and Hello World text:

```
print "Hello, World!"
```

Congratulations, after this everything else is easy!

Data types

As any great language fclang provides you with many different data types such as numbers, strings, and logical values.

Booleans

Boolean values represent if something is true or false, 1 or 0. Declaring them with hard-coded value is simple:

```
bool example_boolean = true
```

However, fclang allows you to dynamically declare them with logical expression (more on them later). For example, if we want to know if number a is smaller than number b we can simply write this:

```
int a = 10
int b = 20
bool is_smaller = a < b
```

You can also print it to see if a is smaller than b:

```
print is_smaller // true
```

Integers

Integers or int is standard when it comes to representing whole numbers in programming languages. You have multiple types of them, however currently we only support 32-bit integers. You can do all sort of mathematical expressions with them:

```
int simple_int = 13
int another_int = 12
int simple_operations = simple_int - another_int
int whole_number_division = (simple_int + another_int) / another_int
```

And the output:

```
print simple_int // 13
print another_int // 12
print simple_operations // 1
print whole_number_division // 2
```

Decimals

When you do not want only whole numbers, but you want precision you should decimals or more knowledgeable in other languages as doubles. They are the same however is more natural to call them decimals, simple test:

```
decimal pi = 3.14
decimal r = 2.0
decimal p = r * r * pi
decimal half = 5.0 / 2.0
```

And the output:

```
print p // 12.56
print half // 2.5
```

Strings

They are the best way to represent text, however, they are still under development, so we currently support only basics, however, you can do a lot with print statements:

```
string s = "Hello"
string s1 = ", World!"
```

And output:

```
print s | s1 // Hello, World
print 12 + 2 // 14
```

```
print "Answer: " | 14 + 2 // Answer: 16
```

As you can see you can attach multiple expressions by using | operator.

If statements

Arguably one of the most important statements in ant language, it allows you to execute and skip other parts of code depending on logical expressions, lets see a couple of examples: Let's declare a couple of variables

```
int a = 10
int b = 12
bool ok = true
```

And lets use them

```
if (a > b) {
    print "10 is bigger than 12"
} else {
    print "12 is bigger then 10" // this line will get executed
}

if (ok && true) {
    print "ok" // this line will get executed becuse of logic
} else {
    print "not ok"
}
```

As we can see they are powerful, you can make them nested as much as you want and use as complex logical expressions as possible.

For loops

Another essential part of programming is to know how to use loops, they are used when you need to repeat the same code multiple times while some condition is not met. In fclang we made it so we support for loops where conditions are expressions between two numbers, lets take look at an example:

```
for ( i | 0 | 5 | < | + | 1 )
```

Let's analyze this, first variable i is temporary int available only in the for loop, while it is running. Secondly, we see 0, the initial value of i, and then 5, goal value of i, after 5 we see <, that is the operator with whom we compare i and end value (5), after that we see + and 1, + is the operator with whom we change i and 1 is set, by how much. So let's run it:

```
for ( i | 0 | 5 | < | + | 1 ) {
    print i // 0 1 2 3 4
}
```

As we can see it executes while i is less than 5. You can use any operator variable names and values, so feel free to try it out in **playground**. To summarise, we have a variable name, start value, end value, the operator for comparing them, operator ad

value, known as a step, to increase variable each time for executes. Not so hard right?

Goto statements

One of the main instructions you can give to CPU though assembly is jmp or goto, therefore we felt obligated to include it. Goto allows you to skip parts of code, or simply jump to a certain line, you can break for loops and if statements with them, making them powerful. Example:

```
goto L-skip // skip lines of code till you find goto label L-skip
int a = 10
L-skip // there you are
int a = 12
```

As you might guess if we now print the output will be:

```
print a // 12
```

They are confusingly powerful, sometimes evil when not used properly (one of the reasons why they were abandoned in java and other object-oriented languages), so experiment with them and learn how to use them, after that, you will be able to break if statements and for loops.

Arrays

Arrays are used to store multiple values within themsels, perfect when you do not know how many values have to store. Lets look are this code:

```
IntArray intTest = new IntArray{5} // create int array with 5 places for values
intTest.set(0, 3) // set value at position 0 to 3
intTest.set(1, 2) // set value at position 1 to 2
intTest.set(2, 1) // set value at position 2 to 1
// array can have empty cells
intTest.sort() // sort array
int a = 100 + intTest.get(4) // gets int from array at posstion 4
```

Lets whan are our values:

```
for ( i | 0 | 5 | < | + | 1 ) {
    print intTest.get(i)
}
// Output will be 0 0 1 2 3, becouse default values for int array are 0
// and you did not fill entire array
print a // 103 beacuse arrays count from 0 so last elemnt is 5 - 1 = 4
print intTest.size() // 5 this is usefull when you need to loop trought entire array
```

Arrays are useful, so be sure to experiment with them in **playground**, also you can use decmals, booleans, and sting in arrays.

Matixes

Matrixes are 2D arrays, they are like achess board and just like there each matix cell has its coordinate with whom you can get value at specific postion, some code:

```
IntMatrix testInt = new IntMatrix{2, 2} // new matix with 2 rows and 2 columns
testInt.set(0, 0, 10) // set value ar cordinates 0, 0
testInt.set(1, 1, -10) // set value at cooordinat 0, 0
int rows = testInt.rowSize()
int columns = testInt.colunmSize()
int a = testInt.get(1, 1) + 10
```

Lets whan are our values:

```
print testInt // this syntac wont work, but try to print matrix in playgrounf using for loops
// But if you were to print, this would be the output:
// 10 0
// 0 -10
print rows // 2
print columns // 2
print a // 0
```

Matrixes are useful, so be sure to experiment with them in **playground**, also you can use decimals, booleans, and sting in matrixes.

Libraries

Libraries are an extra set of functions that you can use, they are extremely common and essential in everyday programming, fclang currently supports two.

I/O

I/O or Input / Ouput is simple library that allows you to retrieve user input and to print output (in playtgound by clicking button input you can add you input), example code:

```
int i = getInt <
decimal d = getDecimal <
bool b = getBool <
string s = getString <
```

Now let's say that this is what you have set as input in the dialog:

```
10
3.14
false
Hello
```

This would be output:

```
print i // 10
print d // 3.14
print b // false
print s // Hello
```

There we saw how to use basic operations of I/O library, input, and output.

Math

Every programing language has multiple math libraries, they allow us to preform complex mathematical equasions using squer roots, pow and other functions, code:

```
int a = abs(10 - 20) // absolute value
int b = max(a, 20) // max out of two
int c = min(a, b) // min out of two
int d = pow(2, 3) // 2 on power of 3
int e = sqrt(36) // square root of 36
```

And the output:

```
print a // 10
print b // 20
print c // 10
print d // 8
print e // 6
```

Math library is something you will you often, it is also important to know that allow these functions are supported for decimals as well as for ints.

Report bugs

Fclang is currently still **under development**, so if you find any bugs or issues while testing in playground, or if you find any bugs with NDP app, well free to email us at naucidaprogramiras@gmail.com