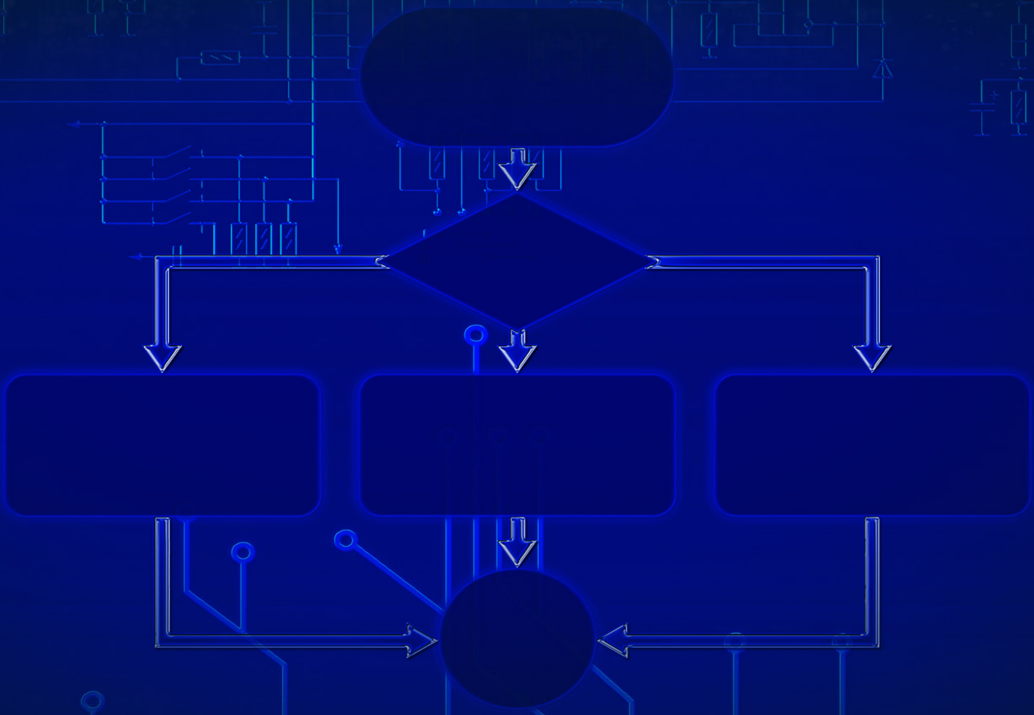
LEARN PROGRAMMING WITH FlowRun

A VISUAL INTRODUCTION INTO THE WORLD OF PROGRAMMING



SAKIB HADŽIAVDIĆ

Learn Programming with FlowRun

A visual introduction into the world of programming

Sakib Hadžiavdić

This book is available at http://leanpub.com/learnprogrammingwithflowrun

This version was published on 2025-02-14

Tweet This Book!

Please help Sakib Hadžiavdić by spreading the word about this book on Twitter!

The suggested hashtag for this book is #flowrun.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#flowrun

To my beloved wife and daughter, who had understanding for my long working hours.

Contents

A Message to You

Preface

This book will teach you the basics of programming using flowcharts. It has lots of solved examples you can try in your browser. Of course, you should try to implement them by yourself first.

Let's begin our journey with some basic terminology. A **computer program** is a *sequence of instructions* that a computer *executes* to perform a task. **Programming** is the act of *writing those instructions*.

You might have heard about **algorithms** too. Is an algorithm the same as a computer program? Not exactly. An algorithm is an **abstractly defined sequence of steps** for performing a task. Note the word "abstract" - it is not an implementation but rather a *description of what needs to be done*. A computer program is essentially an *implementation of an algorithm*. We can run a program but we can not run an algorithm. People might use these two terms interchangeably, and that is fine..

An algorithm can be specified in many ways. Here are a few:

- with text
- with spoken words
- with a mathematical formula
- with a table
- visually with a flowchart

In this book we focus mainly on flowcharts. Flowcharts are *easier to understand* than the textual and other approaches. Our eyes naturally follow the arrows which point us in the right direction. It is easy to see where the decisions happen, because branches of code are visually separated.

Example of an algorithm described with text:

- declare a variable `temperature`
- take input from the user and store it in the `temperature` variable
- if the `temperature` is greater than 20°C then print “It is hot!”
  else if the `temperature` is greater than 5°C then print “It is chilly!”
  else print “It is cold!”

Example of the same algorithm described with a flowchart:

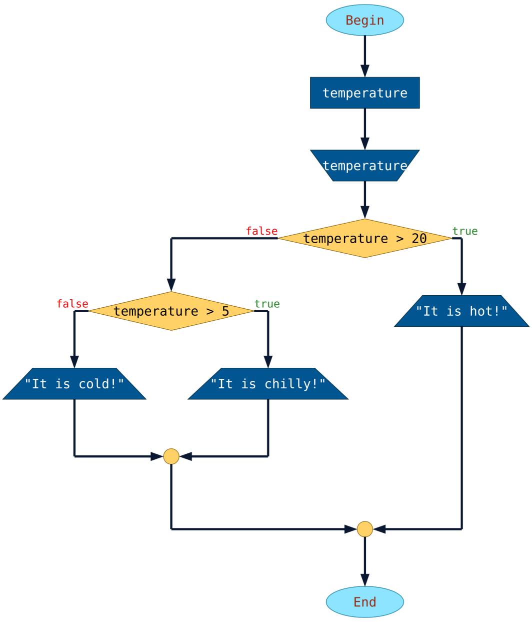


Figure 1. Temperature Algorithm Example

Try it now!

Figure 2. Temperature Algorithm Example QR Code

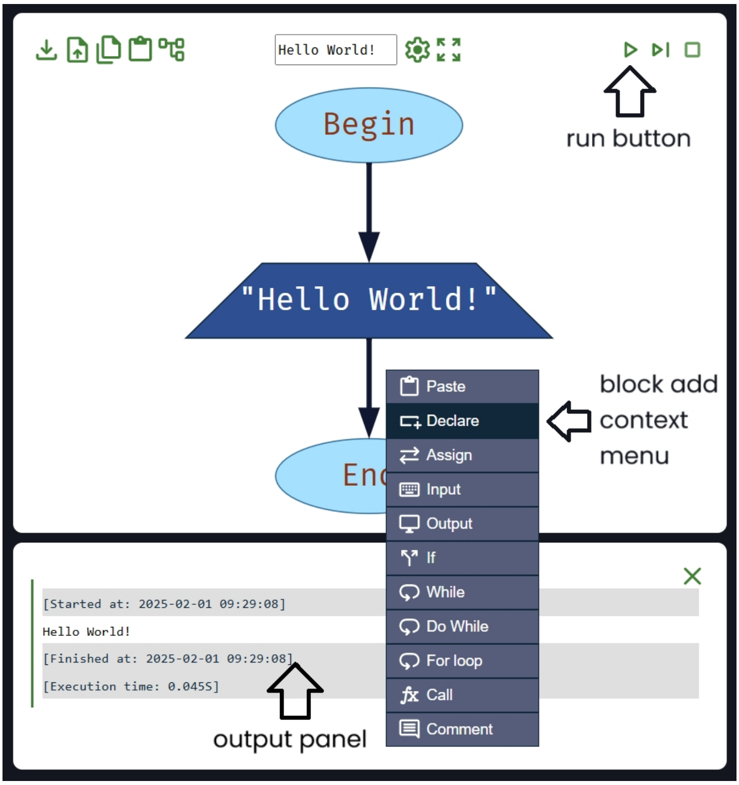
Which one is easier to follow? Maybe in this simple example the textual form is not that bad. But imagine if it was a longer program, or if it contains repeated steps, recursion etc. It quickly gets hard to follow.

FlowRun Editor

FlowRun is a flowchart editor **that can also run the flowchart as a program**.
So we get two birds with one stone, we can:

- visually represent our algorithm
- run it to see if it works (execute the program)

FlowRun can be considered as a *visual* programming language. You can access it at https://flowrun.io/scratchpad (no login required). Here is how it looks like:



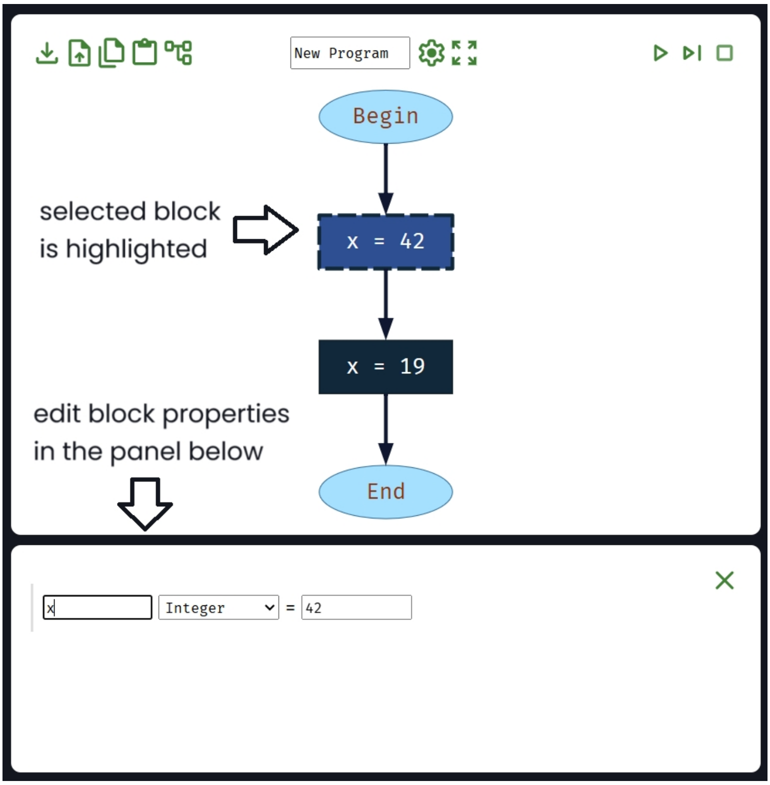
```
[Started at: 2025-02-01 09:29:08]
Hello World!
[Finished at: 2025-02-01 09:29:08]
[Execution time: 0.045S]
```

Figure 3. FlowRun Editor

Basics you need to know:

- click on the Run button (the triangle Play button) to execute the program
- click on any arrow to show the menu for adding a block
- the output panel shows the program output as it runs

This is how you edit a block:



New Program
Begin
selected block is highlighted
x = 42
x = 19
edit block properties in the panel below
End
Integer

x Integer = 42

Figure 4. FlowRun Edit Block

Steps:

- click on a block
- edit its properties in the panel below

The interesting part is how we can combine the blocks and what values to set in their properties.

How Does FlowRun Help You Learn Better?

The **experience of running a program is priceless**.
By looking at a flowchart you can learn a lot, but when you *actually execute it*, that is a whole new level!

> A picture is worth a thousand words.
> ~ Unknown

How much is an interactive program worth then?

When you start using FlowRun, you will make some errors.
But even when you make mistakes - you are learning, what works, what doesn't, and why. The editor will help you by highlighting the failing blocks, validating syntax as you type, printing the errors in the output panel etc.

The syntax is minimal, so you don't have to worry about missing semicolons, incorrect indentation, or compiler flags.
You can **focus on the problem at hand**, the most important part!

FlowRun generates real code. This helps you see how it looks like in popular textual programming languages. The blocks used in FlowRun can be found in almost every popular programming language. This means you are **actually learning the core of programming** as a discipline and **your effort is not wasted**. Whether you plan to become a web developer, a QA tester, or just trying to pass an exam in school, this book will give you a good introduction.

ℹ All FlowRun examples have a "Try it now!" link below. If you are reading this on your computer or smartphone, click it to try the example at the flowrun.io website.

ℹ If you are reading this from a printed book, you can scan the QR code to try the example!

Good luck!

The Basics

In this chapter you will learn the basic blocks needed to make simple programs in FlowRun.

Hello World!

The “Hello World!” program is traditionally the first program you make when learning a new programming language. It prints the “Hello World!” text on the screen.

Steps:

- open https://flowrun.io/scratchpad
- click on the arrow that points from `Being` to `End`
- select `Output`
- click on the new output block
- change its text to `"Hello World!"` (you need the quotes)
- click the Run button

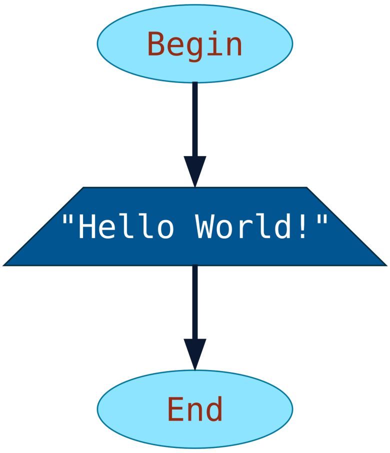


Figure 5. Hello World Example

Try it now!

Figure 6. Hello World Example QR Code

When you run it you will see output like this:

```
Hello World!
```

We need quotes around `Hello World!` because we want plain text. The word `Hello` could mean something special to the program: a variable name, function name etc (you will learn about these later in this book).

You can save this program at flowrun.io if you want, you just need to Log In first. Click on the “Save” button.
You can have your own collection of programs at flowrun.io, share them with your friends and teacher, export as image etc.

Numbers

We have two types of numbers:

- integers (whole numbers) like `5`, `-23`, `0`, `19`
- real numbers (numbers with comma) like `3.14`, `0.004`, `-3.33`

We can do the basic operations on them:

- `+` sum
- `-` subtract
- `*` multiply
- `/` divide
- `%` remainder

Why do we need two types of numbers? Because the mathematical operations on them can give *different results*. We need both, depending on the problem we are trying to solve.

The main difference is when we use division:

- dividing integers gives an integer: `5 / 2 = 2`
- dividing reals gives a real: `5.0 / 2.0 = 2.5`

Use the `Output` block you learned in previous section and print these expressions:

- `5 / 2`
- `5.0 / 2.0`
- `8 * 3.14`

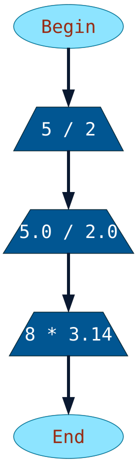


Figure 7. Numbers Example

Try it now!

Figure 8. Numbers Example QR Code

When you run it you will see output like this:

```
2
2.5
25.12
```

The decimal point is written with a dot (`.`), for example `3.14`. The comma (`,`) is used for other purposes, which you will learn later.

Modulo operator

The modulo operator (`%`) returns the *remainder after division*. For example, `11` divided by `4` is `2`, with a remainder of `3`. Or to put it in another way `4 * 2 + 3` equals `11`. We can get this remainder (`3`) with the modulo operator: `11 % 4`.

This is useful for:

- determining if a number is *divisible* by some other number, if it is then the remainder is zero
- determining if a number is even (divisible by `2`) or odd (not divisible by `2`)
- getting the last digit of a number, by dividing it by `10` (e.g. `49 % 10` is `9`)

Variables

We use variables to **store data**. A variable has a **name** and a **type**.

For example, we can declare a variable `count: Integer`. The name of the variable is `count`, and its type is `Integer`. They are separated by a colon (`:`) in the notation used by this book.

Let's say that the `count` variable stores value `10`. Names are important because we can refer to `count` in the program instead of repeating the number `10` everywhere. It is easy to update it from value `10` to some other value.

By telling the program that the variable type is `Integer`, the program can help us more. For example, it can forbid storing `Real`s in that variable (or any other type), it can make sure that some functions can be executed on certain types etc (you will learn them later in the book).

*   *   *

Before we use a variable, we have to **declare it**, tell the program that we will use it. We can immediatelly give it an **initial value**, which is a good programming practice. If we don't give it a value, the program might crash. (Nothing will happen with your computer, don't worry!)

In the following example we declare 3 variables (using `Declare` blocks):

- `amount: Integer`, with initial value of `3`
- `price: Real`, with initial value of `1.5`
- `total: Real`, with initial value of `amount * price`

Then we print the `total` on the screen.

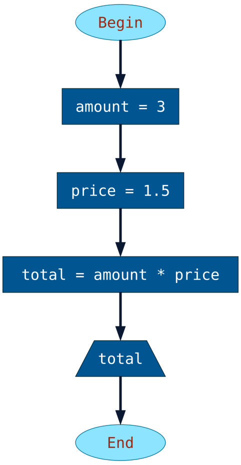


Figure 9. Variables Example

Try it now!

Figure 10. Variables Example QR Code

When you run it you will see output like this:

```
4.5
```

This is a trivial example, and you could certainly do it with your calculator much easier than here. But, you have to start somewhere, and variables are an essential building block!

> You can think of variables as containers for chemicals. First, they can be empty. Second, each chemical needs a special kind of container. You can put water in a plastic container, but can't put sulfuric acid (it would melt the plastic). In the same way you can't put a `Real` value in a variable that has type `Integer`. But sometimes you can combine these values with operators and functions (just like chemical reactions). In the example above we multiplied a `Real` with `Integer` and it worked.

Changing Variables

The meaning of the word "variable" is "to vary", "to change". You can update a variable by **assigning it a new value**.

Let's make an example:

- declare a variable `x: Integer` with value 5
- print x
- add an `Assign` block
- set assign block's variable name to x, and the value to 19
- print x again

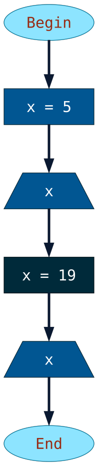


Figure 11. Changing Variables Example

Try it now!

Figure 12. Changing Variables Example QR Code

When you run it you will see output like this:

```
5
19
```

The initial value is 5, so that is printed first. After the assign block, the value changed to 19.

User Input

In previous examples we used fixed (hardcoded) values in programs. But usually we need a value as an **input from the user**. When a value gets read from the user, FlowRun **stores it in a variable**.

Let’s make an example:

- declare a variable `x: Integer`
- add an `Input` block with variable name `x`
- print `x * x`

You have actually implemented a program that calculates square of a number!

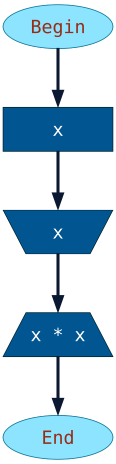


Figure 13. User Input Example

Try it now!

Figure 14. User Input Example QR Code

When you run it, it will ask you to enter a number. If you enter 5 for example, you will see output like this:

```
25
```

Branching with If

We saw that a program executes block after block, from top to bottom. But we usually need some decisions to make (branching) while executing a program ("if this then do x else do y"). We do that with the `If` block.

Let's make a program that will tell us whether a number is positive or negative:

- declare a variable `x: Integer`
- input `x`
- add an `If` block with value `x > 0`
- in the `true` branch output `"Positive"` (line with label `true`)
- in the `false` branch output `"Negative"` (line with label `false`)

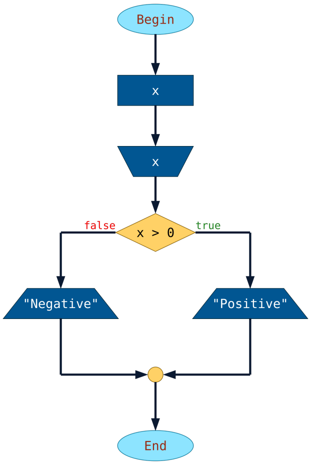


Figure 15. Branching Example

Try it now!

Figure 16. Branching Example QR Code

If the user enters 9, the output will be `"Positive"`.
If the user enters `-55`, output will be `"Negative"`.

> The expression `x > 0` is of type `Boolean`. A `Boolean` variable can be either `true` or `false`.

We use the comparison operators (<, <=, ==, `!=`, >, >=) to compare numbers, which return a `Boolean` value. We use equals and not-equals (== and `!=`) operators to compare other types.

Note that equals is written with double equals sign (==). The equals sign (=) is used for updating a variable, in most programming languages.

Booleans

A value of type `Boolean` can only be `true` or `false`.
We have 3 basic operators for combining them:

- `!`, the negation operator
- &&, the AND operator
- `||`, the OR operator

Negation Operator

The negation operator flips the value.
It is written *before the boolean value/variable*.
Let's say we have a variable `b: Boolean`.
If `b == true`, then `!b == false`.
If `b == false`, then `!b == true`

AND Operator

The AND operator (&&) combines two `Boolean` values with logical AND. We use it when we want to execute some code only if **both values are true**. Here is a table with all possible combinations:

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

OR Operator

The OR operator (`||`) combines two `Boolean` values with logical OR. We use it when we want to execute some code if **any value is true**. Here is a table with all possible combinations:

a	b	a || b
false	false	false
false	true	true
true	false	true
true	true	true

* * *

Let's now make a program that takes two numbers from user. It should tell us if both numbers are positive.

Steps:

- declare two integers: x and y
- input x and y
- add an `If` block with expression `x>0 && y>0`
- add `Output` block in the `true` branch, with value `"Both are positive"`
- add `Output` block in the `false` branch, with value `"One of the numbers is not positive"`

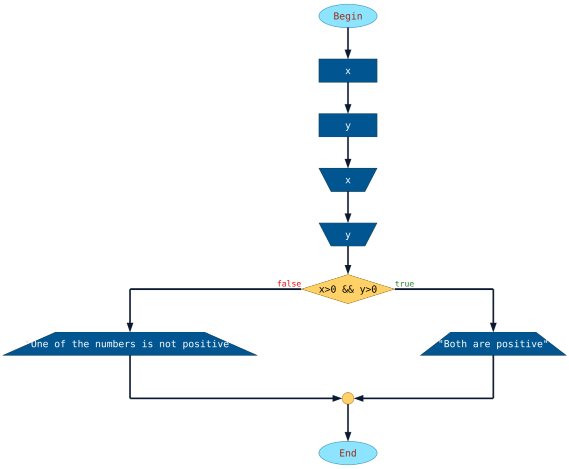


Figure 17. Booleans Example

Try it now!

Figure 18. Booleans Example QR Code

If you are not sure in which order the operations are evaluated, you can use parens to make sure it does what you want. The last example would be same as `(x>0) && (y>0)`. So when you have many subexpressions the parens are very useful.

The && operator binds stronger than ||, so `a || b && c` is evaluated as `a || (b && c)`. You can think of && as multiplication and || as addition.

Strings

We introduced the `String` type in the "Hello World!" section. `String` is essentially just text.

There is one very common operator used on `Strings`, called "concatenation operator". The syntax used in FlowRun is `"abc" + "def"`, we use `+` to combine two strings into one. The result is `String` value `"abcdef"`.

You can also concatenate other types with a string. This is useful for printing values in a nicer way. For example, you can print `"You entered: " + x` (where `x` is a number or something else). That would result in a string that has text "You entered: " and the value of the variable `x` appended to it.

Let's make a program that does exactly that:

- declare `x: Integer`
- input `x`
- output `"You entered: " + x`

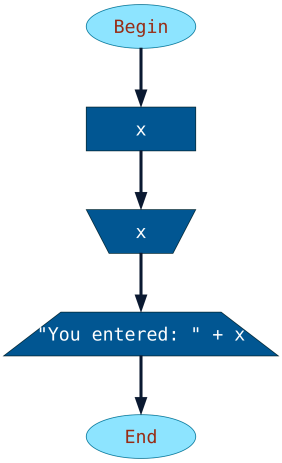


Figure 19. Strings Example

Try it now!

Figure 20. Strings Example QR Code

Exercises

1. input `name: String` from the user and greet them, e.g. “Hello Bob!”
2. calculate circumference of a square, using user-provided `sideLength: Int`
3. calculate area of a square, using user-provided `sideLength: Int`
4. calculate area of a rectangle, using user-provided side lengths `a: Int` and `b: Int`
5. calculate area of a triangle, using user-provided `base: Real` and `height: Real`
6. enter two angles of a triangle and calculate the third one
7. print sum, product, difference, quotient and remainder for two user-provided integers
8. swap two user-provided integers `x` and `y` using a third variable `t`
9. let user enter their year of birth and then print “You are xyz years old”, where `xyz` is their age
10. input `value: Int` and `percentage: Real` (0-100) and calculate that percentage of the `value` variable
11. input `value: Int` and print its absolute value
12. enter three numbers and find the biggest one
13. print whether an integer is even or odd
14. print “yes” if a user-provided integer is *greater or equal* to `0` and less than `100`, else print “no”
15. print “yes” if a user-provided integer is divisible by `3` or `5`, else print “no”
16. calculate the sum of digits for a user-provided integer
17. input integer `n` and print it with its digits reversed, e.g. `123` -> `321`
18. enter `milligrams: Integer` and calculate number of kilograms, grams and milligrams
19. enter `milliseconds: Integer` and calculate number of hours, minutes, seconds, milliseconds
20. enter `bytes: Integer` and calculate number of MB, KB, and bytes B

Loops

When we want to execute some code *multiple times*, we can always copy-paste those blocks. But that is **not very efficient or readable**, so we have **loop blocks** like `While`, `Do While` and `For`.

While Loop

A `While` loop has two parts:

- `condition` - boolean value that is checked *each time* **before** the body is executed
- `body` - sequence of blocks executed when condition is `true`

Let's make a program that prints numbers 1 to 5:

- declare a variable `x: Integer = 1`
- add a `While` block with condition `x <= 5`
- output `x` in While body
- assign `x = x + 1` in While body

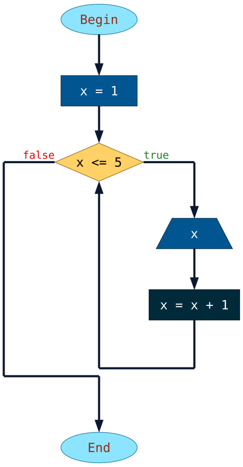


Figure 21. While loop

Try it now!

Figure 22. While loop QR Code

You should see this printed:

```
1
2
3
4
5
```

Notice the Assign block: `x = x + 1`. You can use the previous value of `x` (*before* Assign was executed), while calculating its new value! How cool is that? :)

All loops must have a stopping condition to avoid **infinite loops**. Ensure the condition makes sense, and that you update the condition variables so the loop eventually stops.
If you do get an infinite loop don't worry, click the **Stop button**, fix the program, and try again.

Do While Loop

The `Do While` block is very similar to `While`. The main difference is that `Do While` **checks the condition after it executes the body**. This means that it executes the block *at least once*.

Let's make a program that reads a number from user. If the entered number is not positive, we ask the user to enter a number again.
Steps:

- declare `x: Integer`
- add a `Do While` block with condition `x <= 0`

- input x in the `Do While` body (arrow pointing from circle to `Do While` diamond)

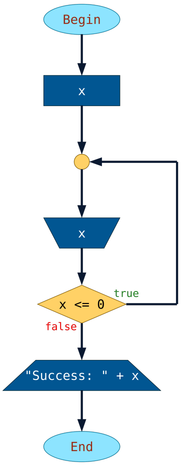


Figure 23. Do While loop

Try it now!

Figure 24. Do While loop QR Code

We read it like this: “Enter number x, as long as it is less than or equal to zero”

Run the program a few times, try various inputs: `1`, `17`, `-55`, `6`. It will exit the `Do While` block only when you enter a positive number.

For Loop

The `For` loop is a bit more complex. It is made of:

- loop **variable name**
- loop variable **initial value**

- loop variable **end value**
- **step** (increment), by how much the loop variable is incremented in each iteration

The `For` loop replaces the following code:

- declare `i: Integer` with an *initial value*
- add `While` with condition `i <= end`
- assign `i = i + step` at the end of `While`

* * *

It is easier to see it in action. Let's make the same program as before, print first 5 numbers:

- declare a `For` loop with variable name `i`
- set `i` start value to `1`
- set `i` end value to `5`
- set step to `1`
- output `i` in the `For` body

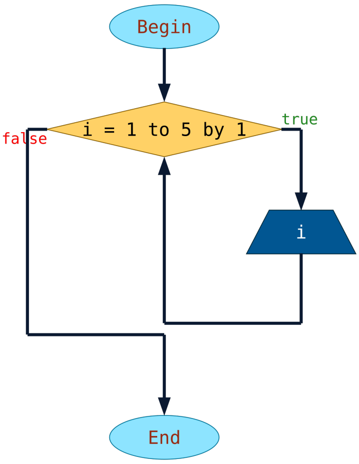


Figure 25. For Loop Example

Try it now!

Figure 26. For Loop Example QR Code

We got the same result as with `While` before, but with a lot less code. Plus, it is more readable than before!

In FlowRun, the `For` variable (usually called `i`) is automatically declared. If you declare the variable `i` before the loop, FlowRun will use it.

Loop in a Loop

When you use a loop *iniside another loop*, that inner loop is called an “inner loop” (or a “nested loop”). The outer loop is of course “outer loop”.

Let’s make a 5x5 multiplication table, using nested `For` loops:

- add a `For` loop with `i` as a loop variable, starting from `1` to `5`
- add a `For` loop with `j` as a loop variable, starting from `1` to `5`
- output `"" + (i*j) + " "` inside the inner loop body, and make sure “New line” is disabled
- output an empty string in outer loop, after the inner loop, make sure “New line” is enabled

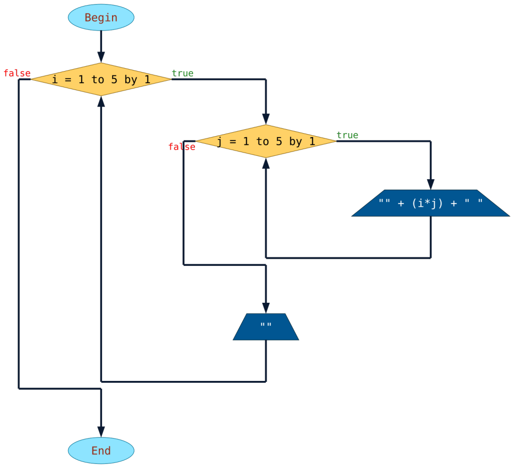


Figure 27. Multiplication Table

Try it now!

Figure 28. Multiplication Table QR Code

When you run it, you should see this in the output:

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
```

Make sure you give **different names to nested loops variables**. In our example we had `i` and `j`, which is fine. If you accidentally put `i` in both loops, FlowRun will use the `i` declared in the outer loop, and you will probably get wrong result (you rarely want to do this).

* * *

Let’s make another program, it will print a “stars triangle” like this one:

```
*
**
***
****
*****
******
```

Steps:

- add a `For` loop with `i` as a loop variable, starting from `0` to `5`
- add a `For` loop with `j` as a loop variable, starting from `0` to `i`
- output `"*"` inside the inner loop body, and make sure “New line” is disabled
- output an empty string in outer loop, after the inner loop, make sure “New line” is enabled

When you run it you will see the pattern as above.

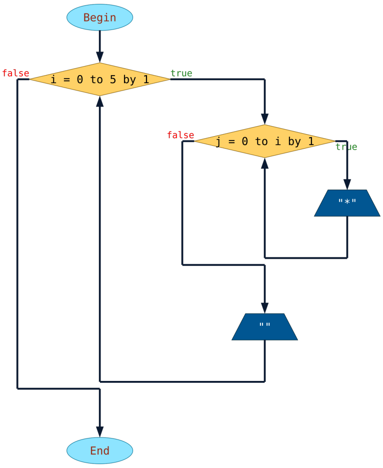


Figure 29. Stars Triangle

Try it now!

Figure 30. Stars Triangle QR Code

Exercises

1. loop through integers `1..n` and print every number that is divisible by both `5` and `3`
2. find sum of first `n` integers
3. enter numbers from user as long as they are `> 0` and print the biggest number they entered
4. enter numbers from user as long as they are `> 0` and print their average
5. calculate a factorial using a while loop (`n! = n * (n-1) * (n-2)... * 1`)
6. calculate a factorial using a for loop (`n! = n * (n-1) * (n-2)... * 1`)
7. print a reversed triangle, first 5 stars, then 4 and so on..
8. print a 10x10 square made of stars (only outer stars of course)
9. print your name in the middle of a 10x10 square of stars

Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Predefined Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Custom Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

The Return of a Function

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Recursive Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Calculating Factorial Recursively

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Function Scope

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Calculating Average Grade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Finding Item in Array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Numbers Frequency

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Sorting Array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Strings are Arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Matrices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Adding Matrices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Check Identity Matrix

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Real World Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Statistics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Tic Tac Toe

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Generating Real Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Use Meaningful Names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Use Temporary Variables

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Use Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Tell User What to Do

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Talk with User

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Handle Errors Gracefully

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Common Problems and Solutions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Variable Not Declared

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Variable Not Initialized

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Wrong Input Format

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Dividing by Zero

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Accessing an Array Index Out of Bounds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Fixing Issues

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

What Next?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Resources

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

FlowRun.io Docs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

FlowRun YouTube Channel

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

FlowRun Live Discussion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

FlowRun Source Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Predefined Functions Reference

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/learnprogrammingwithflowrun.