

# **Лабораторная работа №1**

## **Основные понятия алгоритмов.**

### **1. Теоретическая часть**

Алгоритм — совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи.

Обычно, когда студент впервые обучается программированию, часто для этого используется текстовый язык программирования, и, в зависимости от выбранного языка, опыт такого обучения может быть лёгким или наоборот раздражающе сложным. Зачастую для вывода простого “Hello World!” на экран, требуется написать много строк кода, которые в принципе никак не относятся к поставленной задаче. Это нормально для многих объектно-ориентированных языков программирования (ООЯП), но обучающимся программированию впервые ещё далеко до этих концептов.

Потому данная лабораторная будет сфокусирована на изучение базовых концептов в программировании, не привязываясь к определённому языку программирования – используя блок-схемы.

Мы будем использовать инструмент программирования Flowgorithm.

#### **Установка и использование Flowgorithm:**

Инструмент Flowgorithm можно скачать по следующей ссылке: <http://www.flowgorithm.org/download/index.html>, если на вашем компьютере в лабораторной аудитории это программа не установлена, то можно скачать портативную версию под секцией «Executable Only».

Основное окно программы показано на Рисунке 1.

В верхней части окна присутствуют элементы управления, большинство из которых нас сейчас не интересует. Самые важные элементы это:

- 1й элемент: Открытие сохранённого файла проекта
- 2й элемент: Сохранение вашего текущего проекта в файл
- 3й элемент: Запуск программы

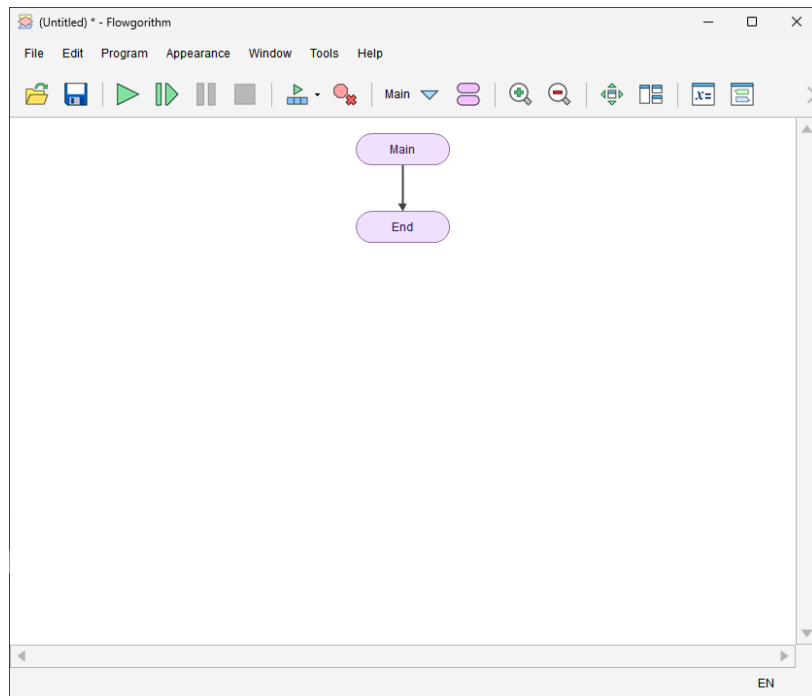


Рисунок 1. Основное окно программы Flowgorithm

Каждый элемент блок схемы синтезируется автоматически, и добавить элемент «в стороне» нельзя, соответственно меню для создания элементов блок-схемы привязан к переходам от блока к блоку. Нажав правую клавишу мыши на стрелку перехода, вы сможете открыть это меню.

Вид этого меню показан на Рисунке 2.

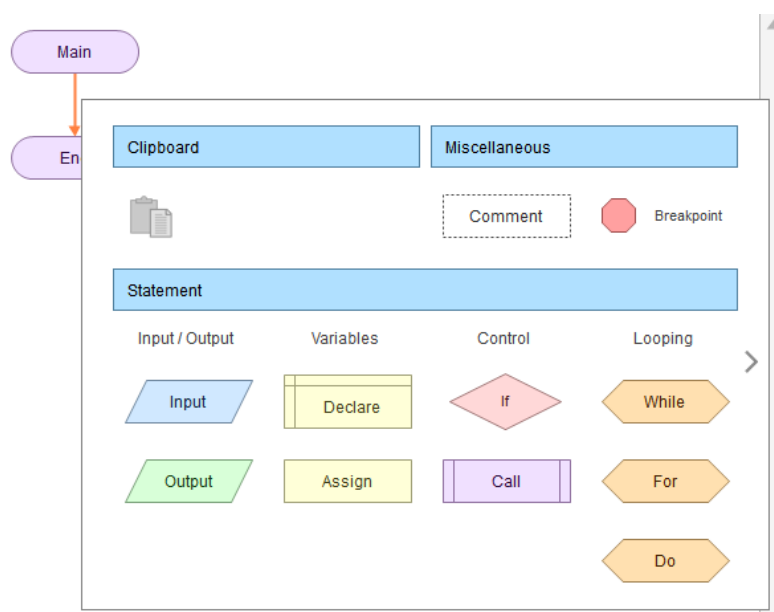


Рисунок 2. Меню элементов блок-схемы программы Flowgorithm

## Переменные.

Переменная – это именованный объект, который может содержать в себе какое-либо значение. В программировании, подразумевается, что переменная это так же область памяти компьютера, адресуемая тем или иным способом, которая может изменять своё значение в ходе выполнения программы.

Переменная – основной блок при построении алгоритмов, программ и т.д. Даже всем известный первый компьютер в мире: «Машина Тьюринга» не может работать без переменных, без ячеек в которые можно записывать то или иное состояние программы. В переменных хранятся значения, на которые тем или иным способом реагирует наша программа.

Важные отличительные черты переменных:

- Название – некий идентификатор, который программа может превратить в определённый адрес в памяти компьютера.
- Тип – также некий идентификатор, который говорит программе, сколько байт данных необходимо прочитать или записать по указанному именем адресу, а также, как эти данные интерпретировать.

Типы переменных в программе Flowgorithm следующие:

- Integer – целое число, может принимать значения, 0, 1, 2, -1 и т.п.
- Real – действительное число (дробное), может принимать значения вида 0.231, 1.234, и т.п.
- String – текстовая строка.
- Boolean – булева переменная.

Без объявления переменной, программа не способна понять, какой адрес памяти компьютера необходимо использовать в тех или иных действиях, потому **очень важно** не забывать объявлять переменные заранее. Однако не перетруждайтесь, объявляя все переменные в самом начале программы создавая так называемые «глобальные переменные».

Переменные должны иметь под собой некоторую локальность, иначе это может приводить к проблемам с безопасностью вашей программы. Если проще – вашу программу будет проще взломать.

В Flowgorithm объявление переменной происходит под блоком «Declare» (Рисунок 2).

С переменными можно проводить две операции:

- Операция присваивания – запись значения в переменную.
- Операция чтения (доступа) – чтение значения из переменной.

Операция присваивания обычно видна явно, знаком «=» или соответствующим блоком (блок «Assign» или блок ввода «Input»), однако про операцию доступа к переменной часто забывают. Любое использование переменной в коде приведёт к тому, что программа обратится к адресу.

Попробуйте провести простейшую операцию над переменной: объявите какую-нибудь новую переменную, присвойте этой переменной значение и выведете его на экран используя блок вывода («Output»)

Добавив блок объявления переменной, на него можно кликнуть 2 раза, для того чтобы открыть окно настроек (Рисунок 3):

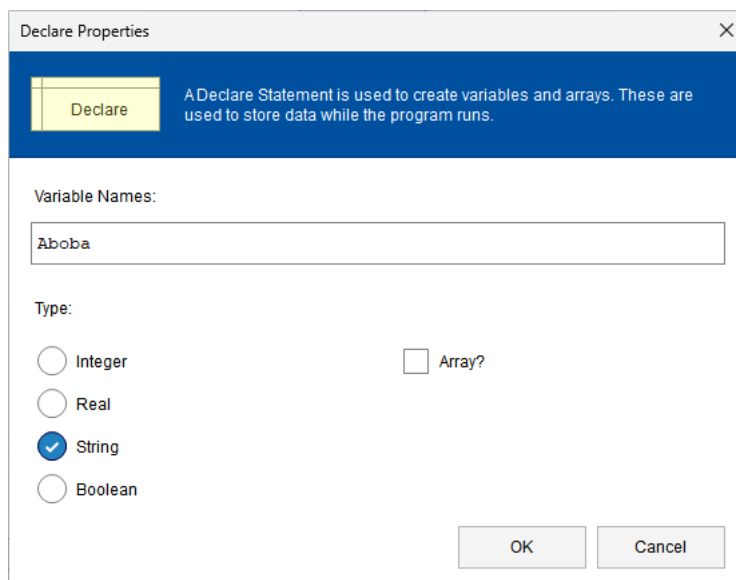


Рисунок 3. Окно настроек объявления переменной(-ых)

Далее, после блока объявления добавьте блок «Assign» и таким же образом откройте его настройки и задайте выражение, которое вы хотите присвоить (Рисунок 4):

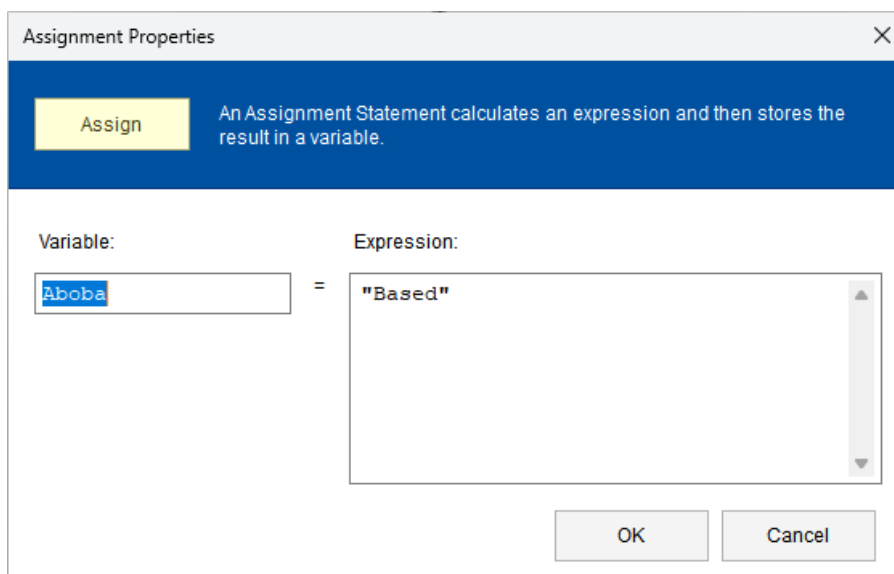


Рисунок 4. Присваивание значение текстовой переменной.

И далее, вывод этого значения на экран. В примере используется более сложное выражение с использованием оператора конкатенации (последовательная запись строк друг за другом). Но в вашем случае это не необходимо. Вы можете просто вывести значение на экран. (Рисунок 5)

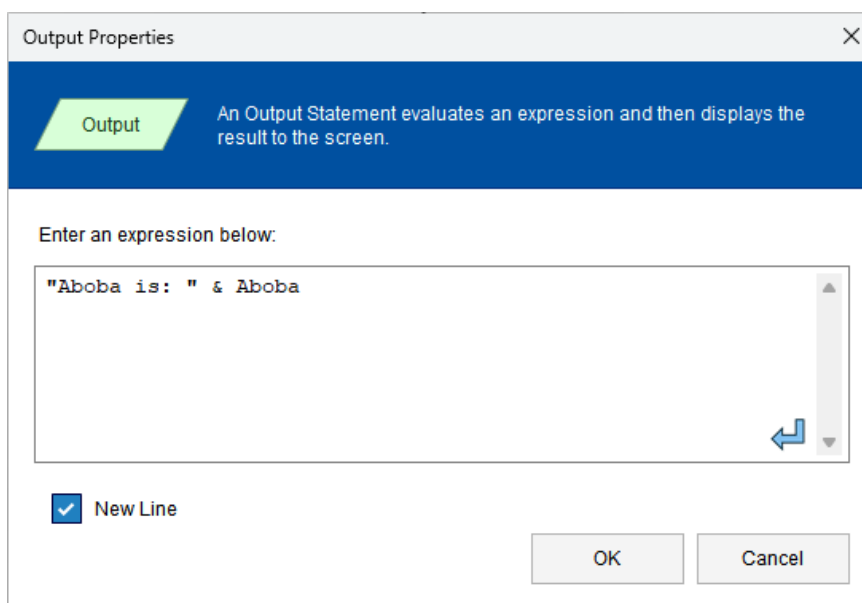


Рисунок 5. Настройки блока вывода данных.

В итоге алгоритм должен выглядеть примерно следующим образом (Рисунок 6):

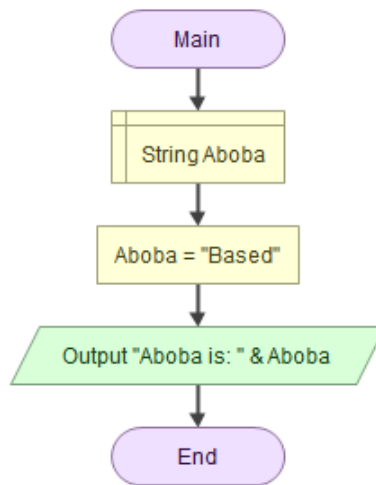


Рисунок 6. Вид алгоритма

Запустим программу, и увидим, что на экран выводится именно то значение, которое мы присвоили переменной (Рисунок 7):

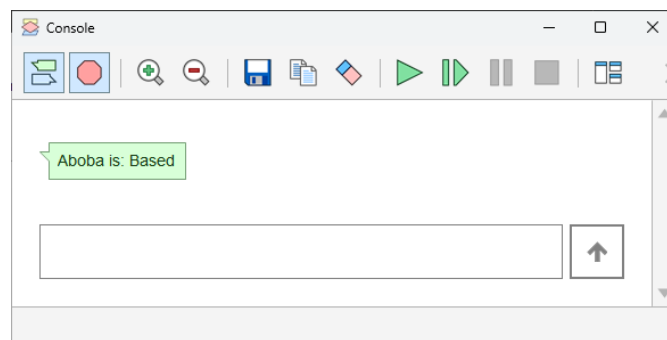


Рисунок 7. Вывод программы

## Ветвления и условия

Второй важнейший концепт программирования – ветвления. В зависимости от значений переменных, ваша программа может выполнить либо один набор действий, либо другой, и этот процесс контролируют ветвления.

Основная форма реализации ветвления – условный оператор («if») и оператор многозначного выбора («case» и «switch»). Условный оператор на свой вход принимает булеву переменную или как говорят чаще «логическое выражение».

Булева переменная, это переменная, которая может хранить в себе лишь два значения «Да» и «Нет». Важно не путать булеву переменную и бит, хоть на самом деле они, в сущности, одинаковы (они представляют бинарное состояние): бит – действительно бит, и внутри памяти компьютера занимает «1» бит памяти. Булева переменная в 99% реализаций

занимает как минимум 1 байт, и мы просто говорим, что «0» это нет, а всё что угодно остальное это «Да».

Если говорить простым языком, то условный оператор, в зависимости от того, что вы подадите ему на вход («Да» или «Нет») выполнит либо один кусочек кода, либо другой. Это всё что делает данный оператор. Самое главное обычно происходит как раз в условиях, сравнение разных данных и переход к выполнению того или иного кода программы называют **«логикой программы»**.

Логические операторы делятся на два вида:

- Операторы сравнения
- Булевы операторы.

Первые сравнивают значения, которые не являются булевыми, например числа, строки и т.д., вторые выполняют операции преобразования над булевыми значениями, по тем самым «табличкам истинности» которые изучались в школе.

Важно заметить, что операторы сравнения всегда бинарные, так как для сравнения необходимо минимум 2 значения, а для сравнения большего количество значений, бинарные операторы можно объединять через булевы. Однако булевы операторы бывают так же унарные (1 операнд), бинарные (2 операнда), тернарные (3 операнда) и т.д.

Операции сравнения представляются следующими операторами:

- Оператор равенства «==»
- Оператор неравенства «!=»
- Оператор больше «>» или больше или равно «>=»
- Оператор меньше «<» или меньше или равно «<=»

Булевы операции все на свете не описать, однако базовые рассмотрим

- Унарный оператор инверсии (НЕ) «!»
- Бинарный оператор «логическое И» «&&»
- Бинарный оператор «логическое ИЛИ» «||»

Из данных 3х булевых операторов можно построить любое условное выражение на свете, то есть этот набор операторов, так называемое «полон по Тьюрингу»

Добавив в нашу программу некоторые изменения, поставим вместо блока вывода, блок условного оператора «If», внутри него напишем логическое выражение, сравнив нашу переменную, с каким-либо значением (Рисунок 8):

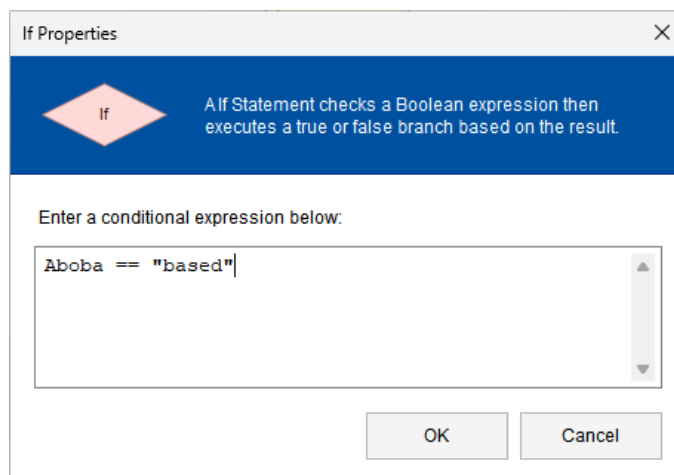


Рисунок 8. Установка условного выражение в блоке условного оператора

Таким образом мы превратили нашу переменную в булево значение, с которым блок может работать. Далее по бокам нашего оператора появились «ветки» с подписью «True» и «False» программа выполнит действия справа, если условие верно (булево значение равно «да»), и действия слева, если условие не верно (булево значение равно «нет»).

Добавим какой-нибудь различный вывод в этих ветках, и тогда наш алгоритм будет выглядеть следующим образом (Рисунок 9):



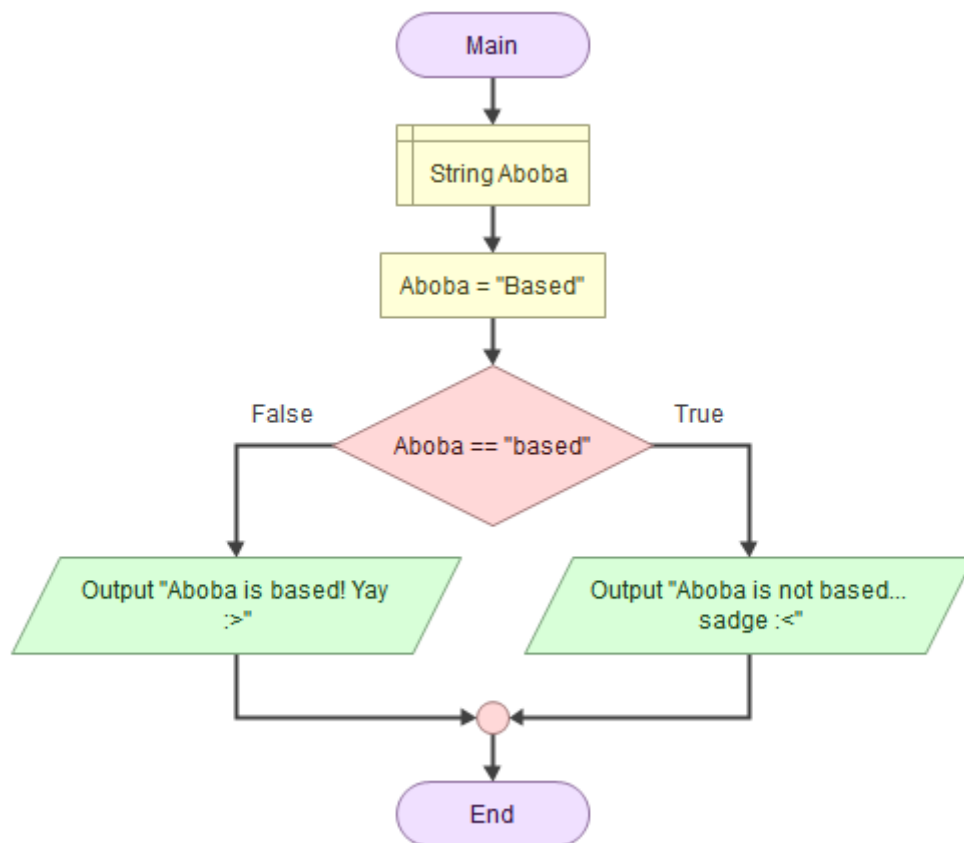


Рисунок 9. Алгоритм с использованием ветвлений и логического оператора.

Попробуйте изменить значение в блоке присваивания, и вы увидите, что на экран консоли выводятся разные значения. Таким образом, контролируя значение переменной, мы можем контролировать поведение программы.

## Циклы

Циклы – это блоки программы, которые выполняют один определённый кусочек программы до тех пор, пока указанное условие выполняется.

Можно рассматривать циклы, как условные операторы, в котором один кусочек кода – это тело цикла, код, который мы хотим выполнять много раз, а второй кусочек кода – возвращение выполнения кода в начало. Так мы получим логику программы, которая выполняет ветку с условием «Да» до тех пор, пока условие не станет «Нет».

Циклы делятся на три вида:

- Безусловный цикл (бесконечный цикл) – он выполняется вечно, и программа никогда не завершает своё выполнение.
- Цикл с предусловием – перед тем как выполнять тело кода, проверяется условие. Такой цикл может не выполниться ни разу.
- Цикл с постусловием – сначала выполняется тело кода, и только потом проверяется условие, необходимо ли завершить цикл. Такой цикл выполнится как минимум 1 раз.

В программе Flowgorithm цикл с предусловием представлен блоком «While», а цикл с постусловием представлен блоком «Do». Помимо этих блоков существует блок цикла «For».

Цикл «For» – это так называемый «цикл с счётчиком», цикл, который выполняется заданное количество раз, и предоставляет доступ к переменной итерации – индексу. В действительности, это «синтаксический сахар» созданный для упрощения итерации по массивам и спискам, цикл «For» – это цикл с предусловием, который достаточно «удобно» предоставляет возможность дополнительного действия над переменной итерации.

Сделаем в нашей программе очередную модификацию: будем вводить и проверять наше значение до тех пор, пока оно не станет таким, какое мы хотим. Для этого удобно применить цикл с постусловием, так как нам как минимум 1 раз нужно присвоить значение нашей переменной.

Вместо условного оператора, поставим блок цикла «Do» с обратным условием тому, чтобы был в условном операторе (Рисунок 10):

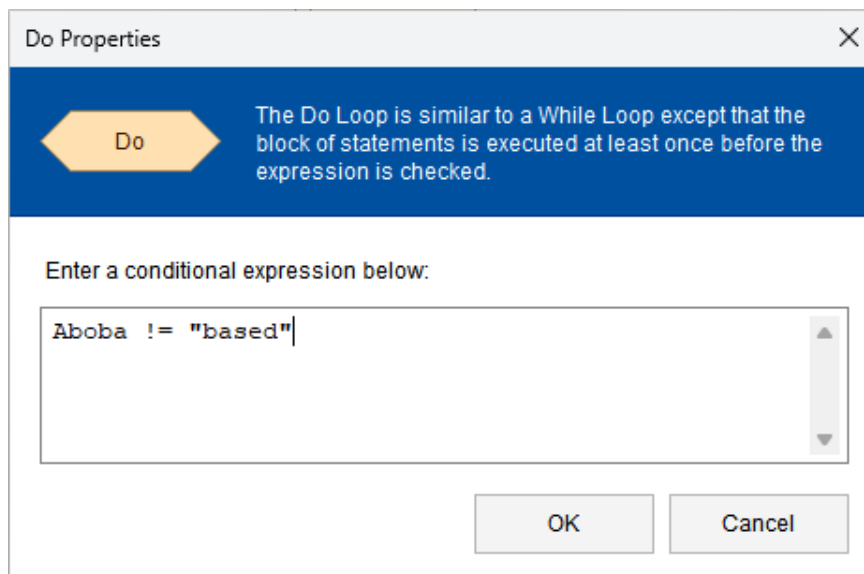


Рисунок 10. Настройка блока оператора цикла «Do»

Также, удалим блок присваивания значение, иначе мы можем случайно создать бесконечный цикл, и программа может «зависнуть». И добавим в тело цикла (ветка сбоку) блок ввода («Input»), установим, что значение ввести надо в нашу переменную (Рисунок 11).

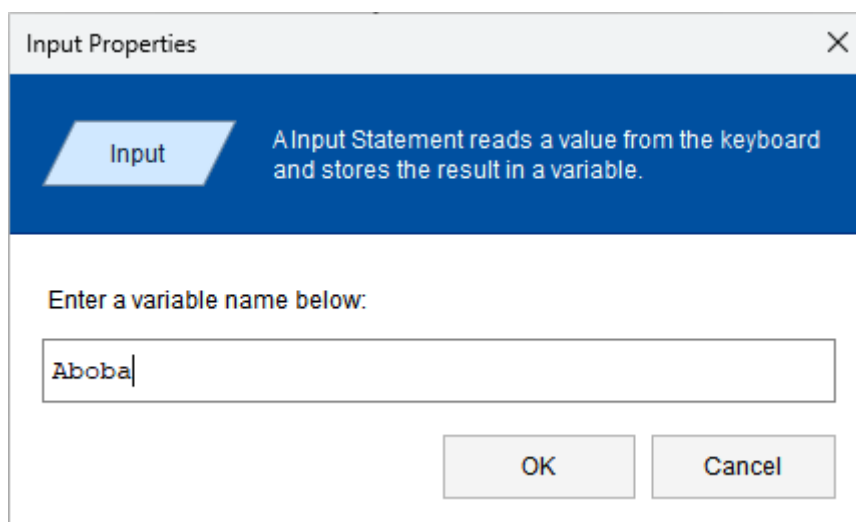


Рисунок 11. Настройка блока ввода.

И добавим какой-нибудь вывод после выполнения цикла, тогда общий алгоритм будет выглядеть примерно таким образом (Рисунок 12).

При работе с циклами необходимо держать в голове с какими данными, и в каком формате вы работаете. Цикл будет работать до тех пор, пока условие **верно**, и неправильно оформленное логическое выражение цикла может привести к образованию бесконечного цикла, что почти всегда нежелательное поведение программы.

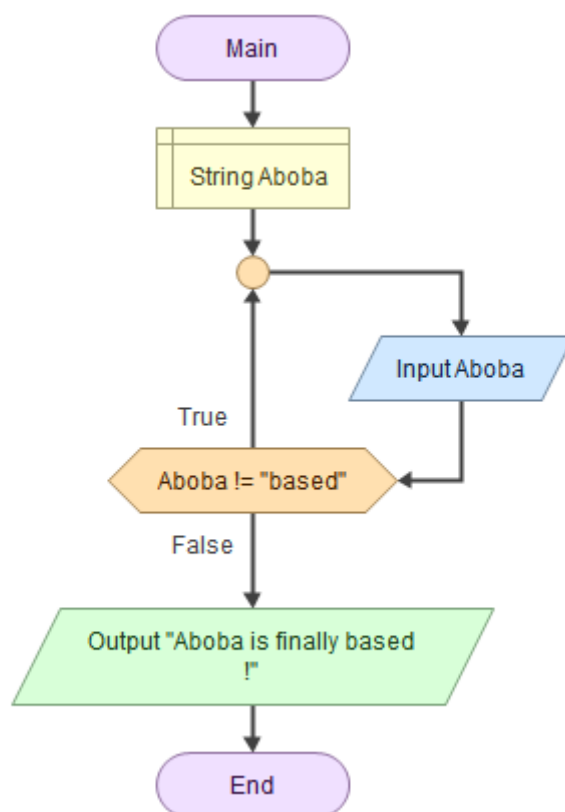


Рисунок 12. Вид алгоритма ввода до необходимого значения.

### Сложность алгоритмов.

В информатике, при исследовании алгоритмов, применяется такая терминология как «вычислительная сложность» – это мера того, как растёт время выполнения программы (количество шагов / итераций) в зависимости от входных условий, чаще всего, от размера входных данных –  $n$ .

Описывают сложность при помощи нотации «Большое  $O$ », то есть – асимптотическое ограничение. Функция  $f(n)$  принадлежит  $O(g(n))$  если она асимптотически ограничена сверху функцией  $g(n)$ . На пальцах –  $O(g(n))$  это функция бесконечно большая относительно  $g$ , но растёт она не быстрее чем  $g$ .

В таком случае мы можем говорить например, что если наш алгоритм принимает в себя массив размером  $10(n)$  чисел, и он выполняет  $10(n)$  итераций, и в каждой итерации ему нужно выполнить  $10 + 2(n + 2)$  шагов, а так же, дополнительно, 300 шагов в начале для подготовки, общая сложность алгоритма будет такой:  $f(n) = n * (n+2) + 300 = n^2 + 2n + 300$  шагов. Однако рост такой функции – квадратичный, и потому мы скажем, что сложность алгоритма  $O(n^2)$ . Если время выполнения алгоритма не зависит от входных данных, говорят, что сложность алгоритма равна  $O(1)$ .

## 2. Практическая часть

1. Установить программу Flowgorithm
2. Реализовать программы, показывающие работу переменных, логических операторов, циклов.
3. Выполнить задания по вариантам.

Отчёт по выполнению заданий должен содержать в себе:

1. Блок-схему алгоритма задания
2. Пояснения использованных блоков в блок-схеме
3. Расчёт количества итераций в зависимости от входных данных
4. Расчёт сложности полученного алгоритма
5. Ответы на контрольные вопросы.

№ Варианта = № Студента в списке mod 4 (остаток от деления) + 1.  
то есть: студент №1 = вариант №1, С№2 = В№2, С№3 = В№3,  
С№4 = В№1, С№5 = В№2, и т.д....

Варианты заданий представлены в таблице 1.

Таблица 1: Варианты заданий

Вариант	Задание	Описание
1	1	Реализовать алгоритм расчёта объёма цилиндра с основанием $r$ и высотой $h$ . [ $V = \pi * r^2 * h$ ]
	2	Реализовать алгоритм расчёта $N$ -го числа последовательности Фибоначчи [ $X(n) = X(n - 1) + X(n - 2)$ , $X(1) = 1$ , $X(0) = 1$ ]
2	1	Реализовать алгоритм расчёта скалярного произведения двух двумерных векторов [ $a \cdot b = x_a * x_b + y_a * y_b$ ]
	2	Реализовать алгоритм расчёта $N$ -го факториала ( $N!$ ) [ $N! = N * (N-1) * (N-2) * \dots * 1$ ]
3	1	Реализовать алгоритм расчёта площади сферы [ $A = 4 * \pi * r^2$ ]
	2	Реализовать алгоритм расчёта суммы первых $N$ натуральных целых чисел. [ $S = 0 + 1 + 2 + 3 + \dots + N$ ]

## Контрольные вопросы

1. Что такое переменная?
2. Что такое тип переменной?
3. Что такое булева переменная и условное выражение?
4. Как работает условный оператор?
5. На какие типы различают циклы
6. Чем отличается цикл с предусловием от цикла с постусловием
7. Что такое вычислительная сложность алгоритма
8. \*Как вычислить сложность алгоритма, сложнее чем полином?

## Рекомендуемая литература

1. <http://www.flowgorithm.org/documentation/index.html>
2. <http://www.flowgorithm.org/documentation/language/operators.html>
3. <http://www.flowgorithm.org/documentation/language/constants.html>
4. [https://ru.wikipedia.org/wiki/Цикл\\_\(программирование\)](https://ru.wikipedia.org/wiki/Цикл_(программирование))
5. [https://ru.wikipedia.org/wiki/Ветвление\\_\(программирование\)](https://ru.wikipedia.org/wiki/Ветвление_(программирование))
6. Шведов И. А. Компактный курс математического анализа. Часть 1. Функции одной переменной. — Новосибирск, 2003. — С. 43.