

Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Физико-технологический институт

Кафедра «Технической физики»

Оценка

Руководитель учебной

практики

к.т.н., ст. преподаватель Каримова О.Х.

**ОСНОВЫ АЛГОРИТМИТИЗАЦИИ**

ОТЧЕТ

по учебной практике

Студент Козыренко Вячеслав Николаевич

Специальность (направление подготовки)

09.03.02 Информационные системы и технологии»

Группа Фт-140008

Екатеринбург

2025

**СОДЕРЖАНИЕ**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc172307919)

[1 Алгоритмы с условным переходом 4](#_Toc172307920)

[1.1 Задание №3. Вариант №9 4](#_Toc172307921)

[2 Счетные циклы (циклы с заданным числом повторений) 6](#_Toc172307922)

[2.1 Задание №6. Вариант №9 6](#_Toc172307923)

[3 Условные циклы (с неизвестным числом повторений 7](#_Toc172307924)

[3.1 Задание №12. Вариант №9 7](#_Toc172307925)

[4 Одномерные массивы 9](#_Toc172307926)

[4.1 Задание №14. Вариант №9 9](#_Toc172307927)

[4.2 Задание №20. Вариант №9 11](#_Toc172307928)

[5 Двумерные массивы 13](#_Toc172307929)

[5.1 Задание №22. Вариант №9 13](#_Toc172307930)

[ЗАКЛЮЧЕНИЕ 15](#_Toc172307931)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 16](#_Toc172307932)

# ВВЕДЕНИЕ

Настоящий отчет подготовлен по результатам прохождения практики по дисциплине «Основы алгоритмизации». Главной целью практической работы стало углубленное изучение принципов алгоритмизации через совмещение теоретических знаний с их практическим применением. В процессе обучения особое внимание уделялось освоению методов разработки алгоритмов, их программной реализации на языке Python, а также визуализации в форме блок-схем.

В ходе выполнения практических заданий были детально изучены фундаментальные основы алгоритмизации, включая базовые алгоритмические структуры и способы их графического представления. Работа строилась вокруг решения задач возрастающей сложности, что потребовало самостоятельной разработки алгоритмов, их последующей реализации в коде и оформления в виде блок-схем.

В отчете подробно рассматриваются выполненные задания, представлены разработанные алгоритмы и соответствующие блок-схемы, проведен анализ полученных результатов. Также сформулированы выводы, отражающие приобретенные компетенции в области алгоритмизации и программирования.

Работа выполнялась студентом Козыренко Вячеславом Николаевичем (группа Фт-140008) согласно индивидуальному варианту заданий. На основании порядкового номера в списке группы (9) в соответствии с методическими указаниями были решены задачи №3, №6, №12, №14, №20 и №22. Все поставленные перед практикантом задания успешно выполнены, а их решения представлены в данном отчете.

# 1 Алгоритмы **с условным переходом**

# 1.1 Задание №3. Вариант №9

Даны вещественные числа A, B. Найти большее из них и сумму.

Алгоритм решения получился следующим: компьютер получает два числа, введённых пользователем. После чего проверяет какое из чисел больше. Затем переменной А присваивается большее значение, а переменной B сумма введённых чисел. После выполненных операций программа выводит большее значение и сумму, что и требовалось в задании.

Алгоритм продемонстрирован на рисунке 1.

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – Блок-схема к заданию №3

После разработки алгоритма была написана программа на языке программирования Python. В коде программы использовался условный оператор, целью которого стало определение большего числа. Реализация данного алгоритма представлена ниже.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

В ходе тестирования данного кода было выяснено, что программа работает корректно, что означает правильность составленного алгоритма.

# 2 Счетные циклы (циклы с заданным числом повторений)

# 2.1 Задание №6. Вариант №9

Алгоритм вычисления U*n*​ начинается с ввода натурального числа n, которое должно быть не меньше 4. Затем инициализируются начальные значения последовательности: v1=0*v*1​=0, v2=0*v*2​=0, v3=1,5*v*3​=1,5. Для каждого i от 4 до n последовательно вычисляется очередной член U*n* по рекуррентной формуле, использующей три предыдущих значения (vi−1*vi*−1​, vi−2*vi*−2​, vi−3*vi*−3​) и зависящей от текущего индекса i. Если n равно 4, результат выводится сразу, иначе цикл продолжается до достижения i=n. В случае некорректного ввода (например, n<4*n* <4) программа выводит сообщение об ошибке, а после завершения вычислений выводится значение U*n*. Для точных расчётов рекомендуется использовать дроби или высокую точность вычислений, особенно при больших значениях n.

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – Блок-схема к заданию №6

Чтобы убедиться в корректности разработанного алгоритма, его реализовали в виде программы на Python. Ниже приведён соответствующий программный код.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Проведённые тесты подтвердили корректную работу программы, что свидетельствует о верности реализованного алгоритма.

1. Условные циклы (с неизвестным числом повторений**)**

# 3.1 Задание №12. Вариант №9

Дана последовательность отрицательных целых чисел, оканчивающаяся нулём (ноль — признак окончания ввода последовательности, количество элементов изначально неизвестно). Требуется определить сумму элементов этой последовательности. Программа начинает с инициализации переменной для накопления суммы. Пользователь вводит целые числа последовательно, пока не введет ноль, что служит сигналом окончания ввода. Каждое введенное число проверяется на отрицательность — если условие выполняется, число добавляется к сумме. Процесс продолжается до тех пор, пока не будет получен ноль. В завершение программа выводит общую сумму всех введенных отрицательных чисел. Блок-схема задания №12 представлена на рисунке 3.

Изображение выглядит как диаграмма, зарисовка, План, Технический чертеж

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – Блок-схема к заданию №12

Для проверки правильности составленного алгоритма была написана программа на языке программирования Python. Код программы представлен ниже.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Тестирование программы показало, что код функционирует без ошибок, доказывая правильность разработанного алгоритма.

# 4 Одномерные массивы

# 4.1 Задание №14. Вариант №9

Дан массив целых чисел. Требуется вывести на экран сначала его чётные элементы, а затем нечётные.

Программа начинает с запроса у пользователя размера массива, который сохраняется в переменную k. Затем создается пустой массив arr для хранения вводимых значений. В цикле от 1 до k программа последовательно запрашивает у пользователя каждый элемент массива, преобразует его в целое число и добавляет в массив arr. После заполнения основного массива создаются два дополнительных пустых списка: chet для четных чисел и nechet для нечетных. Далее программа проходит по каждому элементу массива arr в цикле от 0 до k. Для каждого элемента выполняется проверка на четность с помощью операции взятия остатка от деления на 2. Если остаток равен нулю, элемент добавляется в список chet, в противном случае - в список nechet. После завершения цикла проверки всех элементов программа выводит на экран сначала список четных чисел chet, затем список нечетных чисел nechet. Алгоритм позволяет четко разделить элементы исходного массива на две категории по признаку четности. Результат работы программы дает наглядное представление о распределении чисел в исходном массиве. Для корректной работы важно, чтобы все вводимые значения были целыми числами, иначе программа завершится с ошибкой. В текущей реализации отсутствует обработка возможных ошибок ввода, что можно считать ограничением данного алгоритма.

Изображение выглядит как текст, диаграмма, зарисовка, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 – Блок-схема задания №14

Для проверки работоспособности алгоритма разработан код на Python, который демонстрируется ниже.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Результаты тестирования подтвердили, что программа работает исправно, что указывает на грамотное построение алгоритма.

# 4.2 Задание №20. Вариант №9

Алгоритм (а) без массива:

Программа начинает с инициализации переменной для хранения максимальной плотности. Затем в цикле последовательно считываются масса и объем каждого из 20 предметов, после чего вычисляется плотность каждого предмета. Если вычисленная плотность оказывается больше текущего максимального значения, оно обновляется. По завершении цикла программа выводит найденное значение максимальной плотности.

Алгоритм (б) с массивом:

Программа создает пустой массив для хранения значений плотности. В цикле считываются масса и объем каждого из 20 предметов, вычисляется их плотность, которая добавляется в массив. После обработки всех предметов находится максимальное значение в массиве плотностей. Итоговое значение максимальной плотности выводится на экран.

Сравнение подходов:

Первый алгоритм экономит память, так как не хранит все промежуточные значения, а второй позволяет при необходимости сохранить все данные для дальнейшего анализа. Оба метода гарантируют корректное нахождение максимальной плотности среди 20 предметов.

Особенности реализации:

В варианте (а) используется пошаговое сравнение, а в варианте (б) применяется встроенная функция поиска максимума, что делает код более компактным. Оба варианта предусматривают ввод данных в одинаковом формате и вывод результата с одинаковой точностью.

Алгоритм представлен на рисунке 5.

Изображение выглядит как диаграмма, текст, План, зарисовка

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 – Блок-схема к заданию №20

Для проверки работоспособности алгоритма «a» разработан код на Python, который демонстрируется ниже.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Проведённые тесты подтвердили корректную работу программы, что свидетельствует о верности реализованного алгоритма.

Ниже представлен код программы, написанной по составленному алгоритму «б».

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

После проверки работы данной программы было выяснено, что результаты достоверны, следовательно, алгоритм составлен верно.

# 5 Двумерные массивы

# 5.1 Задание №22. Вариант №9

Дан двумерный массив. Определите, в какой строке массива сумма элементов больше: во второй или в третьей.

Пользователь вводит количество строк и столбцов для создания двумерного массива. Затем построчно заполняются элементы матрицы. Если количество строк меньше трёх, программа выводит сообщение об ошибке и завершит работу.

Для корректного массива вычисляются суммы элементов во второй и третьей строках. Полученные суммы сравниваются между собой. Если сумма второй строки больше, программа выводит соответствующее сообщение, в противном случае указывается, что больше сумма третьей строки. Результат работы программы позволяет определить, в какой из двух строк (второй или третьей) сумма элементов массива больше. Алгоритм представлен на рисунке 6.Изображение выглядит как текст, диаграмма, зарисовка, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – Блок-схема к заданию №22

Ниже представлен код программы на языке программирования python, составленный по заданному алгоритму.Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Алгоритм был протестирован путём его программной реализации на Python. Исходный код программы показан в следующем разделе.

# ЗАКЛЮЧЕНИЕ

Практика стала важным этапом в закреплении теоретических основ алгоритмизации и позволила развить практические навыки создания алгоритмов, их программной реализации на Python и визуализации в формате блок-схем. В ходе работы были освоены ключевые принципы алгоритмизации, основные структуры данных и методы представления алгоритмов графически.

В рамках практики решались задачи разного уровня сложности, что потребовало самостоятельной разработки алгоритмов, их кодирования на Python и оформления в виде блок-схем. Этот опыт не только углубил понимание алгоритмических подходов, но и улучшил навыки программирования.

Прохождение практики по основам алгоритмизации было успешным. Был получен ценный практический опыт в реализации алгоритмов на языке Python.

В будущем планируется продолжить изучение алгоритмов и совершенствование навыков программирования на Python, а также применение полученных знаний для решения более сложных задач.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

**–**Основы алгоритмизации [Электронный ресурс] // <https://elearn.urfu.ru/mod/resource/view.php?id=225157>, (Дата обращения: 05.07.2025)

–Гост 19.701–90 [Электронный ресурс] // <https://elearn.urfu.ru/mod/resource/view.php?id=225153>, (Дата обращения: 05.07.2025)

– Методические рекомендации для подготовки и оформления отчётов [Электронный ресурс] // <https://elearn.urfu.ru/mod/resource/view.php?id=225155>, (Дата обращения: 06.07.2025)