**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение   
высшего образования

«Национальный исследовательский Томский политехнический университет»

|  |  |
| --- | --- |
| Школа | *Инженерная школа ядерных технологий* |
| Обеспечивающее подразделение | Отделение ядерно-топливного цикла |
| Направление подготовки / специальность | *14.03.02 Ядерные физика и технологии* |
| Образовательная программа (направленность (профиль)) | *Ядерные физика и технологии* |

**ОТЧЕТ О ПРАКТИКЕ (Вариант №2)**

|  |  |
| --- | --- |
| Вид практики | *учебная* |
| Тип практики | *Учебная практика по развитию цифровых компетенций* |
| Место практики | *ОЯТЦ ИЯТШ* |

|  |  |
| --- | --- |
| Выполнил обучающийся | Бекбаев А.А. |
| Группа | 0А22 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись обучающегося)

Руководитель практикиТПУ:

Ассистент ОЯТЦ ИЯТШ Балачков М.М.

Дата проверки 14.10.2023г.

Допустить / не допустить к защите

Подпись \_\_\_\_\_\_\_\_\_\_\_\_

Итоговая оценка по практике4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(традиционная оценка, балл)

Томск 2023

Оглавление

[1. Введение. 3](#_Toc144329962)

[2. Лабораторные работы 4](#_Toc144329963)

[a) Лабораторная работа 1 4](#_Toc144329964)

[b) Лабораторная работа 2 4](#_Toc144329965)

[c) Лабораторная работа 3 4](#_Toc144329966)

[d) Лабораторная работа 4 4](#_Toc144329967)

[e) Лабораторная работа 5 4](#_Toc144329968)

[f) Лабораторная работа 6 4](#_Toc144329969)

[g) Лабораторная работа 7 5](#_Toc144329970)

[h) Лабораторная работа 8 5](#_Toc144329971)

[i) Лабораторная работа 9 5](#_Toc144329972)

[j) Лабораторная работа 10 5](#_Toc144329973)

[3. Циклы 6](#_Toc144329974)

[4. Языки программирования. 7](#_Toc144329975)

[a) Пакет Wolfram Mathematica 7](#_Toc144329976)

[b) Python. 8](#_Toc144329977)

[c) С++ 8](#_Toc144329978)

[5. Циклические операции. 9](#_Toc144329979)

[a) Цикл с предусловием 9](#_Toc144329980)

[b) Цикл с постусловием 10](#_Toc144329981)

[c) Цикл с параметром 10](#_Toc144329982)

[d) Другие методы реализации циклических операций 11](#_Toc144329983)

[6. Глава о циклах в Mathematica 12](#_Toc144329984)

[a) Виды циклов 12](#_Toc144329985)

[b) Особенности циклов в функциональном языке 12](#_Toc144329986)

[7. Циклы в Python 13](#_Toc144329987)

[a) Циклы for и while 13](#_Toc144329988)

[b) Особенности циклов в императивном языке 13](#_Toc144329989)

[8. Циклы в C++ 14](#_Toc144329990)

[a) Циклы for и while 14](#_Toc144329991)

[b) Особенности циклов в императивном языке 15](#_Toc144329992)

[9. Сравнение 16](#_Toc144329993)

[c) Цикл с параметром: 16](#_Toc144329994)

[d) Цикл с предусловием: 16](#_Toc144329995)

[e) Генерация данных: 16](#_Toc144329996)

[f) Рекурсия 16](#_Toc144329997)

[10. Вывод 17](#_Toc144329998)

[11. Заключение 18](#_Toc144329999)

1. Введение.

В рамках проведения Летней учебной практики студенты первого курса приобретают навыки работы в пакете Wolfram Mathematica. Wolfram Mathematica - это один из лидеров среди пакетов аналитической математики.

Многие математические вычисления в экономике, физике и других современных науках требуют достаточно большого количества расчетов. Очень часто приходится обрабатывать полученные результаты и визуализировать их. Выполнение таких операций вручную является достаточно трудоемкой работой. Поэтому для упрощения аналитических и численных расчетов были придуманы системы компьютерной математики. Они интегрируют в себе современный интерфейс пользователя, возможности решения как аналитических, так и численных задач, а также мощные средства визуализации полученных результатов с использованием различных типов графиков.

Ярким представителем систем компьютерной математики (СКМ) является пакет Mathematica, разработанный фирмой Wolfram Research Inc, во главе с ее президентом и главным разработчиком программ Стивеном Вольфрамом (Stephan Wolfram). Этот пакет считается одной из лучших СКМ наряду с Maple и MatLab.

В рамках проведения Летней учебной практики студентами первого курса получаются базовые знания по работе в пакете Wolfram Mathematica, признанного одним из лидеров среди пакетов аналитической математики.

Целями практики является формирование у обучающихся компетенций, необходимых для подготовки к профессиональной/научно-исследовательской деятельности.

Задачами практики являются следующие:

* ознакомление с базовыми возможностями пакета Wolfram Mathematica и решение различных математических и физических задач, как в аналитическом виде, так и в численном;
* построение графиков различных видов, как по аналитическим функциям, так и по табличным данным;
* введение в обработку физических данных.

В результате прохождения курса у студента формируются компетенции, необходимые для работы в пакте Wolfram Mathematica и способствующие изучению математических и физических наук, программирования и обработки результатов экспериментальных, статистических данных.

1. Лабораторные работы

В ходе практики были освоены базовые навыки работы с пакетом Wolfram Mathematica и возможности этого пакета. Такие как: решение различных математических и физических задач, как в аналитическом виде, так и в численном; построение графиков различных видов по аналитическим функциям и по табличным данным; обработка физических данных.

1. Лабораторная работа 1

Цель работы: приобрести базовые умения и навыки работы с пакетом Mathematica на примере решения простых задач.

Были приобретены базовые умения работы с пакетом Mathematica на примере простых задач. Использовались функции для решения задач и функция, выводящая затраченное на расчёты время.

1. Лабораторная работа 2

Цель работы: приобрести базовые умения и навыки оформления с пакетом Mathematica на примере решения уравнений.

Использовались функции для решения уравнений различными методами, которые выводили решение и численно, и аналитически.

1. Лабораторная работа 3

Цель работы: выяснить, что такое список и каким образом их можно создавать в пакете Mathematica

В ходе лабораторной работы были освоены навыки работы со списками. А именно создание разными командами, и понимание разницы форматов чтения.

1. Лабораторная работа 4

Цель работы: научиться выполнять операции и решать задачи с использованием векторов и матриц

В ходе лабораторной работы были освоены навыки выполнения операций с векторами и матрицами. А также построения графиков на основе различных примеров.

1. Лабораторная работа 5

Цель работы: научиться выполнять различные операции над списками

Были освоены навыки создания, редактирования, импортирования списков и произведения различных операций над ними.

1. Лабораторная работа 6

Цель работы: научиться пользоваться: функциями преобразования выражений; подстановками; функциями для выполнения математических вычислений

В ходе лабораторной работы были освоены навыки пользования функциями преобразований выражений, подстановками и функциями для выполнения математических вычислений.

1. Лабораторная работа 7

Цель работы: научиться пользоваться функциями построения различных типов графиков от функций, заданных аналитическими функциями

В ходе лабораторной работы были освоены навыки использования различных методов для построения графиков разных типов от функций. Были построены графики от простой функции, параметрической, полярной. Так же контурные графики, графики плотности и 3D графики. Было проведено исследование функции и построен её график.

1. Лабораторная работа 8

Цель работы: научиться строить различные графики на основе табличных данных

В ходе лабораторной работы были освоены навыки построения графиков по заданным табличным данным. Использовались различные графики, такие как контурный, плотности, в виде точек, график по датам.

1. Лабораторная работа 9

Цель работы: научиться решать дифференциальные уравнения с помощью пакета Mathematica

В ходе выполнения лабораторной работы были освоены навыки решения дифференциальных уравнений с помощью пакета Mathematica.

1. Лабораторная работа 10

Цель работы: научиться получать основные параметры экспериментальных данных, интерполировать их и аппроксимировать аналитическими формулами

В ходе выполнения лабораторной работы были освоены навыки получения основных параметров экспериментальных данных, интерполирования и аппроксимации аналитическими формулами.

1. Углубленное изучение

Тема: рассмотреть все возможные способы проведения в пакете Mathematica циклических операций. Привести примеры и провести сравнение с С++ и Python.

1. Циклы

Циклы в программировании являются базовой конструкцией, которая позволяет выполнять какие-то команды многократно. Команда, которая выполняется многократно называется телом цикла.

Зачастую циклы используются для обработки серии данных. К примеру, обработать серию экспериментов и провести для каждого эксперимента расчёты.

Существует два основных вида циклов: цикл с предусловием и цикл с постусловием. В цикле с предусловием сначала проверяется условие, и, если оно истинно, выполняется тело цикла. Этот процесс повторяется до тех пор, пока условие ложно. В цикле с постусловием сначала выполняется тело цикла, а затем проверяется условие для определения необходимости повторного выполнения.

Популярные разновидности циклов с предусловием - **while** и **for**. Цикл **while** выполняется, пока заданное условие истинно. Цикл **for** используется для перебора элементов коллекции или диапазона значений.

Циклы могут быть вложенными, когда внутри цикла организуется еще один или несколько циклов. Это позволяет решать более сложные задачи обработки многомерных данных.

Помимо циклов можно использовать различные операторы и методы, которые позволяют делать циклические операции. В некоторых языках программирования, к примеру в С++, есть оператор **goto**. Этот оператор позволяет переходить к указанному блоку в коде. Если установить блок перед использованием оператора **goto**, то блок выполнится повторно. Таким образом можно делать циклические операции.

Так же можно использовать рекурсию для исполнения операций повторно. Рекурсия — это термин, описывающий функцию, которая может вызвать сама себя. Рекурсии часто используются в различных алгоритмах из-за удобства.

1. Языки программирования.

Языки программирования — это искусственные формализованные языки, предназначенные для написания компьютерных программ. Они позволяют программисту указывать компьютеру, какие действия нужно выполнить для решения определенной задачи.

Существует множество языков программирования, отличающихся синтаксисом, парадигмой и областью применения. Популярные парадигмы - императивное, объектно-ориентированное, функциональное, логическое программирование.

Языки делятся на низкоуровневые (машинно-ориентированные) и высокоуровневые. Последние обладают более развитыми средствами для написания программ.

Примеры языков: Си, Java, Python, JavaScript, C#, PHP, Ruby, Swift. Выбор языка зависит от задачи и зачастую личных предпочтений разработчика.

В целом, языки программирования — это инструмент, позволяющий программисту управлять работой компьютера, создавая программы для решения прикладных задач в различных областях деятельности.

1. Пакет Wolfram Mathematica

Wolfram Mathematica - это популярная система компьютерной алгебры, разработанная Стивеном Вольфрамом и его компанией Wolfram Research. Она предоставляет обширные возможности для аналитических вычислений, визуализации данных, машинного обучения и многого другого.

Одной из ключевых особенностей Mathematica является ее язык программирования - достаточно высокоуровневый, декларативный и функциональный. Это позволяет лаконично выражать сложные математические и алгоритмические конструкции. В то же время Mathematica может выполнять численные расчеты с высокой эффективностью и точностью.

Встроенные функции и операторы Mathematica охватывают обширный спектр областей математики, включая алгебру, математический анализ, статистику, теорию графов и другие. Имеются эффективные алгоритмы работы с векторами, матрицами, списками и другими структурами данных.

Одно из главных преимуществ Mathematica - наглядное представление результатов с помощью визуализации данных в виде графиков, диаграмм, анимации. Так же поддерживается экспорт графики в различные форматы.

Mathematica широко используется в научных вычислениях, образовании, финансовом моделировании. Гибкость языка позволяет эффективно решать задачи обработки данных, машинного обучения и создания интерактивных приложений.

1. Python.

Python — это высокоуровневый язык программирования общего назначения с динамической типизацией. Он был создан в 1991 году и активно развивается по сей день.

Отличительные особенности Python - простой и читаемый синтаксис, похожий на естественный язык, интерпретируемость, динамическая типизация. Поддерживаются процедурное, объектно-ориентированное и функциональное программирование.

Python широко применяется для веб-разработки, анализа данных, машинного обучения, научных вычислений. Среди ключевых достоинств - обширная стандартная библиотека, простота интеграции с другими языками, кроссплатформенность.

Язык отличается высокой производительностью, подходит как для небольших, так и для крупных проектов. Программы на Python легко поддерживать и расширять по мере роста проекта.

1. С++

С++ - это компилируемый, статически типизированный язык программирования общего назначения. Он был разработан в 1980-х годах как расширение языка С с добавлением концепций объектно-ориентированного программирования.

С++ является одним из самых популярных и широко применяемых языков программирования. Он используется для создания различных приложений - от операционных систем и офисных пакетов до игр и мобильных приложений.

Основные особенности С++:

* Низкоуровневый доступ к памяти, возможность работы на уровне битов и байтов.
* Высокая производительность и возможность низкоуровневой оптимизации.
* Поддержка procedural, object-oriented и generic программирования.
* Обширный стандартный набор библиотек.
* Перегрузка функций и операторов, шаблоны классов.
* Управление памятью с помощью сборщиков мусора или вручную.

С++ сложен для освоения, но позволяет писать высокооптимизированный код для самых разных областей. Это делает его очень востребованным в промышленной разработке ПО.

1. Циклические операции.

Циклические операции — это операции, которые повторяются многократно в рамках цикла.

Цикл в программировании представляет собой конструкцию, позволяющую многократно выполнять некоторый набор инструкций (тело цикла).

К циклическим операциям относят:

* Выполнение арифметических операций в цикле (сложение чисел, возведение в степень и т.д.)
* Обработка элементов массива или другой структуры данных в цикле (перебор элементов, сортировка, поиск и т.п.)
* Вывод или считывание данных в цикле (например, вывод элементов массива)
* Вызов функций в цикле (вычисление функции для разных аргументов)
* И т.д.

Основные виды циклов:

* Цикл с предусловием (while) - повторяется, пока выполняется условие
* Цикл с постусловием (do while) - сначала выполняется тело, потом проверка условия
* Цикл с параметром (for) - повторение по элементам коллекции

1. Цикл с предусловием

Цикл с предусловием (while) - это один из основных видов циклов в программировании. Его особенности:

* Сначала вычисляется условие, пока оно истинно - выполняется тело цикла.
* Как только условие становится ложным - цикл прекращается.
* Тело цикла может выполняться как минимум один раз или ни разу.

Синтаксис на языке Python:

while *условие:*

*тело цикла*

Пример:

i = 0

while i < 10:

print(i)

i = i + 1

Здесь будут выведены числа от 0 до 9.

Условие проверяется перед каждой итерацией цикла. Это позволяет реализовать цикл с заранее неизвестным количеством повторений.

Цикл с предусловием часто используется для обработки последовательностей элементов, пока не будет достигнуто какое-то завершающее условие.

Также, в теле цикла можно менять значения, влияющие на условие, чтобы регулировать количество повторений.

1. Цикл с постусловием

Цикл с постусловием (do while) - это разновидность цикла в программировании со следующими особенностями:

* Сначала выполняется тело цикла, а затем проверяется условие.
* Если условие истинно - цикл повторяется, если ложно - цикл прекращается.
* Тело цикла выполнится хотя бы один раз, даже если условие сразу ложно.

Синтаксис на языке Python:

while True:

*тело цикла*

if *условие*:

break

Пример:

i = 10

while True:

print(i)

i = i - 1

if i == 0:

break

Этот цикл выведет числа от 10 до 1.

Цикл с постусловием удобен, когда нужно сначала выполнить действие, а затем проверить условие продолжения. Например, когда нужно считать данные до тех пор, пока они есть, не зная заранее их количества.

Главное отличие от предусловного цикла - гарантированное минимальное количество итераций. Такой цикл используется реже цикла с предусловием.

1. Цикл с параметром

Цикл с параметром (for) - это цикл, который повторяется по элементам некоторой последовательности или коллекции данных. Его особенности:

* Не нужно задавать условие, цикл повторится для каждого элемента коллекции.
* Используется для перебора и обработки массивов, списков, строк и других структур данных.
* В качестве коллекции может выступать iterator-объект или диапазон значений.

Синтаксис цикла for на Python:

for элемент in коллекция:

тело цикла

Пример:

for i in [1, 2, 3, 4, 5]:

print(i)

Этот цикл выведет числа от 1 до 5.

Цикл for часто используется вместе с функциями range() и len() для перебора индексов:

for i in range(len(some\_list)):

print(some\_list[i])

Такой подход позволяет обрабатывать элементы по индексу.

1. Другие методы реализации циклических операций

Рекурсия также может использоваться для организации циклических вычислений. Суть рекурсивного подхода - функция вызывает сама себя для решения подзадачи.

Например, для перебора элементов списка можно использовать рекурсивную функцию:

def print\_list(lst, i=0):

if i >= len(lst):

return

print(lst[i])

print\_list(lst, i+1)

Здесь на каждой итерации элемент с индексом i выводится, а затем следует рекурсивный вызов с индексом i+1 для перехода к следующему элементу.

Другой пример - рекурсивная реализация факториала:

def factorial(n):

if n == 0:

return 1

return n \* factorial(n-1)

Факториал N вычисляется через факториал N-1. Вызовы будут повторяться, пока n не станет равным 0.

Основное преимущество рекурсии - простота кода, а основной недостаток - большие накладные расходы из-за множества вызовов.

1. Глава о циклах в Mathematica
2. Виды циклов

В Mathematica доступны следующие основные циклические конструкции:

* Do-цикл - простейший цикл с параметром для выполнения кода фиксированное число раз

Синтаксис:

Do[*выражения*, {*переменная*, *начало*, *конец*}]

Do-цикл принимает выражения, которые нужно выполнить, и переменную счетчика. Переменная пробегает значения от начала до конца, выполняя на каждой итерации заданные выражения.

Пример:

Do[Print[i], {i, 1, 5}]

* While - цикл с предусловием, повторяется пока истинно условие

Синтаксис:

While[*условие*, *выражения*]

While выполняет блок выражений пока истинно условие. Проверка происходит перед каждой итерацией.

Пример:

i = 1;

While[i < 6, i++]

* Table - конструирует таблицу значений для заданных значений параметра

Синтаксис:

Table[*выражение*, {*переменная*, *начало*, *конец*}]

Table строит таблицу значений выражения для заданных значений переменной.

Пример:

Table[i^2, {i, 1, 5}]

* Рекурсивные функции - в Mathematica легко определять рекурсивные функции благодаря поддержке функционального стиля.

Рекурсивная функция определяется через вызов самой себя для подзадачи.

Пример:

factorial[n]: = n \* factorial[n - 1]

1. Особенности циклов в функциональном языке

В целом циклы в Mathematica во многом схожи с императивными языками. Но есть некоторые отличия, обусловленные функциональной парадигмой:

* Отсутствие побочных эффектов - циклы не изменяют глобального состояния
* Нельзя использовать состояние цикла (индекс и т.п.) за его пределами
* Циклы часто используются для генерации результирующей структуры данных

1. Циклы в Python
2. Циклы for и while

В Python есть два основных цикла:

* Цикл for - итерируется по элементам коллекции:

Синтаксис:

for *итератор* in *итерируемый объект*:

*тело цикла*

тут итератор — это указатель на элемент итерируемого объекта, итерируемый объект — это объект, по которому будет проходиться цикл. К примеру:

for i in [1, 2, 3]:

print(i)

где, i – является итератором, в который будут записываться значения, взятые из массива [1, 2, 3]. А массив [1, 2, 3] будет итерируемым объектом. Пример выведет числа находящиеся внутри массива по одному элементу на строку.

* Цикл while - повторяется пока истинно условие:

Синтаксис:

while *условие*:

*тело цикла*

В случае этого цикла будет проверяться условие и если оно истинно, то выполнится тело цикла. После выполнения тела цикла схема повторяется и так до тех пор пока условие не станет ложным.

К примеру:

i = 0

while i < 5:

print(i)

i += 1

тут, создаётся переменная i и ей присваивается численное значение 0. Начинается цикл с условия, если i строго меньше 5. Так как i равен 0, условие истинно и тело цикла выполнится. В теле цикла выводится нынешнее значение переменной i и после его увеличивают на 1. После нескольких итераций, когда i будет равен 4, в теле цикла он увеличится на 1 и станет равен 5. Так как 5 не меньше 5, тело цикла не будет выполняться и цикл завершится. Этот код выведет числа от 0 до 4 по одному числу на строку.

1. Особенности циклов в императивном языке

Основные черты циклов в Python:

* Изменяют состояние переменных при каждой итерации
* Могут иметь побочные эффекты в виде изменения глобальных переменных
* Часто используются для обработки данных и вычислений

Пример сортировки пузырьком:

def bubble\_sort(lst):

n = len(lst)

for i in range(n):

for j in range(0, n-i-1):

if lst[j] > lst[j+1]:

lst[j], lst[j+1] = lst[j+1], lst[j]

Пример решения задачи о выводе квадратов чисел:

n = 10

i = 1

while i <= n:

print(i\*i)

i += 1

1. Циклы в C++
2. Циклы for и while

Основные циклы в C++:

* Цикл for - итерация с счетчиком:

Синтаксис:

for (действие до начала цикла; условие продолжения цикла; действия в конце каждой итерации цикла) {

тело цикла

}

Зачастую используется такая схема:

for (счетчик = значение; счетчик < значение; шаг цикла) {

тело цикла;

}

Как показано выше в языке С++ цикл for так же использует условие для работы. Этот цикл часто используется для прохождения по элементам объекта, к примеру массива.

К примеру:

for (int i = 0; i < 10; i++) {

printf("%d\n", i);

}

Тут, инициализируется переменная i в блоке «действие до начала цикла» и ей присваивается значение 0. В блоке условия цикла ставится условие работать пока значение i меньше 10. И в блоке «действия в конце каждой итерации цикла» происходит инкрементирование на 1 используя оператор «++». Приведенный код выведет числа от 0 до 9 по одному числу на строку.

Так же цикл for можно использовать с итератором, к примеру:

std::array<int, 5> arr = {1, 2, 3, 4, 5};

std::array<int, 5>::iterator it = arr.begin();

std::array<int, 5>::iterator end = arr.end();

for (; it != end; ++it) {

std::cout << \*it << std::endl;

}

В примере с использованием итератора создается массив с числами от 1 до 5 и два указателя этого массива. В переменную it записывается указатель на начало массива, а в переменную end записывается указатель на конец массива. В блок условие стоит сравнение переменных it и end, пока они не равны цикл будет выполнятся.

* Цикл while - повторение пока условие истинно:

Синтаксис:

while( условие ){

тело цикла;

}

Цикл while используется так же, как и в python. Сначала проверяется условие и если оно истинно, то выполняется тело цикла.

К примеру:

int i = 0;

while (i < 10) {

printf("%d\n", i);

i++;

}

Пример кода выведет числа от 0 до 9 по одному числу на строку.

1. Особенности циклов в императивном языке

Характерные черты циклов в C++:

* Полный контроль индекса цикла и изменение состояния
* Возможность работы на низком уровне с памятью
* Используются для вычислений и обработки данных

Пример сортировки пузырьком

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j+1])

swap(arr[j], arr[j+1]);

}

}

}

Пример вывода квадратов чисел

int n = 10;

for (int i = 1; i <= n; i++) {

cout << i \* i << endl;

}

1. Сравнение
2. Цикл с параметром:

Mathematica:

Do[Print[i], {i, 1, 5}]

Python:

for i in range(1, 6):

print(i)

C++:

for (int i = 1; i <= 5; i++) {

std::cout << i << std::endl;

}

Во всех трех языках есть возможность организовать простой цикл по счетчику. Синтаксис различается.

1. Цикл с предусловием:

Mathematica:

While[i < 6, i++]

Python:

while i < 6:

i += 1

C++:

while (i < 6) {

i++;

}

В этом случае цикл с условием реализуется аналогично во всех языках.

1. Генерация данных:

Mathematica:

Table[i^2, {i, 1, 5}]

Python:

[i\*\*2 for i in range(1, 6)]

C++ (через алгоритм):

std::vector<int> data;

std::generate\_n(std::back\_inserter(data), 5, [](){return i\*i; });

Здесь видны различия в подходах языков.

1. Рекурсия

Рекуррентные операции аналогичны во всех трёх языках.

1. Вывод

В рамках данной работы было проведено рассмотрение и сравнение возможностей организации циклических вычислений в системе компьютерной алгебры Wolfram Mathematica и в языках программирования Python и C++.

Mathematica предоставляет развитые средства для проведения аналитических преобразований и численных расчётов. При этом язык Mathematica является декларативным и функциональным. Циклы в Mathematica позволяют компактно выразить повторяющиеся вычисления, генерировать результирующие данные. Основные циклические конструкции - Do, While, Table. Поддерживается рекурсия.

В то же время Python и C++ относятся к числу популярных императивных языков программирования. Они имеют процедурную парадигму, мутацию данных. Циклы используются для непосредственного управления вычислительным процессом. В Python применяются циклы for и while. C++ использует те же виды циклов, а также возможности работы на низком уровне.

Сравнительный анализ показал, что во всех рассмотренных языках присутствует необходимый набор возможностей для организации циклических вычислений. При этом имеются различия в подходах и синтаксисе циклов, обусловленные парадигмой языка.

1. Заключение

В ходе практики были освоены базовые навыки пакета Mathematica и рассмотрены разные задачи. В которые входят: работа с простыми и сложными арифметическим задачами, работа с уравнениями, работа со списками, матрицами и векторами, построение различных графиков на основе табличных и аналитических данных, решение комплексных задач, исследования функций и обработка физических данных.

Основные задачи и цели курса были выполнены в ходе проведения лабораторных работ. Модульно рассмотрены разные задачи и проведены анализы методов, функций и возможностей пакета Mathematica. А в ходе написания отчёта были проведены дополнительные исследования по теме языков программирования С++ и Python, так же подробно были рассмотрены цикловые операции в пакете Mathematica и в других языках программирования.

# Список литературы

Microsoft. (б.д.). *Документация по языку C++*. Получено из Microsoft: https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170

Python Software Foundation. (б.д.). *Документация языка программирования Python*. Получено из https://docs.python.org/3/

Wolfram. (б.д.). *КРАТКОЕ ВВЕДЕНИЕ ДЛЯ СТУДЕНТОВ МАТЕМАТИЧЕСКИХ СПЕЦИАЛЬНОСТЕЙ*. Получено из https://www.wolfram.com/language/fast-introduction-for-math-students/ru/notebook-documents/

В., Б. А. (б.д.). *Электронный курс "Летняя учебная практика (Пакет Mathematica) 2021"*. Получено из https://stud.lms.tpu.ru/course/view.php?id=4374

*Циклы в C++*. (б.д.). Получено из Портал о программировании: https://code-live.ru/post/cpp-loops/#for