Муниципальное бюджетное общеобразовательное учреждение города Новосибирска «Средняя общеобразовательная школа № 160»

ПРОЕКТ

На тему «История технологии программирования»

По технологии

Выполнил:

Ученик 8 «А» класса

Бордунов Сергей Викторович

Руководитель: учитель \_\_\_\_\_\_\_\_

Тимур Алдарович Авидын

Г. Новосибирск

2023 г.

Оглавление

Введение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

Глава 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

Глава 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

Заключение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

# Введение

Технологии программирования сегодня стали неотъемлемой частью нашей жизни. Они используются в разных сферах, от медицины до науки, и позволяют нам решать сложные задачи с помощью вычислительной техники. Однако, не всегда это было так. Развитие технологий программирования прошло долгий путь, начиная с электромеханических устройств и до современных высокопроизводительных компьютеров и облачных технологий.

История технологий программирования интересна тем, что каждый этап развития был связан с появлением новых методов программирования и языков программирования. Новые языки программирования облегчали процесс создания программ и позволяли программистам работать на более высоком уровне абстракции. Каждый язык программирования имел свои уникальные особенности, а некоторые языки внесли революционные изменения в области программирования.

В данном проекте мы рассмотрим историю технологий программирования, начиная с ранних электромеханических устройств и заканчивая современными облачными технологиями. Мы рассмотрим основные этапы развития, а также ключевые языки программирования, которые сыграли важную роль в этом процессе. Кроме того, мы обсудим некоторые проблемы и вызовы, с которыми столкнулась индустрия программирования на протяжении истории.

Методы исследования

Я дам разным языкам программирования одинаковую программу, после этого замерю, какой из них быстрее с этим справиться.

Актуальность: Актуальность данной темы обусловлена тем, что программирование и информационные технологии в настоящее время занимают центральное место в мировой экономике и обществе. С развитием технологий все больше процессов автоматизируются, а программирование становится ключевой профессией для многих людей.

Также стоит отметить, что технология программирования является быстро развивающейся областью, и знание ее истории поможет понимать текущие тенденции и направления развития технологий. Поэтому изучение истории технологии программирования имеет актуальное значение для студентов, профессионалов и всех, кто интересуется информационными технологиями.

Цель: Конечной целью данного проекта является ознакомление читателя с историей развития технологии программирования, включая ключевые моменты и инновации, которые позволили современным программным приложениям достигать высокой эффективности и функциональности. При этом важным аспектом является понимание влияния технологии программирования на различные сферы человеческой деятельности, а также возможностей, которые она открывает для решения различных задач и проблем.

Задачи:

1. Рассмотреть основные (часто используемые) языки программирования, историю их возникновения и их развитие.
2. Проанализировать ключевые моменты и инновации в истории программирования, описать их особенности и влияние на современные технологии.
3. Описать, какие задачи и проблемы можно решать с помощью технологий программирования, и какие возможности они открывают для различных сфер деятельности.
4. Сравнить языки программирования в различных тестах на время.
5. Узнать, какие из языков программирования наиболее востребованы на данный момент.

Гипотеза: Я считаю, что различные языки будут иметь разное время в разных тестах, причём в каких-то аспектах, некоторые будут быстрее, а каких-то – медленнее.

Методы исследования:

* Изучение литературы: анализ исторических и современных источников, включая научные статьи, книги, журналы, сборники, доклады и прочие публикации, связанные с технологией программирования.
* Статистический анализ: использование статистических методов для анализа данных, связанных с различными языками программирования, их использованием, популярностью, эффективностью и т.д.
* Экспертное мнение: сбор мнений от экспертов в области программирования, в том числе разработчиков, ученых и иных профессионалов, имеющих опыт работы с различными технологиями программирования.
* Исследования случаев: анализ опыта использования технологий программирования в реальных проектах, оценка их эффективности и возможных проблем.
* Анализ сообществ: изучение сообществ, связанных с различными технологиями программирования, их целей, интересов, мнений и других характеристик.

# ТЕОРИЯ

**Глава 1. Истоки программирования**

Первые формы "программирования" возникли задолго до появления первых компьютеров и электронных устройств. Некоторые ученые считают, что первыми "компьютерами" были абаки, которые использовались для выполнения простых арифметических операций еще в античные времена.

Однако, первые механические устройства для решения математических задач были созданы в XVII веке. К примеру, в 1642 году французский ученый Блез Паскаль создал механический калькулятор, который можно было использовать для сложения, вычитания, умножения и деления.

Другим важным механическим устройством была аналитическая машина Чарльза Бэббиджа, которую он разрабатывал в течение нескольких десятилетий в начале XIX века. Эта машина была задумана как устройство для автоматического выполнения математических операций на основе заданных инструкций. Все программы для аналитической машины Бэббиджа должны были быть записаны на перфокартах.

**1.1 Появление электронных компьютеров**

Первые электронные компьютеры были созданы в середине XX века и открыли новые возможности для программирования. Ключевыми моментами в развитии электронных компьютеров были:

• **1937 год** - Алан Тьюринг предложил концепцию универсальной машины Тьюринга, которая могла бы эмулировать работу любой другой машины,

• **1941 год** - Конрад Цузе предложил концепцию программируемой электронной машины, но из-за Второй мировой войны, его проект не был реализован.

• **1944 год** - Говард Эйкен и его команда в Гарвардском университете создали Mark I, первый электронный компьютер в США. Он использовался для расчетов по баллистическим траекториям.

• **1946** год - Джон фон Нейман представил модель компьютера, в которой программа хранилась в том же виде, что и данные. Это было революционным новшеством, позволяющим создавать более гибкие и универсальные компьютеры.

• **1951** год - Грейс Хоппер разработала первый компилятор, который преобразовывал код на высокоуровневом языке в машинный код. Это значительно ускорило процесс программирования и сделало его более доступным.

• **1957** год - IBM разработала FORTRAN (Название Fortran является сокращением от FORmula TRANslator (формульный транслятор)), первый высокоуровневый язык программирования, который позволял программистам писать код на более простых и понятных языках, вместо того, чтобы писать машинный код.

• **1960** год – ALGOL (algorithmic language) был разработан как международный язык программирования, чтобы стандартизировать и упростить процесс программирования. Он стал предшественником языков программирования, таких как Pascal и C.

• В последующие годы наблюдался резкий рост интереса к программированию, и в 1970 году был разработан язык Pascal, который стал одним из наиболее популярных языков для обучения программированию в учебных заведениях.

• В 1980-е годы резко возросла персональных компьютеров, что привело к созданию новых языков программирования, таких как C++ и Objective-C. Они позволяли создавать более сложные программы с графическим интерфейсом, а также использовать объектно-ориентированный подход к программированию.

В 1990-е годы интернет начал активно развиваться, что привело к созданию языков программирования, которые специально предназначены для работы с сетью. Одним из таких языков стал JavaScript, который стал широко использоваться для создания динамических веб-сайтов.

**1.2 Современные языки программирования**

Сегодня существует огромное количество различных языков программирования, каждый из которых предназначен для выполнения определенных задач. Некоторые из них используются для создания веб-приложений, другие - для научных расчетов, а еще другие - для создания мобильных приложений.

Одним из сам ими популярных языков программирования является Java, разработанный компанией Sun Microsystems (SUN - Stanford University Networks) в 1995 году. Java используется для создания приложений для компьютеров и мобильных устройств, а также для создания веб-приложений.

Еще одним популярным языком программирования является Python, который был разработан в конце 1980-х годов. Python используется для создания приложений в области машинного обучения, научных исследований, веб-приложений и многого другого.

JavaScript – тоже довольно популярный язык программирования, который используется для создания динамических и интерактивных веб-страниц. Он был создан компанией Netscape в 1995 году и быстро стал одним из самых популярных языков программирования в веб-разработке.

Существуют также специализированные языки программирования, которые используются для конкретных задач. Например, SQL (Structured Query Language) используется для работы с базами данных, а MATLAB (Matrix Laboratory) - для научных вычислений.

Как уже упоминалось, на заре программирования основным языком программирования был ассемблер, но с развитием вычислительной техники и появлением новых задач возникла необходимость в более высокоуровневых языках программирования. Первым языком, который отвечал этим требованиям, стал Фортран (FORmula TRANslation), созданный в 1957 году Джоном Бэкусом и его коллегами в IBM. Он предназначался для численных расчетов и операций с матрицами. Фортран имел удобный синтаксис, позволяющий описывать математические формулы так же, как их записывают на бумаге, и автоматически переводил их в машинный код, что значительно ускоряло процесс программирования.

Следующим значительным языком стал COBOL (COmmon Business-Oriented Language), созданный в 1959 году. Он предназначался для решения бизнес-задач, таких как учет и финансовые расчеты. COBOL имел схожий с английским язык синтаксис, что делало его более понятным для бизнес-специалистов, которые не имели опыта в программировании.

Другим важным моментом в истории технологии программирования было появление языка С, созданного в 1972 году в Bell Labs. Этот язык программирования имел низкоуровневый синтаксис, позволяющий написать программы, которые могут быть скомпилированы в машинный код с небольшими потерями производительности. С языком С связаны такие важные события, как создание операционной системы UNIX и разработка языка С++, который стал расширением С и добавил объектно-ориентированное программирование.

С развитием вычислительной техники появились и другие языки программирования, такие как Паскаль, Бейсик, Дели, Ада и многие другие. Каждый из них имел свои преимущества и недостатки, что делало их более или менее подходящими для решения тех или иных задач.

На сегодняшний день, самым популярным языком программирования является Python. По словам экспертов, кроется в относительной простоте Python в изучении, большом разнообразии библиотек (подключаемые дополнения, расширяющие функционал языка) и активном развитии технологий искусственного интеллекта, где применяется язык. А также в использовании Python при разработке различных продуктов машинного обучения, в анализе данных и научно-исследовательской деятельности.

В 21 веке технологии программирования продолжают развиваться и эволюционировать. Разработчики используют новые методы и инструменты, такие как искусственный интеллект, машинное обучение, интернет вещей (технология, которая позволяет устройствам подключаться к интернету и взаимодействовать друг с другом, передавая данные без необходимости человеческого вмешательства) и блокчейн (технология, которая позволяет создавать надежную и безопасную запись транзакций и информации в цифровой форме), чтобы создавать новое и улучшать существующее программное обеспечение. Кроме того, появление облачных вычислений (вычисления, происходящие на сервере) и распределенных систем позволило создавать программное обеспечение, которое может работать на многих устройствах и платформах, делая его более гибким и доступным для пользователей.

В целом, история технологии программирования отражает широкий спектр изменений в нашей жизни и обществе, начиная от первых механических устройств до современных мощных компьютеров и мобильных устройств. Каждый этап в этой истории был связан с различными вызовами и возможностями, которые привели к развитию новых методов и инструментов для программирования. Понимание этой истории может помочь нам лучше понять, как мы пришли к современному программному обеспечению и куда мы можем двигаться дальше в будущем.

#  ПРАКТИКА

**Глава 2. Основные языки программирования:**

**2.1. Python**

***Python*** – это высокоуровневый интерпретируемый язык программирования. Обладает простым и легко читабельным синтаксисом. На данный момент самый популярный язык программирования.

История языка программирования Python началась в конце 1980-х. Гвидо ван Россум задумал Python в 1980-х годах, а приступил к его созданию в декабре 1989 года в центре математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Амёба. Ван Россум является основным автором Python и продолжал выполнять центральную роль в принятии решений относительно развития языка вплоть до 12 июля 2018 года.

В феврале 1991 года Гвидо ван Россум опубликовал код Python, помеченный версией 0.9.0, на alt.sources. На этой стадии в нём уже присутствовали классы с наследованием, обработка исключений, функции и основные типы данных: list, dict, str и т. д.. Также в этом начальном релизе были модули, заимствованные из Modula-3. Ван Россум описывал модуль как «один из главных элементов в программировании на Python». Модель обработки исключений в Python тоже походила на Modula-3 с добавлением оператора else. В 1994 году с ростом числа пользователей сформировалась группа comp.lang.python — основной форум Python.

январь 1994 года – появление Python 1.0. Основными новыми возможностями, включёнными в этот релиз, были средства функционального программирования: лямбда-исчисление, map, filter и свёртка списка. Ван Россум утверждал, что «Python приобрёл lambda, reduce(), filter() и map() благодаря любителю Lisp, которому их не хватало, и он предоставил патчи, реализующие эти функции». К версии 1.4 Python включал в себя множество новых функций, среди которых наиболее заметными были позаимствованные в Modula-3 именованные параметры (также подобные параметрам Common Lisp) и встроенная поддержка комплексных чисел. Также в 1.4 появилась простая форма сокрытия данных при помощи name mangling (которое, правда, легко обходится).

2000 год – ядро команды разработчиков перешло в BeOpen.com, сформировав команду BeOpen PythonLab. В этом же году были выпущены версии 1.6 и 2.0. Python 2.0 был единственным релизом BeOpen.com. После него Ван Россум и остальные разработчики PythonLab присоединились к Digital Creations.

В версии Python 2.0 появилось списковое включение — функция, заимствованная из функциональных языков программирования SETL и Haskell. Синтаксис в Python для этой конструкции очень похож на Haskell, за исключением того, что в Haskell предпочли использовать символы пунктуации, а в Python — ключевые слова. Также в Python 2.0 была добавлена система сборки мусора с поддержкой циклических ссылок.

17 апреля 2001 года – Выход Pytnon 2.1. Релиз включал изменение в спецификацию языка, поддерживающее вложенные области видимости, как в языках со статической (лексической) областью видимости. Эта возможность была выключена по умолчанию и не потребовалась до Python 2.2.

21 декабря 2001 года – Выход 2.2. Главным нововведением было объединение базовых типов Python и классов, создаваемых пользователем, в одной иерархии. Это сделало Python полностью объектно-ориентированным языком. Тогда же были добавлены генераторы, идея которых заимствована из Icon.

3 декабря 2008 года – был выпущен Python 3.0. Разрабатывался с целью устранения фундаментальных изъянов в языке. Эти изменения не могли быть сделаны при условии сохранения полной обратной совместимости с 2.x версией, поэтому потребовалось изменение главного номера версии. Ведущим принципом разработки Python 3 было: «уменьшение дублирующейся функциональности устранением устаревших способов сделать это».

Основные изменения, внесённые в версии 3.0:

1. Синтаксическая возможность для аннотации параметров и результата функций (например, для передачи информации о типе или документирования).
2. Полный переход на unicode для строк.
3. Введение нового типа «неизменяемые байты» и типа «изменяемый буфер». Оба необходимы для представления бинарных данных.
4. Новая подсистема ввода-вывода (модуль io), имеющая отдельные представления для бинарных и текстовых данных.
5. Абстрактные классы, абстрактные методы (доступно уже в 2.6).
6. Иерархия типов для чисел.
7. Выражения для словарей и множеств {k: v for k, v in a\_dict} и {el1, el2, el3} (по аналогии со списковыми выражениями). Эта возможность была также добавлена в Python 2.7
8. Изменения print из встроенного выражения во встроенную функцию. Это позволит модулям делать изменения, подстраиваясь под разное использование функции, а также упростит код. В Python 2.7 эта возможность активируется вводом from \_\_future\_\_ import print\_function.
9. Перемещение reduce (но не map или filter) из встроенного пространства в модуль functools (использование reduce существенно менее читабельно по сравнению с циклом).
10. Удаление некоторых устаревших возможностей, поддерживаемых в ветке 2.x для совместимости, в частности: классы старого стиля, целочисленное деление с обрезанием результата как поведение по умолчанию, строковые исключения, неявный относительный импорт, оператор exec и т. п.
11. Реорганизация стандартной библиотеки.
12. Новый синтаксис для метаклассов.
13. Изменён синтаксис присваиваний. Стало возможным, например, присваивание a, \*rest, b = range(5). С другой стороны, формальные параметры функций вроде def foo(a, (b, c)) более недопустимы.

Python используется в различных областях, включая:

Веб-разработка: Python используется для создания веб-приложений и серверных приложений, например, Django и Flask.

Научные вычисления: Python используется в научных и исследовательских областях, таких как математика, физика, биология и другие науки. NumPy, SciPy и Pandas - это некоторые из наиболее популярных библиотек Python для научных вычислений.

Машинное обучение и искусственный интеллект: Python используется для создания алгоритмов машинного обучения и обработки данных, таких как TensorFlow, Keras, PyTorch, Scikit-learn и другие.

Автоматизация задач: Python может использоваться для автоматизации рутинных задач, таких как обработка текстовых файлов, создание отчетов и скриптов для анализа данных.

Игры: Python может использоваться для создания компьютерных игр, как в качестве основного языка, так и в качестве языка для написания сценариев.

Desktop-приложения: Python может использоваться для создания desktop-приложений, как с помощью стандартной библиотеки, так и с помощью фреймворков, таких как PyQt и wxPython.

Блокчейн: Python может использоваться для создания приложений на основе технологии блокчейн, таких как Ethereum и Bitcoin.

Web scraping: Python может использоваться для извлечения данных из веб-страниц, например, с помощью библиотеки BeautifulSoup.

Разработка приложений для мобильных устройств: Python может использоваться для создания приложений для Android и iOS, например, с помощью фреймворка Kivy.

Разработка GUI-интерфейсов: Python может использоваться для создания приложений с графическим интерфейсом пользователя, например, с помощью фреймворка Tkinter.

**2.2. C**

***C*** - компилируемый язык программирования общего назначения, используется для создания операционных систем и низкоуровневых приложений. 2 по популярности язык программирования.

Существует несколько легенд, касающихся причин разработки Си и его отношения к операционной системе UNIX, включая следующие:

По одной легенде разработка Си стала результатом того, что его будущие авторы любили компьютерную игру, подобную популярной игре Asteroids. Они уже давно играли в неё на главном сервере компании, который был недостаточно мощным и должен был обслуживать около ста пользователей. Томпсон и Ритчи посчитали, что им не хватает контроля над космическим кораблём для того, чтобы избегать столкновений с некоторыми камнями. Поэтому они решили перенести игру на свободный PDP-7, стоящий в офисе. Однако этот компьютер не имел операционной системы, что заставило их её написать. В конце концов, они решили перенести эту операционную систему ещё и на офисный PDP-11, что было очень тяжело, потому что её код был целиком написан на ассемблере. Было вынесено предложение использовать какой-нибудь высокоуровневый портируемый язык, чтобы можно было легко переносить ОС с одного компьютера на другой. Язык Би, который они хотели сначала задействовать для этого, оказался лишён функциональности, способной использовать новые возможности PDP-11. Поэтому они и остановились на разработке языка Си.

Существует и другая легенда. Первый компьютер, для которого была первоначально написана UNIX, предназначался для создания системы автоматического заполнения документов. Первая версия UNIX была написана на ассемблере. Позднее для того, чтобы переписать эту операционную систему, был разработан язык Си.

В 1978 году Брайан Керниган и Деннис Ритчи опубликовали первую редакцию книги «Язык программирования Си». Эта книга, известная среди программистов как «K&R», служила многие годы неформальной спецификацией языка. Версию языка Си, описанную в ней, часто называют «K&R C». Вторая редакция этой книги посвящена более позднему стандарту ANSI C, описанному ниже.

1. K&R ввёл следующие особенности языка:
2. структуры (тип данных struct);
3. длинное целое (тип данных long int);
4. целое без знака (тип данных unsigned int);
5. оператор += и подобные ему (старые операторы =+ вводили анализатор лексики компилятора Си в заблуждение, например, при сравнении выражений i =+ 10 и i = +10).

После публикации K&R C в язык было добавлено несколько возможностей, поддерживаемых компиляторами AT&T и некоторых других производителей:

функции, не возвращающие значение (с типом void), и указатели, не имеющие типа (с типом void\*); функции, возвращающие объединения и структуры; имена полей данных структур в разных пространствах имён для каждой структуры; присваивания структур; спецификатор констант (const); стандартная библиотека, реализующая большую часть функций, введённых различными производителями; перечислимый тип (enum); дробное число одинарной точности (float).

***Стандарты языка C***

**ISO C**

В конце 1970-х годов Си начал вытеснять Бейсик с позиции ведущего языка для программирования микрокомпьютеров. В 1980-х годах он был адаптирован для использования в IBM PC, что привело к резкому росту его популярности. В то же время Бьёрн Страуструп и другие в лабораториях Bell Labs начали работу по добавлению в Си возможностей объектно-ориентированного программирования. Язык, который они в итоге сделали, C++, оказал большое влияние на разработку ПО, но так и не смог сравняться по популярности с Си, особенно в UNIX-подобных системах.

В 1983 году Американский национальный институт стандартов (ANSI) сформировал комитет для разработки стандартной спецификации Си. По окончании этого долгого и сложного процесса в 1989 году он был наконец утверждён как «Язык программирования Си» ANSI X3.159-1989. Эту версию языка принято называть ANSI C или C89. В 1990 году стандарт ANSI C был принят с небольшими изменениями Международной организацией по стандартизации (ISO) как ISO/IEC 9899:1990.

Одной из целей этого стандарта была разработка надмножества K&R C, включающего многие особенности языка, созданные позднее. Однако комитет по стандартизации также включил в него и несколько новых возможностей, таких, как прототипы функций (заимствованные из C++) и более сложный препроцессор.

ANSI C сейчас поддерживают почти все существующие компиляторы. Почти весь код Си, написанный в последнее время, соответствует ANSI C. Любая программа, написанная только на стандартном Си, гарантированно будет правильно выполняться на любой платформе, имеющей соответствующую реализацию Си. Однако большинство программ написаны так, что они будут компилироваться и исполняться только на определённой платформе. Происходит это из-за того, что разные ОС (Операционные системы) используют нестандартные библиотеки, например, для графических дисплеев;

они используют специфические платформо-зависимые средства;

они рассчитаны на определённое значение размера некоторых типов данных или на определённый способ хранения этих данных в памяти для конкретной платформы.

**С99**

После стандартизации в ANSI спецификация языка Си оставалась относительно неизменной в течение долгого времени, в то время как C++ продолжал развиваться. Однако в конце 1990-х годов стандарт подвергся пересмотру, что привело к публикации ISO 9899:1999 в 1999 году. Этот стандарт обычно называют «C99». В марте 2000 года он был принят и адаптирован ANSI.

Некоторые новые особенности C99:

1. подставляемые функции (inline);
2. объявление локальных переменных в любом операторе программного текста (как в C++);
3. новые типы данных, такие, как long long int (для облегчения перехода от 32- к 64-битным числам), явный булевый тип данных \_Bool и тип complex для представления комплексных чисел;
4. массивы переменной длины;
5. поддержка ограниченных указателей (restrict);
6. именованная инициализация структур: struct { int x, y, z; } point = { .y=10, .z=20, .x=30 };
7. поддержка однострочных комментариев, начинающихся на //, заимствованных из C++ (многие компиляторы Си поддерживали их и ранее в качестве дополнения);
8. несколько новых библиотечных функций, таких, как snprintf;
9. несколько новых заголовочных файлов, таких, как stdint.h.

**С11**

8 декабря 2011 опубликован новый стандарт для языка Си (ISO/IEC 9899:2011). Основные изменения:

1. поддержка многопоточности;
2. улучшенная поддержка Юникода;
3. обобщённые макросы (type-generic expressions, позволяют статичную перегрузку);
4. анонимные структуры и объединения (упрощают обращение ко вложенным конструкциям);
5. управление выравниванием объектов;
6. статичные утверждения (static assertions);
7. удаление опасной функции gets (в пользу безопасной gets\_s);
8. функция quick\_exit;
9. спецификатор функции \_Noreturn;
10. новый режим эксклюзивного открытия файла.

Язык программирования C был создан для написания операционных систем, но он также нашел широкое применение в других областях. Вот некоторые из них:

1. Разработка операционных систем: C используется для разработки операционных систем, таких как UNIX, Linux, Windows и Mac OS X.
2. Разработка приложений: C используется для разработки приложений, таких как Adobe Photoshop, Microsoft Office и MySQL.
3. Разработка игр: C используется для разработки игр, таких как Doom, Quake и Grand Theft Auto.
4. Разработка драйверов: C используется для разработки драйверов устройств, которые обеспечивают коммуникацию между аппаратным и программным обеспечением.
5. Разработка веб-серверов: C используется для разработки веб-серверов, таких как Apache и Nginx.
6. Разработка криптографических приложений: C используется для разработки криптографических приложений, таких как OpenSSL и TrueCrypt.
7. Разработка встроенных систем: C используется для разработки встроенных систем, таких как микроконтроллеры и микропроцессоры.
8. Научные вычисления: C используется для научных вычислений в области физики, математики, биологии и других наук.
9. Разработка компиляторов и интерпретаторов: C используется для разработки компиляторов и интерпретаторов, которые переводят исходный код на другие языки программирования или выполняют его непосредственно.
10. Разработка сетевых протоколов: C используется для разработки сетевых протоколов, таких как TCP/IP, DNS и SMTP.

**2.3. Паскаль**

Разработчик Pascal — швейцарский ученый Никлаус Вирт. Это один из наиболее заметных специалистов-информатиков своего времени, создатель многих языков программирования. В конце 60-х годов прошлого века он участвовал в комиссии по разработке стандарта языка Алгол-68. Часть этих наработок он использовал затем при создании Pascal. Название новому языку Вирт дал в честь французского ученого, механика и изобретателя Блеза Паскаля, создавшего одно из первых механических счетных устройств (арифмометр).

Главной целью Вирта было создание удобного инструмента для обучения студентов структурному программированию. Благодаря интуитивно понятному синтаксису Pascal даже новичок-любитель способен написать на нем стройный, понятный код.

Дальнейшее развитие Pascal привело к появлению многих диалектов этого языка, созданных как самим Виртом, так и сторонними разработчиками:

В 1983 году специалистами фирмы Borland был разработан диалект Turbo Pascal и интегрированная среда разработки с отладчиком и встроенными библиотеками на его основе. Он оказался очень популярным за счет быстрой компиляции и исполнения кода, поэтому стал основой для программирования прикладных приложений на популярных микрокомпьютерах того времени.

В 1986 году на базе Pascal появился объектно-ориентированный язык программирования Delphi с собственной средой разработки. Он в целом сохранил простоту и интуитивность своего прародителя и стал распространенным инструментом прикладного программирования. Однако, в отличие от ориентированного на обучение Pascal, язык и среда Delphi стали использоваться и для решения практических задач — например, программирования промышленных контроллеров.

Важным этапом стало появление свободных версий языка Pascal. Например, Free Pascal и GNU Pascal представляют собой диалекты Object Pascal со свободно распространяемым компилятором. Благодаря ему написанные на этом языке приложения можно было запускать на различных платформах под управлением разных операционных систем.

Несмотря на то, что Pascal больше не является самым популярным языком программирования, его до сих пор используют в учебных заведениях и в некоторых коммерческих проектах.

Ниже приведены некоторые области, в которых Pascal может использоваться:

Обучение программированию: Pascal часто используется в качестве языка для обучения программированию в университетах и колледжах. Этот язык был создан для обучения программированию и изучения основных понятий, таких как переменные, циклы и условия.

Разработка приложений: Pascal можно использовать для создания различных типов приложений, таких как консольные приложения и графические интерфейсы пользователя.

Научные вычисления: Pascal может использоваться для вычисления математических формул, а также для выполнения научных и инженерных расчетов.

Системное программирование: Pascal может использоваться для создания операционных систем и другого системного программного обеспечения.

Создание игр: Pascal может использоваться для создания простых игр

Разработка веб-приложений: Pascal может использоваться для создания веб-приложений, используя такие инструменты, как Lazarus и Free Pascal.

Разработка баз данных: Pascal может использоваться для создания приложений, работающих с базами данных, например, при создании систем управления базами данных.

Разработка приложений для мобильных устройств: Pascal может использоваться для создания приложений для мобильных устройств, используя такие инструменты, как Lazarus и Free Pascal.

Автоматизация задач: Pascal может использоваться для автоматизации задач, таких как обработка данных, генерация отчетов и т.д.

Создание инструментов: Pascal может использоваться для создания инструментов для разработки программного обеспечения, таких как отладчики, компиляторы и интерпретаторы.

**2.4. Javascript**

Javascript - язык программирования, используемый для создания веб-приложений и интерактивных пользовательских интерфейсов.

Первые предпосылки для появления этого языка появились еще в 1992 году, когда была начата разработка скриптового встраиваемого языка Cmm (Си минус минус). Позже он был переименован в ScriptEase, так как название Си минус минус имело негативный оттенок. В целом. Перед тем, как язык обрел свое современное имя, его название менялось еще несколько раз.

В 1995 году Брендан Эйх получил задачу внедрить язык программирования в браузер Netscape. Изначально язык назывался Mocha, затем LiveScript. Наконец, он получил свое современное имя – JavaScript. Здесь разработчики пошли на хитрость. В то время, когда они занимались улучшением LiveScript, довольно большой популярностью пользовался язык Java. Для того, чтобы привлечь больше разработчиков для работы с новым языком, было решено использовать в его названии Java. В итоге получился JavaScript.

Последняя на сегодняшний день версия языка ES6 вышла в 2015 году. С ее появлением язык обрел вторую жизнь. Появились новые стандарты, а также возможность работы с константами. Претерпел изменения и сам код. Язык придерживается принципа сокращения кода при большей функциональности.

JavaScript (JS) - это язык программирования, который обычно используется для создания интерактивных веб-страниц и веб-приложений. JS может выполняться на стороне клиента (веб-браузере) и на стороне сервера (внутри Node.js). Среди основных областей, где может использоваться JavaScript, можно выделить:

1. Разработка веб-приложений: JS используется для создания интерактивных пользовательских интерфейсов и обработки данных на клиентской стороне.
2. Разработка серверных приложений: JS может выполняться внутри сервера с помощью Node.js, что позволяет создавать серверные приложения и API.
3. Создание мобильных приложений: JS используется для создания гибридных мобильных приложений, которые могут работать как на Android, так и на iOS.
4. Разработка игр: JS может использоваться для создания игр, особенно для браузерных игр.
5. Разработка десктопных приложений: JS может быть использован для создания десктопных приложений, основанных на Electron или NW.js.
6. Автоматизация задач: JS может использоваться для автоматизации рутиных задач, например, для написания скриптов для сборки и развертывания приложений.
7. Разработка расширений для браузеров: JS используется для создания расширений для браузеров, которые могут изменять функциональность браузера и добавлять новые возможности.
8. Разработка IoT-устройств: JS может использоваться для программирования устройств IoT, таких как роботы, умные дома и т.д.

## Тестирование на время

Для тестов, я буду использовать программу по сортировке массива. Программе будет даваться список с 10000 элементов со случайными значениями от 0 до 10000. Программа должна будет отсортировать этот список в порядке возрастания. В качестве списка я буду использовать константу (прилагается в файле *array.txt*).

Я буду писать программы для выше упомянутых языков (код приложен в файлах «тест\_на\_время.\*\*\*» с разными расширениями для разных языков.

Так как каждый из этих языков программирования я знаю, я написал на них одинаковые по «смыслу» программы. Точно также я прикрепил все файлы (но не рекомендую смотреть на файл паскаля).

Вот время выполнения программ в таблице

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Pascal | JavaScript (Node JS) | C |
| 9.8 | 0.4 | 0.25 | 0.3 |

Как можно из этого заметить, самым быстрым оказался Node JS, после него идёт C, после чего идёт Pascal, и с ОГРОМНОЙ разницей Python.

Помимо этого, я также решил провести тест с другой программой: она будет находить факториал из 1000. Программы тоже прикреплены

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Pascal | Javascript (Node JS) | C |
| 0.003 | < 0.001 | 0.007 | 0.001 |

Как можно заметить, результаты довольно заметно отличаются. Например, в первом случае Python выполнялся дольше всех, а во втором случае, наоборот – 2-ой по быстроте. При этом быстрее всех был C, а стал Pascal.

Я считаю, что разные языки программирования устроены по разному (какие-то из них компилируются в c, какие-то напрямую в assembly), из-за чего их скорость в разных функциях может менятся.

Помимо этого, в интернете я нашёл статистику с популярностью разных языков программирования.

**Python** – 1

**Pascal** - 211

**Javascript** – 7

**C** – 2

Таким образом, не смотря на его скорость, Python – самый популярный язык, вместе с тем и самый лёгкий, из-за чего его часто учат новички.

В сравнении с ним C – довольно извилистый, в нём можно очень долго разбираться, но при этом он считается одним из быстрых языков, поскольку выполняется практически на системном уровне, давая огромные возможности.

# Заключение

Нельзя однозначно сказать, какой язык программирования лучше, или хуже. Разные языки имеет разные достоинства, но и при этом разные минусы. (Python – быстрый, для него есть много пользовательских библиотек, в каких-то задачах он очень медленный, C – сложный в изучении, но позволяет делать много всего встроено и быстрый).

Я считаю, что современному программисту нужно владеть разными языками, ведь таким образом можно будет гораздо легче решать другие задачи. Это сделает более универсальным и востребованным в современном мире.

# Используемая литература

1. Computer History Museum - https://www.computerhistory.org/
2. IEEE Computer Society - https://www.computer.org/
3. History of Programming Languages (HOPL) - https://hopl4.sigplan.org/
4. The History of Computers - https://www.computerhistory.org/timeline/
5. History of Programming Languages - https://en.wikipedia.org/wiki/History\_of\_programming\_languages
6. A Brief History of Programming Languages - https://www.lewisu.edu/experts/wordpress/index.php/a-brief-history-of-programming-languages/
7. Brief History of Computer Science - https://www.cs.cmu.edu/~15110-s13/Wing06-ctc.pdf
8. Top 10 Most Popular Programming Languages - https://www.investopedia.com/top
9. Tiobe (The Importance of Being Earnest) - https://www.tiobe.com/tiobe-index/
10. Лидерство Python в рейтинге Tiobe: долгий путь к первому месту и зарплатные ожидания разработчиков - https://habr.com/ru/company/skillbox/blog/648415/
11. История Python - https://ru.wikipedia.org/wiki/История\_языка\_программирования\_Python
12. История C - https://ru.wikipedia.org/wiki/История\_языка\_Си
13. История Pascal - https://blog.skillfactory.ru/glossary/pascal/
14. История JS - https://progkids.com/blog/javascript