МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»**

**(СибГУ)**

ИНСТИТУТ (ФАКУЛЬТЕТ) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРОФИЛЬ ПОДГОТОВКИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ЕЖЕДНЕВНОМ БЫТУ

Студент Хакимов Дамир Рашитович

Руководитель Коморовский Витольд Станиславович

Рецензент (Ф.И.О.)

Нормоконтроль (Ф.И.О.)

Допускается к защите

Зав. Кафедрой (Ф.И.О.)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

Красноярск 20\_\_\_ г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»**

**(СибГУ)**

УТВЕРЖДАЮ

Зав. кафедрой ИУС

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ЗАДАНИЕ**

**на магистерскую диссертацию**

Студенту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

направления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

магистерской программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема диссертации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Утверждена приказом по университету №\_\_\_\_\_\_\_от «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Срок сдачи студентом первого варианта диссертации «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Срок сдачи студентом законченной диссертации «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

Исходные данные к диссертации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание диссертации (перечень вопросов, подлежащих разработке)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень графического материала с указанием обязательных чертежей

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

Нормоконтроль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_»\_\_\_\_\_\_ 20\_\_г. \_\_\_\_\_\_\_\_\_\_ /Коморовский Витольд Станиславович/

**АННОТАЦИЯ**

ХАКИМОВ ДАМИР РАШИТОВИЧ

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ЕЖЕДНЕВНОМ БЫТУ

НАПРАВЛЕНИЕ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МАГИСТЕРСКАЯ ПРОГРАММА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УЧЕБНАЯ ГРУППА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ГОД ЗАЩИТЫ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРИМЕР.** В магистерской диссертации исследован вопрос и далее по работе.

**THE SUMMARY**

DEVELOPMENT OF THE SYSTEM OF SUPPORT OF DECISION-MAKING DURING EVERYDAY LIFE.

**EXAMPLE.** In master’s thesis investigated the issue of strategic planning and etc.

Содержание

[Введение 6](#_Toc482816185)

[1 Системы поддержки принятия решений 8](#_Toc482816186)

[1.1 Классификации СППР 10](#_Toc482816187)

[1.2 Архитектура СППР 11](#_Toc482816188)

[1.3 Структура СППР 12](#_Toc482816189)

[1.4 Динамическое моделирование 13](#_Toc482816190)

[1.5 Преимущества 13](#_Toc482816191)

[1.6 Метод анализа иерархий 13](#_Toc482816192)

[1.7 Пример задачи многокритериального выбора с простейшей иерархией 15](#_Toc482816193)

[1.8 Сфера образования и научных исследований 16](#_Toc482816194)

[1.9 Методика применения МАИ 16](#_Toc482816195)

[1.10 Моделирование проблемы в виде иерархии 18](#_Toc482816196)

[1.10.1 Определение иерархической структуры 18](#_Toc482816197)

[1.10.2 Объяснение иерархических структур, используемых в МАИ 19](#_Toc482816198)

[1.11 Расстановка приоритетов 20](#_Toc482816199)

[1.11.1 Определение приоритетов и пояснения 20](#_Toc482816200)

[2 Человеко-машинное взаимодействие. 24](#_Toc482816201)

[2.1 Высокоуровневый язык программирования Python 24](#_Toc482816202)

[2.2 Философия 25](#_Toc482816203)

[2.3 История 27](#_Toc482816204)

[2.4 Влияние других языков на Python 28](#_Toc482816205)

[2.5 Портируемость 29](#_Toc482816206)

[2.6 Типы и структуры данных 30](#_Toc482816207)

[2.7 Синтаксис и семантика 31](#_Toc482816208)

[2.8 Функциональные возможности 35](#_Toc482816209)

[2.9 Прикладное применение Python 37](#_Toc482816210)

[2.10 Преимущество Python перед другими языками высокого уровня 38](#_Toc482816211)

[2.11 Реализации Python 39](#_Toc482816212)

[2.12 Недостатки 41](#_Toc482816213)

[3 Основые причины выбора мессенджера Telegram 43](#_Toc482816214)

[3.1 Telegram (мессенджер) 47](#_Toc482816215)

[3.2 История 48](#_Toc482816216)

[3.3 Технология 51](#_Toc482816217)

[3.4 Боты 51](#_Toc482816218)

[3.5 Конкурсы поиска уязвимостей 52](#_Toc482816219)

[3.6 Критика 53](#_Toc482816220)

[3.7 Цензура 54](#_Toc482816221)

[4 Моделирование системы поддержки принятия решений при обращении к боту мессенджеру? 56](#_Toc482816222)

[Источники 58](#_Toc482816223)

Введение

В той или иной степени Системы Поддержки Принятия Решений (СППР) присутствуют в любой информационной системе (ИС). Поэтому к задаче создания системы поддержки принятия решений организации приступают сразу после приобретения вычислительной техники и установки программного обеспечения. По мере развития бизнеса, упорядочения структуры организации и налаживания межкорпоративных связей, проблема разработки и внедрения СППР становится особенно актуальной. Одним из подходов к созданию таких систем стало использование хранилищ данных.

СППР можно, в зависимости от данных, c которыми они работают, разделить на оперативные, предназначенные для немедленного реагирования на текущую ситуацию, и стратегические - основанные на анализе большого количества информации из разных источников с привлечением сведений, содержащихся в системах, аккумулирующих опыт решения проблем.

СППР первого типа получили название Информационных Систем Руководства (ИСР). По сути, они представляют собой конечные наборы отчетов, построенные на основании данных из транзакционной информационной системы предприятия или OLTP-системы, в идеале адекватно отражающей в режиме реального времени все аспекты производственного цикла предприятия. Для ИСР характерны следующие основные черты:

- отчеты, как правило, базируются на стандартных для организации запросах; число последних относительно невелико;

- ИСР представляет отчеты в максимально удобном виде, включающем, наряду с таблицами, деловую графику, мультимедийные возможности и т. п.;

- как правило, ИСР ориентированы на конкретный вертикальный рынок, например финансы, маркетинг, управление ресурсами.

СППР второго типа предполагают достаточно глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня являются правила принятия решений, которые на основе агрегированных данных подсказывают менеджерскому составу выводы и придают системе черты искусственного интеллекта. Такого рода системы создаются только в том случае, если структура бизнеса уже достаточно определена и имеются основания для обобщения и анализа не только данных, но и процессов их обработки. Если ИСР есть не что иное как развитие системы оперативного управления производственными процессами, то СППР в современном понимании - это механизм развития бизнеса, который включает в себя некоторую часть управляющей информационной системы, обширную систему внешних связей предприятия, а также технологические и маркетинговые процессы развития производства.

Не окончательный вариант, необходима доработка.

1 Системы поддержки принятия решений

Система поддержки принятия решений - это компьютерные системы, предназначенные для сбора и анализа больших объемов информации с целью оценивания возможных вариантов решений.

СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных.

Система поддержки принятия решений предназначена для поддержки многокритериальных решений в сложной информационной среде. При этом под многокритериальностью понимается тот факт, что результаты принимаемых решений оцениваются не по одному, а по совокупности многих показателей (критериев) рассматриваемых одновременно. Информационная сложность определяется необходимостью учета большого объема данных, обработка которых без помощи современной вычислительной техники практически невыполнима. В этих условиях число возможных решений, как правило, весьма велико, и выбор наилучшего из них "на глаз", без всестороннего анализа может приводить к грубым ошибкам.

Основные функции:

- оценивание проблемной ситуации и имеющихся ограничений;

- выявление приоритетов и их ранжирование;

- формулировка предпочтений ЛПР;

- генерирование альтернатив;

- оценивание альтернатив;

- анализ возможных последствий и рисков;

- выбор наилучшей альтернативы.

Для анализа и выработок предложений в СППР используются разные методы.

Методы:

- информационный поиск;

- анализ данных;

- поиск знаний в базах данных;

- имитационное моделирование;

- когнитивное моделирование;

- нейронные сети;

- ситуационное моделирование и д.р.

Некоторые из этих методов были разработаны в рамках искусственного интеллекта. Если в основе работы СППР лежат методы искусственного интеллекта, то говорят об интеллектуальной СППР или ИСППР.

Близкие к СППР классы систем — это экспертные системы и автоматизированные системы управления.

Система позволяет решать задачи оперативного и стратегического управления на основе учетных данных о деятельности компании.

Система поддержки принятия решений представляет собой комплекс программных инструментальных средств для анализа данных, моделирования, прогнозирования и принятия управленческих решений, состоящий из собственных разработок корпорации и приобретаемых программных продуктов (Oracle, IBM, Cognos).

Теоретические исследования в области разработки первых систем поддержки принятия решений проводились в технологическом институте Карнеги в конце 50-х начале 60-х годов XX века. Объединить теорию с практикой удалось специалистам из Массачусетского технологического института в 60-х годах. В середине и конце 80-х годов XX столетия стали появляться такие системы, как EIS, GDSS, ODSS. В 1987 году компания Texas Instruments разработала для United Airlines Gate Assignment Display System. Это позволило значительно снизить убытки от полетов и отрегулировать управление различными аэропортами, начиная от Международного аэропорта O’Hare в Чикаго и заканчивая Stapleton в Денвере, штат Колорадо. В 90-х годах сфера возможностей СППР расширялась благодаря внедрению хранилищ данных и инструментов OLAP. Появление новых технологий отчетности сделало СППР незаменимой в менеджменте.

1.1 Классификации СППР

По взаимодействию с пользователем выделяют три вида СППР:

- пассивные помогают в процессе принятия решений, но не могут выдвинуть конкретного предложения;

- активные непосредственно участвуют в разработке правильного решения;

- кооперативные предполагают взаимодействие СППР с пользователем.

Выдвинутое системой предложение пользователь может доработать, усовершенствовать, а затем отправить обратно в систему для проверки. После этого предложение вновь представляется пользователю, и так до тех пор, пока он не одобрит решение.

По способу поддержки различают:

- модельно-ориентированные СППР, используют в работе доступ к статистическим, финансовым или иным моделям;

- СППР, основанные на коммуникациях, поддерживают работу двух и более пользователей, занимающихся общей задачей;

- СППР, ориентированные на данные, имеют доступ к временным рядам организации. Они используют в работе не только внутренние, но и внешние данные;

- СППР, ориентированные на документы, манипулируют неструктурированной информацией, заключенной в различных электронных форматах;

- СППР, ориентированные на знания, предоставляют специализированные решения проблем, основанные на фактах.

По сфере использования выделяют:

- общесистемные;

- настольные СППР.

Общесистемные работают с большими системами хранения данных и применяются многими пользователями. Настольные являются небольшими системами и подходят для управления с персонального компьютера одного пользователя.

1.2 Архитектура СППР

Функциональные СППР

Являются наиболее простыми с точки зрения архитектуры. Они распространены в организациях, не ставящих перед собой глобальных задач и имеющих невысокий уровень развития информационных технологий. Отличительной особенностью функциональных СППР является то, что анализу подвергаются данные, содержащиеся в файлах операционных систем. Преимуществами подобных СППР являются компактность из-за использования одной платформы и оперативность в связи с отсутствием необходимости перегружать данные в специализированную систему. Из недостатков можно отметить следующие: сужение круга вопросов, решаемых с помощью системы, снижение качества данных из-за отсутствия этапа их очистки, увеличение нагрузки на операционную систему с потенциальной возможностью прекращения ее работы.

СППР, использующие независимые витрины данных.

Применяются в крупных организациях, имеющих несколько подразделений, в том числе отделы информационных технологий. Каждая конкретная витрина данных создается для решения определенных задач и ориентирована на отдельный круг пользователей. Это значительно повышает производительность системы. Внедрение подобных структур достаточно просто. Из отрицательных моментов можно отметить то, что данные многократно вводятся в различные витрины, поэтому могут дублироваться. Это повышает затраты на хранение информации и усложняет процедуру унификации. Наполнение витрин данных достаточно сложно в связи с тем, что приходится использовать многочисленные источники. Отсутствует единая картина бизнеса организации, вследствие того, что нет окончательной консолидации данных.

СППР на основе двухуровневого хранилища данных.

Используется в крупных компаниях, данные которых консолидированы в единую систему. Определения и способы обработки информации в данном случае унифицированы. На обеспечение нормальной работы подобной СППР требуется выделить специализированную команду, которая будет ее обслуживать. Такая архитектура СППР лишена недостатков предыдущей, но в ней нет возможности структурировать данные для отдельных групп пользователей, а также ограничивать доступ к информации. Могут возникнуть трудности с производительностью системы.

СППР на основе трехуровневого хранилища данных.

Такие СППР применяют хранилище данных, из которого формируются витрины данных, используемые группами пользователей, решающих сходные задачи. Таким образом, обеспечивается доступ, как к конкретным структурированным данным, так и к единой консолидированной информации. Наполнение витрин данных упрощается ввиду использования проверенных и очищенных данных, находящихся в едином источнике. Имеется корпоративная модель данных. Такие СППР отличает гарантированная производительность. Но существует избыточность данных, которая ведет к росту требований на их хранение. Кроме того, необходимо согласовать подобную архитектуру с множеством областей, имеющих потенциально различные запросы.

1.3 Структура СППР

Выделяют четыре основных компонента:

- хранилища данных;

- средства обработки данных;

- многомерные базы данных;

- data mining.

1.4 Динамическое моделирование

Особый класс систем стратегического управления и поддержки принятия решений представляют собой системы, позволяющие осуществлять динамическое моделирование процессов. При использовании методов динамического моделирования деятельность компании описывается в виде математической модели, в которой все бизнес-задачи и процессы представляются как система взаимосвязанных вычисляемых показателей.

1.5 Преимущества

СППР позволяет облегчить работу руководителям предприятий и повысить ее эффективность. Они значительно ускоряют решение проблем в бизнесе. СППР способствуют налаживанию межличностного контакта. На их основе можно проводить обучение и подготовку кадров. Данные информационные системы позволяют повысить контроль над деятельностью организации. Наличие четко функционирующей СППР дает большие преимущества по сравнению с конкурирующими структурами. Благодаря предложениям, выдвигаемым СППР, открываются новые подходы к решению повседневных и нестандартных задач.

1.6 Метод анализа иерархий

Метод Анализа Иерархий (МАИ, иногда МетАнИе) — математический инструмент системного подхода к сложным проблемам принятия решений. МАИ не предписывает лицу, принимающему решение (ЛПР), какого-либо «правильного» решения, а позволяет ему в интерактивном режиме найти такой вариант (альтернативу), который наилучшим образом согласуется с его пониманием сути проблемы и требованиями к её решению. Этот метод разработан американским математиком Томасом Саати, который написал о нем книги, разработал программные продукты и в течение 20 лет проводит симпозиумы ISAHP (англ. *International Symposium on Analytic Hierarchy Process*). МАИ широко используется на практике и активно развивается учеными всего мира. В его основе наряду с математикой заложены и психологические аспекты. МАИ позволяет понятным и рациональным образом структурировать сложную проблему принятия решений в виде иерархии, сравнить и выполнить количественную оценку альтернативных вариантов решения. Метод Анализа Иерархий используется во всем мире для принятия решений в разнообразных ситуациях: от управления на межгосударственном уровне до решения отраслевых и частных проблем в бизнесе, промышленности, здравоохранении и образовании. Для компьютерной поддержки МАИ существуют программные продукты, разработанные различными компаниями. Анализ проблемы принятия решений в МАИ начинается с построения иерархической структуры, которая включает цель, критерии, альтернативы и другие рассматриваемые факторы, влияющие на выбор. Эта структура отражает понимание проблемы лицом, принимающим решение. Каждый элемент иерархии может представлять различные аспекты решаемой задачи, причем во внимание могут быть приняты как материальные, так и нематериальные факторы, измеряемые количественные параметры и качественные характеристики, объективные данные и субъективные экспертные оценки. Иными словами, анализ ситуации выбора решения в МАИ напоминает процедуры и методы аргументации, которые используются на интуитивном уровне. Следующим этапом анализа является определение приоритетов, представляющих относительную важность или предпочтительность элементов построенной иерархической структуры, с помощью процедуры парных сравнений. Безразмерные приоритеты позволяют обоснованно сравнивать разнородные факторы, что является отличительной особенностью МАИ. На заключительном этапе анализа выполняется синтез (линейная свертка) приоритетов на иерархии, в результате которой вычисляются приоритеты альтернативных решений относительно главной цели. Лучшей считается альтернатива с максимальным значением приоритета.

1.7 Пример задачи многокритериального выбора с простейшей иерархией

В данной задаче необходимо выбрать из трех кандидатов одного на должность руководителя. Кандидаты оцениваются по критериям: возраст, опыт, образование и личные качества. На рисунке показана иерархия для этой задачи. Простейшая иерархия содержит три уровня: цель, критерии и альтернативы. Числа на рисунке показывают приоритеты элементов иерархии с точки зрения цели, которые вычисляются в МАИ на основе парных сравнений элементов каждого уровня относительно связанных с ними элементами вышерасположенного уровня. Приоритеты альтернатив относительно цели (глобальные приоритеты) вычисляются на заключительном этапе метода путём линейной свертки локальных приоритетов всех элементов. В данном примере лучшим кандидатом является Дик, так как имеет максимальное значение глобального приоритета.

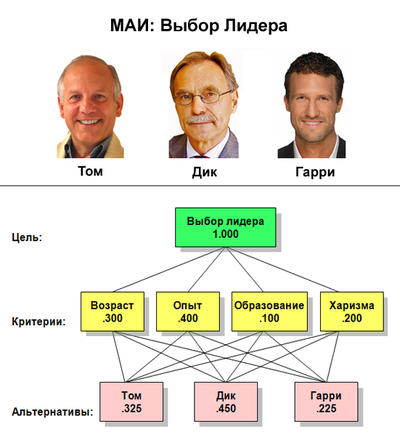


Рисунок 1.7.1 – МАИ. Выбор лидера

1.8 Сфера образования и научных исследований

Хотя для практического применения МАИ отсутствует необходимость специальной подготовки, основы метода преподают во многих учебных заведениях. Кроме того, этот метод широко применяется в сфере управления качеством и читается в рамках многих специализированных программ, таких как Six Sigma, Lean Six Sigma, и QFD. Около ста китайских университетов предлагают курсы по основам МАИ, и многие соискатели научных степеней выбирают МАИ в качестве объекта научных и диссертационных исследований. Опубликовано более 900 научных статей по данной тематике. Существует китайский научный журнал, специализирующийся в области МАИ. Раз в два года проводится Международный симпозиум, посвященный МАИ (International Symposium on Analytic Hierarchy Process, ISAHP), на котором встречаются как ученые, так и практики, работающие с МАИ. В 2007 году симпозиум проходил в Вальпараисо, Чили, где было представлено более 90 докладов ученых из 19 стран, включая США, Германию, Японию, Чили, Малайзию, и Непал.

1.9 Методика применения МАИ

Метод анализа иерархий содержит процедуру синтеза приоритетов, вычисляемых на основе субъективных суждений экспертов. Число суждений может измеряться дюжинами или даже сотнями. Математические вычисления для задач небольшой размерности можно выполнить вручную или с помощью калькулятора, однако гораздо удобнее использовать программное обеспечение (ПО) для ввода и обработки суждений. Самый простой способ компьютерной поддержки — электронные таблицы, самое развитое ПО предусматривает применение специальных устройств для ввода суждений участниками процесса коллективного выбора. Порядок применения Метода Анализа Иерархий:

1. Построение качественной модели проблемы в виде иерархии, включающей цель, альтернативные варианты достижения цели и критерии для оценки качества альтернатив.
2. Определение приоритетов всех элементов иерархии с использованием метода парных сравнений.
3. Синтез глобальных приоритетов альтернатив путём линейной свертки приоритетов элементов на иерархии.
4. Проверка суждений на согласованность.
5. Принятие решения на основе полученных результатов.

1.10 Моделирование проблемы в виде иерархии

Первый шаг МАИ — построение иерархической структуры, объединяющей цель выбора, критерии, альтернативы и другие факторы, влияющие на выбор решения. Построение такой структуры помогает проанализировать все аспекты проблемы и глубже вникнуть в суть задачи.

1.10.1 Определение иерархической структуры

Иерархическая структура — это графическое представление проблемы в виде перевернутого дерева, где каждый элемент, за исключением самого верхнего, зависит от одного или более выше расположенных элементов. Часто в различных организациях распределение полномочий, руководство и эффективные коммуникации между сотрудниками организованы в иерархической форме.

Иерархические структуры используются для лучшего понимания сложной реальности: мы раскладываем исследуемую проблему на составные части; затем разбиваем на составные части получившиеся элементы и т. д. На каждом шаге важно фокусировать внимание на понимании текущего элемента, временно абстрагируясь от всех прочих компонентов. При проведении подобного анализа приходит понимание всей сложности и многогранности исследуемого предмета.

В качестве примера можно привести иерархическую структуру, которая используется при обучении в медицинских вузах. В рамках изучения анатомии отдельно рассматривается костно-мышечная система (которая включает такие элементы, как руки и их составляющие: мышцы и кости), сердечнососудистая система (и её множественные уровни), нервная система (и её компоненты и подсистемы) и т. д. Степень детализации доходит до клеточного и молекулярного уровня. В конце изучения приходит понимание системы организма в целом, а также осознание того, какую роль играет в нем занимает каждая часть. С помощью подобного иерархического структурирования студенты приобретают всесторонние знания об анатомии.

Аналогичным образом, когда мы решаем сложную проблему, мы можем использовать иерархию как инструмент для обработки и восприятия больших объемов информации. По мере проектирования этой структуры у нас формируется все более полное понимание проблемы.

Чтобы избежать беспорядка в диаграммах МАИ, связи, соединяющие Альтернативы и их покрывающие Критерии, часто опускаются, или их количество искусственно уменьшается. Несмотря на такие упрощения в диаграмме, в самой иерархии каждая Альтернатива связана с каждым из покрывающих её Критериев (см. рисунок 1.10.1.1).

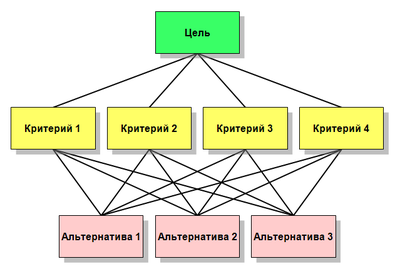


Рисунок 1.10.1.1 – Простейшая иерархия МАИ

1.10.2 Объяснение иерархических структур, используемых в МАИ

Иерархические структуры, используемые в МАИ, представляют собой инструмент для качественного моделирования сложных проблем. Вершиной иерархии является главная цель; элементы нижнего уровня представляют множество вариантов достижения цели (альтернатив); элементы промежуточных уровней соответствуют критериям или факторам, которые связывают цель с альтернативами. Существуют специальные термины для описания иерархической структуры МАИ. Каждый уровень состоит из узлов. Элементы, исходящие из узла, принято называть его детьми (дочерними элементами). Элементы, из которых исходит узел, называются родительскими. Группы элементов, имеющие один и тот же родительский элемент, называются группами сравнения. Родительские элементы Альтернатив, как правило, исходящие из различных групп сравнения, называются покрывающими Критериями. Используя эти термины для описания представленной ниже диаграммы, можно сказать, что четыре Критерия — это дети Цели; в свою очередь, Цель — это родительский элемент для любого из Критериев. Каждая Альтернатива — это дочерний элемент каждого из включающих её Критериев. Всего на диаграмме присутствует две группы сравнения: группа, состоящая из четырех Критериев и группа, включающая три Альтернативы. Вид любой иерархии МАИ будет зависеть не только от объективного характера рассматриваемой проблемы, но и от знаний, суждений, системы ценностей, мнений, желаний и т. п. участников процесса. Опубликованные описания применений МАИ часто включают в себя различные схемы и объяснения представленных иерархий. Последовательное выполнение всех шагов МАИ предусматривает возможность изменения структуры иерархии, с целью включения в неё вновь появившихся, или ранее не считавшихся важными, Критериев и Альтернатив

1.11 Расстановка приоритетов

После построения иерархии участники процесса используют МАИ для определения приоритетов всех узлов структуры. Информация для расстановки приоритетов собирается со всех участников и математически обрабатывается. В данном разделе приведена информация, на простом примере поясняющая процесс вычисления приоритетов.

1.11.1 Определение приоритетов и пояснения

Приоритеты — это числа, которые связаны с узлами иерархии. Они представляют собой относительные веса элементов в каждой группе. Подобно вероятностям, приоритеты — безразмерные величины, которые могут принимать значения от нуля до единицы. Чем больше величина приоритета, тем более значимым является соответствующий ему элемент. Сумма приоритетов элементов, подчиненных одному элементу выше лежащего уровня иерархии, равна единице. Приоритет цели по определению равен 1.0. Рассмотрим простой пример, поясняющий методику вычисления приоритетов.

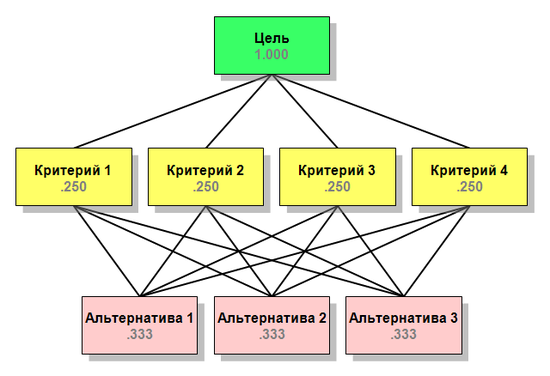


Рисунок 1.11.1.1 – Простейшая иерархическая структура МАИ с приоритетами, определенными по умолчанию

На рисунке показана иерархия, в которой приоритеты всех элементов не устанавливались ЛПР. В таком случае по умолчанию приоритеты элементов считаются одинаковыми, то есть все четыре критерия имеют равную важность с точки зрения цели, а приоритеты всех альтернатив равны по всем критериями. Другими словами, альтернативы в этом примере неразличимы. Заметим, что сумма приоритетов элементов любого уровня, равна единице. Если бы альтернатив было две, то их приоритеты были бы равны 0.500, если бы критериев было 5, то приоритет каждого был бы равен 0.200. В этом простом примере приоритеты альтернатив по разным критериям могут не совпадать, что обычно и бывает на практике. Приведем пример, в котором локальные приоритеты альтернатив по разным критериям не совпадают. Глобальные приоритеты альтернатив относительно цели вычисляются путём умножения локального приоритета каждой альтернативы на приоритет каждого критерия и суммирования по всем критериям.

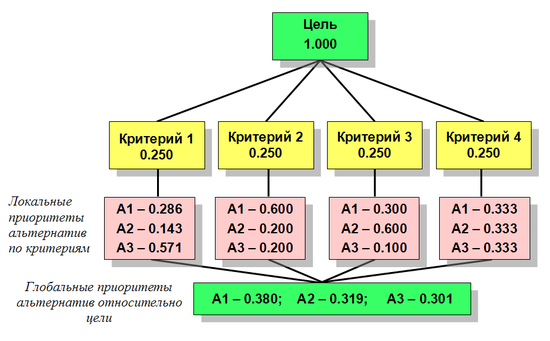


Рисунок 1.11.1.2 – Более сложная иерархическая структура, содержащая глобальные и локальные значения приоритетов по умолчанию

Если приоритеты критериев изменятся, то изменятся значения глобальных приоритетов альтернатив, следовательно, может измениться их порядок. На рисунке 1.11.1.3 показано решение данной задачи с изменившимися значениями приоритетов критериев, при этом наиболее предпочтительной альтернативой становится A3.

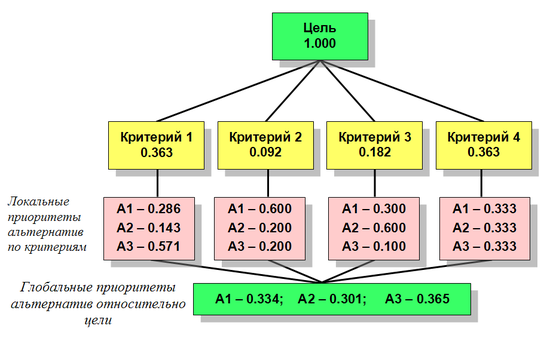


Рисунок 1.11.1.3 – Изменившиеся значения приоритетов критериев

2 Человеко-машинное взаимодействие.

2.1 Высокоуровневый язык программирования Python

В связи с наблюдаемым в настоящее время стремительным развитием персональной вычислительной техники, происходит постепенное изменение требований, предъявляемых к языкам программирования. Все большую роль начинают играть интерпретируемые языки, поскольку возрастающая мощь персональных компьютеров начинает обеспечивать достаточную скорость выполнения интерпретируемых программ. А единственным существенным преимуществом компилируемых языков программирования является создаваемый ими высокоскоростной код. Когда скорость выполнения программы не является критичной величиной, наиболее правильным выбором будет интерпретируемый язык, как более простой и гибкий инструмент программирования.

В связи с этим, определенный интерес представляет рассмотрение сравнительно нового языка программирования Python (пайтон), который был создан его автором Гвидо ван Россумом (Guido van Rossum) в начале 90-х годов.



Рисунок 2.1.1 – Логотип Python в настоящее время

Python (в русском языке распространено название питон) — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Python с использованием JIT-компиляции, которая значительно увеличивает скорость выполнения Python-программ.

Python — активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

2.2 Философия

Разработчики языка Python придерживаются определённой философии программирования, называемой «The Zen of Python» («Дзен Питона», или «Дзен Пайтона»). Её текст выдаётся интерпретатором Python по команде *import this* (работает один раз за сессию). Автором этой философии считается Тим Петерс (Tim Peters).

Текст философии:

|  |  |
| --- | --- |
| Перевод на русский язык. | Оригинальное написание на английском языке. |
| * Красивое лучше, чем уродливое. * Явное лучше, чем неявное. * Простое лучше, чем сложное. * Сложное лучше, чем запутанное. * Плоское лучше, чем вложенное. * Разреженное лучше, чем плотное. * Читаемость имеет значение. * Особые случаи не настолько особые, чтобы нарушать правила. * При этом практичность важнее безупречности. * Ошибки никогда не должны замалчиваться. * Если не замалчиваются явно. * Встретив двусмысленность, отбрось искушение угадать. * Должен существовать один — и, желательно, только один — очевидный способ сделать это. * Хотя он поначалу может быть и не очевиден, если вы не голландец. * Сейчас лучше, чем никогда. * Хотя никогда зачастую лучше, чем прямо сейчас. * Если реализацию сложно объяснить — идея плоха. * Если реализацию легко объяснить — идея, возможно, хороша. * Пространства имён — отличная штука! Будем делать их побольше! | * Beautiful is better than ugly. * Explicit is better than implicit. * Simple is better than complex. * Complex is better than complicated. * Flat is better than nested. * Sparse is better than dense. * Readability counts. * Special cases aren’t special enough to break the rules. * Although practicality beats purity. * Errors should never pass silently. * Unless explicitly silenced. * In the face of ambiguity, refuse the temptation to guess. * There should be one — and preferably only one — obvious way to do it. * Although that way may not be obvious at first unless you’re Dutch. * Now is better than never. * Although never is often better than 'right now'. * If the implementation is hard to explain, it’s a bad idea. * If the implementation is easy to explain, it may be a good idea. * Namespaces are one honking great idea — let’s do more of those! |

2.3 История

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределённой ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале 1991 года Гвидо опубликовал исходный текст в группе новостей alt.sources. С самого начала Python проектировался как объектно-ориентированный язык.

Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще связывают именно со змеёй, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте python.org (до выхода версии 2.5) изображают змеиные головы.

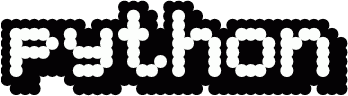


Рисунок 2.3.1 – Логотип Python (1990 – 2005)

Важная цель разработчиков Python — создавать его забавным для использования. Это отражено в его названии, которое пришло из Монти Пайтона. Также это отражено в иногда игривом подходе к обучающим программам и справочным материалам, таким как примеры использования, которые используют понятия спама и яиц вместо стандартных foo и bar.

https://upload.wikimedia.org/wikipedia/ru/8/89/Python-icon.png

Рисунок 2.3.2 – Эмблема Python до выхода версии 2.5

Наличие дружелюбного, отзывчивого сообщества пользователей считается наряду с дизайнерской интуицией Гвидо одним из факторов успеха Python. Развитие языка происходит согласно чётко регламентированному процессу создания, обсуждения, отбора и реализации документов PEP (англ. Python Enhancement Proposal) — предложений по развитию Python.

3 декабря 2008 года, после длительного тестирования, вышла первая версия Python 3000 (или Python 3.0, также используется сокращение Py3k). В Python 3000 устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. На сегодня поддерживаются обе ветви развития (Python 3.x и 2.x).

2.4 Влияние других языков на Python

Появившись сравнительно поздно, Python создавался под влиянием множества языков программирования:

* ABC — отступы для группировки операторов, высокоуровневые структуры данных (map) (Python фактически создавался как попытка исправить ошибки, допущенные при проектировании ABC);
* Modula-3 — пакеты, модули, использование else совместно с try и except, именованные аргументы функций (на это также повлиял Common Lisp);
* С, C++ — некоторые синтаксические конструкции (как пишет сам Гвидо ван Россум — он использовал наиболее непротиворечивые конструкции из С, чтобы не вызвать неприязнь у С-программистов к Python);
* Smalltalk — объектно-ориентированное программирование;
* Lisp — отдельные черты функционального программирования (lambda, map, reduce, filter и другие);
* Fortran — срезы массивов, комплексная арифметика;
* Miranda — списочные выражения;
* Java — модули logging, unittest, threading (часть возможностей оригинального модуля не реализована), xml.sax стандартной библиотеки, совместное использование finally и except при обработке исключений, использование @ для декораторов;
* Icon — генераторы.

Большая часть других возможностей Python (например, байт-компиляция исходного кода) также была реализована ранее в других языках.

2.5 Портируемость

Python портирован и работает почти на всех известных платформах — от КПК до мейнфреймов. Существуют порты под Microsoft Windows, практически все варианты UNIX (включая FreeBSD и Linux), Plan 9, Mac OS и Mac OS X, iPhone OS 2.0 и выше, Palm OS, OS/2, Amiga, HaikuOS, AS/400 и даже OS/390, Windows Mobile, Symbian и Android.

По мере устаревания платформы её поддержка в основной ветви языка прекращается. Например, с серии 2.6 прекращена поддержка Windows 95, Windows 98 и Windows ME. Однако на этих платформах можно использовать предыдущие версии Python — на данный момент сообщество активно поддерживает версии Python начиная от 2.3 (для них выходят исправления).

При этом, в отличие от многих портируемых систем, для всех основных платформ Python имеет поддержку характерных для данной платформы технологий (например, Microsoft COM/DCOM). Более того, существует специальная версия Python для виртуальной машины Java — Jython, что позволяет интерпретатору выполняться на любой системе, поддерживающей Java, при этом классы Java могут непосредственно использоваться из Python и даже быть написанными на Python. Также несколько проектов обеспечивают интеграцию с платформой Microsoft .NET, основные из которых — IronPython и Python.Net.

2.6 Типы и структуры данных

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». В Python имеются встроенные типы: булевый, строка, Unicode-строка, целое число произвольной точности, число с плавающей запятой, комплексное число и некоторые другие. Из коллекций в Python встроены: список, кортеж (неизменяемый список), словарь, множество и другие[25]. Все значения являются объектами, в том числе функции, методы, модули, классы.

Добавить новый тип можно либо написав класс (class), либо определив новый тип в модуле расширения (например, написанном на языке C). Система классов поддерживает наследование (одиночное и множественное) и метапрограммирование. Возможно наследование от большинства встроенных типов и типов расширений.

Все объекты делятся на ссылочные и атомарные. К атомарным относятся int, long (в версии 3 любое число int, так как в версии 3 нет ограничения на размер), complex и некоторые другие. При присваивании атомарных объектов копируется их значение, в то время как для ссылочных копируется только указатель на объект, таким образом, обе переменные после присваивания используют одно и то же значение. Ссылочные объекты бывают изменяемые и неизменяемые. Например, строки и кортежи являются неизменяемыми, а списки, словари и многие другие объекты — изменяемыми. Кортеж в Python является, по сути, неизменяемым списком. Во многих случаях кортежи работают быстрее списков, поэтому если вы не планируете изменять последовательность, то лучше использовать именно их.

2.7 Синтаксис и семантика

Язык обладает чётким и последовательным синтаксисом, продуманной модульностью и масштабируемостью, благодаря чему исходный код написанных на Python программ легко читаем. При передаче аргументов в функции Python использует вызов по соиспользованию (call-by-sharing)[27].

Операторы.

Набор операторов достаточно традиционен.

* Условный оператор if (если). Альтернативный блок после else (иначе). Если условий и альтернатив несколько, можно использовать elif (сокр. от else if).
* Операторы цикла while (пока) и for (для). Внутри цикла возможно применение break и continue для прерывания цикла и перехода сразу к следующей итерации, соответственно.
* Оператор определения класса class.
* Оператор определения функции, метода или генератора def. Внутри возможно применение return (возврат) для возврата из функции или метода, а в случае генератора — yield (давать).
* Оператор обработки исключений try — except — else или try — finally (начиная с версии 2.5, можно использовать finally, except и else в одном блоке).
* Оператор pass ничего не делает. Используется для пустых блоков кода.

Одной из интересных синтаксических особенностей языка является выделение блоков кода с помощью отступов (пробелов или табуляций), поэтому в Python отсутствуют операторные скобки begin/end, как в языке Паскаль, или фигурные скобки, как в Си. Такой «трюк» позволяет сократить количество строк и символов в программе и приучает к «хорошему» стилю программирования. С другой стороны, поведение и даже корректность программы может зависеть от начальных пробелов в тексте. Некоторым такое поведение может показаться неинтуитивным и неудобным.

Выражения.

Выражение является полноправным оператором в Python. Состав, синтаксис, ассоциативность и приоритет операций достаточно привычны для языков программирования и призваны минимизировать употребление скобок.

Отдельно стоит упомянуть операцию форматирования для строк (работает по аналогии с printf() из Си), которая использует тот же символ, что и взятие остатка от деления:

>>> print ("Здравствуй, %s!" % "Мир")

Здравствуй, Мир!

Python имеет удобные цепочечные сравнения. Такие условия в программах — не редкость:

1 <= a < 10 and 1 <= b < 20

Кроме того, логические операции (or и and) являются ленивыми: если для вычисления значения операции достаточно первого операнда, этот операнд и является результатом, в противном случае вычисляется второй операнд логической операции. Это основывается на свойствах алгебры логики: например, если один аргумент операции «ИЛИ» (or) является истиной, то и результат этой операции всегда является истиной. В случае, если второй операнд является сложным выражением, это позволяет сократить издержки на его вычисление. Этот факт широко использовался до версии 2.5 вместо условной конструкции:

a < b and "меньше" or "больше или равно"

Встроенные типы данных, как правило, имеют особый синтаксис для своих литералов (записанных в исходном коде констант):

# для версии меньше 3

"строка"

'строка'

"""тоже строка"""

u"Юникод-строка"

True or False # булевы литералы

3.14 # число с плавающей запятой

0o12 + 0xA # числа в восьмеричной и шестнадцатеричной системах счисления

1 + 2j # комплексное число

[1, 2, "a"] # список

(1, 2, "a") # кортеж

{'a': 1, 'b': 'B'} # словарь

{'a', 6, 8.8} # множество

lambda x: x\*\*2 # анонимная функция

# для версии 3

"строка и Юникод-строка одновременно"

'строка и Юникод-строка одновременно'

"""тоже строка и Юникод-строка одновременно"""

True or False # булевы литералы

3.14 # число с плавающей запятой

0b1010 + 0o12 + 0xA # числа в двоичной, восьмеричной и шестнадцатеричной системах счисления

1 + 2j # комплексное число

[1, 2, "a"] # список

(1, 2, "a") # кортеж

{'a': 1, 'b': 'B'} # словарь

{'a', 6, 8.8} # множество

lambda x: x\*\*2 # анонимная функция

Для списков (и других последовательностей) Python предлагает набор операций над срезами. Особенностью является индексация, которая может показаться новичку странной, но раскрывает свою согласованность по мере использования. Индексы элементов списка начинаются с нуля. Запись среза s[N:M] означает, что в срез попадают все элементы от N включительно до M не включая. В качестве иллюстрации можно посмотреть пример работы с последовательностями.

Имена.

Имя (идентификатор) может начинаться с латинской буквы (В Python 3 — буквы любого алфавита в Юникоде, например кириллицы) любого регистра или подчёркивания, после чего в имени можно использовать и цифры. В качестве имени нельзя использовать ключевые слова (их список можно узнать по import keyword; print(keyword.kwlist)) и нежелательно переопределять встроенные имена. Имена, начинающиеся с символа подчёркивания, имеют специальное значение.

В каждой точке программы интерпретатор имеет доступ к трём пространствам имён (то есть отображениям имён в объекты): локальному, глобальному и встроенному.

Области видимости имён могут быть вложенными друг в друга (внутри определяемой функции видны имена из окружающего блока кода). На практике с областями видимости и связыванием имён связано несколько правил «хорошего тона», о которых можно подробнее узнать из документации.

Строки документации.

Python предлагает механизм документирования кода pydoc. В начало каждого модуля, класса, функции вставляется строка документации — docstring (англ.). Строки документации остаются в коде на момент времени исполнения, и в язык встроен доступ к документации(переменная \_\_doc\_\_), что используется современными IDE (например, Eclipse).

В интерактивном режиме можно получить помощь, сгенерировать гипертекстовую документацию по целому модулю или даже применить doctest (англ.) для автоматического тестирования модуля.

Директивы.

Начиная с Python 2.3, для использования в тексте программы символов, не входящих в ASCII, необходимо явно указывать кодировку исходного кода в начале модуля, например:

# -\*- coding: utf-8 -\*-

# или

# coding: utf-8

После этого можно использовать, например, кириллицу в Unicode-литералах. Но на самом деле, даже если вы напишете:

# coding: utf

То Python «поймёт», что вы хотели сделать.

2.8 Функциональные возможности

С точки зрения функциональных возможностей Python можно назвать гибридом. Его инструментальные средства укладываются в диапазон между традиционными языками сценариев (такими как Tcl, Scheme и Perl) и языками разработки программных систем (такими как C, C++ и Java). Python обеспечивает простоту и непринужденность языка сценариев и мощь, которую обычно можно найти в компилирующих языках. Превышая возможности других языков сценариев, такая комбинация делает Python удобным средством разработки крупномасштабных проектов. Ниже приводится список основных возможностей, которые есть в арсенале Python:

*Динамическая типизация.*

Python сам следит за типами объектов, используемых в программе, благодаря чему не требуется писать длинные и сложные объявления в программном коде. В действительности, в языке Python вообще отсутствуют понятие типа и необходимость объявления переменных. Так как программный код на языке Python не стеснен рамками типов данных, он автоматически может обрабатывать целый диапазон объектов.

*Автоматическое управление памятью.*

Python автоматически распределяет память под объекты и освобождает ее ("сборка мусора"), когда объекты становятся ненужными. Большинство объектов могут увеличивать и уменьшать занимаемый объем памяти по мере необходимости.

*Модульное программирование.*

Для создания крупных систем Python предоставляет такие возможности, как модули, классы и исключения. Они позволяют разбить систему на составляющие, применять ООП для создания программного кода многократного пользования и элегантно обрабатывать возникающие события и ошибки.

*Встроенные типы объектов.*

Python предоставляет наиболее типичные структуры данных, такие как списки, словари и строки, в виде особенностей, присущих самому языку программирования. Эти типы отличаются высокой гибкостью и удобством. Например, встроенные объекты могут расширяться и сжиматься по мере необходимости, могут комбинироваться друг с другом для представления данных со сложной структурой.

*Встроенные инструменты.*

Для работы со всеми этими типами объектов в составе Python имеются мощные и стандартные средства, включая такие операции, как конкатенация (объединение коллекций), получение срезов (извлечение части коллекции), сортировка, отображение и многое другое.

*Библиотеки утилит.*

Для выполнения более узких задач в состав Python также входит большая коллекция библиотечных инструментов, которые поддерживают практически все, что только может потребоваться, – от поиска с использованием регулярных выражений до работы в сети. Библиотечные инструменты языка Python – это то место, где выполняется большая часть операций.

*Утилиты сторонних разработчиков.*

Python – это открытый программный продукт и поэтому разработчики могут создавать свои предварительно скомпилированные инструменты поддержки задач, решить которые внутренними средствами невозможно.

2.9 Прикладное применение Python

Python используется не только отдельными пользователями, он также применяется компаниями для создания продуктов, приносящих настоящую прибыль. Например:

* Компания Google широко использует Python в своей поисковой системе и оплачивает труд создателя Python.
* Служба коллективного использования видеоматериалов YouTube в значительной степени реализована на языке Python.
* Популярная программа BitTorrent для обмена файлами в пиринговых сетях (peer-to-peer) написана на языке Python.
* Популярный веб-фреймворк App Engine от компании Google использует Python в качестве прикладного языка программирования.
* Такие компании, как EVE Online и Massively Multiplayer Online Game (MMOG), широко используют Python в своих разработках.
* Мощная система трехмерного моделирования и создания мультипликации Maya поддерживает интерфейс для управления из сценариев на языке Python.
* Такие компании, как Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM, используют Python для тестирования аппаратного обеспечения.
* Такие компании, как Industrial Light & Magic, Pixar и другие, используют Python в производстве анимационных фильмов.
* NASA, Los Alamos, Fermilab, JPL и другие используют Python для научных вычислений.
* iRobot использует Python в разработке коммерческих роботизированных устройств.
* ESRI использует Python в качестве инструмента настройки своих популярных геоинформационных программных продуктов под нужды конечного пользователя.
* NSA использует Python для шифрования и анализа разведданных.
* В реализации почтового сервера IronProt используется более 1 миллиона строк программного кода на языке Python.
* Проект "ноутбук каждому ребенку" (One Laptop Per Child, OLPC) строит свой пользовательский интерфейс и модель функционирования на языке Python.

Универсальная природа языка обеспечивает возможность его применения в самых разных областях. Фактически с определенной долей уверенности можно утверждать, что Python так или иначе используется практически каждой достаточно крупной организацией, занимающейся разработкой программного обеспечения, – как для решения краткосрочных тактических задач, так и для разработки долгосрочных стратегических проектов.

2.10 Преимущество Python перед другими языками высокого уровня

* Имеет более широкие возможности, чем Tcl. Язык Python поддерживает "программирование в целом", что делает его применимым для разработки крупных систем.
* Имеет более четкий синтаксис и более простую архитектуру, чем Perl, что делает программный код более удобочитаемым, простым в сопровождении и снижает вероятность появления ошибок.
* Проще и удобнее, чем Java. Python – это язык сценариев, а Java унаследовала сложный синтаксис от таких языков программирования, как C++.
* Проще и удобнее, чем C++, но нередко он не может конкурировать с C++, поскольку, будучи языком сценариев, Python предназначен для решения другого круга задач.
* Более мощный и более переносимый, чем Visual Basic. Открытая природа Python также означает, что нет какой-то отдельной компании, которая его контролирует.
* Более удобочитаемый и более универсальный, чем PHP. Иногда Python используется для создания веб-сайтов, но он способен решать гораздо более широкий круг задач, от управления роботами до создания анимационных фильмов.
* Более зрелый и имеет более ясный синтаксис, чем Ruby. В отличие от Ruby и Java, объектно-ориентированный стиль программирования является необязательным в Python – он не вынуждает использовать ООП в проектах, где этот стиль неприменим.

Обладает динамическими особенностями таких языков, как SmallTalk и Lisp, но имеет более простой и традиционный синтаксис, доступный как для разработчиков, так и для конечных пользователей настраиваемых систем.

2.11 Реализации Python

CPython является основной, но не единственной реализацией языка программирования Python. Существуют также следующие реализации:

* Jython — реализация Python, использующая JVM в качестве среды исполнения. Позволяет прозрачно использовать Java-библиотеки.
* PyS60 — реализация языка для смартфонов фирмы Nokia на платформе Series 60.
* IronPython — Python для .NET Framework и Mono. Компилирует Python программы в MSIL, таким образом предоставляя полную интеграцию с .NET-системой.
* Stackless — также написанная на Си реализация Python. Это не полноценная реализация, а патчи к CPython. Предоставляет расширенные возможности многопоточного программирования и значительно большую глубину рекурсии.
* Python for .NET — ещё одна реализация Python для .NET. В отличие от IronPython эта реализация не компилирует Python код в MSIL, а только предоставляет интерпретатор, написанный на C#. Позволяет использовать .NET-сборки из Python кода.
* PyPy — реализация Python, написанная на Python. Позволяет легко проверять новые возможности. В PyPy кроме стандартного CPython включены возможности Stackless, Psyco. В проект интегрированы возможности анализа Python кода и трансляция в другие языки и байткоды виртуальных машин.
* python-safethread — версия CPython без GIL, что позволяет одновременно исполнять Python потоки на всех доступных процессорах. Внесены также некоторые другие изменения.
* Unladen Swallow — начатый Google проект по разработке высокоэффективного, максимально совместимого с CPython JIT-компилятора, а на базе LLVM. Согласно планам по развитию Python, планировалось перенести исходный код Unladen Swallow в CPython в версии 3.3. Но PEP-3146 был отменён в связи с отсутствием интереса к Unladen Swallow со стороны Google, основного спонсора разработки.
* tinypy — минималистическая версия Python. Часть возможностей CPython не реализована.
* Micro Python — эффективная реализация Python 3 для встроенных систем с малым объёмом оперативной памяти.

2.12 Недостатки

Низкое быстродействие.

Классический Python, как и многие другие интерпретируемые языки, не применяющие, например, JIT-компиляторы, имеют общий недостаток — сравнительно невысокую скорость выполнения программ. Сохранение байт-кода позволяет интерпретатору не тратить лишнее время на перекомпиляцию кода модулей при каждом запуске, в отличие, например, от языка Perl. Кроме того, существует специальная JIT-библиотека psyco, позволяющая ускорить в выполнение программ (однако приводящая к увеличению потребления оперативной памяти). Эффективность psyco сильно зависит от архитектуры программы.

Существуют реализации языка Python, вводящие высокопроизводительные виртуальные машины (ВМ) в качестве бэк-энда компилятора. Примерами таких реализаций может служить PyPy, базирующийся на LLVM; более ранней инициативой является проект Parrot. Ожидается, что использование ВМ типа LLVM приведёт к тем же результатам, что и использование аналогичных подходов для реализаций языка Java, где низкая вычислительная производительность в основном преодолена.

Множество программ/библиотек для интеграции с другими языками программирования предоставляют возможность использовать другой язык для написания критических участков.

Невозможность модификации встроенных классов.

По сравнению с Ruby и некоторыми другими языками, в Python отсутствует возможность модифицировать встроенные классы, такие, как int, str, float, list и другие, что, однако, позволяет Python потреблять меньше оперативной памяти и быстрее работать. Ещё одной причиной введения такого ограничения является необходимость согласования с модулями расширения. Многие модули (в целях оптимизации быстродействия) преобразуют Python-объекты элементарных типов к соответствующим Си-типам вместо манипуляций с ними посредством Си-API. Также это избавляет от многих потенциальных ошибок при неконтролируемом динамическом переопределении встроенных типов

1. Основые причины выбора мессенджера Telegram
2. Скорость работы.

В telegram скорость отправки сообщений моментальная, даже при низкой скорости интернета. Все передается за секунды, да и сервера падают крайне редко. Это очень большой плюс для данного мессенджера.



Рисунок 3.1 – Что можно делать в telegram.

1. Дизайн.

Он очень эстетичный, все подогнано под один стиль, ни что не выделяется. По канонам Material Design, быстро и в минимальное количество кликов. Так же плюсом является размер приложения, он меньше чем у Whatsapp, Viber and etc.



Рисунок 3.2 – Скриншот дизайна мессенджера

1. Всеядность.

Абсолютно везде применим данный мессенджер – Android, iOS, Windows, Mac OS и даже Web-клиент. Причем работает одинаково хорошо во всех платформах.

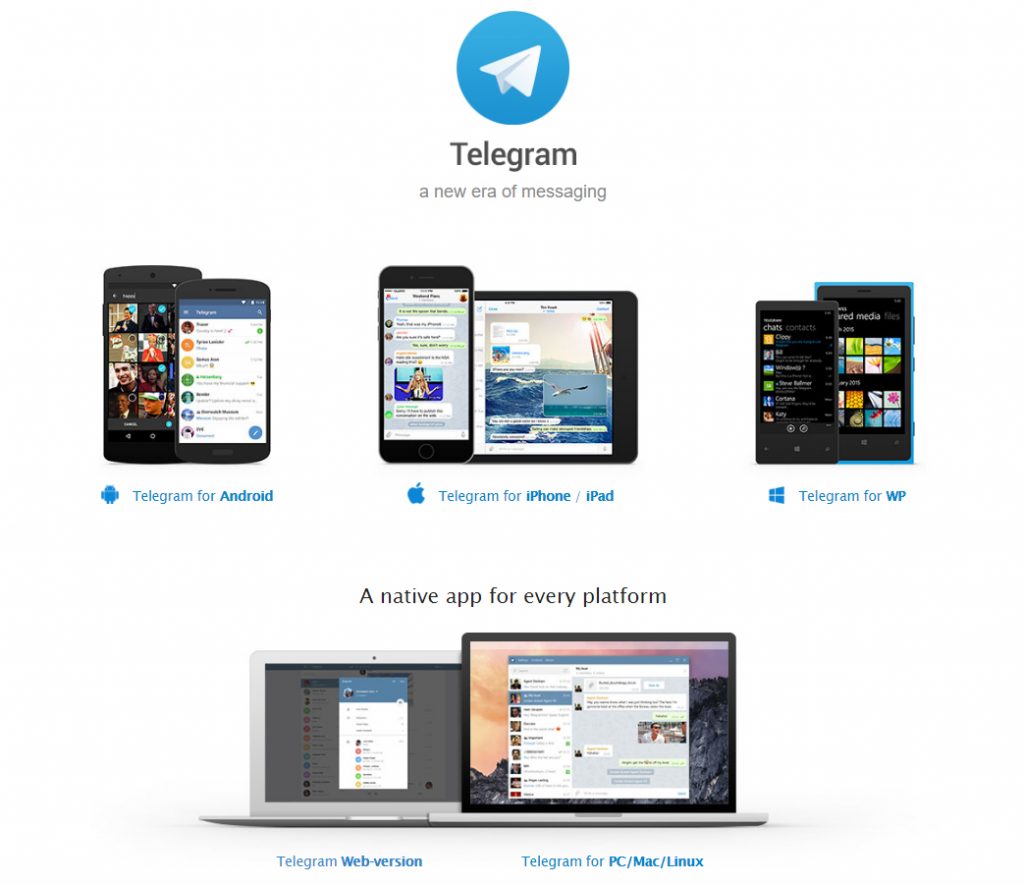


Рисунок 3.3 – Пример запуска на разных платформах

1. Облачность.

Все переписки, фото, видео, хранятся в облаке. Все данные сжаты(урезаны), но они не потеряются. Это конечно не ново, но реализовано лучше чем у WhatsApp, где вас спрашивают сохранять или нет. В данном случае, здесь всё реализовано по стандарту и ничего не пропадёт.

1. Безопасность.

Секретные чаты с авто удалением, пароли на вход и двухэтапная авторизация. То есть, он уже более защищен чем остальные, и это может стать решающим фактором для многих Но работает все как часы, пароль ни как не обойти, двухэтапная авторизация при попытке взлома.



Рисунок 3.4 – Безопасность и анонимность

1. Боты, каналы и стикеры.

Возможность создания бота с помощью клиента мессенджера, которого можно использовать по своему усмотрению. Создание большого количества каналов для общения и делиться информацией в виде массовой рассылки. Ещё огромное количество разнообразных стикеров, которые доступны всем бесплатно, а так же можно добавлять свои стикеры в коллекцию.

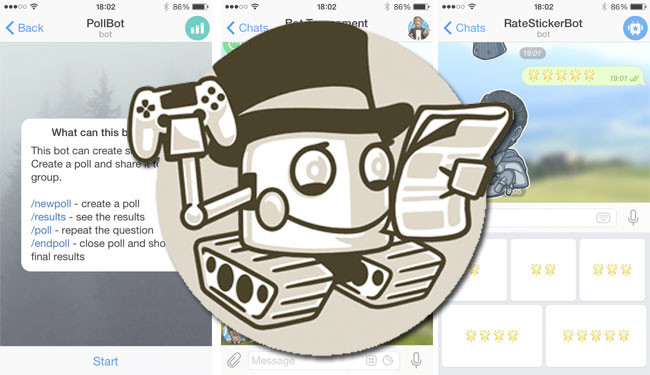


Рисунок 3.5 – Создание бота в telegram

Для реализации цели данной магистерской работы, является возможность создания бота в мессенджере telegram.

3.1 Telegram (мессенджер)



Рисунок 3.1.1 – Логотип telegram

Telegram — бесплатный кроссплатформенный мессенджер для смартфонов и других устройств, позволяющий обмениваться текстовыми сообщениями и медиафайлами различных форматов. Используются проприетарная серверная часть c закрытым кодом, работающая на мощностях нескольких компаний США и Германии, финансируемых Павлом Дуровым в объёме порядка 12 млн долларов США ежегодно, и несколько клиентов с открытым исходным кодом, в том числе под GNU GPL.

Количество активных пользователей сервиса на февраль 2016 года составляло более 100 млн человек, а количество ежедневно пересылаемых сообщений достигло 10 миллиардов на август 2015.

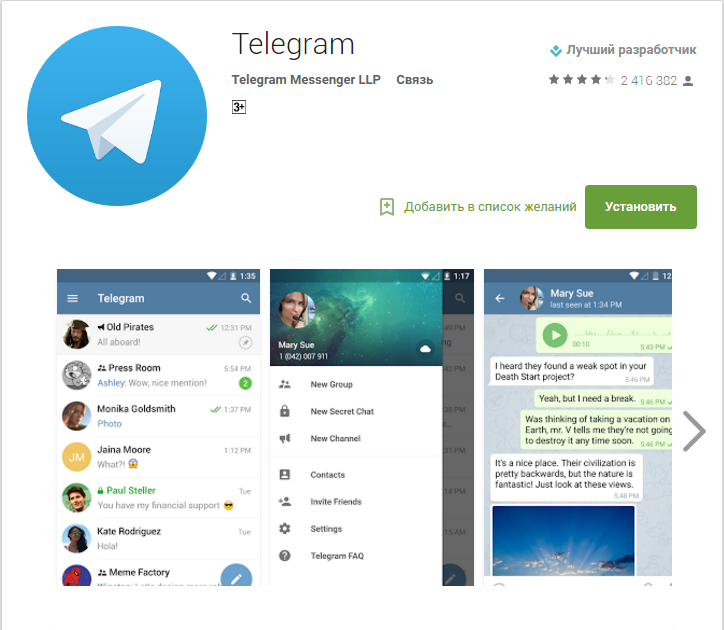


Рисунок 3.1.2 – Скриншот Telegram для Android

3.2 История

Проект создан Павлом Дуровым, основателем социальной сети «ВКонтакте». В интервью The New York Times Павел рассказал, что первоначальная идея приложения пришла ему ещё в 2011-м, когда к его двери приходили спецназовцы. Когда последние всё-таки ушли, Дуров сразу же написал своему брату Николаю. Тогда же он и осознал, что у него нет безопасного способа коммуникации с братом. Сервис построен на технологии шифрования переписки MTProto, разработанной братом Павла — Николаем. Сам «Телеграм» изначально был экспериментом принадлежащей Павлу компании Digital Fortress с целью протестировать MTProto на больших нагрузках.

* 14 августа 2013 года представлен первый клиент Telegram для устройств на платформе iOS.
* 22 августа 2013 года один из участников конкурса Durov’s Android Challenge написал и выложил в открытый доступ первое приложение для операционной системы Android, совместимое с Telegram, (использует тот же протокол MTProto).
* В октябре у проекта открылся веб-сайт и была представлена официальная версия Telegram под Android с открытым исходным кодом (GPL2). Предыдущая версия программы доступна под названием «Unofficial Telegram S».
* 7 ноября 2013 года появились сторонние клиенты сервиса для Windows и OS X с ограниченным функционалом. Также был разработан концепт веб-версии клиента.
* В ноябре у программы насчитывалось, по данным «TJournal», около 1 миллиона установок.
* В январе 2014 года вышла неофициальная веб-версия Webogram от бывшего разработчика ВКонтакте Игоря Жукова.
* 21 июля 2014 года в App Store появилось приложение Telegram HD для iPhone и iPad, которое загрузила компания Telegram Messenger LLP.
* Новое приложение получило специальную версию для Apple iPad, улучшило поддержку видео и фотографий высокого разрешения, добавило возможность пересылки анимированных изображений в формате gif. На официальном сайте мессенджера в качестве клиента для iOS указано именно это приложение.
* 15 октября 2014 года в Telegram была добавлена поддержка псевдонимов, по которым возможно связываться с пользователями, даже не зная их телефонного номера, а также запущен веб-клиент.
* 2 января 2015 года в Telegram была добавлена поддержка стикеров. Изначально в приложении 14 стикеров, но любой пользователь может модифицировать их или добавить свои собственные. В отличие от многих приложений, в Telegram стикеры полностью бесплатные.
* В феврале 2016 года один из создателей Telegram Павел Дуров заявил, что мессенджером пользуются уже более 100 миллионов человек, при этом сервис доставляет около 15 миллиардов сообщений ежедневно. Еще в сентябре 2015 года Telegram передавал 12 миллиардов посланий в день.
* В апреле 2016 года стало известно, что в мае 2015 года корпорация Google рассматривала возможность покупки мессенджера за более чем 1 млрд долларов США.
* В мае 2016 года появилась возможность редактирования отправленных сообщений. Внести изменения возможно в течение двух суток с момента отправки. В этом случае в сообщении появится специальная метка.
* 22 ноября 2016 года разработчиками был запущен проект Telegraph — блог-платформа, бесплатный издательский инструмент, который позволяет создавать публикации, обзоры, вставлять фотографии и все виды embedded-кода. Telegraph — гибрид блог-платформы, мессенджера и платишера (подобного Medium), с концепцией анонимных имиджборд. Проект Павла Дурова, основателя социальной сети «ВКонтакте»
* 3 января 2017 года один из разработчиков добавил возможность удаления своих отправленных сообщений. После удаления отправителем сообщения, собеседник не сможет увидеть удаленное сообщение.
* В марте 2017 года В. Д. Соловей со ссылкой на анонимный источник сообщил, что российские спецслужбы получили доступ к сообщениям пользователей и их архиву за три года. Павел Дуров назвал это заявление уткой.
* 15 мая 2017 года стало известно о том, что декстопная версия Telegram получила возможность совершать звонки.
* 16 мая 2017 года администрация Telegram заявила, что не будет предоставлять информацию российским госорганам.

3.3 Технология

Для мессенджера был создан протокол MTProto, предполагающий использование нескольких протоколов шифрования. При авторизации и аутентификации используются алгоритмы RSA-2048, DH-2048 для шифрования, при передаче сообщений протокола в сеть они шифруются AES с ключом, известным клиенту и серверу. Также применяются криптографические хеш-алгоритмы SHA-1 и MD5.

Безопасность от перехвата пересылаемых сообщений со стороны сервера Telegram обеспечивается лишь в режиме «секретных» чатов (Secret Chats), доступном с 8 октября 2013 года. Этот режим реализует шифрование, при котором лишь отправитель и получатель обладают общим ключом (end-to-end шифрование), с применением алгоритма AES-256 в режиме IGE (англ. Infinite Garble Extension) для пересылаемых сообщений. В отличие от обычного режима, сообщения в секретных чатах не расшифровываются сервером, история переписки сохраняется лишь на тех двух устройствах, на которых был создан чат.

При обмене файлами можно как отправить файлы с устройства, так и искать медиаконтент в интернете, в том случае, если используется мобильная версия для iOS или Android. Размер передаваемых файлов ограничен 1,5 Гб. Программа использует систему докачки файлов после обрыва связи.

Имеется возможность организовывать мультичаты до 200 участников, начиная с ноября 2015 года, супергруппы до 1000 участников, с 14 марта 2016 — супергруппы до 5000 участников.

3.4 Боты

При помощи специального API сторонние разработчики могут создавать «ботов», специальные аккаунты, управляемые программами. Типичные боты отвечают на специальные команды в персональных и групповых чатах, также они могут осуществлять поиск в интернете или выполнять иные задачи, применяются в развлекательных целях или в бизнесе.

В сентябре 2015 года Павел Дуров заявил о скором появлении возможностей монетизации и размещения рекламы в ботах.

3.5 Конкурсы поиска уязвимостей

В декабре 2013 года Павел Дуров объявил конкурс до 1 марта 2014 года на «взлом» защиты Telegram с призовым фондом в 200 тысяч долларов. Условиями конкурса ставилась задача расшифровать личную переписку Павла с его братом Николаем через «секретные чаты», используя зашифрованные данные, которыми обменивались приложения и сервер. В их сообщениях, отправляемых ежедневно, содержался секретный адрес электронной почты, расшифровка которого позволила бы получить приз.

Требуемая для такого «взлома» модель атаки, атака на основе шифротекста, является самой слабой и, в то же время, наиболее сложной и неудобной для криптоаналитика. Существуют чрезвычайно слабые алгоритмы, которые могут быть устойчивыми в данной модели, но уязвимыми для других методов. Обычно при анализе новых криптографических алгоритмов используются более сильные модели атаки, в которых атакующему может быть известен текст до шифрования, дана возможность отправлять на шифрование любые тексты или возможность изменения данных, пересылаемых по сети. Таким образом, в случае, если конкурс никто не выиграет, это не докажет криптографическую безопасность протокола.

23 декабря 2013 года, всего через несколько дней после начала конкурса, пользователь «Хабрахабра», не являющийся экспертом в криптографии, обнаружил уязвимость, заключавшуюся в том, что клиент получал параметры для генерации ключей DH (константы для определения поля вычетов) от сервера без проверки, благодаря чему проприетарный сервер MTProto мог передать некорректные параметры, не обеспечивающие криптографическую стойкость, и скрытно провести MITM-атаку на секретные чаты. Так как ему не удалось прочитать переписку, размер выигрыша составил только 100 тысяч долларов. После этого клиент был обновлён, в нём была добавлена проверка параметров, получаемых от сервера, с целью значительно уменьшить вероятность подобной атаки.

В ноябре 2014 года был организован новый трёхмесячный конкурс, в котором модель атаки была расширена, у атакующего появилась возможность выступать в качестве сервера MTProto, изменяя пересылаемые данные. По условиям конкурса требуется взломать «секретный чат», при этом участники чата проводят сверку ключей, согласованных при открытии чата, по независимым каналам связи.

По оценкам исследователя Moxie Marlinspike и других подобные конкурсы не могут доказать безопасность шифрования и лишь вводят в заблуждение. Отсутствие выигравших не означает безопасности продукта, многие такие конкурсы в целом нечестны, анализ не контролируется и проводится случайными людьми, а вознаграждения зачастую слишком малы, чтобы оправдать многолетнюю работу нескольких компетентных криптоаналитиков.

3.6 Критика

Учётные записи пользователей привязываются к телефонным номерам, что является одним из самых существенных аргументов критиков Telegram, поскольку это не обеспечивает полной анонимности при общении. При регистрации в сервисе и последующих авторизациях новых устройств, производится проверка телефонного номера через отправку SMS-сообщения с кодом (на некоторых ОС — перехватывается приложением) или телефонный вызов.

Основатель WhatsApp Ян Кум указал в комментарии для Cossa.ru, что идеи, реализованные в его приложении, используются в Telegram.

Серверы Telegram не сохраняют сообщения из секретных чатов, но сохраняют историю обычных чатов и содержимое адресной книги пользователей на срок использования сервиса и на срок неактивности, указанный в настройках аккаунта (от одного месяца до года). Используемое в мессенджере шифрование не обеспечивает PFS во всех случаях.

Официальные клиенты Telegram по умолчанию активно рассылают всем контактам метаинформацию об открытии и закрытии приложения, при этом подписаться на эту метаинформацию может любой пользователь. Для отключения подобной рассылки требуется изменить настройки аккаунта.

Также неоднократно высказывались сомнения в безопасности протокола MTProto.

Имеются сообщения, что мессенджером могут пользоваться различные террористические группы как для общения, так и для пропаганды. В частности, террористическая группировка ИГ (ИГИЛ) использовала Телеграм для распространения своих заявлений более чем 14 тысячам подписчиков в более чем 30 каналах на различных языках. Однако команда Telegram активно занимается поиском и дальнейшей блокировкой подобных каналов.

3.7 Цензура

Телеграм выборочно применял цензуру. В частности, некоторое время мессенджер использовался в Иране для распространения порнографии и сатирических комментариев о правительстве. Руководство Telegram ограничило деятельность некоторых ботов и запретила определённые наборы изображений-«стикеров» по запросу правительства Ирана. В то же время чаты Телеграма не подвергались цензуре. В октябре 2015 года Дуров заявил, что компания Telegram Messenger LLP отказалась помогать Ирану в слежке за гражданами и в цензуре, из-за чего работа приложения некоторое время блокировалась.

Мессенджер блокировался властями в некоторых регионах Китая, где он мог использоваться для координации антиправительственных выступлений.

4 Моделирование системы поддержки принятия решений при обращении к боту мессенджеру?

Порядок выполнения действий бота мессенджера при запросе пользователя:

1. Определение корректности вопроса;
2. Обработка данного запроса;
3. Вывод результата пользователю.

Алгоритм может быть составлен по блочной схеме или по логической схеме. Блочная схема алгоритма представляет собой графическое построение операций и логических условий. Простейшие алгоритмы применялись с давних времен в математике. Смысл этого понимается как точное предписание выполнения в заданном порядке некоторой системы операции для решения задач определенного типа.

Под алгоритмом необходимо понимать последовательность математических, логических, технологических или вместе взятых операций, отличающихся детерминированностью, массовостью, направленностью и приводящих к решению всех задач данного класса за конечное число шагов.

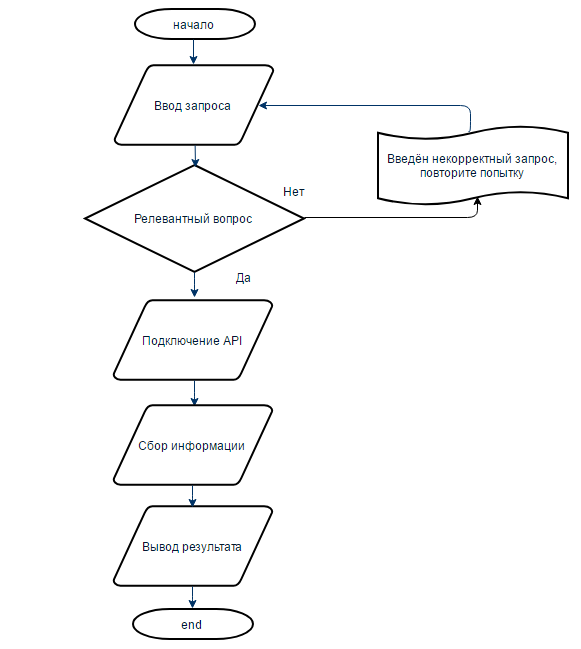


Рисунок 4.1 – Блок-схема алгоритма

Источники

<http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D0%B8_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9#.D0.9F.D1.80.D0.B5.D0.B8.D0.BC.D1.83.D1.89.D0.B5.D1.81.D1.82.D0.B2.D0.B0>

<http://bourabai.ru/tpoi/dss.htm#1>

<http://elar.urfu.ru/bitstream/10995/1676/5/1335843_schoolbook.pdf>

<http://pro-spo.ru/erp/1816-dss>

<https://www.youtube.com/watch?v=dDwTKNsrv54>

<http://mirznanii.com/a/111739/sozdanie-sistem-podderzhki-prinyatiya-resheniy>

<http://www.treko.ru/show_article_763>

<https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0_%D0%B8%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D0%B9>

<https://ru.wikipedia.org/wiki/Python>

<https://pythonworld.ru>

<https://habrahabr.ru/post/31180/>

<https://www.ibm.com/developerworks/ru/library/l-python_part_1/>

<https://ru.wikipedia.org/wiki/Telegram_(мессенджер)>

<http://telegram-free.ru/kak-iskat-kanaly-v-telegram.html>

<https://telegram-store.com/blog/anonymous-telegram/>

<https://telegram.org>

<https://keddr.com/2016/07/6-prichin-pochemu-telegram-top/>

<http://telegram-online.ru>