МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЖЕНИЕ ПЕРЕСНЯНСКАЯ СРЕДНЯЯ ШКОЛА

Индивидуальный проект

по теме: ” Программирование на языке Python

на примере создания архива по творчеству Николая Лескова “

Выполнил проект:

Ученик 10 класса

Хакризоев Дмитрий

Учитель:

Яковлева Анна Александровна

Учитель-консультант:

Миренкова С.А.

Пересна

2021

**ОГЛАВЛЕНИЕ**

Введение

Основная часть

1.Теоретическая часть

1.1. Понятие о языках программирования

2. Практическая часть

Заключение

Список используемой литературы

Приложение

**Дима оглавление поправь по пунктам в тексте (сделал) нет, не сделал, я тебе пример привела, как и Дим, приложение введение и заключение и список литературы не нумеруется**

**ВВЕДЕНИЕ**

Рубеж XX и XXI столетий ознаменовался не только развитием биотехнологий и микроэлектроники, вступлением США и стран западной Европы в постиндустриальный этап развития, но и массовым распространением персональных компьютеров. Это позволило человечеству взглянуть на информацию с другой стороны, с позиции двоичного кода. С информатизацией общества количество информации, которое потребляется человеком в день, многократно возросло.

Преподавание информатики в школах нашей страны фактически начиналось с преподавания программирования. В то время даже был провозглашен лозунг: «Программирование - это вторая грамотность». Заметим, что компьютеры в школах тогда практически отсутствовали. А та техника, которой оснащались школы в конце 80-х - начале 90-х годов прошлого века, практическую составляющую курса информатики все равно невольно сводила к программированию.

Одновременно с революционным развитием аппаратного и программного обеспечения и оснащением современной компьютерной техникой учебных заведений курс информатики претерпел существенные изменения. Наиболее яркая характеристика такого изменения - вымывание программирования из школьного курса информатики. Основное внимание в большинстве школ стало уделяться освоению современных информационных технологий. Эти тенденции отражены и в новом «Стандарте по информатике», в котором собственно обучению программированию отводится очень мало времени. Но, как заметил А.Г. Гейн, «…очевидно, что именно алгоритмизация с самого начала вытянула на школьную арену курс информатики и ныне во многих реально существующих курсах информатики позволяет уйти от умных, но пустоватых разговоров к конкретному делу».

 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, который был создан его автором Гвидо ван Россумом в начале 90-х годов. Python – популярный язык программирования, используемый как для разработки самостоятельных программ, так и для создания прикладных сценариев в самых разных областях применения. Это мощный, переносимый, простой в использовании и свободно распространяемый язык. Программисты, работающие в самых разных областях, считают, что ориентация Python на эффективность разработки и высокое качество программного обеспечения дает ему стратегическое преимущество, как в маленьких, так и в крупных проектах.

Такие известные компании, как Google и Intel, Cisco и Hewlett-Packard, используют язык Python, выбрав его за гибкость, простоту использования и обеспечиваемую им высокую скорость разработки. Он позволяет создавать эффективные и надежные проекты, которые легко интегрируются с программами и инструментами, написанными на других языках.
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.

**Цель проекта**: создание архива по творчеству Н. С. Лескова на языке программирования Python

**Задачи проекта:**

1. Сбор и ознакомление с информацией о Н. С. Лескове и его творчестве
2. Анализ и систематизация полученной информации
3. Написание алгоритма создания программы
4. Создание программы на языке программирования Python

Моя работа содержит введение, основную часть: теоретический материал и практическая часть, заключение, список используемой литературы и электронное приложение.

**1. Теоретическая часть**

* 1. **Понятие о языках программирования**

Язык программирования — формальный язык, предназначенный для записи компьютерных программ.

Фактически, язык программирования, это набор правил описания команд и заранее созданных функций.

  Сегодня практически все программы создаются с помощью языков программирования. Теоретически программу можно написать и на естественном языке (говорят: программирование на метаязыке), но из-за неоднозначности естественного языка автоматически перевести такую программу в машинный код пока невозможно.

 *Языки программирования –* это формальные искусственные языки. Как и естественные языки, они имеют *алфавит, словарный запас, грамматику* и *синтаксис,* а также *семантику.*

*Алфавит –* разрешенный к использованию набор символов, с помощью которого могут быть образованы слова и величины данного языка.

*Синтаксис –* система правил, определяющих допустимые конструкции языка программирования из букв алфавита.

*Семантика –* система правил однозначного толкования каждой языковой конструкции, позволяющих производить процесс обработки данных.

Взаимодействие синтаксических и семантических правил определяет основные *понятия языка,* такие как *операторы, идентификаторы, константы, переменные, функции, процедуры* и т.д. В отличие от естественных, язык программирования имеет ограниченный запас слов (операторов) и строгие правила их написания, а правила грамматики и семантики, как и для любого формального языка, явно однозначно и четко сформулированы.

Языки программирования, ориентированные на команды процессора и учитывающие его особенности, называют языками *низког*о *уровня.* «Низкий уровень» не означает неразвитый, имеется в виду, что операторы этого языка близки к машинному коду и ориентированы на конкретные команды процессора.

Языком самого низкого уровня является *ассемблер.* Программа, написанная на нем, представляет последовательность команд машинных кодов, но записанных с помощью *символьных мнемоник.* С помощью языков низкого уровня создаются компактные оптимальные программы, так как программист получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется хорошо понимать устройство компьютера, а использование такой программы на компьютере с процессором другого типа невозможно. Такие языки программирования используются для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнее компактность, быстродействие, прямой доступ к аппаратным ресурсам.

Языки программирования, имитирующие естественные, обладающие укрупненными командами, ориентированные «на человека», называют языками *высокого уровня.* Чем выше уровень языка, тем ближе структуры данных и конструкции, использующиеся в программе, к понятиям исходной задачи. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому исходные тексты программ легко переносимы на другие платформы, имеющие трансляторы этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, число ошибок, допускаемых в процессе программирования, намного меньше. В настоящее время насчитывается несколько сотен таких языков (без учета их диалектов).

Таким образом, языки программирования высокого уровня, ориентированные на решение больших содержательных прикладных задач, являются аппаратно-независимыми и требуют использования соответствующих программ-переводчиков для преобразования текста программы в машинный код, который в итоге и обрабатывается процессором.

**1.2. Компиляторы и интерпретаторы**

С помощью языка программирования создается текст программы, описывающий разработанный алгоритм. Чтобы программа была выполнена, надо либо весь ее текст перевести в машинный код (это действие и выполняет программа – *компилятор)* и затем передать на исполнение процессору, либо сразу выполнять команды языка, переводя на машинный язык и исполняя каждую команду поочередно (этим занимаются программы – *интерпретаторы).*

*Интерпретатор* функционирует следующим образом: берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет. После успешного выполнения текущей команды интерпретатор переходит к анализу и исполнению следующей. Если один и тот же оператор в программе выполняется несколько раз, интерпретатор всякий раз воспринимает его так, будто встретил впервые. Поэтому программы, в которых требуется произвести большой объем повторяющихся вычислений, будут работать медленно. Для выполнения программы на другом компьютере также необходимо установить интерпретатор, так как без него программа представляет собой набор слов, и работать не может.

*Компиляторы* полностью обрабатывают весь текст программы (его называют *исходным кодом* или source code). Они осуществляют поиск синтаксических ошибок, выполняют семантический анализ и только затем, если текст программы в точности соответствует правилам языка, его автоматически переводят (транслируют) на машинный язык (говорят: генерируют *объектный код* или object code). Нередко при этом выполняется оптимизация с помощью набора методов, позволяющих повысить быстродействие программы. Сгенерированный объектный код обрабатывается специальной программой *сборщиком* или *редактором связей,* который производит связывание объектного и машинного кодов. Текст программы преобразуется в готовый к исполнению ЕХЕ-файл *(исполнимый код),* его можно сохранить в памяти компьютера или на диске. Этот файл имеет самостоятельное значение, и может работать под управлением операционной системы. Его можно перенести на другие компьютеры с процессором, поддерживающим соответствующий машинный код.

Основной недостаток компиляторов – трудоемкость трансляции языков программирования, ориентированных на обработку данных сложной структуры, заранее неизвестной или динамически меняющейся во время работы программы. Для таких программ в машинный код вводятся дополнительные проверки и анализ наличия ресурсов операционной системы, средства динамического захвата и освобождения памяти компьютера, что на уровне статически заданных машинных инструкций осуществить достаточно сложно, а для которых задач практически невозможно.

С помощью интерпретатора, наоборот, для исследования содержимого памяти допустимо в любой момент прервать работу программы, организовать диалог с пользователем, выполнить любые сложные преобразования данных и при этом постоянно контролировать программно-аппаратную среду, что и обеспечивает высокую надежность работы программы. Интерпретатор при выполнении каждой команды подвергает проверке и анализу необходимые ресурсы операционной системы, при возникающих проблемах выдает сообщения об ошибках. В реальных системах программирования смешаны технологии компиляции и интерпретации. В процессе отладки программу можно выполнять по шагам (трассировать), а результирующий код не обязательно будет машинным, он может быть, например, аппаратно-независимым промежуточным кодом абстрактного процессора, который в дальнейшем будет транслироваться в различных компьютерных архитектурах с помощью интерпретатора или компилятора в соответствующий машинный код.

Процесс создания программы включает (Рис. 1):

* Составление исходного кода программы на языке программирования.
* Этап трансляции, необходимый для создания объектного кода программы
* Построение загрузочного модуля, готового к исполнению.

Исходный код

Объектный код

Загрузочный модуль

трансляция

редактор

связей

Рис. 1 Процесс создание программы, готовой к исполнению

Все перечисленные выше действия требуют наличия специальных программных средств.

Совокупность этих программных средств входит в состав *системы программирования:*

* Текстовый редактор (необходимый для создания и редактирования исходного кода программы на языке программирования).
* Компилятор.
* Редактор связей.
* Отладчик.
* Библиотеки функций.
* Справочная система.

**1.3. Классификация и обзор языков программирования**

Современное состояние языков программирования можно представить в виде следующей классификации (Рис. 2).

Языки программирования

Процедурные

(императивные)

Объектно-ориентированные

Декларативные

оперативные

структурные

объектные

визуальные

функциональные

логические

Рис. 2 Классификация языков программирования

Сначала нужно узнать общие виды языков программирования и их назначение. Все они подразделяются на две категории:

* процедурные;
* непроцедурные.

Процедурная (алгоритмическая) программа — это система формальных предписаний, направленных на решение конкретных задач, которые выполняет ЭВМ.

Непроцедурное программирование представляет собой прямо противоположную методологию (парадигму) разработки, когда компьютеру ставится определённая задача в более или менее общем виде, без написания формализованного алгоритма, который отдаётся на усмотрение машины.

Процедурные языки отличаются тем, на кого в первую очередь направлены: на машину или человека. Они подразделяются на две категории:

* низкого уровня (или машинно-ориентированные);
* высокого уровня.

Низкоуровневые языки ориентированы на конкретные компьютерные архитектуры и учитывают их особенности. Они являются следующим уровнем после машинного кода и следуют конкретным указаниям, исходящим от процессора. Работать с ними тяжело, но созданные с их помощью программы (обычно это системные программы и драйверы) занимают меньше места в памяти и работают быстрее.

Высокоуровневые языки более понятны для человека. Соответственно, программировать на них гораздо проще и удобнее. В них не учитываются особенности конкретных типов процессоров, поэтому такие программы легко переносятся с одной платформы на другую. Для этого используют специальные программы-трансляторы.

Непроцедурные языки включают две основные языковые группы:

* объектно-ориентированные;
* декларативные.

Объектно-ориентированные состоят из ряда независимых объектов, которые функционируют как отдельные компьютеры. С помощью этих блоков можно решать задачи, не вникая во «внутреннюю кухню» их работы.

Работа с декларативным языком подразумевает установление взаимосвязей между исходными информационными структурами и свойствами конечного результата. При этом в нём не существует понятия «команда», а программист не создаёт алгоритмы.

Декларативные языки подразделяются на два семейства:

* логические;
* функциональные.

Логическое программирование описывает проблемы в виде фактов и формул, а система решает их посредством механизмов логического вывода.

Функциональное, в свою очередь, формулирует задачу как совокупность определённых функций.

Теперь от общей классификации можно перейти к обзору ключевых современных языков программирования, а именно самых популярных и любимых профессионалами. Их всего десять в списке: C++, Python, Visual Basic, PHP, Delphi, Java, JavaScript, Ruby, ActionScript, Nemerle. У каждого из них есть свои характерные особенности, недостатки и преимущества, которые можно увидеть в описаниях ниже.

 **С++** (Си-плюс-плюс) С++ появился очень давно, но до сих пор пользуется большой популярностью у профессионалов, которые считают владение им практически обязательным. Изучать его как первый язык программирования не нужно, но приобретение профессионализма характеризуется тем, что человек начинает использовать С++ на постоянной основе.

Со времени создания он прошёл несколько процедур стандартизации и обновления, поэтому сохраняет актуальность и по сей день. виды языков программирования Главным его плюсом является то, что он полностью универсален.

На С++ можно писать всё что угодно, именно это и обеспечило ему любовь профессионалов. Кроме того, он относительно прост в освоении — тому, кто уже изучил С, Python или Java, будет нетрудно разобраться и в С++.

Верно и обратное, так что С++ часто представляют как своего рода универсальную формулу и некий ключик к миру программирования вообще. В то же время критики отмечают неудобный синтаксис и слишком длинный, громоздкий программный код, который часто появляется в результате работы на С++. Однако эта проблема уже отчасти исправлена с помощью дополнительных шаблонов. Существуют разные точки зрения на этот счёт, но факт в том, что С++ пользуется неослабевающей популярностью на протяжении почти 30 лет.

**Python** (Пайтон или Питон) Python — весьма популярный в наше время язык, созданный с целью как можно более простого написания сложных программ. Он был образован на основе ранних разработок и впитал в себя все их достижения. При этом постоянно выходят новые обновления, с каждым из которых он становится всё совершеннее.

К основным плюсам Python относятся простота и многофункциональность. Однако за многофункциональностью скрывается низкая скорость исполнения, а за простотой — невнятный системный код, зачастую содержащий множество ошибок. Но, несмотря на это, у Python сейчас достаточно почитателей.

**Visual Basic** (Вижуал Бейсик) Вряд ли найдётся тот, кто не слышал об этом продукте от Microsoft, ведь многие программисты именно на нём познавали азы работы с кодом. Visual Basic - простой, многофункциональный и подходящий для быстрого прототипирования язык программирования . языки программирования и их назначение VB даёт широкие возможности по созданию программного кода, а также позволяет разрабатывать UI (пользовательский интерфейс) программ. Именно его чаще всего используют специалисты Microsoft для создания таблиц с данными.

Впрочем, не обошлось и без недостатков, таких как отсутствие комментариев и доступ к памяти компьютера на низком уровне.

Кроме того, у этой программы невысокая скорость работы, использовать её можно только на ОС Windows и macOS.

**PHP** (Пи-эйч-пи) Расшифровывается как Personal Home Page Tools («инструменты для создания личной странички в интернете»). Чаще всего PHP применяют веб-мастера для разработки сайтов.

В настоящее время он наиболее популярен в среде хостов (интернет-провайдеров), которые следуют тенденции использовать и поддерживать PHP, в чём и заключается его огромное преимущество. Но одним интернетом его функциональность не ограничивается.

PHP завоевал большую симпатию у сайтостроителей благодаря многоплатформенности(подходит для любой операционной системы), быстроте работы, простоте и удобству. Кроме того, с его помощью можно создавать GUI (графический пользовательский интерфейс).

К его недостаткам относятся несовместимость разных версий и невозможность создания в нём многопотоковых программ, усиливающих производительность компьютера.

**Delphi** (Делфи) Delphi — императивный объектно-ориентированный язык, разработанный на основе высокоуровневого Object Pascal. современные языки программирования.

Он широко распространён благодаря многофункциональности, а также множеству компиляторов и диалектов, среди которых программист найдёт именно то, что подходит под его конкретные задачи.

С помощью Delphi написано много программ, включая плееры, файловые менеджеры, мессенджеры и многое другое. Диалектное богатство одновременно является и достоинством, и недостатком. Дело в том, что программа, написанная на одном из его видов, скорее всего, просто не откроется на другой версии.

**Java** (Джава) Java известен своей мультиплатформенностью и тем, что находится в составе большинства современных операционных систем, так как работа многих приложений без него будет недостаточно результативной или вообще невозможной.

Практически каждый пользователь сталкивался с необходимостью установить или обновить Java-модуль. К подводным камням Java относится медлительность написанных на нём программ и их «прожорливость» (то есть они задействуют оперативную память в большом объёме). Зато у Java есть несколько модификаций, которые характеризуются собственными библиотеками данных и структурой, заточенной под специальные задачи.

**JavaScript**  (Джава-скрипт) Этот язык прост, многофункционален и уже давно стал неотъемлемым атрибутом большинства современных интернет-браузеров. Обычно его используют для придания сайтам большей интерактивности, то есть возможности пользователя взаимодействовать с контентом.

Двоичный код JavaScript применяется, где только можно — это браузеры, плагины, прикладное программное обеспечение, офисные и серверные приложения. Дополнительную популярность ему обеспечили специальные высокоабстрактные библиотеки.

Однако JavaScript довольно небезопасен, а написанные на нём приложения обычно пестрят ошибками. Любой пользователь интернета с ними сталкивался, причем не раз. Но стереть его из истории пока что нельзя, ведь браузеры без него попросту перестанут работать.

**Ruby** (Руби) Кроссплатформенный и по-настоящему универсальный скриптовый язык, относящийся к сфере объектно-ориентированного программирования. Его синтаксис прост и лаконичен, благодаря чему новичкам будет несложно его освоить. Профессионалы любят его за бережливость по отношению к компьютерным ресурсам.

Ruby часто хвалят за динамизм и хорошую сбалансированность, где красота не принесена в жертву удобству и наоборот. Кроме того, он абсолютно открыт для использования, изменения, копирования и распространения, а ещё у него довольно много разных библиотек, которые обновляются одна за другой и позволяют решать разные задачи.

**ActionScript** (Экшн-скрипт) ActionScript — это мультимедийный скриптовый язык программирования, который предназначен для написания объектно-ориентированных программ, а именно Flash-приложений. Написание программы Его заточенность под конкретные задачи является одновременно и плюсом, и минусом, так как в пределах Flash он отличается многофункциональностью, но ничего другого на нём написать практически невозможно.

Кроме того, его характеризует гибкость, позволяющая решать задачи разными и порой нестандартными способами, но иногда такой код трудно понять кому-либо, кроме самого автора.

Однако даже и он через время может запутаться в своём коде, а ведь это чревато трудностями с отладкой, обновлениями и повторным использованием.

**Nemerle** (Немерле) Это высокоуровневый гибридный язык программирования, в котором сочетаются элементы функционального и объектно-ориентированного программирования. Он появился не так давно и вобрал в себя лучшие наработки языков-предшественников, но при этом дал и реальный вклад в программирование — возможность создания метапрограмм. Этот метод заключается в выработке алгоритмов, которые в процессе своей работы создают возможные сценарии для дальнейшего образования исполняемых файлов.

Проще говоря, в Nemerle соединились лучшие и самые полезные черты предыдущих языковых поколений. Это пример языка, который совмещает в себе новизну и универсальность.

1. **Практическая часть**

Для выполнения своего проекта я выбрал среду программирования Python. На это есть несколько причин:

1) Простота в освоении и доступность языка. На Python очень легко начать программировать, язык обладает исключительно простым синтаксисом, что облегчает ознакомление с программированием.

2) Свободность и открытость.

Python – пример свободного и открытого программного обеспечения – FLOSS (Free/Libre and Open Source Software). Проще говоря, вы имеете право свободно распространять копии этого программного обеспечения, читать его исходные тексты, вносить изменения, а также использовать его части в своих программах. В основе свободного ПО лежит идея сообщества, которое делится своими знаниями. Это одна из причин, по которым Python – так хорош: он был создан и постоянно улучшается сообществом, которое просто хочет сделать его лучше.

3) Обилие обучающих материалов. О программировании на языке Python написано множество книг (например, “A byte of a Python” за авторством Swaroop C H)

**Python** - программа-интерпретатор для языка программирования высокого уровня **Python**, имеющего несложный синтаксис и позволяющего реализовать множество задач. Разработчики скриптового языка подчеркивают высокую производительность, легкую читаемость исходного кода. Основными чертами языка является автоматическое управление памятью, динамическая типизация данных, поддержка многопоточных вычислений, удобных высокоуровневых структур. Python успешно поддерживает все основные парадигмы программирования, основные модели: объектно-ориентированную, функциональную, структурную, императивную, аспектно-ориентированную. Код в самом языке организован посредством различных функций, классов, объединенных в определенные модули. Он предлагает достаточно мощную поддержку для взаимодействия с другими языками, инструментами. Поставляется с обширным набором стандартных библиотек. Python имеет версию на Java., .NET.

## Особенности программы:

- Имеет довольно простой и читаемый синтаксис.

- Отлично поддерживает интуитивную ориентацию объектов.

- Может организовывать различные модули, которые организуются в пакеты.

- Позволяет быстро обрабатывать ошибки.

- Имеет огромную библиотеку со стандартными функциями.

- Поддерживает интеграцию сторонних модулей для решения представленных задач.

- Успешно встраивается в любое приложение в виде скрипта.

- Имеет довольно сильный потенциал интроспекции.

- Поддерживает натуральное выражение кода для различных процедур.

## Недостатки программы:

- Отсутствует портативная (portable) версия.

- Отсутствует русский язык.

## Системные требования:

- Процессор с тактовой частотой 800 MHz или более мощный.

- Оперативная память 256 Мб или больше.

- Свободное место на жёстком диске от 77 Мб.

- Архитектура с разрядностью 32 бит или 64 бит (x86 или x64).

- Операционная система Windows XP, Windows Vista, Windows 7, Windows 8.

* 1. **Постановка информационной задачи**

Создание архива по творчеству Николая Лескова путем создания автоматизированной информационной системы, используя среду программирования Python.

**Информационная задача:** создание архива по творчеству Николая Лескова на языке программирования Python.

* 1. **Ход работы**

Для создания архива я изучил, структурировал и представил в наглядной форме информацию о Николае Семеновиче Лескове. Для написания программы я ознакомился со средой программирования Python посредством чтения специализированной литературы (‘Укус Питона’ за авторством Swaroop Chitlur) мне сильно в этом помог. Также я отрабатывал практические навыки, используя курсы Яндекс.Практикум и ‘Поколение Python’ на образовательной платформе Stepik.

После изучения специализированной литературы и поиска информации:

* 1. Создал структуру проекта (Рис. 4).

2. Для решения поставленной задачи написал алгоритм (Рис. 5).

3. Написал программу на языке программирования Python (Приложение).

4. Произвел тестирование программы.

5. Исправил ошибки и недочеты программы.

6. Разработал системные требования и инструкцию для пользователя.



Рис. 4. Структура проекта

Рис. 5 Алгоритм

Если биография

начало

нет

да

Вводим число 2

Вводим число 1

Если детство

Если начало пути

Если детство

нет

нет

да

да

Важные произведения

Личная жизнь

Вводим число 1

Вводим число 1

нет

да

да

выйти

выйти

нет

Вводим число 2

выйти

Вводим число 3

Вводим число 2

Вводим число 3

да

конец

да

выйти

нет

да

Вводим число 1

Если «Некуда»

конец

да

выйти

конец

да

конец

выйти

конец

да

Если «На ножах»

да

выйти

конец

да

нет

выйти

конец

да

Вводим число 2

Если «Соборяне

нет

да

Вводим число 3

Вводим число 4

выйти

выйти

да

да

конец

конец

**Системные требования**

Для работы программы необходимы следующие минимальные системные требования:

1. Операционная система: Windows 7 или новее, 64-битная macOS 10.10+ или Linux, включая Ubuntu, RedHat, CentOS 6+ и другие.
2. Архитектура системы:
Windows - 64-битная x86, 32-битная x86;
MacOS - 64-битная x86;
Linux - 64-битная x86, 64-битная Power8 / Power9.
3. Оперативная память: от 256 Мб и выше (RAM: 3+ GB)
4. Процессор с тактовой частотой 800 MHz или более мощный (CPU: 2+ cores)
5. Программное обеспечение: Архиватор (WinRar/7-zip)

**Инструкция для пользователя**

1. Извлечь архив
2. Запустить файл Nikolai\_Leskov\_archive.exe
3. Основным средством навигации в программе является совершаемые пользователем выбор. (Например: да-1, нет-2; пользователь должен осуществлять выбор с помощью ввода соответствующих цифр на клавиатуре и подтверждения выбора нажатием клавиши Enter.)
4. Для выхода из программы введите слово ‘exit‘ и нажмите Enter

**ЗАКЛЮЧЕНИЕ**

«Чтение хорошей программы на Python очень напоминает чтение английского текста, хоть и достаточно строго!... псевдо-кодовая природа Python является одной из его самых сильных сторон. Она позволяет вам сосредоточиться на решении задач, а не на самом языке».

Целью данной работы было изучение возможностей языка Python, его особенностей и создания архива по творчеству Николая Лескова путем создания автоматизированной информационной системы, используя среду программирования Python.

Для достижения данной цели были изучены и проанализированы источники информации о языке Python, как литературные, так и электронные. На основании полученной информации были рассмотрены характерные особенности и функциональные возможности языка программирования Python, было проведено сравнение с другими языками программирования, изучены практические основы его реализации.

Вот лишь некоторые из большого ряда возможностей python:

− поддержка работы с xml/html файлами; − поддержка работы с http запросами;

− наличие GUI (графического интерфейса);

− создание веб-сценариев;

− поддержка работы с FTP;

− возможность обработки изображений, аудио и видео файлов;

− управление робототехникой;

− программирование в сфере математических и научных вычислений. Python изначально относится к объектно-ориентированным языкам программирования. В его объектной модели поддерживаются такие понятия ООП, как полиморфизм, перегрузка операторов и множественное наследование, но, тем не менее, с учетом простоты синтаксиса и типизации, присущих Python, использование ООП не вызывает больших сложностей в применении. Еще одним преимуществом Python является то, что он может распространяться и использоваться свободно и совершенно бесплатно.

Так же, как и для других открытых программных продуктов, таких как Tel, Perl, Linux и Apache, в Интернете можно свободно найти и получить полные исходные коды реализации Python. Не существует никаких ограничений на осуществление его копирования, встраивания в свои системы или для распространения в составе своих продуктов даже с коммерческой целью.

Безусловно, как и любой язык программирования, Python имеет некоторые свои недостатки. Но преимущества, которые он предоставляет при создании программного обеспечения, гораздо существеннее и не оказывают особого влияния на его функциональность и практичность. В случае же критичности некоторых недостатков при решении некоторых специфических задач, универсальность и гибкость Python позволяет обходить эти недостатки без ущерба для решаемой задачи.

Таким образом, можно с большой уверенностью утверждать, что Рython подходит для решения подавляющего большинства повседневных задач, будь то подключение к сети интернет, чтение-отправка электронной почты, резервное копирование, либо же какая-нибудь игрушка. Язык программирования Python практически не имеет никаких ограничений или запретов на использование, поэтому также может свободно использоваться при создании крупных проектов. К примеру, Рython интенсивно применяется многими IT-гигантами, такими как, например, Google и Yandex. А универсальность и простота и Рython делают его одним из лидеров среди языков программирования как для профессионалов, так и для тех, кто только начинает пробовать свои силы в программировании.

**СПИСОК ЛИТЕРАТУРЫ**

1. А. Н. Чаплыгин. Учимся программировать вместе с Питоном. Учебник. – ревизия 226. – 135 с.

 2. Бизли, Д. Python. Подробный справочник, 4-е издание/ Д. Бизли - М.: Символ-Плюс, 2010. - 326 с.

3. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.

4. И. А. Хахаев. Практикум по алгоритмизации и программированию на Python. Учебник. – М.: Альт Линукс, 2010. – 126 с.

5. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

6. МакГрат, М. Программирование на Python для начинающих / М. МакГрат. - М.: Эксмо, 2015. - 192 c.

7. Ноа Гифт, Джереми М. Джонс. Python в системном администрировании UNIX и Linux. – Перевод с английского. – СПб.: Символ-Плюс, 2009. – 512 с.

8. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.

9. С. Шапошникова. Основы программирования на Python. Учебник. Вводный курс. – версия 2. – 2011. – 44 с.

10. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - М.: Символ, 2016. - 608 c.

11. Сузи Р. А. Язык программирования Python: Учебное пособие. – М.: ИНТУИТ, БИНОМ. Лаборатория знаний, 2006. – 328 с.

12. Форсье, Дж. Django. Разработка веб-приложений на Python./ Дж. Форсье, П. Биссекс, У. Чан, - СПб.: Символ-Плюс, 2010. - 456 с.

13. Самоучитель Python. [Электронный ресурс] - URL: https://pythonworld.ru/samouchitel-python (дата обращения: 16.06.2017).

14. Welcome to Python.org – 2017. [Электронный ресурс] - URL: https://www.python.org/ (дата обращения: 12.06.2017).

15. Python – Википедия. [Электронный ресурс] - URL: https://ru.wikipedia.org/wiki/Python (дата обращения: 20.06.2017).

16. Электронный ресурс [<https://it.wikireading.ru/1000005435>]

1. Электронный ресурс URL: [https://history.wikireading.ru/337629]
2. Электронный ресурс URL: [<https://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.02.pdf>]
3. Электронный ресурс URL: [<https://nauka.club/informatika/yazyki-programmirovaniya.html>]

**Приложение**

Текст программы

def texter(filename):
 with open(filename, encoding='utf-8', mode='r') as file:
 data = file.readlines()
 print(\*data, sep='\n')
def biography(): #функция выбора нужной информации в биографии писателя
 answer = int(input('вам интересно детство, личная жизнь или образование и карьера писателя? (детство - 1, личная жизнь - 2, образование и карьера - 3) '))
 if answer == 1:
 texter('детство.txt')
 loop()
 elif answer == 2:
 texter('личная жизнь.txt')
 loop()
 elif answer == 3:
 texter('образование и карьера.txt')
 loop()
 else:
 print('Неизвестный запрос')
 loop()
def oeuevre(): #функция выбора нужной информации в творчестве писателя
 answer = int(input('вам интересно начало писательского пути, ключевые произведения, наследие и память о писателе? (начало писательского пути - 1, важнейшие произведения - 2, наследие и память - 3)'))
 if answer == 1:
 texter('начало писательского пути.txt')
 elif answer == 2:
 texter('важнейшие произведения.txt')
 book\_choice()
 elif answer == 3:
 texter('наследие и память.txt')
 loop()
 else:
 print('Неизвестный запрос')
 loop()
def oeuevre\_biography(): #главный выбор: биография или творчество писателя
 answer = int(input('Хотели бы вы узнать биографию писателя или перейти к его творчеству? (1-творчество, 2-биография)'))
 if answer==1:
 oeuevre()
 elif answer==2:
 biography()
def origin(): #вывод текста приветствия и запуск функции выбора интересующей информации
 print('Добро пожаловать в архив творчества Николая Лескова')
 oeuevre\_biography()
def loop(): #цикл программы после работы основного кода: перезапуск или завершение
 answer = int(input('Вы бы хотели вернуться в начало или закрыть архив?(1-вернуться в начало,2-закрыть архив)'))
 if answer==1:
 origin()
 elif answer==2:
 print('Благодарим за Ваш визит!')
 exit
def book\_choice(): #выбор произведения для ознакомления
 answer = int(input('''С каким произведением вы желаете ознакомиться?:
1 - "Некуда"
2 - "На ножах"
3 - "Соборяне"
4 - "Левша" '''))
 if answer == 1:
 print('https://ru.wikipedia.org/wiki/Некуда\_(роман)')
 elif answer == 2:
 print('https://ru.wikipedia.org/wiki/На\_ножах')
 elif answer == 3:
 print('https://ru.wikipedia.org/wiki/Соборяне')
 elif answer == 4:
 print('https://ru.wikipedia.org/wiki/Левша\_(сказ)')
 loop()
origin()