Муниципальное бюджетное общеобразовательное учреждение
города Ростов-на-Дону «Школа №10»

Тема индивидуального проекта
«Создание пользовательского программного обеспечения»

Выполнил:
ученик 10 «А» класса
МБОУ «Школа №10»
Асмаев Даниил Дмитриевич

Оглавление

Введение 3
Глава 1. Теоретические аспекты 4
1.1 Понятие о языках программирования 4
1.2 Понятие о прикладном программном обеспечении 7
1.3 Виды прикладного программного обеспечения 9
Глава 2. Практические аспекты 13
2.1 Особенности языка программирования Python 13
2.2 Среда разработки 16
2.3 Ход работы 18
Заключение 21
Список используемой литературы 22

Введение

Почти каждый сталкивался с проблемой невероятного загрязнения своего жесткого диска: музыка, фото, видео, документы и т.д. в одной папке под названием «Новая папка(582)». Когда нужно найти что-то важное, вы часами перебираете каталоги файлов, в попытке найти тот документ, но, если вы конечно не ярый перфекционист и заранее не сортируете свои файлы, но большинство людей склонны к образованию хаоса среди файлов на ПК, автор этого проекта также относится к таковым. Часто именно в папке «Загрузки», многие не могут понять, что происходит. Собственно, по этой причине было принято решение разработать приложение, которое поможет решить данную проблему.

Дабы не просто создать программу, а ещё и получить новые знания – поставим ещё и дополнительные цели:

1. Создать рабочие приложение
2. Изучить работу с файлами
3. Изучить работу с интерфейсом

Следуя из целей, можно выстроить цепочку задач:

1. Изучить библиотеки нужные для работы с файлами
2. Научиться применять библиотеки для работы с файлами
3. Изучить библиотеки нужные для работы с интерфейсом
4. Научиться применять библиотеки для работы с интерфейсом
5. Создать рабочий код сортировки
6. Создать прототип интерфейса
7. Доработать код, с целью получения новых возможностей сортировки
8. Создать финальную версию интерфейса
9. Связать код сортировки с интерфейсом

Глава 1. Теоретические аспекты

1.1 Понятие о языках программирования

Для создания программ вам необходимы знания языков программирования. При наличии огромного желания и большого количества времени, программу можно написать и на естественном языке (такое программирование называется: программирование на метаязыке), но перевести такую программу в машинный код автоматически пока невозможно, из-за невероятной неоднозначности естественного языка.

Языки программирования – это искусственные формальные языки. Как и естественные языки, у них в ассортименте имеется алфавит, словарный запас, грамматику, синтаксис и семантику.

Алфавит – дозволенный к использованию набор символов, при помощи которого могут быть образованы слова и величины любого языка.

Синтаксис – свод правил, определяющих возможные допустимые конструкции языка программирования из букв алфавита этого языка.

Семантика – система правил однозначного понимания каждой языковой конструкции, это позволяет производить быстрый процесс обработки данных.

У языков программирования имеется ограниченный запас слов (или же операторов) и строгие правила их написания. Как и для любого формального языка, правила семантики и грамматики четко и однозначно сформулированы. Основные понятия языка определяются взаимодействием синтаксических и семантических правил, таких как операторы, функции, константы, переменные, идентификаторы, процедуры и т.д.

Языками программирования, ориентированными на команды процессора и учитывающие его особенности, называют языки низкого уровня. Формулировка «низкий уровень» не означает неразвитость, имеется в виду, что операторы этого языка близки к машинному коду и ориентированы на конкретные команды процессора.

«Ассемблер» - он является языком самого низкого уровня. Программы, написанные на нем, представляют собой определённую последовательность команд из машинных кодов, но при этом записанных с помощью символьных мнемоник. При помощи языков низкого уровня, имеется возможность создать компактные оптимальные программы, так как программист имеет доступ ко всем возможностям процессора напрямую. С другой стороны, при этом ему требуется отличное понимать устройство компьютера (именное своего процессора или процессора для которого пишется данная программа (без наличия процессора для которого пишется данная программа, само написание невозможно)), а использование такой программы на компьютере с процессором другого типа невозможно, так как у каждого процессора имеются собственные особенности. Такие языки программирования зачастую используются для написания небольших системных приложений, модулей стыковки с нестандартным оборудованием, прямой доступ к аппаратным ресурсам, драйверов устройств, когда важнее компактность, быстродействие.

Языками «высокого уровня» называют языки программирования, имитирующие естественные, обладающие более массивными командами, ориентированы «на человека». Следовательно, чем выше уровень языка, тем ближе конструкции и структуры данных, которые используются в программе, к понятиям исходной задачи. Исходный текст программ легко переносить на другие платформы, имеющие трансляторы этого языка, так как особенности конкретных компьютерных архитектур в них не учитываются. Разрабатывать программы на языках высокого уровня при помощи мощных и понятных команд значительно проще, следовательно, и число ошибок,

допускаемых в процессе программирования, значительно уменьшается. Сейчас насчитывается несколько сотен таких языков.

1.2 Понятие о прикладном программном обеспечении

Прикладное программное обеспечение - это совокупность прикладных программ, при помощи которых на определённом рабочем месте выполняются конкретные задачи. Это программы конечного потребителя, общего и специального назначения. Они используются для решения задач в конкретной предметной сфере.

Помимо этого, есть множество прикладных программ специализированного назначения для профессиональной деятельности. Их зачастую называют пакетами прикладных программ. Это, к примеру, бухгалтерские программы, производящие эффективные начисления заработной платы и другие расчеты, которые производятся в бухгалтериях; системы автоматизированного проектирования, которые в свою очередь, помогают конструкторам разрабатывать проекты различных технических устройств; пакеты, позволяющие быстро и качественно решать сложные математические задачи без нужды в составлении программ; обучающие программы всевозможным школьным предметам и многое другое.

Пакеты прикладных программ (ППП) исполняют роль программного инструментария для решения функциональных задач и составляют самый многочисленным класс программных продуктов. В этот класс входят программные продукты, используемые для обработки информации различных предметных областей.

Для установки программных продуктов на компьютер потребуется прибегнуть к услугам квалифицированных пользователей, а конкретно их эксплуатацию осуществляют, в основном, конечные пользователи — конкретно, потребители информации, в большинстве случаев, деятельность оных весьма далека от IT области. Этот класс программных продуктов может быть весьма специфичным для отдельных предметных областей.

Невероятно популярным видом прикладного программного обеспечения можно назвать компьютерные игры, которые на данный момент захватили огромный пласт потребителей и плотно там обосновались.

В особых случаях, когда имеющиеся прикладные программы чересчур громоздки или не предоставляют возможность выполнять нужную обработку

данных, возникает потребность в индивидуальных прикладных программах. В основном, подобная ситуация может возникнуть при обработке исследовательских результатов. В данном случае исследователь сталкивается с необходимостью создания собственных прикладных программ.

Над созданием ППО трудятся множество профессиональных специалистов такие, как системные аналитики и программисты. Прикладное программное обеспечение предоставляется в форме комплекса программ на машинных носителях и сопровождается документацией к ним. В комплект документов, также включены руководства по работе с приложениями, сведения о конкретных программах и их назначении для решения определенных задач.

С большей частью прикладного программного обеспечения идут пакеты прикладных программ.

Во время создания подобных продуктов обязательно учитывается их удобство для конечных пользователей. Следовательно, учитывая интересы специалистов, разработчики стараются сделать максимально простым использование компьютерной техники.

Для упрощения процесса взаимодействия пользователя с компьютером, используются принципы интуитивно понятного интерфейса программного обеспечения, который осуществляется несколькими способами:

1.Входной язык пакета, то есть ввод с клавиатуры, выполнение команд, просмотр информации.

2.Указания встроенного средства или мастер-решения для пошагового решения задачи.

Прикладное программное обеспечение может отличаться назначением, функционалом и пользовательскими характеристиками. Такими, как язык интерфейса, особенности меню и работы с командами. В нынешних реалиях можно найти огромное количество приложений, с которыми можно работать в онлайн режиме. Программы можно открыть в интернет-браузере. Также есть другой тип программного обеспечения, который требует установки на компьютер пользователя или другое его электронное устройство.

1.3 Виды прикладного программного обеспечения

Сейчас крупнейшие фирмы мира разрабатывают программное обеспечение для всевозможных категорий пользователей. Соответственно, в зависимости от интересов пользователей все прикладные программы делятся на классы:

1. Текстовые редакторы – используются для создания и редактирования текста, без какого-то-либо оформления (Примеры: Notepad или Блокнот (входит в ОС MS Windows), TextPad)

2. Текстовые процессоры – используются для создания и редактирования текста с оформлением (задание шрифта, его размера, цвета текста, выравнивания и др.) также с внедрением таблиц, графиков и формул (Примеры: MS Word, WordPad (входит в ОС MS Windows))

3. Электронные таблицы – используются для обработки данных в табличной форме (Примеры: MS Excel)

4. Графические редакторы – используются для создания и редактирования изображений:

4.1 Растровые – используются для работы с растровыми изображениями (Примеры: MS Paint (входит в ОС MS Windows), Adobe Photoshop)

4.2 Векторные – используются для работы с векотрными изображениями (Примеры: CorelDRAW, Adobe Illustrator)

5. Просмотрщики – используются для просмотра файлов универсальных форматов:

5.1. Просмотрщики изображений – используются для просмотра изображений (Примеры: CDSee, FastStone Image Viewer, FastPictureViewer

5.2. Просмотрщики HTML-страниц (браузеры, веб-обозреватели) – используются для просмотра страниц веб-сайтов (Примеры: MS Internet Explorer, Mozila Firefox, Google Chrome, Opera, Safari)

5.3. Просмотрщики медиа контента (медиаплееры, медиапроигрыватели) – используются для воспроизведения медиа контента:

5.3.1. Аудиоплееры – используются для воспроизведения аудиофайлов (Примеры: AIMP, Foobar2000, Spider player,MusicBee, Media Monkey)

5.3.2. Мультимедиа-центры – используются для воспроизведения видео- и аудиофайлов (Примеры: Windows Media Player (WMP, входит в ОС MS Windows), QuickTime Player(входит в ОС Mac OS X), Winamp, VLC media player, Media Player Classic)

5.4. Просмотрщики flash-контента (Flash-плееры) – используются для воспроизведения видео и аудиофайлов на веб-сайтах, для игр он-лайн (Примеры: Adobe Flash Player)

5.5. Просмотрщик pdf-файлов – используются для просмотра и печати pdf-файлов (Примеры: Adobe Reader)

6. Системы управления базами данных (СУБД) настольные (файл-серверные) – используются для управления созданием и работой с базами данных (Примеры: MS Access, Paradox)

7. Компьютерные игры – используются для развлечения или обучения (Примеры: 3D-шутер, "Кот Леопольд. Учим английский язык")

8. Переводчики:

8.1. Электронные словари – используются для перевода отдельных слов (Примеры: ABBYY Lingvo, МультиЛекс)

8.2. Переводчики текстов – используются для перевода текста (Примеры: ПРОМТ)

9. Настольные издательские системы – используются для электронной верстки газет, журналов, книг, буклетов (составление страниц определенного размера из текста и иллюстраций для получения печатной формы) (Примеры: QuarkXPress, Adobe InDesign, Adobe FrameMaker, Corel Ventura, MS Publisher)

10. Электронные энциклопедии, учебники, словари, справочники – используются для получения знаний в определенной сфере (Примеры: "Энциклопедия современной техники. Строительство", "Справочник мастера-строителя", "Музыкальный словарь", интерактивный мультимедиа учебник "Органическая химия")

11. Системы автоматизированного перевода (CAT-программы) – используются для перевода профессиональных текстов с использованием баз знаний предметных областей (Примеры: Trados, Deja Vu, Star Transit)

12. Серверные СУБД(клиент-серверные) – используются для управления созданием и работой с базами данных информационных систем (Примеры: mySQL, MS SQL Server, Sybase Adaptive Server Enteprise, Oracle Database)

13. Редакторы трехмерной (3D) графики – используются для создания и редактирования трехмерной графики (Примеры: Autodesk 3ds Max (ранее 3D Studio MAX), Autodesk Maya, Blender, Cinema 4D, ZBrush)

14. Видеоредакторы (системы видеомонтажа) – используются для обработки видеоматериала:

14.1. Профессиональные – используются для монтажа фильмов, телепередач (Примеры: Adobe Premiere Pro, Grass Valley Ediu, Sony Vegas Pro)

14.2. «Домашние» – используются для монтажа "домашних" фильмов (Примеры: Windows Movie Maker (входит в ОС MS Windows), Corel VideoStudio Pro, Pinnacle Studio)

15. Аудиоредакторы (системы аудиомонтажа) – используются для обработки аудиоматериала:

15.1. Профессиональные – используются для записи музыкальных композиций, подготовки фонограмм для радио, озвучивания фильмов, компьютерных игр, реставрации старых фонограмм (Примеры: Adobe Audition, Steinberg WaveLab, Sony Sound Forge. Audacity)

15.2. «Домашние» – используются для записи любительских музыкальных композиций, для создания собственных рингтонов для мобильных телефонов и др. (Примеры: CyberPower Audio Editing Lab, Akram Audio Editor)

16. Нотные редакторы – используются для создания и редактирования нотного текста с оформлением, а также для проигрывания набранного текста (Примеры: Finale, Encore, Cakewalk Overture, Sibelius,MuseScore)

17. Экспертные системы – используются для решения задач некоторых предметных областей (заменяет специалиста-эксперта) (Примеры: MYCINACES, ACE, CODES, DENDRAL, PROSPECTOR, РЕМОРАМА)

18. Системы автоматизированного проектирования (САПР, CAD/CAM/CAE) – используются для разработки на компьютере чертежей, схем, 3D-моделей, конструкторской и технологической документации (Примеры: Компас, AutoCAD, ZwCAD, nanoCAD Электро, BtoCAD, Стройэкспертиза BASE)

19. Геоинформационные системы (ГИС):

19.1. Универсальные и специализированные – используются для создания, редактирования и анализа электронных географических карт, для поиска информации об объектах карты: городах, дорогах, зданиях, улицах, реках, рельефе, , среднегодовой температуре (Примеры: MapInfo, CREDO_DAT, ArcGIS, Arcview, GeoServer, GRASS, gvSIG, Арго, Полигон, Панорама, ГИС Метео)

19.2. Информационно-справочные – используются для просмотра карт города и окрестностей, для поиска организаций, маршрутов транспорта, поиска проезда по городу (Примеры: 2ГИС)

20. Общие системы для различных предприятий и организаций:

20.1. Интегрированные системы делопроизводства – используются для поддержки полного цикла документа в организации: создание документов (документирование), организация движения и учёта документов (документооборот), хранение документов (архивное дело) (Примеры: Дело, LanDocs, Золушка, Гран-Док)

20.2. Бухгалтерские системы – используются для ведения бухгалтерского и налогового учета (Примеры: 1С:Бухгалтерия)

20.3. Финансовые аналитические системы – используются для ведения аналитического учета финансово-хозяйственной деятельности организации (систематизация информации, расчет аналитических показателей и формирование отчетов) (Примеры: Альт – Финансы, Audit Expert, ИНЭК – Аналитик, ФинЭкАнализ, модуль для MS Office Excel "Excel Financial Analysis")

Для каждого из перечисленных классов конкурирующие фирмы создают разностороннее программное обеспечение. Кроме того, прикладные программы привязаны к конкретным операционным системам. Несмотря на многоплатформенность программного обеспечения перед пользователем возникает проблема выбора программного обеспечения, наилучшим образом отвечающего его интересам.

Глава 2. Практические аспекты

2.1 Особенности языка программирования Python

Говоря о технологических особенностях языка, есть несколько вещей которые можно объединить терминами академичность, продуманность, лаконичность, красивость языка. Если красивость языка - это всё-таки нечто субъективное, то архитектурная продуманность языка - это вещ абсолютно очевидная и наглядная. Например, если мы говорим о PHP, то там есть функция «str_replace», которая понимает 3 аргумента: 1-й аргумент – это «что ищем?»; 2-й аргумент «на что заменяем?»; 3-й аргумент «где ищем?».

```php
<?php

str_replace('Вася', 'Петя', 'Вася, привет!');
```

А функция «strpos»: 1-й аргумент «где ищем?»; 2-й аргумент «что ищем?».

```php
<?php

strpos('Вася, привет!', 'Вася');
```

То есть даже на уровне порядка аргументов в стандартных функциях PHP есть рассогласование, то есть PHP развивался абсолютно хаотично, зачастую не продумано, и его основатель Расмус Лердорф говорил, что он программировать не любит.

```
«Есть люди, которые любят программировать. Я их не понимаю.»

«Я не настоящий программист. Я просто соединил
вместе вещи, которые работали. Настоящий
программист сказал бы: "Это работает, но есть утечки
памяти на кажом шаге. Нужно пофиксить." А я вот
просто перезагружаю Apache через каждые
10 запросов»

Расмус Лердорф, основатель PHP
```

Если мы говорим про Python, то ребята, которые занимаются поддержкой и разработкой самого языка Python, программирование любят и прежде чем внедрить новую фичу или не внедрить в язык, происходит достаточно серьезный процесс обдумывания и обсуждения той или иной фичи, и это действительно здорово. То, что в Python есть «The Zen of Python» вот даже сама эта фича говорит о том, что в Python задумываются о том «каким должен быть язык?», «насколько он должен быть продуманным,

правильным?» и «как нём правильно писать код?» - это здорово. Вы никогда не задумывались «почему для измерения длины последовательности в Python используется глобальная функция «len»? Вроде одной стороны, Python - это объектно-ориентированный язык программирования и в объектно-ориентированных языках популярным подходом является получение длины как:

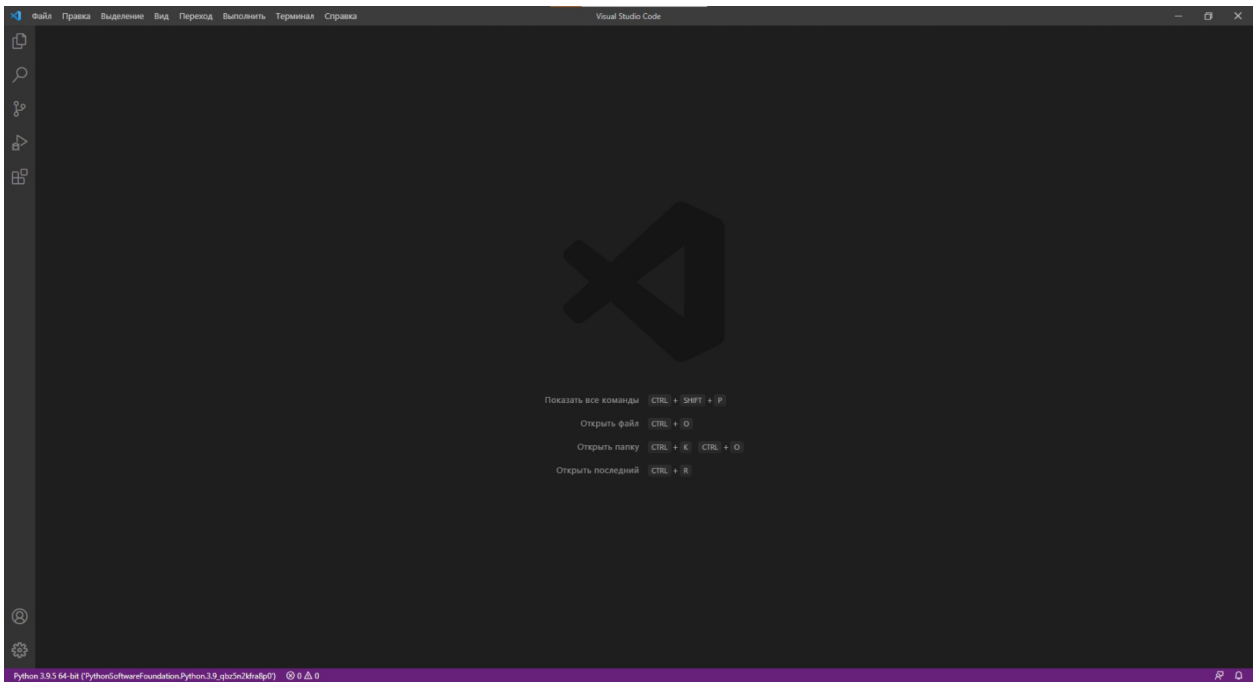
```
'привет'.length # JS, Ruby
'привет'.length() # Java

len('привет') # Python
```

Мы берем объект и вызываем его метод «len» или «length» или читаем атрибут «.length» (как в JS), а в Python пошли другим путем: используют глобальную функцию «len()» - это можно посчитать не объектно-ориентированным подходом, на самом деле это не так: это не ошибки, не какая-то непродуманность языка - это продуманная, хорошая архитектура. В Python не используем наследование, наследование большинстве случаев - это проблемный путь. Композиция, в большинстве случаев, - это хороший путь. Хорошо, что в Python понимают вот такие архитектурные принципы и закладывают их в сам язык. То есть, мы для того, чтобы создать последовательность, не наследуемся от какого-то встроенного типа данных, мы используем композицию и это потрясающе. То же самое касательно других способов работы с последовательностью, если мы знаем, как рандомно выбирать какой-то элемент из любой последовательности Python и реализовать вот таким простейшим образом свою последовательность, мы уже знаем, как рандомно оттуда достать элемент и все пользователи нашего класса тоже уже знают, как оттуда рандомно достать любой элемент. Знают «как проектироваться?», «как получить длину?», «как рандомно достать элемент?», «как работают слайсы?» - всё это работает абсолютно идентично для всех последовательностей, которые, как является встроенным спальным, так и которые мы реализуем - и это здорово, то есть мы получаем стандартизированный, продуманный, лаконичный, красивый язык и именно поэтому Python с технологической точки зрения крут. Именно поэтому на нём сейчас пишут много кода и это потрясающе. Критикуя Python многие говорят, что Python медленный. С одной стороны, это действительно так, то есть если нам нужно получить максимальное быстродействие какой-то очень нагруженный сервис, который выполняет какую-то одну узенькую задачу, но он должен работать очень быстро - это можно и нужно конечно же сделать на C. Но нельзя сказать, что Python «медленный и глупый» - то есть он медленный, потому что он неправильно спроектирован. Нет, это не так, то

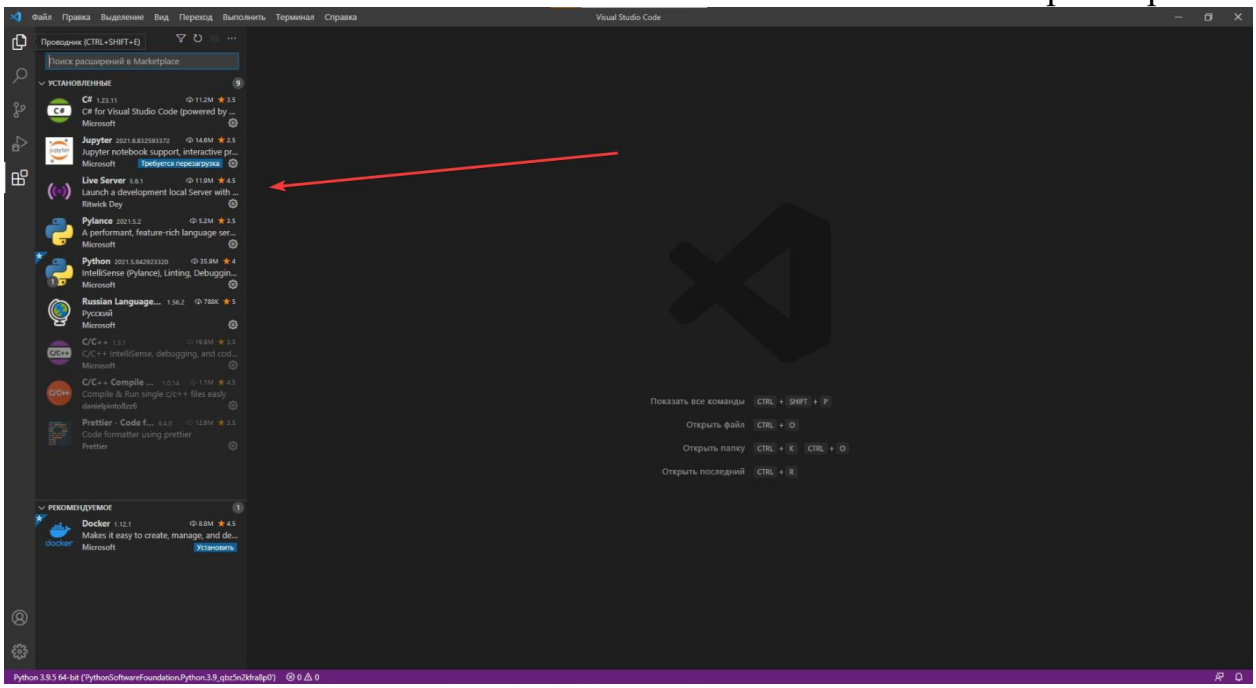
есть, если вам нужно сделать что-то быстрое, вы можете написать модуль на C для Python использовать Python просто как удобную обёртку над функционалом, который фактически разработан на C. То есть, если мы говорим про библиотеки анализа данных, то все эти библиотеки фактически написаны на C, то есть на низкоуровневом, быстром языке, а для Python реализована просто удобная обёртка для доступа к этим методам, потому что Python в данном случае используется как своего рода клей, то есть мы можем к сложным, низкоуровневых алгоритмам получать доступ очень красиво, очень наглядно используя красивые выразительные язык Python.

2.2 Среда разработки

Для разработки проекта Parallax использовалась IDE VS Code. VS Code была выбрана, так как она просто больше всего симпатизирует разработчику.



VS Code очень многофункциональная IDE, в основном это заслуга множества расширений:



благодаря которым VS Code позволяет писать код в максимально комфортных условиях, ведь они добавляют множество полезных функций, позволяющих сильно ускорять рабочий процесс. Подсветка ошибок и автодополнение команд, также очень влияет на скорость работы. VS Code позволяет писать код на более чем 30-языках программирования: на Python,

JavaScript, CoffeeScript, TypeScript, CSS и других популярных языках. В добавок к удобным и полезным функциям, есть и красивых и понятный дизайн, который тоже возможно кастомизировать и настроить под себя.

Так же VS Code имеет множество полезных инструментов для работы. В сам VS Code встроен отладчик запуска тестов, терминал. В VS Code введена система контроля версий git.

VS Code базируется на продуктах с открытым кодом, что иногда является важным критерием для разработчиков, поддерживает интеграции с системами контроля версий, встроенный отладчик и возможности подключения внешних инструментов.

Любой разработчик хотел бы подстроить программу, для написания кода, максимально под себя, для облегчения работы и повышения своего комфорта. Именно в VS Code любой может настроить всё для своего удовольствия, включая инструменты. Можно адаптировать как цветовую гамму, так и сочетания клавиш, для быстрого доступа к функциям.

VS Code можно использовать на компьютерах под управлением Windows, OS X и Linux. Инструмент вышел весной 2015 года, и постоянно обновлялся. За время существования Visual Studio Code расширил свой функционал, список поддерживаемых языков, основываясь на отзывах и пожеланиях пользователей.

Системные требования:

1. 1 ГБ ОЗУ

2. Минимум 200 МБ на диске

3. Процессор с частотой 1,6 ГГц и выше

2.3 Ход работы

В начале разработки встал вопрос: на каком языке писать приложение?

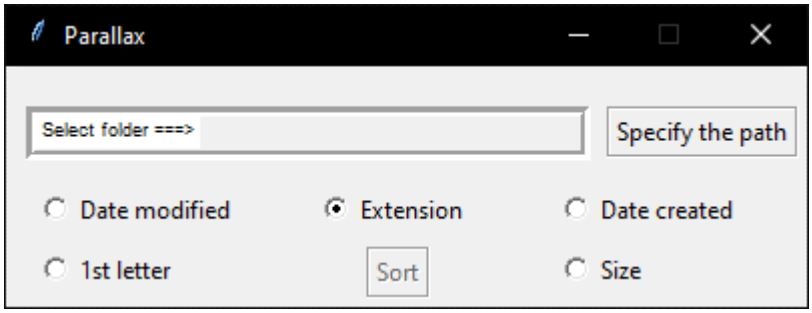
В итоге, было принято решение использовать Python, так как он показался самым простым и в тоже время очень функциональным. VS Code – это IDE которая была использована для написания данной программы, мы его выбрали из-за его простоты и функциональности.

Далее начался процесс подбора библиотек Python – были выбраны следующих:

```
import sys, re, os, shutil
from tkinter import*
import tkinter.filedialog
import tkinter.messagebox
import win32com.client
```

1. sys – модуль обеспечивает доступ к некоторым переменным, используемый или поддерживаемым интерпретатор, и к функциям, которые тесно взаимодействуют с интерпретатором
2. re – работа с регулярными выражениями
3. os – взаимодействие программы с операционной системой
4. shutil – взаимодействие с файлами
5. tkinter – библиотека при помощи, которой был реализован интерфейс.
6. win32com – модуль содержит константы, связанные с программированием Win

При помощи библиотеки tkinter был сделан интерфейс, при помощи которого происходит взаимодействие с функциями сортировки:



Далее нужно было определиться, как будут сортироваться файлы.

В первых прототипах, каждый вид сортировки был реализован отдельными крупными блоками полноценной сортировки с переносом файлов.

В последних прототипах было принято решение перейти на работу с метадатой. Это решение было безусловно верным, ведь это позволило стандартизировать параметры и создать единое ядро сортировки:

```
def meta_sorting_core(sorting_func):
    count_file = 0

    path = temp_file
    files = os.listdir(path)
    for file_name in files:
        if os.path.splitext(file_name)[1][1:] == "":
            continue

        file_data = get_file_metadata(path, file_name, ['Name', 'Size', 'Item type', 'Date modified', 'Date created'])

        path1 = sorting_func(path, file_data)

        if lang.get() == 1:
            if os.path.splitext(file_data['Name'])[0] == None:
                continue

        if lang.get() == 2:
            if not size_name[0][0].isdigit:
                continue

        if lang.get() == 3:
            if file_data['Date modified'] == None:
                continue

        if lang.get() == 4:
            if file_data['Date created'] == None:
                continue


        if not os.path.exists(path1):
            os.mkdir(path1)

        os.chdir(path)
        shutil.move(path + '/' + file_name, path1)

        count_file += 1
    if count_file == 0:
        tkinter.messagebox.showinfo(title="Attention", message="There is nothing to sort in the folder")
    if len(temp_file) > 0 and count_file != 0:
        tkinter.messagebox.showinfo(title="Successfully", message="Files sorted")
```

Ядро, в первую очередь, (если не берём в расчёт прокладку пути к файлам) проверяет является ли проверяемый объект папкой – это делается для исключения последних, так как сортировка иных является не целесообразной. Далее происходит процесс сбора информации об объекте, для последующего распределения. В последующем мы и производим сортировку по средствам вызова функции с интересующим параметром, которая в свою очередь вызывает ядро с функцией определённого параметра

в виде аргумента.

```
def sort_by_first_letter():
    meta_sorting_core(first_letter_sorting_func)


def size_sorting_func(ppath, file_data):
    global size_name
    size_name = file_data['Size'].split(' ')
    print(os.path.splitext(file_data['Name'])[1][1:])

    return name_of_folder_of_size(ppath, convert_size_to_str(size_name))
```

Далее происходит обработка возможных ошибок на случай отсутствия интересующего нас элемента про заданной функции сортировки.

Проверяется наличие папки по заданному паттерну, если она отсутствует, то таковая создаётся.

При успешном прохождении всех проверок объект переносится в соответствующую папку

После прохождения всеми файлами всех проверок, всплывает окно:
1.Если папка назначения пуста или в ней нет объектов, которые можно сортировать по заданному параметру, то окно содержит предупреждение об отсутствие файлов подлежащих сортировке
2. Если какое-то количество файлов было успешно отсортированно по заданному параметру, то окно содержит сообщение об успешном окончании сортировки.

Заключение

В исследовательской работе рассмотрены теоретические и методические аспекты для сортировки файлов на языке Python с использованием IDE VS Code. Во введении определена актуальность выбора темы, сформулирована цель и выдвинуты задачи исследования. Следуя пунктам, указанным в вышеописанных главах, используя нужное программное обеспечение и собственные навыки программирования, по итогу удалось создать приложение для сортировки файлов, совместившее в себе множество компонентов.

В первой главе были задействованы основные понятия и термины, которые в последующих главах сыграли основную роль. Во второй – применение новых знаний на деле.

Рассмотрев получившийся результат, было выдвинуто общее мнение о том, что же в итоге вышло из всей этой работы. Изначально перед нами была поставлена цель – создать удобное приложение для сортировки файлов. Исходя из общего консенсуса было выдвинуто соответствующее мнение – цель данного проекта была достигнута. По общей приложение действительно получает положительные отзывы от различных пользователей, использующих данную работу.

Для достижения данной цели были выполнены задачи, поставленные в начале пути. Для приложения был разработан интерфейс, который интуитивно понятен пользователю. Для непосредственного написания кода приложения были подробно изучены основы и особенности Python файлов, в которых как раз-таки и был построен интерфейс и создано ядро сортировки.

В рамках выполнения выпускной квалификационной работы все поставленные задачи исследования решены, цель исследования достигнута. Данный проект показал потенциал разработки приложений и общую перспективу строить на основе этого карьеру. Для этого нужно и дальше совершенствовать собственные навыки программирования, которые в будущем обязательно принесут свои плоды.

Список используемой литературы

Документация «tkinter» - https://docs.python.org/3/library/tkinter.html

Документация «win32» - http://timgolden.me.uk/pywin32-docs/contents.html

Документация «os» - https://docs.python.org/3/library/os.html

Документация «re» - https://docs.python.org/3/library/re.html

Документация «sys» - https://docs.python.org/3/library/sys.html

Документация «shutil» - https://docs.python.org/3/library/shutil.html