Программирование

Программирование – процесс создания программ для ЭВМ. В широком смысле может включать в себя все этапы от проектирования до выпуска программы в работу.

Языки программирования в узком смысле это инструмент «общения» с ЭВМ, а в широком это инструмент для кодирования алгоритмов. Для эффективного использования технологии программирования существуют среды разработки. Так VC и PyCharm – это интегрированные среды разработки(IDE). IDE по сути представляют собой набор инструментов, объединенных в одну программную систему для обеспечения процесса разработки. IDE удобны для работы над проектами. Jupyter notebook – это интерактивная среда разработки. Google Colab - является реализацией Jupyter от компании Google и развернутой на ее облачных мощностях, т.е. вычисления проводятся не на рабочей станции пользователя. Google Colab запускает ноутбуки(.ipynb) на облачной виртуальной машине под Lunix. Основная область применения :

- Для исследования
- Обучения
- Создания прототипов или MVP(минимального функционирующего продукта)

Ключевое отличие между классическим исполнением программы и интерактивным в том, что при классическом код пишется заранее, а далее компилируется/интерпретируется и потом исполняется в соответствии с точкой входа(так например в VC и PyCharm это точка входа либо указывается, либо ищется по умолчанию и исполняется интерпретатором Python), а для интерактивного происходит запуск ядра, после которого возможен процесс дальнейшего написания кода. Файлы «.py» - скрипты, модули, файлы библиотек и т.д. (это файлы с кодом Python), а файлы «.ipynb» - это рабочие блокноты, которые используются для интерактивного взаимодействия путем исполнения отдельных ячеек кода. Иными словами «.py» готовые схемы (код исполнения должен быть написал заранее), а «.ipynb» чертеж (код исполнения пишется в процессе и может исполнятся минимальными единицами –ячейками).

LLM для программирования (кодирования)

Большие языковые модели (LLM) способны генерировать простой и работоспособный программный код на различных языках программирования. При этом давая пояснения по его работе и способов запуска на исполнение. Данный функционал полезен для:

- Создания прототипов
- Исследований
- Небольших проектов
- Обучения
- Поиска идей реализации

Особенно полезным видится применения для кодирования рутинных операций или относительно быстрого получения рабочего варианта алгоритма/подзадачи/скрипта/модуля.

Существуют специализированные LLM для программирования, называемые ассистенты разработки. Тем не менее значимо ускорить процесс разработки на данном этапе получается только для программистов начальной и средней квалификации. Так как даже для кода полученного от специализированных LLM требуется проверка и дальнейшая коррекция.

Мы будем использовать LLM для решения подзадач программы (реализации отдельного алгоритма). А именно, для каждой единицы функционала разрабатываемой программы произведем постановку задачи в форме «вход-выход», а далее добьемся соответствия с полученным от LLM кодом.

Технология решения проблемы.

Далее три интерпретации одного процесса

I

Проблема (актуальность) – постановка задачи(формализация ограничений) – реализация решения(выполнения списка задач)

II

0. **Описание проблемы (Актуальность)**
1. **Цель: Решить проблему**
2. **Задачи: Шаги для решения**
3. **Постановка задачи – формализация решения задачи**
4. **Выбор инструмента и его использование для решения**

III

В общем виде алгоритм решения будет соответствовать этой структуре:
Для решения **проблемы** – **Цель**
Для достижения **Цели** – **Задачи**
Для конкретизации **задачи** – **постановка задачи** (Дано – Найти, Вход – Выход)
Для выполнения **задачи** – выбор **инструмента** (пути достижения) и **реализация**

«Разработка автоматизированной системы проверки работы студентов»

Актуальность. **Процесс проверки большого количества эссе трудозатратен**.

Цель. **Автоматизация процесса проверки эссе**

Задачи:

1. *Анализ предметной области (специфика и особенности, поиск и анализ существующих решений частично или полностью взять готовые, дополнить, предложить свое).*
2. *Обзор и выбор (опционально) инструментов в соответствии с анализом предметной области и необходимыми технологиями для решений в области.*
3. *Разработка алгоритмов для оценки эссе.*
4. *Проектирование программной системы (модуля) и ее реализация в соответствии с задачей 3.*
5. *Интеграция (разработка интерфейса).*

I Аналитическая часть.

Исследуем, какими методами и инструментами можно достигнуть цели

1.1. Анализ предметной области.

1.2. Инструменты и методы (технологии).

II Разработка алгоритмов

Декомпозируем задачу оценки на подзадачи и разрабатываем алгоритмическое решение

2.1. Разработать алгоритм оценки эссе. Здесь решаем, как именно оценивать работу. Например, считаем, что оценивать нужно по 5 критериям и суммированному резюме по качеству и содержанию эссе.

2.2. Разработать алгоритм сбора статистики по тексту эссе. Решаем, какую статистику по тексту работы хотим увидеть. Например, хотим видеть частотность n самых частных слов и их общее количество.

2.3. Разработать алгоритм для вывода отчета. Решаем, в каком виде и формате хотим получить итоговый результат. Например, хотим получить отчет по работе предыдущих алгоритмов в виде pdf файла.

Таким образом, на входе у нас будет текст работы студента, а на выходе отчет с оценкой в формате pdf.

III Проектирование программной системы

Если есть алгоритм, то его можно реализовать. В данном случае алгоритмы мы можем частично закодировать сами, частично с помощью LLM. Наша задача декомпозирована на три алгоритма, поэтому выделяем три отдельных модуля, по одному на каждый. Модули (подмодули), которые нам необходимо реализовать программно:

1) *Модуль для оценки эссе*
2) *Модуль для статистики по тексту*
3) *Модуль для отчета по 1 и 2*

Инструменты:

Основной. Язык программирования *Python*. Среды *Visual Code, Google Colab*.

Вспомогательные:

1) LLM *GiGaCha –для оценки эссе по выбранным критериям, взаимодействуем из Python по API*
2) LLM *DeepSeek – для помощи в кодировнии остальных задач (частотность слов, отчет)*

Интеграция. Решаем, что итоговый код подмодулей объединим в один либо в функцию, либо в файл, чтобы была возможность интегрировать в любой другой программный код на Python (в том числе в телеграмм – бота). Формализуем вход и выход модуля.