Сизов

Данил

Евгеньевич

Информатика

или

Введение в специальность

# **#1** **Гибкие методологии проектирования**

**Кабан и скрап** – это гибкие методологи проектирования в управ­лении проек­тами, которая активно используется в разработке ПО и не только.

Это методология визуального управления рабочим процессом, которая по­могает увеличить эффективность и гибкость работы команды. Основная идея заключается в визуализации всех за­дач на доске (физической или цифро­вой), где каждая задача представлена в виде карточки, проходящей через несколько этапов.

**Пример:**

Ожидает => В процессе => Готово

Kanban помогает выявлять узкие места и оптимизировать поток задач. Глав­ные элементы Kanban включают:

* Визуализация работы
* Лимитирование незавершенных задач
* Постоянное измерение и совершенствование процесса

Scrum

Это гибкая методология, которая фокусируется на **медитатив­ной** и **инкре­ментальной** методологии.

Основные компоненты Scrum:

1. Роли:

* Владелец продукта (PO) - отвечает за приватизацию задач и об­щение с клиентами
* Scrum мастер (SM) - помогает команде работать эф­фективно и ре­шает её проблемы
* Команда разработки (DT) - работает над задачами и создаёт про­граммный продукт

1. События:

* Планирование Sprint
* Ежедневные Scrum встречи
* Обзор Sprint
* Ретроспектива

1. Артефакты:

* BackLog продукта
* BackLog Sprint
* Инкремент

**Определения:**

**BackLog** - список задач, которые надо выполнить для создания продукта. Он содержит все изменения, требования и функции, которые надо реализовать. BackLog является основным артефактом Scrum и управляется владельцем продукта.

Так же **BackLog** имеет некоторые характеристики:

* Он является приоре тезированным
* Он является изменяемым, задачи могут быть изменены, добавлены или обновлены в любое время
* Он доступен для всей команды, чтобы обеспечить прозрачность и понимание задач

**Sprint** — это короткий временной интервал, в течение которого Scrum-ко­манда выполняет заданный объем работы.

* Фиксированная длинна. Команда знает сколько времени у неё есть на выполнение задач
* Он является итеративным. Команда работает в цикле спринтов
* Он сосредоточен на доставке продукта заказчику

# **#****2 Колмогоровская сложность**

**Колмогоровская сложность** — это мера сложности алгоритма или компью­терной программы, предложенная Андреем Колмо­горовым в 1960-х годах. Она измеряет количество информации, необходимой для описания объекта или алгоритма.

**Колмогоровская сложность объекта x**, обозначаемая как K(x), — это длина самого короткого бинарного кода, который может быть использован для опи­сания объекта x.

**Колмогоровская сложность** может быть рассчитана по следую­щей фор­муле:

**x** - объект, для которого мы хотим рассчитать сложность

**p** - бинарный код, описывающий объект x

**U** - универсальная машина Тьюринга, которая может интерпре­тировать би­нарный код p и выдавать объект x

**|p|** - длина бинарного кода p

**Пример**

Допустим, мы хотим рассчитать Колмогоровскую сложность строки "Hello, world!". Самый короткий бинарный код, который может быть использован для описания этой строки, — это сама строка, закодированная в бинарном фор­мате. Длина этой строки составляет 96 бит (12 символов x 8 бит на символ). По­этому Колмогоровская сложность строки "Hello, world!" состав­ляет 96 бит.

**Важные свойства**

Колмогоровская сложность имеет следующие важные свойства:

* Асимптотическая неопределенность: Колмогоровская сложность не может быть точно рассчитана для большин­ства объектов, потому что это проблема остановки для ма­шины Тьюринга.
* Аддитивность: Колмогоровская сложность объекта, со­стоящего из двух частей, равна сумме Колмогоровских сложностей этих частей.
* Непереводимость: Колмогоровская сложность не зави­сит от исполь­зуемого языка программирования или ма­шины Тьюринга.

# **#3** **Списки (массивы) в Python**

Списки — это тип данных в Python, который позволяет хранить коллек­цию элементов одного типа или разных типов. Списки являются изменяе­мыми, то есть их содержимое может быть изменено после создания.

**Создание списка**

Список создается с помощью квадратных скобок и элемен­тов, разделен­ных запятыми. Например:

**Индексация списка**

Каждый элемент списка имеет свой индекс, который начина­ется с 0. Индекс используется для доступа к элементу списка. Например:

**Методы списка**

Списки имеют несколько методов, которые могут быть использо­ваны для ма­нипуляции элементами списка:

добавляет элемент в конец списка

добавляет все элементы из iterable в конец списка

вставляет элемент на указанный индекс

удаляет первый элемент, равный указанному

удаляет элемент на указанном индексе и возвращает его

сортирует список в порядке возрастания

сортирует список в порядке убывания

**Примеры использования методов списка**

**Вывод**

В этом параграфе мы рассмотрели основы работы со списками в Python, включая создание списка, индексацию, методы списка и примеры их ис­пользования. Списки — это мощный инструмент для работы с коллекциями данных в Python

# **#4 Turbo Pascal**

**Turbo Pascal** — это интегрированная среда разработки и компилятор для языка программирования Pascal, созданный компанией Borland. Он стал по­пулярным в 1980-х и 1990-х годах благодаря своей высокой скорости компи­ляции и удобному интерфейсу. *Turbo Pascal* использовался для разработки программного обеспечения, учебных материалов и научных исследований.

**История развития языка**

* **1970-е годы**: язык Pascal был разработан Никлаусом Виртом в каче­стве учебного языка, который помогал студентам понять основы про­граммирования.
* **1983 год**: Borland выпустила первую версию Turbo Pascal, которая предоставила пользователям мощный компилятор и интегрирован­ную среду разработки.
* **1986 год**: появилась версия Turbo Pascal 3.0, которая добавила под­держку графики и улучшенные функции.
* **1990-е годы**: Разработаны новые версии, включая Turbo Pascal 7.0, которая стала стандартом для многих разработчиков.

**Основные особенности**

* Высокая скорость компиляции: Turbo Pascal известен своей быстрой компиляцией, что делает его удобным для разработки.
* Интегрированная среда разработки: включает текстовый редактор, компилятор и отладчик в одном приложении.
* Поддержка модульного программирования: позволяет разбивать код на модули для улучшения структуры и читаемости.
* Графические возможности: поддерживает работу с графикой и ин­терфейсами, что делает его подходящим для разработки игр и прило­жений.

**Примеры простых программ на Pascal**

1. Программа "Hello, World!"

*Эта программа выводит на экран сообщение "Hello, World!».*

1. Программа для сложения двух чисел

*Эта программа запрашивает у пользователя два числа, суммирует их и выводит результат.*

1. Программа для нахождения факториала числа

*Эта программа вычисляет факториал введенного пользователем числа.*

Содержание

Оглавление

[**#1 Гибкие методологии проектирования** 3](#_Toc180314869)

[**#2 Колмогоровская сложность** 5](#_Toc180314870)

[**#3 Списки (массивы) в Python** 6](#_Toc180314871)

[**#4 Turbo Pascal** 8](#_Toc180314872)