**Рубцов Евгений Александрович**

Радости профессии:

1. Радость творчества
2. Помощь другим людям
3. Разгадывание головоломок
4. Постоянная новизна – скучать не придется
5. Работа с податливым материалом

Печали профессии:

1. Безукоризненная точность действий
2. Зависимость от других людей
3. Зависимость от ПО
4. Процесс поиска и устранения ошибок
5. Устаревание ПО

***Когнитивное сопротивление – “паралич мозга”***

Способ решения:

Устал – отдохни

Ненависть – понять причину ненависти

Не нравится область работы – скажи

Основное: гармония с собой и ЗОЖ + СПОРТ

***Апперцепция***

Восприятие информации с учетом знаний, которые есть у вас (поиск знакомой информации относительно новой в своей голове)

|

\/

*Метафоры процесса программирования*

1. Приборостроение
2. Строительство (Архитектура ПО)
3. Растениеводство (Всё идет не сразу, а в процессе)
4. Хирургия (Внесение изменений в действующую систему)
   1. Подготовка к операции – планы и согласование замены кода
   2. Операция – замена кода
   3. Постоперационное наблюдение – мониторинг результатов

**Особенности ПО**

1. Сложность (мн-во элементов и связь между ними)
2. Незримость
3. Изменчивость
4. Бесконечное многообразие (Потребность в согласовании действий)

**Два результата программирования**

1. Исходный код
2. Процесс

Программа пишется не для компьютера, а для человека

**ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ**

Императивное программирование – последовательный подход (подробная последовательная инструкция)

Фредерик Тейлор и его принцип : найти лучшего – записать как он делает работу – научить остальных

Декларативное программирование – даем конечный результат (закладывается база знаний) Логическое программирование – язык Prolog или SQL (описали что выбрать, а как выбирает – не имеет значения)

Функциональное программирование – подход с конца, разделяем работу на микропроцессы (формулы Excel = ОКРУГЛ(СР.ЗНАЧ(ТОЧНОСТЬ……)) Lisp ЯП

*Подразделение парадигм:*

Структурное программирование – подход к программированию, при котором программа представляет собой множество вложенных друг в друга блоков, каждый из которых имеет только один вход и только один выход (NO goto). Управление между блоками передается 3 способами:

1. Последовательность
2. Ветвление
3. Цикл

Читабельный код:

1. Отступы
2. Разделение символов пробелами
3. Пустые строки
4. Осмысленное название

Процедурное программирование – появилась идея функций, макросов, подпрограмм. (выносить код в блоки и повторно его вызывать) ~~глобальные переменные~~

Каждая процедура выполняет только ОДНУ задачу

Суть в независимости функций

Модульное программирование – разбиение на модули по связям между элементами.

Связи между элементами внутри одного модуля должны быть сильнее чем связи между элементами, находящимися в разных модулях.

ООП – создается класс, с полями и методами. С объектами класса можно плодить много модулей.

Инкапсуляция:  
 1) Доступ напрямую к данным запрещен

2) Доступ к данным осуществляется при помощи методов

Наследование:

- это создавать новые типы на основании существующих. Объект – экземпляр класса

Полиморфизм:

- это создание пустого метода для реализации его в других классах. Каждый класс выполняет этот метод по-своему.

Комментарии

1. Построчные
2. Вводные (описание блока кода в его начале)
3. Оглавление (список функций или классов)

Комментарий должен описывать содержание, а не форму!

Неверный комментарий хуже его отсутствия!

Псевдокоды – это комментарии без кода, а потом по ним вставлять код.

Принцип компиляции программы с первого раза – ну тут всё сказано

Оборонительное программирование – защита от дурака

Использование промежуточных переменных – не писать огромные формулы, а делить их на переменные.

Как сократить код?

1. Убрать то, что не нужно
2. Убрать повторы
3. Упростить код

Ин инглиш YAGNI (You aren't gonna need it) – тебе это не надо

DRY (Don’t repeat yourself) – не повторяйся

KISS (Keep It Simple, stupid) – как будет проще

Программа = алгоритмы + структуры данных

2 направления разработки ПО от функции и от данных:

Обычное программирование – от функций, SQL от данных, ООП – от всего

**Типы данных**

Типизированные и не типизированные языки программирования

Строгая и не строгая типизация: си-подобные VS питон (про объявление)

Явная и не явная типизация – про преобразование типов

Статическая и динамическая типизация – если ошибка до компиляции – статическая, во время компиляции – динамическая.

Простые и составные типы – простые:

Число

Строка

Boolean

Data

Сложные:

Массивы

Списки

Классы