МДК. 02.01 Технология разработки программного обеспечения

Тема 2.1.1 Основные понятия и стандарти­зация требований к программному обеспе­чению

Лекция 4:

Стандарты кодирования.

Различные модули, указанные в проектной документации, кодируются на этапе кодирования в соответствии со спецификацией модуля. Основной целью этапа кодирования является кодирование из проектной документации, подготовленной после этапа проектирования, на языке высокого уровня, а затем модульное тестирование этого кода.

Хорошие организации по разработке программного обеспечения хотят, чтобы их программисты придерживались определенного и стандартного стиля кодирования, называемого стандартами кодирования. Обычно они создают свои собственные стандарты и руководства по кодированию в зависимости от того, что лучше всего подходит для их организации, и на основе типов программного обеспечения, которое они разрабатывают. Для программистов очень важно поддерживать стандарты кодирования, в противном случае код будет отклонен во время проверки кода.

**Цель наличия стандартов кодирования:**

* Стандарт кодирования дает единообразный вид кодам, написанным разными инженерами.
* Это улучшает удобочитаемость и удобство сопровождения кода, а также снижает сложность.
* Это помогает в повторном использовании кода и помогает легко обнаружить ошибку.
* Это продвигает здоровые методы программирования и увеличивает эффективность программистов.

Некоторые из стандартов кодирования приведены ниже:

1. **Ограниченное использование глобалов:**  
   Эти правила говорят о том, какие типы данных могут быть объявлены глобальными, а какие нет.
2. **Стандартные заголовки для разных модулей:**  
   Для лучшего понимания и обслуживания кода заголовок различных модулей должен соответствовать стандартному формату и информации. Формат заголовка должен содержать ниже вещи, которые используются в различных компаниях:
   * Наименование модуля
   * Дата создания модуля
   * Автор модуля
   * История изменений
   * Краткое описание модуля о том, что делает модуль
   * В модуле поддерживаются различные функции, а также их входные и выходные параметры
   * Глобальные переменные, доступные или измененные модулем
3. **Соглашения об именах для локальных переменных, глобальных переменных, констант и функций:**  
   Некоторые из соглашений об именах приведены ниже:
   * Содержательное и понятное название переменных помогает любому понять причину его использования.
   * Локальные переменные должны быть названы с использованием букв верблюда, начинающихся с маленькой буквы (например, **localData** ), тогда как имена глобальных переменных должны начинаться с заглавной буквы (например, **GlobalData** ). Имена констант должны быть сформированы только заглавными буквами (например, **CONSDATA** ).
   * Лучше избегать использования цифр в именах переменных.
   * Названия функции должны быть написаны в верблюжьем регистре, начиная с маленьких букв.
   * Название функции должно четко и кратко описывать причину ее использования.
4. **Отступ:**  
   Правильный отступ очень важен для улучшения читабельности кода. Чтобы сделать код читабельным, программисты должны правильно использовать пробелы. Некоторые из интервалов даны ниже:
   * После запятой между аргументами функции должен быть пробел.
   * Каждый вложенный блок должен иметь правильные отступы и интервалы.
   * Надлежащий отступ должен быть в начале и в конце каждого кадра в программе.
   * Все фигурные скобки должны начинаться с новой строки, а код, следующий за окончанием фигурных скобок, также начинается с новой строки.
5. **Возвращаемые значения ошибок и соглашения об обработке исключений:**  
   Все функции, которые сталкиваются с ошибкой, должны возвращать 0 или 1 для упрощения отладки.

С другой стороны, руководящие принципы кодирования дают некоторые общие рекомендации относительно стиля кодирования, которому необходимо следовать для улучшения понятности и читабельности кода. Некоторые из руководств по кодированию приведены ниже:

1. **Избегайте использования стиля кодирования, который слишком сложен для понимания:**  
   Код должен быть легко понятным. Сложный код делает обслуживание и отладку трудной и дорогой.
2. **Избегайте использования идентификатора для нескольких целей:**  
   Каждой переменной должно быть дано описательное и осмысленное имя, указывающее причину ее использования. Это невозможно, если идентификатор используется для нескольких целей, что может привести к путанице у читателя. Более того, это приводит к большим трудностям при будущих улучшениях.
3. **Код должен быть хорошо документирован:**  
   Код должен быть правильно прокомментирован для понимания. Комментарии относительно утверждений повышают понятность кода.
4. **Длина функций не должна быть очень большой:**  
   Длинные функции очень трудно понять. Вот почему функции должны быть достаточно маленькими, чтобы выполнять небольшую работу, а длинные функции должны быть разбиты на маленькие для выполнения небольших задач.
5. **Старайтесь не использовать оператор GOTO:**  
   Оператор GOTO делает программу неструктурированной, что снижает ее понятность и затрудняет отладку.

**Преимущества Руководства по кодированию:**

* Рекомендации по кодированию повышают эффективность программного обеспечения и сокращают время разработки.
* Рекомендации по кодированию помогают выявлять ошибки на ранних этапах, что помогает снизить дополнительные расходы, связанные с программным проектом.
* Если правила кодирования поддерживаются должным образом, то программный код повышает читабельность и понятность, таким образом, уменьшает сложность кода.
* Это снижает скрытые затраты на разработку программного обеспечения.