Тема: Инструментальные средства разработки программного обеспечения

Цель работы: Изучить современные инструментальные среды разработки программного обеспечения.

Основные понятия темы

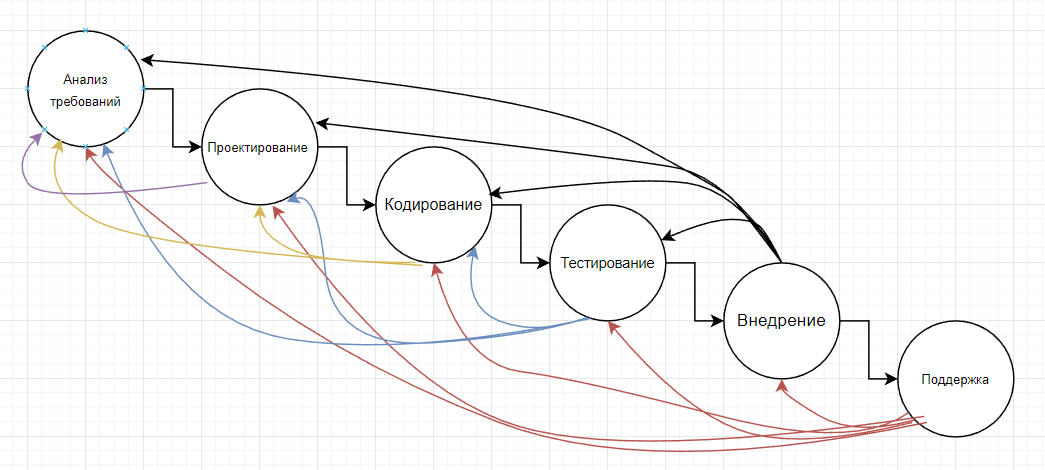
Программное обеспечение (ПО) — совокупность программ, процедур и документации, обеспечивающих функционирование вычислительной системы и выполнение определённых функций.

Инструменты разработки ПО — программное обеспечение, используемое разработчиками для написания, тестирования, отладки и управления проектами.

Процесс разработки ПО включает этапы анализа требований, проектирования, кодирования, тестирования и сопровождения.

Циклическая модель проектирования ПО

Схема циклической модели проектирования ПО:



Каждый этап сопровождается возвратом на предыдущие стадии для исправления ошибок и улучшения качества продукта.

Процесс описания реализации программного кода

Этапы процесса описания реализации программного кода:

Проектирование архитектуры: Определение структуры программы, модулей и интерфейсов взаимодействия.

Написание спецификаций: Формализация функциональных и нефункциональных требований.

Кодирование: Реализация функциональности согласно проектному решению.

Тестирование: Проверка работоспособности приложения посредством юнит-тестов, интеграционных тестов и нагрузочного тестирования.

Документирование: Создание технической документации для разработчиков и пользователей.

Развёртывание: Размещение готового решения на целевых серверах.

Поддержка: Устранение багов, улучшение производительности и обновление функционала.

Состав современных систем программирования

Современная система программирования состоит из:

1. Лексический анализ

Задача: Преобразовать исходный код в поток лексем (символьных единиц языка программирования).

Примеры операций: выделение токенов (переменных, операторов, чисел, знаков пунктуации и т.п.).

1. Синтаксический анализ (парсинг)

Задача: Проверить синтаксис программы и построить абстрактное синтаксическое дерево (AST).

Примеры операций: построение дерева выражения, проверки соответствия правилам грамматики языка.

1. Семантический анализ

Задача: Проведение семантических проверок (корректность типов переменных, область видимости, доступ к ресурсам и т.д.) и сбор информации о структуре программы.

Примеры операций: контроль типов, проверка имен переменных и функций, разрешение ссылок.

1. Оптимизация кода

Задача: Улучшить качество генерируемого кода путем устранения избыточных конструкций, оптимизации вычислений и управления памятью.

Примеры операций: удаление неиспользуемых фрагментов кода, уменьшение количества обращений к памяти, применение inline-функций, устранение ненужных ветвлений.

1. Генерация целевого кода

Задача: Создание исполняемого файла или объектного кода, соответствующего целевой платформе.

Примеры операций: генерация ассемблерного кода, двоичного машинного кода, промежуточного байт-кода.

1. Управление зависимостями и библиотеками

Задача: Автоматически отслеживать внешние библиотеки и зависимости проекта.

Примеры операций: подключение библиотек, автоматическое связывание динамических и статических библиотек.

1. Поддержка стандартов и соглашений платформ

Задача: Обеспечить совместимость с разными аппаратными платформами и операционными системами.

Примеры операций: использование директив препроцессора (#ifdef/#ifndef), поддержка многоцелевых платформ (Windows, Linux, macOS).

1. Диагностика и предупреждение ошибок

Задача: Предоставлять информативные предупреждения и диагностические сообщения о возможных ошибках или потенциальных проблемах.

Примеры операций: вывод предупреждений о некорректных операциях с памятью, подозрительном поведении программы, устаревших конструкциях языка.

1. Обработка макросов и предварительная обработка

Задача: Предварительно обрабатывать исходный код перед началом основной трансляции.

Примеры операций: замена макроопределений, условная компиляция, включение заголовочных файлов.

1. Профилировка и анализ производительности

Задача: Помогать выявлять узкие места и проблемные участки кода для последующей оптимизации.

Примеры операций: встроенные профилировщики, подсказки по улучшению производительности.

Редактора исходного кода.

Средства сборки проекта (компилятор, интерпретатор).

Отладчика для обнаружения и устранения ошибок.

Библиотек и фреймворков для ускорения разработки.

Инструментов автоматического тестирования.

Система контроля версий (Git, SVN).

Функции современных компиляторов

Современные средства программирования

|  |  |
| --- | --- |
| Средство | Краткое описание |
| Python | Высокоуровневый язык общего назначения с простым синтаксисом и богатой экосистемой библиотек. |
| C++ | Язык низкого уровня, используется для высокопроизводительных приложений, драйверов и встроенных систем. |
| VSS | Версионная система SourceSafe от Microsoft, позволяющая отслеживать изменения в проекте. |
| MS Visual Studio | Интегрированная среда разработки от Microsoft для .NET Framework, поддерживающая языки C#, VB.NET, F# и др. |
| Oracle | Реляционная СУБД корпоративного класса с поддержкой сложных запросов и транзакционной целостности. |
| MS SQL Server | Реляционная база данных от Microsoft, широко применяемая в корпоративных приложениях Windows. |
| MySQL | Открытая реляционная СУБД, используемая в веб-приложениях и системах баз данных среднего масштаба. |

Этапы проектирования приложений

1. Определение требований: Сбор и формализация бизнес-требований заказчика.
2. Архитектура: Выбор технологии и общей схемы будущей системы.
3. Детальное проектирование: Описание конкретных компонентов и алгоритмов.
4. Реализация: Написание и тестирование кода.
5. Интеграция: Соединение отдельных частей в единую систему.
6. Тестирование: Проведение комплексных проверок качества продукта.
7. Документация: Подготовка технического руководства и пользовательской документации.

Нотации и средства для этапа проектирования

1. UML-диаграммы: визуализируют структуру и поведение системы.
2. ER-модели: используются для моделирования структур базы данных.
3. CASE-средства: специализированные программы для автоматизации процессов проектирования (Rational Rose, Enterprise Architect).

Вывод: Мы с напарником изучили современные инструментальные средства разработки программного обеспечения.