**Инструментальные средства разработки программного обеспечения**

Цель работы: изучить современные инструментальные средства разработки программного обеспечения.

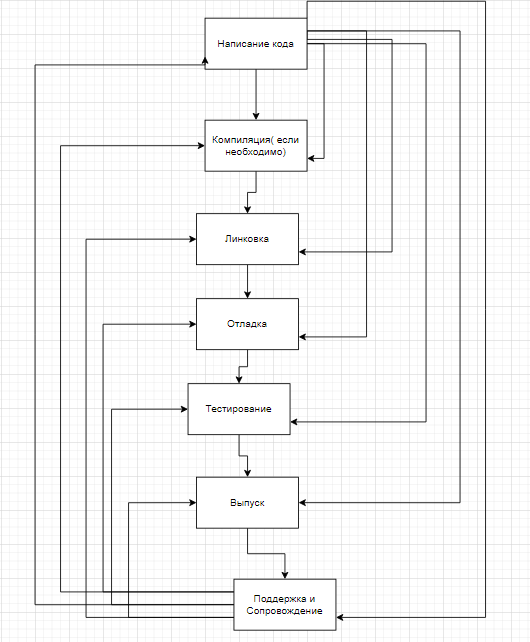
Оборудование: ПК, браузер.

Отчет

Основные понятия темы, схему циклической модели проектирования ПО;

* ПО – совокупность программ и сопровождающей их документации,
* обеспечивающую решение прикладных задач и функционирование аппаратных средств ЭВМ.
* Инструментальные средства разработки ПО – это программные продукты, которые помогают разработчикам создавать, тестировать и поддерживать программное обеспечение. Они автоматизируют многие рутинные задачи, повышая эффективность и качество разработки.
* Циклическая модель проектирования ПО – это подход к разработке, при котором проект развивается поэтапно, через серию итераций. Каждая итерация включает в себя планирование, анализ, проектирование, реализацию, тестирование и оценку. После каждой итерации происходит пересмотр требований и внесений изменений в проект.
* Программирование — сравнительно молодая и быстро развивающаяся отрасль науки и техники. Опыт ведения реальных разработок и совершенствования имеющихся программных и технических средств постоянно переосмысливается, в результате чего появляются новые методы, методологии и технологии, которые, в свою очередь, служат основой более современных средств разработки программного обеспечения. Исследовать процессы создания новых технологий и определять их основные тенденции целесообразно, сопоставляя эти технологии с уровнем развития программирования и особенностями имеющихся в распоряжении программистов программных и аппаратных средств.

Схема процесса описания реализации программного кода с подробным описанием каждого этапа



Этапы

1. **Написание кода:**

* Разработчик создает исходный код программы на выбранном языке программирования, следуя требованиям и спецификациям.
* Из этапа "Написание кода" возможен переход напрямую на этапы "Отладка" и "Тестирование". Это нужно для быстрой проверки отдельных участков кода.

2. **Компиляция (если необходимо):**

* (Только для компилируемых языков, таких как C++, Java). Компилятор преобразует исходный код в объектный код, который может быть выполнен компьютером.
* Если во время компиляции возникают ошибки, возвращаемся к этапу "Написание кода" для их исправления.

3. **Линковка:**

* Объектный код объединяется с необходимыми библиотеками и другими объектными файлами для создания исполняемого файла.
* Если во время линковки возникают ошибки (например, отсутствуют необходимые библиотеки), возвращаемся к этапу "Написание кода", чтобы подключить их.

4. **Отладка:**

* Разработчик использует отладчик для поиска и устранения ошибок в программе.
* Обнаружив ошибку на этапе "Отладка", разработчик возвращается к этапу "Написание кода" для исправления кода, а затем к этапу "Компиляция" (если язык компилируемый) и "Линковка".

5. **Тестирование:**

* Различные тесты (unit-тесты, интеграционные тесты, системные тесты) проводятся для проверки работоспособности и соответствия требованиям программы.
* Если в процессе тестирования выявляются ошибки, то разработчик возвращается к этапу "Отладка" и, возможно, к "Написанию кода".

6. **Выпуск/Деплоймент:**

* Готовая программа развертывается на целевой платформе (сервер, компьютер пользователя, мобильное устройство и т.д.).

7. **Поддержка и Сопровождение:**

* После выпуска программа нуждается в поддержке, исправлении ошибок и добавлении новых функций.
* На основе отзывов пользователей и выявленных проблем, процесс разработки может вернуться к любому из предыдущих этапов, от "Написания кода" до "Проектирования", для внесения необходимых изменений и улучшений.

Состав современных систем программирования

**Современные системы программирования** — это комплекс инструментов и программного обеспечения, предназначенный для создания, тестирования, отладки и оптимизации программного кода.

Состав:

* Редакторы исходного кода – текстовые редакторы с подсветкой синтаксиса и автодополнения (например, Visual Studio Code, Sublime text).
* Компиляторы и интерпретаторы – переводят исходный код в машинный или промежуточный код (GCC, Clang).
* Интегрированные среды разработки (IDE) – объединяют редактор, компилятор, отладчик и инструменты сборки (MSI Visual Studio).
* Отладчики (debugger) – позволяют искать и исправлять ошибки в программе.
* Системы контроля версий – управляют изменениями в исходном коде и координируют работу команды (Git, SVN).
* Средства автоматизации сборки и тестирования – упрощают процесс компиляций, сборки и проверки кода (Make, Jenkins, Maven).
* Средства профилирования – анализируют производительность и расход ресурсов программы.
* Инструменты управления базами данных – обеспечивают взаимодействие с СУБД (SQL Server Management Studio, Oracle SQL Developer).

4. Функции современных компиляторов;

* Разбор кода: лексический, синтаксический и семантический анализ.
* Промежуточное представление (IR) для анализа и оптимизаций.
* Оптимизации: локальные, глобальные, инлайнинг, векторизация, PGO.
* Раcпределение ресурсов: регистрация, спиллы, управление памятью, поддержка GC.
* Генерация кода: машинные инструкции, отладочная и сервисная информация.
* Диагностика и статический анализ: ошибки, предупреждения, отладочные данные.
* Поддержка платформ и интеграция: бэкенды, кросс-компиляция, IDE/CI.
* Безопасность: вставка mitigations, проверки потенциально опасных операций.
* Суть: компилятор не только переводит код в машинный, но и анализирует, оптимизирует и помогает разработке.

Современные средства программирования

|  |  |
| --- | --- |
| Язык | Описание |
| Python | Это высокоуровневый язык программирования, который отличается эффективностью, простотой и универсальностью использования. |
| C++ | Высокая производительность и контроль ресурсов. Сложность и риск ошибок памяти |
| VSS | Система контроля версий (устар.), простота для маленьких команд, ненадёжность и устаревание |
| MS Visual Studio | Мощный отладчик и интеграция, тяжёлая и частично платная |
| Oracle | Реляционная СУБД, масштабируемость и надёжность, высокая стоимость лицензий |
| MS SQL | Реляционная СУБД, интеграция с MS-экосистемой, "Лицензирование, Windows-ориентированность" |
| MySQL | Реляционная СУБД, "Простота, широкая поддержка", меньше корпоративных команд. |

Этапы проектирования приложений

1. Анализ требований - Сбор требований от заказчика и пользователей

* Выявление функциональных и нефункциональных требований (производительность, безопасность, масштабируемость)
* Приоритизация требований и формирование спецификации

2. Проектирование архитектуры

* Выбор архитектурного стиля (монолит, микросервисы, клиент‑сервер, серверлесс и т.д.)
* Определение основных компонентов, интерфейсов и взаимодействий
* Выбор технологий, баз данных и инфраструктуры
* Оценка масштабируемости, отказоустойчивости и безопасности

3. Детальное проектирование (компонентное)

* Разработка диаграмм классов, модулей, API и потоков данных
* Проектирование схемы БД и контрактов API
* Определение форматов сообщений, протоколов и контрактов интеграции

4. Проектирование пользовательского интерфейса

* Создание пользовательских сценариев и макетов
* Проработка UX, навигации и доступности
* Прототипирование и тестирование

5. Проектирование безопасности и соответствия

* Анализ угроз, моделирование атак (например, STRIDE)
* Определение требований аутентификации, авторизации и шифрования
* Соответствие нормативам и политике конфиденциальности

6. Планирование тестирования

* Формирование стратегии тестирования: юнит, интеграционные, системные, нагрузочные, приемочные тесты
* Определение критериев качества и метрик приемки

7. Планирование внедрения и эксплуатации

* Разработка плана развертывания, CI/CD, резервного копирования и восстановления
* Мониторинг, логирование и поддержка (SLA)
* Переход на сопровождение и обучение пользователей

8. Оценка рисков и управление ими

* Идентификация рисков проекта (технические, организационные, временные)
* Разработка мер по снижению и планов на случай непредвиденных ситуаций

9. Документирование

* подготовка архитектурной документации, спецификаций API, эксплуатационных руководств
* Обновление документации по мере изменений

10. Ревью и валидация проектных решений - архитектурные и код

* Ревью, прототипные проверки
* Валидация с заказчиком и корректировка проекта перед.

Нотации и средства для этапа проектирования

Общие принципы:

* Выбор нотаций зависит от цели: архитектурное описание, детальное проектирование, моделирование поведения, проектирование данных, протоколы взаимодействия, UI/UX.
* Нужно сочетать формальные (для точности) и полуструктурированные (для удобства коммуникации) нотации.
* Инструменты выбирают по интеграции в рабочий процесс (CI, репозиторий), поддержке совместной работы и экспортируемости артефактов.

**Нотации:**

* **UML**: универсальная нотация для моделирования ПО.
* **ERD**: схемы для проектирования баз данных.
* **BPMN**: стандарт для моделирования бизнес-процессов.

**Инструменты:**

* **Enterprise Architect**, **StarUML**, **PlantUML**: для UML.
* **MySQL Workbench**, **PowerDesigner**: для проектирования баз данных.
* **Visual Paradigm**: комплексное решение для архитектурного моделирования.

Заключение и выводы

Современные инструментальные средства разработки ПО значительно упрощают и ускоряют процесс создания качественного программного обеспечения. Использование IDE, систем контроля версий, средств автоматизации тестирования и других инструментов позволяет разработчикам сосредоточиться на решении сложных задач и повышении качества кода. Важно постоянно изучать новые инструменты и подходы, чтобы оставаться конкурентоспособным в быстро развивающейся индустрии разработки ПО.