УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию в своей рабочей тетради, законспектируйте основные тезисы, дайте *письменно* ответы на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 16.01.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 2

Тема: Интегрированные среды разработки. Средства разработки: компиляторы, интерпретаторы, компоновщики.

Цель: Познакомится с существующими интегрированными средствами разработки.

Интегрированная среда разработки, ИСР (англ. IDE, Integrated Development Environment или Integrated Debugging Environment) — система программных средств, используемая программистами для разработки программного обеспечения (ПО).

Обычно среда разработки включает в себя:

- текстовый редактор;
- компилятор и / или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для использования при объектно-ориентированной разработке ПО. Хотя и существуют ИСР, предназначенные для нескольких языков программирования — такие, как Eclipse, NetBeans, Embarcadero RAD Studio, Qt Creator или Microsoft Visual Studio, но обычно ИСР предназначается для одного определённого языка программирования - как, например, Visual Basic, PureBasic, Delphi, Dev-C++.

Частный случай ИСР, их эволюционное развитие — среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы.

Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволит разработчику делать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. Однако, так как IDE является

сложным программным комплексом, то лишь после долгого процесса обучения среда разработки сможет качественно ускорить процесс разработки ПО.

Обычно IDE ориентирована на определенный язык программирования, предоставляя набор функций, который наиболее близко соответствует парадигмам этого языка программирования. Однако, есть некоторые IDE с поддержкой нескольких языков, такие как Eclipse, ActiveState Komodo, последние версии NetBeans, Microsoft Visual Studio, WinDev и Xcode.

IDE обычно представляет из себя единственную программу, в которой проводилась вся разработка. Она обычно содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель среды разработки заключается в том, чтобы абстрагировать конфигурацию, необходимую, чтобы объединить утилиты командной строки в одном модуле, который позволит уменьшить время, чтобы изучить язык, и повысить производительность разработчика. Также считается, что трудная интеграция задач разработки может далее повысить производительность. Например, IDE позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках. В то время как большинство современных IDE являются графическими, они использовались еще до того, как появились системы управления окнами (которые реализованы в Microsoft Windows или X11 для *nix-систем). Они были основаны на тексте, используя функциональные клавиши или горячие клавиши, чтобы выполнить различные задачи (например, Turbo Pascal). Использование IDE для разработки программного обеспечения является прямой противоположностью способа, в котором используются несвязанные инструменты, такие как vi (текстовый редактор), GCC (компилятор), и т.п.

Интегрированные среды разработки также часто поддерживают пометки в комментариях в исходном тексте программ, отмечающие места, требующие дальнейшего внимания или предполагающие внесение изменений, такие как TODO. В дальнейшем эти пометки могут выделяться редакторами (напр. vim, emacs, встроенный редактор Visual Studio) или использоваться для организации совместной работы с построением тегов и задач (например, в IntelliJ). Использование комментариев с TODO так же является стандартом оформления кода на Object Pascal, Delphi. Microsoft в руководстве по Visual Studio рекомендует использовать тег TODO (наравне с HACK, UNDONE) для следующих пометок:

- добавление новых функций;
- известных проблем, которые нужно устранить;
- предполагаемых к реализации классов;
- мест размещения кода обработчиков ошибок;
- напоминаний о необходимости переработки участка кода.

Обычно интегрированная среда разработки - это совокупность программных средств, поддерживающая все этапы разработки программного обеспечения от написания исходного текста программы до ее компиляции и отладки, и обеспечивающая простое и быстрое взаимодействие с другими инструментальными средствами (программным отладчиком-симулятором, внутрисхемным эмулятором, эмулятором ПЗУ и программатором).

Строго говоря, интегрированные среды разработки не относятся к числу средств отладки. Отладка – лишь одно из свойств интегрированных сред, которые представляют собой основу любой визуальной среды разработки или RAD-среды.
При традиционном подходе, начальный этап написания программы строится следующим образом:

1. Исходный текст набирается при помощи какого-либо текстового редактора.
2. По завершении набора, работа с текстовым редактором прекращается и запускается кросс компилятор.
3. Как правило, вновь написанная программа содержит синтаксические ошибки, и компилятор сообщает о них на консоль оператора.
4. Вновь запускается текстовый редактор, и оператор должен найти и устранить выявленные ошибки, при этом сообщения о характере ошибок выведенные компилятором уже не видны, так как экран занят текстовым редактором.

И этот цикл может повторяться не один раз. Если программа имеет большой объем, собирается из различных частей, и подвергается длительному редактированию или модернизации, то даже этот начальный этап может потребовать много сил и времени. После этого наступает этап отладки программы и к редактору с компилятором добавляется эмулятор или симулятор, за работой которого хотелось бы следить прямо по тексту программы в текстовом редакторе.
Интегрированные среды (оболочки) разработки (Integrated Development Environment, IDE) позволяют избежать большого объема однообразных действий и тем самым существенно повысить эффективность процесса разработки и отладки позволяют, то есть они являются RAD-средами различной степени автоматизации процесса программирования.
Работа в интегрированной среде дает программисту:

- Возможность использования встроенного многофайлового текстового редактора, специально ориентированного на работу с исходными текстами программ;
- Иметь автоматическую диагностику выявленных при компиляции ошибок, когда исходный текст программы, доступный редактированию, выводится одновременно с диагностикой в многооконном режиме;
- Возможность параллельной работы над несколькими проектами. Менеджер проектов позволяет использовать любой проект в качестве шаблона для вновь создаваемого проекта;
- Минимум перекомпиляции. Ей подвергаются только редактировавшиеся модули;
- Возможность загрузки отлаживаемой программы в имеющиеся средства отладки, и возможность работы с ними без выхода из оболочки;
- Возможность подключения к оболочке практически любых программных средств.

В последнее время, функции интегрированных сред разработки становятся стандартной принадлежностью программных интерфейсов эмуляторов и отладчиков-симуляторов.
Подобные функциональные возможности, в сочетании с дружественным интерфейсом, в состоянии существенно увеличить скорость разработки программ, особенно для микроконтроллеров и процессоров цифровой обработки сигналов, являющихся очень трудоемкими и труднообозримыми процессами.

Методология проектирования

В реальных условиях проектирование - это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

Системный подход - любая система представляет собой совокупность взаимосвязанных элементов, функционирующих совместно для достижения общей цели.

Метод проектирования - огранизационная совокупность процессов создания ряда моделей, которые описывают различные аспекты создаваемой системы с использованием четко определенной нотации.

Технология проектирования - совокупность технологических операций в их последовательности и взаимосвязи, приводящая к разработке проектной системы.

Информационное обеспечение - совокупность единой классификации и кодирования информации, унифицированных документов и вспомогательных информационных массивов (обычно - классификаторы, таблицы кодирования и пр.)

Техническое обеспечение - комплекс технических средств (gg, тут переключили слайд)

Математическое обеспечение - совокупность математических методов, моделей и алгоритмов, использующихся для управления системой и реализации прикладных задач.

Лингвистическое обеспечение - множество языков, использующихся при разработке и эксплуатации ИС (языки программирования, языки общения, набор словарей)

Правовое обеспечение - совокупность правовых норм, определяющих создание, юридических статус и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации.

- Метод "снизу - вверх" ("лоскутная автоматизация")
- Метод "сверху - вниз"
- Метод многокомпонентности
- Технология проектирования DATARUN
- Технология проектирования RUP

Классификация инструментального программного обеспечения

Инструментальное программное обеспечение – это программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ.

Инструментальные средства можно разбить на 4 группы:

1. Необходимые инструментальные средства
   Редакторы текста, компиляторы и ассемблеры (компьютерные программы, осуществляющие преобразование программы в форме исходного текста на языке ассемблера в машинные команды в виде объектного кода), компоновщики (программы, которые принимают на код один или несколько объектных модулей и собирают по ним исполняют модуль) или редакторы связи
2. Частоиспользуемые инструментальные средства
   Это средства, использование которых в процессе разработки весьма облегчает и ускоряет процесс разработки. Например, утилиты автоматической сборки проекта, отладчики, программа создания файлов помощи
3. Специализированные инструментальные средства используются в исключительных случаях и решают специфические задачи
4. Интегрированные среды разработки содержат большую часть из приведенных выше программ и позволяет упростить процесс создания приложений

Компиляторы, интерпретаторы и компоновщики

Компиляторы и *интерпретаторы* преобразуют исходный текст, написанный на языке программирования, в машинный код (более точнее в объектный код).

Компилятор преобразует сразу весь исходный текст, на выходе получается один или несколько файлов объектного кода, а в некоторых языках программирования процесс компиляции может управляться так называемыми *директивами компиляции* (это специальные указания о том, как компилировать отдельные блоки текста).

Этапы компиляции:

1. Лексический анализ. На этом этапе весь текст исходного файла преобразуется в последовательность лексе
2. Синтаксический анализ. На данном этапе последовательности лексин преобразуется в дерево разбора
3. Семантический анализ. На данном этапе осуществляется проверка по смыслу. Дерево разбора для установления его смысла
4. Оптимизация. На этом этапе выполняется удаление лишних конструкций и упрощение кода с сохранением его смысла
5. Генерация кода. На данном этапе из промежуточного представления порождается код на целевом языке В результате компиляции получаем объектный код.

Интерпретатор преобразует текст на языке высокого уровня, но делает это построчно. Например, любой интернет-браузер, который преобразует HTML и CSS «на лету».

Основными достоинствами интерпретаторов являются:

1. Большая переносимость интерпретируемых программ
2. Более совершенные и наглядные средства диагностики
3. Упрощение отладки исходного кода
4. Меньшие размеры кода по сравнению с машинным кодом

Основными недостатками интерпретаторов являются:

1. Интерпретируемая программа не может выполняться отдельно без программы-интерпретатора
2. Интерпретируемая программа выполняется медленно
3. Практически отсутствует оптимизация

Компоновщики (linker) занимается созданием готового исполняемого файла из кусочков объектного кода. Чаще всего компоновщик включается в компилятор (они идут парами). Для связывания модулей компоновщик использует таблицы имен, созданную компилятором в каждом из объектных модулей. Такие имена могут быть двух типов: определенные (экспортируемые) и неопределенные (импортируемые). Работа компоновщика заключается в том, что определить и связать ссылки на неопределенные имена в каждом модуле. Обычно компоновщик не выполняет проверку типов и количество параметров процедур и функций.

Контрольные вопросы:

1.Что представляет собой Технология визуального программирования?

2.Что представляет собой Интегрированная среда разработки?

3.Приведите классификацию инструментального ПО.