№1

Этапы технологического цикла создания ПП:
1. Анализ (определение) требований
2. Проектирование
3. Написание текста программ (программирование, “кодирование”)
4. Компоновка или интеграция программного комплекса
5. Верификация, тестирование и отладка
6. Документирование
7. Внедрение
8. Тиражирование
9. Сопровождение, повторяющее все предыдущие этапы

№2

Система программирования (СП) — это комплекс программных инструментов и библиотек, который поддерживает весь технологический цикл создания программного продукта (ПП).

ПП — программа, оформленная, документированная и специфицированная таким образом, что ее можно использовать независимо, отчужденно от автора программы.

Основные компоненты системы программирования:
1. Транслятор (переводит программы с языка программирования на машинный язык, что и позволяет выполнить их на ЭВМ).
2. Макрогенератор или макропроцессор (работает непосредственно перед транслятором, используется для получения макрорасширения исходной программы).
3. Редактор текстов (используется для составления программ на языке программирования).
4. Редактор связей или компоновщик (предназначен для связывания между собой (по внешним данным) объектных файлов, порождаемых компилятором, а также файлов библиотек, входящих в состав СП).
5. Отладчик (используется для проверочных запусков программ и исправления ошибок).
6. Библиотеки стандартных программ (облегчают работу программиста, используются на этапе трансляции и исполнения).

№3

Стратегии трансляции:
1. Компиляторы и ассемблеры (переводят программу в объетный файл, который не зависит непрерывно от входных данных)
2. Интерпретаторы (выполняют программу, непрерывно используя входные данные)
3. Смешанная стратегия (байт-код, JIT-компиляция)

№4

Общая схема работы компилятора:
1. Начальные установки
   Фазы анализа программ:
1. Лексический анализ

2. Синтаксический и семантический анализ (контроль контекстных условий)
Фазы оптимизации программ
Фазы синтеза программ:
1. Распределение памяти
2. Распределение регистров
3. Генерация команд и машинно-зависиая оптимизация

№32

Основные стратегии распределения памяти:
Память бывает:
1. Локальная
2. Глобальная
3. Динамическая:
а) стековая дисциплина
б) дисциплина произвольного распределения:
- распределённая программистом
- распределённая компилятором
4. Стековая

Стратегии распределения:
1. Стратегия статического распределения
Память автоматически распределяется в статических областях компилятором. Размеры объектов, размещаемых в этих областях, никогда не меняются, как и та часть адресов этих объектов, которая указывает их положение внутри области. Единственное, что может измениться – это начальный адрес самой области, но и это изменение происходит до выполнения программы, перед загрузкой ее в память.

2. Стратегия динамического распределения
Выбирается в тех случаях, когда на стадии компиляции не удается определить положение объекта в некоторой области памяти и/или его размер. При динамическом распределении возможно применение различных дисциплин, наиболее известны из которых стековая дисциплина распределения и дисциплина произвольного распределения (распределение в “куче”).

№33

Машинно-независимая и машинно-зависимая оптимизация. Примеры оптимизирующих преобразований.

Машинно-независимая оптимизация:
1. Вычисление выражений из констант на стадии компиляции,
2. Арифметические преобразования,
3. Устранение общих подвыражений (избыточных вычислений),
4. удаление ненужных присваиваний и других операций, распространение копий значений,
5. перестановка независимых смежных участков программ,
6. удаление недостижимых фрагментов программы,
7. оптимизация вычисления логических выражений.

Машинно-зависимая оптимизация:
1. учет регистровой структуры вычислительной аппаратуры,

2. удаление излишних команд,
3. оптимизация потока управления и удаление недостижимых участков
программ,
4. снижение “стоимости” программы,
5. использование машинных идиом,
6. слияние, дробление и развертывание циклов, иногда требующееся из-за
7. технических особенностей аппаратуры,
8. учет векторных и конвейерных свойств архитектуры.

№34

ИСР (Интегрированная среда разработки, IDE, integrated development environment) — комплекс программных средств, поддерживающих полный
жизненный цикл ПП.

№35

Редакторы называются текстовыми, поскольку они предназначены для редактирования и хранения в архиве любой текстовой информации – от текстов программ до документации по вычислительной системе. Текстовые редакторы делятся на две категории по видам запуска, они бывают пакетными и диалоговыми.

Возможности текстового редактора в ИСР:
1. Подготовка текста программы (обычные действия по созданию, редактированию, сохранению файла с текстом программы).
2. Многооконный интерфейс, управление окнами и вкладками.
3. Закладки, настраиваемые сочетания клавиш, шаблоны фрагментов текста, программное управление самим редактором
4. Интеграция с компилятором и средствами статического анализа кода.
5. Интеграция с отладчиком.

№36

Задачи отладчика в рамках ИСР
1. пошаговое выполнение программы
2. выполнение программы до строки, в которой в редакторе стоит
курсор,
3. выделение выполняемой в данный момент строки,
4. приостановка выполнения программы для просмотра состояния программы
5. расстановка/снятие точек останова, которые визуализируются в
текстовом редакторе,
6. выдача всей информации в терминах исходной программы.

Отладчик может быть интегрирован совместно с текстовым редактором

№37

Редактор внешних связей, его назначение и принципы работы. Загрузчик.

Основное назначение редактора связей – завершить ту часть работы, которая принципиально не могла быть выполнена компилятором, а именно, осуществить привязку нескольких модулей друг к другу. Компилятор не мог выполнить такую привязку потому, что он всегда

работает только с одной компилируемой программой, зная о других программных компонентах только то, что они должны существовать, а также их программные интерфейсы.

Основные задачи редактора связей таковы:
• связывание между собой по внешним данным объектных модулей, порождаемых компилятором и составляющих единую программу;
• подготовка таблицы трансляции относительных адресов для загрузчика;
• статическое подключение библиотек с целью получения единого исполняемого модуля;

Редактор связей должен заново прочитать все объектные модули (как откомпилированные, так и библиотечные), необходимые для формирования одной
полной программы, и выявить в них все упоминания внешних объектов (процедур, функций, констант, переменных и т. д.).

Работа редактора связей заканчивается формированием собранной программы, для которой остались неизвестными лишь начальные адреса размещения разделов памяти. Ни компилятор, ни редактор связей не в состоянии знать, в какой именно области физической памяти будет размещаться программа в момент ее выполнения. Эти компоненты работают лишь с относительными адресами, которые отсчитываются от некоторой условной точки (обычно она совпадает с началом разделов памяти, отводимых для объектов самого первого модуля, поданного для компоновки редактору связей). Задача следующего преобразования – преобразовать условные адреса разделов памяти в истинные (абсолютные). Такое преобразование выполняется загрузчиками.

Загрузчики могут включаться в состав систем программирования, но чаще они оказываются составными частями операционных систем, поскольку выполняемые ими функции не только зависят от архитектуры вычислительной системы, в которой должна выполняться программа, но также и от конкретной физической конфигурации этой системы

№38

Профилировщики. Назначение. Принцип работы.

Один из способов повысить эффективность работы программы – провести ее профилирование, то есть определить время, затрачиваемое на выполнение отдельных ее фрагментов. Профилировщик аналогичен программному логическому анализатору, способному выдавать огромные объемы информации. Профилировщик позволяет разработчику точно настраивать поведение системы в условиях реальной эксплуатации и визуализировать события для быстрого обнаружения проблемы.

№39

Библиотеки. Основные типы библиотек.

Библиотеки делятся на статические и динамические по выбору того момента, когда система программирования извлекает из них элементы.

Статические библиотеки фактически представляют собой процедуры и функции, встраиваемые внутрь программ, подготавливаемых системами программирования на этапе обработки этих программ. Все статические библиотеки подключаются к программам, готовящимся к выполнению, ровно один раз в момент формирования редактором связей полной программы.

Компоненты динамических библиотек (динамически загружаемые компоненты и библиотеки) подключаются к программам во время выполнения этих программ. В этом состоит основное отличие динамических библиотек от статических. Компоненты динамических библиотек не связаны с программами, которые к ним обращаются, и распространяются отдельно от них.

По функциональному наполнению все используемые в составе современных
систем программирования библиотеки можно классифицировать следующим образом:
• библиотеки функций, процедур и макроопределений
• библиотеки классов,
• библиотеки компонентов

№40

Критерии проектирования стандартных библиотек.

Стандартная библиотека языка программирования является в настоящее времяобязательнойчастьюсистемной библиотеки. Все современные языки программирования нуждаются в поддержке имеющихся в них средств, причем поддержка им необходима именно в период выполнения программ, полученных путем компиляции с этих языков. При проектировании любой стандартной библиотеки необходимо принимать во внимание ее основное назначение: быть именно стандартной библиотекой, то есть библиотекой средств, необходимых для каждой реализации данного языка программирования.

Требования:
1. обеспечивать поддержку свойств языка
2. предоставлять информацию о зависящих от реализации аспектах языка
3. предоставлять функции, которые не могут быть написаны оптимально для всех вычислительных систем на данном языке программирования, например, функции вычисления квадратного корня sqrt() или пересылок блоков памяти memmove();
4. предоставлять программисту нетривиальные средства, на которые он может рассчитывать, заботясь о переносимости программ, например, средства работы со списками, функции сортировки, потоки ввода/вывода;

№41

Способы подключения библиотек функций.

1. статическое подключение библиотек с целью получения единого исполняемого модуля (выполняет компоновщик(редактор связей))
2. динамическое подклчение библиотек (динамическая загрузка (выполняет загрузчик))