ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования

Тихоокеанский государственный университет

УТВЕРЖДАЮ

Проректор по учебной работе

_____ С.В. Шалобанов

“_____” _____2007 г.

ПРОГРАММА ДИСЦИПЛИНЫ
по кафедре Вычислительной техники

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Утверждена научно-методическим советом университета для направлений подготовки (специальностей) в области «**Информатики и вычислительной техники»**

Специальность 230101.65
«Вычислительные машины, комплексы, системы и сети»

Хабаровск 2007 г.

Программа разработана в соответствии с требованиями государственного образовательного стандарта, предъявляемыми к минимуму содержания дисциплины и в соответствии с примерной программой дисциплины, утвержденной департаментом образовательных программ и стандартов профессионального образования с учетом особенностей региона и условий организации учебного процесса Тихоокеанского государственного технического университета.

Программу составил (и)

Миронов А. С кафедра Вычислительной техники, преподаватель

_____ _____

Программа рассмотрена и утверждена на заседании кафедры
протокол №_16_ от «_13_»_сентябрь_2007г.

Завкафедрой_____«__»_____ 2007г _____
Подпись дата Ф.И.О.

Программа рассмотрена и утверждена на заседании УМК и рекомендована к изданию
протокол №_6_от «_14_»_сентябрь_2007г

Председатель УМК _____«__»_____ 2007г _____
Подпись дата Ф.И.О.

Директор института _____«__»_____ 2007г _____
(декан факультета) Подпись дата Ф.И.О.

1. Цели и задачи дисциплины

Основной целью и задачей курса «Технологии программирования» ставится изучение студентами процесса проектирования и разработки программного обеспечения (ПО), с использованием современных принципов и правил программирования, присущих различным технологиям программирования, применяя системы автоматического проектирования и разработки. А также получения навыков по продвижению, сопровождению и защите авторского права разработанного программного обеспечения.

Студенты специальности 230101.65, должны изучить следующие обязательные разделы: задача проектирования программных систем; организация процесса проектирования ПО; использование декомпозиции и абстракции при проектировании ПО; специфики процедур и данных; декомпозиция системы; методы проектирования структуры ПО; методология объектно-ориентированного программирования; технологические средства разработки программного обеспечения: инструментальная среда разработки, средства поддержки проекта, отладчики; методы отладки и тестирования программ; документирование и оценка качества программных продуктов; методы защиты программ и данных; проектирование интерфейса с пользователем; структуры диалога; поддержка пользователя; многооконные интерфейсы; примеры реализации интерфейсов с пользователем с использованием графических пакетов.

Изучение дисциплины базируется на знаниях, полученных при изучении дисциплин «Информатика», «Основы алгоритмизации и программирования», «Системное программное обеспечение», «Операционные системы».

2. требования к уровню освоения содержания дисциплины

В результате изучения дисциплины студент должен:

- знать

теоретические основы по применению современных технологий программирования и правил анализа, проектирования и разработки программного обеспечения;

-уметь

проектировать и разрабатывать различные виды программного обеспечения на основе современных технологий (ООП, MPI, NET, ASP и тд) с использованием сред разработки и поддержки процесса создания ПО;

-иметь опыт

по реализации унифицированного процесса создания и описания ПО различной сложности с применением унифицированного языка моделирования (UML), на различных этапах проектирования ПО.

-иметь представление

о библиотеках классов, шаблонах проектирования и инструментальных средствах применяемых при разработке программного обеспечения;

о тенденциях и перспективах развития современных технологий разработки ПО.

3. Объём дисциплины и виды учебной работы.

Таблица 1.

Наименование	По учебным планам (УП)	
	С максимальной трудоёмкостью	С минимальной трудоёмкостью
Общая трудоёмкость дисциплины		
по ГОС	170	170
по УП	176	176
Изучается в семестрах	9	9
Вид итогового контроля по семестрам		
зачет	10	10
экзамен	6	6
Курсовой проект (КП)		
Курсовая работа (КР)		
Расчетно-графические работы (РГР)		
Реферат (РФ)		
Домашние задания (ДЗ)		
Аудиторные занятия:		
всего	96	96
В том числе: лекции (Л)	48	48
Лабораторные работы (ЛР)	32	32
Практические занятия (ПЗ)	16	16
Самостоятельная работа		
общий объем часов (С2)	80	80
В том числе на подготовку к лекциям	16	16
на подготовку к лабораторным работам	32	32
на подготовку к практическим занятиям	32	32
на выполнение КР		
на выполнение РГР		
на написание РФ		
на выполнение ДЗ		

4. Содержание дисциплины

№	Тема	Наименование тем лекционного курса
1	Введение. Технология программировании.	Определение технологии программирования. Программы и программное средство (ПС). Причины ошибок в ПС. Надежность ПС.
2	Современные подходы в разработке программного обеспечения.	Современные подходы в разработке программного и решения задачи проектирования программных систем. Организация процесса проектирования программного обеспечения (ПО). Жизненный цикл ПС.
3	Руководство программным проектом.	Руководство программным проектом. Команда разработчиков проекта. Классические методы анализа.
4	Средства разработки программного обеспечения	Технологические средства разработки программного обеспечения. Средства поддержки проекта. Отладчики. Средства тестирования и автоматизации. Средства подготовки инсталляционных версий.
5	Парадигмы и правила программирования	Парадигмы и правила программирования. Шаблонное программирование.
6	Объектно-ориентированное программирование(ООП)	Методология ООП. Основы объектно-ориентированного представления программных систем. Принципы. Объекты. Классы. MFC
7	Паттерны проектирования.	Паттерны проектирования. Назначение и применение. Основные, порождающие, структурные и поведенческие паттерны. Библиотека шаблонов STL.
8	Основы проектирования программных систем.	Структурирование. Моделирование. Декомпозиция. Модульность. Абстракция. Классические методы проектирования.
9	Внешнее описание ПС.	Понятие внешнего описания. Определение требований к ПС. Спецификации качества ПС. Устойчивость ПС. Защищенность ПС. Коммуникабельность ПС. Функциональная спецификация ПС. Современные методы описания.
10	Унифицированный язык моделирования UML.Статические модели	Отношения. Диаграммы. Механизмы расширения. Статическое и представление данных.
11	Унифицированный язык моделирования UML.Динамические модели	Динамические модели. Модели реализации объектно-ориентированных программных систем.
12	Тестирование программного обеспечения.	Понятие тестирования. Структурное. Функциональное. Объектно-ориентированное тестирование. Организация процесса тестирования программного обеспечения. Оценка трудозатрат. Автоматизирование тестирование программного обеспечения.
13	Отладка.	Понятие отладки. Жизненный цикл ошибок. Систематика ошибок. Методики наладки. Постналадка и предналадка.
14	Методы защиты программ и данных.	Внутренняя защита программы. Методы защита от взлома и несанкционированного доступа. Пароли. Открытый и закрытый ключ. Информационная безопасность.
15	Рефакторинг.	Понятие рефакторинга. Понятие качества ПО. Обеспечение функциональности и надежности программного средства. Номенклатура показателей качества. Сертификация ПО.
16	Документирование ПО	Документирование ПО. Документация пользователя. Документация по сопровождению.
17	Пользовательский интерфейс	Процесс разработки и проектирования пользовательского интерфейса. Качество. Модели. Правила. Стандарты и руководящие принципы. Когнитивный подход. Поддержка пользователя. Тестирование на удобство применения.
18	Виды графического пользовательского интерфейса	Графический пользовательский интерфейс. Объектно-ориентированный пользовательский интерфейс. Многооконные интерфейсы. Диалоговые интерфейсы. Меню.
19	Проектирование Web приложений	Правила проектирования Web-приложений. Методы и технологии разработки пользовательского Web - интерфейса
20	Унифицированный процесс разработки (УПР).	Понятие УПР. Вариантно–управляемый УПР. Архитектурно-ориентированный УПР. Итеративный УПР. Инкрементный УПР.
21	Экстремальное программирование.	Гибкие методологии разработки ПО. Оснвные практики экстремального программирования.
22	Авторское право	Разработка и продвижение программ. Защита программ.
23	Распределенные вычисления и	Сравнительный обзор технологий MPI и CORBA.

	программирование.	
24	Платформа Microsoft Net Framework.	Архитектура. Управляемый и неуправляемый код. MSIL. JIT-компилятор. Язык C#. Технология ASP.NET.

Разделы дисциплины и виды занятий и работ

№	*Раздел дисциплины*	*Л*	ЛР	ПЗ	КП (КР)	РГР	ДЗ	РФ	С2
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1.	Введение. Технология программировании.	*							*
2.	Современные подходы в разработке программного обеспечения.	*	*	*					*
3.	Руководство программным проектом.	*	*					*	*
4.	Средства разработки программного обеспечения	*	*	*				*	*
5.	Парадигмы и правила программирования	*	*						*
6.	Объектно-ориентированное программирование (ООП)	*	*	*					*
7.	Паттерны проектирования.	*	*					*	*
8.	Основы проектирования программных систем.	*		*					*
9.	Внешнее описание ПС.	*		*					*
10.	Унифицированный язык моделирования UML.Статические модели	*		*					*
11.	Унифицированный язык моделирования UML.Динамические модели	*		*					*
12.	Тестирование программного обеспечения.	*	*	*				*	*
13.	Отладка.	*	*	*					*
14.	Методы защиты программ и данных.	*						*	*
15.	Рефакторинг.	*	*						*
16.	Документирование ПО	*		*					*
17.	Пользовательский интерфейс	*	*						*
18.	Виды графического пользовательского интерфейса	*	*						*
19.	Проектирование Web приложений	*	*					*	*
20.	Унифицированный процесс разработки (УПР).	*	*						*
21.	Экстремальное программирование.	*						*	*
22.	Авторское право	*	*					*	*
23.	Распределенные вычисления и программирование.	*	*					*	*
24.	Платформа Microsoft Net Framework.	*	*					*	*

5.1.Лабораторный практикум

1. Практика программирования. Стили. Рефакторинг

Цель работы: Получение углубленных навыков в практике написания кода программ.

Исполнение: В ходе данной лабораторной работы обучающийся должен выполнить различные упражнения направленные на улучшение стилистики написания программных кодов. Научиться, при визуальном исследовании кода, выявлять нарушения стилистики кода программ по заданным правилам. Получить представление о рефакторинге кода. Выполнить упражнения по применению рефакторинга.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005

Оценка:.В результате выполнения лабораторной работы обучающийся должен предоставить отчет о проделанной работе с выполненными упражнениями. Уметь свободно ориентироваться в предоставленном отчете. Устно ответить на вопросы по ходу выполнения лабораторной работы.

Время выполнения работы: 3 часа.

2. STL-библиотека шаблонов.

Цель работы: Знакомство с библиотекой шаблонов STL.

Исполнение: 1. Обучающийся должен написать программу на языке C++ с использованием библиотеки абстрактных типов данных Stl (например программу реализующую и использующую стек, элементами которого являются структуры). В процессе написания программы обучающийся должен использовать четыре основных компонента составляющие структуру STL библиотеки: итераторы, алгоритмы, контейнеры и функциональные объекты. 2. Обучающийся должен составить диаграмму объектов для разрабатываемой им программы на языке UML . 3. Составить функциональное описание программы.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose

Оценка:. Обучающийся должен устно ответить на вопросы по составу и функциональным возможностям библиотеки STL, продемонстрировать разработанную им программу с использованием компонентов библиотеки STL, обосновать целесообразность применения им выбранных компонентов. Оценивается полнота предоставленного отчета.

Время выполнения работы: 3 часа.

3. Паттерны проектирования.

Цель работы: Научиться на практике применять паттерны проектирования при реализации ПО с использованием технологии ООП.

Исполнение: . 1. Изучение принципов, свойств и области применения простых паттернов проектирования «Абстрактная фабрика», «Декоратор», «Компоновщик», «Адаптер» и тд. 2.Обучающийся должен написать программу, используя технологию ООП с применением одного или нескольких паттернов проектирования (каталог паттернов предоставляется преподавателем). 3. Обучающийся должен составить диаграмму объектов для разрабатываемой им программы на языке UML . 4. Составить функциональное описание программы.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose

Оценка: Обучающийся должен устно ответить на вопросы по теории применения паттернов проектированию, продемонстрировать разработанную им программу с использованием паттернов проектирования, обосновать целесообразность применения им выбранных паттернов проектирования. Оценивается полнота предоставленного отчета.

Время выполнения работы: 4 часа.

4. Автоматизированные средства тестирования.

Цель работы: *Получить навыки использования автоматизированного средства тестирования nUnit*

Исполнение: В ходе выполнения данной лабораторной работы обучающийся должен изучить основные функции пакета nUnit, используя прилагающиеся статьи. Выполнить упражнения направленные на закрепление полученных знаний.

Обеспечение:Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005

Оценка: В результате выполнения лабораторной работы обучающийся должен предоставить отчет о проделанной работе с выполненными упражнениями. Уметь свободно ориентироваться в предоставленном отчете. Устно ответить на вопросы по ходу выполнения лабораторной работы.

Время выполнения работы: 2 часа.

5. Система контроля версий Visual Source Safe.

Цель работы: Знакомство со средой контроля версий Visual Source Safe 6.0d

Исполнение: С помощью системы Visual Source Safe 6.0d организовать структуру хранения версий кода одного программного модуля и различных программных модулей одного приложения, научиться просматривать изменения, сделанные от версии к версии для одним или несколькими пользователями.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005

Оценка:. В результате выполнения обучающийся должен продемонстрировать работу написанных приложений. Показать уверенное пользование системой контроля версий Visual Source Safe для решения задач управления параллельной разработкой. Устно ответить на вопросы по ходу выполненной работы и предоставленному отчету. Также оценивается полнота отчета.

Время выполнения работы: 2 часа.

6. Система контроля версий CVS.

Цель работы: Знакомство со средой контроля версий CVS.

Исполнение: Ознакомление с документацией на систему управления параллельными версиями. С помощью системы CVS организовать структуру хранения версий кода одной программы, научиться обращаться к любой версии этой программы, просматривать изменения, сделанные от версии к версии.

Обеспечение: Персональный компьютер; Операционная система Linux. Компилятор gcc.

Оценка: В результате выполнения обучающийся должен продемонстрировать работу написанных приложений. Показать уверенное пользование системой контроля версий CVS для решения задач управления параллельной разработкой. Устно ответить на вопросы по ходу выполненной работы и предоставленному отчету. Также оценивается полнота отчета.

Время выполнения работы: 2 часа.

7. Web приложения. ASP. NET

Цель работы: Получение навыков по созданию web-приложений с использованием ASP технологии.

Исполнение:. В ходе выполнения данной лабораторной работы обучающийся должен изучить основные принципы построения web-приложений с использованием технологии ASP.NET, а также изучить методы и технологии разработки пользовательского Web – интерфейса.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005

Оценка:. В результате выполнения обучающийся должен продемонстрировать работу написанных приложений. Устно ответить на вопросы по ходу выполненной работы и предоставленному отчету.

Время выполнения работы: 4 часа.

8. Аналитические исследования программного обеспечения.

Цель работы: Получение навыков в аналитическом исследование программного продукта стороннего разработчика.

Исполнение:. Подготовка статьи на тему «Аналитическое исследование программного продукта». Статья должна быть выполнена по представленному в методическом указании на лабораторную работу плану.

Обеспечение: Персональный компьютер; Операционная система Windows, Linux. Internet. Текстовый редактор. Исследуемая программа.

Оценка:. Оценка на соответствие и полноту изложения предоставленной к защите статьи по аналитическому исследованию программного продукта. Обучающийся должен устно ответить на вопросы по ходу выполнения лабораторной работы.

Время выполнения работы: 4 часа.

9. Авторское право.

Цель работы: Изучение основ авторского права РФ.

Исполнение:. Ознакомиться с теоретическими сведениями по авторскому праву в РФ (Закон Российской Федерации о правовой охране программ для электронных вычислительных машин и баз данных №3523-1 от 23 сентября 1992 года). Подготовка бланков для регистрации программного продукта (ИКАП, РТО). Получение выписки о регистрации. Защита работы.

Обеспечение: Персональный компьютер; Операционная система Windows.

Оценка:. Для защиты лабораторной работы обучающийся должен успешно пройти процедуру регистрации. Предоставить отчёт о проделанной работе. Показать достаточный уровень знаний по содержанию работы. Ответить на устные вопросы.

Время выполнения работы: 4 часа.

10. Знакомство с библиотекой MPI 1.0

Цель работы: Знакомство с библиотекой MPI версии 1.0. Получение первоначальных навыков работы с библотекой MPI, изучение функций, предоставляемых библиотекой для организации передачи сообщений для параллельно выполняющихся программ.

Исполнение:. 1.Необходимо написать программу, использующую систему неблокирующей передачи сообщений. 2. Необходимо написать программу, использующую систему блокирующей передачи сообщений. В итоге сравнить результаты времени выполнения разработанных программ для разного количества используемых процессоров и различных входных данных программы. Программы должны реализовывать вычисления интеграла с использованием различных численных методов заданной точностью, при этом входными данными программ будут являться: границы интервала, шаг внутри интервала.

Обеспечение: Персональный компьютер; Операционная система Linux, Windows. Компилятор gcc, компилятор C++. Среда MPICH, библиотека MPI.

Оценка: В результате выполнения обучающийся должен продемонстрировать работу написанных программ при заданных преподавателем входных данных, представить в графическом виде результаты, содержащий информацию о времени выполнения программ, времени выполнения алгоритма на разном количестве компьютеров и времени пересылке данных между компьютерами.

Время выполнения работы: 4часа.

Лабораторные занятия и их взаимосвязь с содержанием лекционного курса

№ п/п	№ раздела по варианту содержания	Наименование лабораторной работы
1	5,15	Практика программирования. Стили. Рефакторинг.
2	5,7	STL-библиотека.
3	5,7	Паттерны проектирования
4	3,2,12,13,20	Автоматизированные средства тестирования
5	2,4	Система контроля версий Visual Source Safe
6	2,4	Система контроля версий CVC
7	17,18,19,24	Web-приложения. ASP.NET
8	2,5,6,8,15,17,18	Аналитические исследования программного обеспечения.
9	22	Авторское право
10	23	Знакомство с библиотекой MPI 1.0

5.2. Практические занятия

Практические задания выполняются при изучении дисциплины "Технология программирования" и имеют целью выработку у студентов навыков в трех направлениях: 1. Применение соответствующих методологий для разработки информационных систем и программного обеспечения; 2. Применение языка UML для моделирования и проектирования информационных систем; 3. Применение программного инструментария - Rational Rose.

1. **Варианты использования и действующие лица**

Цель работы: Научится создавать диаграмму Вариантов Использования.

Исполнение: 1.Создать диаграмму Вариантов Использования, задать варианты использования и действующих лиц. 2. Создать абстрактный вариант использования. 3. Добавить ассоциации. 4. Добавить связи расширения. 5. Добавить описания к вариантам использования. 6. Добавить описания к действующим лицам. 7. Прикрепить файл к варианту использования.8. Сохранить файл модели, подготовить отчет.

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose

Оценка:. Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.

Время выполнения работы: 2 часа.

2. **Взаимодействие объектов. Поведение объектов**

Цель работы: Разработать диаграммы Последовательностей и Кооперации.

Исполнение: 1. Создать диаграмму Последовательностей: настройка, создание диаграммы Последовательности, добавление на диаграмму действующего лица и объектов, добавление сообщений на диаграмму, добавление на диаграмму дополнительных объектов, назначение ответственностей объектам, соотнесение объектов с классами, соотнесение сообщений с операциями
2. Создать диаграмму Кооперации: создание корпоративной диаграммы, добавление действующего лица и объектов на диаграмму, добавление сообщений на диаграмму, добавление на диаграмму дополнительных объектов, назначение ответственностей объектам, соотнесение сообщений с операциями, соотнесение объектов с классами
3. Сохранить файл модели и подготовить отчет

Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose

Оценка: Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.
Время выполнения работы: 4 часа.

3. **Классы и пакеты**
Цель работы: **Научиться создавать диаграммы классов и пакетов.**
Исполнение: 1.Создание пакетов.2. Создание Главной диаграммы Классов.3. Создание диаграммы Классов для сценария "Ввести новый заказ" с отображением всех классов. 4. Добавление стереотипов к классам. 5. Объединение классов в пакеты. 6. Добавление диаграмм Классов к каждому пакету. 7. Сохранение файл модели, составление отчета
Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose
Оценка:. Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.
Время выполнения работы: 2 часа.

4. **Атрибуты и операции. Связи.**
Цель работы: **Научиться определять и назначать параметры и типы возвращаемых значений, атрибуты классов. Научиться определять связи между классами.**
Исполнение:*1.*Добавление нового класса. 2. Добавление атрибутов. 3. Добавление операций к классу OrderItem.4. Подробное описание операций с помощью диаграммы Классов
Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose
Оценка:. Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.
Время выполнения работы: 2 часа.

5. **Представление компонентов. Представление размещения**
Цель работы: **Научиться с**оздавать диаграмму Компонентов системы обработки заказов и осуществлять работу с ней. Научиться создавать диаграмму размещения и осуществлять работу с ней.

Исполнение:1. Создание пакетов компонентов. 2.Добавление пакетов на Главную диаграмму Компонентов. 3.Отображение зависимостей между пакетами. 4. Добавление компонентов к пакетам и отображение зависимостей. 5 Создание диаграммы Компонентов системы. 6. Размещение компонентов на диаграмме Компонентов системы. 7.Добавление оставшихся зависимостей на диаграмму Компонентов системы. 9. Создать диаграмму размещения. 10. Добавление узлов к диаграмме Размещения. 11. Добавление связей. 12.Добавление процессов.13. Показ процессов на диаграмме
Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose
Оценка:. Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.
Время выполнения работы: 3 часа.

6. **Генерация программного кода C++.**

Цель работы: Научиться производить автоматическую генерацию кода на языке C++, основываясь на диаграмме размещений.
Исполнение:1. Ввод тел пакетов на диаграмму Компонентов системы. 2. Установка языка C++ 3. Генерация программного кода C++
Обеспечение: Персональный компьютер; Операционная система Windows. Среда программирования Visual Studio2005. Rational Rose
Оценка:. Проверяется правильность выполнения заданий. Полнота отчета. Обучающийся должен продемонстрировать уверенной пользование системой Rational Rose при решении аналогичных задач.
Время выполнения работы: 3 часа.

Практические занятия и их взаимосвязь с содержанием лекционного курса

№ п/п	№ раздела по варианту содержания	Наименование практической работы
1	2,6,8,910,11,16	Варианты использования и действующие лица
2	2,6,8,910,11,16	Взаимодействие объектов. Поведение объектов
3	2,6,8,910,11,16	Классы и пакеты
4	2,6,8,910,11,16	Атрибуты и операции. Связи. Поведение объектов
5	2,6,8,910,11,16	Представление компонентов. Представление размещения
6	2,6,8,910,11,16	Генерация программного кода C++

6. Реферат

Студентам дневной формы обучения во время самостоятельной работы может быть предложена подготовка реферата по современным актуальным направлениям в области разработки ПО. Подготовка реферата преследует цель ознакомления студентов с последними достижениями в рассматриваемой сфере, т.к. развитие информационных технологий происходит чрезвычайно быстро и последние разработки могут быть не включены в курс обучения.

Объем в страницах – до 20 стр. Время на разработку, включая поиск информации - -8-10 часов.

Примерные темы рефератов

1. Создание оптимальных и достаточных групп разработчиков ПО.
2. Области целевого и обоснованного применения паттернов.
3. Системы распределенных, параллельных вычислений, оценка производительности.
4. Достоинства и недостатки экстремального программирования.
5. Методы защиты программ.
6. Сравнение платформ Framework и Java

И др.

7. Контроль знаний студентов

1. Тематика вопросов входного контроля.

Студент должен знать:

- Теоретические основы информатики и вычислительной техники (системы счисления, арифметические и логические операции, представление информации в ЭВМ, кодирование информации);

- Алгоритмизацию и программирование (основы алгоритмизации, язык программирования С, С++, понятие функционального программирования);
- Основные положениями объектно-ориентированного анализа, проектирования и программирования. Понятия класс, объект, диаграммы взаимодействия;
- Операционные системы (назначение, виды, основные функции, общую структуру ОС)

2. Текущий контроль знаний студентов.

Текущий контроль осуществляется на лабораторных и практических занятиях путем решения задач, ответов на контрольные вопросы, защите лабораторных и практических работ. Тематика лабораторных и практических работ приведена выше.

3. Выходной контроль знаний студентов.

Дисциплина завершается зачетом и экзаменом. На экзамене проверяется степень усвоения студентами основных понятий дисциплины, понимание их взаимосвязи, знание основ современных технологий программирования, умение применять полученные навыки при проектировании и анализе ПО.

Примерный состав вопросов в билетах экзамена по дисциплине

1 Понятие программного средства.
2 Понятие ошибки в программном средстве. Модель перевода и источники ошибок при разработке программных средств.
3 Специфические особенности разработки программных средств.
4 Жизненный цикл программного средства.
5 Определение требований к программному средству.
6 Функциональная спецификация программного средства.
7 Табличный подход к спецификации семантики функций. Метод таблиц решений.
8 Основные классы архитектур программных средств.
9 Понятие программного модуля и его основные характеристики.
10 Методы разработки структуры программ.
11 Метод целенаправленной конструктивной реализации.
12 Отладка программного средств.
13 Автономная отладка и тестирование программного средства.
14 Комплексная отладка и тестирование программного средства.
15 Обеспечение устойчивости программного модуля.
16 Обеспечение защиты от влияния «чужих» программ.
17 Обеспечение защиты от несанкционированного доступа к программным средствам и защиты от взлома защиты.
18 Обеспечение легкости применения программного средства.
19 Обеспечение эффективности программного средства.
20 Обеспечение сопровождаемости программного средства.
21 Виды документов программного средства.
22 Структура управления разработкой программного средства.
23 Особенности внешнего описания программных средств при объектном подходе к разработке.
24 Инструментальные среды разработки и сопровождения программных средств. Принципы их классификации.
25 Инструментальные системы технологии программирования и их общая архитектура.
26 Технологии распределенного программирования
27 MPI.
28 CORBA

29 Технология DotNet
30 Интегрированности инструментальной среды разработки и сопровождения программных средств.
31 Языково-ориентированной инструментальной среды программирования
32 Компьютерной технологии программирования.
33 Понятие языково-зависимой инструментальной системы технологии программирования

8. Контроль самостоятельной работы студентов-заочников

Каждый студент должен выполнить 2 контрольных и 2 лабораторных работ.
Выполнение контрольных работы является важным звеном в обучении студентов-заочников и преследует следующие цели:
-оказать помощь студенту в овладении необходимыми навыками правильной организации самостоятельной работы в межсессионный период;
-привить навыки самостоятельного изучения материала по учебной дисциплине;
-указать правильную последовательность в изучении данной учебной дисциплины;
-закрепить знания основных положений учебной дисциплины;
-систематизировать знания по учебной дисциплине;
-выработать умение анализировать достоинства и недостатки отдельных технических решений;
-привить навыки применения теоретических знаний для решения практических вопросов;
-научить студента грамотно, лаконично излагать материал;
-проверить работу студента-заочника в межсессионный период по изучению данной дисциплины.
В контрольной работе студент должен продемонстрировать понимание предложенных в контрольной работе вопросов, показать знание теории технологии программирования.

Примерный состав вопросов для контрольной работы

1. Ошибки в программном средстве.
2. Определение и характеристики программного средства.
3. Надежность программного средства.
4. Технологии программирования. Определение. Классификация.
5. Сопровождение программного средства.
6. Качество программного средства.
7. Мобильность программного средства.
8. Состав внешнего описания программного средства.
9. Устойчивость программного средства.
10. Защищенность программного средства.
11. Коммуникабельность программного средства.
12. Расширяемость программного средства.
13. Модульность программного средства.
14. Архитектура программного средства.
15. Архитектурная функции программного средства.
16. Программный модуля.
17. Отладка программного средства.
18. Тестирование программного средства.
19. Управление конфигурацией программного средства
20. Управление разработкой программного средства.
21. Аттестация программного средства.
22. Пользовательский объекта в программном средстве.

23. Инструментальный среды разработки и сопровождения программных средств.
24. Интегрированность инструментальной среды разработки и сопровождения программных средств.
25. Языково-ориентированная инструментальная среда программирования
26. Компьютерная технология программирования.
27. Инструментальная система технологии программирования.
28. Языково-зависимая инструментальная система технологии программирования

Студенту предоставляется к ответу 5 вопросов для выполнения каждой контрольной работы. Приступая к выполнению контрольной работы, студент должен выписать из общего списка вопросы, которые включены в его вариант контрольной работы, уяснить какого ответа требуют предлагаемые вопросы. Затем в первом приближении изучить учебную дисциплину по рекомендованной литературе, руководствуясь учебной программой или списком вопросов для подготовки к экзамену с тем, чтобы иметь общее представление по всем вопросам учебной дисциплины и чувствовать взаимосвязь предложенных в контрольной работе вопросов с другими вопросами дисциплины. После этого можно приступить к более глубокому изучению материала по тем вопросам, которые заданы в контрольной работе и подготовке ответа на них.
Отрабатывать вопросы контрольной работы следует по нескольким рекомендованным пособиям, делая в тетради отдельные выписки и приводя необходимые рисунки (схемы). При отработке вопросов контрольной работы можно привлекать и другие источники, не приведённые в списке рекомендованной литературы. После сбора необходимого материала для ответа на вопросы контрольной работы, разработки необходимых схем, следует написать черновой вариант контрольной работы, используя сделанные ранее выписки. После этого следует отредактировать текст контрольной работы и оформить работу начисто. Писать текст контрольной работы следует собственным языком. Не допускается компиляция и плагиат текста из используемой литературы.

Лабораторная работа выбирается из предложенного списка в разделе «Лабораторные работы».

9. Учебно-методическое обеспечение дисциплины

Основная литература

1. Иванова Г. С.Технология программирования: Учебник для вузов. – М.: Издательство МГТУ им. Н. Э. Баумана, 2002 – 320с.
2. Технология разработки программного обеспечения: учебник. Орлов. С. СПб.: BHV, 2002 г.
3. Брауде Э. Технология разработки программного обеспечения. – СПБ.: Питер, 2004. – 655 с.
4. Фаулер М. Рефакторинг: Улучшение существующего кода. – Пер с англ. – СПб: Символ плюс,2003. – 432 с.
5. UML. Классика CS. 2-е изд. Буч Г. СПБ.: Питер: 2006 г.
6. Е. А. Жоголев. Лекции по технологии программирования: Учебное пособие. М., Издательский отдел факультета ВМиК МГУ, 2001.

Дополнительная литература

1. Основы многопоточного, параллельного и распределенного программирования. Эндрюс Г.Р. М.: Вильямс, 2003 г.
2. Параллельные вычисления Воеводин В.В., Воеводин В. Изд-во: BHV-СПб. : 2002 г.
3. Основы С++: Методы программной инженерии. Штерн В. Изд-во: Лори: 2003 г.
4. Гамма Э. Приемы объектно - ориентированного проектирования. Паттерны проектирования. – СПБ.: Питер, 2001. – 368 с.

5. Фаулер М. UML. Основы. . – Пер с англ. – СПб: Символ плюс,2002. – 192 с.
6. Р. Андерсон и др. ASP.NET для профессионалов. Тома I, II. Лори, 2005.

Методические указания

Методические указания к лабораторным и практическим работам доступны в виде электронного ресурса на сайте http://evm.khstu.ru в разделах:
- Дисциплины/ Технология программирования/Лабораторные работы.
- Дисциплины/Технология программирования/Методические указания.

Методические указания представлены в рукописном виде.

10. Материально-техническое обеспечение дисциплины.

Персональный компьютер, подключенный к сети. Операционная система Windows,Linux. Инструментальные среды разработки Visual Studio 2005, Rational Rose. Текстовый редактор.

11. Методические рекомендации по организации изучения дисциплины

Курс рассматривает теоретические основы современных технологий программирования, используемых при создании программных средств.

Рассмотрение ведется на базе теории технологии программирования, детально отраженной в основных литературных источниках 1,2,3,6. Все разделы лекционного курса представленных студентам направлены на введение обучающегося в проблематику программирования и изучение критериев подбора инструментальных средств, изучение основных технических приемов программирования и отладки программ, формирование у обучающихся твердых представлений о роли структур данных в построении и реализации информационных моделей, а так же, знакомство с основными положениями объектно-ориентированного анализа, проектирования и программирования для различных технологий программирования. По мере изучения материала студентам предоставляется наглядные примеры применения конкретной технологии программирования при решении определенных задач проектирования программных средств.

На лабораторных и практических работах значительное внимание уделяется применению всестороннего унифицированного процесса разработки программного средства с использованием языка проектирования и описания UML при использовании различных средств предоставляемых конкретной технологией создания ПО, а так же на всесторонний анализ ПО созданного во время выполнения лабораторных работ или полученного от стороннего производителя.

Организация самостоятельной работы

Самостоятельная работа предполагает, что:

1) отдельные темы могут быть отнесены на самостоятельное изучение;
2) теоретическая подготовка к лабораторным работам с использованием МУ может осуществляться дома самостоятельно.

Словарь.

АЛГОРИТМ -
Последовательность действий, которая приводит к решению поставленной задачи. Алгоритмы для ЭВМ удобнее составлять в виде блок-схемы.

АЛГОРИТМИЧЕСКАЯ МОДЕЛЬ -
С ее помощью можно сравнить последствия всех вариантов возможных действий, не выполняя этих действий в реальности. Это дает возможность избежать многих ошибок. Алгоритмическая модель отвечает на вопрос: как мы будем делать?

АВТОР ПРОГРАММЫ ИЛИ БАЗЫ ДАННЫХ -
в РФ - физическое лицо, в результате творческой деятельности которого созданы программа или база данных.

АДАПТАЦИЯ ПРОГРАММЫ ИЛИ БАЗЫ ДАННЫХ -
по законодательству РФ - внесение изменений, осуществляемых исключительно в целях обеспечения функционирования программы для ЭВМ или базы данных на конкретных технических средствах пользователя или под управлением конкретных программ пользователя.

АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР
МОДЕЛЬ КЛИЕНТ-СЕРВЕР -
архитектура распределенной вычислительной системы, в которой приложение делится на клиентский и серверный процессы.
В зависимости от того, как распределены логические компоненты приложения между клиентами и серверами, различают четыре модели архитектуры клиент-сервер:
- модель "файл-сервер";
- модель "сервер базы данных";
- модель "сервер транзакций";
- модель "сервер приложений".

АСПЕКТНО-ОРИЕНТИРОВАННОЕ СБОРОЧНОЕ ПРОГРАММИРОВАНИЕ -
разновидность сборочного программирования, основанная на сборке полнофункциональных приложений из многоаспектных компонентов, инкапсулирующих различные варианты реализации.

БЕСПЛАТНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ(FREEWARE) -
свободно распространяемое программное обеспечение: которое пользователь поддерживает самостоятельно; и в которые пользователь правомочен вносить изменения.

ВИРТУАЛЬНАЯ МАШИНА -
совокупность вычислительных ресурсов, реализующая поведение некоторого реального компьютера. На одном реальном компьютере может быть построено несколько виртуальных машин, каждая из которых выполняет свою программу и не препятствует работе другой виртуальной машины

ВОДОПАДНАЯ МОДЕЛЬ ПРОГРАММИРОВАНИЯ(WATERFALL) -
линейная модель жизненного цикла проекта разработки программ, состоящая из четко определенных фаз: сбор требований, проектирование, кодирование, тестирование и эксплуатация. Каждый их этих этапов должен быть завершен до начала следующего. Допускаются возвраты к предыдущему этапу

ДАННЫЕ -
Информация, представленная в такой форме, которую можно распознать автоматическому устройству или человеку.

ДВУХУРОВНЕВАЯ АРХИТЕКТУРА, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА -
архитектура приложения, в которой прикладные и пользовательские сервисы реализованы на клиентской рабочей станции, а данные централизованно хранятся на сервере. В этой модели клиенты подключаются непосредственно к серверу, на все время работы приложения

ДЕКОМПОЗИЦИЯ ЦЕЛЕЙ СИСТЕМЫ -
Представление исходной цели совокупностью упрощенных (локальных) целей, при достижении которых должна быть достигнута исходная цель

ДИАГРАММА ДЕЯТЕЛЬНОСТИ -
методология объектно-ориентированного проектирования, предназначенная для детализации особенностей алгоритмической и логической организации системы. При этом каждое действие расчленяется на фундаментальные процессы. На диаграмме деятельности управление осуществляется:
- либо через потоки управления (явно);
- либо через определяемые потоки данных (неявно).

ДИАГРАММА КЛАССОВ -
методология объектно-ориентированного проектирования, предназначенная для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

ДИАГРАММА КОМПОНЕНТОВ -
метод объектно-ориентированного проектирования, описывающий особенности физического представления системы.
Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, устанавливая зависимости между компонентами
ДИАГРАММА КООПЕРАЦИИ -
метод объектно-ориентированного проектирования, основанный на графическом представлении всех структурных отношений между объектами, участвующими во взаимодействии.
Диаграмма кооперации представляет собой граф, в вершинах которого располагаются объекты, соединенные дугами-связями. При этом дуги могут быть аннотированы сообщениями, которыми обмениваются объекты.
ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ -
методология объектно-ориентированного проектирования, предназначенная для моделирования взаимодействия во времени. Диаграмма последовательности позволяет отслеживать поведение взаимодействующих групп объектов.
ДИАГРАММА РАЗВЕРТЫВАНИЯ, ДИАГРАММА ПРИМЕНЕНИЯ; ДИАГРАММА РАЗМЕЩЕНИЯ -
метод объектно-ориентированного проектирования, отображающий физические взаимосвязи между программными и аппаратными компонентами системы.
ДИАГРАММА СОСТОЯНИЙ -
методология объектно-ориентированного проектирования, предназначенная для представления жизненного цикла объектов в реальном или абстрактном мире. Диаграмма состояний состоит
- из множества состояний объектов;
- из множества событий, сообщающих о перемещении чего-либо в новое состояние;
- из множества правил переходов, определяющих новое состояние объекта при возникновении тех или иных событий;
- из множества действий, которые должны быть выполнены объектом, когда он переходит в новое состояние.
ДИАГРАММА ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ -
инструмент разработки функциональных спецификаций в виде диаграмм, фрагментов текста и глоссария, связанных перекрестными ссылками. В состав диаграммы входят: блоки, изображающие активность моделируемой системы; и дуги, связывающие блоки вместе и изображающие взаимодействия и взаимосвязи между ними. Место соединения дуги с блоком определяет тип интерфейса: - управляющая информация входит в блок сверху;
- входная информация, подвергающаяся обработке, показывается с левой стороны блока;
- выходная информация показывается с правой стороны;
- механизм, осуществляющий операцию, представляется дугой, входящей в блок снизу.
ГИПЕРССЫЛКА -
Это выделенный фрагмент документа (текст или иллюстрация), с которым ассоциирован адрес дру¬гого Web-документа. Гиперссылки обеспечивают навигацию в среде WWW. Такой фрагмент документа называют ещё указателем ссылки.
При использовании гиперссылки происходит переход по гиперссылке — открытие Web-страницы, на которую указывает ссылка. Механизм гипер¬ссылок позволяет организовать тематическое путешествие по World Wide Web без использования (и даже без знания) адресов конкретных страниц. Документы, на которые сделаны ссылки, могут находиться на удаленных компьютерах.
ГИПЕРТЕКСТ -
Это форма организации информации, разделенной на фрагменты, для каждого из которых указаны переходы к другим фрагментам с указанием типа их взаимосвязи.
ГРАФ -
Это структура, которая состоит из вершин, связанных дугами или рёбрами. Вершины могут быть изображены кругами, овалами, точками, прямоугольниками и пр. Связи между вершинами изображаются линиями. Если линия направленная (т.е. со стрелкой), то она называется дугой, если не направленная (без стрелки), то ребром. Принято считать, что одно ребро заменяет две дуги, направленные в противоположные стороны.
ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ -
период разработки и эксплуатации программного обеспечения, в котором обычно выделяют этапы:
-1- возникновение и исследование идеи;
-2- анализ требований и проектирование;
-3- программирование;
-4- тестирование и отладка;
-5- ввод программы в действие;
-6- эксплуатация и сопровождение;
-7- завершение эксплуатации.

ИНКАПСУЛЯЦИЯ -
в объектно-ориентированном программировании - сокрытие внутренней структуры данных и реализации методов объекта от остальной программы. Другим объектам доступен только интерфейс объекта, через который осуществляется все взаимодействие с ним.
ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА (SOFTWARE TOOLS) -
программное обеспечение, используемое в ходе разработки, корректировки или развития других программ: редакторы, компиляторы, отладчики, вспомогательные системные программы, графические пакеты и др.
ИНТЕРПРЕТАТОР -
Специальная программа, которая последовательно преобразует по смыслу каждый отдельный оператор программы и исполняет его
ИНТЕРФЕЙС -
Система подключения к портам ввода/вывода внешних устройств — клавиатуры, дисплея, графопостроителя, магнитофона, дисковода
Система взаимодействии между приложением пользователя (пользователем) и ОС через операционную среду или с помощью графической системы
ИСПОЛНЕНИЕ АЛГОРИТМА -
процесс пошагового, точного выполнения алгоритма исполнителем
КАНАЛ -
Совокупность технических и/или программных средств для передачи данных от одного устройства к другому.
КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ -
способность программного продукта подтвердить свою спецификацию при условии, что спецификация ориентирована на характеристики, которые желает получить пользователь
КЛАСС -
Это шаблон, определяющий набор свойств, методов и событий, по которому создаются объекты
КЛАСС ОБЪЕКТА -
Группа объектов, обладающих одинаковыми общими свойствами
КЛИЕНТСКИЙ ПРОЦЕСС, КЛИЕНТ-БАЗИРОВАННЫЙ ПРОЦЕСС -
в архитектуре клиент-сервер - процесс, выполняемый на стороне клиента и посылающий запрос серверному процессу на выполнение некоторой задачи. Обычно клиентский процесс:
- управляет пользовательским интерфейсом;
- контролирует вводимые пользователем данные;
- распределяет запросы серверным процессам;
+ может выполнять бизнес-логику приложений.
КОД -
Данные, представленные в определенном виде.
КОДИРОВАНИЕ -
Кодирование в узком смысле – это перевод одной формы представления информации в другую.
С этой точки зрения перевод текста с одного языка на другой – тоже кодирование, хотя в бытовом смысле мы этот процесс кодированием не считаем.
Кодирование в широком смысле – это присвоения имен объектам и явлениям с использованием определенного алфавита.

КОММЕНТАРИЙ -
Фрагмент программы, предназначенный для восприятия ее человеком. При вводе в ЭВМ игнорируется.
КОМПИЛЯТОР -
Программа, переводящая программу, написанную на каком-либо языке программирования, на другой язык.
ЛИСТИНГ -
Вывод данных на печатающие устройства; распечатанные данные
МАСШТАБИРУЕМОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (SCALABILITY) -
способность программного обеспечения корректно работать на малых и на больших системах с производительностью, которая увеличивается пропорционально вычислительной мощности системы.
МНОГОУРОВНЕВАЯ АРХИТЕКТУРА, ТРЕХУРОВНЕВАЯ АРХИТЕКТУРА -
архитектура приложения, разделяющая пользовательские сервисы, прикладные сервисы и сервисы данных
МОДЕЛЬ "СЕРВЕР БАЗЫ ДАННЫХ - архитектура вычислительной сети типа "клиент-сервер", в которой пользовательский интерфейс и логика приложений сосредоточены на машине-клиенте, а информационные функции (функции СУБД) - на сервере. Обычно клиентский процесс посылает запрос серверу на языке SQL.
МОДЕЛЬ "СЕРВЕР ПРИЛОЖЕНИЙ" -
архитектура вычислительной сети типа "клиент-сервер", в которой функциональная логика размещена на сервере, а на машине-клиенте выполняется только компонент представления.

МОДЕЛЬ "СЕРВЕР ТРАНЗАКЦИЙ" -
архитектура вычислительной сети типа "клиент-сервер", в которой сервер выполняет специальные, реализующие наиболее часто используемые алгоритмы обработки (модули; удаленные процедуры). Алгоритмы получают параметры от клиентского процесса и ему же возвращают результат
МОДЕЛЬ "ФАЙЛ-СЕРВЕР" -
архитектура вычислительной сети типа "клиент-сервер", в которой сервер предоставляет в коллективное пользование дисковое пространство, систему обслуживания файлов и периферийные устройства
МЕТОД ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ ДЕКОМПОЗИЦИИ -
основной метод объектно-ориентированного программирования, описывающий:
- статическую структуру системы в терминах объектов и связей между ними;
- поведение системы в терминах обмена сообщениями между объектами
МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ -
метод разработки программ, предполагающий разбиение программы на независимые модули. Считается, что:
- оптимальный по размерам модуль целиком помещается на экране дисплея;
- разделение большой программы на модули облегчает ее разработку, отладку и сопровождение.
МОНИТОР -
Специальная программа, входящая в состав системного программного обеспечения микроЭВМ, осуществляющая взаимодействие микропроцессора с клавиатурой, дисплеем, ОЗУ. Она управляет вводом и исполнением других программ. В директивы МОНИТОРА входят также функции отладки, запуска и исполнения программ пользователя.
ОБЪЕКТ -
в программировании - программный модуль:
- объединяющий в себе данные (свойства) и операции над ними (методы);
- обладающий свойствами наследования, инкапсуляции и полиморфизма.
Объекты взаимодействуют между собой, посылая друг другу сообщения.
ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА -
архитектура, основой которой является множество взаимодействующих друг с другом объектов.
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ -
технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ СБОРОЧНОЕ ПРОГРАММИРОВАНИЕ -
разновидность сборочного программирования:
- основанная на методологии объектно-ориентированного программирования; и
- предполагающая распространение библиотек классов в виде исходного кода или упаковку классов в динамически компонуемую библиотеку.
ОБРАБОТКА ДАННЫХ -
процесс выполнения последовательности операций над данными. Обработка данных может осуществляться в интерактивном и фоновом режимах
ОТЛАДКА ПРОГРАММЫ -
этап разработки компьютерной программы, в процессе которого происходят обнаружение, локализация и устранение явных ошибок в программе.
Обычно отладка выполняется на контрольных примерах с известными результатами.
ПЛАНИРОВЩИК -
программа, определяющая порядок использования прикладными процессами совместных ресурсов. Планировщик учитывает:
- приоритеты прикладных процессов;
- требования к эффективности использования ресурсов системы;
- заданные сроки выполнения заданий и т.п.
ПОСТАНОВКА ЗАДАЧИ - В ПРОГРАММИРОВАНИИ -
точная формулировка решения задачи на компьютере с описанием входной и выходной информации.
ПРОГРАММНЫЙ МОДУЛЬ -
согласно ГОСТ 19781-90 - программа или функционально завершенный фрагмент программы, предназначенный для:
-1- хранения;
-2- трансляции;
-3- объединения с другими программными модулями; и
-4- загрузки в оперативную память.
Различают:
- стандартные модули, входящие в язык программирования; и
- пользовательские модули, предназначенные для упрощения работы программистов.

ПРОГРАММИРОВАНИЕ -
процесс подготовки задач для их решения с помощью компьютера; итерационный процесс составления программ
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ (ПО) (SOFTWARE) -
комплекс программ:
- обеспечивающих обработку или передачу данных;
- предназначенных для многократного использования и применения разными пользователями.
По видам выполняемых функций программное обеспечение подразделяется на системное, прикладное и инструментальное.
Программное обеспечение - согласно ГОСТ 19781-90 - совокупность программ системы обработки информации и программных документов, необходимых для их эксплуатации.
ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ -
этап жизненного цикла программного обеспечения, во время которого исследуется структура и взаимосвязи элементов разрабатываемой системы. Результатом этого этапа является проект, содержащий достаточное количество информации для реализации системы. Различают проектирование архитектуры системы и детальное проектирование программных модулей
ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ -
программное обеспечение, состоящее из:
- отдельных прикладных программ и пакетов прикладных программ, предназначенных для решения различных задач пользователей; и
- автоматизированных систем, созданных на основе этих (пакетов) прикладных программ.
ПРОМЕЖУТОЧНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ(СРЕДНИЙ СЛОЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ - MIDDLEWARE) -
совокупность программ, осуществляющих управление вторичными ресурсами:
- конструируемыми самим программным обеспечением; и
- ориентированными на решение определенного класса задач.
К промежуточному программному обеспечению относятся: менеджеры транзакций, серверы баз данных, серверы коммуникаций и другие программные серверы.
ПОДПРОГРАММА -
Вспомогательная программа, облегчающая и ускоряющая процесс обработки данных или работу другой, более сложной программы.
ПСЕВДОКОД -
система обозначений и правил, предназначенная для единообразной записи алгоритмов. Псевдокод занимает промежуточное место между естественным и формальным языками
РАСПРЕДЕЛЕННАЯ СРЕДА ОБРАБОТКИ ДАННЫХ, СРЕДА РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ (DISTRIBUTED COMPUTING ENVIRONMENT (DCE)) -
технология распределенной обработки данных, представляющая собой стандартный набор сетевых служб для выполнения прикладных процессов, рассредоточенных по группе абонентских систем (по гетерогенной сети). Функции распределенной среды включают:
- службу каталогов, позволяющую клиентам находить серверы;
- службу интерфейса многопоточной обработки;
- службу удаленного вызова процедур;
- службу обслуживания файлов;
- службу безопасности данных;
- службу времени, синхронизирующей часы в абонентских системах.
РАСПРЕДЕЛЕННОЕ ПРИЛОЖЕНИЕ -
приложение, которое выполняется в среде распределенных вычислений. Модули такого приложения могут выполняться на разных вычислительных системах.
РЕДАКТОР -
Специальная программа для составления и корректировки программ пользователя с учетом языка программирования и архитектуры ЭВМ
СВОЙСТВО ОБЪЕКТА -
в объектно-ориентированном программировании - характеристика объекта. Обычно свойства изменяются с помощью методов
СЕГМЕНТАЦИЯ ПРИКЛАДНОЙ ПРОГРАММЫ -
разделение прикладной программы на части, которые могут быть распределены по системам локальной сети. Сегментация осуществляется с помощью специального инструментального программного обеспечения
СЕРВЕР ПРИЛОЖЕНИЙ -
сервер, предназначенный для выполнения прикладных процессов. Сервер приложений:
- взаимодействует с клиентами, получая задания; и
- взаимодействует с базами данных, выбирая данные, необходимые для обработки.

СОВМЕСТИМОСТЬ -
Свойство одной или нескольких моделей ЭВМ одинаково исполнять программы
СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ИЗДЕЛИЯ -
процесс модификации существующей программы для ЭВМ, обусловленный необходимостью устранения выявленных в ней ошибок и/или изменения ее функциональных возможностей.
СИСТЕМНАЯ ПРИКЛАДНАЯ АРХИТЕКТУРА -
предложенная корпорацией IBM модель распределенной обработки данных в сети, обеспечивающая:
- согласование характеристик прикладных процессов, работающих в среде различных операционных систем;
- создание распределенных систем управления базами данных;
- разнообразные формы доступа пользователей к распределенным по абонентским системам ресурсам.
СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ -
совокупность программ и программных комплексов для обеспечения работы компьютера и вычислительных сетей.
Системное программное обеспечение ориентировано:
- на создание операционной среды функционирования других программ;
- на обеспечение надежной и эффективной работы самого компьютера и вычислительной сети;
- на диагностику и профилактику аппаратуры компьютера и вычислительных сетей;
- на выполнение вспомогательных технологических процессов: копирование, архивация, восстановление файлов и т.п. вспомогательных программ, осуществляющих:
- защиту, архивацию, восстановление данных;
- всевозможные тесты и т.д.
СПЕЦИФИКАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ -
описание системы, которое полностью определяет ее цель и функциональные возможности. Различают:
- словесные спецификации на естественном языке;
- модельные спецификации;
- формальные спецификации.
СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ -
методология и технология разработки программных комплексов, основанная на принципах:
- программирования "сверху-вниз";
- модульного программирования.
При этом логика алгоритма и программы должны использовать три основные структуры: последовательное выполнение, ветвление и повторение.
ТРАНСЛЯТОР -
Программа, переводящая исходную программу и объектный (машинный) код.
ТЕСТ -
Программа для проверки работоспособности ЭВМ в целом и ее отдельных узлов.
ТЕСТИРОВАНИЕ ПРОГРАММ -
этап разработки компьютерной программы, в процессе которого проверяется работоспособность программы, не содержащей явных ошибок.
Тестирование - процесс выполнения программ с целью обнаружения факта наличия ошибок

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ -
дисциплина, изучающая технологические процессы программирования и порядок их прохождения.
ТРАНСЛЯТОР -
Специальная программа преобразования программ на некотором языке в программу на другом языке с сохранением ее функций.
УПРАВЛЕНИЕ ПРОЕКТОМ -
деятельность, направленная на реализацию проекта с максимально возможной эффективностью при заданных ограничениях по времени, денежным средствам и ресурсам, а также качеству конечных результатов проекта.
ЭРГОНОМИКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ -
подраздел микроэргономики, ориентированный на системы "человек-компьютер", "человек-компьютер-человек", "человек-компьютер-процесс", "человек-программа" и т.п.
ЯЗЫК ПРОГРАММИРОВАНИЯ -
искусственный (формальный) язык, предназначенный для записи алгоритмов. Язык программирования задается своим описанием и реализуется в виде специальной программы: компилятора или интерпретатора.

ЯЗЫК ПРОГРАММИРОВАНИЯ COBOL -
язык программирования, предназначенный для решения экономических задач и задач обработки коммерческой информации. Язык COBOL отличается развитыми средствами работы с файлами и формой записи, приближенной к английскому языку.

ЯЗЫК ПРОГРАММИРОВАНИЯ АДА -
универсальный язык программирования:
- предназначенный для создания сложных систем;
- характеризующийся высокой степенью независимости от операционных систем;
- обеспечивающий поддержку средств параллельной обработки данных в реальном времени.

ЯЗЫК ПРОГРАММИРОВАНИЯ АЛГОЛ -
язык программирования, предназначенный для решения численных задач. Программа на языке ALGOL имеет блочную структуру, позволяющую эффективно реализовывать механизмы динамического распределения памяти

ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ -
процедурно-ориентированный язык программирования высокого уровня, предназначенный для широкого класса задач. Язык Паскаль считается языком структурного программирования.

ЯЗЫК ПРОГРАММИРОВАНИЯ ФОРТ -
процедурно-ориентированный язык программирования, предназначенный для эффективной работы с персональными системами. Форт близок к языку ассемблера.