Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Амурский государственный университет»

Кафедра математического анализа и моделирования

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

Системное и прикладное программное обеспечение

Основной образовательной программы направления 010500.62 – прикладная математика и информатика

Благовещенск 2012 г.

УМКД разработан доцентом Труфановым Виктором Александровичем

Рассмотрен и рекомендован на заседании кафедры
Протокол заседания кафедры от «___» _____ 201_ г. №___
Зав. кафедрой _____ / Н.Н.Максимова /

УТВЕРЖДЕН
Протокол заседания УМСС 010501 – Прикладная математика и информатика
от «___» _____ 201_ г. №___
Председатель УМСС _____ / В.В.Сельвинский /

СОДЕРЖАНИЕ

I РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

1. Цели и задачи изучения дисциплины

1.1. Цель преподавания дисциплины

Основной целью дисциплины является получение знаний по основным принципам построения, функционирования и использования современных средств вычислительной техники (ВТ), овладение основными приемами и методами программного управления средствами ВТ; обучение студентов теоретическим основам и практическим навыкам работы с прикладным программным обеспечением, ориентированным на решение различного рода практических задач

1.2. Задачи изучения дисциплины:

знать и уметь использовать:

- базовые устройства современных персональных компьютеров;
- решать задачи, возникающие в процессе сопровождения и эксплуатации программных средств;
- прикладное программное обеспечение для решения различных практических задач пользователя;

2. Место дисциплины в учебном процессе

ОПД.Ф.06 Системное и прикладное программное обеспечение: 102

основные этапы, методы, средства и стандарты разработки программного обеспечения; системы программирования (принципы организации, состав и схема работы); основные типы операционных систем, принципы управления ресурсами в операционной системе; сети ЭВМ и протоколы передачи информации.

Курс входит в цикл ОПДФ (общепрофессиональные дисциплины направления).

Дисциплина «Системное и прикладное программное обеспечение» базируется на знаниях, полученных в рамках курса «Информатика», «Языки программирования и методы трансляции», «Практикум на ЭВМ».

Курс дисциплины «Системное и прикладное программное обеспечение» используется в дисциплинах «Операционные системы и сетевые технологии», «Программное обеспечение вычислительных сетей и систем».

3. Требования к уровню освоения содержания дисциплины «Системное и прикладное программное обеспечение»

В результате изучения дисциплины студент должен приобрести навыки:

- разработки, отладки, тестирования и документирования прикладного программного обеспечения;
- использования программного обеспечения ПК;
- об основных типах ОС;
- о сетях ЭВМ.

4. Структура и содержание дисциплины «Системное и прикладное программное обеспечение»

Общая трудоемкость дисциплины составляет 102 часа.

№ п/п	Раздел дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)				Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
				Лек.	Прак. зан.	Лаб. зан.	Сам. раб.	
1	Система MAPLE.	4	1-3	6		16	10	Индивидуальные задания.
2	Программное обеспечение(ПО) компьютеров.	4	4-9	12		14	10	Индивидуальные задания.
3	Операционные системы (ОС) и сети.	4	10-13	8		6	5	Индивидуальные задания.
4	Технология программирования.	4	14-18	10			5	

5. Содержание разделов и тем дисциплины

Наименование темы	Лекции	Лабораторные работы
1. ***Система MAPLE*** 1.1 Основы работы в системе Maple. Графический интерфейс пользователя. Пакеты расширений. Алфавит Maple-языка и его синтаксис. Определение функций пользователя. 1.2. Основные объекты и команды Maple. Определение, ввод, действия с объектами. Внутренняя структура объектов Maple. Подстановка и преобразование типов. Встроенные элементарные математические функции. Операции и функции математического анализа.	3	8
2. ***ПО компьютеров.*** 2.1. ПО. Классификация ПО. Прикладные и системные программы. 2.2. Программы-оболочки. Транслятор, компилятор, интерпретатор. Системы программирования. 2.3. Инструментальные программы. Текстовый редактор. Графический редактор. Возможности систем деловой и научной графики. Табличный процессор. Системы управления базами данных. 2.4. Пакеты прикладных программ. 2.5. Сетевое ПО.	6	7
3. ***Операционные системы и сети.*** 3.1. Эволюция ОС.		

3.2. Архитектура ОС. 3.3. Координация действий компьютера. 3.4. Сети. 3.5. Сетевые протоколы 3.6. Безопасность.	4	3
4. Технология программирования. 4.1. Предмет технологии разработки ПО. 4.2. Жизненный цикл ПО в целом. 4.3. Традиционный этап разработки программ. 4.3.1. Реализация модулей. 4.3.2. Связанность модулей. 4.3.3. Связность элементов модуля.	5	

6. Самостоятельная работа

№ п/п	№ раздела (темы) дисциплины	Форма (вид) самостоятельной работы	Трудоём-кость в ч.
1	1	Индивидуальное задание.	10
2	2	Индивидуальное задание.	10
3	3	Индивидуальное задание.	5
4	4	Домашнее задание	5

8. Образовательные технологии

Учебные занятия: лекции, в которых используется традиционное и проблемное изложение теоретического материала с текущим устным опросом; лабораторные занятия, с использованием интерактивных форм проведения занятий (компьютерных симуляций, разбор конкретных ситуаций, психологические и иные тренинги), с компьютерным тестированием; консультации; самостоятельная работа.

9.Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов

Экзаменационные вопросы.

1. Программное обеспечение (ПО).
2. Классификация ПО.
3. Прикладные программы.
4. Роль и назначение системных программ.
5. Операционная система (ОС).
6. Файловая система ОС.
7. Структура ОС MS DOS.
8. Программы–оболочки.
9. ОС Windows, UNIX, Linux.
10. Транслятор, компилятор, интерпретатор.
11. Табличный процессор.
12. Системы программирования.
13. Инструментальные программы.
14. Текстовый редактор.
15. Графический редактор.
16. Возможности систем деловой и научной графики.
17. Системы управления базами данных.
18. Библиотеки стандартных программ.

19. Пакеты прикладных программ.
20. Интегрированные пакеты программ.
21. Оргонайзеры.
22. Сетевое ПО.
23. Предмет технологии разработки ПО.
24. Жизненный цикл ПО в целом.
25. Традиционный этап разработки программ.
26. Реализация модулей.
27. Связанность модулей.
28. Связность элементов модуля.

10. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ «Прикладное программное обеспечение»

а) основная литература:

1. Безручко В. Т. Информатика (курс лекций): учеб. пособие: рек. НМС / В. Т. Безручко. - М. : ФОРУМ : ИНФРА-М, 2006.
2. Информатика: практикум по технологии работы на компьютере: рек. Мин. обр. РФ / под ред. Н. В. Макаровой. - 3-е изд., перераб. - М. : Финансы и статистика, 2005. - 256 с.
3. Информатика. Базовый курс [Текст] : учебник: рек. Мин. обр. РФ / Ред. С.В. Симонович. - 2-е изд. - СПб. : Питер , 2004, 2005, 2006, 2007. - 640 с.

б) дополнительная литература:

1. Брукшир Дж. Г. Информатика и вычислительная техника [Текст] : [Вводный курс]: Пер. с англ. / Дж. Г. Брукшир. - 7-е изд. - СПб. : Питер, 2004. - 620 с.
2. Захарова И. Г. Информационные технологии в образовании: учеб. пособие: рек. УМО вузов / И.Г. Захарова. - М. : Академия, 2003.
3. Камаев В. А. Технологии программирования [Текст] : учеб.: доп. Мин. обр. РФ / В. А. Камаев, В. В. Костерин. - 2-е изд., перераб. и доп. - М.: Высш. шк., 2006. - 455 с.
4. Могилев А. В. Практикум по информатике [Текст] : учеб. пособие / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера. - М.: Академия, 2002. - 608 с.
5. Семакин И. Г. Основы программирования [Текст] : учебник: Рек. Мин. обр. РФ / И.Г. Семакин, А.П. Шестаков. - 2-е изд., стер. - М.: Академия, 2003. - 432 с.
6. Фуфаев Э.В. Пакеты прикладных программ: учеб. пособие: рек. Мин. обр. РФ / Э. В. Фуфаев, Л. И. Фуфаева. - М. : Академия, 2004. - 352 с.

11. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Стандартно оборудованные лекционные аудитории.

Лаборатория численных методов исследования динамических систем (327 ауд.), оснащенная 8-мью ПК Pentium 2,49 ГГц с ОС Microsoft Windows XP.

II КРАТКОЕ ИЗЛОЖЕНИЕ ПРОГРАММНОГО МАТЕРИАЛА

Тема 1 *Система MAPLE.*

Maple — система компьютерной математики, рассчитанная на широкий круг пользователей. До недавнего времени ее называли системой компьютерной алгебры, И это указывало на особую роль символьных вычислений и преобразований, которые способна осу-

ществлять эта система. Но такое название сужает сферу применения системы. На самом деле она уже способна выполнять быстро и эффективно не только символьные, но и численные расчеты, причем сочетает это с превосходными средствами графической визуализации и подготовки электронных документов.

Maple — типичная интегрированная система. Она объединяет в себе:

- мощный язык программирования (он же язык для интерактивного общения с системой);
- редактор для подготовки и редактирования документов и программ;
- современный многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме;
- мощную справочную систему со многими тысячами примеров;
- ядро алгоритмов и правил преобразования математических выражений;
- численный и символьный процессоры;
- систему диагностики;
- библиотеки встроенных и дополнительных функций;
- пакеты функций сторонних производителей и поддержку некоторых других языков программирования и программ.

Ко всем этим средствам имеется полный доступ прямо из программы. Maple — одна из самых мощных и «разумных» интегрированных систем символьной математики, созданная фирмой Waterloo Maple, Inc. (Канада).

Меню Help

Справочной системе Maple принадлежит исключительная роль — только в ней можно найти полную информацию обо всех почти трех тысячах функций Maple. Использование англоязычной справочной системы может быть полезно и для тех, кто и «двух слов по-английски связать не может», поскольку в ней приведен синтаксис функций и операторов, а также многочисленные примеры их применения — по самым скромным подсчетам их свыше десяти тысяч. К сожалению, справочная система Maple очень громоздка. Но это нельзя считать недостатком справочной системы, поскольку просто велик объем входящего в нее материала. В справочной системе имеются все присущие современным базам данных возможности для быстрого поиска нужной информации и даже для ее структурирования и пополнения.

Основные команды по работе со справочной системой Maple сосредоточены в меню Help,

Оно содержит команды, объединенные в несколько групп. В первую группу входят следующие команды:

- Introduction — показ начального раздела справки (введения);
- Help on Context — вывод оперативной справки по контексту;
- New User's Tours — запуск обучающей системы;
- What's New — описание новых возможностей системы;
- Using Help — описание правил использования справочной системы;
- Glossary — вывод указателя терминов.

Второй раздел меню содержит команды:

- Topic Search — предметный поиск по заданному образцу;
- Full Text Search — предметный поиск с полным обзором текста справки;
- History — вывод истории поиска.

В третьем разделе имеются две команды для работы с базой данных:

- Save to Database — запись данных в базу данных;
- Remove Topic — восстановление базы данных предметного поиска путем удаления дополнительных данных;

Остальные разделы представлены следующими командами:

- Balloon Help — включение всплывающих подсказок;
- Register Maple 7 — регистрация Maple 7;
- About Maple 7 — вывод окна с информацией о Maple 7.

Рассмотрим детально работу справочной системы Maple. Следует отметить, что ценность справочной системы для наших читателей намного снижается из-за того, что она написана на английском языке. Учитывая громоздкость справочной системы и необходимость в наличии компьютера для ее использования, для знакомства с системой Maple более подходят обычные книги, тогда как справочную систему следует применять при необходимости ознакомиться с тонкими деталями применения тех или иных операторов, функций и иных средств Maple

Тема 2 Программное обеспечение (ПО) компьютеров.

Под программным обеспечением (Software) понимается совокупность программ, выполняемых вычислительной системой.

К программному обеспечению (ПО) относится также вся область деятельности по проектированию и разработке ПО:

- технология проектирования программ (например, нисходящее проектирование, структурное и объектно-ориентированное проектирование и др.);
- методы тестирования программ [ссылка, ссылка];
- методы доказательства правильности программ;
- анализ качества работы программ;
- документирование программ;
- разработка и использование программных средств, облегчающих процесс проектирования программного обеспечения, и многое другое.

Программное обеспечение — неотъемлемая часть компьютерной системы. Оно является логическим продолжением технических средств. Сфера применения конкретного компьютера определяется созданным для него ПО.

Сам по себе компьютер не обладает знаниями ни в одной области применения. Все эти знания сосредоточены в выполняемых на компьютерах программах.

Программное обеспечение современных компьютеров включает миллионы программ — от игровых до научных.

Классификация программного обеспечения.

В первом приближении все программы, работающие на компьютере, можно условно разделить на три категории

- прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ;

- системные программы, выполняющие различные вспомогательные функции, например:
  - управление ресурсами компьютера;
  - создание копий используемой информации;
  - проверка работоспособности устройств компьютера;
  - выдача справочной информации о компьютере и др.;
- инструментальные программные системы, облегчающие процесс создания новых программ для компьютера.

При построении классификации ПО нужно учитывать тот факт, что стремительное развитие вычислительной техники и расширение сферы приложения компьютеров резко ускорили процесс эволюции программного обеспечения.

Если раньше можно было по пальцам перечислить основные категории ПО – операционные системы, трансляторы, пакеты прикладных программ, то сейчас ситуация коренным образом изменилась.

Развитие ПО пошло как вглубь (появились новые подходы к построению операционных систем, языков программирования и т.д.), так и вширь (прикладные программы перестали быть прикладными и приобрели самостоятельную ценность).

Соотношение между требующимися программными продуктами и имеющимися на рынке меняется очень быстро. Даже классические программные продукты, такие, как операционные системы, непрерывно развиваются и наделяются интеллектуальными функциями, многие из которых ранее относились только к интеллектуальным возможностям человека.

Кроме того, появились нетрадиционные программы, классифицировать которые по устоявшимся критериям очень трудно, а то и просто невозможно, как, например, программа – электронный собеседник.

На сегодняшний день можно сказать, что более или менее определённо сложились следующие группы программного обеспечения:

- операционные системы и оболочки;
- системы программирования (трансляторы, библиотеки подпрограмм, отладчики и т.д.);
- инструментальные системы;
- интегрированные пакеты программ;
- динамические электронные таблицы;
- системы машинной графики;
- системы управления базами данных (СУБД);
- прикладное программное обеспечение.

Разумеется, эту классификацию нельзя считать исчерпывающей, но она более или менее наглядно отражает направления совершенствования и развития программного обеспечения.

Прикладные программы

Прикладная программа — это любая конкретная программа, способствующая решению какой-либо задачи в пределах данной проблемной области. Например, там, где на компьютер возложена задача контроля за финансовой деятельностью какой-либо фирмы, прикладной будет программа подготовки платежных ведомостей.

Прикладные программы могут носить и общий характер, например, обеспечивать составление и печатание документов и т.п.

В противоположность этому, операционная система или инструментальное ПО не вносят прямого вклада в удовлетворение конечных потребностей пользователя.

Прикладные программы могут использоваться либо автономно, то есть решать поставленную задачу без помощи других программ, либо в составе программных комплексов или пакетов.

Роль и назначение системных программ

Системные программы выполняются вместе с прикладными и служат для управления ресурсами компьютера — центральным процессором, памятью, вводом-выводом.

Это программы общего пользования, которые предназначены для всех пользователей компьютера. Системное программное обеспечение разрабатывается так, чтобы компьютер мог эффективно выполнять прикладные программы.

Среди десятков тысяч системных программ особое место занимают операционные системы, которые обеспечивают управление ресурсами компьютера с целью их эффективного использования.

Важными классами системных программ являются также программы вспомогательного назначения — утилиты (лат. utilitas — польза). Они либо расширяют и дополняют соответствующие возможности операционной системы, либо решают самостоятельные важные задачи.

Тема 3. Операционные системы (ОС) и сети

Назначение и функции операционной системы

Сегодня существует большое количество разных типов операционных систем, отличающихся областями применения, аппаратными платформами и методами реализации. Естественно, это обуславливает и значительные функциональные различия этих ОС. Даже у конкретной операционной системы набор выполняемых функций зачастую определить не так просто — та функция, которая сегодня выполняется внешним по отношению к ОС компонентом, завтра может стать ее неотъемлемой частью и наоборот. Поэтому при изучении операционных систем очень важно из всего многообразия выделить те функции, которые присущи всем операционным системам как классу продуктов.

Операционные системы для автономного компьютера

Операционная система компьютера представляет собой комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями с одной стороны, и аппаратурой компьютера с другой стороны. В соответствии с этим определением ОС выполняет две группы функций:

- предоставление пользователю или программисту вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобней работать и которую легче программировать;
- повышение эффективности использования компьютера путем рационального управления его ресурсами в соответствии с некоторым критерием.

ОС как виртуальная машина

Для того чтобы успешно решать свои задачи, современный пользователь или даже прикладной программист может обойтись без досконального знания аппаратного устройства компьютера. Ему не обязательно быть в курсе того, как функционируют различные электронные блоки и электромеханические узлы компьютера. Более того,

очень часто пользователь может не знать даже системы команд процессора. Пользователь-программист привык иметь дело с мощными высокоуровневыми функциями, которые ему предоставляет операционная система.

Так, например, при работе с диском программисту, пишущему приложение для работы под управлением ОС, или конечному пользователю ОС достаточно представлять его в виде некоторого набора файлов, каждый из которых имеет имя. Последовательность действий при работе с файлом заключается в его открытии, выполнении одной или нескольких операций чтения или записи, а затем в закрытии файла. Такие частности, как используемая при записи частотная модуляция или текущее состояние двигателя механизма перемещения магнитных головок чтения/записи, не должны волновать программиста. Именно операционная система скрывает от программиста большую часть особенностей аппаратуры и предоставляет возможность простой и удобной работы с требуемыми файлами.

Если бы программист работал непосредственно с аппаратурой компьютера, без участия ОС, то для организации чтения блока данных с диска программисту пришлось бы использовать более десятка команд с указанием множества параметров: номера блока на диске, номера сектора на дорожке и т. п. А после завершения операции обмена с диском он должен был бы предусмотреть в своей программе анализ результата выполненной операции. Учитывая, что контроллер диска способен распознавать более двадцати различных вариантов завершения операции, можно считать программирование обмена с диском на уровне аппаратуры не самой тривиальной задачей. Не менее обременительной выглядит и работа пользователя, если бы ему для чтения файла с терминала потребовалось задавать числовые адреса дорожек и секторов.

Операционная система избавляет программистов не только от необходимости напрямую работать с аппаратурой дискового накопителя, предоставляя им простой файловый интерфейс, но и берет на себя все другие рутинные операции, связанные с управлением другими аппаратными устройствами компьютера: физической памятью, таймерами, принтерами и т. д.

В результате реальная машина, способная выполнять только небольшой набор элементарных действий, определяемых ее системой команд, превращается в виртуальную машину, выполняющую широкий набор гораздо более мощных функций. Виртуальная машина тоже управляется командами, но это уже команды другого, более высокого уровня: удалить файл с определенным именем, запустить на выполнение некоторую прикладную программу, повысить приоритет задачи, вывести текст из файла на печать. Таким образом, назначение ОС состоит в предоставлении пользователю/программисту некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальный компьютер или реальную сеть.

ОС как система управления ресурсами

Операционная система не только предоставляет пользователям и программистам удобный интерфейс к аппаратным средствам компьютера, но и является механизмом, распределяющим ресурсы компьютера.

К числу основных ресурсов современных вычислительных систем могут быть отнесены такие ресурсы, как процессоры, основная память, таймеры, наборы данных, диски, накопители на магнитных лентах, принтеры, сетевые устройства и некоторые другие. Ресурсы распределяются между процессами. Процесс (задача) представляет собой

базовое понятие большинства современных ОС и часто кратко определяется как программа в стадии выполнения. Программа — это статический объект, представляющий собой файл с кодами и данными. Процесс — это динамический объект, который возникает в операционной системе после того, как пользователь или сама операционная система решает «запустить программу на выполнение», то есть создать новую единицу вычислительной работы. Например, ОС может создать процесс в ответ на команду пользователя run prgl. exe, где prgl. exe — это имя файла, в котором хранится код программы.

Примечание

Во многих современных ОС для обозначения минимальной единицы работы ОС используют термин «нить», или «поток», при этом изменяется суть термина «процесс». Мы будем придерживаться упрощенного толкования, в соответствии с которым для обозначения выполняемой программы будет использоваться только термин «процесс».

Управление ресурсами вычислительной системы с целью наиболее эффективного их использования является назначением операционной системы. Например, мультипрограммная операционная система организует одновременное выполнение сразу нескольких процессов на одном компьютере, поочередно переключая процессор с одного процесса на другой, исключая простои процессора, вызываемые обращениями процессов к вводу-выводу. ОС также отслеживает и разрешает конфликты, возникающие при обращении нескольких процессов к одному и тому же устройству ввода-вывода или к одним и тем же данным. Критерий эффективности, в соответствии с которым ОС организует управление ресурсами компьютера, может быть различным. Например, в одних системах важен такой критерий, как пропускная способность вычислительной системы, в других — время ее реакции. Соответственно выбранному критерию эффективности операционные системы по-разному организуют вычислительный процесс.

Управление ресурсами включает решение следующих общих, не зависящих от типа ресурса задач:

- планирование ресурса — то есть определение, какому процессу, когда и в каком количестве (если ресурс может выделяться частями) следует выделить данный ресурс;
- удовлетворение запросов на ресурсы;
- отслеживание состояния и учет использования ресурса — то есть поддержание оперативной информации о том, занят или свободен ресурс и какая доля ресурса уже распределена;
- разрешение конфликтов между процессами.

Для решения этих общих задач управления ресурсами разные ОС используют различные алгоритмы, особенности которых в конечном счете и определяют облик ОС в целом, включая характеристики производительности, область применения и даже пользовательский интерфейс. Например, применяемый алгоритм управления процессором в значительной степени определяет, может ли ОС использоваться как система разделения времени, система пакетной обработки или система реального времени.

Задача организации эффективного совместного использования ресурсов несколькими процессами является весьма сложной, и сложность эта порождается в основном случайным характером возникновения запросов на потребление ресурсов. В мультипрограммной системе образуются очереди заявок от одновременно выполняемых программ к разделяемым ресурсам компьютера: процессору, странице памяти, к принтеру, к диску. Операционная система организует обслуживание этих очередей по разным алгоритмам: в порядке поступления, на основе приоритетов, кругового обслуживания и т. д. Анализ и определение оптимальных дисциплин обслуживания заявок является

предметом специальной области прикладной математики — теории массового обслуживания. Эта теория иногда используется для оценки эффективности тех или иных алгоритмов управления очередями в операционных системах. Очень часто в ОС реализуются и эмпирические алгоритмы обслуживания очередей, прошедшие проверку практикой.

Таким образом, управление ресурсами составляет важную часть функций любой операционной системы, в особенности мультипрограммной. В отличие от функций расширенной машины большинство функций управления ресурсами выполняются операционной системой автоматически и прикладному программисту недоступны.

Функциональные компоненты операционной системы автономного компьютера

Функции операционной системы автономного компьютера обычно группируются либо в соответствии с типами локальных ресурсов, которыми управляет ОС, либо в соответствии со специфическими задачами, применимыми ко всем ресурсам. Иногда такие группы функций называют подсистемами. Наиболее важными подсистемами управления ресурсами являются подсистемы управления процессами, памятью, файлами и внешними устройствами, а подсистемами, общими для всех ресурсов, являются подсистемы пользовательского интерфейса, защиты данных и администрирования.

Управление процессами

Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.

Для каждого вновь создаваемого процесса ОС генерирует системные информационные структуры, которые содержат данные о потребностях процесса в ресурсах вычислительной системы, а также о фактически выделенных ему ресурсах. Таким образом, процесс можно также определить как некоторую заявку на потребление системных ресурсов.

Чтобы процесс мог быть выполнен, операционная система должна назначить ему область оперативной памяти, в которой будут размещены коды и данные процесса, а также предоставить ему необходимое количество процессорного времени. Кроме того, процессу может понадобиться доступ к таким ресурсам, как файлы и устройства ввода-вывода.

В информационные структуры процесса часто включаются вспомогательные данные, характеризующие историю пребывания процесса в системе (например, какую долю времени процесс потратил на операции ввода-вывода, а какую на вычисления), его текущее состояние (активное или заблокированное), степень привилегированности процесса (значение приоритета). Данные такого рода могут учитываться операционной системой при принятии решения о предоставлении ресурсов процессу.

В мультипрограммной операционной системе одновременно может существовать несколько процессов. Часть процессов порождается по инициативе пользователей и их приложений, такие процессы обычно называют пользовательскими. Другие процессы, называемые системными, инициализируются самой операционной системой для выполнения своих функций.

Поскольку процессы часто одновременно претендуют на одни и те же ресурсы, то в обязанности ОС входит поддержание очередей заявок процессов на ресурсы, например очереди к процессору, к принтеру, к последовательному порту.

Важной задачей операционной системы является защита ресурсов, выделенных данному процессу, от остальных процессов. Одним из наиболее тщательно защищаемых ресурсов процесса являются области оперативной памяти, в которой хранятся коды и данные процесса. Совокупность всех областей оперативной памяти, выделенных операционной системой процессу, называется его адресным пространством. Говорят, что каждый процесс работает в своем адресном пространстве, имея в виду защиту адресных пространств, осуществляемую ОС. Защищаются и другие типы ресурсов, такие как файлы, внешние устройства и т. д. Операционная система может не только защищать ресурсы, выделенные одному процессу, но и организовывать их совместное использование, например разрешать доступ к некоторой области памяти нескольким процессам.

На протяжении периода существования процесса его выполнение может быть многократно прервано и продолжено. Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды идентифицируется состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т. д. Эта информация называется контекстом прогресса. Говорят, что при смене процесса происходит переключение контекстов.

Операционная система берет на себя также функции синхронизации процессов, позволяющие процессу приостанавливать свое выполнение до наступления какого-либо события в системе, например завершения операции ввода-вывода, осуществляемой по его запросу операционной системой.

В операционной системе нет однозначного соответствия между процессами и программами. Один и тот же программный файл может породить несколько параллельно выполняемых процессов, а процесс может в ходе своего выполнения сменить программный файл и начать выполнять другую программу.

Для реализации сложных программных комплексов полезно бывает организовать их работу в виде нескольких параллельных процессов, которые периодически взаимодействуют друг с другом и обмениваются некоторыми данными. Так как операционная система защищает ресурсы процессов и не позволяет одному процессу писать или читать из памяти другого процесса, то для оперативного взаимодействия процессов ОС должна предоставлять особые средства, которые называют средствами межпроцессного взаимодействия.

Таким образом, подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает синхронизацию процессов, а также обеспечивает взаимодействие между процессами.

Сетевые операционные системы

Операционная система компьютерной сети во многом аналогична ОС автономного компьютера — она также представляет собой комплекс взаимосвязанных программ, который обеспечивает удобство работы пользователям и программистам путем предоставления им некоторой виртуальной вычислительной системы, и реализует

эффективный способ разделения ресурсов между множеством выполняемых в сети процессов.

Компьютерная сеть — это набор компьютеров, связанных коммуникационной системой и снабженных соответствующим программным обеспечением, позволяющим пользователям сети получать доступ к ресурсам этого набора компьютеров. Сеть могут образовывать компьютеры разных типов, которыми могут быть небольшие микропроцессоры, рабочие станции, мини-компьютеры, персональные компьютеры или суперкомпьютеры. Коммуникационная система может включать кабели, повторители, коммутаторы, маршрутизаторы и другие устройства, обеспечивающие передачу сообщений между любой парой компьютеров сети. Компьютерная сеть позволяет пользователю работать со своим компьютером как с автономным и добавляет к этому возможность доступа к информационным и аппаратным ресурсам других компьютеров сети.

При организации сетевой работы операционная система играет роль интерфейса, экранирующего от пользователя все детали низкоуровневых программно-аппаратных средств сети. Например, вместо числовых адресов компьютеров сети, таких как MAC-адрес и IP-адрес, операционная система компьютерной сети позволяет оперировать удобными для запоминания символьными именами. В результате в представлении пользователя сеть с ее множеством сложных и запутанных реальных деталей превращается в достаточно понятный набор разделяемых ресурсов.

Тема 4. Технология программирования

Основы технологии программирования

Проектирование алгоритмов и программ - наиболее ответственный этап жизненного цикла программных продуктов, определяющий, насколько создаваемая программа соответствует спецификациям и требованиям со стороны конечных пользователей. Затраты на создание, сопровождение и эксплуатацию программных продуктов, научно-технический уровень разработки, время морального устаревания и многое другое- все это также зависит от проектных решений.

Методы проектирования алгоритмов и программ очень разнообразны, их можно классифицировать по различным признакам, важнейшими из которых являются:

- степень автоматизации проектных работ;
- принятая методология процесса разработки.

По степени автоматизации проектирования алгоритмов и программ можно выделить:

- методы традиционного (неавтоматизированного) проектирования;
- методы автоматизированного проектирования (CASE-технология и ее элементы).

Проектирование алгоритмов и программ может основываться на различных подходах, среди которых наиболее распространены:

структурное проектирование программных продуктов;

информационное моделирование предметной области и связанных с ней приложений;

объектно-ориентированное проектирование программных продуктов.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные составляющие. Начало развития структурного проектирования алгоритмов и программ падает на 60-е гг. Методы

структурного проектирования представляют собой комплекс технических и организационных принципов системного проектирования.

Типичными методами структурного проектирования являются:
- нисходящее проектирование, кодирование и тестирование программ;
- модульное программирование;
- структурное проектирование (программирование) и др.

Для функционально-ориентированных методов в первую очередь учитываются заданные функции обработки данных, в соответствии с которыми определяется состав и логика работы (алгоритмы) отдельных компонентов программного продукта. С изменением содержания функций обработки, их состава, соответствующего им информационного входа и выхода требуется перепроектирование программного продукта. Основной упор в структурном подходе делается на моделирование процессов обработки данных.

Для методов структурирования данных осуществляется анализ, структурирование и создание моделей данных, применительно к которым устанавливается необходимый состав функций и процедур обработки. Программные продукты тесно связаны со структурой обрабатываемых данных, изменение которой отражается на логике обработки (алгоритмах) и обязательно требует перепроектирования программного продукта.

Объектно-ориентированный подход к проектированию программных продуктов основан на:

выделении классов объектов;

установлении характерных свойств объектов и методов их обработки;

создании иерархии классов, наследовании свойств объектов и методов их обработки.

Каждый объект объединяет как данные, так и программу обработки этих данных и относится к определенному классу. С помощью класса один и тот же программный код можно использовать для относящихся к нему различных объектов.

Объектный подход при разработке алгоритмов и программ предполагает:
- объектно-ориентированный анализ предметной области;
- объектно-ориентированное проектирование.

Для проектирования программных продуктов разработаны объектно-ориентированные технологии, которые включают в себя специализированные языки программирования и инструментальные средства разработки пользовательского интерфейса.

Традиционные подходы к разработке программных продуктов всегда подчеркивали различия между данными и процессами их обработки. Так, технологии, ориентированные на информационное моделирование, сначала специфицируют данные, а затем описывают процессы, использующие эти данные. Технологии структурного подхода ориентированы, в первую очередь, на процессы обработки данных с последующим установлением необходимых для этого данных и организации информационных потоков между связанными процессами.

Объектно-ориентированная технология разработки программных продуктов объединяет данные и процессы в логические сущности - объекты, которые имеют способность наследовать характеристики (методы и данные) одного или более объектов, обеспечивая тем самым повторное использование программного кода. Это приводит к

значительному уменьшению затрат на создание программных продуктов, повышает эффективность жизненного цикла программных продуктов (сокращается длительность фазы разработки).При выполнении программы объекту посылается сообщение, которое инициирует обработку данных объекта.

Парадигма – набор теорий, стандартов и методов, которые совместно представляют собой способ организации научного знания, иными словами, способ видения мира. По аналогии с этим принято считать, что парадигма в программировании – способ концептуализации, который определяет, как следует проводить вычисления, и как работа, выполняемая компьютером, должна быть структурирована и организована.

Известно несколько основных парадигм программирования, важнейшими из которых на данный момент времени являются парадигмы директивного, объектно-ориентированного и функционально-логического программирования. Для поддержки программирования в соответствии с той или иной парадигмой разработаны специальные алгоритмические языки.и Pascal являются примерами языков, предназначенных для директивного программирования (directive programming), когда разработчик программы использует процессно-ориентированная модель, то есть пытается создать код, должным образом воздействующий на данные. Активным началом при этом подходе считается программа (код), которая должна выполнить все необходимые для достижения нужного результата действия над пассивными данными.

Этот подход представляется вполне естественным для человека, который только начинает изучать программирование, и исторически возник одним из первых, однако он практически неприменим для создания больших программ. Первые две главы книги посвящены именно директивному программированию, так как подобный стиль оптимален для программирования в малом, а навыки, которые он позволяет приобрести, необходимы и при использовании других подходов.

Сейчас весьма распространенным стал объектно-ориентированный (object oriented) подход, реализуемый, например, языками C++ и Java. При этом, наоборот, первичными считаются объекты (данные), которые могут активно взаимодействовать друг с другом с помощью механизма передачи сообщений (называемого также и механизмом вызова методов). Функция программиста в этом случае подобна роли бога при сотворении Вселенной – он должен придумать и реализовать такие объекты, взаимодействие которых после старта программы приведет к достижению необходимого конечного результата.

Функциональное и логическое программирование использует языки типа Lisp, Haskell и Prolog. Эта парадигма базируется на принципиально иной трактовке понятия программы. Здесь главным является точная формулировка задачи, а выбор и применение необходимого для ее алгоритма решения -- проблема исполняющей системы, но не программиста. Принцип, на котором зиждется технология структурного программирования – фундаментальная научная и техническая идея о выделении множества базисных элементов, с помощью которых можно выразить (из которых можно собрать) любой объект из некоторого широкого набора.

Итак, основной принцип технологии структурного программирования гласит: для любой простой программы можно построить функционально эквивалентную ей структурную программу, т.е. программу, сформированную на основе фиксированного базисного множества, включающего структуру последовательного действия, структуру выбора одного из двух действий и структуру цикла, то есть многократного повторения некоторого действия с проверкой условия остановки повторения.

Нисходящее и восходящее проектирование

Одна из основных идей, положенных в большинство известных технологий программирования - нисходящее проектирование. Существуют также другие названия: "программирование с пошаговым совершенствованием", "систематическое программирование", "иерархическое программирование". Принцип его - сначала определяются основные функции, которые должны быть обеспечены изготавливаемой программой, а затем доопределяются дополнительные функции, вытекающие из основных.

Вот некоторые принципы нисходящего проектирования:

Подробное формальное и строгое описание проектировщиком входов, функций и выходов всех модулей программ или системы.

Как только Вы убедитесь, что некоторая часть задачи может быть реализована в виде отдельного модуля, Постарайтесь больше не думать об этом.

На каждом уровне проекта пытайтесь записать реализацию модуля в виде символических кодов или блок схемы (размер описания в идеале не должен превосходить одного листа, чтобы при последующем анализе перед глазами находилась наиболее полная картина).

Проектированию структуры данных и их движения следует не меньше времени, чем программе.

Методы проектирования алгоритмов

Методы проектирования алгоритмов включают: нисходящее проектирование, модульность, структурное программирование.

Нисходящее проектирование предполагает последовательное разбиение исходной задачи на подзадачи до такой конкретизации, когда подзадача сможет быть реализована одним оператором выбранного для программирования языка. По ходу нисходящего проектирования та или иная подзадача может сформировать самостоятельный модуль.

III МЕТОДИЧЕСКИЕ УКАЗАНИЯ

3.1 Методические указания по изучению дисциплины

Основной целью курса является изучение основных типов операционных систем, принципы разработки программного обеспечения, организации сетей ЭВМ.

Основными задачами дисциплины являются:

- Выработать практические навыки по освоению программного обеспечения на примере интегрированной системы Maple.
- Изучить схему работы систем программирования, основные положения функционирования и разработки трансляторов (формальные языки и грамматики, типы грамматик, вывод цепочек, конечный и магазинный автоматы, распознаватели и преобразователи, построение автомата по заданной грамматике, структура компиляторов и интерпретаторов, лексический, синтаксический и семантический анализаторы, генератор кода).
- Рассмотреть вопросы распределения ресурсов операционной системой.
- Получить навыки работы с сетями ЭВМ.

3.2 Методические указания к лабораторным занятиям

Правила оформления отчёта по лабораторной работе

По каждой лабораторной работе готовится отчёт. В заголовке отчёта по лабораторной работе указывается:

- название дисциплины;
- номер лабораторной работы;
- тема лабораторной работы;
- фамилия и инициалы студента.

Далее приводится отчёт о выполнении конкретных заданий. Для каждого задания приводится:

- краткое описание задания (задачи);
- решение с подробными комментариями (если это текст программы, то не менее 70 % строк должны содержать комментарии; кроме того, текст программы должен быть структурированным по разделам, блокам, каждому циклу или условному оператору);
- исходные данные и результаты решения;
- результаты решения должны быть оформлены ясны и понятны (каждое число, график, таблица сопровождаются пояснительной надписью или комментарием).

Примерный перечень вопросов к защите лабораторных работ

1. Языки и цепочки символов. Операции над цепочками символов.
2. Понятие языка. Формальное определение языка.
3. Синтаксис и семантика языка. Особенности языков программирования.
4. Определение грамматики. Понятие о грамматике языка.
5. Формальное определение грамматики. Форма Бэкуса—Наура.
6. Принцип рекурсии в правилах грамматики.
7. Классификация грамматик. Четыре типа грамматик по Хомскому.
8. Сентенциальная форма грамматики.
9. Левосторонний и правосторонний выводы.
10. Дерево вывода. Методы построения дерева вывода.
11. Проблемы однозначности и эквивалентности грамматик.
12. Распознаватели. Общая схема распознавателя. Виды распознавателей.
13. Классификация распознавателей по типам языков. Задача разбора (постановка задачи).
14. Регулярные языки и грамматики.
15. Автоматные грамматики.
16. Алгоритм преобразования регулярной грамматики к автоматному виду.
17. Определение конечного автомата.
18. Детерминированные и недетерминированные конечные автоматы. Преобразование конечного автомата к детерминированному виду.
19. Свойства регулярных языков.
20. Распознаватели КС-языков.
21. Автоматы с магазинной памятью. Определение МП-автомата.
22. Эквивалентность языков МП-автоматов и КС-грамматик.
23. Детерминированные МП-автоматы.
24. Свойства КС-языков.
25. Преобразование КС-грамматик. Удаление недостижимых символов. Удаление бесплодных символов. Устранение λ-правил. Устранение цепных правил.
26. КС-грамматики в нормальной форме. Грамматики в нормальной форме Хомского. Устранение левой рекурсии. Грамматики в нормальной форме Грейбах.
27. Распознаватели КС-языков с возвратом. Принципы работы распознавателей с возвратом. Нисходящий распознаватель с возвратом.
28. Распознаватель на основе алгоритма «сдвиг-свертка». Табличные распознаватели для КС-языков. Общие принципы работы табличных распознавателей.

29. Нисходящие распознаватели КС-языков без возвратов.
30. Восходящие распознаватели- КС-языков без возвратов.
31. Грамматики предшествования (основные принципы). Грамматики простого предшествования. Грамматики операторного предшествования.
32. Определение транслятора, компилятора, интерпретатора.
33. Этапы трансляции. Общая схема работы транслятора.
34. Многопроходные и однопроходные компиляторы.
35. Интерпретаторы. Особенности построения интерпретаторов
36. Организация таблиц идентификаторов. Назначение и особенности построения таблиц идентификаторов.
37. Простейшие методы построения таблиц идентификаторов.
38. Построение таблиц идентификаторов по методу бинарного дерева. Хэш-функции и хэш-адресация.
39. Комбинированные способы построения таблиц идентификаторов.
40. Лексические анализаторы (сканеры). Принципы построения сканеров. Назначение лексического анализатора.
41. Принципы построения лексических анализаторов. Построение лексических анализаторов. Автоматизация построения лексических анализаторов.
42. Синтаксические анализаторы. Синтаксически управляемый перевод. Основные принципы работы синтаксического анализатора.
43. Дерево разбора. Преобразование дерева разбора в дерево операций. Автоматизация построения синтаксических анализаторов .
44. Назначение семантического анализа. Этапы семантического анализа.
45. Идентификация лексических единиц языков программирования. Распределение памяти. Принципы распределения памяти. Дисплей памяти процедуры (функции).
46. Стековая организация дисплея памяти.
47. Генерация кода. Методы генерации кода. Общие принципы генерации кода.
48. Синтаксически управляемый перевод.
49. Способы внутреннего представления программ.
50. Обратная польская запись операций.

IV КОНТРОЛЬ ЗНАНИЙ

По дисциплине осуществляется текущий, промежуточный контроль на дневном отделении и итоговый контроль в форме экзамена.

Виды контроля	Формы контроля
Текущий	Устные опросы на занятиях, выполнение индивидуальных и практических заданий, защита индивидуальных заданий, тестирование
Промежуточный	Выполнение самостоятельной работы
Итоговый	Экзамен