Федеральное агентство по образованию

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ГОУВПО «АмГУ»

УТВЕРЖДАЮ

Зав.кафедрой БЖД

_____А.Б.Булгаков

«_____»_____2007г.

Программные комплексы в БЖД

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

для специальности 280101 «Безопасность жизнедеятельности в техносфере»

Составитель: Дрюков А. А., ассистент кафедры БЖД

Благовещенск 2007 г.

Печатается по решению
редакционно-издательского совета
инженерно-физического факультета
Амурского государственного
университета

А. А. Дрюков

Учебно-методический комплекс по дисциплине «Программные комплексы в БЖД» для студентов очной и заочной сокращенной форм обучения специальности 280101 «Безопасность жизнедеятельности в техносфере». - Благовещенск: Амурский гос. ун-т, 2007. – ____ с.

Учебно-методические рекомендации ориентированы на оказание помощи студентам очной и заочной форм обучения по специальности 280101 «Безопасность жизнедеятельности в техносфере» для формирования представления о применении программных комплексах при решении функциональных задач в области безопасности жизнедеятельности, о видах и областях применения программных комплексов, о возможности преобразования обеспечивающих информационных технологий в функциональные объединения, дать основные понятия информации, ее сбора, хранения и переработки с применением совокупности различных средств и методов.

СОДЕРЖАНИЕ

1. Рабочая программа 4
2. Перечень вопросов для проведения зачета со студентами заочной со-кращенной формы обучения 13
3. Требования и порядок сдачи зачета по дисциплине 13
4. Методические рекомендации для проведения самостоятельной работы студентов 14
5. Содержание курса лекций по дисциплине «Программные комплексы в БЖД» 15
Тема 1. Проектирование программных комплексов 15
Тема 2. Модель предметной области 31
Тема 3. Внешнее управление программным комплексом 42
Тема 4. Проектирование модулей программного комплекса 46
Тема 5. Особенности организации управления программным комплексом с вход-ным языком типа меню 49
Тема 6. Планирование вычислительного процесса. Постановка задачи 53
Тема 7. Проектирование обслуживающих модулей программного Комплекса 65
Тема 8. Применение программных комплексов в безопасности Жизнедеятельности 71
6. Методические указания для проведения и выполнения лабораторных занятий 87
Тема 1. Изучение правовой информационной системы Консультант+ 87
Тема 2. Изучение правовой информационной системы Гарант 87
Тема 3. Работа с данными в MS Excel 88
Тема 4. Расчеты с помощью Mathcad 92
Тема 5. Основы работы в MATlab 96
Тема 6. Расчеты с помощью MATlab 106
7. Перечень программных продуктов, реально используемых в практике деятельности выпускников 107
8. Комплекты заданий для лабораторных работ 107
9. Комплекты экзаменационных билетов для экзамена по «Программные комплексы в БЖД» 107

Федеральное агентство по образованию РФ

Амурский государственный университет

УТВЕРЖДАЮ
Проректор по УНР
_____Е.С. Астапова
подпись, И.О.Ф

«__» _____ 2006 г.

РАБОЧАЯ ПРОГРАММА

по дисциплине «Программные комплексы в БЖД»
(наименование дисциплины)

для специальности 280101, Безопасность жизнедеятельности в техносфере
(шифр и наименование специальности)

Курс 3 Семестр 6

Лекции 17 (час.) Экзамен нет
(семестр)

Лабораторные занятия 34 (час.) Зачет 6
(семестр)

Самостоятельная работа 29 (час.)

Всего часов 80

Составитель А. А. Дрюков, ассистент
(И.О.Ф., должность, ученое звание)

Факультет инженерно-физический

Кафедра БЖД

2006 г.

Рабочая программа авторская

Рабочая программа обсуждена на заседании кафедры БЖД

«_____» _____2006г. Протокол № _____

Заведующий кафедрой _____А.Б.Булгаков

Рабочая программа одобрена на заседании УМС 280101 (БЖД в техносфере)
(наименование специальности)

«__» _____200___ г., протокол № _____

Председатель _____ О.Т. Аксенова
(подпись, И.О.Ф.)

Рабочая программа переутверждена на заседании кафедры от _____ протокол № _____ .

Зав.кафедрой _____ _____
подпись Ф.И.О.

СОГЛАСОВАНО

Начальник УМУ

_____ Г.Н.Торопчина

«_____»_____2006г.

СОГЛАСОВАНО

Председатель УМС ИФФ

_____

«_____»_____2006г.

СОГЛАСОВАНО

Заведующий выпускающей кафедрой

_____ А.Б. Булгаков

«____»_____2006г.

1. Цели и задачи дисциплины, ее место в учебном процессе

1.1. Цель преподавания дисциплины

Основной целью дисциплины «Программные комплексы в БЖД» является подготовка специалистов к практической инженерной деятельности в области управления безопасностью жизнедеятельности с использованием программных комплексов. Формирование у студентов теоретических знаний. Познакомить с основными этапами проектирования программных комплексов, показать их специфику для решения практических задач.

1.2 Задачи изучения дисциплины

В результате изучения дисциплины студенты должны уметь пользоваться прикладным компьютерным программным обеспечением, знать какие прикладные программы используются в программных комплексах, применяемых в безопасности жизнедеятельности, принципы их работы.

1.3. Перечень дисциплин, усвоение которых студентами необходимо при изучении данной дисциплины

Курс базируется на ряде дисциплин регионального компонента – «Базы и банки данных в БЖД», дисциплин по выбору – «Программное обеспечение задач в БЖД», утвержденные Советом АмГУ. Эти дисциплины являются разделами государственного стандарта дисциплины «Информационные технологии в управлении безопасностью жизнедеятельности».

2.Содержание дисциплины

Дисциплина «Программные комплексы в БЖД» является одной из основных дисциплин для изучения остальных связанных с информационными системами, информационными технологиями и базами данных в частности дисциплины «Информационные технологии в управлении безопасностью жизнедеятельности» которая является федеральным компонентом. Дисциплина введена по решению совета ВУЗа, входит в блок ОПД.В и является факультативной дисциплиной.

2.1. Наименование тем, их содержание, объем в часах лекционных занятий

Тема 1. Проектирование программных комплексов (2 часа)
Теоретические основы проектирования программных комплексов: определение и составные части. Применение методо-ориентированных и проблемно-ориентированных ППП в программных комплексах. Интегрированные ППП.

Тема 2. Модель предметной области (2 часа)
Данные, используемые в программном комплексе. Описание модели. Типы данных. Структура данных. Функциональные связи. Способы обработки дан-

ных. Анализ модели предметной области. Группы данных. Входные данные. Выходные данные.

Тема 3. Внешнее управление программным комплексом (2 часа)

Цель применения программного комплекса. Формирование задания на выполнение расчетов. Виды внешнего управления программным комплексом. Функции управляющих и обслуживающих модулей программного комплекса. Понятие оболочки программного комплекса.

Тема 4. Проектирование управляющих модулей программного комплекса (2 часа)

Анализ средств внешнего управления комплексом. Организация управления программного комплекса с входным языком командного типа. Метод интерпретации. Метод компиляции.

Тема 5. Особенности организации управления программным комплексом с входным языком типа меню (2 часа)

Деление действий на группы. Конечный автомат. Алгоритм ведущего модуля, управляемого посредством меню.

Тема 6. Планирование вычислительного процесса. Постановка задачи (2 часа)

Последовательность вызовов, обрабатывающих модулей. Задача оптимального планирования вычислительного процесса. Управление памятью в программном комплексе.

Тема 7. Проектирование обслуживающих модулей программного комплекса (2 часа)

Функции обслуживающих модулей. Особенности реализации интерфейса с пользователем. Справочный интерфейс. Интерфейс управления. Информационный интерфейс.

Тема 8. Применение программных комплексов в безопасности жизнедеятельности (3 часа)

Универсальные программы расчета загрязнения атмосферы: Атмосфера, Атмосфера-ПДВ, ПДВ-Эколог. Программы расчета распределения вредных веществ в водных объектах: ПДС-Эколог, Сток, Труд-эколог, Эколог и др.. Программы расчета объемов выбросов, сбросов и количества твердых отходов различных производств и технологических процессов – Отходы автотранспорта, Отходы котельных, отходы строительства, отходы деревообработки.

2.2. Лабораторные занятия, их наименование, содержание и объем в часах

1. Изучение правовой информационной системы Консультант+ – 8 часов.

Изучение всех разделов правовой информационной системы Консультант+, изучение всех возможных в ней методов и структур поиска необходимой информации, изучения инструментария правовой информационной системы Консультант+, поиск нужной информации в документах по индивидуальному заданию.

2. Изучение правовой информационной системы Гарант – 4 часа.
Изучение всех разделов правовой информационной системы Гарант, изучение всех возможных в ней методов и структур поиска необходимой информации, изучения инструментария правовой информационной системы Гарант, поиск нужной информации в документах по индивидуальному заданию.

3. Работа с данными в MS Excel – 4 часа.
Скрытие и отображение столбцов и строк. Структуризация данных. Сортировка данных. Отбор данных: использование автофильтра, использование расширенного автофильтра. Создание списка. Работа со списком. Типы диаграмм. Построение диаграмм. Настройка и редактирование диаграмм: изменение источника данных, текста, шрифта, оформление осей и линий сетки, оформление рядов данных.

4. Расчеты по электробезопасности с помощью Mathcad – 8 часов.
Решение задач по электробезопасности средствами Mathcad.

5. Основы работы в MATlab – 4 часа.
Изучение основ работы в MATlab. Основы графической визуализации вычислений. Обработка данных. Матричные вычисления. Построение графиков функций. Столбцовые диаграммы. Построение трехмерных графиков. Форматирование линий графиков. Работа со справкой и примерами.

6. Расчеты по электробезопасности с помощью MATlab – 6 часов.
Решение задач по электробезопасности средствами MATlab.

2.4. Вопросы к зачету

1. Теоретические основы проектирования программных комплексов: определение и составные части.
2. Применение методо-ориентированных и проблемно-ориентированных ППП в программных комплексах.
3. Интегрированные ППП.
4. Данные, используемые в программном комплексе.
5. Описание модели.
6. Типы данных.
7. Структура данных.
8. Функциональные связи.
9. Способы обработки данных.

10. Анализ модели предметной области.
11. Группы данных. Входные данные. Выходные данные.
12. Внешнее управление программным комплексом
13. Цель применения программного комплекса. Формирование задания на выполнение расчетов.
14. Виды внешнего управления программным комплексом.
15. Функции управляющих и обслуживающих модулей программного комплекса.
16. Понятие оболочки программного комплекса.
17. Проектирование управляющих модулей программного комплекса
18. Анализ средств внешнего управления комплексом.
19. Организация управления программного комплекса с входным языком командного типа.
20. Метод интерпретации.
21. Метод компиляции.
22. Деление действий на группы. Конечный автомат.
23. Алгоритм ведущего модуля, управляемого посредством меню.
24. Планирование вычислительного процесса. Постановка задачи
25. Последовательность вызовов, обрабатывающих модулей.
26. Задача оптимального планирования вычислительного процесса.
27. Управление памятью в программном комплексе.
28. Проектирование обслуживающих модулей программного комплекса
29. Функции обслуживающих модулей.
30. Особенности реализации интерфейса с пользователем.
31. Справочный интерфейс. Интерфейс управления.
32. Информационный интерфейс.
33. Применение программных комплексов в безопасности жизнедеятельности
34. Универсальные программы расчета загрязнения атмосферы: Атмосфера, Атмосфера-ПДВ, ПДВ-Эколог.
35. Программы расчета распределения вредных веществ в водных объектах: ПДС-Эколог, Сток.
36. Программы расчета объемов выбросов, сбросов и количества твердых отходов различных производств и технологических процессов – Отходы автотранспорта, Отходы котельных, отходы строительства, отходы деревообработки.

3. Учебно-методические материалы по дисциплине

3.1. Список рекомендуемой литературы

Основная

1. Информатика: Учебник. Под ред. проф. Н.В. Макаровой. – М. : Финансы и статистика, 1999. – 768с.
2. Excel 2003 для пользователя: полное руководство. – Ь.: Кудиц-Образ, 2003. – 528с.
3. Лавренов С.М. Excel. Сборник примеров и задач. – М.: Финансы и статистика, 2003. – 336с.

Дополнительная

4. Златопольский Д.М. 1700 заданий по Microsoft Excel. – СПб.: БХВ-Петербург, 2003. – 544с.
5. Гроппен В.О., Гвритишвили П.П. Проектирование оптимальных программных комплексов. – Владикавказ, СВГТУ. – 2002. – 146с.

4. Учебно-методическая (технологическая) карта дисциплины.

Номер темы	Темы лекционных занятий	Количество часов лекционных занятий	Темы лабораторных работ	Количество часов лабораторных занятий	Самостоятельная работа студентов		Формы контроля
					содержание	часы	
1	Проектирование программных комплексов	2	Ввод и редактирование данных в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
2	Модель предметной области	2	Форматирование данных в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
3	Внешнее управление программным комплексом	2	Вычисления в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
4	Проектирование управляющих модулей программного комплекса	2	Вычисления с использованием функций в MS Excel	4	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
5	Особенности организации управления программным комплексом с входным языком типа меню	2	Функции в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
6	Планирование вычислительного процесса. Постановка задачи	2	Условное форматирование в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
7	Проектирование обслуживающих модулей программного комплекса	2	Работа с данными в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
8	Применение программных комплексов в безопасности жизнедеятельности	7	Автоматическое вычисление итогов и оформление списков	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
9	-	-	Использование представлений и сценариев в MS Excel	4	Изучить порядок выполнения лабораторной работы	1	Текущий контроль

Номер темы	Темы лекционных занятий	Количество часов лекционных занятий	Темы лабораторных работ	Количество часов лабораторных занятий	Самостоятельная работа студентов		Формы контроля
					содержание	часы	
1	Проектирование программных комплексов	2	Ввод и редактирование данных в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
10	-	-	Диаграммы в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
11	-	-	Сводные таблицы в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
12	-	-	Блоки в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
13	-	-	Табличные формулы в MS Excel	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
14	-	-	Оптимизация	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль
15	-	-	Регрессия	2	Изучить порядок выполнения лабораторной работы	1	Текущий контроль

5. **Основные критерии оценки знаний студентов по дисциплине**

«Программные комплексы в БЖД»

Студенты обязаны сдать зачет в строгом соответствии с учебным планом, а также утвержденным программам, едиными для всех форм обучения.

Зачет по дисциплине «Программные комплексы в БЖД» служит формой контроля усвоения дисциплины в целом.

К зачету допускаются студенты, выполнившие и защитившие в полном объеме лабораторные работы.

Сроки проведения зачета устанавливаются графиком учебного процесса, утвержденным проректором по учебной работе.

Знания, умения и навыки обучающегося определяются оценками «зачтено» и «не зачтено». Критерии приведены в таблице.

Основные критерии оценки знаний студентов

Оценка	Полнота, системность, прочность знаний	Обобщенность знаний
«зачтено»	Изложение полученных знаний в устной, письменной или графической форме, полное, в системе, в соответствии с требованиями учебной программы; допускаются единичные несущественные ошибки, самостоятельно исправляемые студентами	Выделение существенных признаков изученного с помощью операций анализа и синтеза; выявление причинно-следственных связей; формулировка выводов и обобщений; свободное оперирование известными фактами и сведениями с использованием сведений из других предметов
	Изложение полученных знаний в устной, письменной и графической форме, полное, в системе, в соответствии с требованиями учебной программы; допускаются отдельные несущественные ошибки, исправляемые студентами после указания преподавателя на них	Выделение существенных признаков изученного с помощью операций анализа и синтеза; выявлений причинно-следственных связей; формулировка выводов и обобщений, в которых могут быть отдельные несущественные ошибки; подтверждение изученного известными фактами и сведениями
	Изложение полученных знаний неполное, однако это не препятствует усвоению последующего программного материала; допускаются отдельные существенные ошибки, исправленные с помощью преподавателя	Затруднения при выполнении существенных признаков изученного, при выявлении причинно-следственных связей и формулировке выводов
«не зачтено»	Изложение учебного материала неполное, бессистемное, что препятствует усвоению последующей учебной информации; существенные ошибки, неисправляемые даже с помощью преподавателя	Бессистемное выделение случайных признаков изученного; неумение производить простейшие операции анализа и синтеза; делать обобщения, выводы

2. Перечень вопросов для проведения зачета со студентами заочной сокращенной формы обучения

1. Инструментарий технологии программирования.
2. Проблемно-ориентированные ППП.
3. ППП автоматизированного проектирования.
4. ППП общего назначения.
5. Методо-ориентированные ППП.
6. Офисные ППП.
7. Электронные таблицы (Excel).
8. Пакеты для инженерных и научных расчетов (Mathcad).
9. Программы архиваторы.
10. Сущность и проявление компьютерных вирусов.
11. Классификация компьютерных вирусов.
12. Данные, используемые в программном комплексе.
13. Описание модели.
14. Типы данных.
15. Структура данных.
16. Функциональные связи.
17. Способы обработки данных.
18. Анализ модели предметной области.
19. Группы данных. Входные данные. Выходные данные.
20. Внешнее управление программным комплексом
21. Цель применения программного комплекса. Формирование задания на выполнение расчетов.
22. Виды внешнего управления программным комплексом.
23. Функции управляющих и обслуживающих модулей программного комплекса.
24. Понятие оболочки программного комплекса.
25. Проектирование управляющих модулей программного комплекса
26. Анализ средств внешнего управления комплексом.
27. Организация управления программного комплекса с входным языком командного типа.
28. Метод интерпретации.
29. Метод компиляции.
30. Деление действий на группы. Конечный автомат.
31. Алгоритм ведущего модуля, управляемого посредством меню.

3. Требования и порядок сдачи зачета по дисциплине

Зачет сдается в период зачетной недели экзаменационной сессии. Зачет проводится в объеме программы учебной дисциплины. Форма сдачи зачета – устная. При устной форме зачета преподавателю предоставляется право задавать студенту по программе курса дополнительные вопросы, а также помимо

теоретических вопросов, давать практические задания по программе данного курса.

Преподаватель на зачете учитывает не только ответы на вопросы, но наличие всех выполненных лабораторных работ и посещаемость студента в семестре в случае пропуска лекций выполняется проверочная работа по темам пропущенных лекций. Отвечая на зачете, студент должен дать развернутый ответ. При этом показать знание теории и продемонстрировать свободную ориентацию в указанном материале, знание понятий и терминологии, ответить на уточняющие вопросы.

4. Методические рекомендации для проведения самостоятельной работы студентов по дисциплине «Программные комплексы в БЖД»

Самостоятельная работа студентов при изучении «Программные комплексы в БЖД» включает выполнение контрольной работы по следующим темам и вопросам

Тема 1: Структура данных.

Литература
1. Программное обеспечение в области охраны окружающей среды: аналитический обзор. – М. : Приборы и системы. – 1997

Тема 2: Функциональные связи.

Литература
1. Растоскуев В.В. Экспертная система для обработки данных контроля загрязнения атмосферы. – СПб. : 1997. – 261с.

Тема 3: Анализ модели предметной области.

Литература
1.Ларичев О.И., Мошкович Е.М. Качественные методы принятия решений.

2.Литвак Б.Г. Экспертная информация: методы получения и анализа.

3.Цветков В.Я. Геоинформационные системы и технологии.

5. Содержание курса лекций по дисциплине «Программные комплексы в БЖД»

Тема 1. Проектирование программных комплексов (2 часа)
Лекция №1

Вопросы:

1) Принципы построения программных комплексов.

2) Состав программных комплексов.

3) Основные принципы построения программных комплексов.

4) Стадии создания программных комплексов.

5) Отображение процесса проектирования в программное обеспечение программных комплексов.

Проектирование программного комплекса на примере САПР

1) Принципы построения САПР.

Различные возможности и границы применения вычислительной техники для автоматизации проектирования определяются уровнем формализации научно-технических знаний в конкретной отрасли. Чем глубже разработана теория того или иного класса технических систем, тем большие возможности объективно существуют для автоматизации процесса их проектирования.

Применение ЭВМ при проектных работах в своем развитии прошло несколько стадий и претерпело значительные изменения. С появлением вычислительной техники был сделан акцент на автоматизацию проектных задач, имеющих четко выраженный расчетный характер, когда реализовывались методики, ориентированные на ручное проектирование. Затем, по мере накопления опыта, стали создавать программы автоматизированных расчетов на основе методов вычислительной математики (параметрическая оптимизация, метод конечных элементов и т. п.). С внедрением специализированных терминальных устройств появляются универсальные программы для ЭВМ для решения как расчетных, так и некоторых рутинных проектных задач (изготовление чертежей, спецификаций, текстовых документов и т.

п.). В последние годы большое внимание уделяется автоматизации расчетно-конструкторских работ при проектировании типовых узлов и агрегатов, когда синтез конструкции проводится эвристически, а основные параметры выбираются и оптимизируются в интерактивном режиме диалога проектировщика и ЭВМ.

Однако на всех этих стадиях автоматизации проектирования инженеру помимо изучения инструкций по эксплуатации и написанию программ приходится познавать ряд по сути дела ненужных ему подробностей системных программ и языков программирования. Кроме того, при использовании в проектировании специализированных по объектам разрозненных пакетов прикладных программ (ППП) инженер вынужден каждый раз вновь кодировать и вводить информацию согласно инструкции ППП. Отмеченные недостатки приводят к тому, что частичная («позадачная») автоматизация не оказала существенного влияния на повышение качества и производительности проектирования технических систем и средств в целом.

Решение проблем автоматизации проектирования с помощью ЭВМ основывается на системном подходе, т. е. на создании и внедрении САПР — систем автоматизированного проектирования технических объектов, которые решают весь комплекс задач от анализа задания до разработки полного объема конструкторской и технологической документации. Это достигается за счет объединения современных технических средств и математического обеспечения, параметры и характеристики которых выбираются с максимальным учетом особенностей задач проектно-конструкторского процесса. САПР представляет собой крупные организационно-технические системы, состоящие из комплекса средств автоматизации проектирования, взаимосвязанного с подразделениями конкретной проектной организации.

Цель создания САПР

Под автоматизацией проектирования понимают систематическое применение ЭВМ в процессе проектирования при научно обоснованном распределении

функций между проектировщиком и ЭВМ и научно обоснованном выборе методов машинного решения задач.

Цель автоматизации в САПР — повысить качество проектирования, снизить материальные затраты на него, сократить сроки проектирования и ликвидировать рост числа инженерно-технических работников, занятых проектированием и конструированием.

Научно обоснованное распределение функций между человеком и ЭВМ подразумевает, что человек должен решать задачи, носящие творческий характер, а ЭВМ — задачи, решение которых поддается алгоритмизации.

Существенным отличием автоматизированного проектирования от неавтоматизированного является возможность замены дорогостоящего и занимающего много времени физического моделирования — математическим моделированием. При этом следует иметь в виду одно важнейшее обстоятельство: при проектировании число вариантов необозримо. Поэтому нельзя ставить задачу создания универсальной САПР, а необходимо решать вопросы проектирования для конкретного семейства машин.

Для создания САПР необходимо:

- совершенствовать проектирование на основе применения математических методов и средств вычислительной техники;
- автоматизировать процессы поиска, обработки и выдачи информации;
- использовать методы оптимального и вариантного проектирования;
- применять эффективные, отражающие существенные особенности, математические модели проектируемых объектов, комплектующих изделий и материалов;
- создавать банки данных, содержащих систематизированные сведения справочного характера, необходимые для автоматизированного проектирования объектов;
- повышать качество оформления проектной документации;
- повышать творческую долю труда проектировщиков за счет автоматизации нетворческих работ;

- унифицировать и стандартизовать методы проектирования;

- подготавливать и переподготавливать специалистов;

- реализовывать взаимодействие с автоматизированными системами различного уровня и назначения.

Комплекс средств автоматизации проектирования включает методическое, лингвистическое, математическое, программное, техническое, информационное и организационное обеспечение.

2) Состав САПР.

САПР — система, объединяющая технические средства, математическое и программное обеспечение, параметры и характеристики которых выбирают с максимальным учетом особенностей задач инженерного проектирования и конструирования. В САПР обеспечивается удобство использования программ за счет применения средств оперативной связи инженера с ЭВМ, специальных проблемно-ориентированных языков и наличия информационно-справочной базы.

Структурными составными составляющими САПР являются подсистемы, обладающие всеми свойствами систем и создаваемые как самостоятельные системы. Это выделенные по некоторым признакам части САПР, обеспечивающие выполнение некоторых законченных проектных задач с получением соответствующих проектных решений и проектных документов.

По назначению подсистемы САПР разделяют на два вида: проектирующие и обслуживающие.

К проектирующим относятся подсистемы, выполняющие проектные процедуры и операции, например:

- подсистема компоновки машины;

- подсистема проектирования сборочных единиц;

- подсистема проектирования деталей;

- подсистема проектирования схемы управления;

- подсистема технологического проектирования.

К обслуживающим относятся подсистемы, предназначенные для поддержания работоспособности проектирующих подсистем, например:

- подсистема графического отображения объектов проектирования;
- подсистема документирования;
- подсистема информационного поиска и др.

В зависимости от отношения к объекту проектирования различают два вида проектирующих подсистем:

- объектно-ориентированные (объектные);
- объектно-независимые (инвариантные).

К объектным подсистемам относят подсистемы, выполняющие одну или несколько проектных процедур или операций, непосредственно зависимых от конкретного объекта проектирования, например:

- подсистема проектирования технологических систем;
- подсистема моделирования динамики, проектируемой конструкции и др.

К инвариантным подсистемам относят подсистемы, выполняющие унифицированные проектные процедуры и операции, например:

- подсистема расчетов деталей машин;
- подсистема расчетов режимов резания;
- подсистема расчета технико-экономических показателей и др.

Процесс проектирования реализуется в подсистемах в виде определенной последовательности проектных процедур и операций. Проектная процедура соответствует части проектной подсистемы, в результате выполнения которой принимается некоторое проектное решение. Она состоит из элементарных проектных операции, имеет твердо установленный порядок их выполнения и направлена на достижение локальной цели в процессе проектирования. Под проектной операцией понимают условно Выделенную часть проектной процедуры или элементарное действие, совершаемое конструктором в процессе проектирования. **Примерами проектных процедур могут служить процедуры разработки кинематической или компоновочной схемы станка, технологии обработки изделий и т. п.,**

а примерами проектных операций — расчет припусков, решение какого-либо уравнения и т. п.

Структурное единство подсистем САПР обеспечивается строгой регламентацией связей между различными видами обеспечения, объединенных общей для данной подсистемы целевой функцией. Различают следующие виды обеспечения:

- методическое обеспечение — документы, в которых отражены состав, правила отбора и эксплуатации средств автоматизации проектирования;
- лингвистическое обеспечение — языки проектирования, терминология;
- математическое обеспечение — методы, математические модели, алгоритмы;
- программное обеспечение — документы с текстами программ, программы на машинных носителях и эксплуатационные документы;
- техническое обеспечение — устройства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства и их сочетания;
- информационное обеспечение — документы, содержащие описание стандартных проектных процедур, типовых проектных решений, типовых элементов, комплектующих изделий, материалов и другие данные;
- организационное обеспечение — положения и инструкции, приказы, штатное расписание и другие документы, регламентирующие организационную структуру подразделений и их взаимодействие с комплексом средств автоматизации проектирования.

3) Основные принципы построения САПР.

Разработка САПР представляет собой крупную научно-техническую проблему, а ее внедрение требует значительных капиталовложений. Накопленный опыт позволяет выделить следующие основные принципы построения САПР.

1. САПР — человеко-машинная система. Все созданные и создаваемые системы проектирования с помощью ЭВМ являются автоматизированными, важ-

ную роль в них играет человек — инженер, разрабатывающий проект технического средства.

В настоящее время и по крайней мере в ближайшие годы создание систем автоматического проектирования не предвидится, и ничто не угрожает монополии человека при принятии узловых решении в процессе проектирования. Человек в САПР должен решать, во-первых, все задачи, которые не формализованы, во-вторых, задачи, решение которых человек осуществляет на основе своих эвристических способностей более эффективно, чем современная ЭВМ на основе своих вычислительных возможностей. Тесное взаимодействие человека и ЭВМ в процессе проектирования — один из принципов построения и эксплуатации САПР.

2. САПР — иерархическая система, реализующая комплексный подход к автоматизации всех уровней проектирования. Иерархия уровней проектирования отражается в структуре специального программного обеспечения САПР в виде иерархии подсистем.

Следует особо подчеркнуть целесообразность обеспечения комплексного характера САПР, так как автоматизация проектирования лишь на одном из уровней оказывается значительно менее эффективной, чем полная автоматизация всех уровней. Иерархическое построение относится не только к специальному программному обеспечению, но и к техническим средствам САПР, разделяемых на центральный вычислительный комплекс и автоматизированные рабочие места проектировщиков.

3. САПР — совокупность информационно-согласованных подсистем. Этот очень важный принцип должен относиться не только к связям между крупными подсистемами, но и к связям между более мелкими частями подсистем. Информационная согласованность означает, что все или большинство возможных последовательностей задач проектирования обслуживаются информационно согласованными программами. Две программы являются информационно согласованными, если все те данные, которые представляют собой объект переработки в обеих программах, входят

в числовые массивы, не требующие изменений при переходе от одной программы к другой. Так, информационные связи могут проявляться в том, что результаты решения одной задачи будут исходными данными для другой задачи. Если для согласования программ требуется существенная переработка общего массива с участием человека, который добавляет недостающие параметры, вручную перекомпоновывает массив или изменяет числовые значения отдельных параметров, то программы информационно не согласованы. Ручная перекомпоновка массива ведет к существенным временным задержкам, росту числа ошибок и поэтому уменьшает спрос на услуги САПР. Информационная несогласованность превращает САПР в совокупность автономных программ, при этом из-за неучета в подсистемах многих факторов, оцениваемых в других подсистемах, снижается качество проектных решений.

4. САПР — открытая и развивающаяся система. Существует, по крайней мере, две веские причины, по которым САПР должна быть изменяющейся во времени системой. Во-первых, разработка столь сложного объекта, как САПР, занимает продолжительное время, и экономически выгодно вводить в эксплуатацию части системы по мере их готовности. Введенный в эксплуатацию базовый вариант системы в дальнейшем расширяется. Во-вторых, постоянный прогресс техники, проектируемых объектов, вычислительной техники и вычислительной математики приводит к появлению новых, более совершенных математических моделей и программ, которые должны заменять старые, менее удачные аналоги. Поэтому САПР должна быть открытой системой, т. е. обладать свойством удобства использования новых методов и средств.

5. САПР — специализированная система с максимальным использованием унифицированных модулей. Требования высокой эффективности и универсальности, как правило, противоречивы. Применительно к САПР это положение сохраняет свою силу. Высокой эффективности САПР, выражаемой прежде всего малыми временными и материальными затратами при решении проектных задач, добиваются за счет специализации систем.

Очевидно, что при этом растет число различных САПР. Чтобы снизить расходы на разработку многих специализированных САПР, целесообразно строить их на основе максимального использования унифицированных составных частей. Необходимым условием унификации является поиск общих черт и положений в моделировании, анализе и синтезе разнородных технических объектов. Безусловно, может быть сформулирован и ряд других принципов, что подчеркивает многосторонность и сложность проблемы САПР.

4) Стадии создания САПР.

Создание и развитие САПР осуществляется самой проектной организацией с привлечением (при необходимости) других организации-соисполнителей, в том числе научно-исследовательских институтов и высших учебных заведений. Следует подчеркнуть, что создание САПР — сложная и трудоемкая работа, выполнение которой под силу только большому высококвалифицированному коллективу разработчиков.

Процесс создания САПР включает в себя восемь стадий: предпроектные исследования, техническое задание, техническое предложение, эскизный проект, технический проект, рабочий проект, изготовление, отладка и испытание, ввод в действие.

Руководство разработкой, внедрением, эксплуатацией и модернизацией систем и компонентов САПР в проектной организации должно заниматься специализированное подразделение, включающее группы специалистов соответствующих направлений.

Предпроектные исследования проводятся для выявления готовности конкретной проектной организации к внедрению автоматизированных методов. Основу этой работы составляет системное обследование объекта проектирования и используемых в инженерной практике традиционных методов и приемов проектирования, а также объема технической документации, разрабатываемой в процессе проектирования. Процесс обследования осуществляется главным образом опросом опытных проектировщиков и конструкторов.

В результате обследования определяется необходимость и экономическая эффективность создания автоматизированной системы. При этом учитывается объем проектно-конструкторских работ, их периодичность, общие затраты инженерного труда, возможность создания адекватного математического описания и оптимизационных процедур, необходимость повышения качественных показателей проектируемого изделия, сокращение сроков проектирования.

Существенным фактором при решении вопроса о целесообразности создания САПР является подготовленность соответствующего проектного подразделения к созданию и внедрению САПР. Подготовленность может быть оценена по следующим критериям:

- возможность формализации проектно-конструкторских задач и реализации математических методов их решения;
- наличие требуемых технических средств и необходимость приобретения и установки дополнительных агрегатов;
- подготовленность информационных фондов и технических средств хранения и обработки информации.

Кроме того, важно выявить факторы оценки подготовленности кадров для эксплуатации САПР, к которым можно отнести следующие:

- соответствие внедряемой системы принятой организации проектных работ;
- наличие в проектно-конструкторской организации кадров для эксплуатации и поддержания работоспособности САПР;
- отношение руководства организации к созданию си-темы и уровень организации этих работ;
- психологическая подготовленность коллектива к внедрению САПР.

Техническое задание (ТЗ) является исходным документом для создания САПР и должно содержать наиболее полные исходные данные и требования. Этот документ разрабатывает головной разработчик системы. ТЗ на создание САПР должно содержать следующие основные разделы:

«.Наименование и область применения», где указывают полное наименование системы и краткую характеристику области ее применения;

«Основание для создания», где указывают наименование директивных документов, на основании которых создается САПР;

«Характеристика объектов проектирования», где приводят сведения о назначении, составе, условиях применения объектов проектирования;

«Цель и назначение», где перечисляют цель создания САПР, ее назначение и критерий эффективности ее функционирования;

«Характеристика процесса проектирования», где приводят общее описание процесса проектирования, требования к входным и выходным данным, а также требования по разделению проектных процедур (операции), выполняемых с помощью неавтоматизированного и автоматизированного проектирования;

«Требования к САПР», где перечисляют требования к САПР в целом и к составу ее подсистем, к применению в составе САПР ранее созданных подсистем и компонентов и т. п.;

«Технико-экономические показатели», где оценивают затраты на создание САПР, указывают источники получения экономии и ожидаемую эффективность от применения САПР.

На стадиях технического предложения, эскизного и рабочего проектирования выбираются и обосновываются варианты САПР, разрабатываются окончательные решения. При этом выполняются следующие основные виды работ:

- выявление процесса проектирования (его алгоритм), т. е. принятие основных технических решений;
- разработка структуры САПР и ее взаимосвязи с другими системами (определение состава проектных процедур и операции по подсистемам;
- уточнение состава подсистем и взаимосвязи между ними; разработка схемы функционирования САПР в целом);
- определение состава методов, математических моделей для проектных операций и процедур; состава языков проектирования;

- состава информации (объем, способы ее организации и виды машинных носителей информации);
- состава общего, специализированного общего и специального программного обеспечения;
- формирование состава технических средств (ЭВМ периферийные устройства и другие элементы);
- принятие решений по математическому, информационному, программному и техническому видам обеспечения по САПР в целом и отдельно по подсистемам;
- расчет технико-экономических показателей САПР.

Оформление всей документации, необходимой для создания и функционирования САПР, выполняют на стадии рабочего проектирования.

На стадии изготовления, отладки и испытания производят монтаж, наладку и испытание комплекса технических средств автоматизации проектирования, на тестовых примерах доводят программное обеспечение и подготавливают проектную организацию к вводу в действие САПР.

Ввод в действие системы осуществляют после опытного функционирования и приемочных испытаний у заказчика.

5) Отображение процесса проектирования в программное обеспечение САПР.

Важнейшим вопросом при создании САПР после формализации процесса проектирования является вопрос отображения проектно-конструкторской деятельности инженера в программное обеспечение.

В общем, виде процесс проектирования в САПР можно упрощенно представить схемой, показанной на рис. 1. Эта схема отображает элементарную ячейку проектно-конструкторского процесса, из цепочки, которых состоит реальный автоматизированный процесс. Все системы проектирования, создаваемые с помощью современных средств вычислительной техники, являются автоматизированными. Важнейшую роль в этих системах играет че-

ловек-инженер, разрабатывающий проект новых технических средств. Человек в САПР решает все неформализованные проектные задачи и задачи планирования работ. Современная САПР является инструментом высококвалифицированного инженера-проектировщика, поэтому тесное взаимодействие человека и ЭВМ в процессе проектирования — один из важнейших принципов построения и эксплуатации САПР.

Основным блоком в схеме процесса автоматизированного проектирования (рис. 1) является блок проектных решений. В зависимости от полноты формализации наших знаний в конкретной предметной области проектное решение может быть выполнено автоматически или в интерактивном режиме. На основе входных данных и ограничений (независимые параметры проектирования) блок изменяет варьируемые параметры (факторы решения) до получения приемлемых проектных решений (зависимых переменных).

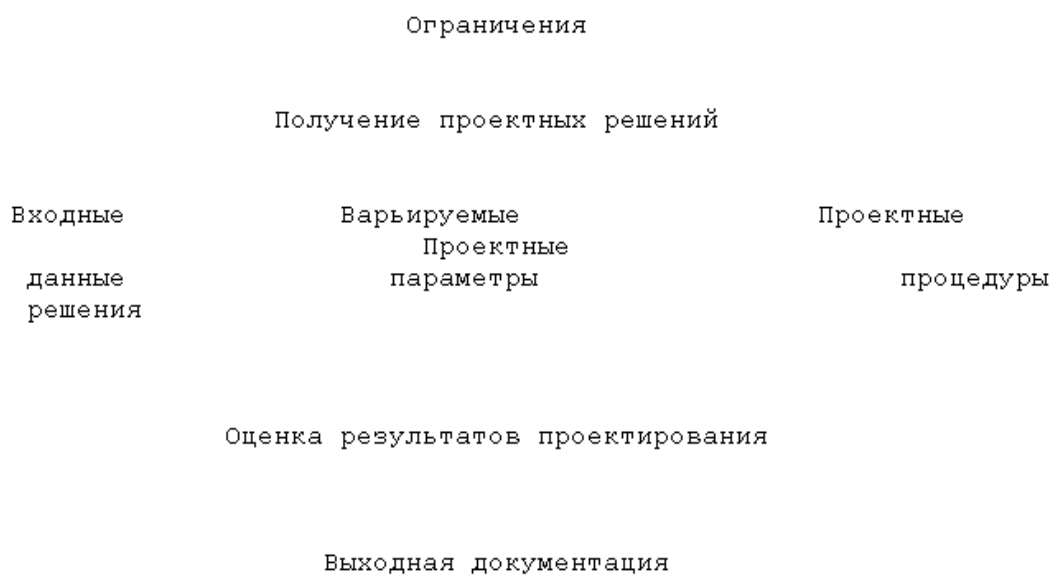


Рис. 1. Схема процесса автоматизированного проектирования

Результаты проектирования должны быть представлены в виде, удобном для восприятия человеком, и содержать информацию, на основе которой инженер мог бы вынести суждение о результатах проектирования.

Если проектное решение утверждается, то оформляется требуемая выходная документация; если необходима корректировка проекта, инженер, уточняя варьируемые параметры, в интерактивном режиме добивается нужных результатов; когда же проектно-конструкторский процесс не приводит к намеченной цели, необходимо уточнить входные данные и ограничения.

Рассмотрение даже такой упрощенной схемы процесса проектирования позволяет уточнить разделение функции между инженером и ЭВМ в САПР.

Получение вариантов проектных решений и их представление в виде, удобном для восприятия человеком, может быть возложено на ЭВМ в той мере, в какой это позволит сделать математическое обеспечение проектных процедур. Но даже при автоматическом получении вариантов проектных решений за инженером остаются важнейшие функции — ввод исходных данных для проектирования, окончательная оценка и утверждение проектных решении. В интерактивном же режиме проектирования инженер непосредственно участвует в ходе решения задач, воздействуя на выбор факторов решения и уточняя независимые переменные. Получение выходной документации в соответствии с существующими требованиями является операцией рутинной и должно выполняться
автоматически.

На основании изложенного модель программного обеспечения автоматизированной проектной процедуры можно представить схемой, показанной на рис. 2.

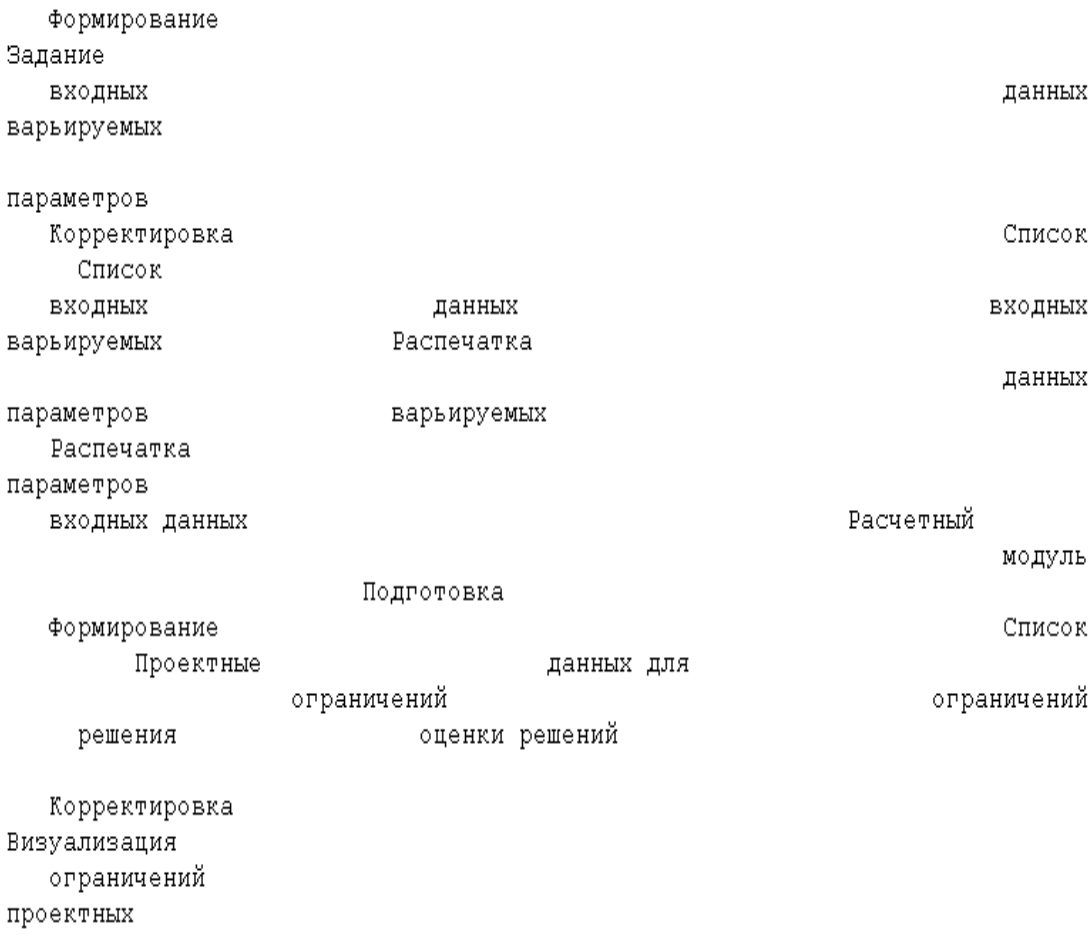


Рис. 2. Модель программного обеспечения проектной процедуры в САПР

Обобщенная модель программного обеспечения проектной процедуры в САПР имеет ряд составляющих и списки данных. В общем, виде каждая составляющая должна реализоваться своим программным модулем.

Назначение **модуля формирования входных данных** состоит в создании списка этих данных для проектирования и его контроля при вводе в систему. Структура и формат списка входных данных зависят от содержания проектной процедуры (расчетного модуля). Необходимо предусмотреть существование нескольких версий списка входных данных, которые с заданными именами хранятся на участках магнитного диска. Структура списка данных определяется разработчиком САПР, а формируется он либо в диалоговом режиме пользователем, либо генерируется автоматически предыдущими проектными процедурами.

Программный **модуль корректировки входных данных** предусматривает редактирование (удаление, вставку и т. п.) списка, потребность в котором возникает из-за ошибок пользователя при вводе данных, обнаруживаемых при контроле, а также при необходимости их уточнения в результате анализа и оценки проектных решений.

Для обеспечения тщательного контроля в САПР должны быть предусмотрены **программные средства для визуализации списков данных**. В общем случае необходимо иметь возможность получения нескольких видов распечатки списка данных: двоичный, десятичный, символьный, табличный и по записям. Для реализации различных требований пользователя распечатка может выводиться на экран дисплея или на АЦПУ. Все эти операции выполняет модуль распечатки входных данных.

Программные модули формирования, корректировки и распечатки ограничении на процесс проектирования функционируют аналогично описанным. Структура и формат ограничений зависят от проектного модуля, но они существенно меньше подвержены изменениям, чем структура и формат исходных данных. Однако необходимо предусматривать существование нескольких версий этих списков (например, общих требований к техническим средствам со стороны различных заказчиков).

Создание и контроль списка варьируемых параметров осуществляются программными модулями их задания и распечатки.

Расчетный модуль программного обеспечения процесса проектирования предназначен для автоматического выполнения ЭВМ всех тех операций проектной процедуры, которые удалось полностью формализовать.

Получаемые варианты проектных решений обрабатываются программным модулем подготовки данных для оценки решений и передаются модулю визуализации. Анализируя результаты проектно-конструкторского процесса, инженер должен иметь возможность просмотра выходных данных на АЦПУ, дисплее и графопостроителе, например, в виде таблиц, схем и чертежей.

Допустимо существование нескольких версий проектных решений, которые хранятся на магнитном диске и могут быть представлены в требуемом виде с помощью программного модуля документирования проектных решений.

Связь между различными программными модулями проектной процедуры и взаимодействие данной проектной процедуры с другими происходит через общую память.

Это позволяет осуществлять интерактивный автоматизированный процесс проектирования с сохранением множества различных версий, как входных данных, так и проектных решений. Для выполнения требования принципа рациональной связи САПР с окружающей средой при проектировании программного обеспечения следует стремиться к тому, чтобы список входных данных был результатом предыдущих проектных процедур или модулей. Это достигается при разработке информационного обеспечения САПР.

Тема 2. Модель предметной области (2 часа)

Лекция № 2

Данные, используемые в программном комплексе. Описание модели. Типы данных. Структура данных. Функциональные связи. Способы обработки данных. Анализ модели предметной области. Группы данных. Входные данные. Выходные данные.

Каждая информационная система в зависимости от ее назначения имеет дело с частью реального мира, которую принято называть предметной областью (ПО) системы. ПО может относится к любому типу организаций: банк, университет, завод, магазин и т.д.

Предметная область информационной системы - это совокупность реальных объектов (сущностей), которые представляют интерес для пользователей.

Объект (сущность) - предмет, процесс или явление, о котором собирается информация, необходимая для решения задачи. Объектом может быть человек, предмет, событие.

Каждый объект характеризуется рядом основных свойств - атрибутов. **Атрибутом** называется поименованная характеристика объекта. Атрибут показывает, какая информация должна быть собрана об объекте.

Например, объект - клиент банка.

Атрибуты - номер счета, адрес, сумма вклада.

Технология анализа предметной области

Первым этапом проектирования БД любого типа является анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание проектируемой БД. На основе проведенного анализа структурируется предметная область. Анализ предметной области не зависит от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы:

1) анализ концептуальных требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение концептуальной модели предметной области и проектирование концептуальной схемы БД.

Анализ концептуальных требований и информационных потребностей

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации.

Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Рассмотрим примерный состав вопросника при анализе различных предметных областей.

Пример 1. Предлагается разработать БД для учета студентов вуза.

Анализ предметной области:

1. Сколько студентов учится в вузе?
2. Сколько факультетов и отделений в вузе?
3. Как распределены студенты по факультетам отделений и курсам?
4. Сколько дисциплин читается на каждом курсе по каждой специальности?
5. Как часто обновляется информация в БД?
6. Сколько преподавателей в вузе?
7. Сколько иногородних студентов живет в общежитии, на частных квартирах?
8. Сколько лекционных аудиторий и аудиторий для проведения практических занятий, лабораторий?
9. Какая преемственность существует между читаемыми курсами?
10. Как информация, представленная в п.п. 1-9, используется в настоящее время (расписание занятий, экзаменов, зачетов и т.д.) и как ее собираются использовать?
11. Сколько раз в день, сколько человек и кто пользуются БД?

Пример 2. Разработать требования к локальной БД "Аэропорт".

Вопрос 1. Для каких типов задач (приложений) проектируется БД?

Ответ. Для трех типов задач:

Задача 1. Информация об обслуживающем персонале.

Задача 2. Информация о полетных средствах.

Задача 3. Информация о графике движения самолетов.

Вопрос 2. Какими информационными объектами характеризуются эти задачи?

Ответ. Задача 1 характеризуется тремя информационными объектами: летный состав, диспетчеры, технический персонал.

Задача 2 характеризуется двумя информационными объектами: самолет, взлетное поле.

Задача 3 характеризуется одним информационным объектом - рейсы.

Вопрос 3. Каким текущим запросам должны удовлетворять данные информационные объекты? Ответ.

1. ФИО, звание, должность членов экипажа самолета.

2. Списочный состав диспетчеров.

3. Состав смены технического персонала.

4. Тип самолета, который может обслуживать тот или иной пилот.

5. Номер самолета, который обслуживает данный пилот, данная смена диспетчеров и технического персонала.

6. Номер личного дела сотрудника аэропорта.

7. Номер смены диспетчеров и технического персонала, обслуживающего аэропорт в заданном интервале времени.

8. Готовность самолета с таким-то номером к полету.

9. Количество часов налета такого-то самолета.

10. Готовность данной взлетной полосы в настоящее время.

11. Длина данной полосы.

12. Номер (номера) рейса до данного пункта назначения.

13. Какие промежуточные посадки совершает рейс №... ?

14. Время вылета и расчетное время прибытия рейса №... .

15. Время и место регистрации рейса №... .

16. Время посадки на рейс №... .

17. До какого времени задерживается рейс №... ?

18. Какие типы самолетов обслуживают рейс №... ?

19. Какой номер самолета обслуживает рейс №... ?

Вопрос 4. Каким перспективным запросам должны удовлетворять информационные объекты в БД "Аэропорт"?

1. С какого года используется самолет с №... в аэропорту, тип самолета?

2. Какое количество часов полета у члена экипажа, ФИО?

3. Расчетное время отпуска члена экипажа, диспетчера, технического работника.

Пример 3. Разработать БД "Видеомагнитофоны".

Вопрос 1. На кого рассчитана эта БД? Ответ. На покупателя видеосистем.

Вопрос 2. Что интересует покупателя?

Ответ. Покупателя интересуют технические характеристики системы, ее цена, фирма-изготовитель, технические характеристики и цена видеокассет, фирма-изготовитель кассет.

Далее проектировщик выбирает по технической документации параметры видеосистем, разрабатывает перечень запросов и уточняет его с будущим пользователем БД. Однако БД пока нет.

Есть только предложения проектировщика и одобрение будущего пользователя. Пользователю кажется, что все проблемы позади, а проектировщика еще ждет очень большая работа.

Выявленные запросы представлены следующим перечнем:

1. Выдать данные на определенную модель системы.
2. Какова цена той или иной системы?
3. Выдать системы определенной страны-изготовителя, цены которых не превышают заданную.
4. Выдать последние модели определенной фирмы.
5. Выдать модели определенной фирмы, габаритные размеры которых не превышают заданные.
6. Выдать № моделей определенной страны-изготовителя, цены которых не превышают заданные.
7. Какова цена последних моделей определенной фирмы?
8. Выдать № модели и фирму-изготовителя самого дешевого видеоплейера, выпущенного в определенном году.
9. Выдать характеристики видеомагнитофонов, выпущенных в определенном году.
10. Выдать наименование модели и фирму-изготовителя видеокассет с наибольшим временем проигрывания.
11. Выдать данные на самую дешевую видеокассету.
12. Выдать данные на самую дорогую видеокассету.
13. Сколько стоит видеокассета определенного наименования и определенной фирмы?

14. Выдать данные на видеокассету, имеющую определенное время проигрывания.

1) Технология анализа предметной области.

Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит:

1 - в выборе информационных объектов,

2 - задании необходимых свойств для каждого объекта,

3 - выявлении связей между объектами,

4 - определении ограничений,

5 - накладываемых на информационные объекты,

6 - типы связей между ними,

7 - характеристики информационных объектов.

Проанализируем предметную область на примере БД "Видеомагнитофоны".

При выборе информационных объектов постараемся ответить на ряд вопросов:

На какие классы можно разбить данные, подлежащие хранению в БД?

Какое имя можно присвоить каждому классу данных?

Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?

Какие имена можно присвоить выбранным наборам характеристик?

Пример. Продолжим создание БД "Видеомагнитофоны", рассчитанной на пользователей, которые хотят приобрести данный вид техники.

После беседы с различными пользователями и просмотра каталогов было выяснено, что интерес представляют три информационных объекта: видеомагнитофон, видеоплейер, видеокассета. Рассмотрим наиболее существенные характеристики каждого информационного объекта.

Объект - ВИДЕОМАГНИТОФОН.

Атрибуты - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число кассетных гнезд, ресурс непрерывной работы, система автопоиска, напряжение в сети, наличие таймера, число программ, габаритные размеры, масса, цена в долларах, год выпуска.

Объект - ВИДЕОПЛЕЙЕР.

Атрибуты - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число воспроизводящих головок, ресурс непрерывной работы, напряжение в сети, наличие таймера, габаритные размеры, масса, цена в долларах, год выпуска.

Объект - ВИДЕОКАССЕТА.

Атрибуты - наименование, страна-изготовитель, фирма-изготовитель, тип кассеты, время проигрывания, цена в долларах.

Далее выделим связи между информационными объектами. В ходе этого процесса постараемся ответить на следующие вопросы:

Какие типы связей между информационными объектами?

Какое имя можно присвоить каждому типу связей?

Каковы возможные типы связей, которые могут быть использованы впоследствии?

Имеют ли смысл какие-нибудь комбинации типов связей?

Попытаемся задать ограничения на объекты и их характеристики.

Под ограничением целостности обычно понимают логические ограничения, накладываемые на данные. Ограничение целостности - это такое свойство, которое мы задаем для некоторого информационного объекта или его характеристики и которое должно сохраняться для каждого их состояния.

Введем следующие ограничения:

1. Значение атрибута "число кассетных гнезд" изменяется от 1 до 2.
2. Значение атрибута "ресурс непрерывной работы" изменяется от 4 до 24.
3. Значение атрибута "напряжение в сети" изменяется от 110 до 240 В.
4. Значение атрибута "число программ" изменяется от 1 до 20 и т.д.

Типы связей. Все информационные объекты предметной области связаны между собой.

Соответствия, отношения, возникающие между объектами предметной области называются **связями.** Различаются связи нескольких типов, для которых введены следующие обозначения:

а) один к одному (1:1);

б) один ко многим (1:М);

в) многие ко многим (М:М).

Рассмотрим эти типы связей на примере.

Пример. Дана совокупность информационных объектов, отражающих учебный процесс в вузе:

СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения. Группа) СЕССИЯ (Номер. Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат) СТИПЕНДИЯ (Результат. Процент) ПРЕПОДАВАТЕЛЬ (Код преподавателя. Фамилия, Имя, Отчество)

Связь один к одному (1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует не более одного экземпляра информационного объекта В и наоборот.

Рис. 2.1. Графическое изображение реального отношения 1:1

Примером связи 1:1 может служить связь между информационными объектами СТУДЕНТ и СЕССИЯ:

СТУДЕНТ <—> СЕССИЯ

Каждый студент имеет определенный набор экзаменационных оценок в сессию.

При связи **один ко многим** (1 :М) одному экземпляру информационного объекта А соответствует 0, 1 или более экземпляров объекта В, но каждый экземпляр объекта В связан не более чем с 1 экземпляром объекта А.

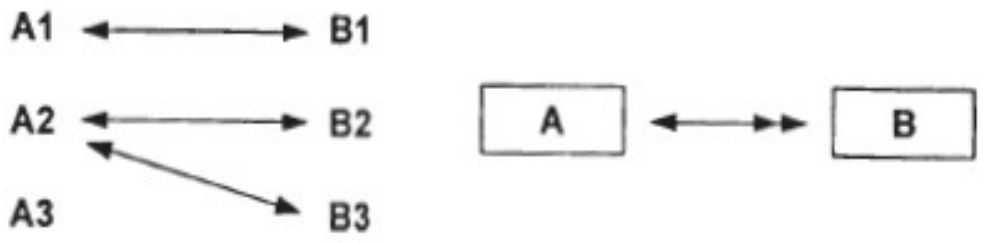


Рис. 2.2. Графическое изображение реального отношения 1:М

Примером связи 1:М служит связь между информационными объектами СТИПЕНДИЯ и СЕССИЯ:

СТИПЕНДИЯ <—» СЕССИЯ

Установленный размер стипендии по результатам сдачи сессии может повторяться многократно для различных студентов.

Связь **многие ко многим** (М:М) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует 0, 1 или более экземпляров объекта В и наоборот.

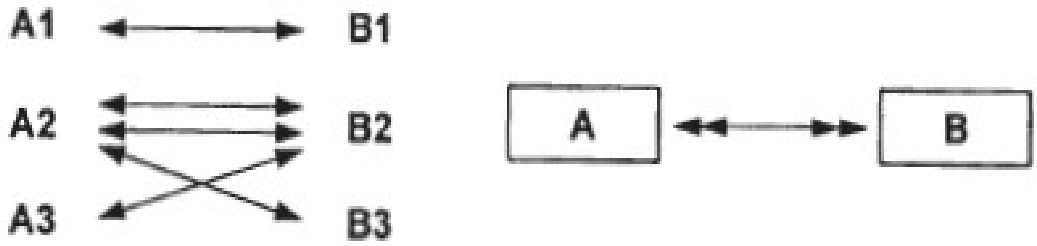


Рис. 2.3. Графическое изображение реального отношения М:М

Примером данного отношения служит связь между информационными объектами СТУДЕНТ и ПРЕПОДАВАТЕЛЬ:

СТУДЕНТ «— » ПРЕПОДАВАТЕЛЬ

Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов.

Построение концептуальной модели предметной области

Заключительная фаза анализа предметной области состоит в проектировании ее информационной структуры или концептуальной модели.

Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных.

Концептуальная модель применяется:

1. для структурирования предметной области с учетом информационных интересов пользователей системы,

2. она дает возможность систематизировать информационное содержание предметной области, позволяет как бы "подняться вверх" над ПО и увидеть ее отдельные элементы. При этом, уровень детализации зависит от выбранной модели.

Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений.

Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «сущность - связь». Основными конструкциями данной модели являются сущности и связи.

Под сущностью понимают основное содержание объекта ПО, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление.

Экземпляр **сущности** - конкретный объект.

Например:

сущность (объект) - служащий

экземпляр сущности - Иванов А. В.;

сущность (объект) - институт экземпляр сущности - СГУ.

Сущность принято определять атрибутами - поименованными характеристиками.

Например:

сущность - служащий

атрибуты: ФИО, год рождения, адрес, образование и т.д.

Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначений атрибута - идентифицировать сущность.

Связь определяет отношения между сущностями. Типы связей: один к одному, один ко многим, многие ко многим.

При построении модели «сущность - связь» используют графические диаграммы. При этом обозначают:

сущности - прямоугольниками,

атрибуты - овалами,

связи - ромбами.

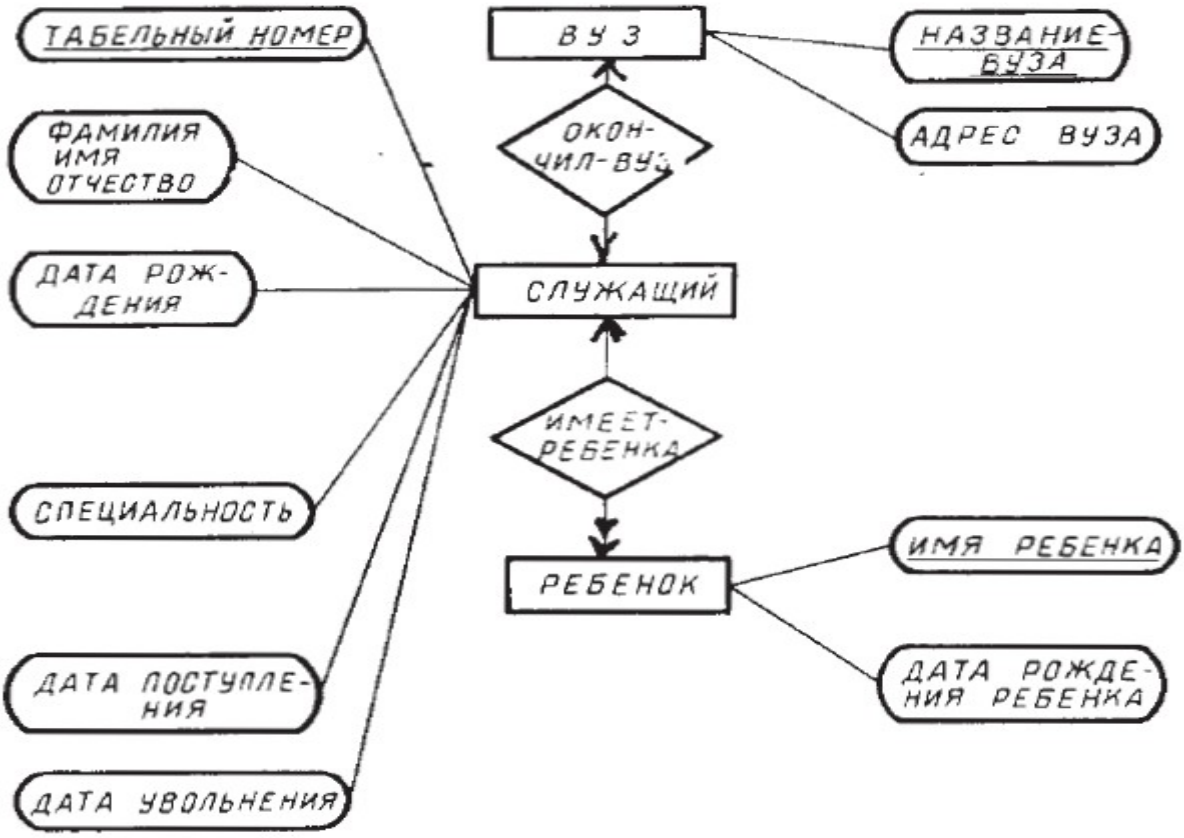


Рис. 2.4. Пример модели "сущность - связь"

На практике приходится строить несколько вариантов моделей, из которых выбирается одна, наиболее полно отображающая предметную область.

Тема 3. Внешнее управление программным комплексом (2 часа)

Лекция №3

1) Цель применения программного комплекса документооборота.

Программный комплекс электронного документооборота — это автоматизированная информационная система, предназначенная для реализации процесса удаленного обмена большими массивами форматированной информации. В наше время, в связи с бурным развитием Интернет-технологий, программные комплексы электронного

документооборота находят широчайшее применение во многих сферах человеческой деятельности, и в первую очередь – в процессе электронного взаимодействия государственных структур и хозяйствующих экономических субъектов.

Организация такого взаимодействия является одной из важнейших задач и приоритетов современного информационного общества. Постоянно растут объемы информации, обрабатываемой в информационных системах органов исполнительной власти для повышения качества и эффективности управления государством, и, соответственно, увеличиваются объемы документооборота между бизнес-структурами и государственными органами, уполномоченными законодательством на прием и обработку различного рода данных.

Именно передача данных в электронном виде по телекоммуникационным каналам связи является единственным естественным способом взаимодействия для современных информационных систем.

На практике создаются и внедряются программные комплексы, ориентированные на решение задач электронного документооборота,

• благодаря применению таких систем становится возможным повышение эффективности государственного и межведомственного управления, за счет ускорения поступления информации в автоматизированные информационные системы государственных органов;

• одновременно требуется внедрение целого ряда систем

документооборота (например, в России потребителями информации выступают налоговая служба, фонды социального и медицинского страхования, служба государственной статистики, таможенная служба, Пенсионный фонд, служба по тарифам, служба финансового мониторинга, служба экологического мониторинга, региональные финансово-бюджетные службы – все со своими требованиями к процедурам обработки информации), и, следовательно, имеется потребность в обобщении принципов их проектирования, и в построении универсальной модели обработки данных, на базе которой была бы возможна интеграция этих систем;

• предлагаются новые, эффективные алгоритмы обработки информации в таких комплексах, связанные с обеспечением возможности работы с информационными массивами больших объемов, хранящими все исторические состояния данных;

• предлагается и обосновывается компонентная модель обработки и хранения информации в узле системы, обеспечивающая автоматическую проверку целостности данных;

Цели разработки программных комплекса документооборота: исследуя процессы обработки, хранения и контроля целостности данных, циркулирующих в среде автоматизированного программного комплекса электронного документооборота

• установить общие закономерности, которым подчиняются данные процессы;

• создать на основе этих закономерностей математическую модель информационных потоков в программных комплексах электронного документооборота;

• разработать алгоритмы работы с данными, обеспечивающие эффективное хранение и обработку информации в узлах системы документооборота;

• предложить методологию применения данной математической модели при проектировании, разработке и внедрении прикладных программных комплексов.

2) Формирование задания на проектирование программных комплексов и информационных систем.

Согласно международному стандарту 180/1ЕС СО 15288 каждая автоматизированная информационная система работающая на основе рограммного комплекса, в том числе государственного значения, проходит жизненный цикл, состоящий из следующих взаимосвязанных этапов:

a) разработка концепции системы;

b) проектирование системы;

c) разработка системы;

d) эксплуатация системы;

e) сопровождение системы;

f) списание системы.

9. Для каждой конкретной системы, в зависимости от ее сложности, этапы, указанные выше, могут объединяться либо разделяться, но их порядок и требования к документации должны строго соответствовать вышеназванному стандарту.

10. В процессе реализации этапа разработки концепции системы определяются требования, которым должна соответствовать система в целом, ее назначение и функции, а также ставятся задачи, которые должны быть решены при эксплуатации системы. Основной целью разработки концепции является определение информационных объектов системы, принципов их идентификации, перечня данных, хранящихся в системе, а также принципов интеграции с другими информационными системами. Структура и содержание концепции определяются соответствующими стандартами.

Концепции автоматизированных информационных систем государственного значения подлежат утверждению Правительством.

11. Этап проектирования системы включает в себя разработку требования к каждому элементу системы, включая требования к программному обеспечению, техническому комплексу, телекоммуникациям, организационной и информационной структуре, разрабатываемым формулярам документов, а также

к необходимой эксплуатационной документации. Эти требования отражаются в Техническом задании, утверждаемом Заказчиком. Структура и содержание Технического задания определяется соответствующими стандартами.

12. Разработка системы и ее элементов должна осуществляться в строгом соответствии с утвержденным техническим заданием.

13. Этап разработки системы включает следующие стадии:

a) разработка и/или поставка программного обеспечения;

b) поставка технических средств и расходных материалов;

c) тестирование системы;

d) внедрение пилот-проекта системы (по соглашению сторон);

e) передача системы в промышленную эксплуатацию.

14. Этап разработки системы завершается ее передачей Заказчику в промышленную эксплуатацию на основании соответствующих документов.

3) Виды внешнего управления программным комплексом.

Чтобы обеспечить взаимодействие пакета с пользователем и управляющих модулей пакета с информационной базой и обрабатывающими модулями, в состав пакета включаются обслуживающие модули.

Под способом применения ППП будем понимать организацию взаимодействия пользователя с пакетом при решении задач. Выбор способа применения ППП зависит от многих факторов, из которых наиболее существенными являются возможности ОС и выбранного языка программирования, объемы обрабатываемых данных, продолжительность решения задачи, частота использования ППП, особенности квалификации пользователей пакета и требования к оперативности решения задач (допустимому времени ожидания результатов расчетов).

Способы применения существующих в настоящее время ППП весьма разнообразны, однако можно выделить некоторые типовые режимы, определяемые построением самого пакета и особенностями используемой ЭВМ и ОС.

Простейший режим с точки зрения построения ППП сводится к использованию отдельных программ пакета как подпрограмм некоторой главной программы, составляемой пользователем на каком-либо языке программирования, например Си. В этом случае ППП состоит только из обрабатывающих модулей и может рассматриваться как расширение библиотеки подпрограмм используемого языка программирования.

Следующий по сложности реализации режим предполагает, что вся управляющая информация для конкретного выполнения пакета передается в виде законченной программы на входном языке при запуске пакета. Дальнейшая рабо-

та пакета проходит без участия пользователя. Такой режим по аналогии с соответствующим режимом ОС часто называют *пакетным.* Пакетный режим удобен, когда требуется решать много однотипных задач с использованием одной и той же программы на входном языке, когда время, затрачиваемое на решение каждой задачи, достаточно велико, когда программа на входном языке сложна и имеет значительный объем.

Большинство ППП, применяемых на персональных ЭВМ, ориентировано на *диалоговое взаимодействие* с пользователем в ходе решения задач.

Простейший диалоговый режим состоит в том, что пользователь инициирует выполнение пакета, вводит задание в форме программы на входном языке и на этом заканчивает управление выполнением пакета. Фактически этот режим отличается от пакетного только возможностью исправления ошибок в ПВЯ, повторного запуска пакета при неудачах.

Более сложный вариант диалогового режима, называемый также *режимом сопровождения,* предусматривает возможность динамического управления выполнением пакета. Управляющая информация вводится по частям и формируется пользователем в процессе работы с пакетом на основе анализа промежуточных результатов. Такая работа в большинстве случаев более естественна для пользователя, в частности, при использовании пакетов редактирования текстов, при работе с электронными таблицами, при решении многих расчетных задач.

Тема 4. Проектирование модулей программного комплекса (2 часа)
Лекция №4

По современным взглядам, ППП — это совокупность совместимых программ для решения определенного класса задач. ППП всегда ориентируется на пользователей определенной квалификации как в программировании, так и в той области, к которой относятся задачи, решаемые с применением этого ППП.
Совместимость программ, составляющих ППП, означает возможность их взаимного использования, общность структуры управляющих данных и используемых информационных массивов. Кроме того, ППП следует рассматривать как самостоятельное программное изделие, как особый вид прикладного ПО.
Исходя из определения можно выделить следующие общие свойства ППП.
Пакет состоит из нескольких программных единиц.
Пакет предназначен для решения определенного класса задач.
В пределах своего класса пакет обладает определённой универсальностью, т.е. позволяет решать все или почти все задачи этого класса.
В пакете предусмотрены средства управления, позволяющие выбирать конкретные возможности из числа предусмотренных в пакете. Пакет допускает настройку на конкретные условия применения.
Пакет разработан с учетом возможности его использования за пределами той организации, в которой он создан, и удовлетворяет общим требованиям к программному изделию.

Документация и способы применения пакета ориентированы на пользователя, имеющего определенный уровень квалификации в той области знаний, к которой относятся решаемые пакетом задачи.

Поскольку ППП предназначен для решения определенного класса задач, можно говорить о функциональном назначении пакета.

В зависимости от функционального назначения выделяют ППП, расширяющие возможности ОС, например, для построения многопользовательских систем, работы с удаленными абонентами, реализации специальной организации файлов, упрощения работы с ОС и т.п. Примерами таких пакетов служат пакет СРВ, реализующий режим разделения времени в ОС ЕС ЭВМ, пакет Norton Commander для облегчения работы с операционной системой MS DOS на персональных ЭВМ.

Среди пакетов, предназначенных для решения прикладных задач пользователей, иногда выделяют методо-ориентированные и проблемно-ориентированные пакеты. Методо-ориентированный пакет предназначен для решения задачи пользователя одним из нескольких методов, предусмотренных в пакете, причем метод либо назначается пользователем, либо выбирается автоматически на основе анализа входных данных. Пример такого пакета — пакет математического программирования, позволяющий решить задачу выпуклого программирования либо методом штрафных функций, либо одним из вариантов методов возможных направлений.

Проблемно-ориентированные пакеты предназначены для решения групп (последовательностей) задач, использующих общие Данные. Это наиболее многочисленная группа пакетов. Проблемная ориентация может выражаться в общем характере операций» выполняемых пакетом. Типичные примеры таких пакетов — текстовые редакторы, табличные процессоры, пакет линейного программирования.

Проблемная ориентация может быть представлена и общей прикладной проблемой, решение которой распадается на отдельные задачи, для каждой из которых в пакете предусмотрен свой алгоритм. Типичные примеры — пакет для проведения расчетов межотраслевых балансов, пакеты, используемые в различных системах автоматизации проектирования.

В последние годы получили распространение так называемые интегрированные пакеты, представляющие собой пакеты широкого назначения, объединяющие текстовый редактор, процессор электронных таблиц, систему управления базой данных, пакет графического отображения данных (деловую графику) и средства обмена данными с удаленными абонентами. Наиболее популярны у пользователей интегрированные пакеты Мастер, Symphong, Framework.

На рис. 5.1 показан вариант классификации пакетов по их функциональному назначению.

При определении пакета программ было отмечено, что пакет состоит из нескольких программных единиц. Такие программные единицы обычно называют программными модулями. Пакет предназначен для решения задач определенного класса. Этот класс задач обычно называют предметной областью пакета. Применительно к ППП для решения расчетных задач предметная область

определяет некоторую структуру данных, т.е. организацию входных, промежуточных и выходных данных. Говорят, что пакет использует информационную базу, соответствующую своей предметной области.

Для реализации выбранных пользователем конкретных действий пакет должен воспринимать от пользователя управляющую информацию. Эта управляющая информация представляется на формальном языке — входном языке пакета. Описание конкретного задания пользователя на входном языке пакета называют программой на входном языке (ПВЯ).

Решение каждой задачи в пакете сводится к выполнению соответствующего алгоритма. Программные модули пакета, реализующие алгоритмы решения задач, предусмотренных в пакете, будем называть обрабатывающими модулями. Обрабатывающие модули выполняют преобразование данных, составляющих информационную базу пакета.

Для того чтобы преобразовать задание пользователя в последовательность вызовов обрабатывающих модулей, в пакет должны входить управляющие модули.

Чтобы обеспечить взаимодействие пакета с пользователем и управляющих модулей пакета с информационной базой и обрабатывающими модулями, в состав пакета включаются обслуживающие модули.

Таким образом, ППП можно рассматривать как объединение входного языка, информационной базы, управляющих, обслуживающих и обрабатывающих программных модулей. Совокупность обрабатывающих модулей часто называют функциональным наполнением пакета. Управляющие и обслуживающие модули называются системной частью пакета, или системным наполнением пакета.

Взаимодействие составных частей пакета схематически показано на рис. 5.2. Средствами операционной системы запускается головной управляющий модуль пакета (ведущий модуль). Затем организуются прием задания пользователя, представляемого в форме программ на входном языке (ПВЯ), и выполнение этого задания путем вызова в нужной последовательности обрабатывающих и обслуживающих модулей.

Под способом применения ППП будем понимать организацию взаимодействия пользователя с пакетом при решении задач. Выбор способа применения ППП зависит от многих факторов, из которых наиболее существенными являются возможности ОС и выбранного языка программирования, объемы обрабатываемых данных, продолжительность решения задачи, частота использования ППП, особенности квалификации пользователей пакета и требования к оперативности решения задач (допустимому времени ожидания результатов расчетов).

Способы применения существующих в настоящее время ППП весьма разнообразны, однако можно выделить некоторые типовые режимы, определяемые построением самого пакета и особенностями используемой ЭВМ и ОС.

Простейший режим с точки зрения построения ППП сводится к использованию отдельных программ пакета как подпрограмм некоторой главной программы, составляемой пользователем на каком-либо языке программирования, напри-

мер ПЛ/1 или Си. В этом случае ППП состоит только из обрабатывающих модулей и может рассматриваться как расширение библиотеки подпрограмм используемого языка программирования.
Следующий по сложности реализации режим предполагает, что вся управляющая информация для конкретного выполнения пакета передается в виде законченной программы на входном языке при запуске пакета. Дальнейшая работа пакета проходит без участия пользователя. Такой режим по аналогии с соответствующим режимом ОС часто называют пакетным. Пакетный режим удобен, когда требуется решать много однотипных задач с использованием одной и той же программы на входном языке, когда время, затрачиваемое на решение каждой задачи, достаточно велико, когда программа на входном языке сложна и имеет значительный объем.
Большинство ППП, применяемых на персональных ЭВМ, ориентировано на диалоговое взаимодействие с пользователем в ходе решения задач.
Простейший диалоговый режим (вариант диалогового взаимодействия) состоит в том, что пользователь инициирует выполнение пакета, вводит задание в форме программы на входном языке и на этом заканчивает управление выполнением пакета. Фактически этот режим отличается от пакетного только возможностью исправления ошибок в ПВЯ, повторного запуска пакета при неудачах.
Более сложный вариант диалогового режима, называемый также режимом сопровождения, предусматривает возможность динамического управления выполнением пакета. Управляющая информация вводится по частям и формируется пользователем в процессе работы с пакетом на основе анализа промежуточных результатов. Такая работа в большинстве случаев более естественна для пользователя, в частности, при использовании пакетов редактирования текстов, при работе с электронными таблицами, при решении многих расчетных задач.

Тема 5. Особенности организации управления программным комплексом с входным языком типа меню (2 часа)

Лекция № 5

Конечный автомат можно охарактеризовать как устройство, имеющее входной и выходной каналы и находящееся в каждый из дискретных моментов времени, называемых тактовыми моментами, в одном из конечного числа состояний.
Макроподход и микроподход.
1. Макроподход изучает внешнее поведение устройства, как оно перерабатывает входную информацию в выходную, отвлекаясь от внутреннего строения.
Вводится понятие абстрактного конечного автомата. Изучает автоматы как преобразователи и акцепторы:
Преобразователи отображают входные сигналы в множество последовательностей выходных сигналов.

Акцепторы какие множества конечных последовательностей входных сигналов можно отличать друг от друга с помощью выходных сигналов автоматов.
2. Микроподход учитывается структура устройства, функционирование и связь между собой его частей. Вводится понятие структурного конечного автомата. Структурный конечный автомат задается конечным множеством абстрактных автоматов, конечной схемой их соединения и указанием влияния частей схемы друг на друга. Изучает задачи анализа и синтеза аппаратных схем, т.е. описание свойств отображений по автоматным схемам и построение аппаратных схем по отображениям соответственно; задачи о выразимости и сложности представления одних аппаратов через другие с помощью различных операций и др.
Понятия конечного абстрактного и конечного структурного автоматов можно считать составляющими понятие конечного автомата.
Теория автоматов имеет широкий круг применений как внутри математики в различных ее областях (в алгебре, математической логике и др.), так и в решении практических задач (в анализе и синтезе ЭВМ, в распознавании образов и т.п.). Понятие автомата может служить модельным объектом в самых разнообразных задачах, благодаря чему возможно применение теории автоматов в различных научных и прикладных исследованиях.
Характерными особенностями дискретных моделей являются:
1. Дискретность "времени".
2. Для задания текущего состояния модели используются не количественные, а качественные характеристики. При этом во многих случаях оказывается возможным считать, что состояния модели суть элементы некоторого конечного множества (быть может, весьма большого).
Изменения состояния дискретной модели от момента к моменту могут иметь как детерминированный так и недетерминированный характеры. Последнее означает, что знание состояния модели в момент t и внешних воздействий на нее в тот же момент еще не позволяет однозначно определить состояния модели к моменту (t+1), а лишь сужает некоторым образом класс ее возможных состояний.
Различают конечные автоматы без памяти, с конечной памятью, с конечным запоминанием. Ограниченность объема памяти конечного автомата накладывает ряд существенных ограничений на вычислительные возможности этого устройства. В ситуациях, представляющих практический интерес, когда требуется реализовать операторы, определенные лишь на конечных множествах, роль этих ограничений невелика, однако они сказываются при теоретическом описании эффективных вычислительных процедур. Поэтому в теории алгоритмов рассматривают конечные автоматы, снабженные бесконечной "внешней памятью" так называемые машины Тьюринга.
Обобщения понятия конечного автомата:
1. Допускаются бесконечные множества состояний, входных и выходных сигналов автомата.
2. Допускается недетерминированность при изменении состояния и определении выходного сигнала.

3. Видоизменяется само понятие функционирования автомата. Например, входная информация предполагается поступающей не в виде слов, а в виде "деревьев" (автоматы над термами); либо в определении функционирования автомата на учитываются поступающие по некоторым входным каналам воздействия (автоматы с переменной структурой) и т.п.
Наиболее часто в литературе рассматриваются автоматы Мили и автоматы Мура. Удачные практические задачи, приводящие к понятию таких автоматов приведены в книге В.Брауэра.
Частичные автоматы.
На практике, особенно часто в тех случаях, когда входы для автомата поступают от некоторой технической системы (от другого автомата), определенные комбинации входов вообще не встречаются. Часто также и не все выходы автомата представляют интерес. Поэтому полезно рассматривать модели автоматов, в которых допускается, что определенные переходы из состояния в состояние или определенные выходы остаются незаданными.
Автоматы Рабина Скотта.
При многих исследованиях функционирования машин, процессов и алгоритмов бывает важен вопрос о том, при каких обстоятельствах из некоторого определенного (начального) состояния может быть достигнуто определенное (конечное или финальное) состояние исследуемого объекта.
Чаще бывает проще и удобнее описывать метод решения в виде недерминированного алгоритма он может быть далее обращен в обычный детерминированный алгоритм, либо непосредственно реализован на вычислительном устройстве, способном работать в параллельном режиме. Часто при этом возникают тупиковые ситуации в параллельных процессах.
ППЗУ контроллер, обеспечивающий быстрое выполнение последовательных операций.
Последовательный контроллер осуществляет переход из одного состояния в другое. Эти контроллеры, применяемые для управления процессами и выполнения других последовательно организованных задач, зачастую не могут допустить тех значительных затрат времени, которые требуются микропроцессору на выборку инструкций и их запоминание. Несмотря на сложность такого последовательного алгоритма, для его реализации требуются только три интегральные схемы два ППЗУ и хранящий регистр. Этот набор микросхем стоит дешевле, чем микропроцессор, однако может также программироваться для ряда разнообразных приложений, что упрощает производство контроллеров. Блок ввода/вывода формирует данные, поступающие в микросистему и выдаваемые из нее, а регистры обеспечивают кратковременное хранение данных в процессе обработки. Блок операндов данных обеспечивает модификацию и изменение данных, а регистраторы индикацию числа событий или циклов, имевших место для конкретной операции.
Вначале система находится в состоянии готовности, ожидая приема входных данных. Принимаемые данные формируются для обработки блоком ввода вывода, который затем посылает УП в контроллер. Контроллер переходит после

этого в свое очередное состояние, выдавая МК для выполнения операции, определяемой его последним состоянием и УП.
Микрокоманды управляют всеми системными функциями. Фактическая обработка данных осуществляется при помощи блока операндов данных; этот блок выполняет такие операции, как сдвиг, сложение и передача данных. УП, поступающие в контроллер от блоков регистраторов, операндов и ввода вывода, служат для контроллера признаками окончания конкретной операции, и в этот момент данные выдаются на выход системы.
Если воспользоваться схемой ППЗУ микроконтроллера, то любой сложный алгоритм можно реализовать при помощи программируемых постоянных запоминающих устройств и хранящего регистра. Регистр в данном ППЗУ микроконтроллере выполнен на триггерах D типа. В триггерном контроллере обычно применяются триггеры JK типа, которые являются менее помехозащищенными.
Входное управляющее ППЗУ хранит таблицу истинности для алгоритма наборы управляющих переменных и кодов адресов состояний, соответствующих конкретным адресам этого ППЗУ. При каждом тактовом импульсе выходные коды ППЗУ передаются в хранящий регистр, с которого они поступают в декодирующее ППЗУ.
Сложность алгоритма, который может быть реализован при помощи ППЗУ контроллера, является функцией числа выходов управляющего ППЗУ. Например, если ППЗУ имеет 4 выхода, то оно может определять 16 состояний. Управляющие ППЗУ могут, однако, соединяться параллельно, чтобы реализовать более сложные алгоритмы.
Управляющее ППЗУ
Взятый в качестве примера алгоритм, имеет 8 состояний, так что необходимы по меньшей мере 3 выходные линии ППЗУ. Число слов, равное 256 соответствует пяти УП и трем адресным линиям. Каждая из 256 ячеек должна содержать двоичное значение, соответствующее одному из восьми состояний алгоритма.
Декодирующее ППЗУ
Преобразует коды состояний в микрокоманды. Число выходов ППЗУ определяется числом микрокоманд алгоритма, которое в свою очередь определяется числом логических функций, выполняемых под управлением контроллера последовательности. Для рассматриваемого примера необходимы семь микрокоманд, управляющих различными регистрами, операндами и счетчиками в системе. Поскольку алгоритм имеет всего восемь состояний, достаточно было бы декодирующего ППЗУ размером 8 слов по 7 бит. Поскольку ППЗУ такого размера промышленностью не выпускаются, использовано ППЗУ емкостью 32*8 бит.
ППЗУ микроконтроллер с мультиплексированием
Возможно построение ППЗУ микроконтроллеров, у которых число входных переменных превышает число входов управляющего ППЗУ. Это позволяет реализовать очень сложные алгоритмы при использовании достаточно малых по размерам ППЗУ. Переменные выбираются по одной в соответствии с состояни-

ем системы. Поэтому через мультиплексор не могут проходить переменные, которые должны возникать одновременно.

Быстродействующий вариант

В приложениях, требующих выполнения очень сложного алгоритма за очень короткий промежуток времени, можно использовать конвейерную структуру, чтобы исключить задержку перед генерацией микрокоманд. Совмещают функции декодирования и управления в одном входном ППЗУ. Подобной системе для декодирования микрокоманд больше не нужно ожидать, пока управляющее ППЗУ не переключиться в следующее состояние. Декодирование микрокоманд теперь осуществляется в предыдущем состоянии и они выдаются одновременно с переключением системы в следующее состояние. Единственным минусом в этом случае является размер декодирующего ППЗУ, которое теперь должно иметь столько же ячеек, сколько и управляющее ППЗУ.

ППЗУ микроконтроллер уже сейчас является привлекательным альтернативным вариантом, заменяющим контроллеры на базе микропроцессора, программируемых логических матриц или JK триггеров. Вскоре, с появлением больших интегральных схем, содержащих мультиплексор, ППЗУ и хранящий регистр на одном кристалле, он станет еще более привлекательным.

Тема 6. Планирование вычислительного процесса. Постановка задачи (2 часа)

Лекция № 6

6.4. Планирование вычислительного процесса в ППП.

ПОСТАНОВКА ЗАДАЧИ

При рассмотрении способов внешнего управления ППП в п. 5.4 были выделены процедурно-ориентированные и проблемно-ориентированные способы управления.

Процедурно-ориентированное управление предполагает, что пользователь сам определяет последовательность действий для решения задачи, а управляющие модули пакета только контролируют допустимость этих действий. В этом случае последовательность вызовов обрабатывающих и обслуживающих модулей полностью определяется пользователем.

Проблемно-ориентированное управление предполагает, что после выполнения действий по формированию состояния модели предметной области пакета, например путем ввода

значений известных пользователю данных, указываются имена данных, значения которых нужно вычислить. Дальнейшее планирование вычислительного процесса должно быть проведено управляющими модулями пакета.

Под *планированием вычислительного процесса* в ППП будем понимать определение такой последовательности вызовов обрабатывающих модулей, которая обеспечивает вычисление значений данных, указанных пользователем.

Можно дать и более общее определение задачи планирования вычислений: на основе сведений о модели предметной области пакета и ее текущем состоянии определить способ решения содержательно сформулированной задачи из пред-

метной области пакета. Применительно к структуре пакета, описанной в предыдущих разделах, способ решения задачи полностью определяется последовательностью вызовов модулей пакета, т.е. управляющим вектором такого же типа, как и управляющий вектор, создаваемый в ППП с командным входным языком в режиме компиляции.

Выше (п. 5.2) было установлено, что текущее состояние модели предметной области пакета можно характеризовать вектором $S = \{s_1, s_2, ..., s_n\}$, где n — число данных (элементов множества данных X), и

Обрабатывающий модуль и соответствующую ему функциональную связь можно рассматривать как функцию $y=f_i(x)$, где x□X есть входные (исходные) данные модуля, y□X — выходные данные (результаты выполнения модуля). В общем случае *хну* могут быть векторами, составленными из элементов X.

В текущем состоянии S обрабатывающий модуль f_i будем называть *выполнимым,* если значения всех компонентов x известны. Выполнимый обрабатывающий модуль можно вызвать (реализовать соответствующую этому модулю функциональную связь). В результате выполнения модуля будут вычислены новые значения компонентов y. Если значения каких-либо компонентов y в текущем состоянии S были неопределенными, то после выполнения модуля f_j будет получено новое состояние □ модели предметной области, и количество известных данных увеличится.

Выполнимый модуль будем называть эффективным в состоянии S, если его вызов переводит модель предметной области в новое состояние □.

Совокупность функциональных связей в модели предметной области пакета можно представить матрицами T и R, имеющими по m строк (m — число функциональных связей) и по n столбцов (n — число данных). Элемент матрицы T есть

Если рассматривать S как вектор-строку с элементами 1 и 0 и операции & (конъюнкцию), ∨ (дизъюнкцию) и ¬ (отрицание) интерпретировать как поэлементные булевские операции над битовыми строками, то условие выполнимости модуля f_i можно записать так:
т.е. результатом вычисления выражения R_i & (¬ S) будет битовая строка, содержащая хотя бы одну единицу.

В п. 5.6 было показано, что совокупность возможных состояний модели предметной области может быть представлена графом переходов, узлы которого соответствуют состояниям модели, а дуги — обрабатывающим модулям (функциональным связям). Указывая значения известных ему данных, пользователь определяет исходное состояние модели предметной области, т.е. один из узлов графа переходов. Дуги, исходящие из какого-либо узла графа переходов, соответствуют эффективным модулям в этом узле. Указывая список данных, значения которых требуется вычислить, пользователь определяет множество конечных состояний — узлов графа переходов, в которых известны как данные, заданные пользователем как исходные, так и данные, которые необходимо вычислить. Если обозначить вектор исходного состояния S_0, а требования к конечному состоянию определить вектором Z, и принимая, что компоненты этих векторов равны единице для известных данных и равны нулю для данных

с неопределенными значениями, то конечное состояние — это любое состояние S_k, удовлетворяющее условию S_k & Z=Z.

Задача планирования вычислительного процесса в этой интерпретации состоит в поиске пути на графе переходов из заданного исходного узла в любой из узлов, соответствующие условию.

Существует несколько алгоритмов решения этой задачи. Два возможных алгоритма описаны в последующих разделах.

6.5. 6.5. Алгоритм планирования с прямым и обратным ходом

Один из возможных алгоритмов планирования вычислений основан на так называемом *методе "прямой волны"*. Сущность этого метода состоит в том, чтобы последовательно включать в управляющий вектор все модули, эффективные в каждом текущем состоянии модели предметной области, и повторять этот процесс до тех пор, пока не будет получено одно из конечных состояний.

Если обозначить вектор искомых данных Z как битовую строку, элементы которой Z_j равны 1, если j-е данное является искомым, а остальные Z_j нулевые, то алгоритм прямой волны можно описать так.

1. 1. Подготовка: очистить формируемый управляющий вектор U, установить его текущую длину $K_u = 0$, установить текущее состояние S_T модели предметной области, $S_T = S_0$.
2. 2. Цикл, пока есть эффективные модули и не все искомые данные известны (S_T & Z ☐ Z), S_T — вектор текущего состояния.
   1.1. 1.1. Цикл по функциональным связям (по строкам матриц (T и R).
      1.1.1. 2.1.1. Если модуль f_i выполним и эффективен, т.е. (T_i & $S_т$= T_i) & (R_i & (⌝ $S_т$)#0),то:
         1.1.1.1. 2.1.1.1. Включить модуль f_1 в очередную позицию управляющего вектора U (например, $K_u = K_u + 1$, $U(K_u = i)$.
         1.1.1.2. 2.1.1.2. Преобразовать текущее состояние модели предметной области: $S_T = S_T \vee R_i$, затем перейти к п.2.2.
      1.1.2. 2.1.2. Конец цикла 2.1.
   1.2. 2.2. Конец цикла 2.
2. 3. Конец алгоритма. Если S_T & Z=Z, то получено решение задачи, в противном случае задача неразрешима.

Если задача планирования вычислений разрешима, то алгоритм прямой волны позволяет получить одно из возможных решений. Вместе с тем в управляющий вектор этот алгоритм включает и модули, выполнение которых не требуется для вычисления искомых данных.

Управляющий вектор можно улучшить, если применить к нему алгоритм метода обратной волны, суть которого состоит в следующем: начиная с конечного состояния, определяемого вектором искомых данных Z, отыскивать в управляющем векторе U модули, результатом которых являются данные, входящие в Z, отмечать их как выбранные и дополнять вектор Z не известными на данном шаге входами выбранного модуля. Пошаговое описание этого алгоритма выглядит так.

1. 1. Очистить (установить в нуль) битовую строку $\vee$ для отметок выбранных модулей.
2. 2. Цикл, пока не все элементы Z обращены в нули.
   1.1. 1.1. Цикл по строкам матриц T и R (по функциональным связям).
      1.1.1. 2.1.1. Если какой-либо результат модуля f_i содержится в Z $(R_1 \& Z \#0)$, то отметить этот модуль как выбранный ($V_1 = 1$) и преобразовать строку Z : $Z = (T_1 \& (\urcorner S,)) \vee (Z \& \urcorner R_1)$. где S_0 — исходное состояние модели предметной области перед выполнением прямого хода, затем перейти к п. 2.2.
      1.1.2. 2.1.2. Конец цикла 2.1.
   1.2. 2.2. Конец цикла 2.
2. 3. Построить улучшенный управляющий вектор U', включая в него только те обрабатывающие модули, которым соответствуют единицы в векторе отметок $\vee$.

Поясним работу алгоритмов прямой и обратной волны на примере. Пусть функциональные связи в модели предметной области ППП представлены табл. 6.2.

Пусть исходное состояние модели предметной области S_0 = (110000), т.е. известны значения данных a и b. Требуется найти последовательность вызовов модулей (управляющий вектор) для вычисления x, т.е. Z = (000001). Применим алгоритм прямой волны (прямой ход).

При первом прохождении цикла 2.1 в управляющий вектор включается модуль М1 и получается S_1 = (111000), на втором обороте этого цикла добавляется модуль М2 и S_2 = (111100), затем модуль М3 и S_3 = (111110) и, наконец, модуль М4 и S_4 = = (111111), после чего прямой ход заканчивается — данное x получило значение, и $K_u = 4$.

При обратном ходе S_0 = (110000), Z = (000001). При выполнении цикла 2 сначала отмечается как выбранный модуль М4 и получается Z_1 = (000010). Далее будет отмечен модуль М3 и Z_2 = (001000), последним отмечается модуль М1 и становится Z_3 = (000000), цикл 2 заканчивается. В управляющий вектор □ включаются модули М1, М3 и М4 в том же порядке, в каком они располагались в векторе U.

Если в графе переходов имеется несколько путей из исходного состояния в одно и то же или в разные конечные состояния, то существует и несколько вариантов управляющего вектора. Метод прямой и обратной волны позволяет найти один из этих векторов, причем конкретный вектор определяется принятым порядком просмотра матриц T и R для выделения эффективных модулей. В частности, для рассмотренного выше примера управляющий вектор из модулей М2 и М5 также решает поставленную задачу. Когда искомый управляющий вектор не единственный, можно ставить задачу о выборе оптимального в определенном смысле управляющего вектора — оптимального проведения вычислительного процесса.

6.6. 6.6. ЗАДАЧА ОПТИМАЛЬНОГО ПЛАНИРОВАНИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

Пусть эффективность вычислительного процесса в ППП при реализации управления на основе управляющего вектора U оценивается некоторой функцией $P(U)$. Тогда можно сформулировать задачу оптимального планирования вычислительного процесса.

Найти управляющий вектор U^*, такой, что

где D — множество всех управляющих векторов, обеспечивающих вычисление требуемых данных.

Выбор первой или второй постановки задачи зависит от физического смысла показателя эффективности $P(U)$. Если, например, $P(U)$ характеризует затраты времени на вычисление искомых данных, естественно стремиться к min $p(U)$, если $P(U)$ характеризует вероятность решения задачи за время, не превышающее некоторый предельный срок, естественно максимизировать $P(U)$.

Значение показателя эффективности $P(U)$ должно зависеть от того, какие и сколько обрабатывающих модулей вызывается при реализации U. Достаточно простая форма $P(U)$ получается, если сопоставить каждому обрабатывающему модулю неотрицательное число g_i, например средние затраты времени на выполнение модуля, а качество всего управляющего вектора оценивать суммой g_i т.е. где k — число компонентов управляющего вектора. Очевидно, что если g_i соответствует затратам времени, то нужно искать.

Рассмотрим один из возможных алгоритмов оптимального планирования вычислительного процесса.

В п. 5.2 было введено понятие графа переходов, отражающего возможные состояния модели предметной области пакета. Если рассматривать только ту часть этого графа, которая содержит переходы из единственного заданного начального состояния S в любое из конечных состояний S_k, в которых искомые данные известны, то задача оптимального планирования сводится к поиску такого пути на графе переходов из заданного узла S_0 в любой из конечных узлов, которому соответствует оптимальное значение $P(U)$. Усеченный граф переходов для условий примера из п.6.5 приведен на рис. 6.6.

Если принять $P(U) = \Sigma\, g_i$ и минимизировать значение $P(U)$, мы приходим к известной задаче определения кратчайшего пути в ориентированном графе, в которой значения g_i интерпретируются как длины соответствующих дуг.

Если допустить дублирование одинаковых состояний, можно преобразовать граф переходов в дерево, т.е. граф специального вида, в котором имеется единственный путь из начальной вершины (корня дерева) в любую из конечных вершин. На рис.6.7 показано дерево, соответствующее графу, изображенному на рис. 6.6, конечные вершины помечены буквой k.

В этой интерпретации задача оптимального планирования также сводится к задаче поиска пути к конечным узлам, но на дереве, а не на графе общего вида. Следует отметить, что не все пути по дереву обязательно ведут к такому конечному узлу, в котором все искомые данные известны. Возможны и тупиковые пути, которые заканчиваются до получения конечного состояния.

Для поиска оптимального пути на дереве возможных путей можно применить метод, часто называемый *методом ветвей и границ.* Сущность этого метода заключается в выполнении некоторого систематического обхода дерева с отбрасыванием ветвей, которые заведомо не ведут к оптимальному решению. Систематичность обхода обычно обеспечивается тем, что в каждом узле исходящие дуги проходят, например, слева направо. Правила отсечения ветвей строятся на основе анализа конкретной задачи.

Для задачи планирования вычислительного процесса с целевой функцией вида Σg_i можно предложить два правила отсечения ветвей.

1. 1. Бели дуга ведет в конечный узел дерева, не принадлежащий множеству допустимых конечных состояний S_k (т.е. если путь заканчивается, а искомые данные не определены).
2. 2. Бели уже найден вариант управляющего вектора, обеспечивающего вычисление всех искомых данных, и вычислено соответствующее ему значение $p(U')$, то при просмотре всех остальных путей можно отсекать последующие ветви, если текущее значение $p(U)$ окажется равным или больше $p(U')$ (в задаче минимизации Σg_i).

Последнее правило основано на монотонной зависимости $p(U)$ от числа модулей, включенных в управляющий вектор, добавление еще одного модуля в любом случае не уменьшает значение $p(U)$.

Очевидно, что построение и размещение в памяти всего дерева возможных путей может потребовать значительных затрат памяти. Поскольку в ходе поиска по дереву одновременно рассматривается только один путь, в памяти достаточно хранить информацию только об уже пройденных ветвях этого пути. Так как при переходе из одного узла в другой состояние модели предметной области меняется только частично, для каждого узла достаточно запоминать номер исходящей дуги (обрабатывающего модуля, включаемого в управляющий вектор) и список дополнительных данных, которые становятся известными при этом переходе.

Для случая, когда $p(U) = \Sigma g_i$ и оптимальным считается управляющий вектор с наименьшим значением $p(U)$, алгоритм оптимального планирования вычислительного процесса методом поиска по дереву возможных путей можно описать следующим образом.

По-прежнему будем считать, что функциональные связи представляются матрицами T и R, состоящими из нулей и единиц, и номера строк и столбцов этих матриц совпадают с номерами обрабатывающих модулей и данных. Узлы пройденного пути будем помещать в стек узлов. Каждый элемент стека содержит номер эффективного модуля, выбранного для перехода к следующему узлу, и список номеров данных, которые стали известны при переходе в этот узел. Алгоритм будет выглядеть так.

1. 1. Подготовка. Очистить стек узлов, поместить в стек информацию о корне дерева: номер эффективного модуля $i_э = 0$, список новых данных L — пустой. Построить вектор текущего состояния, учитывая в нем известные данные ($S_T = S_0$). Установить текущее значение показа-

теля эффективности $p_т = 0$ и положить $p^* = M$, где M — число, большее, чем — сумма весов всех модулей.

2. 2. Цикл, пока стек не пуст.

1.1. 2.1. Если текущее состояние конечное, т.е. асе искомые данные известны ($S_Т$ *& Z=Z),* и $p_т < p^*$, то запомнить полученный управляющий вектор в массиве u^* и заменить p^* на $p_т$.

1.2. 2.2. Если $p_т < p^*$, то удалить элемент из вершин стека с соответствующими преобразованиями $S_Т$ и $p_т$ и перейти к п.2.4.

1.3. 2.3. Для узла, находящегося в вершине стека:

1.3.1. 2.3.1. Найти следующий эффективный модуль, т.е. перебирать функциональные связи с номерами $i = i_э + 1$, $i_э + 2$, ..., m, пока не встретится модуль, удовлетворяющий условию

1.3.2. 2.3.2. Если найден новый эффективный модуль, то установить значение 1. и построить список новых данных L

а также пересчитать значение $p_т$ и построить новый элемент стека с $i_э = 0$ и списком новых данных, определяемых L, и вычислить новое текущее состояние $S_Т = S_Т \square L$, затем перейти к п. 2.4.

1.3.3. 2.3.3. Если в предыдущем пункте новый эффективный модуль не найден (просмотрены все исходящие дуги), то удалить текущий узел из вершины стека с соответствующими преобразованиями $S_Т$ и $p_т$.

1.4. 2.4. Конец цикла для п. 2.

2. 3. Если $p^* = M$, то задача неразрешима и управляющий вектор не найден. Если $p^* < M$, то получен оптимальный управляющий вектор U^*, этому вектору соответствует эффективность p^*.

В ходе выполнения алгоритма на шагах 2.2 и 2.3.3 удаляется элемент из вершины стека с преобразованиями $S_Т$ и $p_т$. Преобразование $S_Т$ состоит в установке нулевых значений элементов s_j, соответствующих новым данным из списка L удаляемого элемента стека. Преобразование $p_т$ заключается в вычитании из $p_т$ показателя эффективности g_i удаляемого узла. Для узла корня дерева $g_0 = 0$.

Применение описанного алгоритма в условиях примера, рассмотренного в п. 6.5, когда функциональные связи заданы табл. 6.2, в начальном состоянии известны значения данных a, b и требуется найти значение x, а все $g_i = 1$, $i = 1$, ..., 5, иллюстрирует табл. 6.3. В этой таблице показаны элементы стека после соответствующих шагов алгоритма, текущее состояние $S_т$ (список всех известных данных), текущее значение целевой функции $P_т$, лучшие значения управляющего вектора.

6.7. 6.7. Управление памятью в ППП

В пакете, предназначенном для решения расчетных задач, обрабатываемые в каждый момент времени данные должны размещаться в основной памяти ЭВМ. Под *управлением памятью* будем понимать совокупность действий по размещению данных в основной памяти, выделению памяти для данных и ее освобождению. В расчетных задачах наиболее используемыми типами данных

являются скалярные переменные, одномерные (векторы) и двухмерные (матрицы) массивы, а также структуры (записи в языке Паскаль) и массивы структур. Поскольку количество скалярных данных и структур обычно невелико, основной расход памяти приходится на размещение массивов, состоящих из элементов базовых типов или структур. Дальнейшее изложение вопросов управления памятью ориентировано на работу с массивами данных.

На выбор способа управления памятью влияют:

- вид модели предметной области (статическая или динамическая модель по отношению к множеству данных *X);*
- доступный объем основной памяти;
- возможности выбранного языка программирования и используемой ОС по управлению памятью.

На основе двух первых из перечисленных факторов можно различать четыре типовые ситуации, возникающие при проектировании ППП.

(С1) — статическая модель предметной области, даже при максимальных размерах всех массивов все данные умещаются в основной памяти.

(С2) — динамическая модель предметной области, при работе пакета данные создаются и уничтожаются, меняются размеры массивов, но все одновременно существующие данные умещаются в основной памяти.

(С3) — статическая модель предметной области, но определенные в модели данные не могут быть одновременно размещены в основной памяти.

(С4) — динамическая модель предметной области, одновременно существующие данные не могут быть размещены в основной памяти.

В первой и второй ситуациях все данные размещаются в основной памяти. Говорят, что в этом случае используется память одного уровня. В третьей и четвертой ситуациях часть данных, не используемых в данный момент времени, должна храниться в файлах на внешних запоминающих устройства: г.е. пакет должен использовать по меньшей мере два уровня памяти.

Наиболее просто решается проблема управления памятью в условиях ситуации (С1). При статической модели предметной области и достаточных размерах основной памяти можно выделить следующие типовые ситуации.

(С1.1) — для размещения каждого данного требуется постоянный объем памяти, не зависящий от желания или потребностей конкретного пользователя или условий решаемых задач.

(С1.2) — применение пакета всегда начинается с ввода пользователем некоторых данных, полностью определяющих объемы памяти для размещения всех остальных данных, не меняющиеся в течение сеанса работы с пакетом.

(С1.3) — используемый объем памяти (размеры массивов данных и необходимость их использования) может менялся в течение сеанса работы с пакетом. Например, в зависимости от решаемых задач некоторые данные могут вообще не требоваться, размеры памяти для других данных могут определяться в ходе работы пользователя с пакетом.

Очевидно, что если размеры всех данных ограничен и известны, при предельных размерах все данные умещаются в основной памяти, то можно выделить для каждого данного максимально необходимую память, и все управление

памятью сведется к ее распределению при проектировании пакета. В этих условиях случаи (С1.2) и (С1.3) сводятся к случаю (С1.1).

Для случая (С1.2) проблема распределения памяти решится в каждом сеансе работы с пакетом, но однократно, в омом начале сеанса или автономной части сеанса. Например, в пакете "Автоматизированное рабочее место для статистической обработки экономической информации" (АРМ СТОЭИ) [21] работа с новыми данными начинается с построения в памяти таблицы, и требуется указать число строк, число столбцов и размер граф таблицы. На основе этой информации выделяется память для статистической таблицы, после чего можно с этой таблицей работать. Для пакета, предназначенного для решения задач межотраслевого баланса производства и потребления, модель предметной области которого рассматривалась в п. 2.6, размеры всех массивов определяются числом отраслей производства, и сеанс работы с пакетом должен начинаться с ввода этого числа.

Реализация 'распределения памяти в условиях ситуации (С1.2) зависит от используемого языка программирования. В версии языка ПЛ/1, реализованной в ОС ЕС ЭВМ, эта задача решается наиболее просто. Массивам, имеющим переменные границы индексов, приписывается управляемый (CONTROLLED) класс памяти, а выделение памяти для таких массивов производится оператором ALLOCATE после того, как станут известны размеры массивов. При этом сохраняется возможность обращаться к массивам и их элементам по идентификаторам массивов. В языках программирования Паскаль, Си и версии ПЛ/1 для ПЭВМ средства динамического управления памятью ограничены, границы индексов массивов можно задавать только целыми константами. Имеющиеся в этих языках процедуры динамического выделения памяти могут реализовываться различно в разных ОС и на ЭВМ разных типов, что ограничивает возможность перенесения пакета на другие ОС и типы ЭВМ. При необходимости динамического размещения в памяти двухмерных массивов (матриц) может потребоваться отображение их на одномерные массивы, т.е. вместо $a[i,j]$ на Паскале придется записывать $a[i \cdot n + j]$, где n — длина строки матрицы. Передачу параметров-массивов подпрограммы следует производить с использованием указателей. Аналогичные проблемы возникают и при использовании языка Си.

В условиях ситуации (С1.3) для управления памятью могут использоваться те же приемы, что и в ситуации (С1.2), но действия по выделению памяти должны проводиться по мере необходимости в использовании данных или при изменении их размеров.

Следует отметить, что системные средства, т.е. реализуемые ОС или в языке программирования средства управления памятью, в ППП нужно применять с определенной осторожностью, если память для каких-либо данных не только выделяется, но и освобождается. В большинстве ОС, не ориентированных на виртуальную память со страничной организацией, механизм управления памятью состоит в следующем. Вначале вся свободная память оформляется как один свободный блок памяти, в первых словах которого помещается дескриптор (описатель) блока, содержащий длину блока и указание, что этот блок последний (поскольку он единственный). В ходе выполнения запросов на выделе-

ние и освобождение памяти свободные и занятые участки (блоки) памяти перемежаются. Свободные блоки объединяются в цепочку, и дескриптор каждого свободного блока содержит, кроме длины блока, ссылку на следующий свободный блок (рис. 6.8). При запросе памяти проводится поиск по этой цепочке и подбирается подходящий по размеру свободный блок, из которого и выделяется память требуемого объема. Подходящим считается либо первый свободный блок размера не менее требуемого, либо наименьший блок, позволяющий выделить требуемую память (для чего требуется либо упорядочить свободные блоки по размеру, либо каждый раз просматривать всю цепочку блоков). При освобождении памяти образуется новый свободный блок, который может объединяться со смежными свободными блоками. Недостатком такого управления памятью является опасность фрагментации свободной памяти, т.е. представление ее большим числом блоков малого размера, из которых ни один не удовлетворяет запросу на выделение памяти.

Опасность фрагментации памяти типична для ситуации (С2), когда состав обрабатываемых данных и их размеры могут меняться в ходе решения задачи. Для преодоления фрагментации памяти можно предложить следующий простой вариант управления памятью. При загрузке пакета выделяется (запрашивается) максимально доступный объем памяти — рабочее поле для размещения всех динамических данных. Для учета выделенной памяти образуется вектор управления памятью, каждый элемент которого содержит три поля: номер или код данного; указатель на первый байт области, занимаемой данным на рабочем поле, и длину данного, т.е. размер занимаемой им области. При инициализации пакета вектор управления памятью очищается. По мере поступления запросов на выделение памяти заполняются элементы вектора памяти. Разумеется, что выделение памяти для данного должно предшествовать присваиванию этому данному какого-либо значения. Если данное в ходе работы пакета теряет свое значение, это отмечается в соответствующем элементе вектора управления памятью или в описании данного в модели предметной области. Поскольку элементы управляющего вектора заполняются последовательно, данные размещаются на рабочем поле в порядке поступления запросов на выделение памяти. Если при поступлении очередного запроса окажется, что свободной памяти на рабочем поле недостаточно, должно проводиться уплотнение памяти. При уплотнении просматривается вектор управления памятью, и если встречается элемент, соответствующий данному, не имеющему значения, все последующие элементы управляющего вектора и данные на рабочем поле сдвигаются к младшим адресам, чтобы заполнить свободное место. В результате такого уплотнения свободная часть рабочего поля будет занимать сплошной участок в старших адресах этого поля. Работа алгоритма уплотнения показана на рис. 6.9.

Если запрос на выделение памяти не может быть выполнен после уплотнения данных на рабочем поле, пользователь получает отказ в выполнении последней введенной им команды.

Описанный алгоритм управления памятью изменяет адреса хранения данных в промежутках между обращениями к этим данным. Следовательно, к дан-

ным можно обращаться только по указателям на их адреса на рабочем поле, а значения этих указателей хранить в описаниях данных.

В ситуациях (С3) и (С4) требуется использование двухуровневой памяти, например в виде рабочего поля основной памяти и рабочего файла на диске. Использование памяти на двух уровнях может замедлить работу пакета, особенно если рабочий файл будет находиться на гибком диске. Если ЭВМ имеет RAM-диск (полупроводниковое запоминающее устройство, имитирующее магнитный диск), то размещение на нем рабочего файла обеспечит практически незаметное для пользователя снижение скорости работы пакета.

При статической модели предметной области (С3), по аналогии с ситуацией (С1), можно выделить случаи (С3.1), (С3.2) и (С3.3). В случаях (С3.1) и (С3.2) состав данных в пакете остается неизменным, размеры массивов либо известны заранее, либо устанавливаются в начале сеанса работы с пакетом, что позволяет определить для каждого массива постоянную позицию *в* рабочем файле, т.е. применить статическое размещение данных в рабочем файле.

При динамической модели предметной области в ситуации (С4) требуется динамическое управление внешней памятью, например, по такому же алгоритму, что и при динамическом распределении памяти в ситуации (С2), учитывать размещение данных во внешней памяти посредством ведения вектора управления внешней памятью и периодически проводить уплотнение внешней памяти. Если запрос на выделение памяти требует большего объема памяти, чем имеется свободного места на рабочем поле, то часть данных переносится с рабочего поля в рабочий файл, проводится уплотнение данных на рабочем поле, и повторяется попытка выполнить запрос на получение памяти на рабочем поле. Для реализации такого обмена между уровнями памяти нужно предусмотреть решение следующих проблем:

- определение данных, которые размещены на рабочем поле и не используются выполняемой подпрограммой пакета;
- организацию размещения данных в рабочем файле и учет его заполнения;
- регистрацию в модели предметной области, где хранится значение данного, на рабочем поле или в рабочем файле.

6.8. 6.8. Особенности вызова обрабатывающих модулей в ППП

Анализ алгоритмов управления ППП в п. 6.1-6.4 показывает, что независимо от процедурной или проблемной ориентации внешнего управления пакетом, применения командного входного языка или системы меню выполнение управляющих функций включает вызов той или иной управляющей или обрабатывающей подпрограммы — программного модуля с передачей в этот модуль фактических параметров, часть из которых играет роль входных данных, а остальные представляют переменные — результаты, для которых вычисляются новые значения.

В обычной прикладной программе для этого достаточно написать оператор вызова подпрограммы, например, CALL *P(X,Y,Z)* — при программировании на

ПЛ/1, просто *p(x,y,z)* — при программировании на Паскале или вызов функции *t=f (x,y,z)* — на языке Си. Вызываемая подпрограмма связывается с вызывающей через аппарат передачи параметров, а также может использовать внешние (в Паскале — глобальные) переменные.

В п. 5.2 отмечалось, что целесообразно различать обрабатывающие модули ППП, вызываемые всегда с одним и тем же набором фактических параметров, и модули, которые могут вызываться с различными наборами фактических параметров. Например, в модели предметной области, приведенной в п. 5.3, для модулей М7, М8, М9 и М10 предусмотрено по два набора параметров. Если ППП использует динамическую модель предметной области и пользователь может определять новые данные в процессе работы с пакетом, для отдельных модулей вообще нельзя перечислить заранее возможные наборы фактических параметров. Методы динамического управления памятью, описанные в предыдущем разделе, предполагают обращение ко всем данным не по их именам, а по указателям (ссылкам) на данные.

Перечисленные выше особенности вызова программных модулей в ППП требуют специальных приемов организации обращения к обрабатывающим и некоторым обслуживающим модулям. Сущность этих приемов обычно сводится к тому, чтобы часть действий по передаче фактических параметров в подпрограммы, выполняемых средствами языка программирования (обеспечиваемых при компиляции программы), выполнять управляющими модулями пакета, а подпрограммы вызывать с постоянными именами фактических параметров, причем все фактические параметры задаются указателями на соответствующие данные. Для этого в пакете предусматриваются специальные процедуры подготовки обращений к подпрограммам, формирующие списки фактических параметров на основе информации из модели предметной области и указаний пользователя пакета.

Возможны различные варианты реализации процедур подготовки обращений к подпрограммам. Например, может быть выделена специальная область связи с подпрограммами, в которую пересылаются все данные, используемые как фактические параметры, и размещаются там в заранее определенном для каждой подпрограммы порядке, а в саму подпрограмму передается только указатель на эту область связи. Другой подход состоит в формировании массива указателей на фактические параметры, в подпрограмму передается имя этого массива или указатель на него и, возможно, число фактических параметров.

Первый из указанных вариантов требует дополнительных затрат памяти для области связи и операций пересылки в эту область входных параметров, а после выполнения подпрограммы — пересылки ее результатов на отведенные места в памяти. Этот недостаток частично компенсируется возможностью сохранить старые значения данных при возникновении в подпрограмме особых ситуаций. Второй вариант позволяет избежать дополнительных затрат памяти и времени, но при ошибочных ситуациях в подпрограммах не гарантирует сохранность старых значений данных.

В результате успешного вызова обрабатывающего модуля или модуля ввода данных происходит изменение состояния модели предметной области, кото-

рое должно быть надлежащим образом зарегистрировано. Возникновение ошибочной ситуации в обрабатывающем модуле также может вызвать изменение состояния модели предметной области, если вследствие ошибки портятся какие-либо данные.

В общем случае вызов обрабатывающего модуля в ППП требует выполнения следующих операций.

1. 1. Проверка допустимости вызова модуля в текущем состоянии модели предметной области, в частности контроль наличия значений у всех входных параметров модуля.
2. 2. Проверка, выделена ли память для размещения результатов модуля (при динамическом управлении памятью).
3. 3. Подготовка списка (массива) указателей на фактические параметры или пересылка параметров в область связи.
4. 4. Собственно вызов программного модуля.
5. 5. Проверка успешности вызова модуля (отсутствие ошибочных ситуаций).
6. 6. Изменение состояния модели предметной области в зависимости от успешности вызова модуля.
7. 7. Если фактические параметры передавались через область связи и вызов модуля был успешным, то пересылка результатов обращения к модулю на закрепленные за ними места в памяти.

Очевидно, что реализация того или иного варианта подготовки параметров для вызова модуля требует соответствующего построения самого модуля. Конкретные правила программирования обрабатывающих модулей зависят от способа передачи параметров и используемого языка программирования.

Тема 7. Проектирование обслуживающих модулей программного комплекса (2 часа)

Лекции № 7

Информационный интерфейс базовых модулей.

Как уже говорилось, первоначальным строительным материалом (локальными вычислимыми функциями) в ГСП являются базовые модули. Информационный интерфейс базовых модулей представляет собой некое отношение, устанавливающее связь между типами данных базового модуля (формальными параметрами) и данными предметной области. В результате установления связи порождаются новые объекта либо акторы, либо предикаты.

Отношение p_j в ГСП формируется "паспортизацией" типов данных базовых модулей, т.е. за счет "опредмечивания" формальных параметров базовых модулей. В этом смысле отношение p_j является по сути "паспортом" модуля и вместе с базовым модулем определяют понятие актора или предиката (см. рис. 2.4).

Таким образом, каждый актор (предикат) через массив указателей P[N] "знает", над какими данными предметной области необходимо производить функциональные преобразования, описанные в соответствующем базовом модуле.

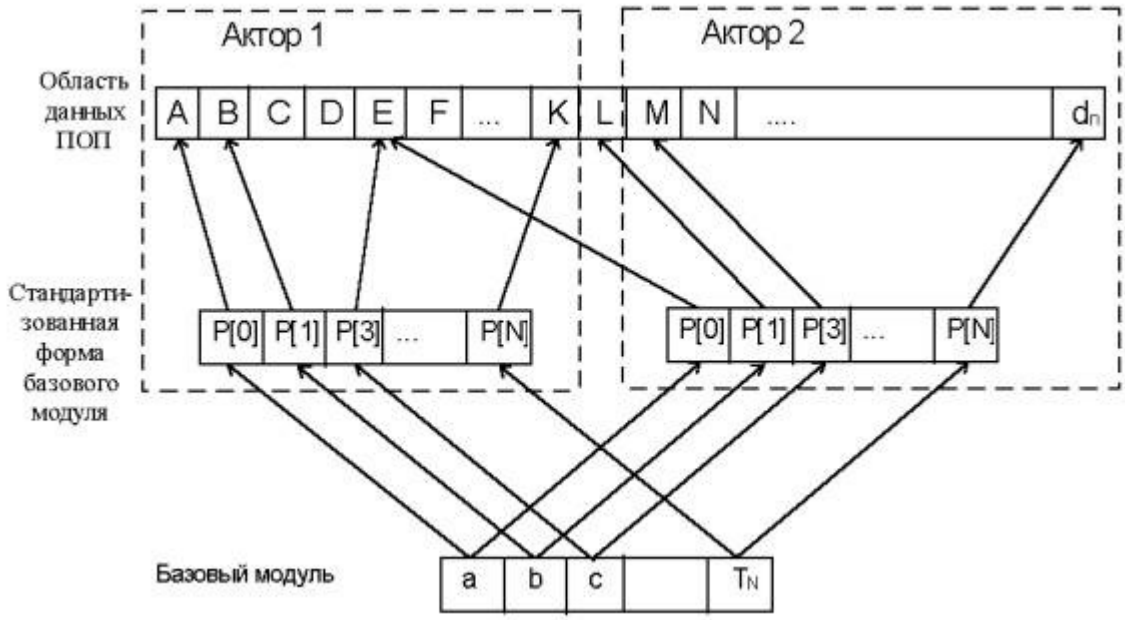


Рис. 2.4. Межмодульный интерфейс базового модуля.

1. Порядок разработки программного модуля.

При разработке программного модуля целесообразно придерживаться следующего порядка [8.1]:

- изучение и проверка спецификации модуля, выбор языка программирования;
- выбор алгоритма и структуры данных;
- программирование модуля;
- шлифовка текста модуля;
- проверка модуля;
- компиляция модуля.

Первый шаг разработки программного модуля в значительной степени представляет собой смежный контроль структуры программы снизу: изучая спецификацию модуля, разработчик должен убедиться, что она ему понятна и достаточна для разработки этого модуля. В завершении этого шага выбирается язык программирования: хотя язык программирования может быть уже предопределен для всего ПС, все же в ряде случаев (если система программирования это допускает) может быть выбран другой язык, более подходящий для реализации данного модуля (например, язык ассемблера).

На втором шаге разработки программного модуля необходимо выяснить, не известны ли уже какие-либо алгоритмы для решения поставленной и или близкой к ней задачи. И если найдется подходящий алгоритм, то целесообразно им воспользоваться. Выбор подходящих структур данных, которые будут использоваться при выполнении модулем своих функций, в значительной степени предопределяет логику и качественные показатели разрабатываемого модуля, поэтому его следует рассматривать как весьма ответственное решение.

На третьем шаге осуществляется построение текста модуля на выбранном языке программирования. Обилие всевозможных деталей, которые должны быть учтены при реализации функций, указанных в спецификации модуля, легко могут привести к созданию весьма запутанного текста, содержащего массу ошибок и неточностей. Искать ошибки в таком модуле и вносить в него требуемые изменения может оказаться весьма трудоемкой задачей. Поэтому весьма важно для построения текста модуля пользоваться технологически обоснованной и практически проверенной дисциплиной программирования. Впервые на это обратил внимание Дейкстра [8.2], сформулировав и обосновав основные принципы структурного программирования. На этих принципах базируются многие дисциплины программирования, широко применяемые на практике [8.3-8.6]. Наиболее распространенной является дисциплина пошаговой детализации [8.3], которая подробно обсуждается в разделах 8.2 и 8.3.
Следующий шаг разработки модуля связан с приведением текста модуля к завершенному виду в соответствии со спецификацией качества ПС. При программировании модуля разработчик основное внимание уделяет правильности реализации функций модуля, оставляя недоработанными комментарии и допуская некоторые нарушения требований к стилю программы. При шлифовке текста модуля он должен отредактировать имеющиеся в тексте комментарии и, возможно, включить в него дополнительные комментарии с целью обеспечить требуемые примитивы качества [8.1]. С этой же целью производится редактирование текста программы для выполнения стилистических требований.
Шаг проверки модуля представляет собой ручную проверку внутренней логики модуля до начала его отладки (использующей выполнение его на компьютере), реализует общий принцип, сформулированный для обсуждаемой технологии программирования, о необходимости контроля принимаемых решений на каждом этапе разработки ПС (см. лекцию 3). Методы проверки модуля обсуждаются разделе 8.4.
И, наконец, последний шаг разработки модуля означает завершение проверки модуля (с помощью компилятора) и переход к процессу отладки модуля.

Пошаговая детализация и понятие о псевдокоде.

Структурное программирование дает рекомендации о том, каким должен быть текст модуля. Возникает вопрос, как должен действовать программист, чтобы построить такой текст. Иногда программирование модуля начинают с построения его блок-схемы, описывающей в общих чертах логику его работы. Однако современная технология программирования не рекомендует этого делать. Хотя блок-схемы позволяют весьма наглядно представить логику работы модуля, при их кодировании на языке программирования возникает весьма специфический источник ошибок: отображение существенно двумерных структур, какими являются блок-схемы, на линейный текст, представляющий модуль, содержит опасность искажения логики работы модуля, тем более, что психологически довольно трудно сохранить высокий уровень внимания при повторном ее рассмотрении. Исключением может быть случай, когда для построения блок-схем используется графический редактор и они формализованы настолько, что

по ним автоматически генерируется текст на языке программирования (как например, это может делаться в Р-технологии [8.6]).
В качестве основного метода построения текста модуля современная технология программирования рекомендует пошаговую детализацию [8.1, 8.3, 8.5]. Сущность этого метода заключается в разбиении процесса разработки текста модуля на ряд шагов. На первом шаге описывается общая схема работы модуля в обозримой линейной текстовой форме (т.е. с использованием очень крупных понятий), причем это описание не является полностью формализованным и ориентировано на восприятие его человеком. На каждом следующем шаге производится уточнение и детализация одного из понятий (будем называть его уточняемым), использованного (как правило, не формализованно) в каком либо описании, разработанном на одном из предыдущих шагов. В результате такого шага создается описание выбранного уточняемого понятия либо в терминах базового языка программирования (т.е. выбранного для представления модуля), либо в такой же форме, что и на первом шаге с использованием новых уточняемых понятий. Этот процесс завершается, когда все уточняемые понятия будут выражены в конечном счете на базовом языке программирования. Последним шагом является получение текста модуля на базовом языке программирования путем замены всех вхождений уточняемых понятий заданными их описаниями и выражение всех вхождений конструкций структурного программирования средствами этого языка программирования.
Пошаговая детализация связана с использованием частично формализованного языка для представления указанных описаний, который получил название псевдокода [8.5, 8.8]. Этот язык позволяет использовать все конструкции структурного программирования, которые оформляются формализованно, вместе с неформальными фрагментами на естественном языке для представления обобщенных операторов и условий. В качестве обобщенных операторов и условий могут задаваться и соответствующие фрагменты на базовом языке программирования.
Головным описанием на псевдокоде можно считать внешнее оформление модуля на базовом языке программирования, которое должно содержать:

- начало модуля на базовом языке, т.е. первое предложение или заголовок (спецификацию) этого модуля [8.1];
- раздел (совокупность) описаний на базовом языке, причем вместо описаний процедур и функций - только их внешнее оформление;
- неформальное обозначение последовательности операторов тела модуля как одного обобщенного оператора (см. ниже), а также неформальное обозначение последовательности операторов тела каждого описания процедуры или функции как одного обобщенного оператора;
- последнее предложение (конец) модуля на базовом языке [8.1].

Внешнее оформление описания процедуры или функции представляется аналогично. Впрочем, если следовать Дейкстре [8.2], раздел описаний лучше также

представить здесь неформальным обозначением, произведя его детализацию в виде отдельного описания.

Неформальное обозначение обобщенного оператора на псевдокоде производится на естественном языке произвольным предложением, раскрывающим в общих чертах его содержание. Единственным формальным требованием к оформлению такого обозначения является следующее: это предложение должно занимать целиком одно или несколько графических (печатных) строк и завершаться точкой.

Для каждого неформального обобщеннного оператора должно быть создано отдельное описание, выражающее логику его работы (детализирующее его содержание) с помощью композиции основных конструкций структурного программирования и других обобщенных операторов. В качестве заголовка такого описания должно быть неформальное обозначение детализируемого обобщенного оператора. Основные конструкции структурного программирования могут быть представлены в следующем виде (см. рис. 8.2). Здесь условие может быть либо явно задано на базовом языке программирования в качестве булевского выражения, либо неформально представлено на естественном языке некоторым фрагментом, раскрывающим в общих чертах смысл этого условия. В последнем случае должно быть создано отдельное описание, детализирующее это условие, с указанием в качестве заголовка обозначения этого условия (фрагмента на естественном языке).

```
Следование:
    обобщенный_оператор
    обобщенный_оператор


Разветвление:
    ЕСЛИ  условие  ТО
      обобщенный_оператор
    ИНАЧЕ
      обобщенный_оператор
    ВСЕ ЕСЛИ


Повторение:
    ПОКА  условие  ДЕЛАТЬ
      обобщенный_оператор
    ВСЕ ПОКА
```

Рис. 8.2. Основные конструкции структурного программирования на псевдокоде.

```
Выход из повторения (цикла):
    ВЫЙТИ


Выход из процедуры (функции):
    ВЕРНУТЬСЯ


Переход на обработку исключительной ситуации:
    ВОЗБУДИТЬ  имя_исключения
```

Рис. 8.3. Частные случаи оператора перехода в качестве обобщенного оператора.

В качестве обобщенного оператора на псевдокоде можно использовать указанные выше частные случаи оператора перехода (см. рис. 8.3). Последовательность обработчиков исключительных ситуаций (исключений) задается в конце модуля или описания процедуры(функции). Каждый такой обработчик имеет вид:

```
ИСКЛЮЧЕНИЕ  имя_исключения
  обобщенный_оператор
ВСЕ ИСКЛЮЧЕНИЕ
```

Отличие обработчика исключительной ситуации от процедуры без параметров заключается в следующем: после выполнения процедуры управление возвращается к оператору, следующему за обращением к ней, а после выполнения исключения управление возвращается к оператору, следующему за обращением к модулю или процедуре (функции), в конце которого (которой) помещено данное исключение.

Рекомендуется на каждом шаге детализации создавать достаточно содержательное описание, но легко обозримое (наглядное), так чтобы оно размещалось на одной странице текста. Как правило, это означает, что такое описание должно быть композицией пяти-шести конструкций структурного программирования. Рекомендуется также вложенные конструкции располагать со смещением вправо на несколько позиций (см. рис. 8.4). В результате можно получить описание логики работы по наглядности вполне конкурентное с блок-схемами, но обладающее существенным преимуществом - сохраняется линейность описания.

```
УДАЛЕНИЕ В ФАЙЛЕ ЗАПИСЕЙ ДО ПЕРВОЙ,
УДОВЛЕТВОРЯЮЩЕЙ ЗАДАННОМУ ФИЛЬТРУ:
  УСТАНОВИТЬ НАЧАЛО ФАЙЛА.
  ПОКА  НЕ КОНЕЦ ФАЙЛА  ДЕЛАТЬ
    ПРОЧИТАТЬ ОЧЕРЕДНУЮ ЗАПИСЬ.
    ЕСЛИ  ОЧЕРЕДНАЯ ЗАПИСЬ УДОВЛЕТВОРЯЕТ
              ФИЛЬТРУ  ТО
      ВЫЙТИ
    ИНАЧЕ
      УДАЛИТЬ ОЧЕРЕДНУЮ ЗАПИСЬ ИЗ ФАЙЛА.
    ВСЕ ЕСЛИ
  ВСЕ ПОКА
  ЕСЛИ  ЗАПИСИ НЕ УДАЛЕНЫ  ТО
    НАПЕЧАТАТЬ "ЗАПИСИ НЕ УДАЛЕНЫ".
  ИНАЧЕ
    НАПЕЧАТАТЬ "УДАЛЕНО н ЗАПИСЕЙ".
  ВСЕ ЕСЛИ
```

Рис. 8.4. Пример одного шага детализации на псевдокоде.

Идею пошаговой детализации приписывают иногда Дейкстре [8.1]. Однако Дейкстра предлагал принципиально отличающийся метод построения текста модуля [8.2], который нам представляется более глубоким и перспективным. Во-первых, вместе с уточнением операторов он предлагал постепенно (по шагам) уточнять (детализировать) и используемые структуры данных. Во-вторых, на каждом шаге он предлагал создавать некоторую виртуальную машину для детализации и в ее терминах производить детализацию всех уточняемых понятий, для которых эта машина позволяет это сделать. Таким образом, Дейкстра

предлагал, по-существу, детализировать по горизонтальным слоям, что является перенесением его идеи о слоистых системах (см. лекцию 6) на уровень разработки модуля. Такой метод разработки модуля поддерживается в настоящее время пакетами языка АДА [8.7] и средствами объектно -ориентированного программирования [8.9].

Контроль программного модуля.

Применяются следующие методы контроля программного модуля:

- статическая проверка текста модуля;
- сквозное прослеживание;
- доказательство свойств программного модуля.

При статической проверке текста модуля этот текст прочитывается с начала до конца с целью найти ошибки в модуле. Обычно для такой проверки привлекают, кроме разработчика модуля, еще одного или даже нескольких программистов. Рекомендуется ошибки, обнаруживаемые при такой проверке исправлять не сразу, а по завершению чтения текста модуля.
Сквозное прослеживание представляет собой один из видов динамического контроля модуля. В нем также участвуют несколько программистов, которые вручную прокручивают выполнение модуля (оператор за оператором в той последовательности, какая вытекает из логики работы модуля) на некотором наборе тестов.
Доказательству свойств программ посвящена следующая лекция. Здесь следует лишь отметить, что этот метод применяется пока очень редко.

Тема 8. Применение программных комплексов в безопасности жизнедеятельности (3 часа)

Лекция № 8

Применение программных комплексов в безопасности жизнедеятельности(на примере ППП “Stalker”)

Запуск программного комплекса “STALKER”.

Выбрать ярлык **«Stalker»** и запустить комплекс (нажать «*Enter*»). Появляется окно комплекса **«Stalker»**.

2. Настройка параметров программного комплекса “STALKER”.
Примечание: Пункты, помеченные (*), для исполнения не обязательны.

2.1 В главном меню комплекса выбираем пункт «**Настройка / Параметры**» с помощью мыши или клавиши «*Alt*» и клавиш навигации. Откроется форма **«Параметры».**

<u>Примечание</u>: Подробное описание настройки параметров “STALKER'a” содержатся в файле помощи: открыть форму «Параметры» и нажать клавиши Ctrl + F1.

2.1.1. Установите значение **ДА** на следующие параметры (при помощи нажатия клавиши «*Space»* (пробел) на нужной строке) (нижеперечисленные параметры являются наиболее употребляемыми):

- «**Автомат плана размещения**» – если заполнена база данных (БД) «План размещения отходов» (см. 6.20.); то, после проведения каждого расчета отходов (в БД «Инвентаризация отходов»), отходы, указанные в плане размещения, автоматически будут занесены в БД-результат «Размещение отходов» (с указанными в плане размещения операциями использования, размещения, передачи для переработки и т.д.);
- «**Автомат вторичной инвентаризации**» – обеспечивает автоматический перерасчет в БД «Инвентаризация отходов» для отходов из БД “Вторичная инвентаризация”;
- «**Использовать рабочий список отходов**» – при ручном заполнении БД «План размещения отходов» будет подаваться список отходов и подразделений из БД «Рабочий список отходов». Также обеспечивает выполнение команды «Создать рабочий список отходов»;
- (*) «**Учитывать массы до 0,5 кг**» – будет производиться расчет нормативных объемов образования с точностью до 0,5 кг, с занесением в БД «Инвентаризация отходов»;
- (*) «**Вести протокол ошибок**» – ошибки, возникающие во время работы программы, будут формироваться в отдельный файл-протокол;
- (*) «**Протокол расчета класса опасности**» – в текстовом виде будет формироваться протокол расчета класса опасности;

2.1.2. После окончания заполнения формы нажимаем управляющую кнопку «ОК».

2.2 В главном меню комплекса выбираем пункт «**Настройка / Параметры печати**» с помощью мыши или клавиши «*Alt*» и клавиш навигации. Откроется форма **«Параметры печати»**

<u>Примечание:</u> Подробное описание настройки параметров печати содержатся в файле помощи: открыть форму «Параметры печати» и нажать клавиши Ctrl + F1.

2.2.1. Установите необходимые параметры печати:

- «Word без запросов» – можно указать ДА (при помощи нажатия клавиши «*Space»* (пробел) в синем поле)
- «Каталог отчетов» - укажите путь к каталогу, в котором должны создаваться отчеты при формировании проекта.
- «Word» - нажмите «*Enter*» и укажите путь к файлу winword.exe в появившемся окне выбора файла.

2.2.2. После окончания заполнения формы нажимаем управляющую кнопку «ОК».

2.3 В главном меню комплекса выбираем пункт «**Настройка**/**Защита справочников и нормативов**».

- Нажмите кнопку «Снять», для того, чтобы можно было вносить свои оригинальные характеристики и нормативы в базы данных (БД) справочников и нормативов.

3. (*)Удаление Контрольного примера (если требуется).

3.1. В главном меню комплекса выбираем пункт «**Данные/Предприятия**». Откроется форма "**Предприятия**"

3.1.1. В меню формы "**Предприятия**" выбрать пункт «**Запись/Удалить… /Удалить все записи**».

3.1.2. В окне запроса «Удалить данные о ВСЕХ предприятиях?» нажмите кнопку ДА.

3.1.3. В окне запроса «Удалить все записи?» нажмите кнопку ДА.

3.1.4. Нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

3.2. В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Адм-территориальное деление**». Откроется древовидная форма "**Адм-территориальное деление**"

3.2.1. В дереве "**Адм-территориальное деление**" установить курсор (синяя полоса) на начало (значок дерева)

3.2.2. Выберем пункт меню «Запись / Удалить» или нажмем кнопку «Удалить запись».

3.2.3. В окне запроса «Удалить запись и все ее подчиненные записи?» нажмите кнопку ДА.

3.2.4. Нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*»

4. Формирование дерева «Адм-территориальное деление».

4.1. В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Адм-территориальное деление**». Откроется древовидная форма "**Адм-территориальное деление**".

4.2. В БД "**Адм-территориальное деление**" нажимаем управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить сына» или нажмите на клавиатуре "*Ctrl* и +" для ввода местоположения предприятия (промплощадки).

<u>Примечание:</u> Древовидная БД "**Адм-территориальное деление**" ***предназначена для создания дерева, отражающего административное деление территории.***

Как минимум, дерево содержит три уровня:

- ***Субъект РФ;***
- ***Район, город;***
- ***Город районного подчинения, район/округ города, сельсовет.***

Рассмотрим простейший случай, когда дерево содержит три уровня: Субъект РФ, Район, Нас. пункт.

4.3. После того, как дерево “**Адм-территориальное деление**” построено, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

5. Заполнение справочников организаций.

<u>Примечание:</u> Пункты, помеченные (*) для исполнения не обязательны.

5.1. (*) В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Вышестоящие организации».** Откроется форма “**Вышестоящие организации**” (для проекта НООЛР не используется).

5.2. (*) В БД “**Вышестоящие организации**” необходимо внести:

- Наименование организации
- (*)Телефоны

5.3. (*) Если вышестоящая организация не одна, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “Ctrl и +” и повторите пункт 5.2.

5.4. (*)Закончив заполнение БД “**Вышестоящие организации**”, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

5.5. В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Объекты размещения общего назначения».** Откроется форма “**Объекты размещения общего назначения**” для ввода сведений об объектах конечного размещения, на которые предприятие вывозит отходы.

5.6. В БД “**Объекты размещения общего назначения**” необходимо внести:

- Наименование объекта
- Тип объекта конечного размещения (выбором из списка «Типы объектов конечного размещения отходов»)
- Город/район (выбором из дерева “**Адм-территориальное деление**”)
- Адрес
- Телефоны

5.7. Если объект размещения не один, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “Ctrl и +” и повторите пункт 5.6.

5.8. Закончив заполнение БД “**Объекты размещения общего назначения**”, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

5.9. В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Утверждающие, контролирующие и лицензирующие организации».** Откроется форма “**Утверждающие, контролирующие и лицензирующие организации**”

5.10. В БД “Утверждающие, контролирующие и лицензирующие организации” необходимо ввести:

- Наименование организации
- Должность ответственного лица
- ФИО ответственного лица
- (*)Телефоны

5.11. (*) Если организация не одна, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 5.10.

5.12. Закончив заполнение БД “**Утверждающие и контролирующие и лицензирующие организации**”, нажмите управляющую кнопку ОК в меню формы или клавишу *Esc*.

5.13. В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Транспортные организации»**.

5.14. В открывшейся БД “**Транспортные организации**” необходимо ввести:

- Наименование организации
- Адрес
- Телефоны
- ИНН
- Код ОКПО
- Код СОАТО
- Номер лицензии
- Дата выдачи
- Срок действия
- Лицензия выдана – название организации, выдавшей лицензию
- Номер договора о вывозе отходов

5.15. Если транспортная организация не одна, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 5.14.

5.16. Закончив заполнение БД “**Транспортные организации**”, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

5.17. (*) В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Лаборатории экоконтроля»**.

5.18. (*) В открывшейся БД “**Лаборатории экоконтроля**” необходимо ввести:

- Наименование лаборатории
- Адрес
- Телефоны
- Номер лицензии
- Дата выдачи
- Срок действия
- Лицензия выдана – название организации, выдавшей лицензию
- Номер договора о проведении анализов

5.19. (*) Если лаборатория не одна, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 5.18.

5.20. Закончив заполнение БД “ **Лаборатории экоконтроля**”, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

5.21. (*) В главном меню комплекса выбираем пункт «**Данные / Другие справочники… / Организации – разработчики проектов»** (для проекта НООЛР не используется).

5.22. (*) В открывшейся БД “**Организации – разработчики проектов**” необходимо ввести:
- Наименование организации
- Адрес
- Телефоны
- Код ОКПО
- Код ОКОНХ
- ИНН
- Номер лицензии
- Дата выдачи
- Срок действия
- Лицензия выдана – название организации, выдавшей лицензию
- Должность – должности ответственных разработчиков
- ФИО – ФИО ответственных разработчиков

5.23. (*) Если организация – разработчик проектов не одна, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 5.22.

5.24. Закончив заполнение БД “**Организации – разработчики проектов**”, нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

6. Подготовка исходных данных для формирования проекта НООЛР.

<u>Примечание</u>: Пункты, помеченные (*) для исполнения не обязательны.

6.1. В главном меню комплекса выбираем пункт «**Данные / Предприятия»**

6.2. В открывшейся БД “**Предприятия**” нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +”.

6.3. Нажмите «*Enter*» на поле «Город/Район» - появится древовидная БД “**Адм-территориальное деление**”. Курсором выберите в ней населенный пункт, в котором расположено предприятие и нажмите «*Enter*».

6.4. Поле «Наименование предприятия (юр. лица) краткое» заполняете, вводя наименование предприятия вручную. В поле «Год» вводите год данных инвентаризации (для формирования лимитов).

6.5. Поля «Полное наименование предприятия (паспортные данные инд. предпринимателя)» «Адрес почтовый», «Юридический адрес», «E-mail», «Должность руководителя», «Руководитель», «Телефон», «Отв. за ООС», «Телефон» заполняются вручную.

6.6. Поле «Подразделения и арендаторы» - при нажатии клавиши «*Enter*» появляется Подч. БД “**Подразделения и арендаторы**”

6.7. В Подч. БД “**Подразделения и арендаторы**” необходимо ввести следующие данные: Наименование подразделения
- Производство (основное или вспомогательное)
- Прочие? (если производство вспомогательное, то можно указать прочее или нет)

- Номер промплощадки, цеха, участка
- Характеристика подразделения (текст)
- Численность работающих
- (*) Арендатор - указать какая организация является арендатором, если имеется (для проекта НООЛР не используется).
- (*) Сведения об арендаторе (Площадь земельного участка, Численность работающих, Единиц автотранспорта) (для проекта НООЛР не используется).
- Включать в проект (ДА или НЕТ (пусто))

6.8. (*) Если подразделение не одно, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре "*Ctrl* и +" и повторите пункт 6.7.

6.9. Закончив заполнение БД "**Подразделения и арендаторы**", нажмите управляющую кнопку ОК в меню формы или клавишу «*Esc*».

6.10. Поле «Объекты размещения отходов» - при нажатии клавиши «*Enter*» появляется Подч. БД **"Собственные объекты размещения отходов**"

6.11. В Подч. БД **"Собственные объекты размещения отходов**" необходимо ввести следующие данные:

- Подразделение (подразделение, которому принадлежит объект; можно выбрать из списка ранее введенных подразделений или выбрать «Предприятие в целом», если объект не принадлежит одному подразделению)
- № объекта (площадки)
- Наименование объекта, краткое
- Временное накопление (ДА или НЕТ (пусто))
- Тип объекта временного накопления (выбор из списка)
- Вид обустройства площадки (выбор из списка)
- Тип объекта конечного размещения (выбор из списка)
- (*) Координаты X, Y
- (*) Площадка? (закрытая, открытая)
- Занимаемая площадь, м2
- (*) Требованиям экобезопасности (соответствует, не соответствует)
- Инвентаризация (остатки и получено) – Подч.БД, в которую вводят те отходы и их общую массу образования в тоннах, которые были получены от других предприятий или являлись остатком на момент проведения инвентаризации за текущий год
- Предельный допустимый объем, т
- Количество емкостей
- Объем одной емкости, м3
- Коэффициент заполняемости
- Плотность отходов, т/м3
- Максимально возможный объем, м3
- Вывоз, раз в год
- (*) Предприятие- арендодатель объекта
- (*) Реквизиты объекта (только для арендуемых объектов)
- (*) Предприятие, куда передается
- (*) Кем вывозится

– (*) Способ удаления отходов
– (*) Мероприятия по оборудованию площадки
– (*) Мониторинг ОПС
– (*) План-график контроля (можно указать для любого отхода) (для проекта НООЛР не используется).
– (*) Характеристика объекта (только для объектов конечного размещения отходов)

6.12. Часть полей БД “Собственные объекты размещения отходов” заполняются автоматически после проведения расчета по какому-нибудь из модулей на основании введенных данных инвентаризации и размещения образованных отходов:

– Объемы размещения;
– Общий объем, т/год;
– Временное накопление, т
– Предельное накопление, т;
– Расчет вывоза, раз в год.

6.13. Если объект размещения отходов не один, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 6.13.

6.14. Закончив заполнение БД **“Собственные объекты размещения отходов**”, нажмите кнопку ОК или «*Esc*».

6.15. Поле “Продукция” - при нажатии клавиши «*Enter*» появляется Подч.БД “**Продукция**”.

<u>Примечание:</u> Если на предприятии нет выпускаемой продукции, то Подч. БД “Продукция” можно не заполнять.

6.16. В Подч. БД “**Продукция**” необходимо ввести следующие данные:

– Подразделение (подразделение, в котором производится продукция; можно выбрать из списка ранее введенных подразделений или выбрать «Предприятие в целом», если невозможно указать конкретное подразделение)
– Наименование продукции
– Код продукции по ОКП
– Количество
– Единицы измерения кол-ва продукции
– Код по ОКЕИ
– Продукция поступает
– Перевозка продукции

6.17. Если продукции несколько видов, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 6.16.

6.18. Закончив заполнение Подч. БД “**Продукция**”, нажмите кнопку ОК или «*Esc*».

6.19. (*) Поле “План размещения отходов” - появляется Подч. БД “**План размещения отходов**” ***(1-ый, ручной способ заполнения)***. Если воспользоваться 2-ым способом заполнения БД “План размещения отходов” (см пункты 8.1-8.5), то пункты 6.20-6.22 можно не выполнять.

<u>Примечание:</u> Подч. БД "План размещения отходов" используется при автоматическом размещении отходов, образующихся в структурных подразделениях предприятия. Например: если известно, в каком подразделении образуется конкретный отход и в БД "План размещения отходов" заранее указать операции обращения с этим отходом (размещение или утилизация отхода), то при проведении расчета в БД «Инвентаризация отходов» сведения об операциях обращения с таким отходом будут автоматически внесены в БД «Размещение отходов».

6.20. (*) В Подч. БД **"План размещения отходов**" необходимо ввести следующие данные:

- Отход - выбрать из справочника отходов тот, данные по которому известны или оставить «все отходы» (тогда все отходы, образованные во время инвентаризации в указанном подразделении, будут размещены или утилизированы в соответствии с планируемой операцией обращения с отходом)
- Подразделение источник, где образуется отход - выбрать из списка подразделений-источников, где образуется выбранный отход
- Код плана размещения - можно указать код размещения (если код не указан, то впоследствии программа автоматически его укажет)
- Планируемая операция обращения с отходом - выбрать операцию по обращению с отходами (использование, размещение и др.)
- (*) Максимальная масса, т - указать максимально возможный объем размещения
- Заполнить «синие» поля, объединенные общим заголовком «План размещения»: Объект временного накопления; (*)Подразделение-приемник, где используется отход; (*)Собственный объект размещения; Объект размещения общего назначения; (*)Предприятие, куда передается; (*)Код 1-ой операции по размещению; (*)Код 2-ой операции по размещению; Операция временного накопления; (*)Операция конечного размещения; Операция по использованию; (*)Цель передачи; (*)Территориальный признак передачи.
- (*)Заполнить «синие» поля, объединенные общим заголовком «План вторичного использования»: (*)Процесс-источник, где образуется отход; (*)Процесс-приемник, где используется отход; (*)Вид работ процесса-приемника; (*)Ресурс процесса-приемника; (*)Норматив процесса-приемника.

  Поля, не затребованные для указанной операции обращения с отходами, будут черного цвета (не для ввода).

6.21. (*) Если подразделение не одно, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре "*Ctrl* и +" и повторите пункт 6.20.

6.22. (*) Закончив заполнение БД **"План размещения отходов**", нажмите кнопку ОК или *Esc*.

6.23. (*) Поле "Плановые мероприятия" - появляется Подч.БД "**Плановые мероприятия**".

<u>Примечание:</u> Если у предприятия нет плановых мероприятий по обращению с отходами, то Подч.БД "Плановые мероприятия" можно не заполнять.

6.24. (*) В Подч.БД "**Плановые мероприятия**" необходимо ввести следующие данные:

- Отход (выбрать из справочника отходов)
- Наименование мероприятия
- Содержание мероприятия
- Год начала выполнения
- Срок выполнения
- Ожидаемая эффективность

6.25. (*) Если плановое мероприятие не одно, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре "*Ctrl* и +" и повторите пункт 6.24.

6.26. (*) Закончив заполнение БД "**Плановые мероприятия**", нажмите кнопку ОК или *Esc*.

6.27. Поле "Установки использования/обезвреживания" - появляется Подч. БД "**Установки использования/обезвреживания**", в которую вводят сведения для отчета "**Сведения о применяемых технологиях, установках использования или обезвреживания отходов».**

6.28. Поле "Очистные сооружения сточных вод" - появляется Подч.БД "**Очистные сооружения сточных вод** ", в которую вводят общие сведения для отчета «Характеристика очистных сооружений и осадка хозяйственно-бытовых и промышленных сточных вод и водоподготовки».

6.29. Поле "ПГУ, оборудов. для очистки воздуха" - появляется Подч.БД **"ПГУ, оборудование для очистки воздуха",** в которую вводят общие сведения для отчета «Характеристика пылегазоочистных устройств и оборудования для очистки воздуха»

6.30. Поле "Сведения в текст проекта" - появляется Подч.БД "**Сведения в текст проекта**".

6.31. В Подч. БД "**Сведения в текст проекта**" необходимо внести следующие данные:

- (*) Вышестоящая организация – выбор из списка, формирование которого описано в пунктах 4.1.-4.4. (для проекта НООЛР не используется).
- (*) Утверждающая организация - выбор из списка, формирование которого описано в пунктах 4.5.-4.8. (для проекта НООЛР не используется).
- (*) Исполнитель проекта - выбор из списка, формирование которого описано в пунктах 4.13.-4.16. (для проекта НООЛР не используется).
- Вид основной деятельности (краткая характеристика)
- Категория предприятия
- Площадь земельного участка, га; Площадь занимаемой территории, м2; Площадь застройки, м2
- Площадь усовершенствованных покрытий, м2
- Площадь неусовершенствованных покрытий, м2
- Площадь озеленения, м2

– Площадь убираемой территории, м2
– Граничит с севера/ востока/ юга/ запада
– Санитарная классификация
– Размер СЗЗ, м
– Ближайшая жилая застройка, м
– Количество корпусов
– Численность работающих
– Режим работы
– Количество смен
– Дней в неделю
– Зоны отдыха, заповедники и т.п.
– Контроль осуществляется
– Характеристика деятельности (полная)
– (*)Перерабатывающие технологии
– Воздействие на ОС (открытые площадки)
– Воздействие на ОС (закрытые площадки)
– Воздействие на ОС (собственные объекты)

6.32. Завершив заполнение БД “**Сведения в текст проекта**”, нажмите кнопку ОК или *Esc*.

6.33. (*) Поле «Результаты подготовки текста» - появляется Подч. БД «**Результаты подготовки текста проекта**», которая заполняется автоматически после выполнения команды «Подготовка текста проекта» и содержит сведения, которые помещаются в текст ПНООЛР во время «Сборки проекта НООЛР» (см. пункт 9.5)

6.34. (*) Поле “Соблюдение правил ТБ и экологической безопасности” - появляется Подч. БД “**Соблюдение правил ТБ и экологической безопасности**”, в которую вручную вводятся сведения, которые затем помещаются в текст ПНООЛР во время «Сборки проекта НООЛР».

6.35. (*) В Подч. БД “**Соблюдение правил ТБ и экологической безопасности**” могут быть введены сведения для каждого образующегося отхода:
- Отход (выбрать из справочника отходов). Если в «Справочнике отходов» для этого отхода есть соответствующая информация, то она будет перенесена в БД “Соблюдение правил ТБ и экологической безопасности”, в поле «Правила безопасности».
- Правила безопасности – вводят текстовую информацию.

6.36. Если есть правила безопасности не для одного отхода, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 6.35.

6.37. Закончив заполнение БД “**Соблюдение правил ТБ и экологической безопасности**”, нажмите кнопку ОК или *Esc*.

6.38. Поле “Предоставляемые документы” - появляется Подч. БД **“Предоставляемые документы”.**

6.39. В Подч.БД **“Предоставляемые документы”** необходимо внести перечень документов, представляемых совместно с проектом ПНООЛР, как то: проект ПДВ, проект ПДС – если есть, договор об аренде земли и т.п.:
- -Порядок - порядковый номер документа
- -Документ – текст (название документа, №, дата выдачи, продления и т.п.)

6.40. Если представляемый документ не один, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункт 6.39.

6.41. Закончив заполнение БД **“Предоставляемые документы”**, нажмите кнопку ОК или *Esc*.

6.42. Поля с кодами предприятия: «**ОКПО**», «**Код ОКОНХ**», «**ИНН**», «**ОКАТО**», «**ОКВЭД**» и сведениями о разрешении на размещение отходов: «**Номер разрешения**», «**Срок действия**», «**Дата выдачи**», «**Дата продления**», а также «**Объект временного накопления**», «**Куда вывозится**» и «**Кем вывозится**» заполняются вручную.

6.43. (*) Если предприятие не одно, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункты 6.3. – 6.42.

6.44. После того, как БД “**Предприятия**” заполнена, нажмите кнопку ОК или *Esc*.

7. Проведение инвентаризации ресурсов и отходов с помощью имеющихся модулей.

7.1 В главном меню комплекса выбираем пункт «**Данные / Инвентаризация ресурсов и отходов**». Открылось меню «Ресурсы и отходы».

<u>Примечание</u>: Расчет проведем на примере модуля “Технологические процессы и виды производств в промышленности” и модуля “Отдельно стоящие объекты. Образование ТБО”.

7.2 В меню «Ресурсы и отходы» для проведения расчета выберите типовой процесс **“Технологические процессы и виды производств в промышленности”.** Нажать кнопку ОК.

7.3 Появилась БД **“Технологические процессы и виды производств в промышленности”**, в которую необходимо ввести следующие данные:

– Предприятие – выбрать из уже введенных
– Подразделение – выбрать из уже введенных
– Вид работ (справочник видов работ) – выбрать из древовидного справочника, поставляемого заполненным
– Ресурс (справочник ресурсов) – выбрать из предложенных ресурсов
– Количество (задается количество израсходованных ресурсов)
– Единица измерения
– Коэффициент (коэффициент перевода единиц измерения (литр, м3 и т.п.) в кг)

7.4 Поле «**Расчет отходов**» - при нажатии «*Enter*» появляется запрос «Рассчитать нормативные массы отходов?». По ответу «ДА» программа выполнит расчет и откроется Подч. БД «**Инвентаризация отходов. …»**

7.5 Подч. БД «**Инвентаризация отходов. …»** содержит результаты расчетов:

– Отход - какой отход(ы) образуется от выбранного вида работ и указанных ресурсов
– Общая масса отхода, т
– Масса до грамма
– Нормообразующее значение – переносится значение, указанное в справочнике нормативов: "**Промышленность. Нормативы образования отходов**"
– и т.д.

<u>Примечание</u>: Для удобства просмотра и работы можно перейти в страничную форму: выберите в меню БД пункт «БД/ Форма» или нажмите на клавиатуре "Ctrl и F".

7.6 (*) Поле «Размещение отходов… – операции» - открывается БД «Размещение М=XX.XXX (отход)», которая заполняется автоматически, если в БД "План размещения отходов" есть размещение этого отхода (см. пункт 6.19) и в признаке «Автомат плана размещения» выставлено «Да» (см. пункт 2.1.1). В противном случае необходимо вручную заполнить поля:

- Операция
- Масса, т
- Объект временного накопления
- Подразделение, где используется
- Собственный объект размещения
- Объект размещения общего назначения
- Предприятие, куда передается
- (*) Код 1-ой операции по размещению (для проекта НООЛР не используется).
- (*) Код 2-ой операции по размещению (для проекта НООЛР не используется).

– Операция временного накопления
– (*)Операция по использованию
– (*)Цель передачи
– (*)Территориальный признак передачи

- Код плана размещения
- Код ручного размещения
- Процесс, где используется отход
- Занести в план

7.7 Нажать кнопку ОК 2 раза.

7.8 (*) Вернулись в БД **"Технологические процессы и виды производств в промышленности".** Если необходимо рассчитать нормативную массу образования отходов от другого вида работ и других ресурсов, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт

меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункты 7.3. – 7.7.

7.9 После завершения расчетов в БД “Технологические процессы и виды производств в промышленности” нажать кнопку ОК.

7.10 Вернулись в форму «**Ресурсы и отходы».** Выберите следующий типовой процесс, по которому будете производить расчет, например “**Отдельно стоящие объекты. Образование ТБО**” и перейдите к следующему пункту, иначе переходите к пункту 7.17.

7.11 В БД “**Отдельно стоящие объекты. Образование ТБО**” необходимо ввести следующие данные:

- Предприятие – выбрать из уже введенных предприятий
- Подразделение – выбрать из уже введенных
- Тип источника образования ТБО – выбрать из поставляемого заполненным справочника нормативов «ТБО. Нормативы образования»
- Название – вручную указать название источника образования ТБО
- Количество (задается количество на единицу измерения)
- Единица измерения

7.12 Поле «**Расчет отходов**» - при нажатии «*Enter*» появляется запрос «Рассчитать нормативные массы отходов?». По ответу «ДА» программа выполнит расчет и откроется Подч. БД «**Инвентаризация отходов Отдельно стоящие объекты. Образование ТБО»** с результатами расчета.

7.13 Выполнить действия, описанные в пунктах 7.5.-7.7.

7.14 (*) Вернулись в БД **“Отдельно стоящие объекты. Образование ТБО”.** Если необходимо рассчитать нормативную массу образования отходов от другого типа источника образования ТБО, то нажмите управляющую кнопку формы «Добавить запись» или выберите пункт меню «Запись / Добавить» или нажмите на клавиатуре “*Ctrl* и +” и повторите пункты 7.11. – 7.13.

7.15 После завершения расчетов в БД “**Отдельно стоящие объекты. Образование ТБО**” нажать кнопку ОК.

7.16 Вернулись в форму «**Ресурсы и отходы**». Выберите следующий типовой процесс, по которому будете производить расчет и перейдите к пункту 7.11, иначе переходите к следующему пункту.

7.17 После завершения всех расчетов по типовым процессам образования отходов нажмите в форме «**Ресурсы и отходы**» кнопку «Отмена» или клавишу *Esc*.

8. План размещения отходов (2-ой способ, автоматическое заполнение по результатам инвентаризации) (1-ый, ручной способ описан в пунктах 6.19. – 6.22.)

8.1 В главном меню комплекса выбираем пункт меню по желанию:

a) (*)«**Данные / План размещения отходов… / Создать рабочий список отходов**»

b) «**Данные / План размещения отходов… / Рабочий список в план размещения**».

8.2 Появляется список предприятий «**Выбор предприятия для работы**». Клавишами ↑ или ↓ (вверх или вниз) выбираем предприятие, для которого будем заполнять план размещения.

8.3 Нажать кнопку ОК для подтверждения выбора.

8.4 Появляется сообщение «**Заполнение рабочего списка отходов. Подождите**», затем появляется Подч. БД "**Рабочий список отходов**", которая содержит все отходы, попавшие в БД «Инвентаризация отходов» в результате проведения расчетов по типовым процессам, сведения о нормативной массе образования каждого отхода и подразделении, в котором отход образуется.

8.5 Нажать кнопку ОК или *Esc*.

8.6 Если был выбран пункт меню из варианта b), то появляется заполненная БД **"План размещения отходов**". Если был выбран пункт меню из варианта a), то надо выполнить пункт меню «**Данные / План размещения отходов… / Рабочий список в план размещения**», после чего появляется заполненная БД **"План размещения отходов**".

8.7 Для выхода нажмите кнопку ОК или *Esc*.

8.8 Пункт меню «**Данные / План размещения отходов… / Подготовить план»** используется перед командой «**Рабочий список в план размещения»**

8.8.1. Появляется запрос «**Подготовить план к автоматическому размещению для** (наименование предприятия)?». Нажмите кнопку ДА.

8.8.2. Появляется сообщение программы «**План для (**наименование предприятия**) к автоматическому размещению подготовлен!**». Нажмите кнопку ОК.

8.9 Пункт меню «**Данные / План размещения отходов… / Автоматическое размещение по плану**» служит для автоматического размещения всех отходов по плану, при этом удаляется все ручное размещение!

9. Подготовка, формирование и сборка проекта

Мастер сборки проекта

9.1 В главном меню комплекса выбираем пункт меню «**Разделы / Разделы ПНООЛР по приказу МПР №115… / Мастер сборки проекта НООЛР»,** который реализует автоматизированный режим пошагового формирования всего проекта НООЛР - от подготовки проекта до сборки проекта (равнозначно выполнению пунктов 9.2-9.36). Все разделы и таблицы проекта (отчеты) формируются только в Word!

9.1.1. Перед началом сборки будет проверен и очищен каталог отчетов. Возможен запрос «Каталог отчетов уже не существует! Назначить новый каталог отчетов?» или «В каталоге отчетов … есть файлы предыдущего проекта. Назначить новый каталог отчетов?».

9.1.2. Каждый шаг сборки предваряет запрос о необходимости его выполнения. Выбор "Нет" - означает пропуск шага. "Отмена" - прервать сборку. Не рекомендуется отвечать на этот запрос до завершения формирования в Word предыдущей таблицы. Такая поспешность может привести к переполнению канала связи с Word и появлению сообщений типа "Передача данных прервана".

9.1.3. Во время работы мастера появится дополнительное информационное окно – «Протокол сборки проекта».

9.1.4. Последним появляется запрос «Сформировать проект в целом? (сборка проекта)». После ответа «Да» появляется сообщение «Сборка проекта выполнена: XX из 25 шагов! Результат в …!». Нажать кнопку ОК.

6. Методические указания для проведения и выполнения лабораторных занятий.

Тема 1. Изучение правовой информационной системы Консультант+.

Цель работы – изучить разделы правовой информационной системы Консультант+, изучить все возможные в ней методы и структуры поиска необходимой информации, изучить инструментарий правовой информационной системы, освоить поиск нужной информации в документах.

Лабораторные работы проводятся и выполняются по методике изложенной в уроках в разделе «Помощь» программы Консултант+.

Тема 2. Изучение правовой информационной системы Гарант

Цель работы – изучить разделы правовой информационной системы Гарнат, изучить все возможные в ней методы и структуры поиска необходимой информации, изучить инструментарий правовой информационной системы, освоить поиск нужной информации в документах.

Лабораторные работы проводятся и выполняются по методике изложенной в уроках в разделе «Помощь» программы Гарант. После их выполнения студент получает задание, в котором приводится список вопросов, по которым необходимо найти информацию с помощью правовой информационной системы Гарант.

Список заданий и вопросов.

1) Для чего предназначен правовой навигатор и в каких случаях им пользуются? В каких разделах находится информация по бухгалтерскому учету, аудиту и статистической отчетности? Найти в разделе «Правовой навигатор» все что касается охраны труда.

2) В разделе «Справочная информация» найти налоговый календарь, в нем же найти список документов в которые были внесены изменения за период с 16. 04. 2006 по 22. 04. 2006.

3) Найти документы по охране труда водителей с помощью раздела «Поиск по реквизитам», в этом же разделе найти информацию по санитарно защитной зоне в нормативно технической документации.

4) Найти журнал «Законодательство».

5) Найти в разделе «Толковый словарь» определения следующим терминам: бухгалтерский учет, коммунитаризм, материальное право, пайщик, личное потребление.

Найти следующие документы, сделать в них закладки, перевести найденные документы в формат doc. и сохранит на рабочем столе в папке 1.

1) Документы по экологической экспертизе.
2) Регламент проведения государственной экологической экспертизы.
3) Положение об оценке воздействия на окружающую среду.
4) Документы по оценке инвестиционных проектов.
5) ПДК загрязняющих веществ в воде, питьевой воде, почве, воздухе рабочей зоны.
6) ОБУВ (ориентировочно безопасные уровни воздействия).
7) ГН (гигиенические нормативы).
8) Охрана поверхности водоемов.
9) Санитарные требования к предприятиям.
10) ПОТ (правила охраны труда) для водителей автотранспорта.
11) Очистка выбросов с очистных сооружений.
12) Требования к искусственному и естественному освещению.
13) Требования к пожаробезопасности.
14) Защита населения в ЧС.
15) Документы по мониторингу окружающей среды.

Методика расчета риска.

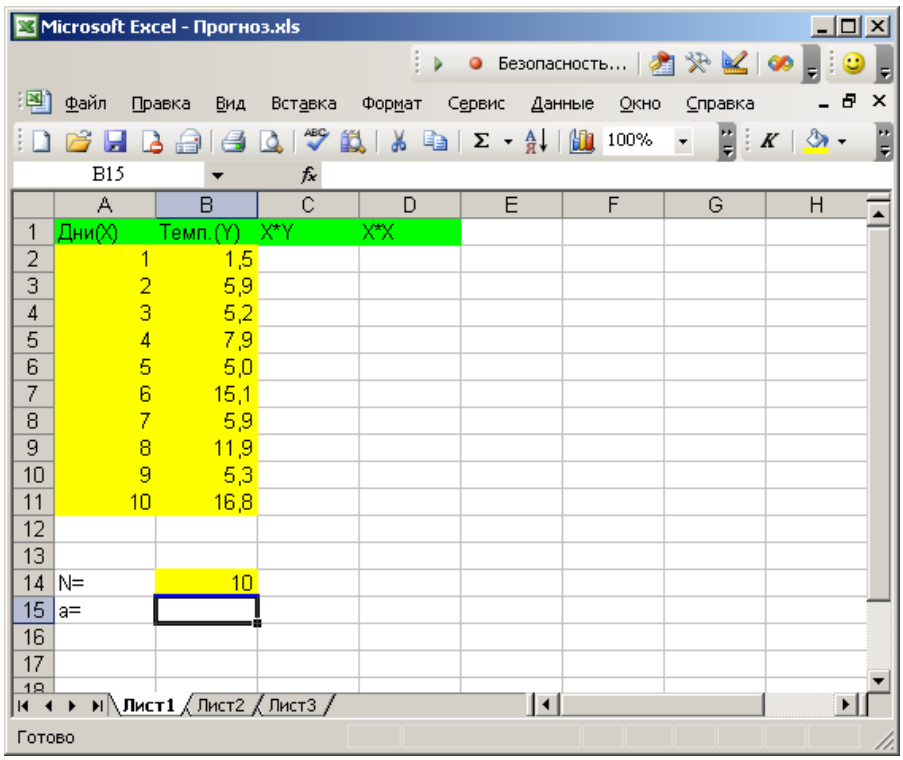
Тема 3. Работа с данными в MS Excel

Цель работы: научиться создавать электронные таблицы и выполнять в них расчеты с представлением результатов в виде графиков

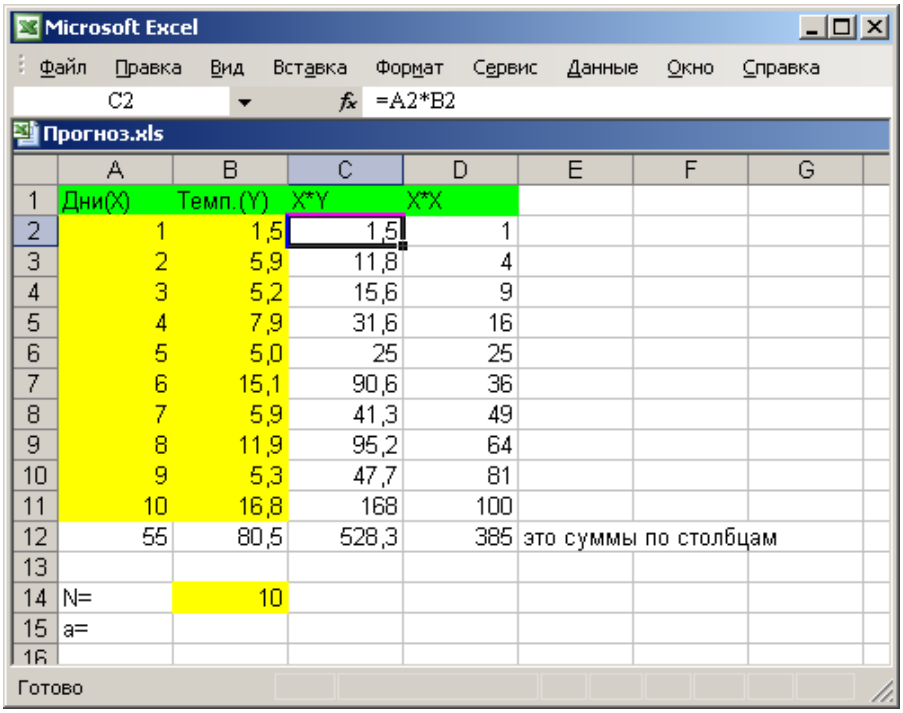
Порядок работы

Занятие 1 (Уровень А)

1.Набрать в первых двух столбцах «массив экспериментальных данных» X и Y, например: «Прогноз температуры» на 10 дней (ячейки A2:A11, B2:B11), первую строку использовать для подписи заголовков столбцов. Внизу таблицы записать количество дней (N=) и заголовок для расчетного коэффициента (a=)



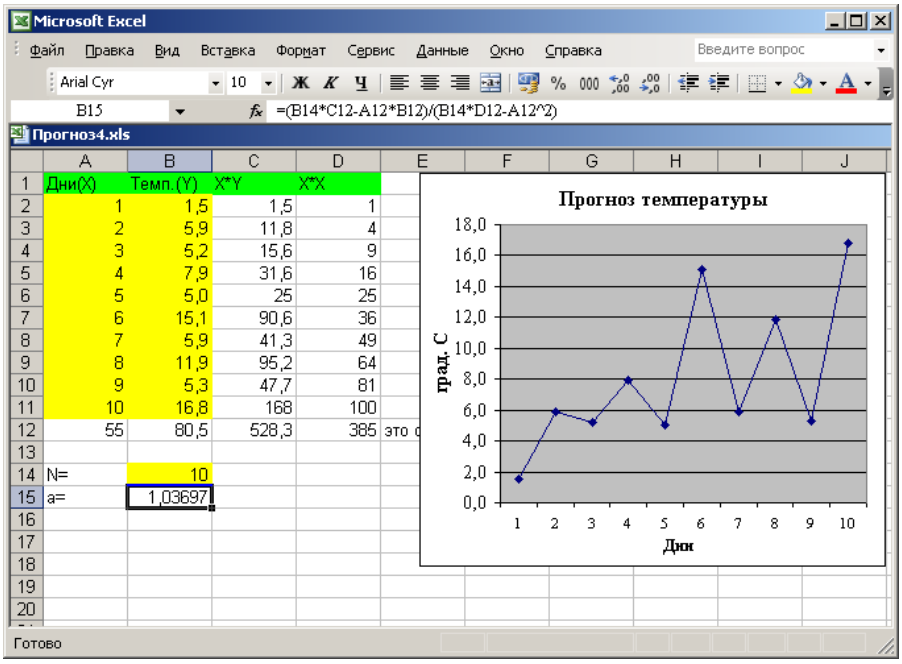
2.С помощью строки формул и копирования ячейки (правой кнопкой) третий столбец заполнить произведением первых двух столбцов (X*Y), четвертый столбец – квадратами от содержимого ячеек первого (X*X). Внизу всех столбцов вычислить сумму вышестоящих числовых ячеек, используя встроенную функцию =СУММ().



3.Записать заданную расчетную формулу

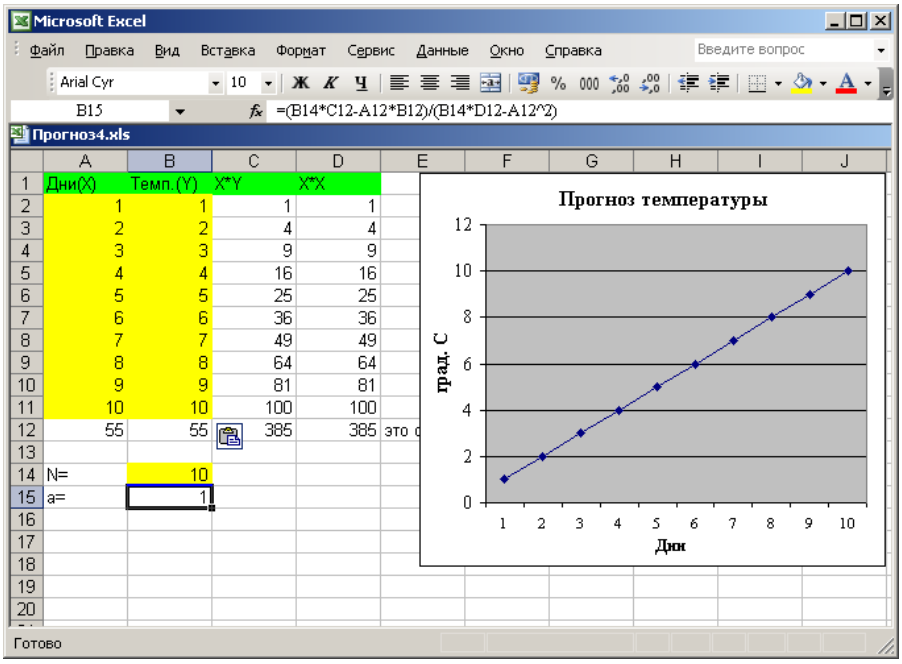
$$a = \frac{N * \sum_{i=1}^{N} X_i * Y_i - \sum_{i=1}^{N} X_i * \sum_{i=1}^{N} Y_i}{N * \sum_{i-1}^{N} (X_i)^2 - (\sum_{i=1}^{N} X_i)^2}$$

в терминах Excel и вычислить коэффициент линейной регрессии (a) в ячейке рядом с заголовком. На листе поместить диаграмму, в которой графически отобразить зависимость данных, размещенных во втором столбце (Y), от данных, размещенных в первом столбце (X). Сделать оформление диаграммы.



	A	B	C	D
1	Дни(X)	Темп.(Y)	X*Y	X*X
2	1	1,5	1,5	1
3	2	5,9	11,8	4
4	3	5,2	15,6	9
5	4	7,9	31,6	16
6	5	5,0	25	25
7	6	15,1	90,6	36
8	7	5,9	41,3	49
9	8	11,9	95,2	64
10	9	5,3	47,7	81
11	10	16,8	168	100
12	55	80,5	528,3	385
13				
14	N=	10		
15	a=	1,03697		

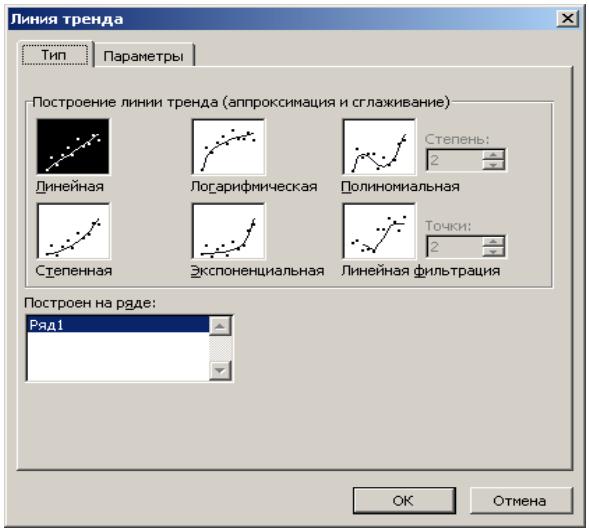
4.Проверить правильность вычисления коэффициента линейной регрессии (a) двумя способами: 1) путем задания «массива экспериментальных данных» в виде прямой линии под углом 45 градусов (при этом должно получиться: a=1), 2) способ см. далее.

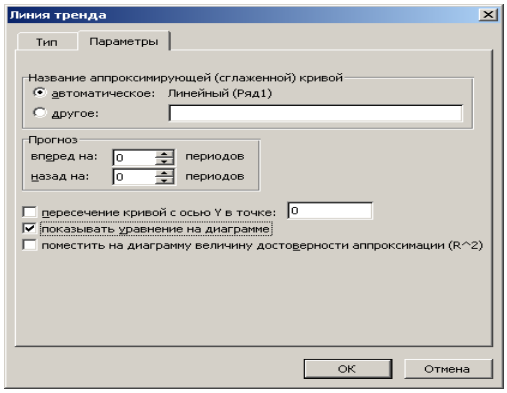


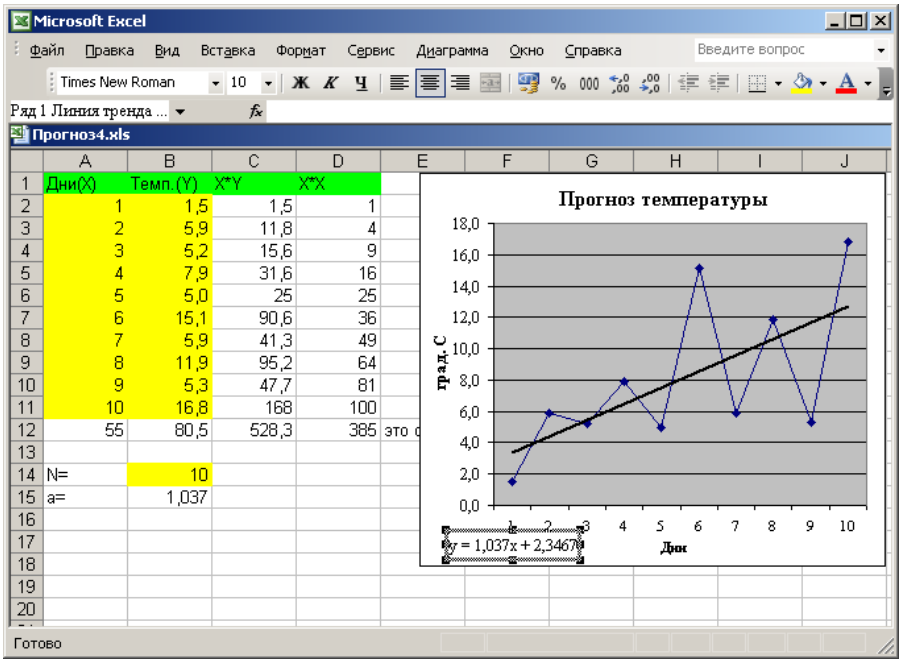
Занятие 2 (Уровень А)

5.Взять у преподавателя индивидуальное задание (число N>10) и повторить предыдущую работу для индивидуально заданного количества дней.

6.Добавить на график температуры линейную линию тренда (с помощью меню) и показать уравнение линии тренда на диаграмме







	A	B	C	D
1	Дни(X)	Темп.(Y)	X*Y	X*X
2	1	1,5	1,5	1
3	2	5,9	11,8	4
4	3	5,2	15,6	9
5	4	7,9	31,6	16
6	5	5,0	25	25
7	6	15,1	90,6	36
8	7	5,9	41,3	49
9	8	11,9	95,2	64
10	9	5,3	47,7	81
11	10	16,8	168	100
12	55	80,5	528,3	385
13				
14	N=	10		
15	a=	1,037		

Коэффициент при **X** в уравнении линии тренда должен совпасть с Вашим коэффициентом (a) в ячейке (в данном примере 1,037). Это второй способ проверки правильности выполненной работы.

(Уровень В)

7. Изменить массив экспериментальных данных во втором столбце 3 раза, задавая колебания температуры с возрастанием, с убыванием и колебания около стационарного значения. Проверить правильность расчета коэффициента (a) в этих трех случаях: 1) линия тренда возрастающая (a>0), линия тренда убывающая (a< 0), линия тренда параллельна оси X (a=0).

8.По результатам работы составить отчет, в котором привести значения заданных температур, расчетных коэффициентов (a) и линии тренда на трех графиках

Контрольные вопросы

1. Для чего предназначена линия тренда?
2. Чем отличается стиль A1 от стиля R1C1 в ячейках?
3. Как сделать относительный адрес ячейки абсолютным при операциях копирования?
4. Какие встроенные функции Excel Вы знаете?
5. Как разместить на одной диаграмме 2 и более графиков?
6. Как в формулах использовать ячейки из другого листа?

Методика выполнения лабораторных работ по данной теме изложена в книге – «А. Н. Васильев. Научные вычисления в Microsoft Excel. - М.: Издательский дом «Вильяме», 2004. – 512 с.: ил.».

Тема 4. Расчеты с помощью Mathcad

Цель работы – вспомнить и закрепить навыки работы в MathCAD.

Методика проведения работы

Задания для выполнения работ

Лабораторная работа № 9 – Определение напряженности электрического поля

Задание

Дано:

Высота от земли h = 1,8 м;

Фазное напряжение линии $U_{ф}$ = 500 кВ;

Расстояние между проводов d = 11.5 м;

Провода АСО-500 с радиусом r_0 = 1,51 см;

Шаг расщепления a = 40 см;

Высота подвеса провода на опоре $H_{п}$ = 20 м;

Габарит линии H_0 = 14 м

Расстояние от плоскости проходящей через ось линии фазы B перпендикулярно поверхности земли до точки P x = 25 м.

Найти:

1) Емкость фазы относительно земли C (используя формулу 9-7, отмеченную галочкой).

2) Линейную плотность заряда провода τ .

3) Найти модули векторов $E_{A(+)x}$, $E_{A(-)x}$, $E_{A(+)y}$, $E_{A(-)y}$.

4) Найти модули векторов E_{Ax} , E_{Ay} .

5) Наитии все отрезки m и n.

5) Найти коэффициенты k_1 - k_6 .

6) Найти комплексы напряжений фаз A, B, C - $\dot{U}_A$, $\dot{U}_B$, $\dot{U}_C$.

7) Найти комплексы вертикальных и горизонтальных составляющих напряженности электрического поля для всех трех фаз - $\dot{E}_{Ax}$, $\dot{E}_{Ay}$, $\dot{E}_{Bx}$, $\dot{E}_{By}$, $\dot{E}_{Cx}$, $\dot{E}_{Cy}$.

8) Найти комплексы суммы вертикальных и горизонтальных составляющих напряженности электрического поля для всех трех фаз - $\dot{E}_x$, $\dot{E}_y$.

9) Найти искомую напряженность электрического поля E трехфазной воздушной линии в заданной точке.

10) Определить зависимость напряженности электрического поля на расстоянии от земли равном 1,8 м от расстояние от плоскости проходящей через ось линии фазы B перпендикулярно поверхности земли до точки P. Выше это

расстояние обозначалось *x,* а выполняя данный пункт эту величину необходимо переопределить и обозначить эту величину другой несистемной переменной.

Лабораторная работа № 10 – Электрическое сопротивление тела человека

Задание

1) Найти емкостное сопротивление наружного слоя кожи $C_н$ по формуле (1-1). Где: ε =150 Ф/м; ε_0= 8,85*10^{-12} Ф/м; d = 0.15 мм (толщина эпидермиса); S=0.02 м^2.

2) Найти активное сопротивление по формуле (1-2). Где $\rho_н$ = 1,5*10^4 Ом* м.

3) Найти полное сопротивление тела человека в комплексной форме по формуле отмеченной галочкой на странице 3. $R_в$= 600 Ом. Из полученной комплексной формы сопротивления найти модуль, действительную и мнимую части и аргумент.

4) Найти полное сопротивление человека в действительной форме по формулам (1-3), (1-4) и (1-5).

5) С помощью формулы (1-6) построить график зависимости $Z_н(U_н)$.

6) С помощью формулы (1-7) построить график зависимости $I_н(U_н)$.

7) С помощью формулы (1-8) построить график зависимости $I_н(Z_н)$.

8) Используя формулу для напряжения пробоя $U_{ПР}$ на восьмой странице построить график зависимости $U_{ПР}(E_{ПР})$. $d_р$ = 0.05 мм; E = 500..200 В/мм.

9) Построить график зависимости $Z_h(S)$ по формуле (1-9) с учетом $R_в$ и по формуле (1-10) без учета $R_в$. f = 50 Гц, S = 8..400 см^2.

10) По формуле (1-9) построить график зависимости $Z_h(f)$, при S = 100 см^2, где f=0...20000 Гц.

Лабораторная работа № 11 – Анализ опасности поражения током в различных электрических сетях

Задание

1) Найти ток проходящий через тело человека при двухфазном прикосновении I_h по формуле на странице 3. Где: $U_Ф$ =220 В; R_h = 10000 Ом.

2) Найти напряжение прикосновения по формуле (4-1). Где U = 220 В; r_1 = 30 кОм; r_2 = 20 кОм.

3) Найти ток проходящий через тело человека I_h по формуле (4-2).

4) Найти I_h и $U_{ПР}$ по системе уравнений (4-3), где r = 40 кОм. Построить графики зависимостей $U_{ПР}(r)$ и $I_h(r)$, где *r* меняется от 0 до 40 кОм.

5) Найти эквивалентное сопротивление $r_Э$ при аварийном режиме по формуле на странице 6 r_2 = 1,2 кОм; $r_{зм}$ = 30 Ом.

6) Найти I_h и $U_{ПР}$ при прикосновении к незаземленному проводу с помощью уравнений (4-4), (4-5). Где r_0 = 11 Ом.

7) С помощью формулы (4-6) найти I_h при $r_П$ = 35 Ом, $r_{ОБ}$ = 25 Ом.

8) Использую формулу (4-6) для I_h построить графики зависимостей $I_h(R_h)$, $I_h(r_П)$, $I_h(r_{ОБ})$, $I_h(r_О)$, где R_h меняется от 3000 Ом до 100000 Ом, $r_П$ меняется от 20 Ом до 50000 Ом, $r_{ОБ}$ меняется от 1 кОм до 30 кОм, $r_О$ меняется от 5 до 50 Ом.

Лабораторная работа № 11 – Трехфазные сети

Задание

1) Найти Y_1, Y_2, Y_3, $Y_Н$ по отмеченным формулам на странице 2. Где: r_1 =10 Ом, r_2=20 Ом, r_3=40 Ом, $r_Н$=30, C_1= 0,03 мкФ, C_2= 0,05 мкФ, C_3= 0,01 мкФ, $C_Н$=0,02мкФ.

2) Найти $\dot{U}_0$ по отмеченной формуле на странице 3. Где $Y_Н$ = 0,1 Ом $^{-1}$, $Y_Н$ = 0,001 Ом $^{-1}$, $U_Ф$ = 220 В.

3) Найти $\dot{U}_{ПР}$ по формуле (4-8) и $\dot{I}_h$ по формуле (4-9).

4) Найти $U_{ПР}$ по формуле (4-11) и I_h по формуле (4-12).

5) С помощью формулы (4-13) построить график зависимости $U_{ПР}(r_{зм})$, где $r_{зм}$ =

= 0,01..200 Ом.

6) С помощью формулы (4-14) построить график зависимости $I_h(r_{зм})$, где $r_{зм}$ =

= 0,01..200 Ом.

7) С помощью формулы (4-16) найти $\dot{I}_h$.

8) С помощью формул (4-17) и (4-18) найти $\dot{I}_h$ и I_h, где *C* = 0,025 мкФ, *r* = 25 Ом.

9) Построить график зависимости $I_h(r)$ по формуле (4-19) $I_h(r_{зм})$, где r =

= 0,01..200 Ом.

10) Найти I_h по формуле (4-20).

Методика выполнения работы по теме 4 изложена в книге «П. А. Долин Основы техники безопасности в электроустановках: учебное пособие для вузов. -Москва.: Энергия, 1979. – 408 с., ил.»

Тема 5. Основы работы в MATlab

Цель работы – изучить основы работы в MATlab, основы графической визуализации вычислений, обработки данных, матричные вычисления, построение графиков функций, столбцовых диаграмм, построение трехмерных графиков, форматирование линий графиков, работу со справкой и примерами.

методические указания

Тема лабораторной работы – "Использование встроенных функций и операторов ППП MatLab for Windows".

Работа включает в себя изучение основных приемов работы в среде MatLab, в частности изучение встроенных функций и операторов для решения математических задач, описываемых в матричной форме.

Необходимо в соответствии с номером варианта решить с помощью MatLab ряд задач , включающих построение графика функции, различные действия над матрицами и решение системы нелинейных алгебраических уравнений.

Обратить внимание на применение основных символов и знаков при вводе данных, правил записи выражений, возможности встроенных функций. Подробнее можно прочитать в справочнике языка MatLab.

Задание

1. Построить график функции F(x) при изменении аргумента x (согласно номеру варианта).

1. $F(x) = (x-2)^2 2^x + x^2 - 20\sin x$.
2. $F(x) = arcctgx + 2(x-2)^2 - |4x-3|$.
3. $F(x) = (x-3)\cos x - 3^x + x^2$.
4. $F(x) = (x-2)^2 \lg(x+11) - 20\sin x$.
5. $F(x) = e^x + (x+2)^2 + x^2 \cos 2x$.
6. $F(x) = (x-3)^2 \log_{0,5}(x-2) - 5\arcsin x$.
7. $F(x) = (x-2)^2 \lg(x+11) + 2arctgx$.
8. $F(x) = \sin(x-0,5) + 2e^x - (x-2)^2 2^x$.
9. $F(x) = x\log_3(x+1) + \cos(x+0,3)$.
10. $F(x) = tg^3 x + x\lg(x+1)$.
11. $F(x) = 2e^x - (x-2)^2 2^x + x\log_3(x+1)$.
12. $F(x) = (x-3)\cos x + 0,5^x + 1$.
13. $F(x) = arctg(x-1) + 2x + e^{-2x} + x^2$.
14. $F(x) = arctg(x-1) + 2x + x^2 \cos 2x$.
15. $F(x) = x\lg(x+1) + x^2 2^x - 20\sin 5x$.
16. $F(x) = x\log_3(x+1)^2 + 2e^x + arcctgx$.
17. $F(x) = \sin(x+1) + (x+2)\log_2(x-3) + |5x-6|$.
18. $F(x) = 3^x + 5x - (x-3)\cos(x-2)$.
19. $F(x) = (x-2)^2 \lg(x+11) + 2^x x^2 - \sin x$.

2. Округлить элементы матрицы к ближайшему целому числу.

Номер варианта	Матрица	Номер варианта	Матрица
1	$A=\begin{pmatrix}1 & 1{,}5 & 2{,}5\\ 1{,}5 & 1{,}3 & 2{,}1\\ 2{,}5 & 2 & 1{,}7\end{pmatrix}$	11	$A=\begin{pmatrix}1{,}5 & 2{,}5 & 2{,}5\\ 1{,}3 & 2 & 1{,}2\\ 2 & 1{,}9 & 1{,}7\end{pmatrix}$
2	$A=\begin{pmatrix}1 & 1{,}2 & 2\\ 1{,}2 & 1 & 0{,}4\\ 2 & 0{,}4 & 2\end{pmatrix}$	12	$A=\begin{pmatrix}1 & 2{,}5 & 3{,}5\\ 1{,}5 & 2 & 1{,}6\\ 2{,}5 & 1 & 1{,}7\end{pmatrix}$
3	$A=\begin{pmatrix}1 & 1{,}2 & 2\\ 1{,}2 & 1 & 0{,}5\\ 2 & 0{,}5 & 2\end{pmatrix}$	13	$A=\begin{pmatrix}1{,}5 & 1{,}6 & 3{,}5\\ 1{,}6 & 2{,}5 & 1{,}2\\ 1{,}7 & 1{,}2 & 1{,}4\end{pmatrix}$
варианта		варианта	
4	$A=\begin{pmatrix}2{,}5 & 1 & 2\\ 1 & 2 & 1{,}2\\ 2 & 0{,}4 & 1{,}5\end{pmatrix}$	14	$A=\begin{pmatrix}1 & 2{,}5 & 3{,}5\\ 1{,}5 & 2 & 1{,}6\\ 2{,}5 & 1 & 1{,}7\end{pmatrix}$
5	$A=\begin{pmatrix}2 & 1 & 1{,}4\\ 1 & 1 & 0{,}5\\ 1{,}4 & 0{,}5 & 2\end{pmatrix}$	15	$A=\begin{pmatrix}2 & 1{,}7 & 1{,}6\\ 1{,}7 & 2 & 3{,}5\\ 1{,}6 & 2 & 1\end{pmatrix}$
6	$A=\begin{pmatrix}2 & 1{,}5 & 3{,}5\\ 1{,}5 & 2 & 2\\ 3{,}5 & 2 & 2\end{pmatrix}$	16	$A=\begin{pmatrix}3 & 1{,}7 & 1{,}6\\ 1{,}7 & 1 & 2\\ 1{,}6 & 2 & 3\end{pmatrix}$
7	$A=\begin{pmatrix}1{,}2 & 0{,}5 & 1\\ 0{,}5 & 1 & 2\\ 2 & 0{,}8 & 1\end{pmatrix}$	17	$A=\begin{pmatrix}1 & 1{,}5 & 1{,}2\\ 1{,}5 & 2 & 0{,}4\\ 1{,}2 & 2 & 1{,}5\end{pmatrix}$
8	$A=\begin{pmatrix}0{,}5 & 1{,}2 & 1\\ 1{,}2 & 2 & 0{,}5\\ 0{,}5 & 0{,}5 & 1\end{pmatrix}$	18	$A=\begin{pmatrix}1 & 2{,}5 & 3{,}5\\ 1{,}5 & 2 & 1{,}6\\ 2{,}5 & 1 & 1{,}7\end{pmatrix}$
9	$A=\begin{pmatrix}1{,}2 & 0{,}5 & 1\\ 0{,}5 & 1{,}4 & 2\\ 2 & 0{,}6 & 1\end{pmatrix}$	19	$A=\begin{pmatrix}1 & 1{,}5 & 0{,}4\\ 1{,}5 & 1 & 1{,}6\\ 0{,}4 & 1 & 2\end{pmatrix}$
10	$A=\begin{pmatrix}1 & 1{,}5 & 1{,}2\\ 1{,}5 & 2 & 0{,}4\\ 2{,}5 & 2 & 1\end{pmatrix}$		

3. Выполнить сложение, умножение и деление матриц.
4. Вычислить кумулятивные суммы для каждого столбца матрицы.

Номер варианта	Матрица	Номер варианта	Матрица
1	$A=\begin{pmatrix}1 & -1 & 2\\ 1 & 2 & 1\\ 2 & 0 & 3\end{pmatrix}$	11	$A=\begin{pmatrix}1 & 2 & 5\\ 1 & 4 & 1\\ 2 & 1 & 7\end{pmatrix}$
2	$A=\begin{pmatrix}3 & 1 & 1\\ 1 & 2 & 3\\ -2 & 5 & -2\end{pmatrix}$	12	$A=\begin{pmatrix}1 & 5 & 3\\ 1 & 2 & 6\\ 2 & 1 & 7\end{pmatrix}$
3	$A=\begin{pmatrix}-1 & 1 & -1\\ 4 & 4 & 2\\ 1 & 2 & 3\end{pmatrix}$	13	$A=\begin{pmatrix}5 & 6 & 3\\ 1 & 2 & 2\\ 7 & 1 & 4\end{pmatrix}$
4	$A=\begin{pmatrix}3 & 2 & -1\\ 2 & 4 & 1\\ -1 & 0 & -2\end{pmatrix}$	14	$A=\begin{pmatrix}1 & 2 & 3\\ 5 & 2 & 6\\ 2 & 1 & 7\end{pmatrix}$
5	$A=\begin{pmatrix}2 & 1 & 1\\ 1 & 1 & 5\\ 1 & 5 & 2\end{pmatrix}$	15	$A=\begin{pmatrix}2 & 7 & 6\\ 1 & 2 & 3\\ 6 & 2 & 1\end{pmatrix}$
6	$A=\begin{pmatrix}6 & 3 & 8\\ 3 & 2 & 3\\ -1 & 0 & -4\end{pmatrix}$	16	$A=\begin{pmatrix}3 & 7 & 1\\ 1 & 2 & 4\\ 6 & 2 & 3\end{pmatrix}$
7	$A=\begin{pmatrix}2 & 5 & 1\\ 5 & 1 & 2\\ 2 & 8 & 1\end{pmatrix}$	17	$A=\begin{pmatrix}1 & 5 & 2\\ 5 & 2 & 4\\ 1 & 2 & 5\end{pmatrix}$
8	$A=\begin{pmatrix}5 & 2 & 1\\ 1 & 2 & 5\\ 5 & 5 & 1\end{pmatrix}$	18	$A=\begin{pmatrix}1 & 5 & 3\\ 5 & 2 & 6\\ 2 & 1 & 7\end{pmatrix}$
9	$A=\begin{pmatrix}2 & 5 & 1\\ 7 & 1 & 3\\ 2 & 6 & 1\end{pmatrix}$	19	$A=\begin{pmatrix}1 & 5 & 4\\ 5 & 1 & 6\\ 4 & 1 & 2\end{pmatrix}$
10	$A=\begin{pmatrix}1 & 5 & 2\\ 1 & 2 & 4\\ 2 & 2 & 1\end{pmatrix}$		

5. Найти максимальный и минимальный элемент матриц.
6. Отсортировать каждый столбец матрицы в возрастающем порядке элементов.
7. Вычислить вектор-строку с суммами элементов каждого столбца матрицы.

1. $A=\begin{pmatrix} -14 & -9 & -4 & 1 & 6 \\ -1 & -2 & -2 & -4 & -14 \\ -14 & -9 & -9 & 3 & -2 \\ -15 & -11 & -11 & -3 & -8 \\ -29 & -20 & -20 & 0 & -10 \end{pmatrix}$

2. $A=\begin{pmatrix} -13 & -8 & -3 & 2 & 7 \\ 0 & -1 & 7 & -3 & -13 \\ -13 & -8 & 3 & 4 & -1 \\ -13 & -9 & 4 & -1 & -6 \\ -26 & -17 & 7 & 3 & -7 \end{pmatrix}$

3. $A=\begin{pmatrix} -12 & -7 & -2 & 3 & 8 \\ 1 & 0 & 8 & -2 & -12 \\ -12 & -7 & 4 & 5 & 0 \\ -11 & -7 & 6 & 1 & -4 \\ -23 & -14 & 10 & 6 & -4 \end{pmatrix}$

4. $A=\begin{pmatrix} -11 & -6 & -1 & 4 & 9 \\ 2 & 1 & 9 & -1 & -11 \\ -11 & -6 & 5 & 6 & 1 \\ -9 & -5 & 8 & 3 & 2 \\ -20 & -11 & 13 & 9 & -1 \end{pmatrix}$

5. $A=\begin{pmatrix} -10 & -5 & 0 & 5 & 10 \\ 3 & 2 & 10 & 0 & -10 \\ -10 & -5 & 6 & 7 & 2 \\ -7 & -3 & 10 & 5 & 0 \\ -17 & -8 & 16 & 12 & 2 \end{pmatrix}$

6. $A=\begin{pmatrix} -9 & -4 & 1 & 6 & 11 \\ 4 & 3 & 11 & 1 & -9 \\ -9 & -4 & 7 & 8 & 3 \\ -5 & -1 & 12 & 7 & 2 \\ -14 & -5 & 19 & 15 & 5 \end{pmatrix}$

7. $A=\begin{pmatrix} -8 & -3 & 2 & 7 & 12 \\ 5 & 4 & 12 & 2 & -8 \\ -8 & -3 & 8 & 9 & 4 \\ -3 & 1 & 14 & 9 & 4 \\ -11 & -2 & 22 & 18 & 8 \end{pmatrix}$

8. $A=\begin{pmatrix} -7 & -2 & 3 & 8 & 13 \\ 6 & 5 & 13 & 3 & -7 \\ -7 & -2 & 9 & 10 & 5 \\ -1 & 3 & 16 & 11 & 6 \\ -8 & 1 & 25 & 21 & 11 \end{pmatrix}$

9. $A=\begin{pmatrix} -6 & -1 & 4 & 9 & 14 \\ 7 & 6 & 14 & 4 & -6 \\ -6 & -1 & 10 & 11 & 6 \\ 1 & 5 & 18 & 13 & 8 \\ -5 & 4 & 28 & 24 & 14 \end{pmatrix}$

10. $A=\begin{pmatrix} -5 & 0 & 5 & 10 & 15 \\ 8 & 7 & 15 & 5 & -5 \\ -5 & 0 & 11 & 12 & 7 \\ 3 & 7 & 20 & 15 & 10 \\ -2 & 7 & 31 & 27 & 17 \end{pmatrix}$

11. $A=\begin{pmatrix} -4 & 1 & 6 & 11 & 16 \\ 9 & 8 & 16 & 6 & -4 \\ -4 & 1 & 12 & 13 & 8 \\ 5 & 9 & 22 & 17 & 12 \\ 1 & 10 & 34 & 30 & 20 \end{pmatrix}$

12. $A=\begin{pmatrix} -3 & 2 & 7 & 12 & 17 \\ 10 & 9 & 17 & 7 & -3 \\ -3 & 2 & 13 & 14 & 9 \\ 7 & 11 & 24 & 19 & 14 \\ 4 & 13 & 37 & 33 & 23 \end{pmatrix}$

13. $A=\begin{pmatrix} -2 & 3 & 8 & 13 & 18 \\ 11 & 10 & 18 & 8 & -2 \\ -2 & 3 & 14 & 15 & 10 \\ 9 & 13 & 26 & 21 & 16 \\ 7 & 16 & 40 & 36 & 26 \end{pmatrix}$

14. $A=\begin{pmatrix} -1 & 4 & 9 & 14 & 19 \\ 12 & 11 & 19 & 9 & -1 \\ -1 & 4 & 15 & 16 & 11 \\ 11 & 15 & 28 & 23 & 18 \\ 10 & 19 & 43 & 39 & 29 \end{pmatrix}$

15. $A=\begin{pmatrix} -4 & 1 & 6 & 11 & 16 \\ 9 & 8 & 16 & 6 & -4 \\ -4 & 1 & 12 & 13 & 8 \\ 5 & 9 & 22 & 17 & 12 \\ 1 & 10 & 34 & 30 & 20 \end{pmatrix}$

16. $A=\begin{pmatrix} -3 & 2 & 7 & 12 & 17 \\ 10 & 9 & 17 & 7 & -3 \\ -3 & 2 & 13 & 14 & 9 \\ 7 & 11 & 24 & 19 & 14 \\ 4 & 13 & 37 & 33 & 23 \end{pmatrix}$

17. $A=\begin{pmatrix} -2 & 3 & 8 & 13 & 18 \\ 11 & 10 & 18 & 8 & -2 \\ -2 & 3 & 14 & 15 & 10 \\ 9 & 13 & 26 & 21 & 16 \\ 7 & 16 & 40 & 36 & 26 \end{pmatrix}$

18. $A=\begin{pmatrix} -1 & 4 & 9 & 14 & 19 \\ 12 & 11 & 19 & 9 & -1 \\ -1 & 4 & 15 & 16 & 11 \\ 11 & 15 & 28 & 23 & 18 \\ 10 & 19 & 43 & 39 & 29 \end{pmatrix}$

19. $A=\begin{pmatrix} -4 & 1 & 6 & 11 & 16 \\ 9 & 8 & 16 & 6 & -4 \\ -4 & 1 & 12 & 13 & 8 \\ 5 & 9 & 22 & 17 & 12 \\ 1 & 10 & 34 & 30 & 20 \end{pmatrix}$

8. Решить графически систему нелинейных алгебраических уравнений.

Номер варианта	Система уравнений	Номер варианта	Система уравнений
1	$y+(x-2)^2 2^x=2$ $x^2-20\sin y=3$	11	$y+2e^x-(x-2)^2=6$ $2^y+x\log_3(x+1)=1$
2	$arcctgx+2(y-2)^2=5$ $y-\lvert 4x-3\rvert=3$	12	$y^3-(x-3)\cos x=4$ $0{,}5^x+2^{-y^2}=1$
3	$y+(x-3)\cos x=5$ $-3^x+y^2=2$	13	$y^2-arctg(x-1)=2$ $2x+e^{-2y}+x^2=4$
4	$y=(x-2)^2\lg(x+11)=7$ $-20\sin x-3^{x-3}=3$	14	$y^5+arctg(x-1)=4$ $2x+x^2\cos 2y=1$
5	$ye^x+(x+2)^2=3$ $x^2\cos 2x-5y^3=2$	15	$4^y-x\lg(x+1)=3$ $x^2 2^x-20\sin 5y=4$
6	$y(x-3)^2\log_{0,5}(x-2)=6$ $-5\arcsin x+2^{x+3}=2$	16	$y^3-x\log_3(x+1)^2=6$ $2e^y+arcctgx=2$
7	$y-(x-2)^2+\lvert x\rvert=6$ $\lg(y+11)+2arctgx=1$	17	$6^y+\sin(x+1)+(x+2)=5$ $\log_2(y-3)+\lvert 5x-6\rvert=1$
8	$y+\sin(x-0{,}5)+2e^x=4$ $(x-2)^2 2^x-y^2=1$	18	$3^y+5x-\ln(x-3)=3$ $\cos(x-2)+e^y=2$
9	$y-x\log_3(x+1)=3$ $\cos(x+0{,}3)+2^y=1$	19	$y^3-(x-2)^2\lg(x+11)=4$ $2^x x^2-\arcsin y=2$

ПРИМЕРЫ

Пример 1. Выполнить действия с матрицами (сложение и умножение). Над матрицами можно выполнять простейшие арифметические действия, используя соответствующие операторы.

```
>> A=[1 2 3; 4 5 6; 7 8 9]; B=[1 4 7; 2 5 8; 9 6 3]
>> C=A+B
C =
     2     6    10
     6    10    14
    16    14    12
>> D=A*B
D =
    32    32    32
    68    77    86
   104   122   140
```

Пример 2. Вычислить кумулятивные суммы для каждого столбца матрицы. Используем функцию CUMSUM(X).

```
>>  x=[2 4 6; 1 3 2];  cumsum(x)
ans =
        2     4     6
        3     7     8
>>
```

Пример 3. Округлить элементы матрицы к ближайшему целому числу. Для этого используем функцию ROUND(A).

```
>> x=[1.2 -3.2 3.3 4.8]; y=round(x)
y =
     1    -3     3     5
>>
```

Пример 4. Решить графически систему нелинейных алгебраических равнений.

$$\begin{cases} y + \cos^2 x = 0, \\ y - \sin^3 x = 0. \end{cases}$$

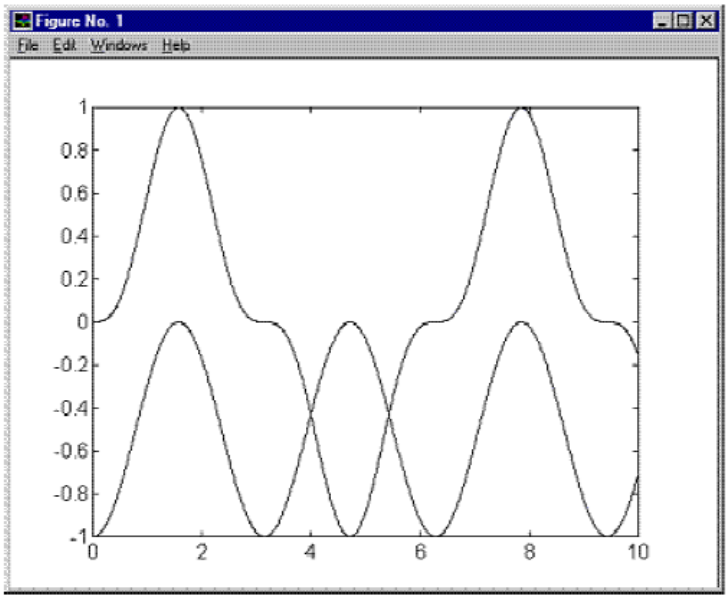
Выразим y из первого и второго уравнений в явном виде, получим:

$y = -\cos^2 x,$

$y = \sin^3 x.$

Построим в одной системе координат графики функций $y(x) = -\cos^2 x$ и $y(x) = \sin^3 x$, координаты точек пересечения которых будут являться решением данной системы уравнений.

```
>> x=-6:0.01:6;
>> y1=-(cos(x)).^2;
>> y2=(sin(x)).^3;
>> plot(x,y1,x,y2)
>>
```



Основные элементы языка MATLAB

Арифметические и логические операторы

Число *арифметических операторов* в MATLAB значительно расширено и включает в себя матричные и арифметические операции. В таблице 1.1 приводится список арифметических операторов.

Таблица 1.1

Функция	Обозначение (синтаксис)
Сложение	**+ (M1+M2)**
Вычитание	**– (M1–M2)**

Матричное умножение	*** (M1*M2)**
Поэлементное умножение массивов	**.* (M1.*M2)**
Возведение матрицы в степень	**^ (M1^ *x*)**
Поэлементное возведение массива в степень	**.^ (M1.^ *x*)**
Деление матриц слева направо	**/ (M1 / M2)**
Поэлементное деление массивов слева направо	**./ (M1 ./ M2)**
Деление матриц справа налево	**\ (M1 \ M2)**
Поэлементное деление массивов справа налево	**.\ (M1 .\ M2)**

В математических выражениях операторы имеют определенный приоритет исполнения. В MATLAB приоритет логических операторов выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше сложения и вычитания. Для повышения приоритета операций нужно использовать круглые скобки. Степень вложения скобок не ограничивается.

Операторы отношения служат для сравнения двух величин, векторов или матриц, все операторы отношения имеют две сравниваемые величины и записываются, как показано в таблице 1.2.

Таблица 1.2

Функция	Оператор (синтаксис)
Равно	**= = (*x* = = *y*)**
Не равно	**~ = (*x* ~ = *y*)**
Меньше	**< (*x* < *y*)**
Больше	**> (*x* > *y*)**
Меньше или равно	**< = (*x* < = *y*)**
Больше или равно	**> = (*x* > = *y*)**

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера и логическое выражение принимает значение 1 (True), если элементы идентичны, и значение 0 (False) в противном случае.

Логические операторы служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов согласно таблице 1.3.

Таблица 1.3

Функция	Оператор (синтаксис)
Логическое И	**&; *and* (*and* (*a, b*))**
Логическое ИЛИ	**| ; *or* (*or* (*a, b*))**
Логическое НЕ	**~ ; *not* (*not* (*a, b*))**
Исключающее ИЛИ	***xor* (*xor* (*a, b*))**
Верно, если все элементы вектора равны нулю	***any* (*any* (a))**
Верно, если все элементы вектора не равны нулю	***all* (*all* (*a*))**

1.3. Элементарные функции

Набор элементарных функций представим их описанием. В тригонометрических функциях углы измеряются в *радианах*.

Таблица 1.4

Функция	Синтаксис
$\lvert x \rvert$ – модуль	*abs*(*x*)
e^x – экспонента	*exp*(*x*)
$\ln x$ – натуральный логарифм	*log*(*x*)
$\log_2 x$ – логарифм по основанию 2	*log*2(*x*)
$\lg x$ – десятичный логарифм	*log*10(*x*)
2^x – 2 в степени *x*	*pow*(*x*)
$\sqrt{x}$ – квадратный корень	*sqrt*(*x*)
arccos x – арккосинус	*acos*(*x*)
arcctg x – арккотангенс	*acot*(*x*)
arccosec x – арккосеканс	*acsc*(*x*)
arcces x – арксеканс	*asec*(*x*)
arcsin x – арксинус	*asin*(*x*)
arctg x – арктангенс	*atan*(*x*)
cos x – косинус	*cos*(*x*)
ctg x – котангенс	*cot*(*x*)
sec x – секанс	*sec*(*x*)
cosec x – косеканс	*csc*(*x*)
sin x – синус	*sin*(*x*)
tg x – тангенс	*tan*(*x*)
arcch x – арккосинус гиперболический	*acosh*(*x*)
arccth x – арккотангенс гиперболический	*acoth*(*x*)
arccosech x – арккосеканс гиперболический	*acsch*(*x*)
arcsech x – арксеканс гиперболический	*asech*(*x*)
Функция	Синтаксис
***arcsh x* – арккосинус гиперболический**	*asinh*(*x*)
arctgh x – арктангенс гиперболический	*atanh*(*x*)
ch x – косинус гиперболический	*cosh*(*x*)
ctgh x – котангенс гиперболический	*coth*(*x*)
cosech x – косеканс гиперболический	*csch*(*x*)
sech x – секанс гиперболический	*sech*(*x*)
sh x – синус гиперболический	*sinh*(*x*)
tgh x – тангенс гиперболический	*tanh*(*x*)

Методика выполнения работы по теме 5 изложена в книге «Е. Л. Еремин, В. В. Еремина, М. С. Капитонова. Математическое и компьютерное моделирование. -Благовещенск.: 2005. – 137 с., ил.»

Тема 6. Расчеты с помощью MATlab

Цель работы – вспомнить и закрепить навыки работы в MATlab.

Методические указания для выполнения работы приведены в лабораторных работах по теме 4, решать их следует средствами MATLAB. Методические указания для проведения приведены в книгах «Е. Л. Еремин, В. В. Еремина, М. С. Капитонова. Математическое и компьютерное моделирование. -Благовещенск.: 2005. – 137 с., ил.», «Половко А. М., Бутусов П. Н. MATLAB для студента. – СПб.: БХВ-Петербург, 2005. – 320 с.: ил.».

7. Перечень программных продуктов, реально используемых в практике деятельности выпускников.

Студенты в специализированной аудитории по подготовке курсовых и дипломных работ имеют доступ на информационные программы MathCAD, пакет MS Office, MATLAB, ГАРАНТ, Консультант+.

8. Комплекты заданий для лабораторных работ изложены в материалах данного УМКД.

9. Комплекты билетов для зачета по дисциплине «Программные комплексы в БЖД».

Билет 1

1. Теоретические основы проектирования программных комплексов: определение и составные части.
2. Организация управления программного комплекса с входным языком командного типа.

Билет 2

1. Применение методо-ориентированных и проблемно-ориентированных ППП в программных комплексах.
2. Метод интерпретации.

Билет 3

1. Интегрированные ППП.
2. Метод компиляции.

Билет 4

1. Данные, используемые в программном комплексе.
2. Деление действий на группы. Конечный автомат.

Билет 5

1. Описание модели.
2. Алгоритм ведущего модуля, управляемого посредством меню.

Билет 6
1. Типы данных.
2. Планирование вычислительного процесса. Постановка задачи.

Билет 7
1. Структура данных.
2. Последовательность вызовов, обрабатывающих модулей.

Билет 8
1. Функциональные связи.
2. Задача оптимального планирования вычислительного процесса.

Билет 9
1. Способы обработки данных.
2. Управление памятью в программном комплексе.

Билет 10
1. Анализ модели предметной области.
2. Проектирование обслуживающих модулей программного комплекса.

Билет 11
1. Группы данных. Входные данные. Выходные данные.
2. Функции обслуживающих модулей.

Билет 12
1. Внешнее управление программным комплексом.
2. Особенности реализации интерфейса с пользователем.

Билет 13
1. Цель применения программного комплекса. Формирование задания на выполнение расчетов.
2. Справочный интерфейс. Интерфейс управления.

Билет 14
1. Виды внешнего управления программным комплексом.
2. Информационный интерфейс.

Билет 15
1. Функции управляющих и обслуживающих модулей программного комплекса.
2. Применение программных комплексов в безопасности жизнедеятельности**.**

Билет 16

1. Понятие оболочки программного комплекса.
2. Универсальные программы расчета загрязнения атмосферы: Атмосфера, Атмосфера-ПДВ, ПДВ-Эколог.

Билет 17

1. Проектирование управляющих модулей программного комплекса.
2. Программы расчета распределения вредных веществ в водных объектах: ПДС-Эколог, Сток.

Билет 18

1. Анализ средств внешнего управления комплексом.
2. Программы расчета объемов выбросов, сбросов и количества твердых отходов различных производств и технологических процессов – Отходы автотранспорта, Отходы котельных, отходы строительства, отходы деревообработки.

Александр Александрович Дрюков,
ассистент кафедры БЖД АмГУ

Программные комплексы в БЖД: УМКД

_____
Изд-во АмГУ. Подписано к печати _____ Формат _____. Усл. печ. л. _____, уч. изд. л. ____. Тираж 100. Заказ _____.
Отпечатано в типографии АмГУ.