ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

Методические рекомендации для самостоятельной работы обучающихся по дисциплине

Б1.Б.19 Информационные технологии

Направление подготовки (специальность) *09.03.01 Информатика и вычислительная техника*
Профиль образовательной программы **"***Автоматизированные системы обработки информации и управления***"**
Форма обучения *заочная*

СОДЕРЖАНИЕ

1. Организация самостоятельной работы.....3
2. Методические рекомендации по выполнению индивидуальных домашних заданий 4
2.1 Темы индивидуальных домашних заданий 4
2.2 Содержание индивидуальных домашних заданий 4
2.3 Порядок выполнения заданий.....4
3. Методические рекомендации по самостоятельному изучению вопросов.....5
4. Методические рекомендации по подготовке к занятиям.....13
4.2 Работа в HTML 13
4.3 Работа с MathCAD. 13
4.4 Решение систем линейных алгебраических уравнений в MathCAD. 14
4.5 СУБД ACCESS.....14

1. ОРГАНИЗАЦИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1.1 Организационно-методические данные дисциплины

№ п.п.	Наименование темы	Общий объем часов по видам самостоятельной работы				
		Подготовка курсового проекта (работы)	подготовка реферата/эссе	индивидуальные домашние задания (ИДЗ)	самостоятельное изучение вопросов (СИВ)	подготовка к занятиям (ПкЗ)
1	2	3	4	5	6	7
1.	**Раздел 1 Введение в информационные технологии**				**10**	**20**
1.1.	**Тема 1** Общие сведения об информационных технологиях					10
1.2.	**Тема 2** Работа в HTML				10	10
2.	**Раздел 2 Основы MathCAD**				**10**	**22**
2.1.	**Тема 3** Работа с MathCAD					10
2.2.	**Тема 4** Решение систем линейных алгебраических уравнений в MathCAD				10	12
6.	**Раздел 3 Базы данных и информационные технологии**			**28**	**12**	**10**
6.1.	**Тема 5** СУБД ACCESS			28	12	10
7.	**Раздел 4 Visual Basic for Applications**			**30**	**12**	**10**
7.1.	**Тема 6** Использование Visual Basic for Applications			30	12	10

2. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ИНДИВИДУАЛЬНЫХ ДОМАШНИХ ЗАДАНИЙ

Индивидуальное домашнее задание выполняется в виде контрольной работы.

2.1 Темы индивидуальных домашних заданий

Работа выполняется по вариантам.

2.2 Содержание индивидуальных домашних заданий

Вариант 1. Строительная организация состоит из нескольких подразделений. В базе данных должны содержаться сведения о:

а) подразделениях строительной организации (подразделение представляется номером подразделения, названием, специализацией);

б) сотрудниках (данными о служащих являются его табельный номер, ФИО, год рождения, должность, подразделение в котором он работает).

Вариант 2. Строительная организация ведет работы на нескольких объектах. В базе данных должны содержаться сведения о:

а) заказчиках (данными о заказчике являются номер заказчика, его наименование, адрес, количество сотрудников);

б) объектах (данными об объекте являются его номер, наименование, сметная стоимость работ, планируемая дата окончания работ, заказчик).

2.3 Порядок выполнения заданий

Вариант 1.

1. Напишите запрос, который увеличивает Количество_ПК во всех подразделениях на 5 шт.
2. Напишите запрос, переводящий сотрудников СМУ–1 в СМУ–2.
3. Напишите запрос, который выводит №_Подразделения, Название и Специализацию из таблицы Подразделение.
4. Напишите запрос, который вывел бы список всех сотрудников Планового отдела.
5. Напишите запрос, который вывел бы таблицу Сотрудник со столбцами в обратном порядке.
6. Напишите запрос, извлекающий из таблицы Сотрудник список подразделений, в которых работают сотрудники. Подразделения не должны повторяться.
7. Напишите запрос, считающий средний возраст сотрудников.
8. Напишите запрос на создание списка, состоящего из ФИО сотрудника и названия его подразделения для всех подразделений, в которых количество компьютеров меньше 10.
9. Напишите запрос на удаление всех сотрудников, работающих в подразделении №23.

Вариант 2.

1. Напишите запрос, который сокращает Количество_сотрудников у всех заказчиков на 5.
2. Напишите запрос, передающий объекты от заказчика ОАО Консул к ООО Корвет.
3. Напишите запрос, который выводит Наименование, Адрес и Количество_сотрудников из таблицы Заказчик.

4. Напишите запрос, который вывел бы список всех объектов заказчика ЗАО Берег.

5. Напишите запрос, который вывел бы таблицу Объект со столбцами в обратном порядке.

6. Напишите запрос, извлекающий из таблицы Объект список заказчиков этих объектов. Заказчики не должны повторяться.

7. Напишите запрос, выводящий наименование и сметную стоимость самого дорогого объекта.

8. Напишите запрос на создание списка, состоящего из Наименования объекта и Наименования его заказчика для всех заказчиков, у которых работает более 100 человек.

9. Напишите запрос на удаление всех объектов заказчика №21.

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ВОПРОСОВ

3.1 Работа в HTML. Создание WEB

Структура html-документа состоит из трех пар тегов:

```
<html>

  <head>
     Заголовок документа
  </head>

  <body>
     Тело документа
  </body>

</html>
```

Теги *<html> </html>* являются контейнером для всех остальных, т.е в них помещаются все остальные. Таким образом, ваш документ должен начинаться с тега *<html>*, а заканчиваться тегом *</html>*.

Сам документ условно разделен на две части - заголовок документа (теги *<head> </head>*) и тело документа (теги*<body> </body>*).

Заголовок документа - тег HEAD и его элементы

Заголовок документа содержит служебную информацию и не влияет на внешний вид документа. Его задачей является предоставление браузеру пользователя или серверу информации о том, как отобразить ваш документ.

Title

Единственным обязательным элементом заголовка документа являются теги *<title> </title>*. Они необходимы, чтобы дать документу название, оно отражается в заголовке окна браузера. Например, если написать следующий код:

```
<html>

  <head>
     <title>Заголовок документа</title>
  </head>
```

```
  <body>
     Тело документа
  </body>

</html>
```

В окне браузера он будет выглядеть так:
* Если вы не знаете, как создать html-страницу, то ознакомьтесь сначала с серией уроков Делаем сайт - первые шаги *

Теоретически название документа может иметь неограниченную длину, на практике рекомендуется ограничиться 60 символами.

Не давайте своим документам безликие названия, типа "Первая страница", во-первых название документа должно характеризовать его содержимое, а во-вторых, содержимое тегов *<title> </title>* играет не последнюю роль при оптимизации и раскрутке сайта.

Base

Одиночный тег *<base>* служит для указания полного URL-адреса документа. Зачем это нужно? Представьте, что блуждая по интернету, вы сохранили какую-нибудь html-страницу себе на компьютер, с тем, чтобы просмотреть ее позже. Все картинки на этой страницы превратятся в красные крестики. Но если вы не отключены от сети, а на странице присутствует тег *<base>*, то браузер будет знать, где искать необходимый файл, найдет его и загрузит картинки.

У этого тега один атрибут *href*, значением которого является адрес страницы. Пример кода:

```
<html>

  <head>
     <title>Структура html</title>
     <base href="http://www.my_site.ru/">
  </head>

  <body>
     Тело документа
  </body>

</html>
```

Link

Одиночный тег *<link>* необходим для подключения внешних файлов. Например, если вы будете использовать каскадную таблицу стилей, то ее удобнее хранить в отдельном файле и подключать этот файл ко всем страницам сайта.

У тега *<link>* несколько атрибутов:

href - указывает URL-адрес подключаемого файла.

rel - указывает на тип отношения данного документа к внешнему (например: rel="stylesheet" указывает, что внешний файл определяет стиль текущего документа).

type - указывает тип и параметры присоединенной таблицы стилей.

Пример кода:

```
<html>

  <head>
     <title>Структура html</title>
     <base href="http://www.my_site.ru/">
     <link rel="stylesheet" type="text/css" href="style.css">
  </head>

  <body>
     Тело документа
  </body>

</html>
```

Meta

Информация в этом теге не имеет никакого отношения к HTML, однако ее использование очень удобно для решения ряда задач:

- указание кодировки страницы, например, для русского текста в кодировке Windows:

```
<meta http-equiv="Content-Type"
    content="text/html;charset=windows-1251">
```

- указание ключевых слов страницы (используется при оптимизации страниц):

```
<meta http-equiv="KEYWORDS"
    content="тег, структура html, заголовок страницы">
```

- указание краткого описания страницы:

```
<meta http-equiv="DESCRIPTION"
    content="Описание структура html и элементов заголовка">
```

Возможны и другие варианты. Рассмотрим параметры тега *<meta>*:

http-equiv - определяет свойство тега (тип контента, ключевые слова, описание и т.д.)

name - используется для дополнительного описания тега, если отсутствует, то считаеся эквивалентным параметру*http-equiv*.

content - значение параметра *http-equiv*.

Пример кода:

```
<html>

  <head>
    <title>Структура html</title>
    <base href="http://www.my_site.ru/">
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta http-equiv="Content-Type"
        content="text/html;charset=windows-1251">
    <meta http-equiv="KEYWORDS"
        content="тег, html, заголовок страницы">
    <meta http-equiv="DESCRIPTION"
        content="Описание элементов заголовка">
  </head>

  <body>
     Тело документа
  </body>

</html>
```

* Пишите теги в одну строчку, здесь они разбиты на две по причине ограничения ширины страницы. *

Script

Теги *<script></script>* используются для подключения внешних файлов скриптов. Это позволяет оптимизировать код страниц. Например, если вы используете функции java script для большинства своих страниц, то поместив эти функции на отдельную страницу - function.js, с помощью тега *<script>* можно указать путь к этой страницы.

Это повышает читабельность кода и ускоряет загрузку страниц. У этого тега несколько параметров:

language - указывает язык написания скрипта, в спецификации HTML 4.0 данный параметр не рекомендуется к употреблению. Вместо него следует указывать параметр *type*.

type - указывает тип MIME для языка.

src - указывает путь к внешнему файлу со скриптами.

Пример кода с относительным адресом скрипта:

```
<html>

  <head>
    <title>Структура html</title>
    <base href="http://www.my_site.ru/">
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta http-equiv="Content-Type"
        content="text/html;charset=windows-1251">
```

```
    <meta http-equiv="KEYWORDS"
        content="тег, html, заголовок страницы">
    <meta http-equiv="DESCRIPTION"
        content="Описание элементов заголовка">
    <script type="text/javascript" src="function.js">
    </script>
  </head>

  <body>
      Тело документа
  </body>

</html>
```

Тело документа - тег BODY

Все, что отображается на web-странице, находится в тегах *<body></body>* . Это текст, картинки и исполняющиеся скрипты, а также теги для оформления всего этого.

Обязательных параметров у тега *<body>* нет, да и применение необязательных параметров тоже не приветствуется. Тем не менее, большинство параметров до сих пор поддерживается разными браузерами. Рассмотрим те, которые пока поддерживаются всеми браузерами:

alink - устанавливает цвет активной ссылки. Текущий цвет ссылки меняется на активный при нажатии на нее.

vlink - устанавливает цвет посещенной ссылки, т.е. той, по которой уже щелкали.

background - указывает на изображение, которое будет использоваться в качестве фонового рисунка. Этот рисунок заполняет собой все видимое пространство окна. Если рисунок меньше окна браузера, то он повторяется, образуя мозаику из одинаковых картинок. На стыке этих картинок возникают видимые переходы. Поэтому к подбору фоновых рисунков следует подходить с большим вниманием.

bgcolor - указывает фоновый цвет документа.

leftmargin - определяет отступ от левого края окна браузера до контента страницы.

rightmargin - определяет отступ от правого края окна браузера до контента страницы.

topmargin - определяет отступ от верхнего края окна браузера до контента страницы.

bottommargin - определяет отступ от нижнего края окна браузера до контента страницы.

text - устанавливает цвет текста для всего документа.

Пример кода:

```
<html>

  <head>
    <title>Тег body в html</title>
```

```
    <base href="http://www.my_site.ru/">
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta http-equiv="Content-Type"
          content="text/html;charset=windows-1251">
    <meta http-equiv="KEYWORDS"
          content="тег, html, заголовок страницы">
    <meta http-equiv="DESCRIPTION"
          content="Описание элементов заголовка">
    <script type="text/javascript" src="function.js">
    </script>
  </head>

  <body bgcolor="khaki" leftmargin="100" topmargin="50"
        rightmargin="50" bottommargin="50" text="gray"
        alink="red" vlink="green">
     Просто текст
     <br>
     <a href="index.html">Ссылка на страницу index.html</a>
     <br>
     <a href="map.html">Ссылка на страницу map.html</a>
  </body>

</html>
```

В окне браузера он будет выглядеть так:

На этом второй урок закончен, мы рассмотрели основную структуру html-документа.

Запомните, любой ваш html-документ должен содержать следующие теги и именно в том порядке, как они указаны:

```
<html>

  <head>
     <title>  </title>
  </head>

  <body>  </body>

</html>
```

Все остальные теги рассмотренные в этом уроке на этом этапе вам не нужны, будете добавлять их позже по мере необходимости.

Все элементы, которые мы будем рассматривать на следующих уроках, будут помещаться внутрь тегов *<body></body>* и их порядок уже не будет иметь принципиального значения.

3.2 Решение систем линейных алгебраических уравнений в MathCAD.MatLAB

Первоначально рассмотрим **СЛАУ** в Mathcad. Для их решения может использоваться блок**given …find()** или специальная функция **lsolve()**. Применение блока **given**

…find()предопределяет необходимость задания начальных значений искомых переменных. Далее после ключевого слова given описывается **СЛАУ** и с помощью **find()** находится решение. Следует указать, что в том случае, когда **СЛАУ** в Mathcad имеет бесконечное множество решений блок **given …find()** дает конкретный результат, что несомненно следует отнести к недостаткам. В случае отсутствия решения будет выдано сообщение "**Matrix is singular. Cannot compute its inversy – Матрица сингулярная. Нельзя вычислить эту инверсию**".

Применение функции **lsolve( )** позволяет избежать этого недостатка.

Функция **lsolve(M,b)**имеет два аргумента. **M** – матрица коэффициентов при неизвестных, **b** – вектор свободных членов. На листинге приведен пример решения **СЛАУ**.

Пример решения **СЛАУ**:

$x := 0 \qquad y := 0$

Given

$x + 3y = 5$

$2 \cdot x + y = 12$

$\text{Find}(x, y) = \begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$

$M := \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \qquad b := \begin{pmatrix} 5 \\ 12 \end{pmatrix}$

$\text{lsolve}(M, b) = \begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$

Пример решения СЛАУ в случае бесконечного множества решений

$z := 0 \qquad p := 0$

Given

$2 \cdot z + 3 \cdot p = 6$

$4 \cdot z + 6 \cdot p = 12$

$\text{Find}(z, p) = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$

$T := \begin{pmatrix} 2 & 3 \\ 4 & 6 \end{pmatrix} \qquad k := \begin{pmatrix} 6 \\ 12 \end{pmatrix}$

$\text{lsolve}(T, k) =$

выдается сообщение "Matrix is singular. Cannot compute its inverse"

Для решения системы нелинейных уравнений используются два блока: **given…find()** и**given…minerr ()**. Так как система нелинейных уравнений может иметь несколько решений, то полученные результаты зависят от начальных значений искомых переменных. В обоих случаях получаются приближенные решения, для которых рекомендуется делать проверку. Обычно в Mathcad требуется, чтобы количество уравнений было равно количеству искомых переменных, но в некоторых случаях, когда с точки зрения классической математики может быть получено точное решение и при

меньшем количестве уравнений, данное условие может быть нарушено. На листинге представлены примеры использования блоков **given…find()** и**given…minerr ()** для решения систем нелинейных уравнений.

$y := 1 \quad x := 1 \quad \text{Given} \quad (x-2)^2 + (y-3)^2 = 0 \quad \text{Find}(x,y) = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$

$z := 0 \quad p := 0 \quad \text{Given} \quad (z-4)^2 + (p-5)^2 = 0 \quad \text{MinErr}(z,p) = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$

$m := 1 \quad n := 0 \quad \text{Given} \quad m^n + n^2 = 10 \quad m^2 + n^m = 2 \quad \text{MinErr}(m,n) = \begin{pmatrix} 0.492 \\ 3.145 \end{pmatrix}$

Проверка $m := 0.492 \quad n := 3.145 \quad m^n + n^2 = 9.998 \quad m^2 + n^m = 1.999$

$p := 1 \quad q := 0 \quad \text{Given} \quad p^q + q^2 = 10 \quad p^2 + q^p = 2 \quad \text{Find}(p,q) = \begin{pmatrix} 0.492 \\ 3.145 \end{pmatrix}$

При изменении начальных условий функция minerr ( ) дает неверный результат, а функция find( ) не находит решения

$m := 0 \quad n := 0 \quad \text{Given} \quad m^n + n^2 = 10 \quad m^2 + n^m = 2 \quad \text{MinErr}(m,n) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

3.3 **СУБД ACCESS.** FoxPro

1. Создание таблиц и форм.

В Access 2007 можно вводить данные непосредственно в таблицу в режиме таблица. Но обычно для ввода данных в БД Access 2007 используют формы (forms). Form ускоряет работу с базой данных. Form в БД - это структурированное интерактивное окно с элементами управления, в котором отображаются поля одной или нескольких таблиц или запросов.

Форму можно использовать для ввода, изменения или отображения данных из таблицы или запроса. В Microsoft Office Access 2007 предусмотрены новые средства, помогающие быстро создавать forms, а также новые типы форм и функциональные возможности.

Формы в БД Access можно создавать с помощью различных средств:

- инструмента Form;
- инструмента Разделенная form;
- инструмента Несколько элементов;
- инструмента Пустая form;
- Мастера form;
- Конструктора form.

2. Создание запросов и отчетов.

В СУБД Access применяются различные типы запросов: на выборку, на обновление, на добавление, на удаление, перекрестный query, выполнение вычислений, создание таблиц. Наиболее распространенным является query на выборку. Применяются два типа запросов: query по образцу (QBE) и query на основе структурированного языка запросов (SQL).

3.4 **Использование Visual Basic for Applications**

Объектно-ориентированные языки программирования

1. Среда VBA.

VBA (Visual Basic for Applications) — это диалект языка Visual Basic, расширяющий возможности Visual Basic и предназначенный для работы с приложениями Microsoft Office и другими приложениями от Microsoft и третьих фирм.

2. Создание макросов.

Алгоритм создания макроса для поставленной задачи:

1. Выберите Сервис/Макрос, Начать запись.

2. В поле Имя макроса введите имя для макроса. Первым символом имени макроса должна быть буква. В имени макроса не допускаются пробелы; в качестве разделителей слов можно использовать знаки подчеркивания.

3. Для того чтобы запускать макрос с помощью сочетания клавиш, введите букву в поле Сочетание клавиш. Допускается использование сочетаний CTRL+ буква (для строчных букв) или CTRL+SHIFT+ буква (для прописных букв), где буква — любая буквенная клавиша на клавиатуре. Не выбирайте стандартного сочетания клавиш, так как выбранное сочетание клавиш подавляет стандартные сочетания клавиш Microsoft Excel на то время с данной книгой.

4. В поле "Сохранить" выберите книгу, в которой требуется сохранить макрос. Сохраните макрос в «Эта книга». Для создания краткого описания макроса, введите необходимый текст в поле Описание. На скриншоте представлен пример заполнения окна диалога «Запись макроса».

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ К ЗАНЯТИЯМ

4.1 Практическое занятие 1 Работа в HTML

При подготовки к занятию необходимо обратить внимание на следующие моменты:

1. Работа в HTML.
2. Создание тегов , парные и не парные теги

4.2 Практическое занятие 2 Работа с MathCAD.

При подготовки к занятию необходимо обратить внимание на следующие моменты:

1. Структура программы
2. Структура программы
3. Ввод и вычисление математических выражений

4.3 Практическое занятие 3 Решение систем линейных алгебраических уравнений в MathCAD

При подготовки к занятию необходимо обратить внимание на следующие моменты:

1. Системы линейных уравнений.
2. Системы нелинейных уравнений.

4.4 Практическое занятие 4 СУБД ACCESS.

При подготовки к занятию необходимо обратить внимание на следующие моменты:

1. Объекты MicrosoftAccess
2. Работа с таблицами