

Евгений Иванов

Лабораторный
практикум



Основы алгоритмизации и программирования

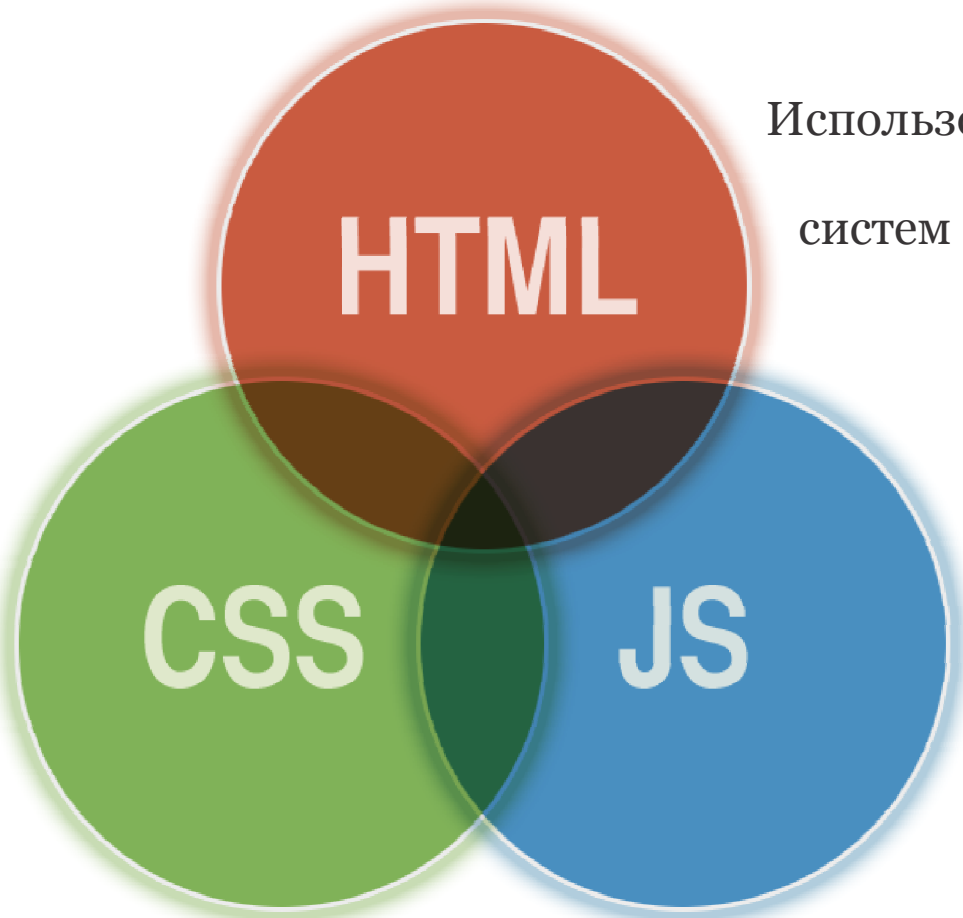
Практическая разработка прикладных программ

Использование популярных
систем программирования

HTML+CSS,

JavaScript,

PHP+MySQL





Министерство образования и науки Республики Марий Эл

ГБПОУ Республики Марий Эл

«Йошкар-Олинский технологический колледж»

Е.С.Иванов

**Лабораторный практикум по дисциплине
«Основы алгоритмизации и
программирования»**

Йошкар-Ола, 2016 г.

Иванов Е.С.

Лабораторный практикум по дисциплине «Основы алгоритмизации и программирования»

После изучения курса и выполнения лабораторных работ вы получите первый опыт программирования на популярных языках программирования HTML, JavaScript, PHP. Узнаете синтаксис языков, основные идеи и механизмы создания программ и попробуете на практике применить полученные знания. Выбор языков обусловлен их широкой распространенностью и востребованностью на рынке. Курс рассчитан на начинающих. Если вы заинтересовались программированием, этот курс поможет познакомиться с основами и не перегрузит вас лишней информацией. Информация курса фундаментальна и подойдет для понимания основных современных языков программирования.

В качестве источника информации использовалась различная справочная, техническая литература и ресурсы сети Интернет.

Содержание

Лабораторная работа №1 «Основы языка JavaScript»	7
• Задание для выполнения	7
• Содержание отчета	7
• Теоретическое обоснование	8
– Первая программа	8
– Числовой тип и автоматическое преобразование в строковый тип	8
– Вычисления	9
– Отладка программы	10
– Переменные. Операция присваивания	10
– Ввод данных	11
– Склеивание строк	11
– Преобразование из строки в число	11
– Логический тип данных	12
Лабораторная работа №2 «Алгоритмы, условные переходы»	14
• Задание для выполнения	14
• Содержание отчета	14
• Теоретическое обоснование	15
– Алгоритмы	15
– Линейный алгоритм	15
– Разветвляющиеся алгоритмы	16
– Вложенные условия и программирование лесенкой	18
– Сложные условия	19
– Сравнение строк	20
– Устройство компьютера с точки зрения программиста	20
– Основные алгоритмические структуры	22
– Графические форматы	25
– Создание схем алгоритмов средствами Microsoft Visio	26
– Создание текстового документа MS Word со схемой алгоритма	37
– Создание схем алгоритмов средствами OpenOffice Draw	40
Лабораторная работа №3 «Основы HTML и CSS»	53
• Задание для выполнения	53
• Содержание отчета	53
• Теоретическое обоснование	54
– HTML+CSS=Сайт	54
– HTML	54
– CSS	55
Лабораторная работа №4 «HTML&CSS»	58
• Задание для выполнения	58
• Содержание отчета	58
• Теоретическое обоснование	59
– Личный сайт	62
– Центральная часть	62
(Подготовка и вставка рисунка)	62
(Заголовок)	63
(Информация о себе)	63
(Ссылки)	63
– Верхняя часть	63
()	64

– Нижняя часть	64
(<Div>).....	64
(Подключение CSS).....	65
Лабораторная работа №5 «JavaScript - Циклы»	68
• Задание для выполнения.....	68
• Содержание отчета	68
• Теоретическое обоснование.....	69
– Вывод без Ok.....	69
– Цикл While	69
(Формат записи цикла While).....	70
(Подсчет суммы арифметической прогрессии)	70
– Цикл For.....	71
(Формат записи цикла For).....	72
– Цикл Do While.....	72
(Блок-схема программы).....	73
(Формат записи цикла Do While)	73
– Веб-страница игры «Угадай число»	74
Лабораторная работа №6 «JavaScript - Функции, DOM»	75
• Задание для выполнения.....	75
• Содержание отчета	75
• Теоретическое обоснование.....	76
– Функции	76
(Параметры функции)	77
(Упрощение логики программы)	77
– Рекурсии	78
(Рекурсивный факториал).....	78
– Объектная модель документа (DOM)	78
– Страница с загадками.....	80
(События (Events)).....	80
– Итоговая программа	81
(Функция CheckAnswer).....	81
(Функция CheckAnswers).....	81
Лабораторная работа №7 «Регистрация на хостинге, знакомство с PHP».....	83
• Задание для выполнения.....	83
• Содержание отчета	83
• Теоретическое обоснование.....	84
– Регистрация на хостинге и публикация сайта	84
Лабораторная работа №8 «Основы PHP»	86
• Задание для выполнения.....	86
• Содержание отчета	86
• Теоретическое обоснование.....	87
– Основы PHP	87
– Установка Веб-сервера XAMPP	87
Лабораторная работа №9 «Шаблон сайта на HTML»	92
• Задание для выполнения.....	92
• Содержание отчета	92
• Теоретическое обоснование.....	93
– Создание шаблона сайта на языке HTML	93

Лабораторная работа №10 «Оформление сайта с помощью CSS»	96
• Задание для выполнения.....	96
• Содержание отчета	96
• Теоретическое обоснование.....	97
– Оформление сайта с помощью таблиц CSS.....	97
Лабораторная работа №11 «Функции и циклы языка PHP».....	101
• Задание для выполнения.....	101
• Содержание отчета	101
• Теоретическое обоснование.....	102
– Функции и циклы языка PHP.....	102
Лабораторная работа №12 «Проектирование блога на языке PHP»	105
• Задание для выполнения.....	105
• Содержание отчета	105
• Теоретическое обоснование.....	106
– Проектирование блога на языке PHP.....	106
Лабораторная работа №13 «Подготовка шаблонов для статей в блоге»	107
• Задание для выполнения.....	107
• Содержание отчета	107
• Теоретическое обоснование.....	108
– Шаблонизация.....	108
Лабораторная работа №14 «MySQL, создание базы данных».....	111
• Задание для выполнения.....	111
• Содержание отчета	111
• Теоретическое обоснование.....	112
– MySQL, создание базы данных.....	112
Лабораторная работа №15 «Взаимодействие PHP и MySQL»	115
• Задание для выполнения.....	115
• Содержание отчета	115
• Теоретическое обоснование.....	116
– Взаимодействие PHP и MySQL	116
Лабораторная работа №16 «Авторизация в панели администратора»	118
• Задание для выполнения.....	118
• Содержание отчета	118
• Теоретическое обоснование.....	119
– Админ панель, авторизация.....	119
Лабораторная работа №17 «Вывод статей в панели администратора»	120
• Задание для выполнения.....	120
• Содержание отчета	120
• Теоретическое обоснование.....	121
– Админ панель, вывод статей	121
Лабораторная работа №18 «Добавление статей в панели администратора»	123
• Задание для выполнения.....	123
• Содержание отчета	123
• Теоретическое обоснование.....	124
– Админ панель, добавление статей.....	124

Лабораторная работа №19 «Редактирование статей в панели администратора»	126
• Задание для выполнения.....	126
• Содержание отчета	126
• Теоретическое обоснование.....	127
– Админ панель, редактирование статей	127
Лабораторная работа №20 «Удаление статей в панели администратора»	130
• Задание для выполнения.....	130
• Содержание отчета	130
• Теоретическое обоснование.....	131
– Админ панель, удаление статей.....	131
Список использованных источников	133

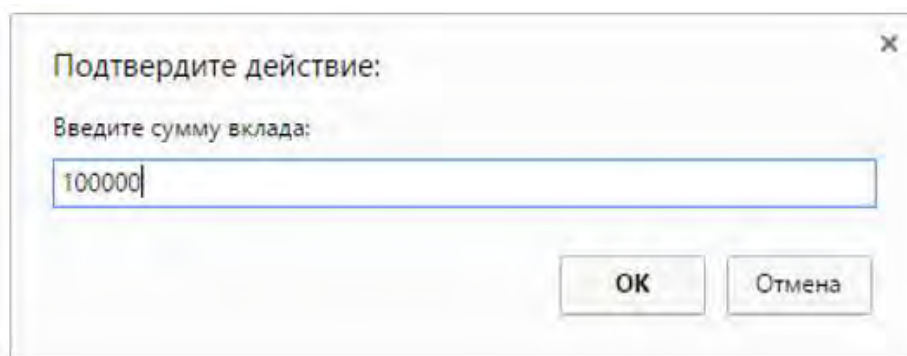
Лабораторная работа №1

Основы JavaScript

Основы языка JavaScript: первая программа; числовой тип и автоматическое преобразование в строковый тип; вычисления; отладка программы; переменные - операция присваивания; ввод данных; склеивание строк; преобразование из строки в число; логический тип данных.

Задание для выполнения

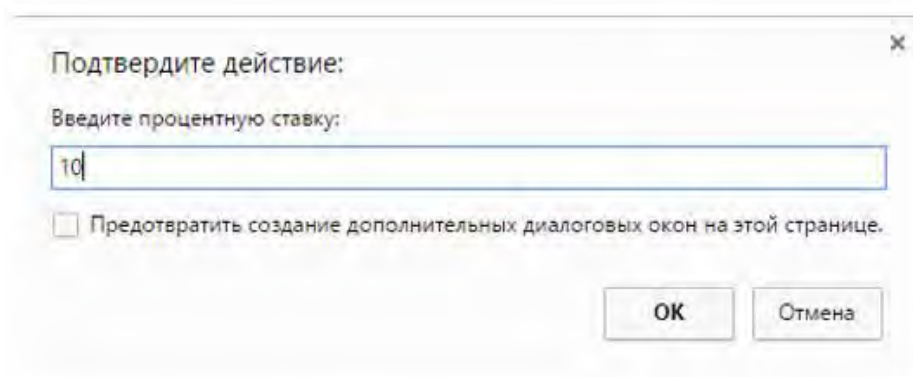
1. Изучите теоретическое обоснование
2. Используя язык программирования JavaScript, разработайте программу для расчета банковских выплат, работающую в соответствии с приведенными рисунками.



Подтвердите действие:

Введите сумму вклада:

OK Отмена



Подтвердите действие:

Введите процентную ставку:

Предотвратить создание дополнительных диалоговых окон на этой странице.

OK Отмена



Подтвердите действие:

Сумма на счете после 1 года = 110000
Сумма на счете после 2 года = 121000
Сумма на счете после 3 года = 133100
Сумма на счете после 4 года = 146410
Сумма на счете после 5 года = 161051

Предотвратить создание дополнительных диалоговых окон на этой странице.

OK

3. Составьте отчет по выполненной работе.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

Первая программа

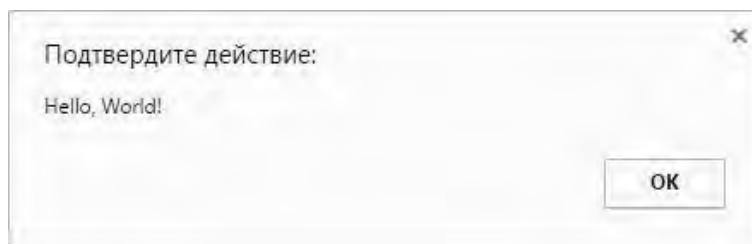
Запустите Блокнот. Наберите в редакторе следующий текст

```
<script>  
    alert("Hello, World!")  
</script>
```

Отступ перед второй строкой можно не делать, на работу программы это не повлияет. Но такое форматирование помогает легче понимать сложные программные коды, поэтому рекомендуется правилами хорошего тона, применяемыми хорошими программистами.

Сохраните файл на Рабочий стол под названием **1.html**. Расширение **.html** необходимо, чтобы данный файл открывался в браузере. Теперь запустите этот файл, щелкнув по нему мышкой два раза.

Если вы все сделали правильно, то у вас запустится ваш браузер и появится следующее окошко.



Вид окна может отличаться в разных версиях браузера. На курсе мы, как правило, используем браузер Chrome, хотя вы можете использовать браузер, к которому вы больше привыкли. Это простая программа знакомит вас с тремя понятиями:

- теги;
- строка;
- команда языка программирования;
- параметры команды.

Теги `<script>` `</script>` не относятся к языку JavaScript. Это указание браузеру, что внутри них заключена программа, которую следует выполнить. Про теги мы поговорим попозже.

Строка. Любая последовательность символов, заключенная в двойные кавычки, является строкой в JavaScript. Строка – это одна из разновидностей данных, с которыми умеет обращаться язык JavaScript. В дальнейшем мы познакомимся с другими типами данных.

Слово **alert** это команда, которая заставляет браузер выводить окошко с кнопкой ОК и текстом, который мы указали в скобках.

Параметры команды. У команд бывают параметры, с помощью которых мы сообщаем, как они должны работать. Описание команды `alert` выглядит так:

```
alert(message:string)
```

message – это подсказка программисту, что `alert` может обработать текст сообщения, а **string** - это подсказка программисту, что это сообщение должно быть строкой. Учтите, что это описание команды `alert`. При использовании команды мы в скобки просто передаем строку, не указывая его тип. В конце строки можно поставить точку с запятой, но она не является обязательной.

Числовой тип и автоматическое преобразование в строковой тип

Теперь давайте изменим нашу программу следующим образом:

```
<script>
  alert(2016)
</script>
```

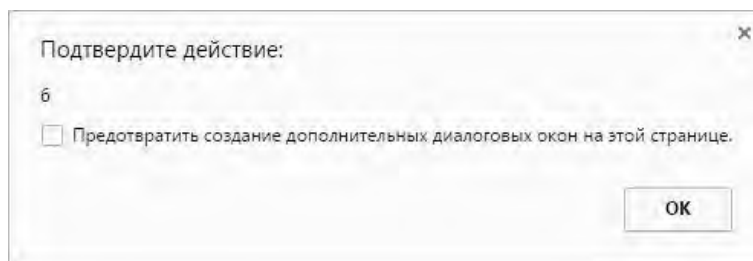
Эта программа выведет на экран число 2016. Но ведь 2016 не является строкой, скажете вы и будете правы. 2016 - это число и относится к числовому типу данных. Тогда почему alert вывел 2016 на экран, хотя в его описании указывается, что он выводит строковой тип данных? Дело в том, что JavaScript автоматически преобразовал 2016 в строковой тип. Чтобы в этом разобраться, нужно изучить, как компьютер хранит в памяти различные данные. Познакомиться с двоичным кодированием. Пока достаточно понять, что при необходимости JavaScript автоматически преобразовывает данные из одного типа данных в другой.

Вычисления

Измените программу, поместив в скобки alert арифметическое выражение:

```
<script>
  alert(2+2*2)
</script>
```

Сохраните и посмотрите результат в браузере



Как вы и ожидали, прежде, чем вывести результат на экран, компьютер вычислил выражение в скобках, преобразовал результат в текст и выполнил команду **alert**. Компьютер достаточно умен, чтобы понимать, какую операцию выполнять в первую очередь. Если нужно изменить приоритет операции, то используйте скобки, например: **alert((2+2)*2)**.

Полезно понять, что для вычисления компьютер анализирует программу и сохраняет числа в оперативной памяти, каждое число в своей ячейке. Для вычисления числа передаются в процессор, где складываются (умножаются, вычитаются, делятся...), и результат помещается обратно в оперативную память. Процесс вычисления 2+2 схематически показан на рисунке ниже.



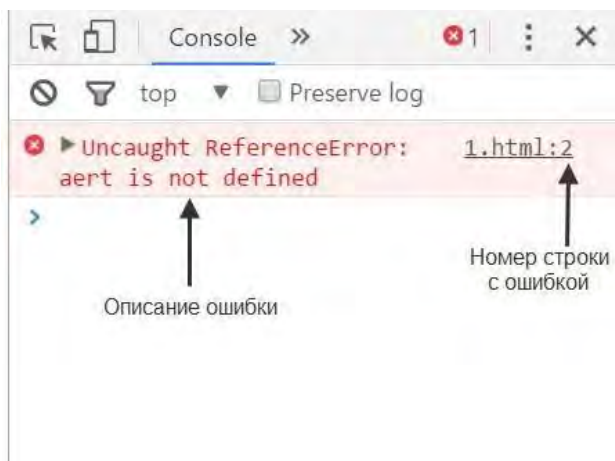
Отладка программы

Удалите в предыдущей программе в команде `alert` какую-нибудь букву. Запустите программу.

```
<script>
  alert(2+2*2)
</script>
```

В этом случае программа не запустится, так как она написана с ошибкой. В браузерах есть специальные инструменты, которые помогают программистам искать ошибки в программах. В браузере Chrome выберите (Дополнительно) -> (Инструменты разработчика) и щелкните на вкладке Console.

Если программа написана с ошибкой, то в консоли выводится сообщение о том, в какой строке допущена ошибка и описание ошибки. Начинающим программистам описание ошибки может и не поможет, но хотя бы увидите, к какой строке нужно присмотреться. Исправьте ошибку, сохраните программу и перезагрузите страницу в браузере.

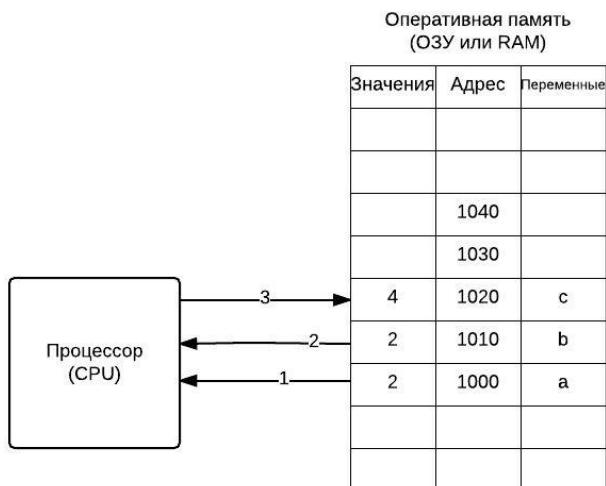


Переменные. Операция присваивания

Наберите и запустите следующую программу

```
<script>
  var a=2;
  var b=2;
  var c=a+b;
  alert(c);
</script>
```

В общем то, ничего особенного не произошло. Программа вывела на экран сумму двух чисел. Но теперь для хранения чисел используются переменные. Переменные нужны для хранения изменяющихся данных.



На самом деле переменные – это именованные адреса ячеек, то есть, обращаясь к переменной **a**, мы обращаемся к ячейке в памяти компьютера, и написав команду **a=2**, мы указываем компьютеру, что хотим положить в ячейку, с которой связана переменная **a**, значение **2**. Первые программисты были настолько суровы, что писали программы без использования переменных.

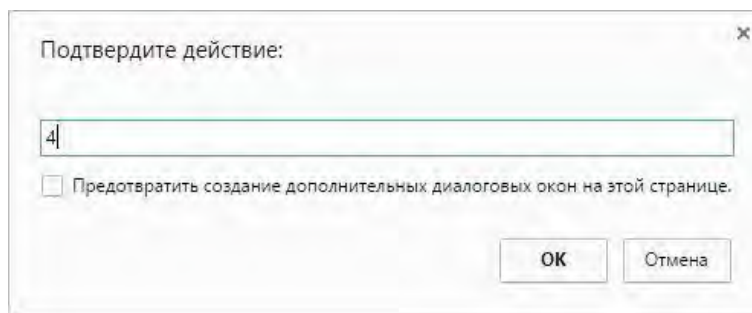
Для изменения значения переменной используется операция **присваивания**, которая в JavaScript обозначается знаком равенство (=). Несмотря на свой незамысловатый вид, это одна из самых важных операций в программировании, поэтому важно понимать, что происходит, когда вы ее используете.

Ввод данных

Ввод данных в программу может производиться разными способами. Это может быть клавиатура, мышь, касание экрана, считывание данных из файла или из базы данных. Но в каждом языке есть команда, с которой начинают изучать ввод данных. В JavaScript это команда **prompt**. Наберите следующую программу и запустите её на выполнение:

```
<script>
  var a;
  a=prompt();
  alert(a);
</script>
```

Эта программа выводит окно со строкой, куда пользователь может вводить данные.



Введённые данные сохраняются в переменной, и теперь мы можем вывести их на экран.

Склеивание строк

Давайте рассмотрим следующий пример:

```
<script>
  var a;
  a=prompt();
  alert(a+" is a good number! ");
</script>
```

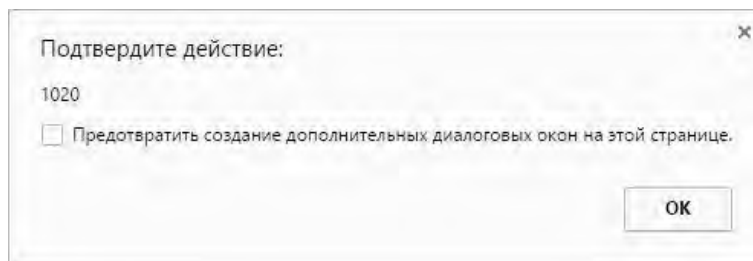
Здесь мы демонстрируем операцию склеивания строк. Для строк знак плюс означает, что должна получиться новая строка, состоящая из нескольких строк, стоящих слева и справа от знака **+**.

Преобразование из строки в число

Мы уже демонстрировали, что, при необходимости, JavaScript переводит число в строку. Что делать, если нужно сделать наоборот? Рассмотрим пример:

```
<script>
  var a=prompt("a:");
  var b=prompt("b:");
  var c=a+b;
  alert(c);
</script>
```

Запустите программу и введите числа 10 и 20. Вместо того, чтобы сложить два числа, программа склеит две строки. В этом нет ничего удивительного, если знать, что результатом выполнения команды `prompt` является строка.



Посмотрим на описание команды `prompt`:

```
function(message:string,value:string):string
```

Описание команды еще называют синтаксисом.

Ничего страшного, что вы пока не понимаете это описание. Обратите внимание, что в конце записи стоит `:string`. Это означает, что `prompt` возвращает строковое значение. (Смысл слова “возвращает” мы разберём подробнее, когда будем изучать функции.) Значит компьютер принял от `prompt` строчку и поступил правильно, когда склеил две строки, вместо того, чтобы сложить числа. Для того, чтобы он записал в переменные числа, а не строки, нужно указать ему на это. Самый простой способ - поставить перед `prompt` символ `+`.

```
<script>
  var a=+prompt("a:");
  var b=+prompt("b:");
  var c=a+b;
  alert(c);
</script>
```

Другой способ преобразования из строки в число - использование функции `parseInt`

```
<script>
  var a=parseInt(prompt("a:"));
  var b=parseInt(prompt("b:"));
  var c=a+b;
  alert(c);
</script>
```

Логический тип данных

Рассмотрим программу.

```
<script>
  var a=2*2==4;
  alert(a);
</script>
```

Эта программа при запуске выведет `true`, что в переводе с английского означает “истина”. Здесь мы с вами знакомимся с еще одним типом данных - логическим. В переменную `a` поместился результат операции **отношения** равенство (`==`). Сначала посчиталась левая часть, операция `2*2`,

и сравнилась с правой частью 4. Так как результаты равны, результат операции отношения “истина”. Результат операции присвоился в переменную **a**.

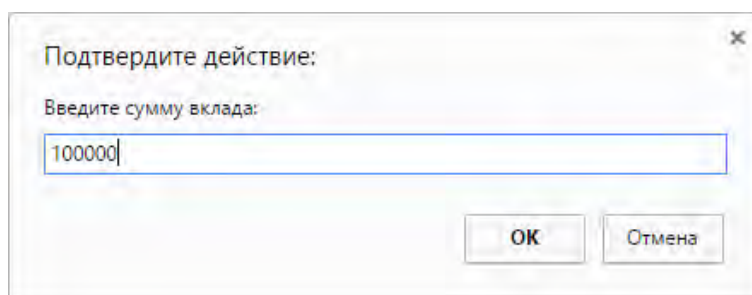
Логический тип данных может принимать всего два значения: *истина* и *ложь*. В JavaScript используется специальные значения для их обозначения: *true* и *false*.

В JavaScript существуют следующие операции отношения:

- == - равенство
- < - меньше
- > - больше
- >= - больше или равно
- <= - меньше или равно
- != - не равно

Задание для выполнения

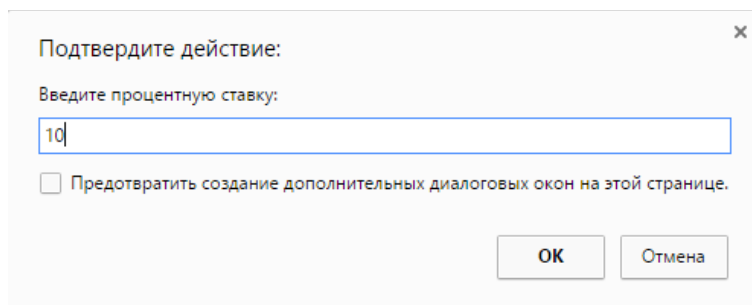
Используя язык программирования JavaScript, разработать программу для расчета банковских выплат, работающую в соответствии с приведенными рисунками.



Подтвердите действие:

Введите сумму вклада:

OK Отмена

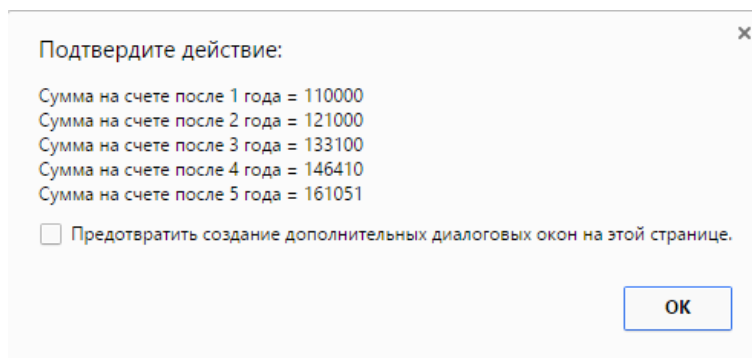


Подтвердите действие:

Введите процентную ставку:

Предотвратить создание дополнительных диалоговых окон на этой странице.

OK Отмена



Подтвердите действие:

Сумма на счете после 1 года = 110000
Сумма на счете после 2 года = 121000
Сумма на счете после 3 года = 133100
Сумма на счете после 4 года = 146410
Сумма на счете после 5 года = 161051

Предотвратить создание дополнительных диалоговых окон на этой странице.

OK

Лабораторная работа №2

Алгоритмы, условные переходы

Алгоритмы: линейный алгоритм, разветвляющиеся алгоритмы, вложенные условия и программирование лесенкой, сложные условия, сравнение строк.

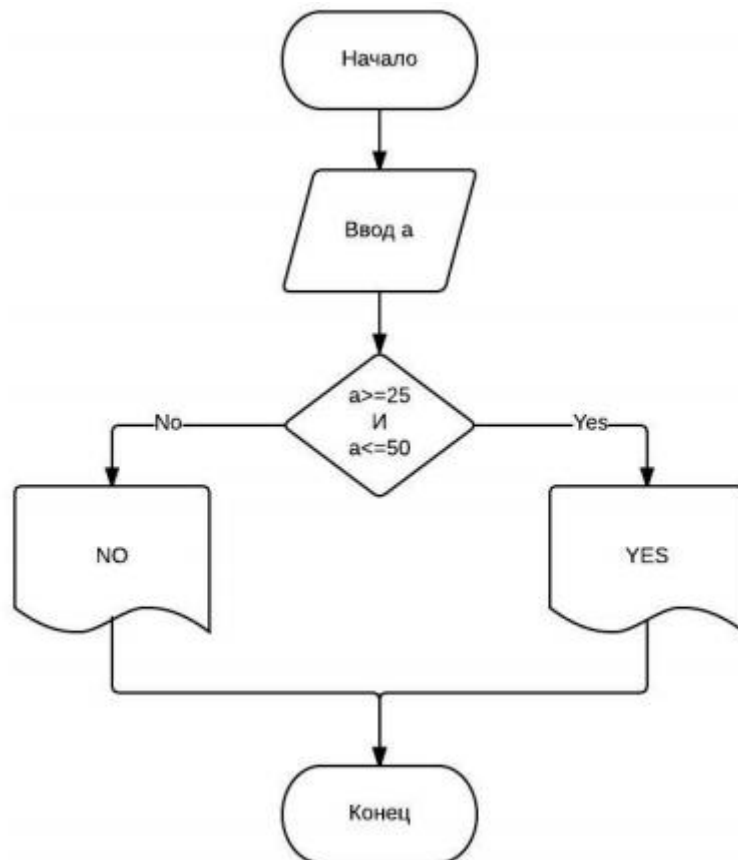
Устройство компьютера с точки зрения программиста.

Задание для выполнения

1. Изучите теоретическое обоснование
2. Используя программу Microsoft Visio создайте и оформите алгоритм решения следующей задачи:
 - программа выводит на экран текст загадки;
 - пользователь вводит ответ;
 - программа сообщает пользователю ответ верен или нет;
 - далее следует ввод еще двух загадок;
 - программа выводит количество правильных ответов и сообщает о том, что игра окончена.
3. Составьте отчет по выполненной работе.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Графическое изображение созданного алгоритма по приведенному образцу



Алгоритмы

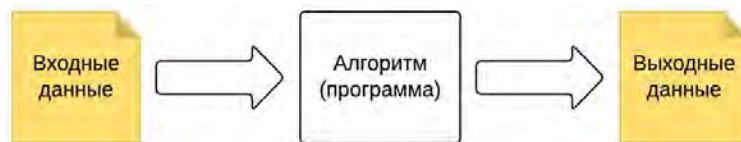
В программистском сообществе постоянно идут споры: нужны алгоритмы или нет. Точнее, обычно все признают, что алгоритмы важны, но вот стоит ли записывать их или можно сразу писать программу - это вызывает бурные дискуссии. Многие "профессиональные" программисты высокомерно заявляют, что они пишут свои программы вообще не используя алгоритмы или, точнее, не записывая их на бумаге. Не будет с ними спорить. Мы считаем, что если изучать язык без алгоритмов, то программист в будущем будет строить программы, используя конструкции первого языка. Вообще-то, вряд ли от этого получится избавиться и при изучении алгоритмов, но, все-таки, написание алгоритма позволяет мыслить более широко, давая возможность сосредоточиться на решении задачи, а не на реализации её на конкретном языке.

Слово "алгоритм" произошло от имени Мухаммеда ал-Хорезми, средневекового ученого, написавшего трактат о вычислениях с использованием десятичной системы счисления.

Алгоритм - это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.

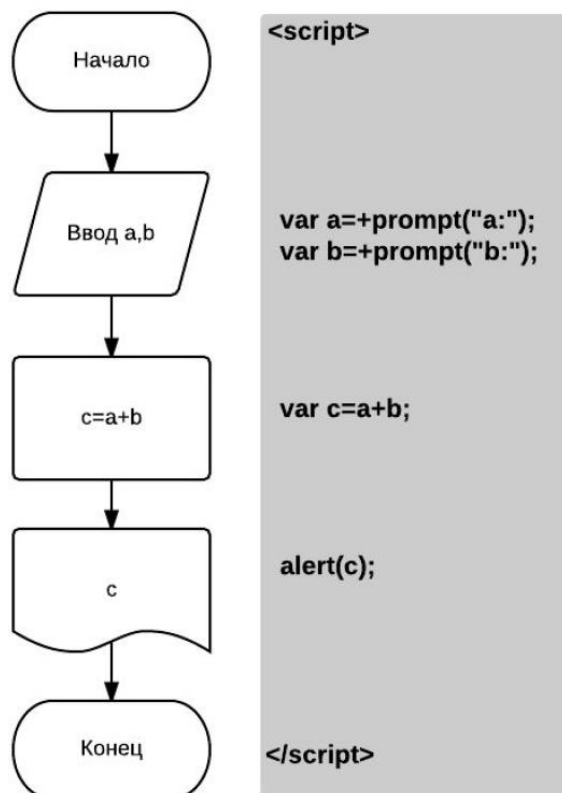
Алгоритм получает на вход некоторый дискретный входной объект (например, набор чисел или слово) и обрабатывает входной объект по шагам (дискретно), строя промежуточные дискретные объекты. Этот процесс может закончиться или не закончиться. Если процесс выполнения алгоритма заканчивается, то объект, полученный на последнем шаге работы, является результатом работы алгоритма при данном входе. Если процесс выполнения не заканчивается, говорят, что алгоритм зациклился. В этом случае результат его работы не определен.

Действительно, операции над десятичными числами, хоть мы об этом и не задумываемся, происходят по определенному, заранее заданному, порядку действий, и, если все действия произвести правильно, то это приведет к определенному результату.



Для описания алгоритма любой сложности необходимо и достаточно иметь всего три алгоритмические конструкции: следование (линейный алгоритм), ветвление и цикл.

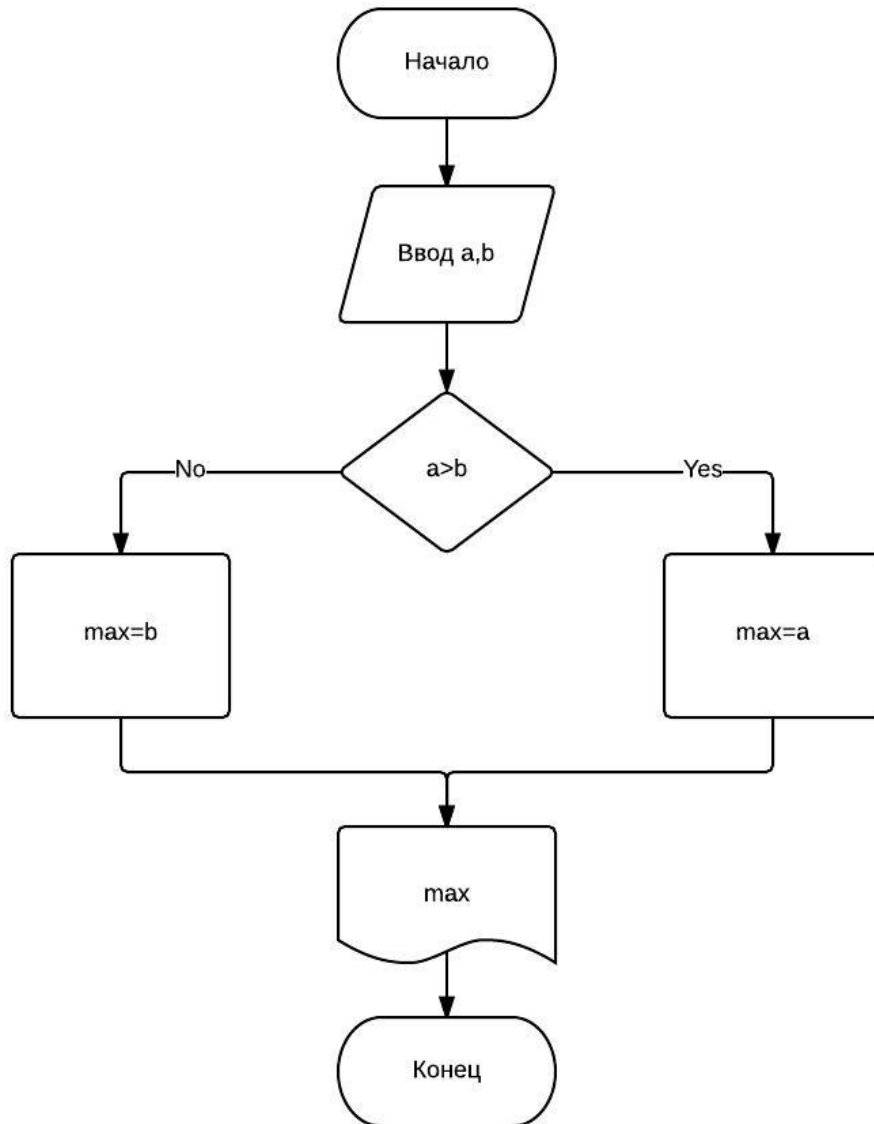
Линейный алгоритм



Разветвляющиеся алгоритмы

Решим задачу нахождения наибольшего из двух чисел.

Идея решения. Вводим два числа. Сравниваем числа, если первое число больше второго, то запоминаем первое число, иначе, запоминаем второе число.



Ветвление в JavaScript реализовано условным оператором

```
if (условие)
{
    {что делать, если условие верно}
}
else
{
    {что делать, если условие неверно}
}
```

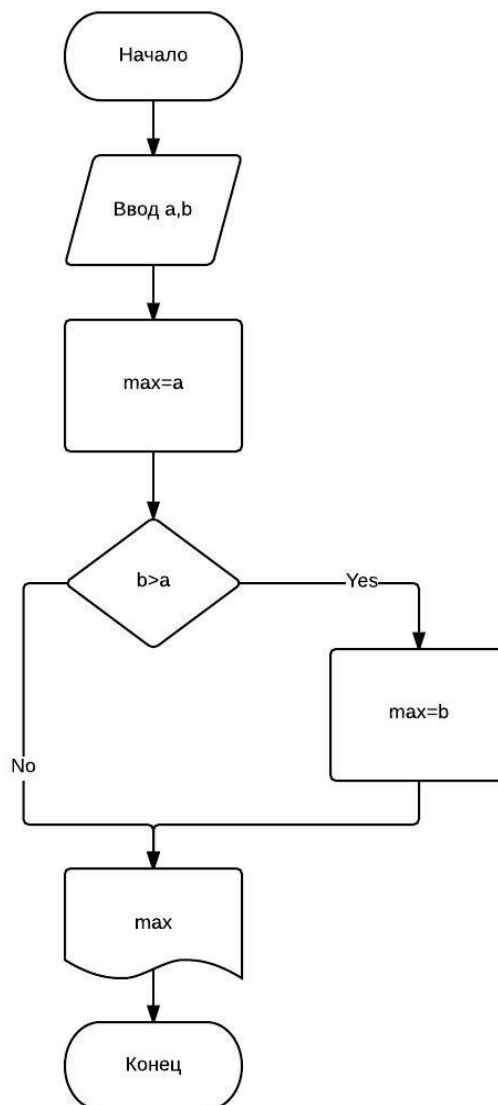
Особенности:

- вторая часть (else) может отсутствовать (не полная форма условного оператора);
- если в блоке один оператор, можно скобки { и } не ставить.

Реализация алгоритма на JavaScript

```
<script>
  var a = +prompt("a:");
  var b = +prompt("b:");
  var max;
  if (a > b)
  {
    max = a;
  }
  else
  {
    max = b;
  }
  alert (max);
</script>
```

Решим задачу нахождения наибольшего числа, используя неполную форму условного оператора



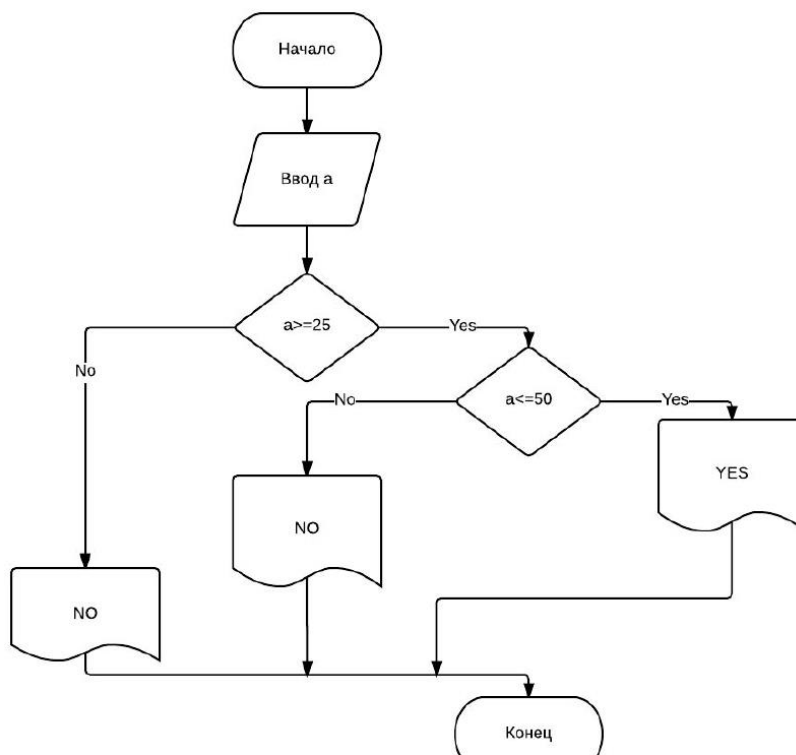
```
<script>
var a = +prompt("a:");
var b = +prompt("b:");
var max = a;
if (b > a)
    max=b;
alert(max);
</script>
```

Вложенные условия и программирование “лесенкой”

Задача. Фирма набирает сотрудников от 25 до 50 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «YES» или «NO»).

Особенность: надо проверить, выполняются ли два условия одновременно.

Можно ли решить задачу уже известными методами? Да, просто нужно будет вложить одно условие в другое.



```
<script>

var a=+prompt("a:");

if (a>=25)
{
    if (a<=50)
    {
        alert("YES");
    }
    else
    {
        alert("NO");
    }
}
else
{
    alert("NO");
}

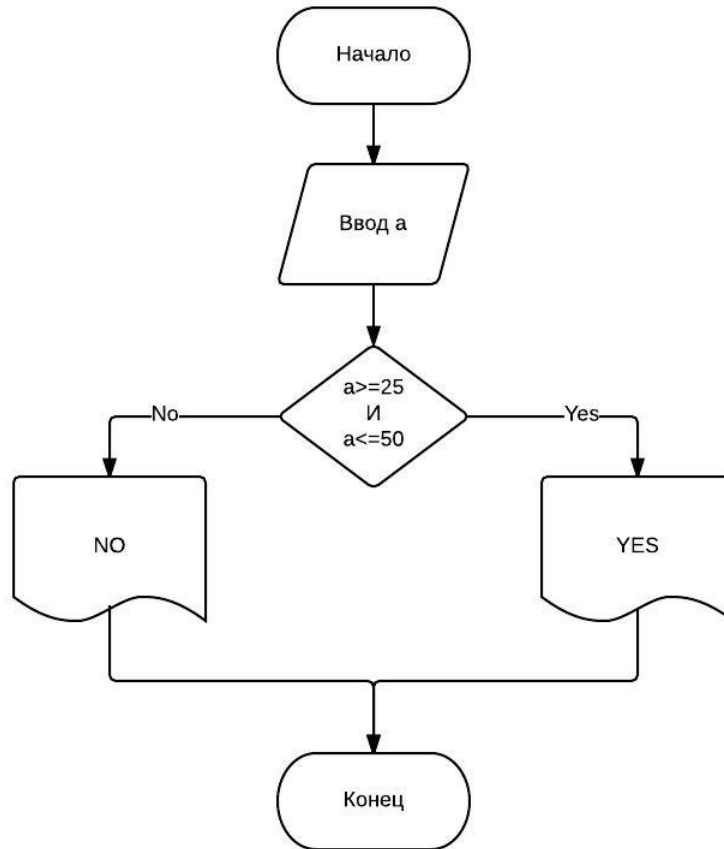
</script>
```

Эта конструкция называется “вложенное условие”. Вложенные условия часто используются программистами, когда одним условием не обойтись. Здесь условие $a \leq 50$ будет выполняться, только если истинно условие $a \geq 25$. Обратите внимание, как написан код программы - “лесенкой”. Мы и раньше использовали такой стиль, немного отодвигая некоторые строчки правее от края. Но это первая программа, которая демонстрирует для чего программисты используют такой стиль. “Лесенка” помогает увидеть вложенность блоков кода друг в друга. Чем сильнее отступ, тем больше вложенность.

Благодаря “лесенке” код становится более читабельным. Поначалу вам может быть не привычно писать, используя такой стиль, но, чем более сложные программы вам придется писать, тем более очевидным будет использование такого стиля.

Сложные условия

Согласитесь, что было бы проще предыдущую задачу решить, если бы у нас была возможность поместить оба условия в одно, как на блок-схеме:



Здесь условие состоит из двух операций сравнения, объединённых логической операцией **И**. В языках программирования условия, которые состоят из нескольких отношений, объединённых логическими операциями, называются сложными условиями.

Рассмотрим логические операции в порядке приоритетов выполнения:

- **!** - Отрицание - если отношение истинно, то отрицание делает его ложным и наоборот;
- **&&** - И - сложное условие истинно, когда оба отношения истинны;
- **||** - ИЛИ - условие истинно, когда истинно хотя бы одно из условий;
- **^** - Исключающее ИЛИ - условие истинно, когда истинно хотя бы одно из условий, но не оба вместе.

Таблица истинности (0-ложь, 1- истина)

A	B	A && B	A B	A ^ B	!A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Тогда программа, использующая сложное условие, будет выглядеть так:

```
<script>
  var a=prompt("a:");
  if (a>=25 && a<=50)
  {
    alert("YES");
  }
  else
  {
    alert("NO");
  }
</script>
```

Сравнение строк

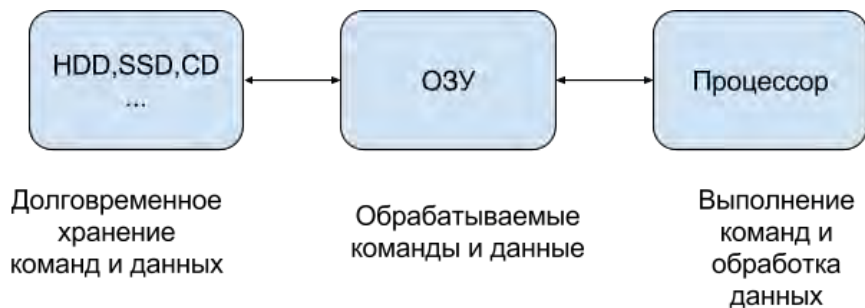
Часто нужно получить от пользователя ответ на вопрос. В примере показано, как можно сравнить строковые значения.

```
<script>
  var answer=prompt("Are you sure?");
  if (answer=="Yes" || answer=="yes" || answer=="YES")
  {
    alert("Ok. Do it!");
  }
  else
  {
    alert("May be later...");
  }
</script>
```

Устройство компьютера с точки зрения программиста

Современные компьютеры, если досконально разбираться в их устройстве, весьма сложны, и чтобы понять, как он функционирует, может потребоваться прочитать не одну книгу. Но откроем вам страшную тайну: программисту, даже профессиональному, не обязательно очень подробно знать его устройство.

Но общую схему функционирования программист, конечно, знать обязан. Она не сложная.



Необходимо познакомиться и понять взаимосвязь всего трех элементов компьютера.

1. Процессор (CPU)
2. Оперативная память (ОЗУ, RAM)
3. Жесткий диск (HDD, SSD)

Процессор

Процессор - это устройство, которое обрабатывает данные и управляет остальными устройствами компьютера. Процессор обладает следующими характеристиками:

- Частота;
- Разрядность;
- Объем кэш-памяти;
- Многоядерность.

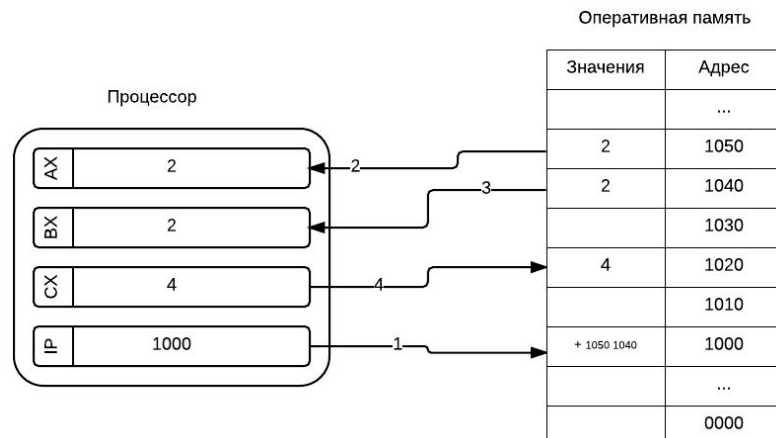
Чтобы понять, на что влияют эти характеристики, сравним процессор с мельницей, которая перемалывает зерно в муку. Тогда частота - это скорость вращения жёрнова, который молит муку. Разрядность - это ширина мола. Чем больше частота и разрядность, тем больше муки можно перемолоть. Или, в нашем случае, данных. Тогда объем кэш-памяти - мешки с зерном, которые лежат рядом с валом, за которыми не нужно бегать в амбар (в ОЗУ). Тогда многоядерность - это мельница с двумя молами для перемалывания.

ОЗУ

В обиходе "оперативка" - это микросхемы, на которых хранятся данные и команды, предназначенные для обработки. Оперативная память так устроена, что работает намного быстрее (в сотни тысяч раз) жесткого диска, но для хранения данных в оперативной памяти требуется электричество. Если питание пропадет даже на доли секунды, данные в оперативной памяти стираются.

Процессор и ОЗУ

Данные и команды, которые обрабатываются в текущий момент, времени хранятся в ОЗУ. Из ОЗУ данные и команды загружаются в процессор, где происходит обработка данных в соответствии с командами. В процессоре присутствуют собственные ячейки памяти, называемые регистрами. Все вычисления производятся в регистрах процессора. Это самая быстрая часть компьютера.



В процессоре существует специальный регистр команд IP. Он хранит в себе адрес ячейки, в которой записана команда, которую необходимо выполнить.

Жесткий диск

Жесткий диск, а также "флешки", CD, DVD и другие относят к внешним запоминающим устройствам. Это связано с тем, что они играют второстепенную роль в обработке информации. То есть непосредственно в обработке информации эти устройства не участвуют, но при этом они выполняют другую важную роль. Их устройство позволяет им хранить информацию долговременно без источника питания.

Задание для выполнения

Используя программу Microsoft Visio создать и оформить алгоритм решения следующей задачи:

- программа выводит на экран текст загадки;
- пользователь вводит ответ;
- программа сообщает пользователю ответ верен или нет;
- далее следует ввод еще двух загадок;
- программа выводит количество правильных ответов и сообщает о том, что игра окончена.

1. Введение

Данная лабораторная работа предполагает практическое изучение способов изображения схем алгоритмов с использованием Microsoft Visio 2003 и OpenOffice Draw. Рассматривается создание фрагмента элементарного алгоритма.

Цель работы – приобретение практических навыков создания текстовых документов, содержащих схемы алгоритмов, с использованием указанных инструментов.

1.1 Понятие алгоритма. Основные алгоритмические структуры

Алгоритмом называют *формально описанную* последовательность действий, которые необходимо выполнить для получения требуемого результата.

Различают последовательности действий (вычислений) линейной, разветвленной и циклической структуры.

Линейная структура процесса вычислений предполагает, что для получения результата необходимо выполнить некоторые операции в определенной последовательности. Например, для определения площади треугольника по формуле Герона необходимо сначала определить полупериметр треугольника, а затем по формуле рассчитать его площадь.

Разветвленная структура процесса вычислений предполагает, что конкретная последовательность операций зависит от значений одного или нескольких параметров. Например, если дискриминант квадратного уравнения не отрицателен, то уравнение имеет два корня, а если отрицателен, то действительных корней оно не имеет.

Циклическая структура процесса вычислений предполагает, что для получения результата некоторые действия необходимо выполнить несколько раз. Например, для того, чтобы получить таблицу значений функции на заданном интервале изменения аргумента с заданным шагом, необходимо соответствующее количество раз определить следующее значение аргумента и посчитать для него значение функции.

Процессы вычислений циклической структуры в свою очередь делятся на три группы:

- *счетные циклы* или *циклы с заданным количеством повторений* – циклические процессы, для которых количество повторений известно;
- *итерационные циклы* – циклические процессы, завершающиеся по достижении или нарушении некоторых условий;
- *поисковые циклы* – циклические процессы поиска некоторой информации в таблицах, выход из которых происходит при нахождении необходимой информации или по завершению процесса поиска, если необходимая информация не найдена.

1.2 Схемы алгоритма

Формальное описание алгоритмов осуществляют с использованием схем алгоритмов и псевдокодов. На изображение *схем алгоритмов* существует ГОСТ 19.701–90, согласно которому каждой группе действий ставится в соответствие блок особой формы. Некоторые, часто используемые обозначения приведены в таблице 1.

Таблица 1


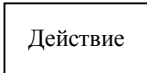




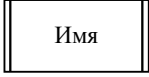


Название блока	Обозначение	Назначение блока
1. Терминатор		Начало, завершение программы или подпрограммы
2. Процесс		Обработка данных (вычисления, пересылки и т.п.)
3. Данные		Операции ввода-вывода

Таблица 1 (окончание)

Название блока	Обозначение	Назначение блока
4. Решение		Ветвления, выбор, итерационные и поисковые циклы
5. Подготовка		Счетные циклы
6. Граница цикла		Любые циклы
7. Предопределенный процесс		Вызов процедур
8. Соединитель		Маркировка разрывов линий
9. Комментарий		Пояснения к операциям

При разработке алгоритма каждое действие обозначают соответствующим блоком, показывая их последовательность линиями, идущими слева направо и сверху вниз. Для удобства чтения схемы желательно, чтобы линия входила в блок сверху, а выходила снизу. Если направление линии отлично от стандартного, то в конце линии ставится стрелка, в противном случае ее можно не ставить.

Если схема алгоритма не умещается на листе, то используют соединители. При переходе на другой лист или получении управления с другого листа в комментарии указывается номер листа, например, «с листа 3», «на лист 1».

Доказано, что для записи любого, сколь угодно сложного алгоритма достаточно трех базовых управляющих структур:

следование – обозначает последовательное выполнение действий (см. рисунок 1.1, а);

ветвление – соответствует выбору одного из двух вариантов действий (см. рисунок 1.1, б);

цикл-пока – определяет повторение действий, пока не будет нарушено некоторое условие, выполнение которого проверяется в начале цикла (см. рисунок 1.1, в).

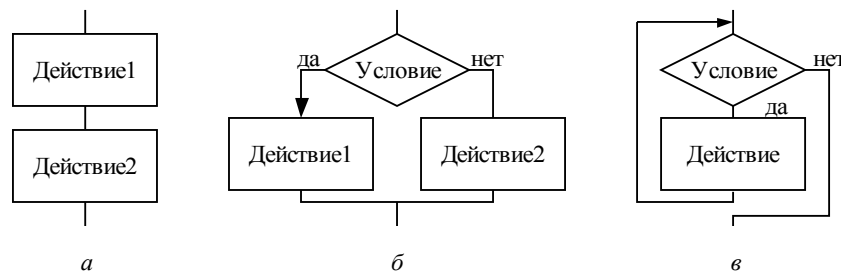


Рисунок 1.1 – Базовые алгоритмические структуры: следование (а), ветвление (б) и цикл-пока (в)

Помимо базовых структур используют еще три дополнительные структуры, производные от базовых:

- *выбор* – обозначающий выбор одного варианта из нескольких в зависимости от значения некоторой величины (см. рисунок 1.2, а);

- *цикл-до* – обозначающий повторение некоторых действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле (см. рисунок 2, в);

- *цикл с заданным числом повторений (счетный цикл)* – обозначающий повторение некоторых действий указанное количество раз (см. рисунок 1.2, д).

На рисунке 1.2 (б, г и е) показано, как каждая из дополнительных структур может быть реализована через базовые структуры.

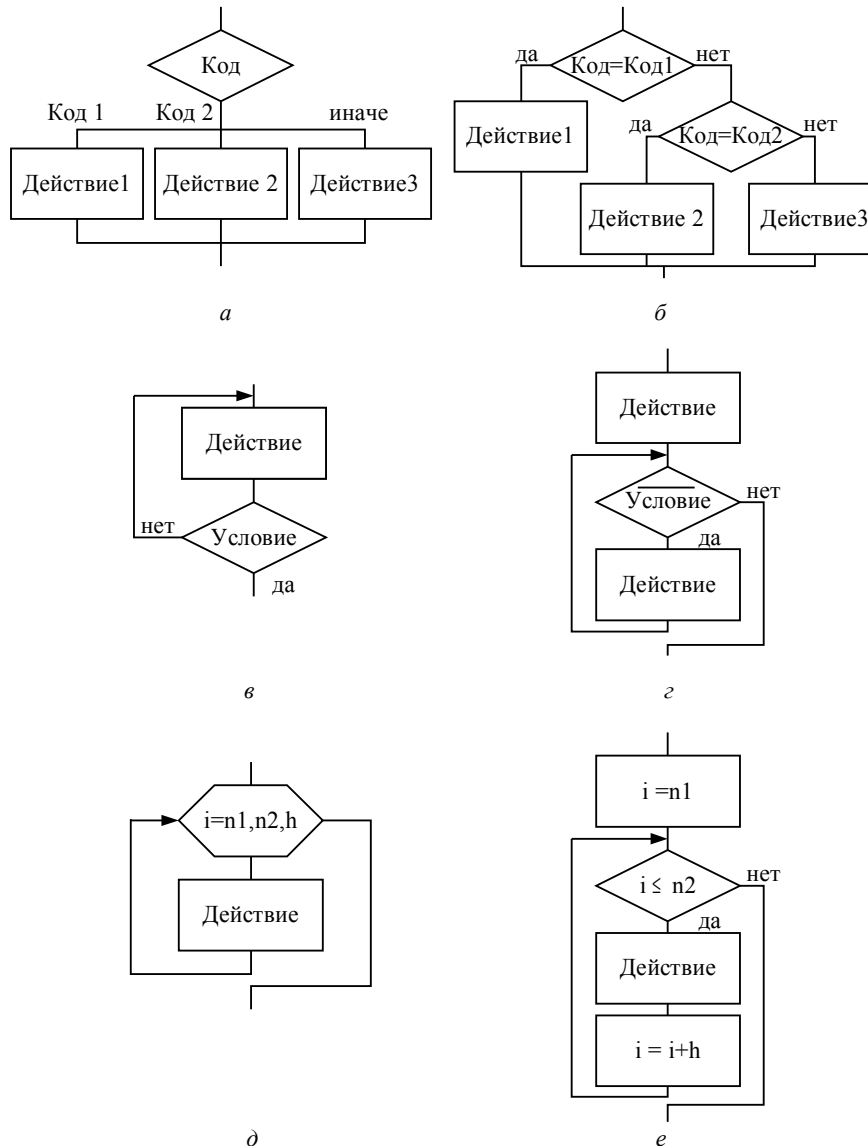


Рисунок 1.2 – Дополнительные структуры и их реализация через базовые структуры: выбор (а-б), цикл-до (в-г) и цикл с заданным числом повторений (д-е)

Перечисленные структуры были положены в основу *структурного* программирования – технологии, которая представляет собой набор рекомендаций по уменьшению количества ошибок в программах. В том случае, если в схеме алгоритма отсутствуют другие варианты передачи управления, алгоритм называют *структурным*, подчеркивая, что он построен с учетом рекомендаций структурного программирования.

1.3 Графические форматы

Графический формат - способ записи графической информации, предназначенный для хранения изображений. Различают растровые и векторные форматы.

Основная область применения растровых форматов — фотографии, небольшие рисунки, сложные цветовые схемы и пр. Характеризуются тем, что изображение формируется в виде матрицы точек (пикселей), каждая из которых содержит информацию о цвете. Изображение может быть любой степени сложности, однако при этом существуют ограничения на отображение, связанное с тем, что четкость изображения зависит от разрешения рисунка и устройства отображения (см. рисунок 1.3). К растровым форматам относятся bmp, jpg, gif, tiff, png.

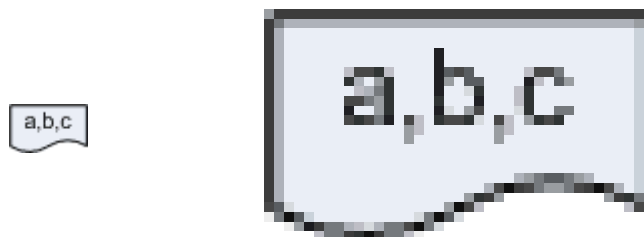


Рисунок 1.3 – Примеры одинаковых растровых рисунков в исходном и увеличенном размерах

Векторные форматы по сути являются программами для формирования растрового изображения с помощью графических примитивов — точки, линии, дуги, шрифт и пр., включая необходимые параметры, например координаты относительно границ изображения, цвет, текст. Основным достоинством векторных форматов является то, что качество формируемого изображения не зависит от разрешения устройства отображения. Несмотря на то, что такие устройства, как видеомонитор, принтер обеспечивают только вывод изображения в растровом формате, изображение, сформированное на основе векторного формата будет иметь в точности такое разрешение, которое необходимо (см. рисунок 1.4). Следует заметить, что растровое изображение может являться графическим примитивом для некоторых векторных форматов (например нанесение надписи поверх фона). К векторным форматам относятся eps, ps, wmf, emf, svg.

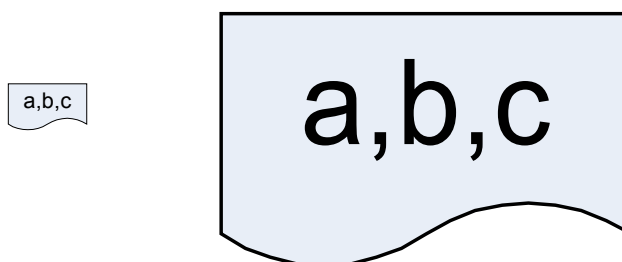


Рисунок 1.4 – Примеры одинаковых векторных рисунков в исходном и увеличенном размерах

2 Создание схем алгоритмов средствами Microsoft Visio 2003

Запускаем Microsoft Visio, используя меню Windows Пуск/Программы/Microsoft Office/ Microsoft Office Visio, и выполняем создание нового файла по типу диаграммы (рисунок 2.1).

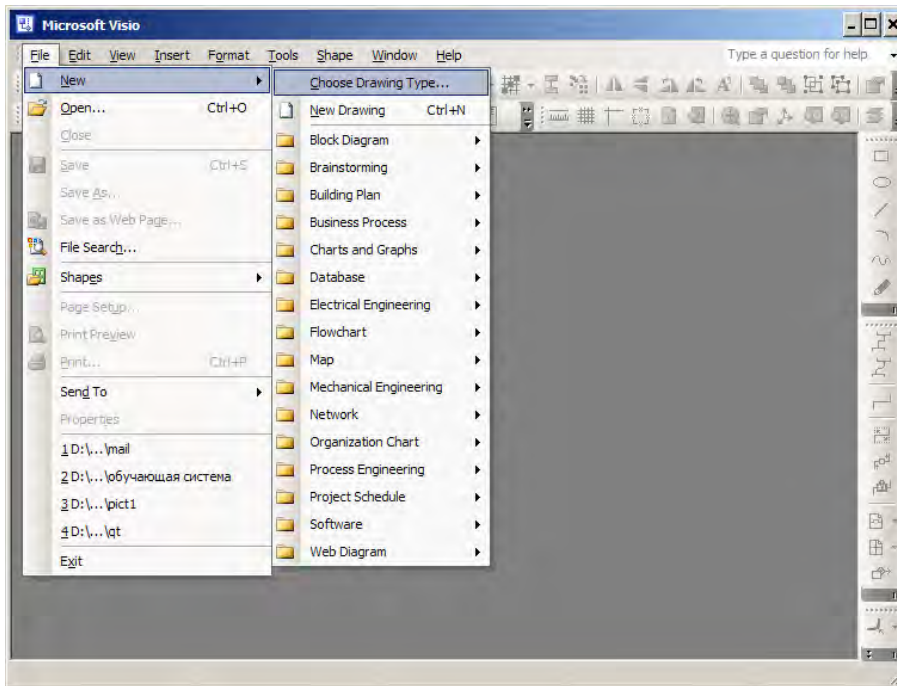


Рисунок 2.1 – Выбор типа создаваемой диаграммы

В открывшемся списке выбираем категорию Flowchart и шаблон SDL Diagrams (рисунок 2.2) или шаблон Basic Flowchart (в этом случае внешний вид элементов будет несколько отличаться от приведенных здесь).

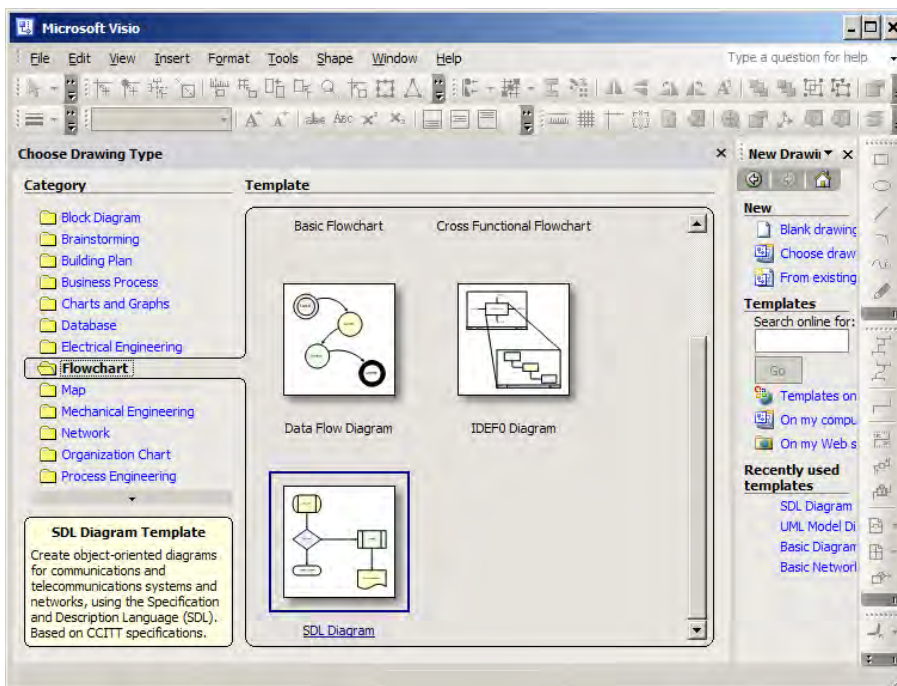


Рисунок 2.2 – Выбор типа диаграммы Flowchart/SDL Diagrams

Из шаблонов “SDL Diagrams Shapes” «перетаскиваем» мышью необходимые элементы на лист в рабочей области справа (рисунок 2.3). Примечание: дополнительные шаблоны элементов доступны через меню File/Shapes/Flowcharts.

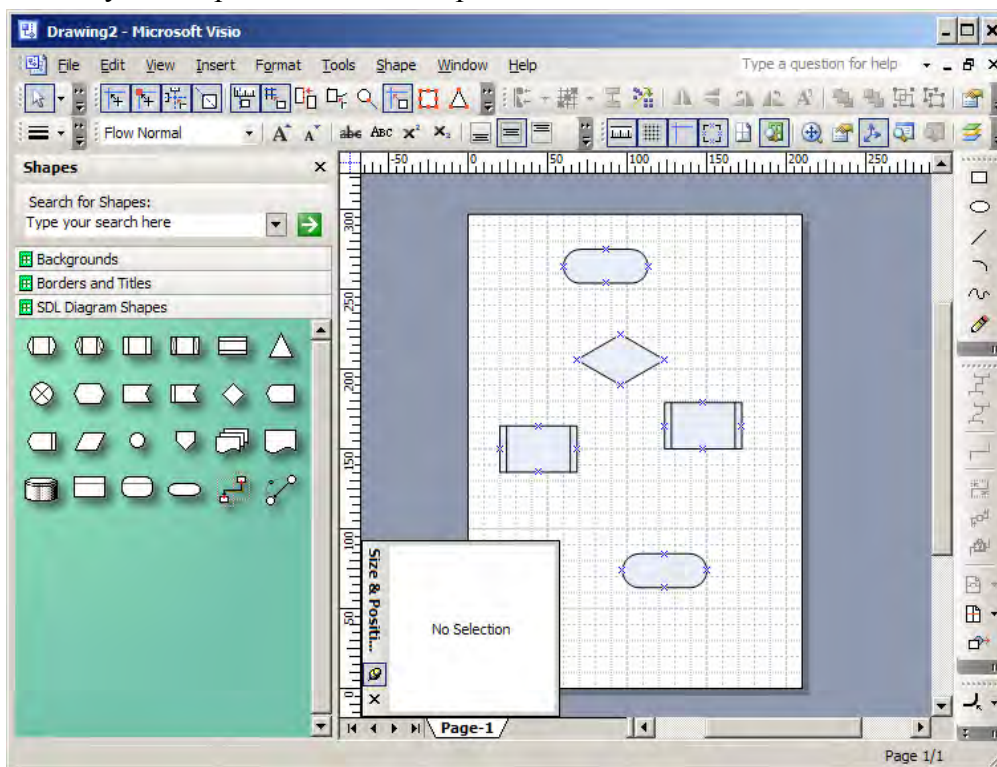



Рисунок 2.3 – Наполнение рабочей области шаблонами

Для смены цвета заливки выделяем элемент и, используя инструмент заливки «» (ведро), нажимаем левой клавишей мыши стрелку справа от знака «ведро». В выпадающем списке выбираем цвет, например белый (рисунок 2.4).

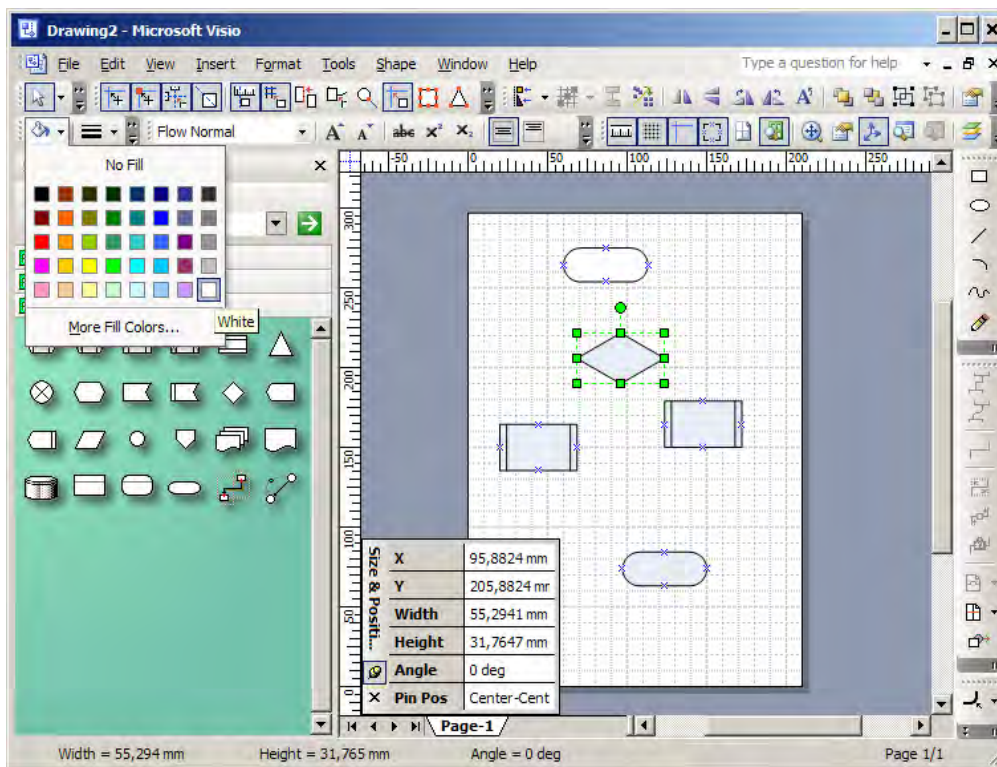


Рисунок 2.4 Выделение элемента на рабочей области

Для изменения вида элемента «предопределенный процесс», выделяем его и, зафиксировав левой клавишей указатель мыши на желтом ромбе, который указан на рисунке 2.5, смещаем внутреннюю линию до наружных границ элемента.

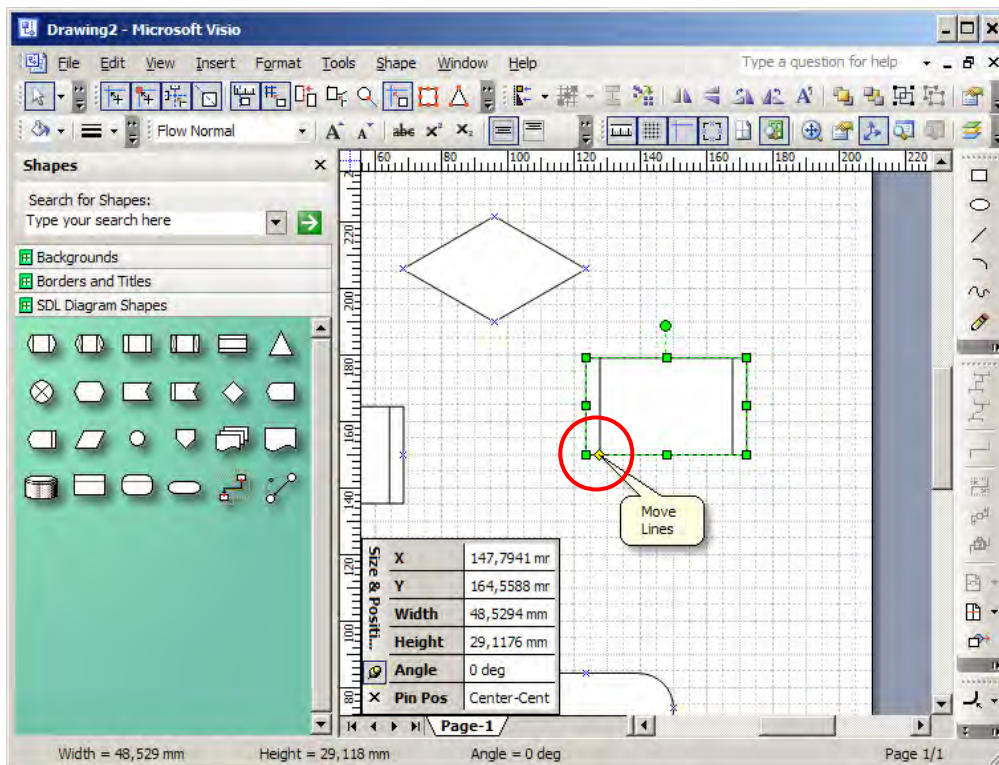


Рисунок 2.5 – Изменение особенностей отображения элемента при наличии такой возможности

2.1 Окно изменения размеров и координат

Для того, чтобы отобразить окно, позволяющее точно задать размеры и координаты элементов, выбираем пункт меню View/Size&Position Window (рисунок 2.6).

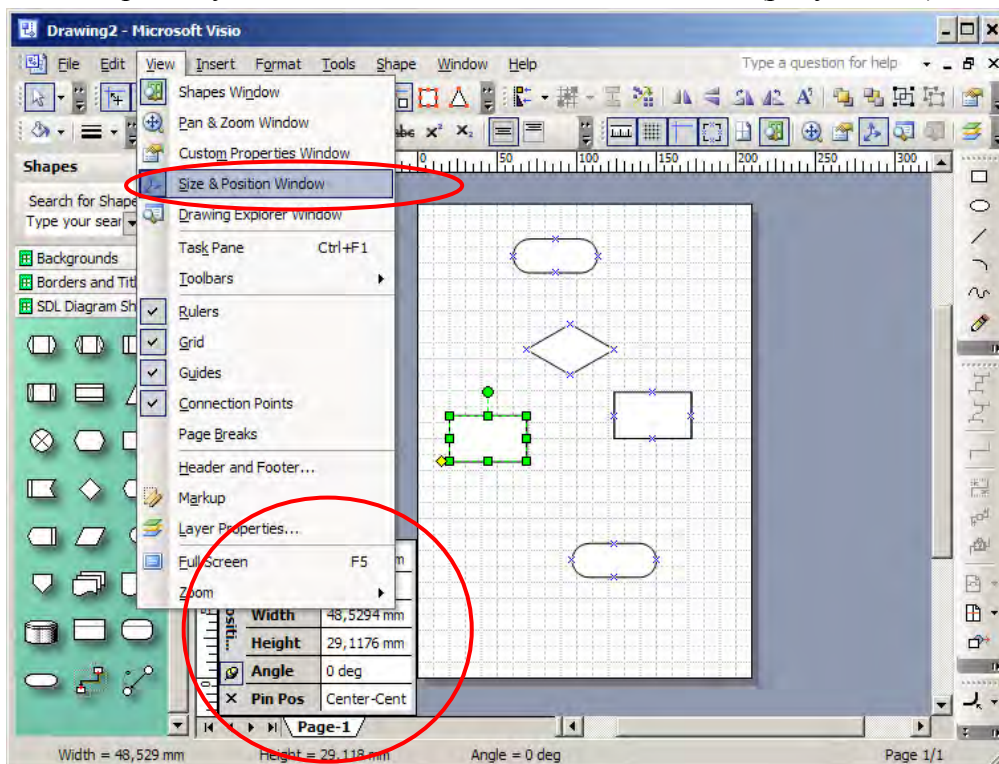


Рисунок 2.6 – Включение окна отображения координат и размеров элемента

2.2 Средства выравнивания и распределения элементов

Для облегчения выравнивания редактируемых элементов, добавляем соответствующие инструменты на инструментальную панель. Для этого нажимаем правую клавишу мыши в любом месте меню или инструментальной панели. В выпадающем списке выбираем пункт «Action» (рисунок 2.7).

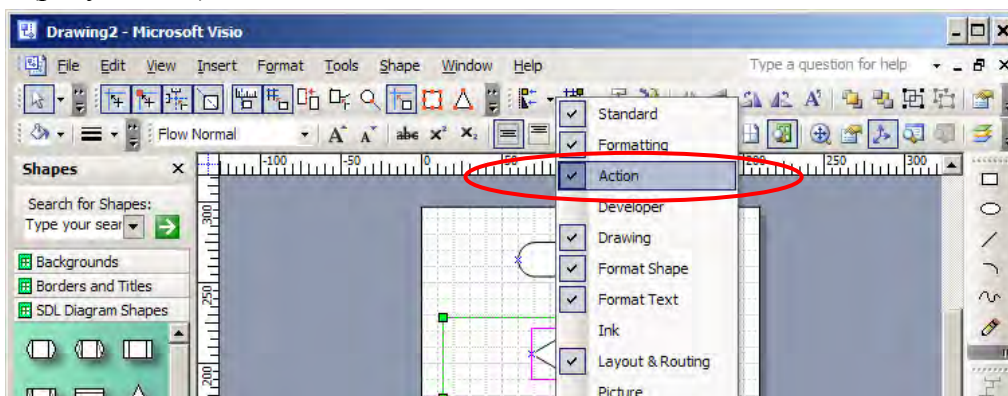



Рисунок 2.7 – Добавление средств выравнивания на инструментальную панель

Распределение элементов возможно либо по критерию равенства расстояний между ними, либо по расстоянию между центрами элементов. Последовательно указателем мыши, фиксируя левой клавишей, выбираем несколько элементов, удерживая при этом нажатой клавишу CTRL на клавиатуре. Далее нажимаем указателем мыши стрелку справа от инструмента распределения, обозначенного как «». В открывшемся списке выбираем тип распределения, например по горизонтали с выравниванием расстояния между элементами (рисунок 2.8).

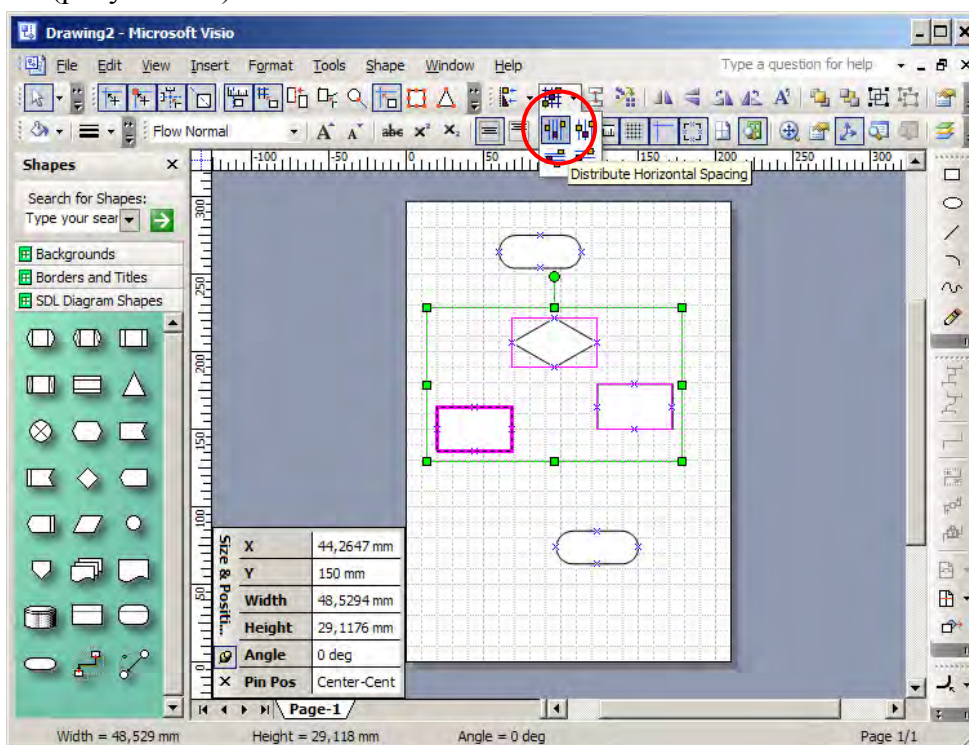


Рисунок 2.8 – Выбор типа распределения расстояния по горизонтали

Аналогично выравниваем расстояние между элементами по вертикали (рисунок 2.9).

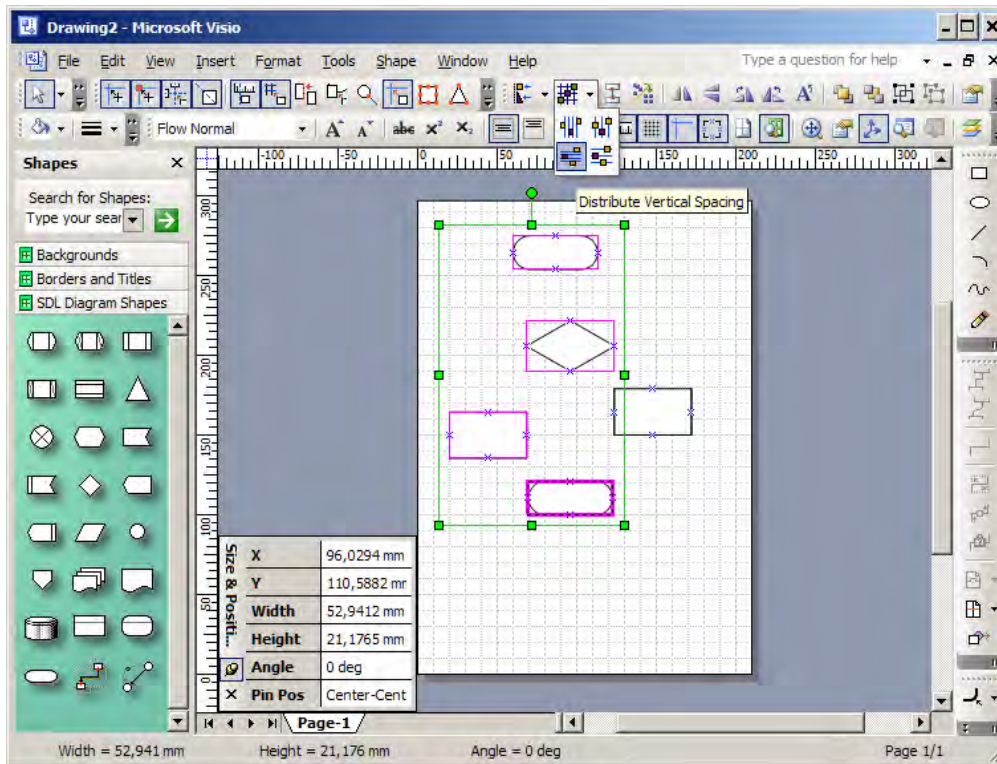



Рисунок 2.9 – Выбор типа распределения расстояния по вертикали

Следующий шаг – выравнивание элементов по одной линии (вертикальной или горизонтальной) по критерию центр или граница (правая/левая или верхняя/нижняя, соответственно). Выделяем требуемые элементы и выбираем из выпадающего списка на инструменте «», необходимый режим выравнивания. Внимание, выравнивание производится по первому выделенному элементу, рамка которого отображается более жирно (рисунок 2.10).

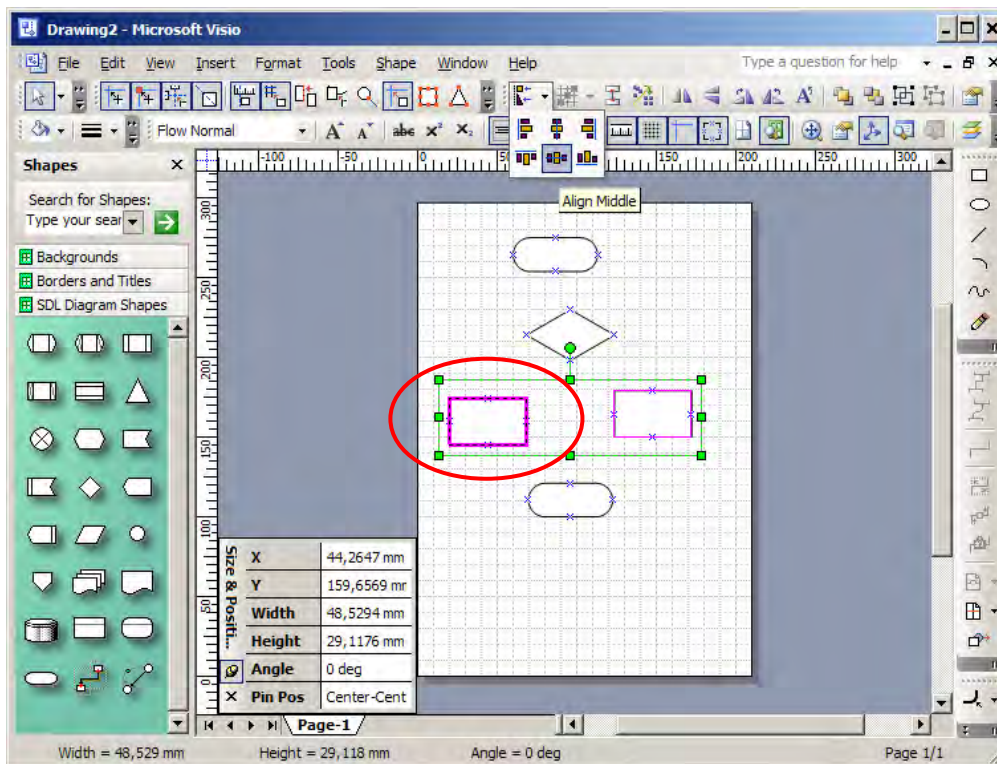


Рисунок 2.10 – Выравнивание элементов по горизонтали на уровне выделенного

Выравнивание элементов по вертикальной оси производим аналогичным образом (рисунок 2.11).

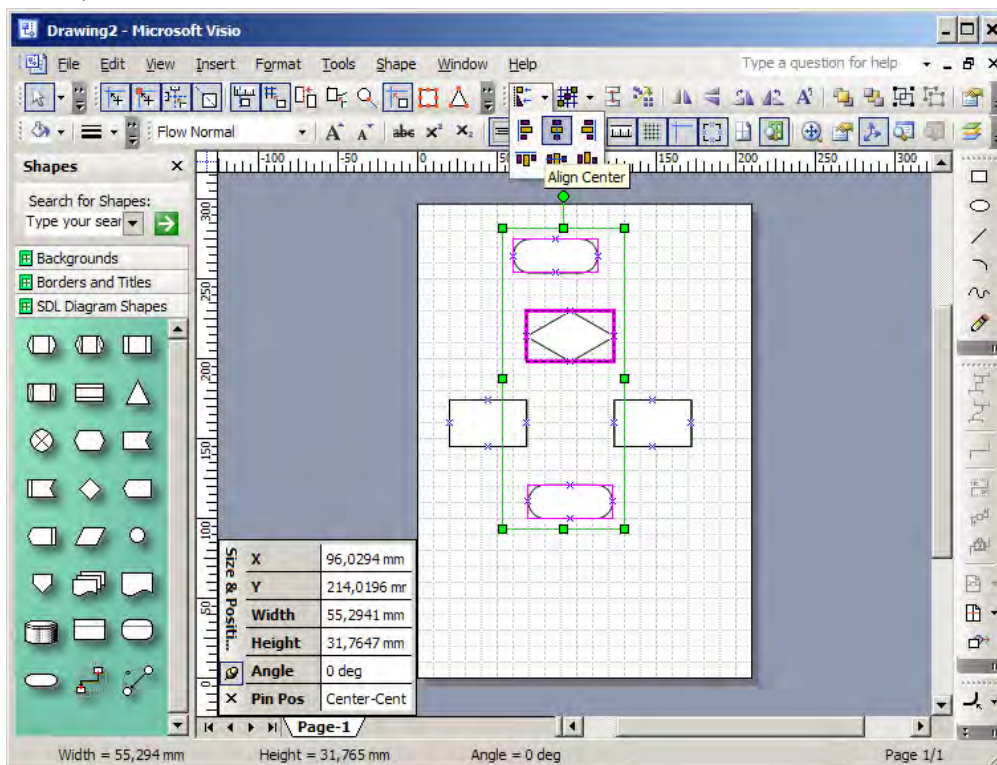



Рисунок 2.11 – Выравнивание элементов по вертикали на уровне выделенного

2.3 Соединительные линии

Для соединения элементов предусмотрены специальные соединители, наиболее универсальным из которых является инструмент, обозначенный на инструментальной панели как  – Connector tool (рисунок 2.12).

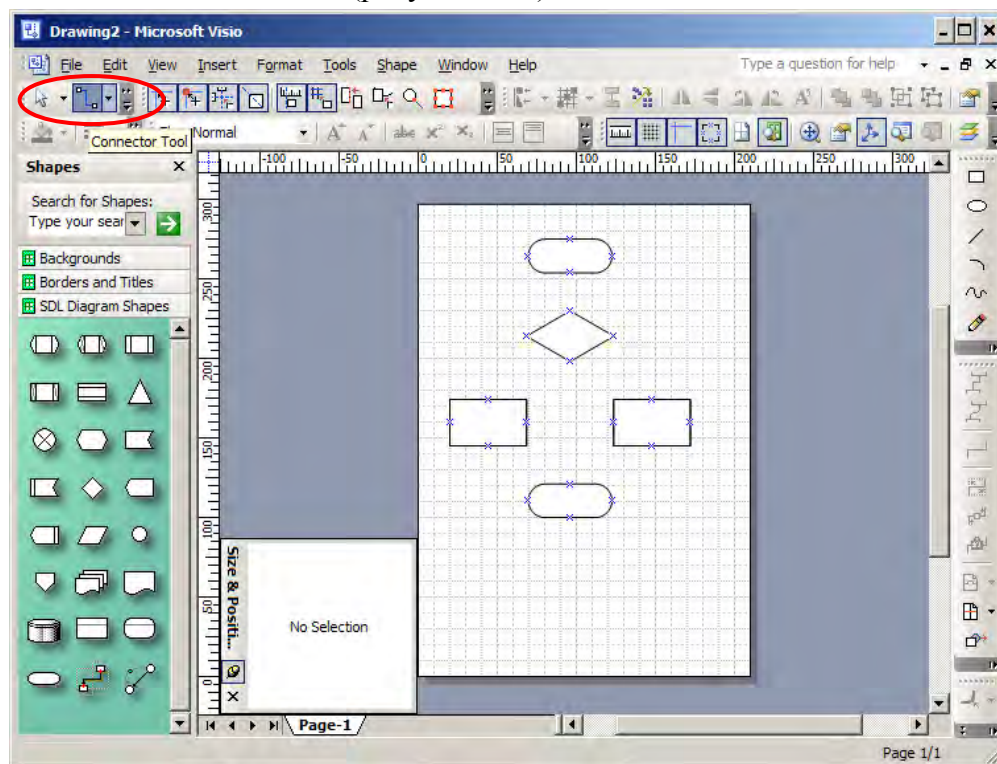




Рисунок 2.12 – Выбор соединителя элементов

Элементы из шаблона “SDL Diagram Shapes” имеют определенные точки для соединителей, обозначаемые как . После того, как выбран «Connector tool» наведение указателя мыши на соответствующую точку вызывает изменение её вида на  (рисунок 2.13).

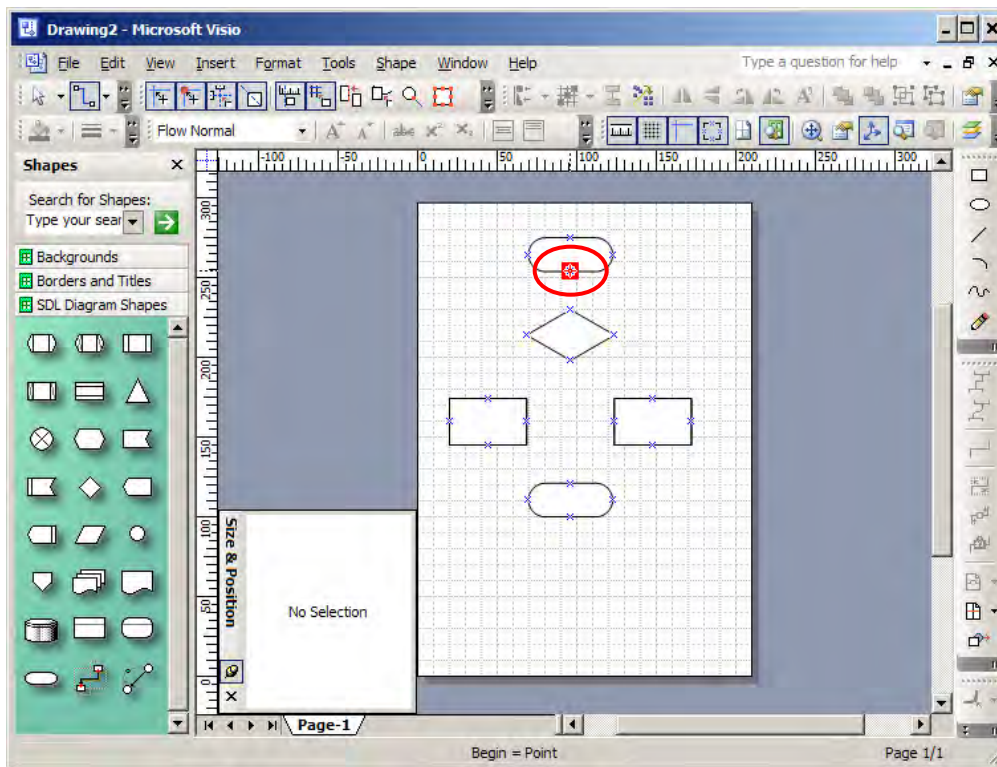


Рисунок 2.13 – Точки присоединения соединителей на элементах

Фиксируем начало соединителя и, не отпуская левую клавишу мыши, перемещаем указатель на точку окончания соединителя (рисунок 2.14).

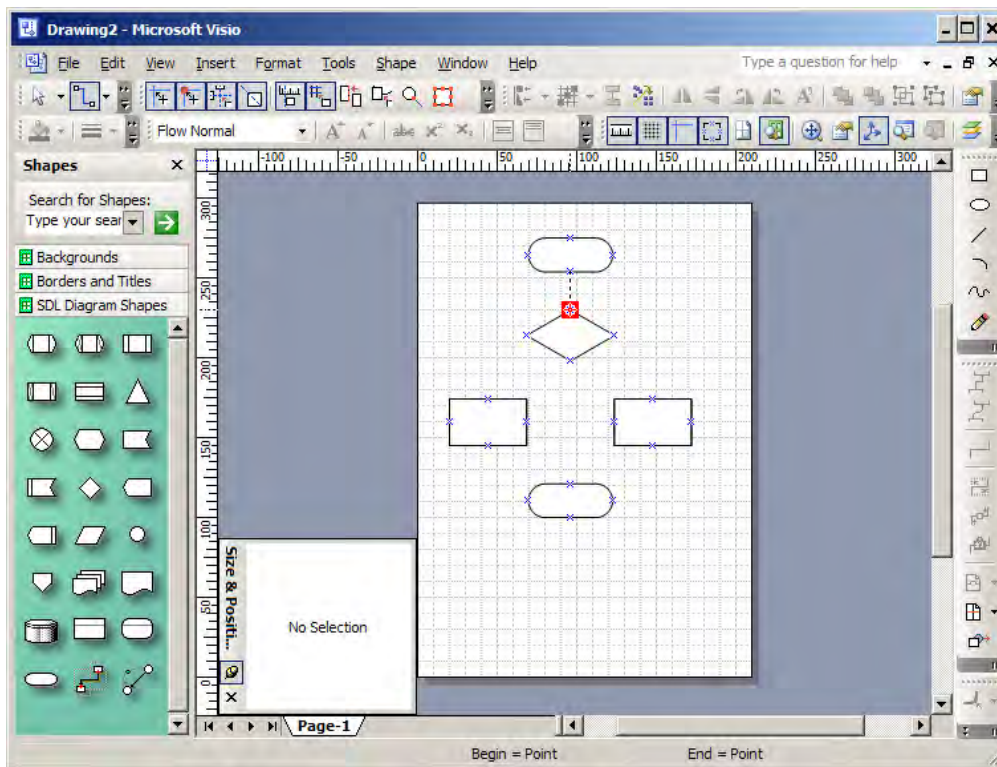



Рисунок 2.14 – Соединение со вторым элементом

После соединения элементов необходимо отключить режим соединения. Для этого выбираем рисунок . При этом схема должна выглядеть подобно рисунку 2.15.

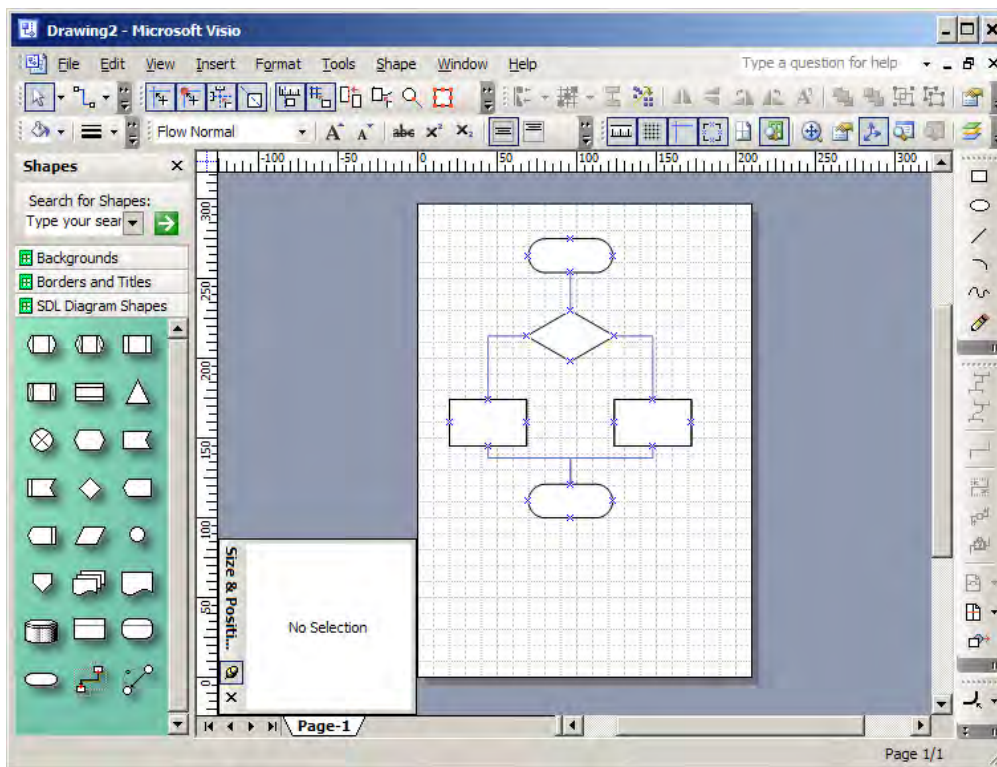


Рисунок 2.15 – Схема с нанесенными линиями-соединителями

Изменяем цвет линий-соединителей, выделив их и выбрав соответствующий инструмент (рисунок 2.16).

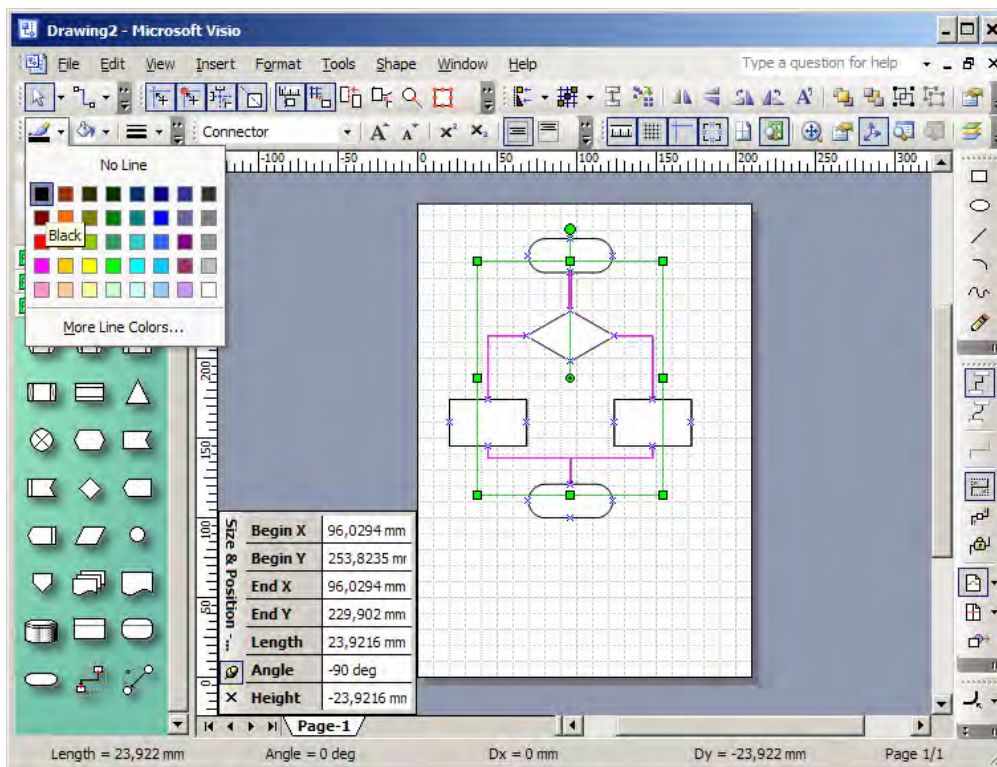



Рисунок 2.16 – Изменение цвета линий-соединителей

Выбор стрелок на концах линий-соединителей осуществляется инструментом, обозначенным как . Нажимаем стрелку справа от изображения и в выпадающем списке выбираем один из предложенных вариантов, либо пункт «More Line Ends» (рисунок 2.17).

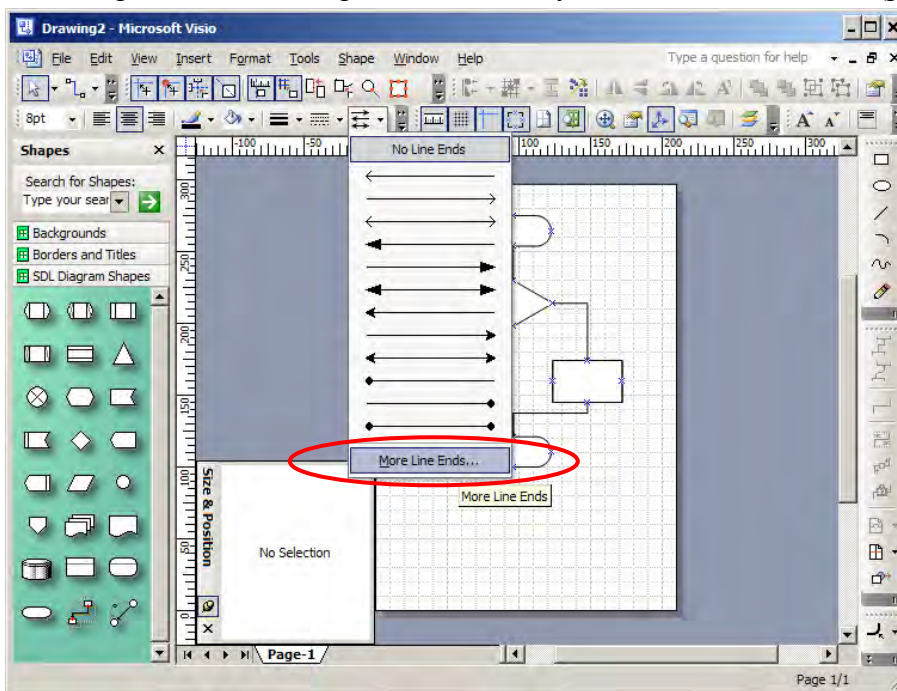


Рисунок 2.17 – Выбор типа начала-окончания линии соединителя

В открывшемся диалоге выбираем тип стрелки для конца соединителя (т.е. вторая выбранная точка), а также размер стрелки (рисунок 2.18). **Внимание!** Размер стрелки зависит от толщины линии, но не зависит от размеров соединяемых элементов. Т.е. при одних и тех же параметрах стрелок, их видимый размер на диаграмме будет зависеть от выбранного размера прочих элементов.

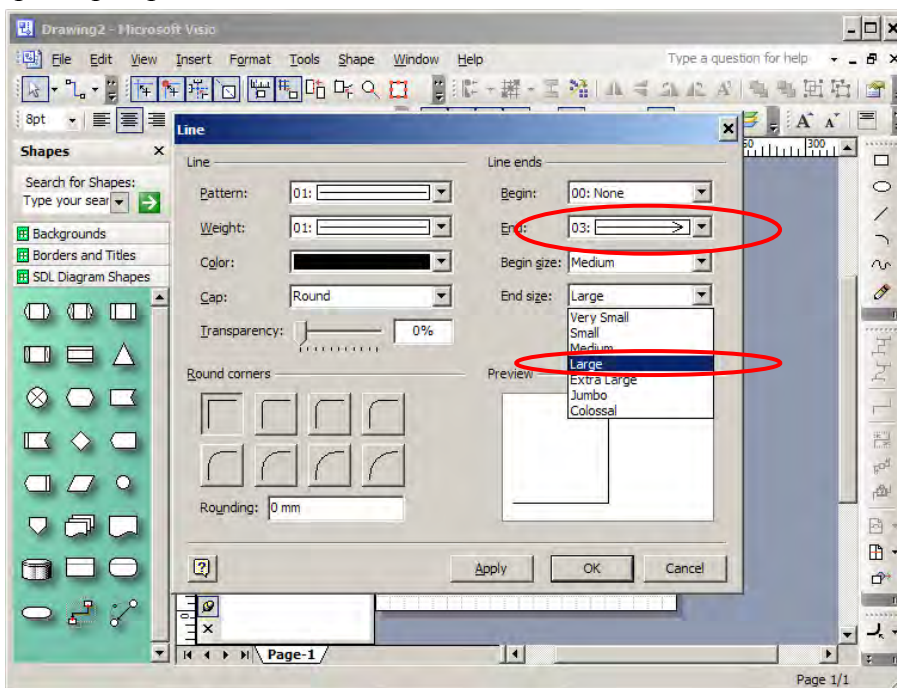


Рисунок 2.18 – Диалог изменения окончаний линии-соединителя

Результат изменения типа окончания линии соединителя приведен на рисунке 2.19.

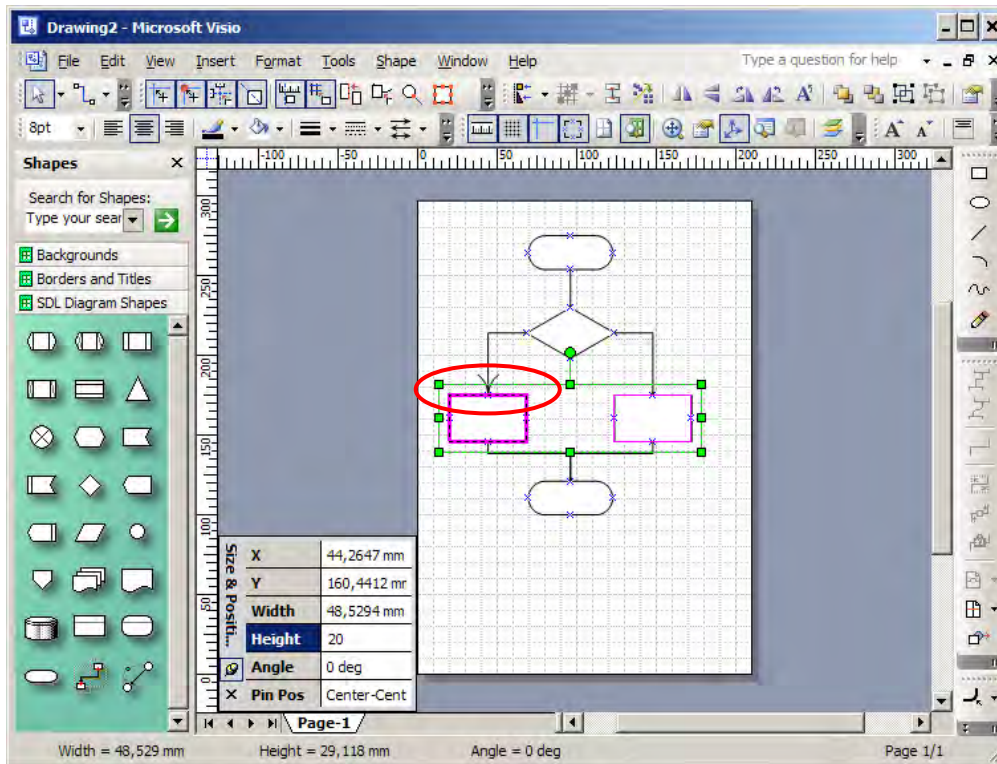


Рисунок 2.19 – Результат изменения типа окончания линии-соединителя

2.4 Добавление текста

Вставка текста в элементы осуществляется следующим образом. Необходимо выполнить указателем мыши двойной щелчок левой клавишей на элементе, в который надо вставить текст. После чего ввести текст (рисунок 2.20).

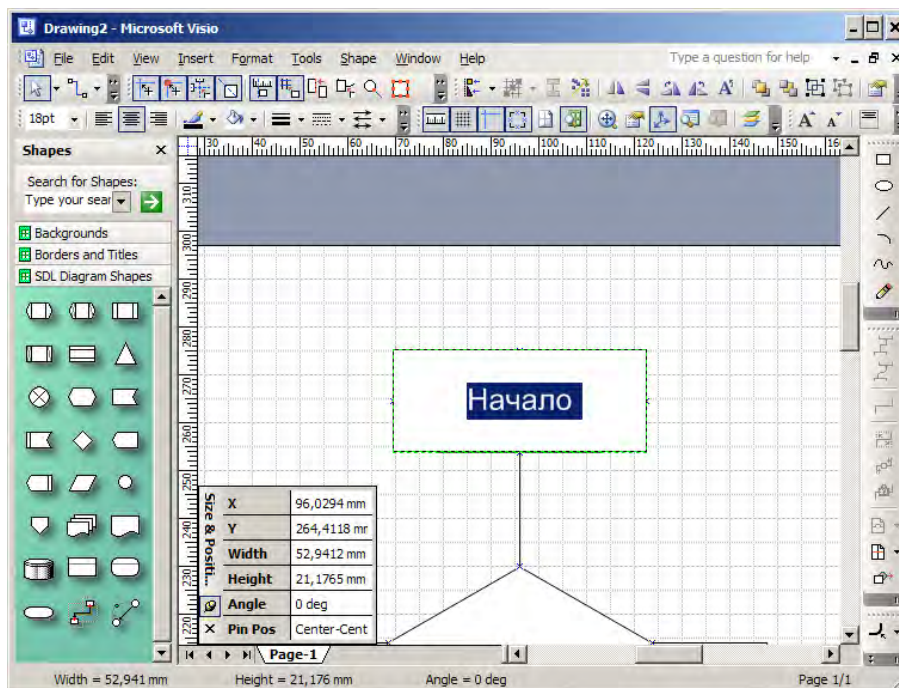



Рисунок 2.20 – Ввод текста в элемент

Используя инструмент «Text Tool», обозначенный как , добавляем текстовые надписи, такие как «Да» и «Нет» над стрелками блока решения.

2.5 Сохранение диаграммы и экспорт в векторном формате

Для того, чтобы можно было **работать с диаграммой в дальнейшем**, необходимо сохранить её в **формате Visio**, выбрав меню File/Save и формат по умолчанию **vsd**. Если диаграмма будет сохранена в других форматах – при попытке открыть их в Visio в дальнейшем, будет потеряна связь между шаблонами элементов и графическим отображением, ранее им соответствовавшим. Visio позволяет открывать и редактировать файлы в формате wmf, однако например текст внутри элементов будет отдельным графическим элементом, никак не связанным с изображением, внутри которого он размещен. Также будут потеряны динамические соединители, изображение которых останется в том виде, в каком они были на момент создания wmf.

Для **вставки диаграмм в текстовый редактор** целесообразно сохранить их в векторном графическом формате, например wmf, поскольку редактирование таких рисунков в дальнейшем не предполагается. Векторный формат позволит при необходимости осуществлять масштабирование рисунков в тексте без потери качества изображения, а также минимизировать размер результирующего файла.

Выделяем всю диаграмму. Выбираем пункт меню File/Save as (рисунок 2.21).

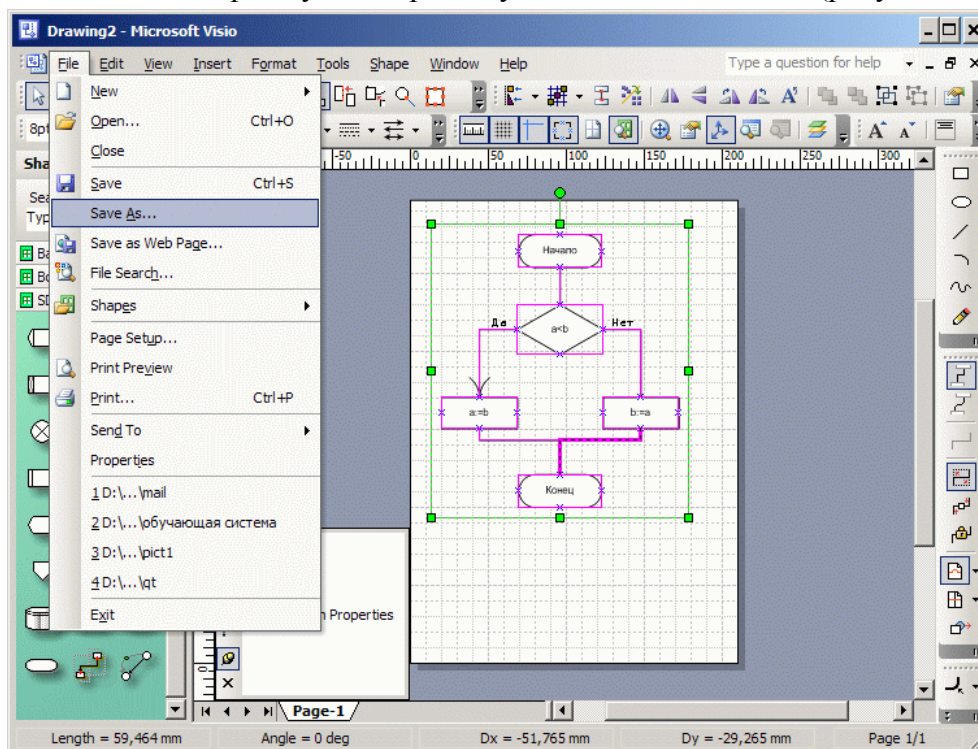


Рисунок 2.21 – Сохранение выделенного фрагмента диаграммы

В открывшемся диалоге выбираем директорию назначения, имя файла и тип «wmf». Нажимаем кнопку Save.

3 Создание текстового документа MS Word со схемой алгоритма

При наличии текстового редактора MS Word, запускаем его. Если этот текстовый редактор не установлен, переходим к выполнению схемы алгоритма в OpenOffice Draw.

После запуска MS Word автоматически создается пустой текстовый документ. Предположим, что схему алгоритма необходимо вставить в текст документа. Месту вставки рисунка будет определяться положением текстового курсора. Для вставки выбираем «Вставка/ рисунок/ из файла» и указываем соответствующее имя файла (рисунок 3.1).

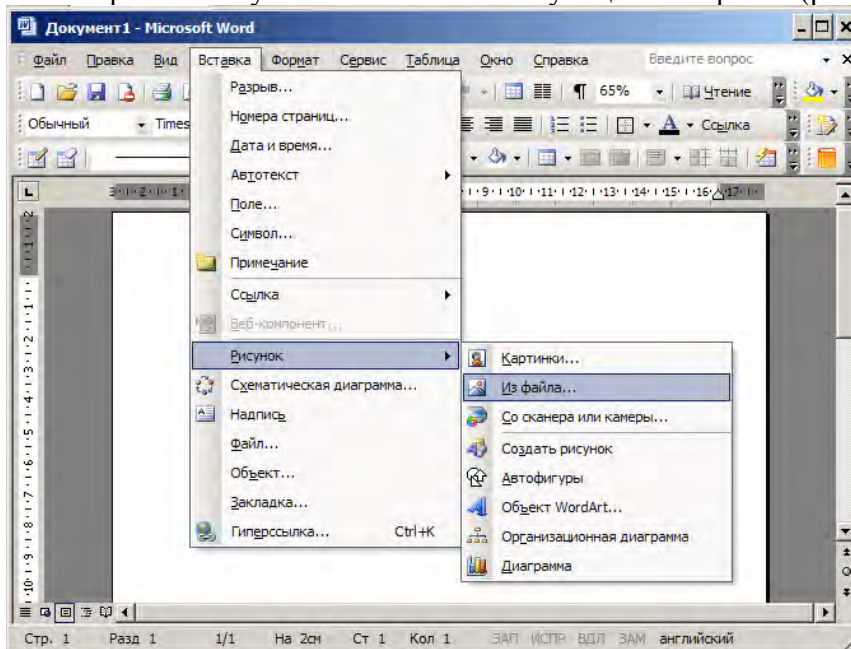


Рисунок 3.1 – Добавление рисунка в текстовом редакторе Word

Результат добавления рисунка в текст документа приводится на рисунке 3.2.

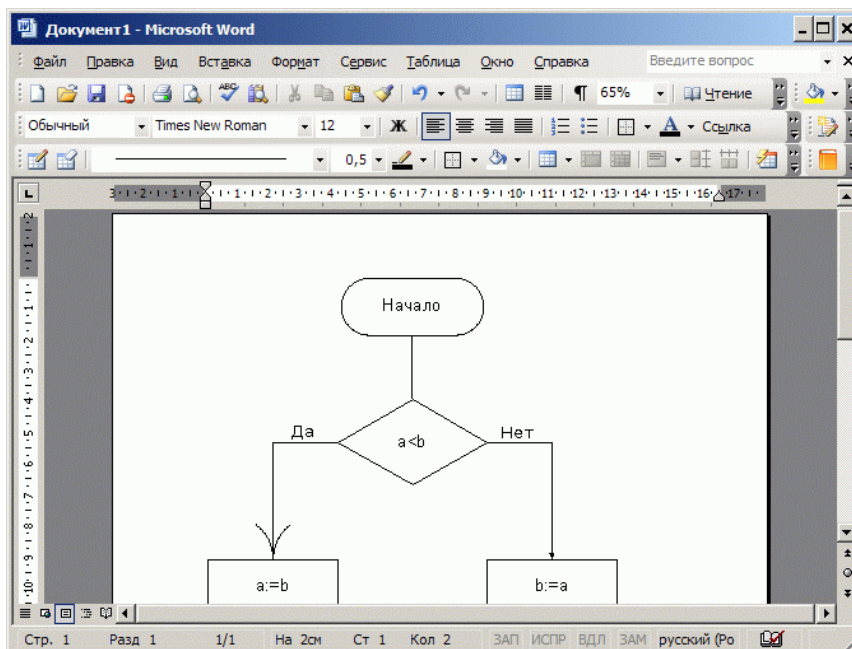


Рисунок 3.2 – Рисунок в текстовом документе

Размер вставленного рисунка может быть изменен. Для этого нажимаем правой клавишей мыши на рисунке и, в открывшемся меню, выбираем «Формат рисунка» (рисунок 3.3).

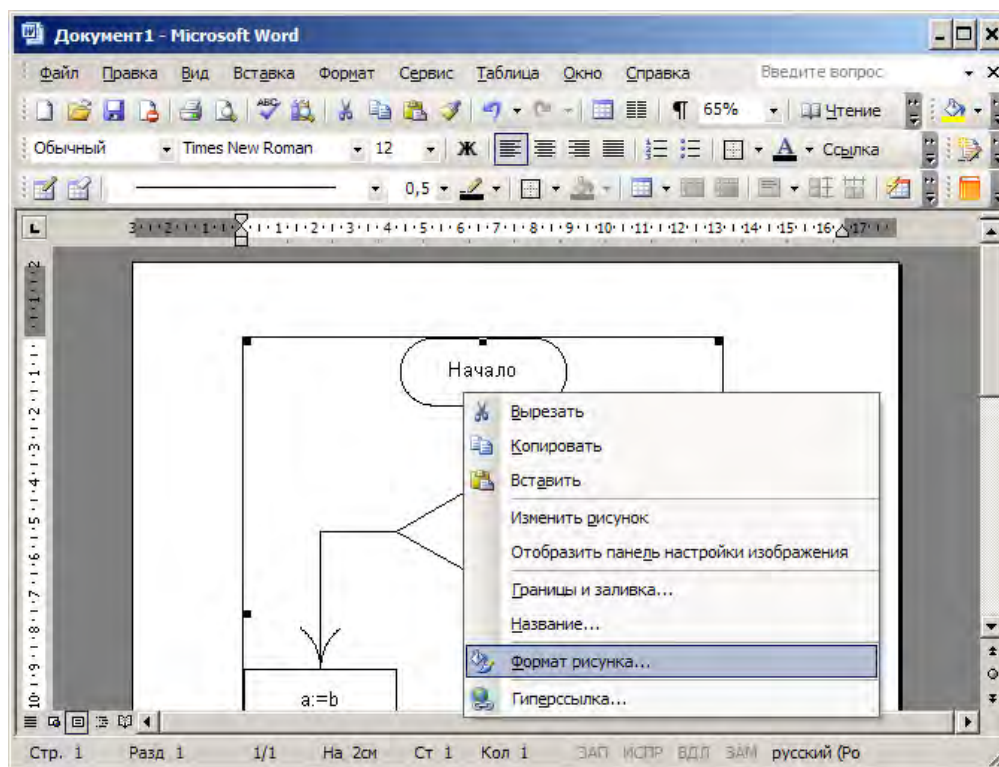


Рисунок 3.3 – Открытие диалога изменения формата рисунка

Находим закладку «Размер» и указываем масштаб по высоте или ширине. Если установлен пункт «сохранять пропорции», изменение высоты или ширины автоматически изменяет второй размер (рисунок 3.4).

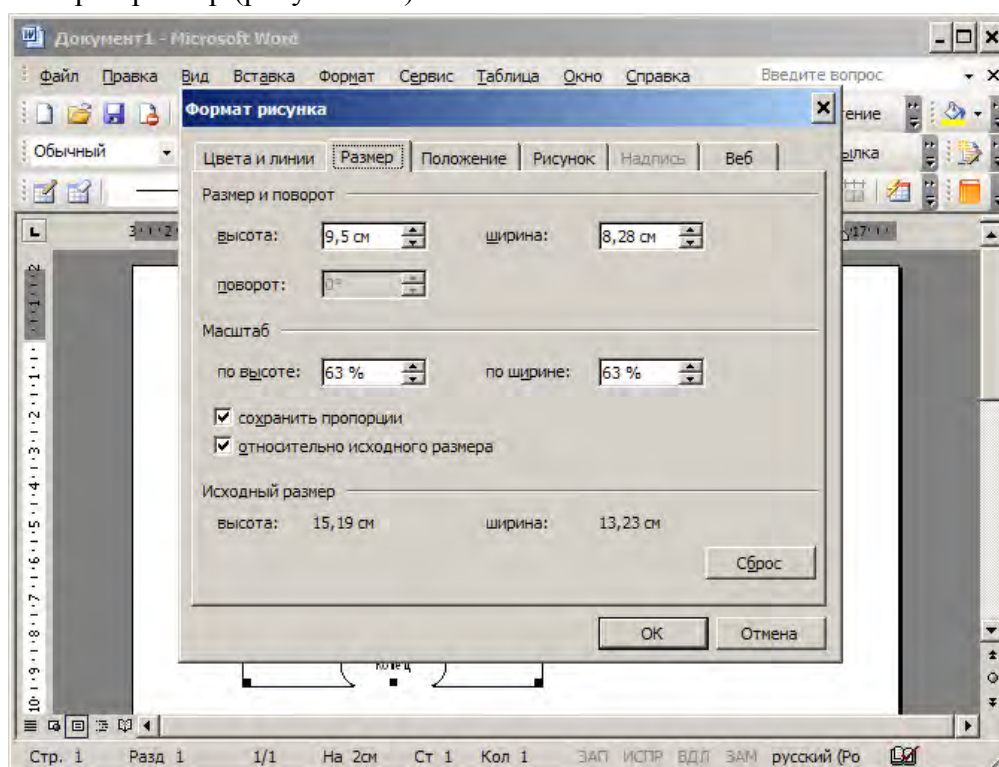


Рисунок 3.4 – Диалог изменения формата рисунка

Изменить размер рисунка можно и другим способом. Щелкнув указателем мыши по рисунку левой клавишей, получаем рамку с отмеченными областями, позволяющими изменять размеры пропорционально или по ширине/высоте (рисунок 3.5).

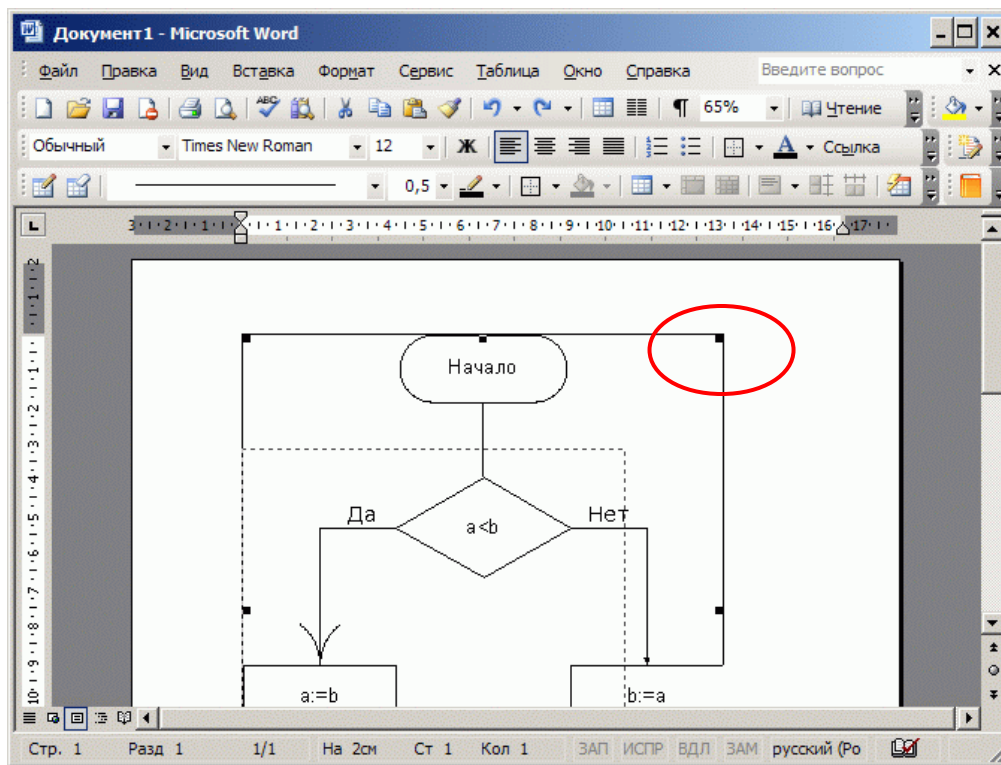


Рисунок 3.5 – Изменение размера рисунка при помощи мыши

Изменив размер, продолжаем редактировать текстовый документ (рисунок 3.6).

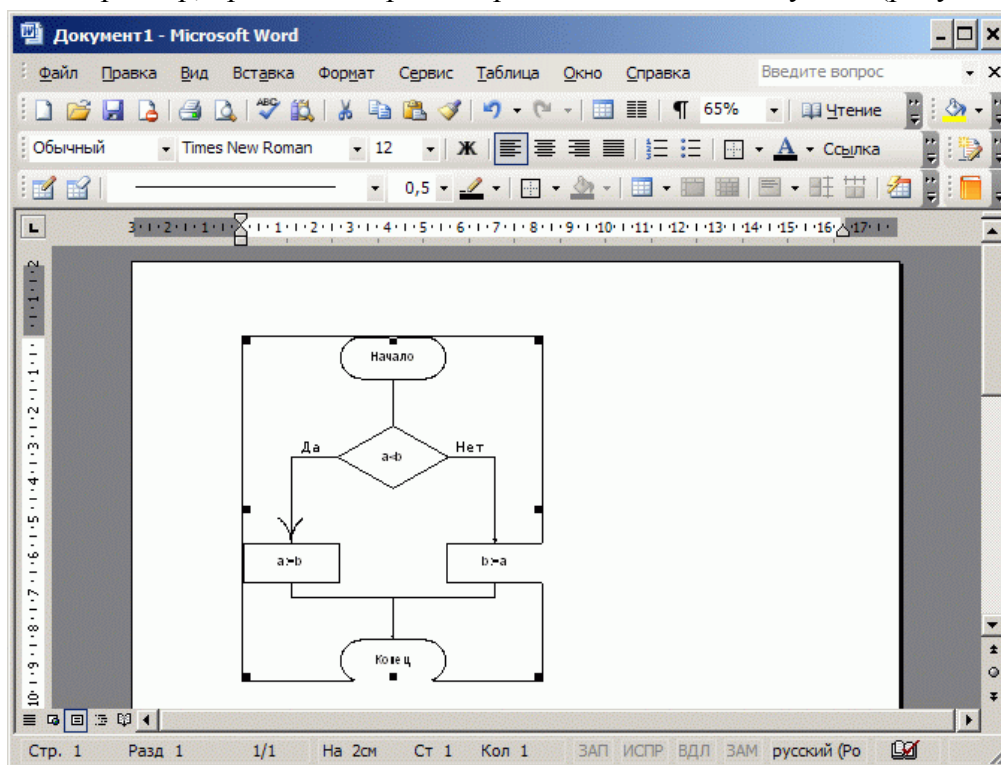


Рисунок 3.6 – Рисунок после изменения размера

По окончании редактирования документа его необходимо сохранить, выбрав пункт меню Файл/Сохранить.

4 Создание схем алгоритмов средствами OpenOffice Draw

Запускаем OpenOffice Draw из меню Пуск/Программы/OpenOffice.org/OpenOffice.org Draw.

После запуска программы автоматически создается пустой документ, в котором можно сразу начинать рисование (рисунок 4.1).

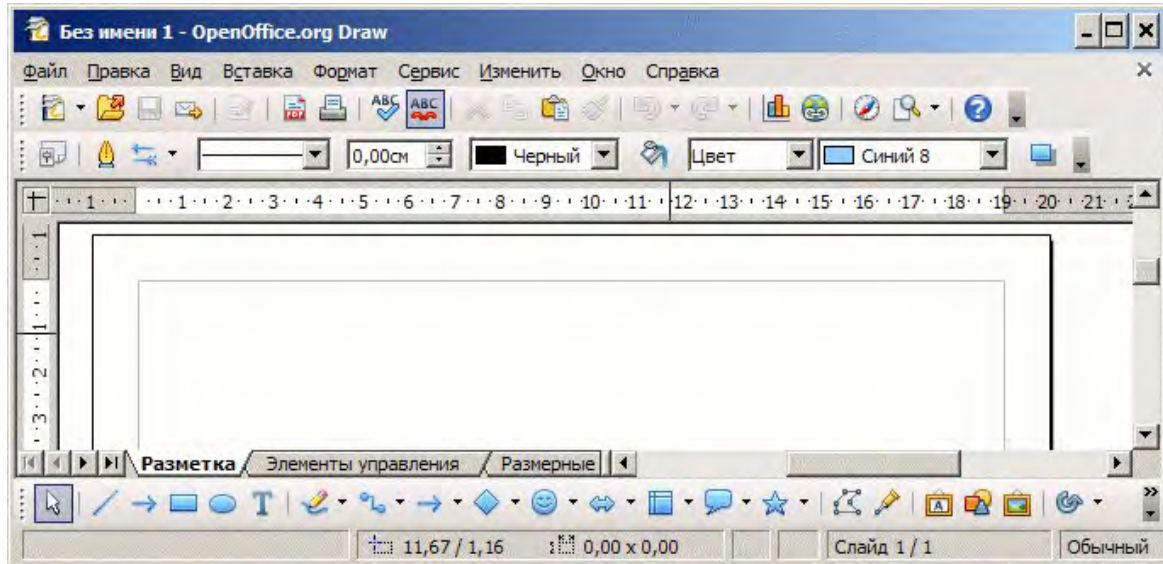


Рисунок 4.1 – Пустая диаграмма OpenOffice Draw

По умолчанию предлагается размещать графические элементы на рабочем пространстве формата А4. Его размер определяется в меню Формат/Страница и может быть установлен как в стандартные величины А0, А1...А6, так и не стандартно. Следует отметить, что размер рабочего пространства не связан с размером бумаги печатающего устройства, непосредственно подключенного к компьютеру, на котором производится рисование.

Используя кнопки панели инструментов «Рисование», выбираем необходимую фигуру и размещаем её на рабочем пространстве. Например, скругленный прямоугольник используем как блок начала и окончания алгоритма (рисунок 4.2).

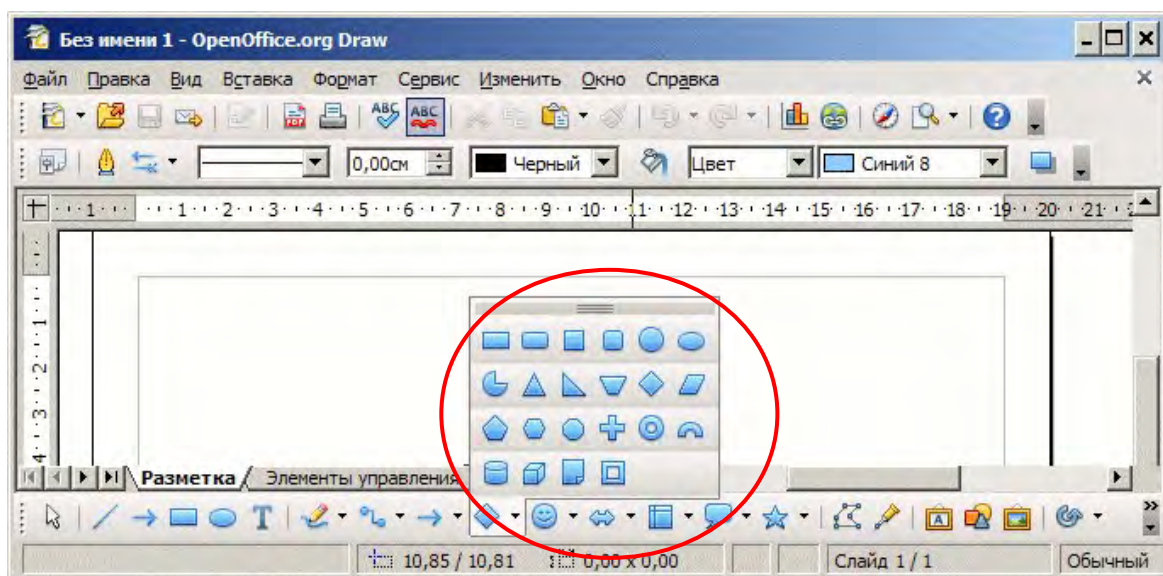


Рисунок 4.2 – Выбор типа графического элемента

Некоторые элементы снабжены средствами изменения их параметров. В частности, для «скругленного прямоугольника» может быть изменен радиус закругления (рисунок 4.3).

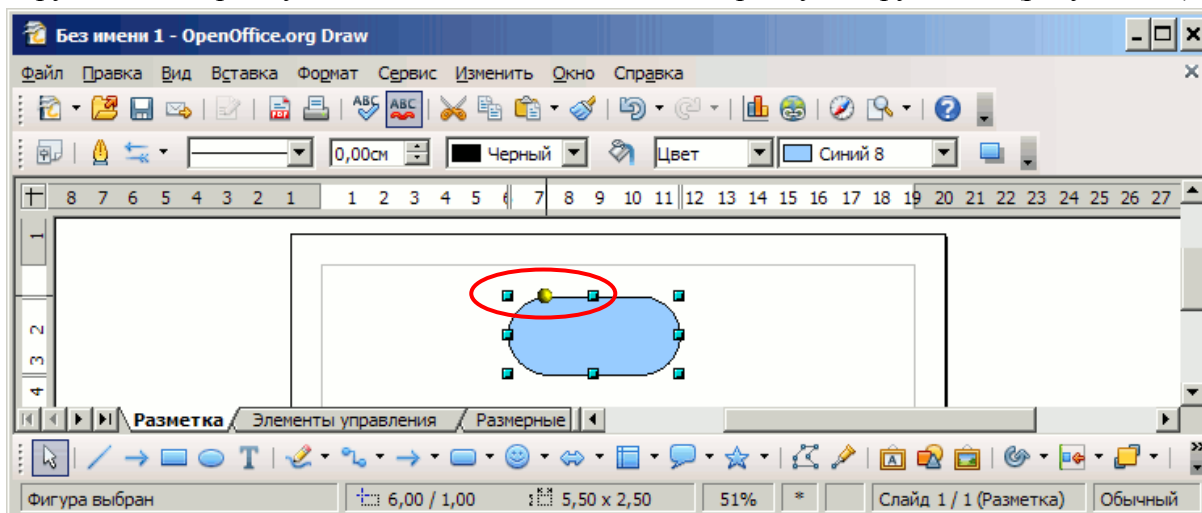


Рисунок 4.3 – Изменение специфических параметров элемента

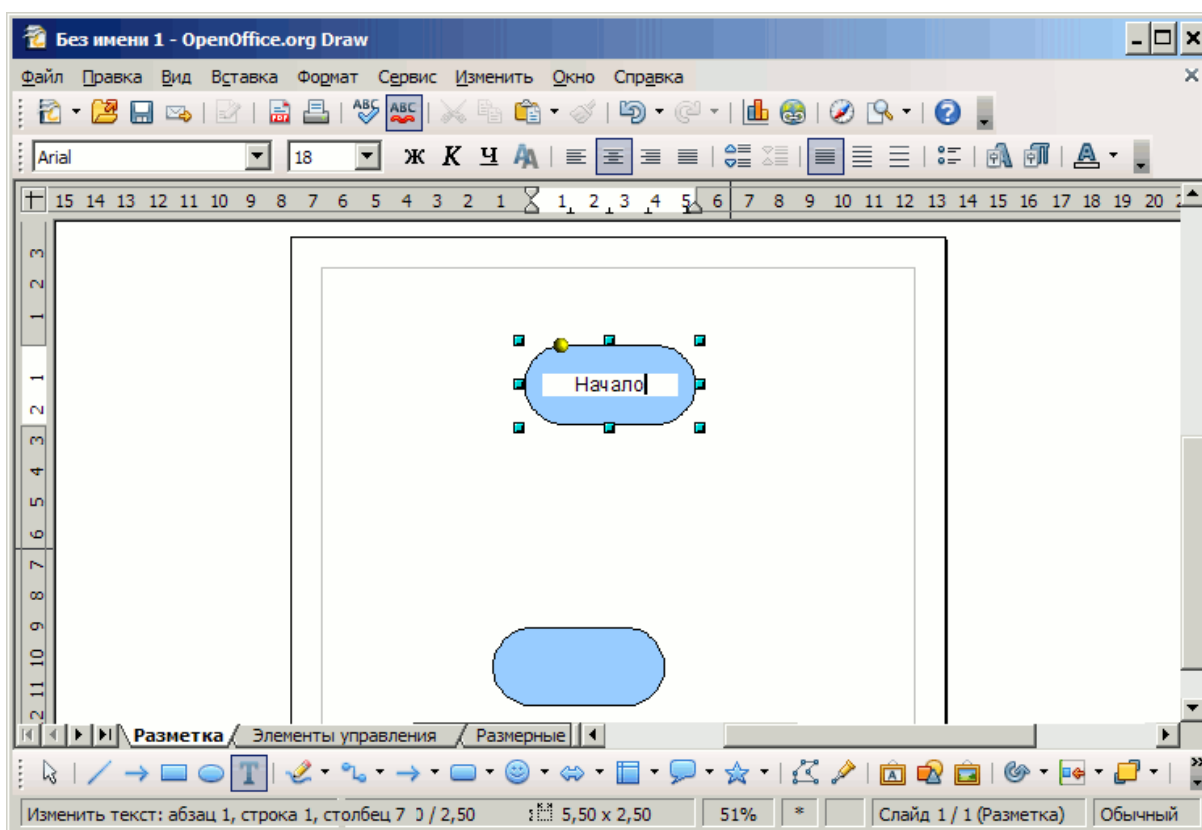


Рисунок 4.4 – Ввод текста в элемент

Копирование блоков может быть выполнено через буфер обмена – меню Правка/Копировать, Вставить или CTRL+C, CTRL+V. Для добавления надписи внутрь графического элемента необходимо выполнить на этом элементе двойное нажатие левой клавишей мыши (рисунок 4.4).

Первоначально, элементы следует размещать приблизительно. Точное выравнивание производится после того, как все элементы размещены и соединены линиями.

Цвет заливки графических элементов может быть изменен при помощи кнопки Стиль/Заливка области панели инструментов Линия и Заливка (рисунок 4.5).

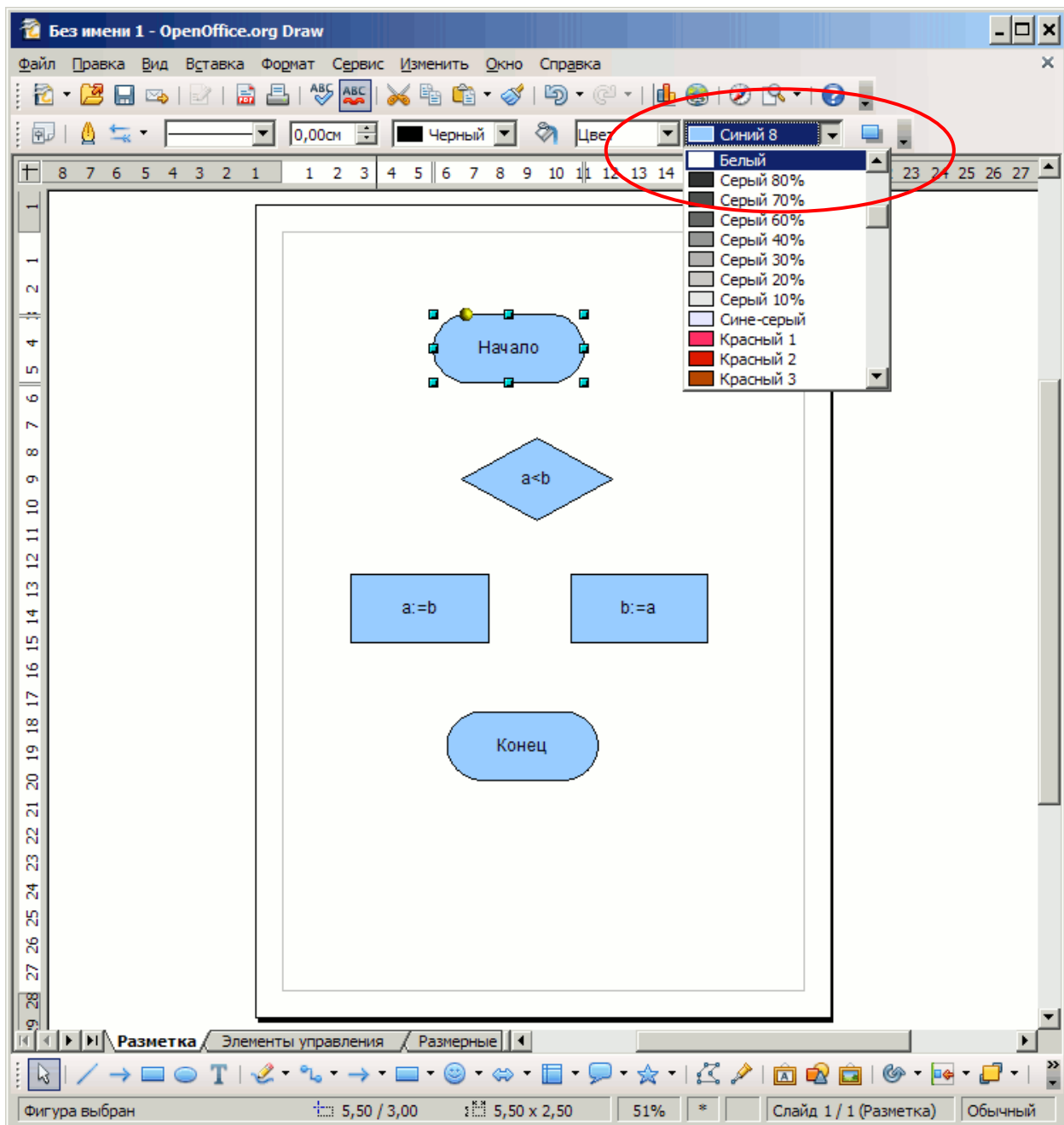


Рисунок 4.5 – Выбор цвета заливки элемента

Связывание элементов обеспечивается при помощи «Соединительных линий». Стрелки на соединительных линиях могут быть изменены позднее. Наиболее универсальная соединительная линия, которая так и называется «Соединительная линия» (рисунок 4.6).

4.1 Соединительные линии

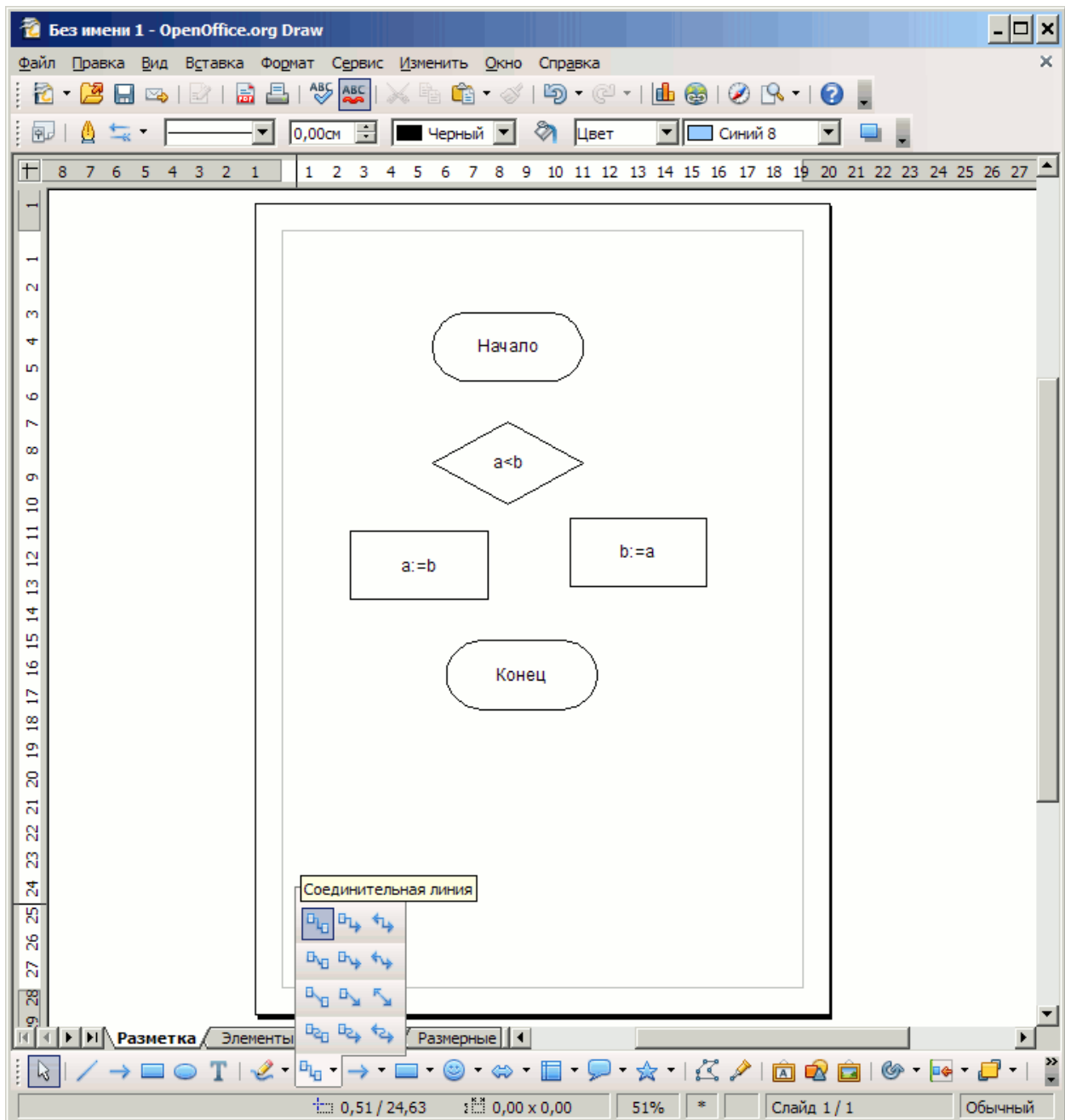


Рисунок 4.6 – Выбор типа линии-соединителя

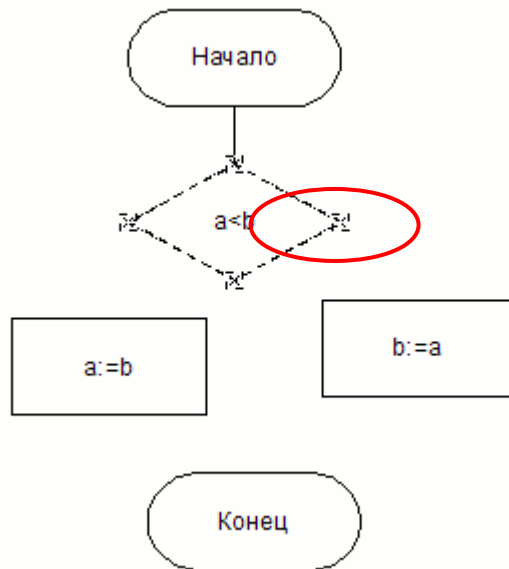


Рисунок 4.7 – Подсветка точек присоединения начала линии-соединителя

Выбрав соединительную линию нужного типа, наводим указатель мыши на элемент-источник. Его контур примет пунктирный вид, а в местах возможного присоединения соединительных линий появятся перекрестия. Нажимаем левую клавишу мыши на нужном перекрестии и, не отпуская клавишу, ведем к элементу, с которым должно быть выполнено соединение (рисунок 4.7).

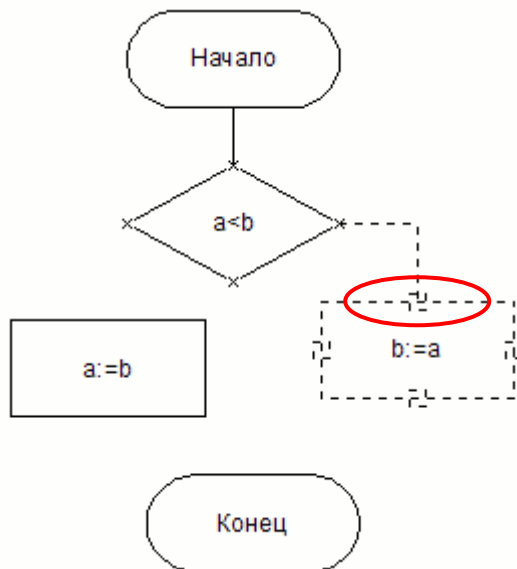


Рисунок 4.8 –Подсветка точек присоединения окончания линии-соединителя

После наведения указателя мыши на подчиненный элемент, его контур также меняет вид, и появляются перекрестия, аналогично первому элементу. Наводим на перекрестие и отпускаем левую клавишу мыши (рисунок 4.8).

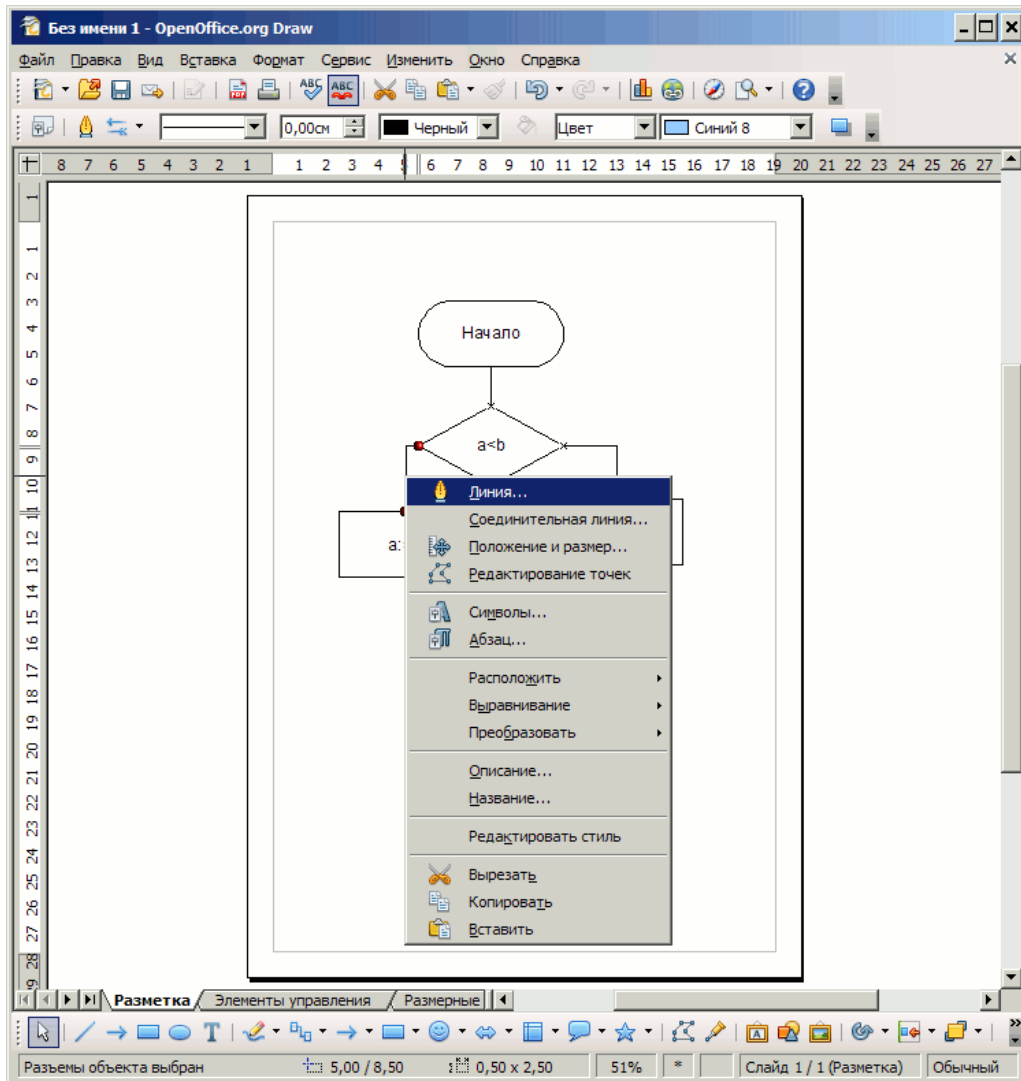


Рисунок 4.9 – Выбор типа линии-соединителя

Соединяем все элементы аналогичным образом. При необходимости изменить внешний вид соединительной линии следует на соответствующей соединительной линии нажать правую клавишу мыши и выбрать пункт контекстного меню «Линия» (рисунок 4.9).

В открывшемся диалоговом окне можно выбрать тип стрелок, цвет и ширину (рисунок 4.10).

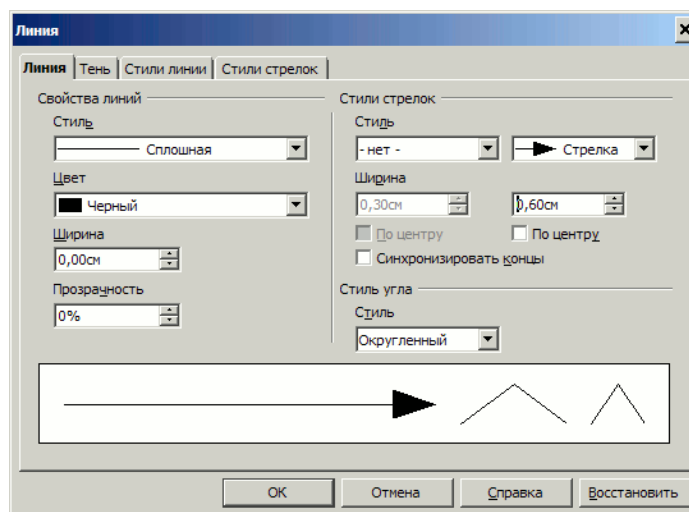


Рисунок 4.10 –Диалог выбора типа линии-соединителя

4.2 Средства выравнивания и распределения элементов

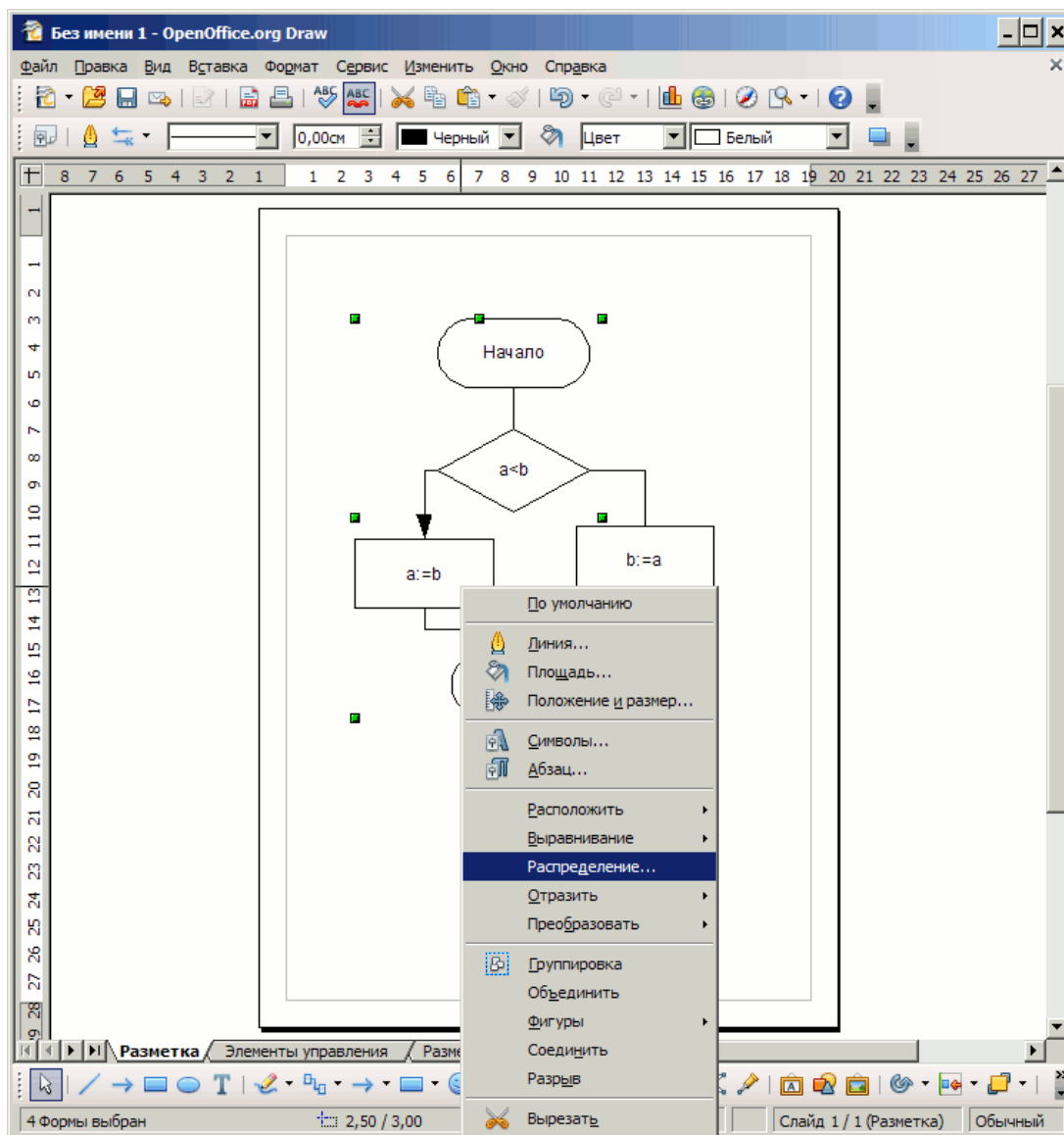


Рисунок 4.11 – Вызов диалога выбора распределения для выделенных элементов

Первоначально элементы были размещены приблизительно, следовательно, необходимо применить «распределение» – т.е. в заданном диапазоне элементов разместить промежуточные элементы и «выравнивание» – размещение выделенных элементов по определенным осям.

Для распределения, необходимо последовательно выделить соответствующие элементы и вызвав контекстное меню (правой клавишей мыши), и выбирать меню «Распределение» (рисунок 4.11). После активации этого пункта меню откроется диалог, изображенный на рисунке 4.12.

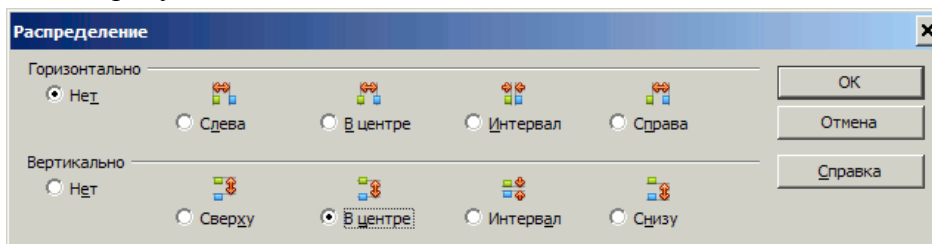


Рисунок 4.12 – Диалог выбора типа распределения элементов

Распределение может быть горизонтальным или/и вертикальным. Возможно выравнивание расстояний между левыми/правыми границами, выравнивание межцентрового расстояния или интервала между границами элементов.

Последовательно выравниваем по межцентровому расстоянию элементы, расположенные по вертикали (в случае нескольких элементов на одной горизонтали распределение осуществляем по одному из них – например по крайнему левому)

Аналогично выравниваем по межцентровому расстоянию элементы по горизонтали (в данном случае процессы – присвоения и блок принятия решения) (см. рисунок 4.13).

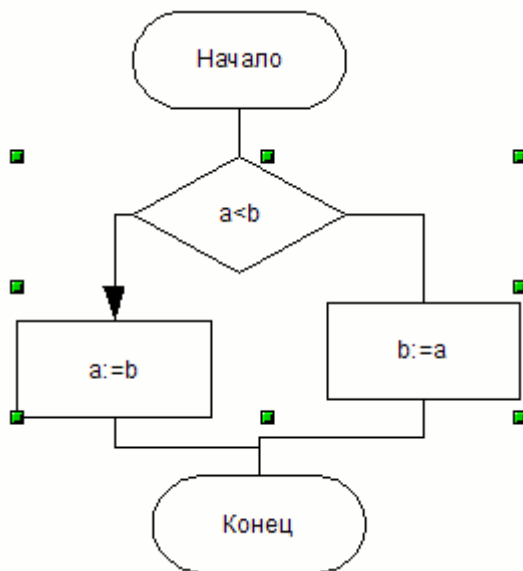


Рисунок 4.13 – Выделение элементов для распределения по горизонтали

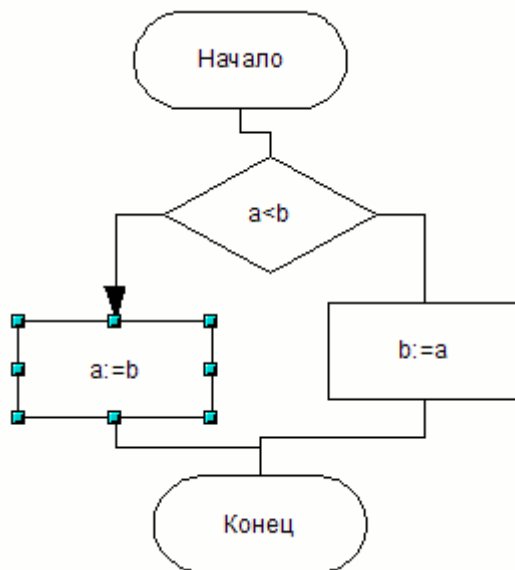


Рисунок 4.14 – Выделение опорного элемента для выравнивания

Следующим шагом является выравнивание элементов. Закрепляем опорный элемент, для чего выделяем его (рисунок 4.14) и нажимаем клавишу F4 (Формат/Положение и размер).

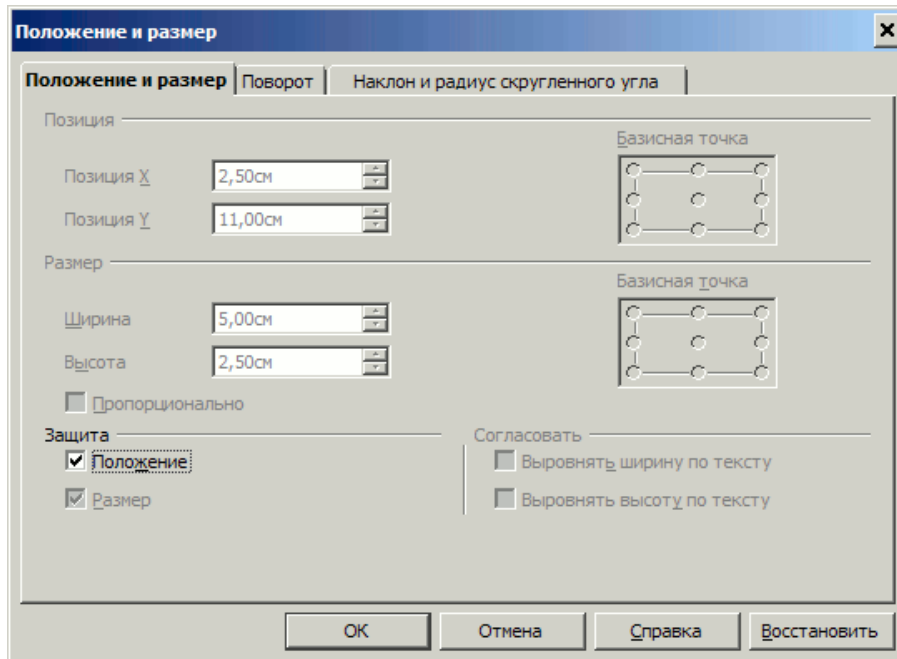


Рисунок 4.15 – Диалог изменения положения и размера элемента

В открывшемся диалоге отмечаем «Защита/Положение» (рисунок 4.15).

Обратите внимание, что этот же диалог может быть использован для установки точных размеров элементов.

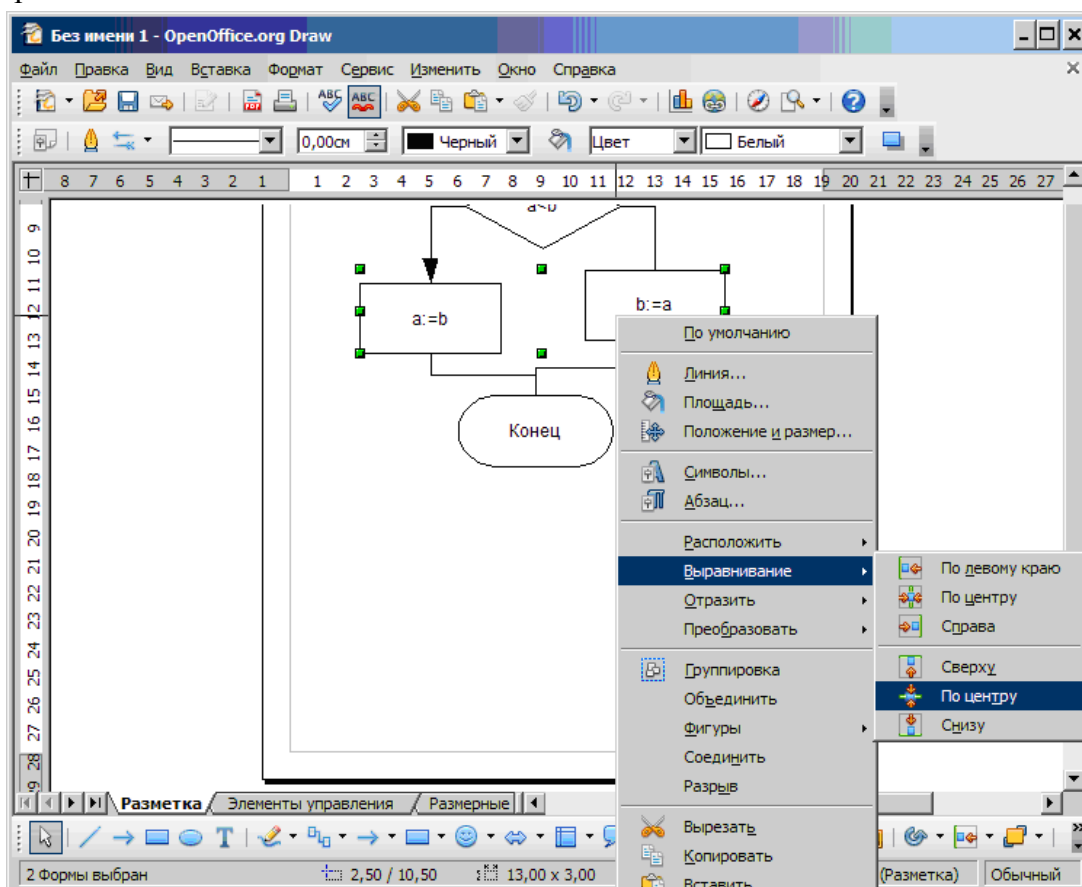


Рисунок 4.16 – Выполнение выравнивания по горизонтали для выделенных элементов

Выделяем все элементы, которые необходимо выделить (процессы присвоения) и в контактном меню выбираем «Выравнивание/По центру (горизонтально)» (рисунок 4.16).

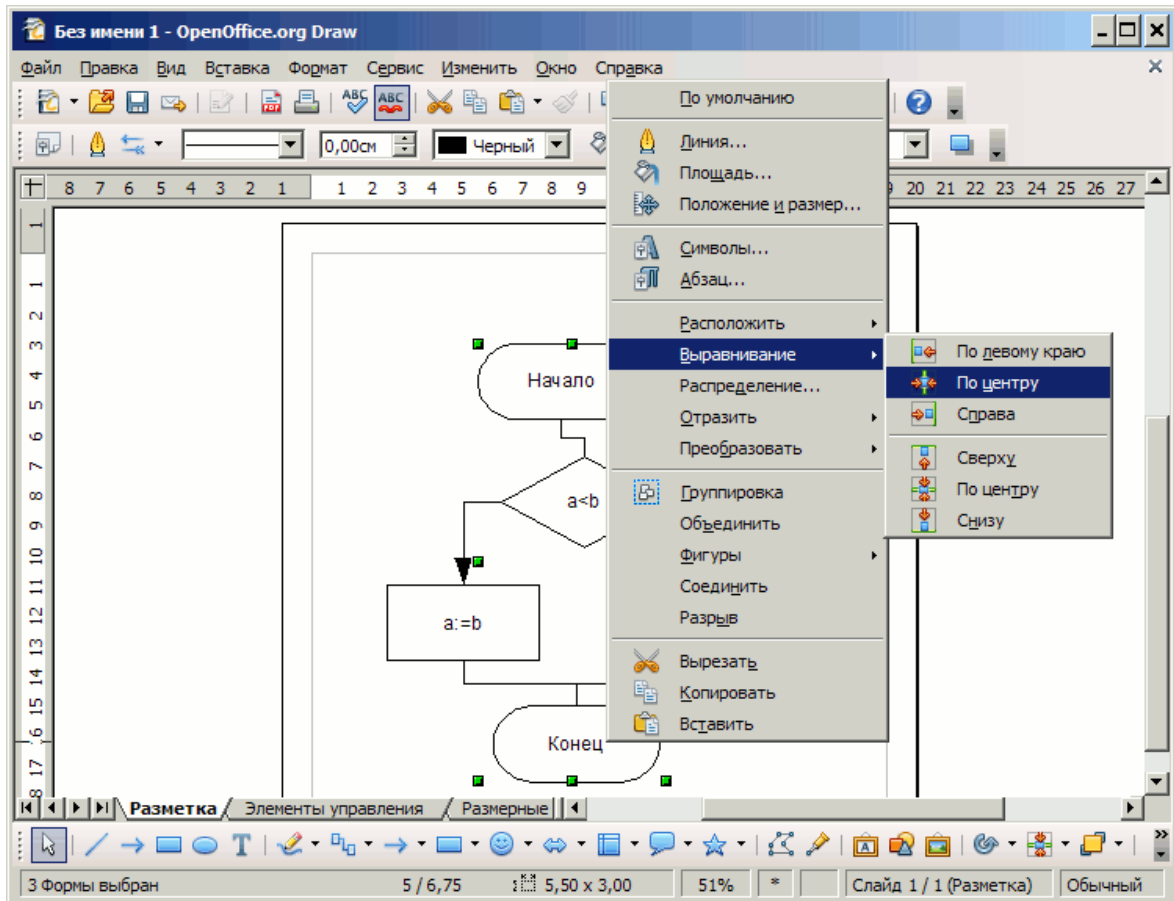


Рисунок 4.17 – Выполнение выравнивания по вертикали для выделенных элементов

Аналогично фиксируем элемент «принятия решения», выделяем вертикально ориентированные элементы и выбираем «Выравнивание/По центру (вертикально)» (рисунок 4.17).

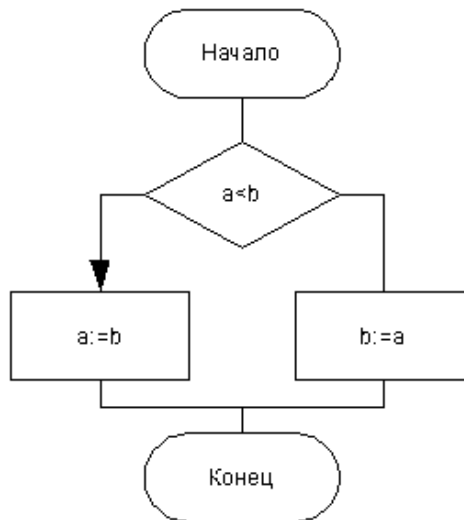


Рисунок 4.18 – Схема алгоритма после выравнивания

В итоге получаем правильно распределенную и выровненную схему алгоритма (рисунок 4.18).

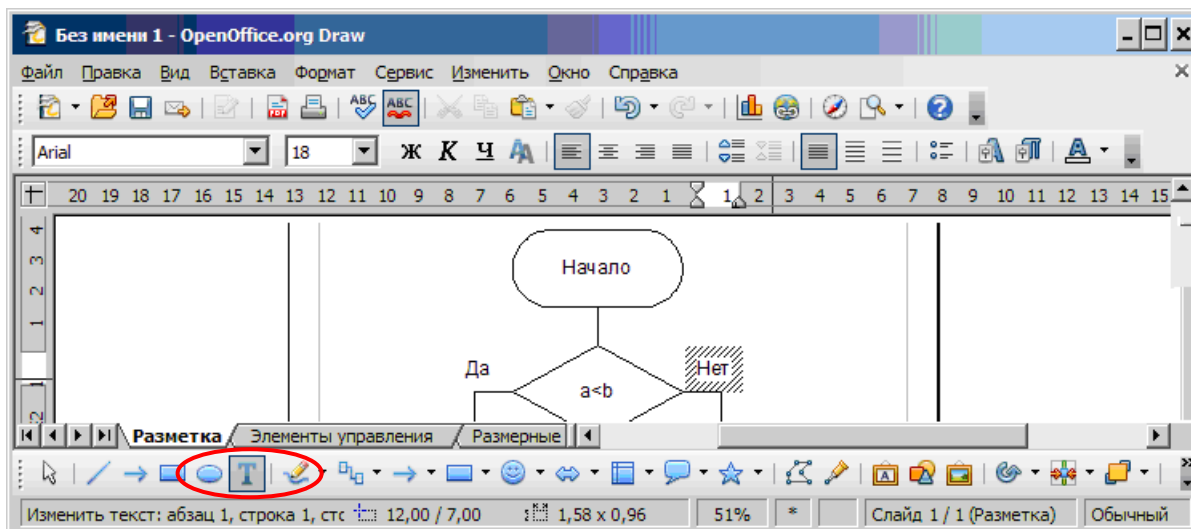


Рисунок 4.19 – Добавление надписей на схему

При помощи кнопки «Текстовые», добавляем необходимые подписи (рисунок 4.19).

4.3 Сохранение диаграммы и экспорт в векторном формате

Внимание! Не забывайте сохранять файл с диаграммой в формате **.odg**, используя меню **Файл/Сохранить**.

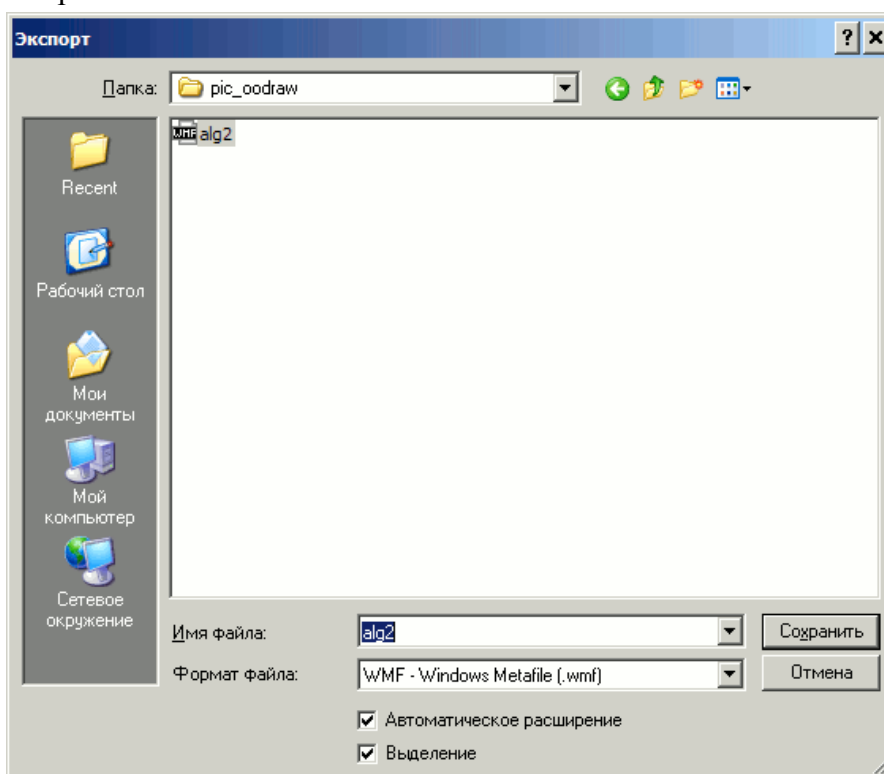


Рисунок 4.20 – Диалог выбора типа сохраняемого файла

Следующим шагом следует экспортировать схему в формате, в котором она может быть вставлена в текстовом редакторе. Для этого выделяем все элементы (CTRL+A) и выбираем меню «Файл/Экспорт». В открывшемся диалоге отмечаем пункт «Выделение» (т.е. экспортировать только выделенную часть диаграммы, а не лист рабочей области в целом). В качестве формата сохранения следует выбрать векторный формат, например «WMF» (рисунок 4.20).

Создаем текстовый документ OpenOffice Writer. Для этого выбираем пункт меню Windows Пуск/Программы/OpenOffice.org/OpenOffice.org Writer. После запуска программы автоматически создается чистый текстовый документ. Предположим, что схему алгоритма необходимо вставить в текст документа. Место вставки рисунка будет определяться положением текстового курсора.

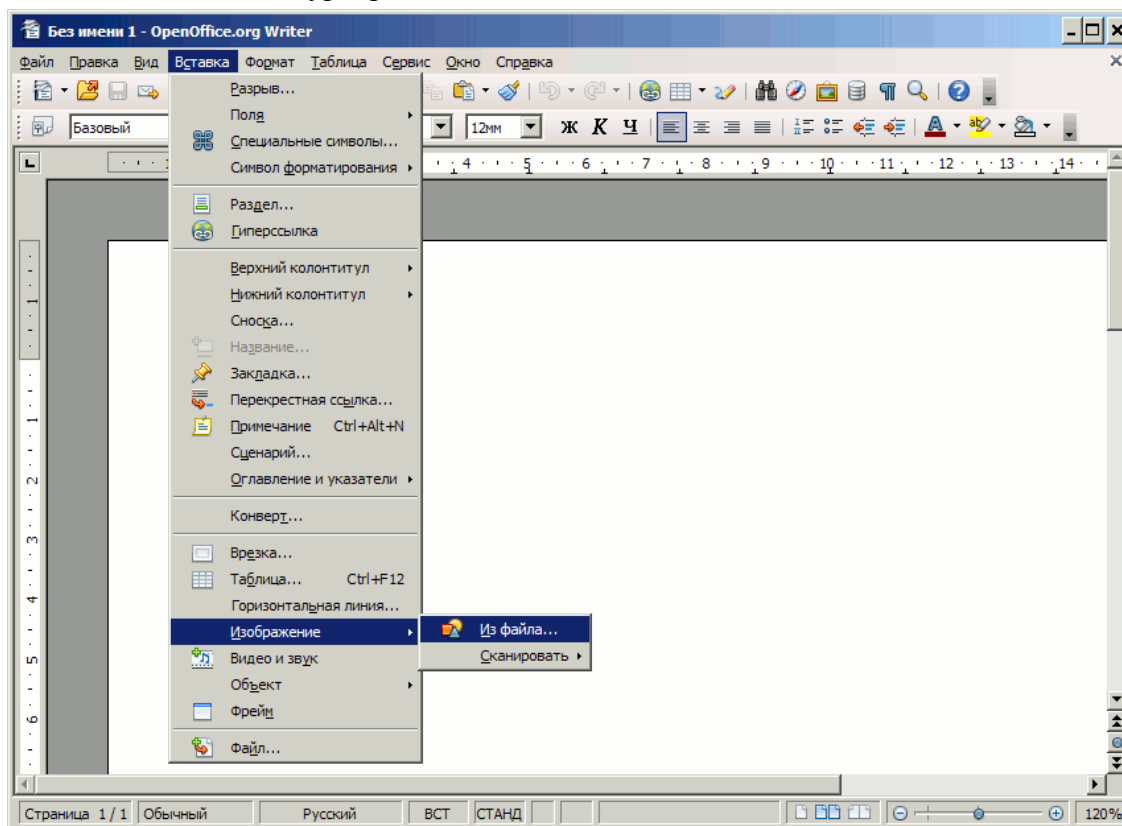


Рисунок 4.21 – Вставка рисунка в документе OpenOffice Writer

В текстовом редакторе (OpenOffice Writer) выбираем пункт меню «Вставка/Изображение/Из файла» и файл, который был ранее получен как результат экспорта диаграммы (рисунок 4.21).

Результат вставки показан на рисунке 4.22.

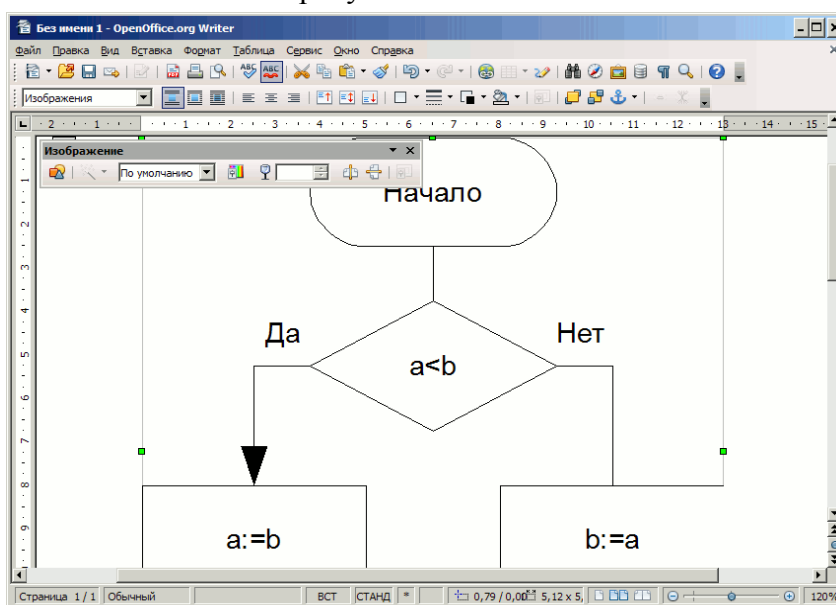


Рисунок 4.22 – Рисунок в текстовом документе

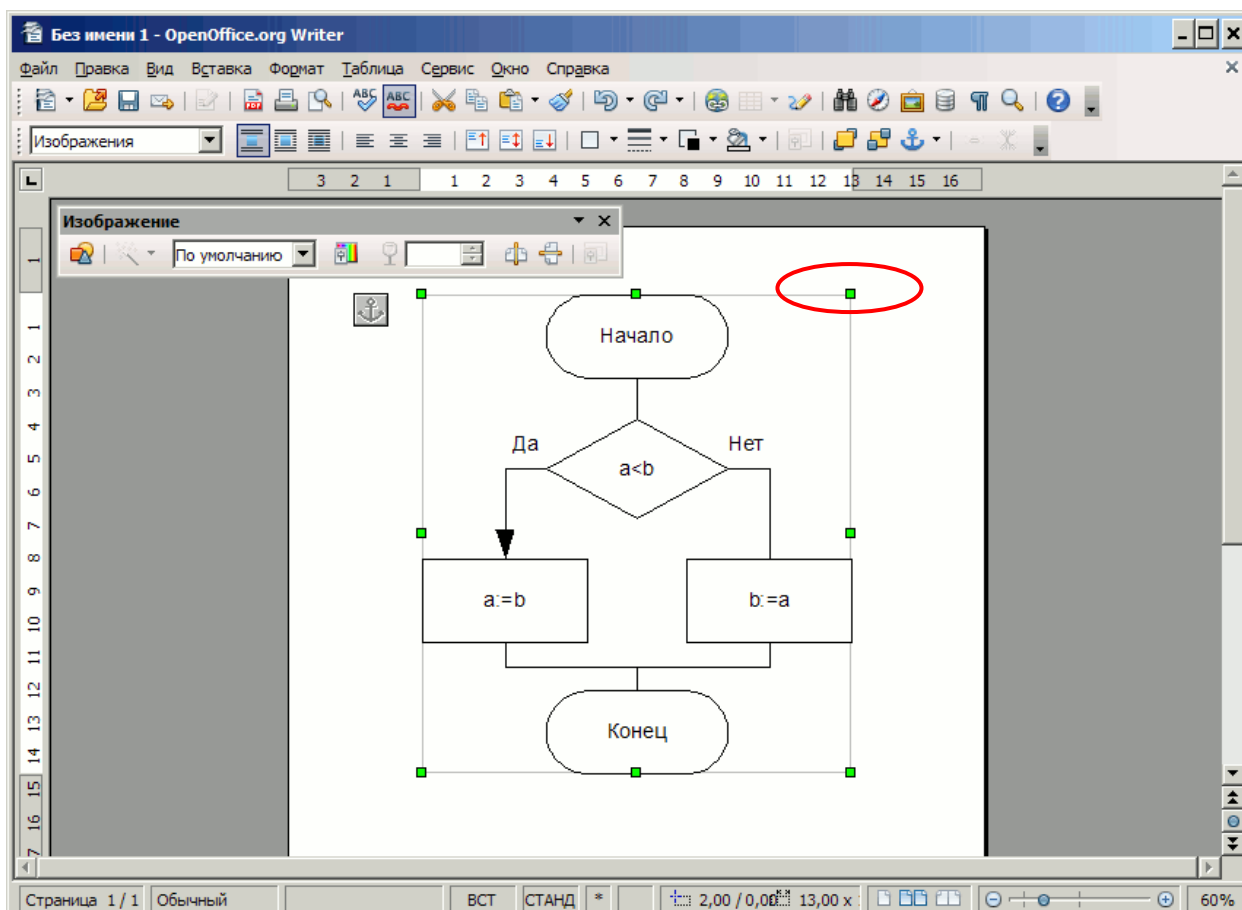


Рисунок 4.23 – Изменение размера рисунка при помощи мыши

Размер вставленного рисунка может быть изменен с использованием опорных точек (рисунок 4.23).

Сохраните файл, содержащий вставленную схему алгоритма.

4.4 Полезные комбинации клавиш в OpenOffice Draw

F4 - Положение и размер.

CTRL+SHIFT+G – сгруппировать выделенные элементы

CTRL+SHIFT+ALT+G – разгруппировать элементы выделенной группы

CTRL+'''' – повысить уровень элемента (при их взаимном наложении)

CTRL+''' – понизить уровень элемента

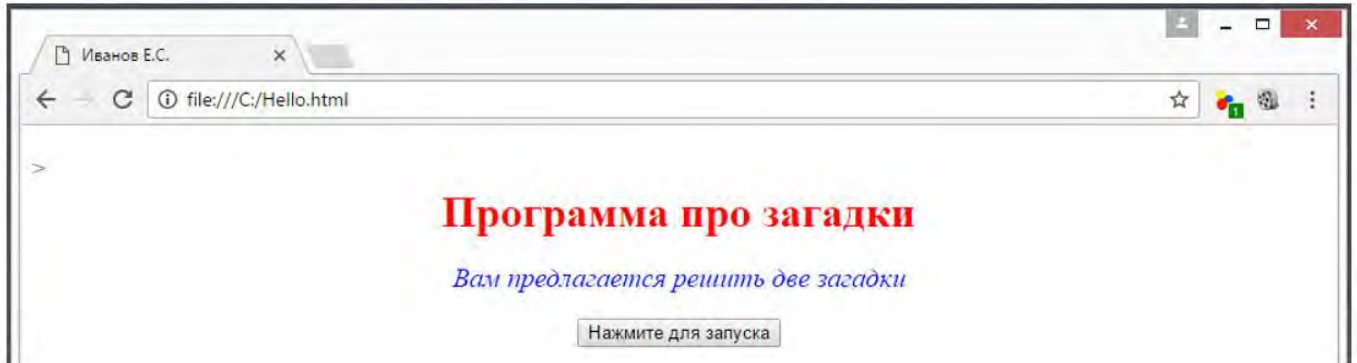
Лабораторная работа №3

Основы HTML и CSS

Основы HTML и CSS: HTML + CSS = сайт.

Задание для выполнения

1. Изучите теоретическое обоснование
2. Создайте веб-страницу на языке HTML по предложенному ниже образцу:



3. При нажатии на кнопку (Нажмите для запуска) должен запуститься программный код на языке JavaScript, реализующий созданный в предыдущем задании алгоритм отгадывания заданных загадок.

Примечания:

- Для оформления должны использоваться стили CSS `.my_style_head` (шрифт 32px, цвет красный, жирный) и `.my_style_body` (шрифт 20px, цвет синий, курсив).
- Для отображения кнопки можно использовать следующий программный код:

```
<form>
  <input type="button" value="Текстнакнопке" onclick="prog()">
</form>
```

- В приведенном примере "prog()" означает вызов функции на языке JavaScript. Сама функция, реализующая алгоритм с загадками должна располагаться после тэга `head` как в приведенном ниже примере:

```
<head>
<meta charset="utf-8">
<script type="text/javascript">
  function prog()
  ...
```

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

HTML+CSS=САЙТ

Мы с вами познакомились с языком программирования JavaScript, но все наши программы выглядели, мягко говоря, не очень красиво. Для того, чтобы сделать веб-страницу более насыщенной давайте познакомимся с языком разметки веб-страниц HTML и языком описания внешнего вида документа CSS. Начнем с HTML

HTML

На самом-то деле, все программки, которые мы с вами писали до этого, были не совсем правильными веб-страницами. Но браузеры так устроены, что, если даже веб-страница написана неправильно, он старается отобразить информацию, так как считает нужным. Давайте посмотрим, как мы можем указать браузеру, что и как выводить.

Наберите в текстовом редакторе следующий текст. Сохраните его под названием: index.html и откройте в браузере просто щелкнув на файле index.html два раза.

```
<!DOCTYPE html>
<html>
<head>

</head>

<body>
  Тело веб-страницы
</body>

</html>
```

Возможно, что у вас выведется текст, а может быть выведутся не понятные символы. От чего это зависит читайте дальше.

Перед вами шаблон веб-страницы. Веб-страница состоит из тегов - ключевых слов заключенных в угловые скобки. Теги бывают парные и одинарные.

`<!DOCTYPE html>` - это указание браузеру, что тип документа, с которым ему придется работать html. Это одинарный тег. Он не требует закрывающего тега.

`<html>` `</html>` - это парные теги внутри которых лежит веб-страница.

Сама веб страница состоит из двух частей: **заголовок и тела**.

Заголовок помещается между тегами `<head>` и `</head>`. Здесь описывает некоторая служебная информация, которая влияет на отображение веб-страницы.

Тело, то есть сама веб-страница, располагается между тегами `<body>` и `</body>`. Большинство текста по описанию веб-страницы помещается сюда.

Давайте добавим в нашу веб-страницу в заголовок указание в какой кодировке вы создаете веб-страницу, заголовок, который будет отображаться на вкладке браузера, и некоторый произвольный текст.

Папа у Васи силен в математике,
Учится папа за Васю весь год,
Где это видано, где это слыхано,
Папа решает, а Вася сдает?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Иванов Иван</title>
```

```

</head>
<body>
  <strong>Папа у Васи силен в математике,</strong><br>
  <em>Учится папа за Васю весь год,</em><br>
  <i>Где это видано, где это слыхано,</i><br>
  <b>Папа решает, а Вася сдает?</b>
</body>
</html>

```

Теги *strong* и *b* - делают шрифт жирным, а теги *em* и *i* - наклонным. Теги *b* и *i* - не рекомендуются к использованию, и здесь продемонстрированы специально, чтобы показать, что есть теги, которые дублируют свое назначение, но одни относятся к более старым версиям HTML (*i* и *b*), а другие к более новым (*strong* и *em*)

Поначалу количество тегов в HTML может навести тоску, но они легко запоминаются, а специальные редакторы, такие как Sublime Text или Brackets имеют встроенную контекстную подсказку по тегам, что делает запоминание тегов еще более легким.

С HTML немного разобрались. Теперь давайте посмотрим, что такое CSS.

CSS

Для примера добавим простенький CSS прямо в заголовок нашей страницы. Для этого между тегами *head* добавим парный тег *style*, а внутри него код CSS.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Иванов Иван</title>
    <style>
      strong
      {
        color:red;
      }
      .my_style
      {
        font-size:30px;
        color:blue;
      }
    </style>
  </head>
  <body>
    <strong>Папа у Васи силен в математике,</strong><br>
    <em class="my_style">Учится папа за Васю весь год,</em><br>
    <i>Где это видано, где это слыхано,</i><br>
    <b>Папа решает, а Вася сдает?</b>
  </body>
</html>

```

Папа у Васи силен в математике,

Учится папа за Васю весь год,

Где это видано, где это слыхано,

Папа решает, а Вася сдает?

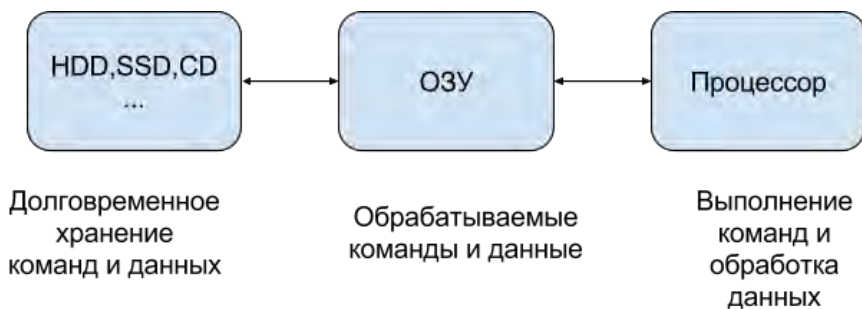
CSS задает стиль отображения информации. Здесь мы продемонстрировали два способа задания стиля. Первый, *strong* - мы описали без точки вначале, что означает, что этот стиль применяется для всех тегов *strong* на этой странице. Второй, *.my_style* с точкой в начале - мы создали класс стиля, который потом можно назначать на веб-странице, что мы и сделали, назначив этот стиль тексту, заключенному между тегами *em*. Сами же стили довольно

простые, в первом случае мы указали, что нужно выводить красным цветом, во втором, что требуется задать размер шрифта в 30 пикселей и выводить информацию синим цветом.

Устройство компьютера с точки зрения программиста

Современные компьютеры, если досконально разбираться в их устройстве, весьма сложны, и чтобы понять, как он функционирует, может потребоваться прочитать не одну книгу. Но откроем вам страшную тайну: программисту, даже профессиональному, не обязательно очень подробно знать его устройство.

Но общую схему функционирования программист, конечно, знать обязан. Она не сложная.



Необходимо познакомиться и понять взаимосвязь всего трех элементов компьютера.

1. Процессор (CPU)
2. Оперативная память (ОЗУ, RAM)
3. Жесткий диск (HDD, SSD)

Процессор

Процессор - это устройство, которое обрабатывает данные и управляет остальными устройствами компьютера. Процессор обладает следующими характеристиками:

- Частота;
- Разрядность;
- Объем кэш-памяти;
- Многоядерность.

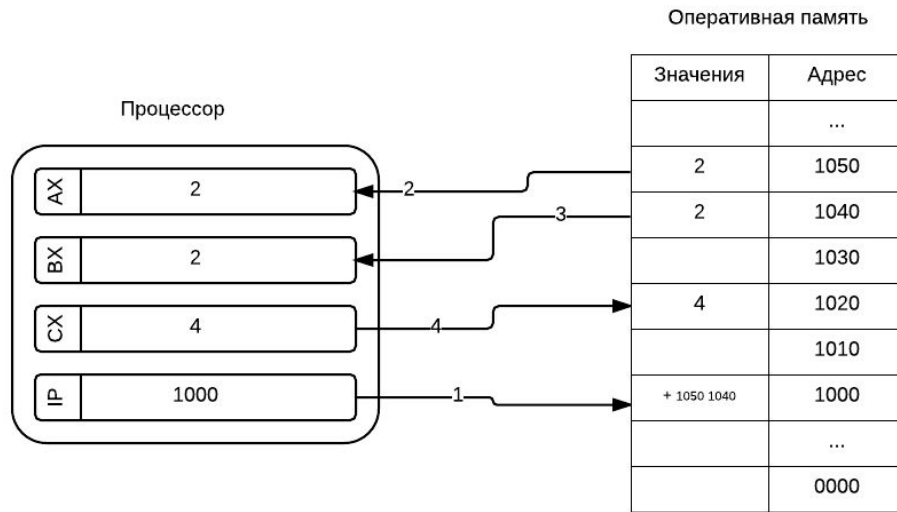
Чтобы понять, на что влияют эти характеристики, сравним процессор с мельницей, которая перемалывает зерно в муку. Тогда частота - это скорость вращения жёрнова, который молит муку. Разрядность - это ширина мола. Чем больше частота и разрядность, тем больше муки можно перемолоть. Или, в нашем случае, данных. Тогда объем кэш-памяти - мешки с зерном, которые лежат рядом с валом, за которыми не нужно бегать в амбар (в ОЗУ). Тогда многоядерность - это мельница с двумя молами для перемалывания.

ОЗУ

В обиходе "оперативка" - это микросхемы, на которых хранятся данные и команды, предназначенные для обработки. Оперативная память так устроена, что работает намного быстрее (в сотни тысяч раз) жесткого диска, но для хранения данных в оперативной памяти требуется электричество. Если питание пропадет даже на доли секунды, данные в оперативной памяти стираются.

Процессор и ОЗУ

Данные и команды, которые обрабатываются в текущий момент, времени хранятся в ОЗУ. Из ОЗУ данные и команды загружаются в процессор, где происходит обработка данных в соответствии с командами. В процессоре присутствуют собственные ячейки памяти, называемые регистрами. Все вычисления производятся в регистрах процессора. Это самая быстрая часть компьютера.



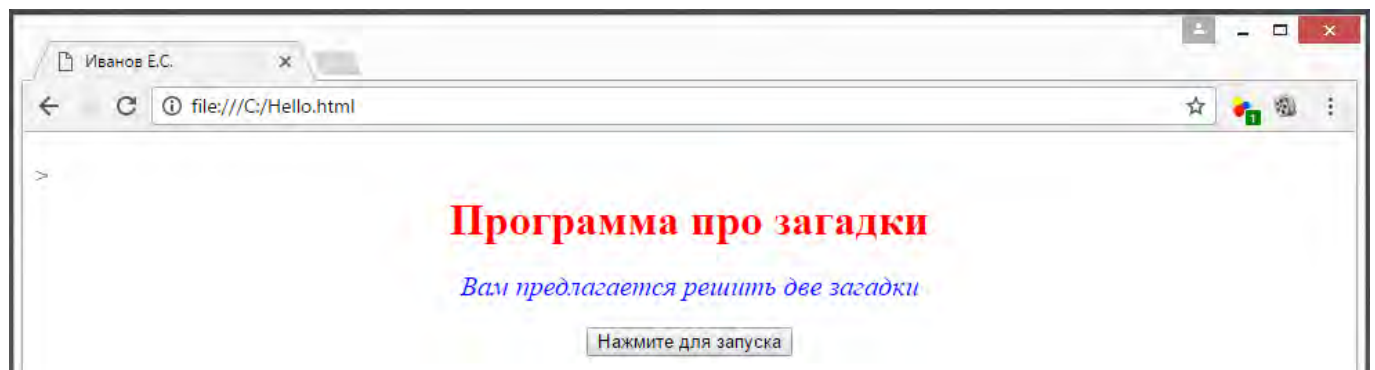
В процессоре существует специальный регистр команд IP. Он хранит в себе адрес ячейки, в которой записана команда, которую необходимо выполнить.

Жесткий диск

Жесткий диск, а также “флешки”, CD, DVD и другие относят к внешним запоминающим устройствам. Это связано с тем, что они играют второстепенную роль в обработке информации. То есть непосредственно в обработке информации эти устройства не участвуют, но при этом они выполняют другую важную роль. Их устройство позволяет им хранить информацию долговременно без источника питания.

Задание для выполнения

1. Создать веб-страницу на языке HTML по предложенному ниже образцу:



2. При нажатии на кнопку (Нажмите для запуска) должен запуститься программный код на языке JavaScript, реализующий созданный в предыдущем задании алгоритм отгадывания заданных загадок.

Примечания:

- Для оформления должны использоваться стили CSS `.my_style_head` (шрифт 32px, цвет красный, жирный) и `.my_style_body` (шрифт 20px, цвет синий, курсив).
- Для отображения кнопки можно использовать следующий программный код:

```
<form>
  <input type="button" value="Текст на кнопке" onclick="prog()">
</form>
```

- В приведенном примере "prog()" означает вызов функции на языке JavaScript. Сама функция, реализующая алгоритм с загадками должна располагаться после тэга `head` как в приведенном ниже примере:

```
<head>
<meta charset="utf-8">
<script type="text/javascript">
  function prog()
  ...
```

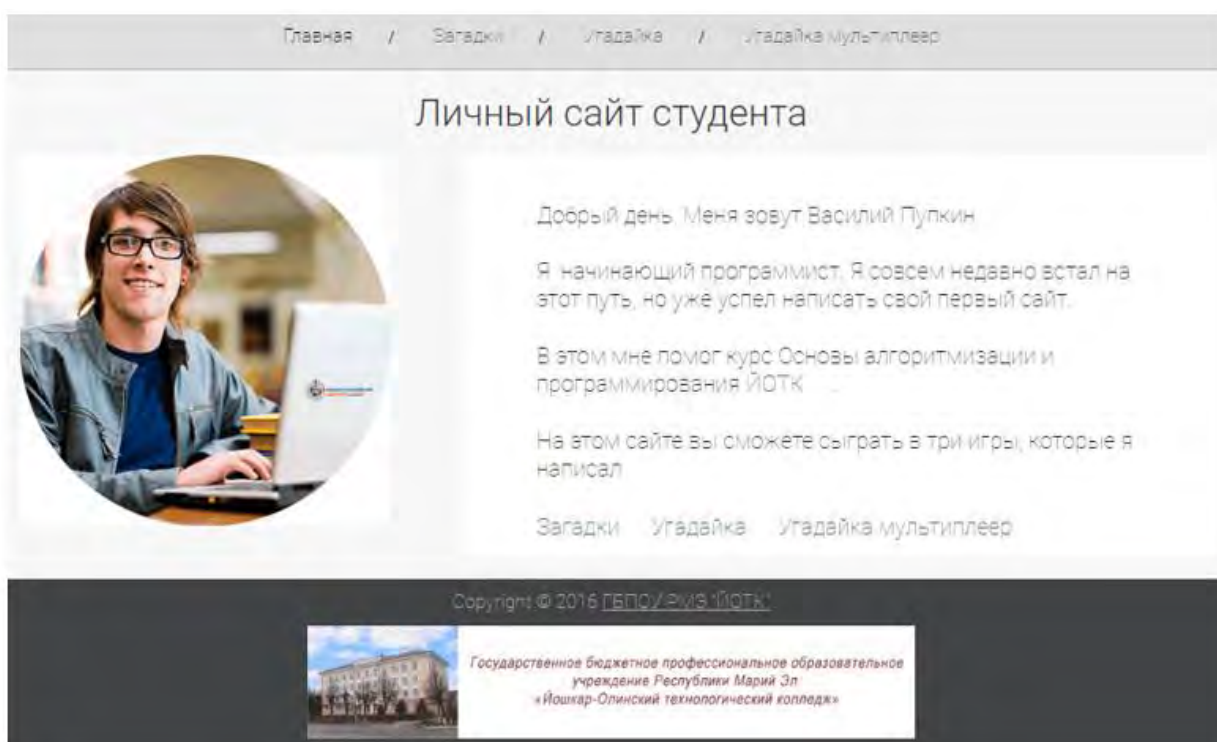
Лабораторная работа №4

HTML&CSS

HTML&CSS: личный сайт, подготовка и вставка картинки, заголовок, ссылки, , <div>, css, стилизация вложенных тэгов.

Задание для выполнения

1. Изучите теоретическое обоснование
2. Внесите необходимые изменения в программный код сайта в соответствии с приведенным образцом:
 - в текст страницы добавлена информация о возможности использования игр;
 - в подвале страницы добавлена фотография колледжа с полным названием;
 - надпись ГБПОУ РМЭ "ЙОТК" сделана ссылкой на сайт www.yotc.ru:



Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

Примечание: ниже приведен готовый листинг файла CSS

```
@import
url(https://fonts.googleapis.com/css?family=Roboto:400,100,100italic,300,300italic,400italic,500,500italic,700,700italic,900,900italic);

* {
  margin: 0;
  padding: 0;
}

body {
  background-color: #f7f7f7;
  font-family: 'Roboto', sans-serif;
}

#header {
  border: 1px solid #b5b7ba;
  background-color: #dfdfdf;
  min-width: 1280px;
  height: 50px;
  text-align: center;
  padding-top: 18px;
}

#header #selected {
  color: #000;
}

#center {
  width: 1280px;
  margin: 0 auto;
}

#header a {
  color: #4d4d4d;
  font-size: 20px;
  font-weight: 100;
  text-align: left;
  text-decoration: none;
  height: 30px;
  line-height: 30px;
  display: inline-block;
}

#header span {
  padding: 0 38px;
}

#content {
  height: 806px;
  min-width: 1200px;
  margin: 0 auto;
}

#content h1 {
  min-width: 1200px;
  text-align: center;
  color: #2e353d;
  font-size: 40px;
  font-weight: 300;
  margin-top: 20px;
  margin-bottom: 20px;
}

#content h3 {
  color: #242424;
  font-size: 35px;
  font-weight: 300;
}
```

```

    text-align: left;
    margin-left: 41px;
    margin-bottom: 52px;
}

#content h5 {
    color: #242424;
    font-size: 24px;
    text-align: left;
}

#content img {
    margin-left: 17px;
}

#content #box_text {
    width: 671px;
    background-color: #ffffff;
    float: right;
    padding: 49px 48px 51px 76px;
}

#content #box_text p {
    color: #242424;
    font-size: 24px;
    font-weight: 100;
    text-align: left;
    margin-bottom: 31px;
}

#content #box_text a {
    color: #40678a;
    font-size: 24px;
    font-weight: 100;
    text-align: center;
    text-decoration: none;
    padding-right: 30px;
}

#content .media {
    margin-right: 0;
}

#content #box {
    background-color: #ffffff;
    width: 1238px;
    height: 446px;
    margin: 0 auto;
    padding-top: 37px;
    padding-left: 38px;
}

#content #box p {
    color: #242424;
    font-size: 24px;
    font-weight: 100;
    text-align: left;
    margin-top: 37px;
    margin-bottom: 30px;
}

#content #box input {
    border: 1px solid #a0a0a0;
    background-color: #f5f5f5;
    width: 457px;
    height: 34px;
    padding-left: 10px;
}

```

```
    font-size:20px;
    font-weight: 100;
    margin-bottom: 12px;
}

#content #box a {
    display: block;
    float: left;
    background-color: #8cbbd3;
    color: #ffffff;
    font-size: 24px;
    font-weight: 300;
    text-align: center;
    text-decoration: none;
    margin-top: 30px;
    padding:11px 51px;
}

#content #box a:hover {
    background-color: #5ba1d3;
}

#footer {
    background-color: #434445;
    min-width: 1920px;
    height: 180px;
    margin: 0 auto;
    text-align: center;
    color: #a4a4a4;
    font-size: 20px;
    font-weight: 100;
    line-height: 50px;
}

#footer a {
    color: inherit;
}
```

HTML & CSS.

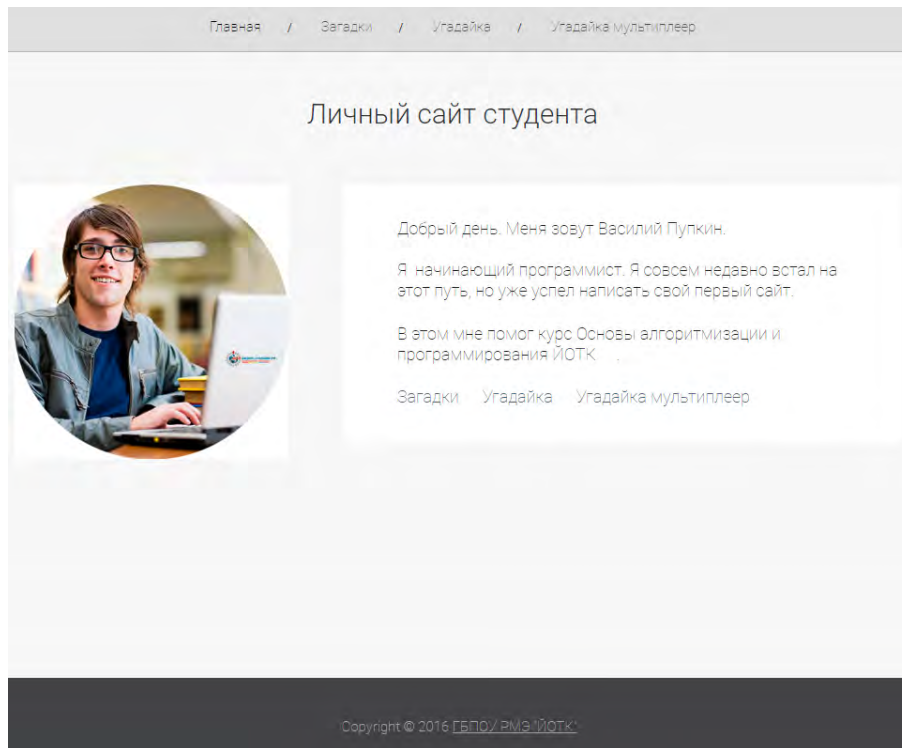
Личный сайт

Когда начинаешь изучать HTML, то, наверное, одним из самых сложных вопросов является: "Какой сайт создать?" Зачастую, так и не ответив на этот вопрос, обучение заканчивается. В лучшем случае пишется страничка в стиле:

"Красные подчеркнутые наклонные буквы на черном фоне"

Давайте не будем мучиться этим вопросом и создадим свой личный сайт, который будет вашим стартом в мире сайтостроительства.

Что примерно должно получиться:



Этот сайт состоит из нескольких частей: верхней части, центральной части и нижней части. Центральная часть разделена на две части. Интересно, что начинающие веб-строители не могут построить сайт, просто потому, что не очень понимают, из каких частей он будет состоять.

Начнем создание.

Центральная часть Подготовка и вставка картинки

Наш сайт будет состоять из нескольких файлов. Поэтому первым делом давайте создадим папку **mysite**, в которой эти файлы будут храниться. Для простоты можно разместить её на рабочем столе. Далее создайте в этой папке папку **img**. В этой папке будут храниться картинки. Поместите туда свою фотографию или другое изображение, которое будет выводиться на месте фотографии. Назовите её **photo.png**. Если вы умеете обрабатывать фотографии, то сделайте размер картинки 390x390. Если нет, ничего страшного, при вставке картинки мы заставим браузер задать нужный размер.

Для вставки картинки используется одинарный тег `image`.

```
<image src="img/photo.png" alt="фотография">
```

Атрибут **src** указывает расположение картинки. Атрибут **alt** нужен для того, чтобы показать какой текст будет выводиться, если вдруг у пользователя отключен вывод картинок, или картинка будет не доступна.

Должно получиться:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title</title>
  </head>
  <body>
    
  </body>
</html>
```

Заголовок

Добавим заголовок. Для этого используется парный тег `<h1>`. Добавим перед картинкой текст:

```
<h1>Личный сайт студента</h1>
```

Должно получиться:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title</title>
  </head>
  <body>
    <h1>Личный сайт студента</h1>
    
  </body>
</html>
```

Информация о себе

Добавим информацию о себе. Для этого используем парный тег `<p>` и одинарный `
`. Добавьте под тегом `img` следующий текст

```
<p>Добрый день. Меня зовут Василий Пупкин.<br>
Я - начинающий программист. Я совсем недавно встал на этот путь, но уже успел написать свой первый сайт.</p>
<p>В этом мне помог курс Основы алгоритмизации и программирования Йошкар-Олинского технологического колледжа.</p>
<p>На этом сайте вы сможете сыграть в три игры, которые я написал.</p>
```

Ссылки

Парный тег `<a>` указывает браузеру, что элемент (этом может быть и текст, и картинка) внутри этого тега будет отображаться как ссылка, и, при щелчке по ссылке, браузер попытается перейти по адресу указанном в атрибуте `href`.

```
<p>В этом мне помог курс Основы алгоритмизации и программирования <a href="http://yotc.ru">Йошкар-Олинского технологического колледжа</a>.</p>
```

Тег `<a>` поддерживает атрибут `target`, который указывает как отобразить страницу при переходе на неё. Например, `target="_blank"` - указание загрузить страницу в новое окно.

```
<p>В этом мне помог курс Основы алгоритмизации и программирования <a href="http://yotc.ru" target="_blank">Йошкар-Олинского технологического колледжа</a>.</p>
```

Верхняя часть

Добавьте перед тегом `<h1>` сразу после `<body>` следующий текст.

```
<a href="#">Главная</a><span></span>
<a href="puzzles.html">Загадки</a><span></span>
<a href="guess.html">Угадайка</a><span></span>
```


`Угадайка мультимплеер`

Атрибут `href="#"` - указание, что при щелчке по этой ссылке никуда переходить не нужно.

``

Парный тег `/>` - это, наверное, самый трудный для понимания тег. Он как бы говорит, что внутри заключен кусочек текста. На самом деле в HTML есть еще теги, которые позволяют задать некоторую логику страницы, `span` один из них. Он не влияет непосредственно на сам вывод, но позволит в дальнейшем элементам, заключенным в этот тег, придать стиль с помощью CSS. С помощью `` и CSS мы добавим промежутки между пунктами меню

Нижняя часть

В нижнюю часть добавим ссылки на будущие веб-страницы и информацию об авторском праве.

```
<a href="puzzles.html">Загадки</a>
<a href="guess.html">Угадайка</a>
<a href="media.html">Угадайка мультимплеер</a>
Copyright © 2016 ГБПОУ РМЭ "ИОТК"
```

Итоговый код должен получиться примерно таким

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title</title>
  </head>
  <body>
    <a href="#">Главная</a><span>/</span>
    <a href="puzzles.html">Загадки</a><span>/</span>
    <a href="guess.html">Угадайка</a><span>/</span>
    <a href="media.html">Угадайка мультимплеер</a>
    <h1>Личный сайт студента</h1>
    
    <p>Добрый день. Меня зовут Василий Пупкин.<br>
    Я - начинающий программист. Я совсем недавно встал
    на этот путь, но уже успел написать свой первый сайт.</p>
    <p>В этом мне помог курс Основы алгоритмизации и программирования
    <a href="http://yotc.ru" target="_blank">ИОТК</a>.</p>
    <p>На этом сайте вы сможете сыграть в три игры, которые я написал:</p>
    <a href="puzzles.html"> Загадки</a>
    <a href="guess.html"> Угадайка</a>
    <a href="media.html"> Угадайка мультимплеер</a>
    Copyright © 2016 ГБПОУ РМЭ "ИОТК"
  </body>
</html>
```

DIV - разделяй и властвуй

Пока сайт выглядит очень невзрачно. Для того, чтобы он стал более красивым, нужно задать стили оформления, но перед этим нужно задать логику сайта с помощью тегов `<div>` и атрибутов `id`.

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="header">
    <a id="selected" href="#">Главная</a><span>/</span>
    <a href="puzzles.html">Загадки</a><span>/</span>
    <a href="guess.html">Угадайка</a><span>/</span>
    <a href="media.html">Угадайка мультимплеер</a>
```

```

</div>
<div id="content">
  <h1>Личный сайт студента</h1>
  <div id="center">
    
    <div id="box_text">
      <p>Добрый день. Меня зовут Василий Пупкин.<br><br>
      Я - начинающий программист. Я совсем недавно встал
      на этот путь, но уже успел написать свой первый сайт.</p>
      <p>В этом мне помог курс Основы алгоритмизации и программирования
      <a href="http://yotc.ru" target="_blank">ЙОТК</a>.</p>
      <a href="puzzles.html"> Загадки</a>
      <a href="guess.html"> Угадайка</a>
      <a href="media.html"> Угадайка мультимплеер</a>
    </div>
  </div>
</div><br>
<div id="footer">
  Copyright © 2016 <a href="http://yotc.ru/" target="_blank">ГБПОУ РМЭ "ЙОТК"</a>
</div>
</body>
</html>

```

Сам тег `div`, так же как и ``, не влияет на вывод информации на экран. Но он позволяет разделить веб-страницу на разделы, на которые можно будет потом влиять с помощью CSS кода. Атрибут `ID` дает название разделам, чтобы по ID можно было обратиться к нужному разделу и изменить его стиль оформления.

Подключим CSS

Добавьте в `<head>` тег загрузки стиля

```
<link rel="stylesheet" href="style.css">
```

Должно получиться примерно следующее:

```

<head>
<meta charset="UTF-8">
<title>Title</title>
<link rel="stylesheet" href="style.css">
</head>

```

Теперь перейдем к созданию файла, который будет влиять на стиль отображения информации на экране

Его величество CSS

Теперь мы подошли к одновременно самой сложной и самой простой части урока. Сложность заключается в большом количестве новой информации, которая на вас обрушится. Простота же заключается в том, что на первых порах не обязательно понимать всё, что вы увидите. CSS не так сложен, как может показаться на первый взгляд. Для начала поймите, что он будет влиять на отображения страницы.

CSS (Cascading Style Sheets) — каскадные таблицы стилей. Для начала слова “каскадные” и “таблицы” можно не запоминать. Важно только “стили”. Действительно, CSS подменяет ранее созданную логику создания сайтов. Первое время создатели сайтов использовали различные теги и их атрибуты, чтобы указать браузеру, как выводить информацию на экран. Например, раньше только тег `<h1>` позволял создать заголовок, а тег `` - задать размер шрифта. Теперь же с помощью тега `<h1>` указывается браузеру, что текст внутри будет заголовком, а как будет выглядеть этот заголовок мы можем задать с помощью таблицы стилей. Можно сказать, что сейчас основное предназначение тегов - создать логику сайта, а уже CSS управляет тем, как этот текст будет выглядеть в браузере.

На заметку. Если же для каких-то тегов мы не задали стиль, то браузер использует свою внутреннюю таблицу стилей.

Структура CSS

CSS - это уже не HTML, хотя и используется совместно. При описании CSS чаще всего используется следующая структура:

```
Название_стиля
{
описание1;
описание2;
...
}
```

Название_стиля - грамотно называть селектором, а описание - свойством.

body и #header

Перед названием стиля могут стоять значки . или #, которые указывают, к чему требуется применить этот стиль.

Например, для тегов можно задать стиль таким образом:

```
body
{
background-color: #555;
font-family: sans-serif;
}
```

Это указание, что для тега **body** сделать фон цветом #555, а шрифт sans-serif

Таким образом можно указать стиль для тегов у которых атрибут id="header":

```
#header
{
border: 1px solid #b5b7ba; background-color: #dfdfff; min-width: 1280px;
height: 50px; text-align: center; padding-top: 18px;
}
```

***. padding и margin**

* - звездочка - это указание применить стиль для всех тегов

```
*{
margin: 0;
padding: 0;
}
```

padding и margin - это отступы. Первое создаёт отступы вокруг содержимого, второе расширяет поле до границы элемента.

Стилизация вложенных тегов

```
#header a {
color: #4d4d4d; font-size: 20px; font-weight: 100; text-align: left; text-decoration: none; height: 30px; line-height: 30px; display: inline-block;
}
```

Здесь мы как бы говорим, "Эй! Все теги <a>, которые имеют id="header", будьте со своим стилем"

@import

Позволяет импортировать содержимое CSS-файла в текущую стилевую таблицу. В нашем примере с помощью этой команды мы подключим набор шрифтов.

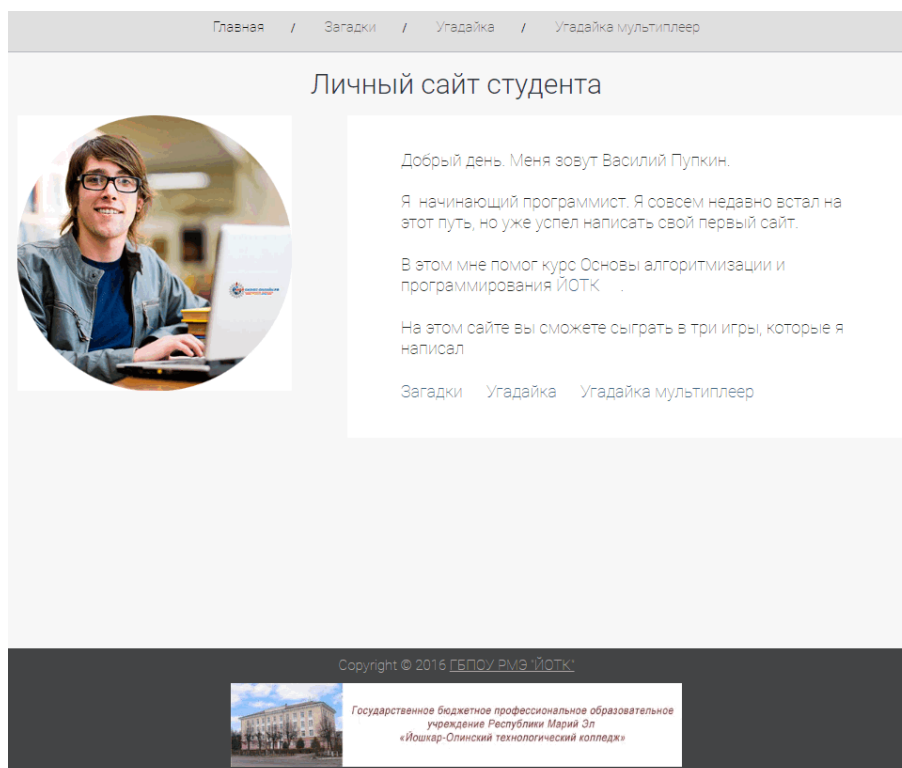
Магия CSS

На этом теоретическое введение закончим и рассмотрим готовый CSS файл. Скопируйте файл с именем `style.css` в папку `site`. После этого обновите в браузере страницу с `html` файлом, и она магическим образом преобразится. Можете изучить содержимое файла и попытаться понять, почему это произошло.

Задание для выполнения

Внести необходимые изменения в программный код сайта в соответствии с приведенным образцом:

- в текст страницы добавлена информация о возможности использования игр;
- в подвале страницы добавлена фотография колледжа с полным названием;
- надпись ГБПОУ РМЭ "ИОТК" сделана ссылкой на сайт www.yotc.ru



Лабораторная работа №5

Циклы

Циклы: знакомство с объектно-ориентированным программированием, цикл *while*, формат записи цикла *while*, цикл *for*, формат записи цикла *for*, цикл *do while*, формат записи цикла *do while*.

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Внесите изменения в игру "Угадай число" для двух игроков. Если используемый браузер некорректно использует *document.write*, переделайте программу для использования *alert*.
3. Доработайте созданный в предыдущем задании веб-сайт, активировав ссылки "Загадки" и "Угадайка" для запуска созданных игровых программ.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

Циклы

Вывод без ОК

Рассмотрим простую программу:

```
<script>
  document.write("Hello!");
</script>
```

Запустите и убедитесь, что в окне браузера появилась надпись *Hello!*

Теперь познакомимся с новым способом вывода информации на экран и заодно начнем (весьма беглое) знакомство с его величеством Объектно-Ориентированным Программированием. До этого момента большинство из вас использовала команду `alert` для вывода информации на экран. Преимущества этой команды в её простоте, но главный недостаток в том, что нужно обязательно щелкнуть на ОК, чтобы продолжить выполнение скрипта. Давайте познакомимся с ещё одной командой вывода на экран, которая лишена этого недостатка. Это команда **document.write**

Ее синтаксис следующий:

```
function(html:string)
```

Это означает, что это команда принимает на вход строку и не возвращает значение. Слово `html` - это подсказка программисту, что это команда может принимать HTML-код, который будет передан браузеру на обработку.

Эту команду нужно писать, указывая в начале слово `document`, что означает, что команда `write` находится в объекте `document`. На самом деле и команда `alert` была в объекте `window`, и мы могли бы писать `window.alert()`, но `window` можно опускать при написании, а `document` нет.

Рассмотрим, что означает подсказка `html`. Наберите программу:

```
<script>
  document.write("<strong>Hello!</strong><br>");
  document.write("I am <em>HTML!</em>");
</script>
```

Здесь мы использовали теги HTML, и браузер их обработал. `` сделал "Hello!" более жирным, а `` HTML наклонным. `
` - это переход на следующую строку.

Цикл while

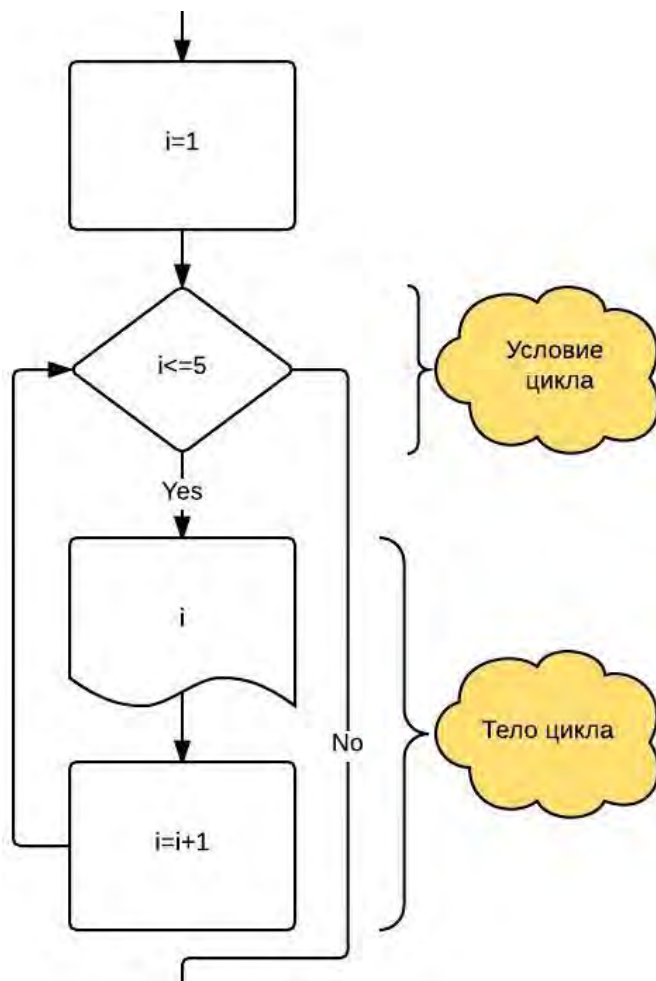
Решим задачу. Вывести на экран числа от 1 до 5. Вот ее решение:

```
<script>
  document.write("1<br>");
  document.write("2<br>");
  document.write("3<br>");
  document.write("4<br>");
  document.write("5<br>");
</script>
```

Мы добились результата, но как быть, если нам нужно вывести 100 чисел? Воспользуемся циклом.

```
<script>
  var i=1;
  while (i<=5)
  {
    document.write(i+"<br>");
    i=i+1;
  }
</script>
```

Эту программу легко переделать, чтобы она выводила хоть 100, хоть 1 000 000 чисел. Для подробного разбора цикла запишем алгоритм программы в виде блок-схемы.



Этот цикл будет выполняться, пока условие цикла истинно. Условие стоит перед телом цикла. Отсюда его название - цикл с предусловием

Формат записи цикла *while*

```
while (условие)
{
    тело цикла
}
```

Особенности:

- Условие может быть сложным;
- Операторные скобки могут отсутствовать, если тело цикла состоит из одного оператора

Подсчитать, сколько положительных чисел на промежутке от *a* до *b*.

```
var a=+prompt();
var b=+prompt();
var i=1;
while (a>0 && a<b)
    i=i+1;
```

Подсчет суммы арифметической прогрессии

Решим задачу: Подсчитать сумму чисел от 10 до *N* с шагом 2 и вывести получившуюся сумму на экран.

```

<script>
  var start=10; // Начало
  var next=start; // Следующее число
  var n=5; // Количество чисел
  var step=2; // Шаг
  var i=1; // Счетчик
  var sum=0; // Сумма
  while (i<=n)
  {
    sum=sum+next;
    i=i+1;
    next=next+step;
  }
  document.write("From:" + start + "<br>");
  document.write("Number:" + n + "<br>");
  document.write("Step:" + step + "<br>");
  document.write("Sum:" + sum + "<br>");
</script>

```

Для лучшего понимания программы заполним таблицу трассировки. Такие таблицы позволяют программистам отлаживать программы без использования компьютера

<i>i<=n</i>	<i>i</i>	<i>sum</i>	<i>next</i>	<i>n</i>	<i>step</i>	<i>start</i>
true	0	0	10	5	2	10
true	1	10	12	5	2	10
true	2	24	14	5	2	10
true	3	36	16	5	2	10
true	4	52	18	5	2	10
true	5	70	20	5	2	10
false	6	70	22	5	2	10

Цикл for

Программистам весьма часто приходится решать задачи, в которых что-то подсчитывается, а для подсчета используется специальная переменная - счётчик. Для таких задач удобно использовать специальный цикл - цикл со счётчиком

Рассмотрим программу:

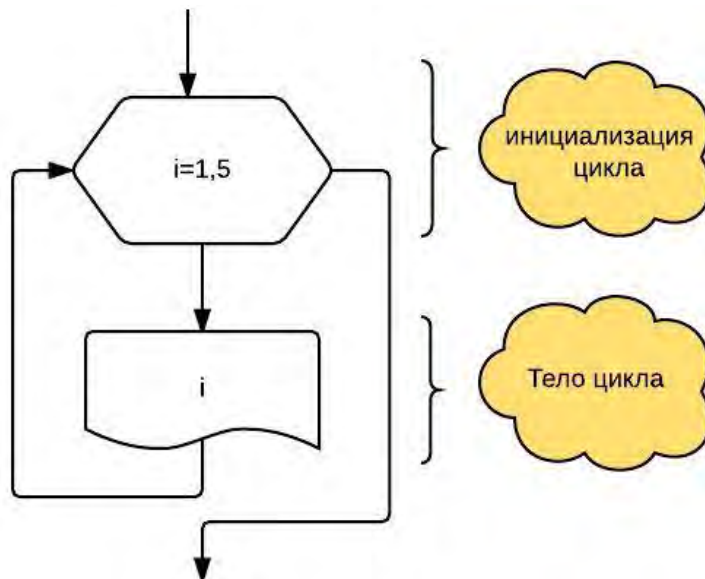
```

<script>
  for(var i=1;i<=5;i++)
  {
    document.write(i+"<br>")
  }
</script>

```

Здесь мы выводим числа от 1 до 5 с использованием цикла for. В этом цикле условие инициализации начальным значением и изменения счетчика описывается непосредственно в заголовке цикла. Для решения многих задач такое описание может быть более удобным, поэтому этот цикл довольно часто заменяет цикл while.

Блок-схема цикла for



Это цикл использует переменную счетчик для подсчета количества итераций, поэтому этот цикл еще называют цикл со счетчиком или цикл с параметром.

В качестве упражнения, можете попробовать самостоятельно переписать программу подсчета арифметической прогрессии с использованием цикла `for` или посмотреть на пример ниже:

```
<script>
var start=10; // Начало
var next=start; // Следующее число
var n=5; // Количество чисел
var step=2; // Шаг
var sum=0; // Сумма
for(var i=1;i<=n;i=i+1)
{
    sum=sum+next;
    next=next+step;
}
document.write("From:"+ start + "<br>");
document.write("Number:" + n + "<br>");
document.write("Step:" + step + "<br>");
document.write("Sum:" + sum + "<br>");
</script>
```

Формат записи цикла for

```
for (переменная = начальное значение; условие; изменение переменной)
{
    тело цикла
}
```

Особенности:

- Условие может быть сложным;
- Операторные скобки могут отсутствовать, если тело цикла состоит из одного оператора.

Цикл do while

Предположим нам нужно попросить пользователя ввести число, но при этом ограничить ввод только положительными значениями. Программа с использованием цикла `while` будет выглядеть вот так:

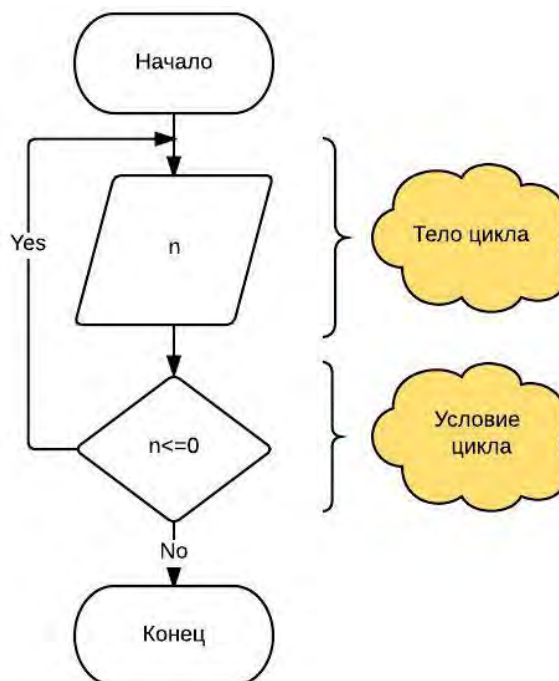
```
<script>
var n=+prompt("Input positive number:");
while(n<=0)
{
    n=+prompt("Input positive number:");
}
```

```
}  
</script>
```

Так как в цикле *while* условие проверяется перед телом цикла, нам придётся сделать ввод данных до входа в цикл, и ещё раз то же самое внутри цикла. Для упрощения решения подобных задач используется ещё одна разновидность циклов: цикл *do while*

```
<script> var n;  
do  
{  
  n+=prompt("Input positive number:");  
}  
while(n<=0)  
</script>
```

Блок-схема программы



Этот цикл так же называется циклом с постусловием, так как условие расположено после тела цикла

Формат записи цикла *do while*

```
do  
{  
  тело цикла  
}  
while (условие)
```

Особенности:

- Условие может быть сложным;
- Операторные скобки { и } обязательны

Пора применить полученные знания для написания вашей первой интерактивной веб-страницы.

Веб-страница игры “Угадай число”

Эта простая игра поможет нам закрепить полученные знания. Смысл игры довольно простой. Компьютер загадывает число в заданном диапазоне чисел от 1 до 100. Пользователь делает попытку угадать число, вводя его с клавиатуры. Если введённое число оказывается больше или меньше загаданного, компьютер сообщает об этом и даёт ещё одну попытку. Если же введённое число совпадает с загаданным, то игра заканчивается, и компьютер сообщает о победе пользователя. Для большего интереса, ограничим количество попыток для угадывания числа 7 попытками. Чтобы поработать с CSS, зададим стили для тегов `body`, `b` и `strong`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Угадай число</title>
    <style>
      body
      {
        font-size: 20px;
        color: brown;
      }
      b
      {
        color: crimson;
      }
      strong
      {
        color: coral;
      }
    </style>
  </head>
  <body>
    <script>
      var min=1;
      var max=100;
      var guessNumber=Math.floor(Math.random() * (max - min + 1)) + min; // Загадываем случайное число
      document.write("Компьютер загадал число от "+min+" до "+max+". Угадайте его за 7 попыток<br>");
      var tryCount=0; // Номер попытки
      var userAnswer;
      do
      {
        tryCount=tryCount+1; // Увеличиваем количество попыток на единицу
        userAnswer=prompt(tryCount + " попытка. Ваш вариант:");
        document.write(tryCount + ". " + userAnswer + " ");
        if (userAnswer>guessNumber)
          document.write("<b>Перелет</b><br>");
        if (userAnswer<guessNumber)
          document.write("<strong>Недолет</strong><br>");
      }
      while(userAnswer != guessNumber && tryCount<7);
      if (userAnswer == guessNumber)
        document.write("Поздравляем! Вы угадали!");
      else
        document.write("Вы проиграли! Попробуйте еще раз");
    </script>
  </body>
</html>
```

Задание для выполнения

1. Дописать игру “Угадай число” для двух игроков. Если используемый браузер некорректно использует `document.write`, переделать программу для использования `alert`.
2. Доработать созданный в предыдущем задании веб-сайт, активировав ссылки “Загадки” и “Угадайка” для запуска созданных игровых программ.

Лабораторная работа №6

Функции, DOM

Функции, DOM: функции, упрощение логики программы, рекурсии, рекурсивный факториал, Document Object Model (DOM), input, события (events).

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Изменить предложенную программу для использования пяти загадок и разместить ее на сайте.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

Разработка веб-сайта с играми. Функции.

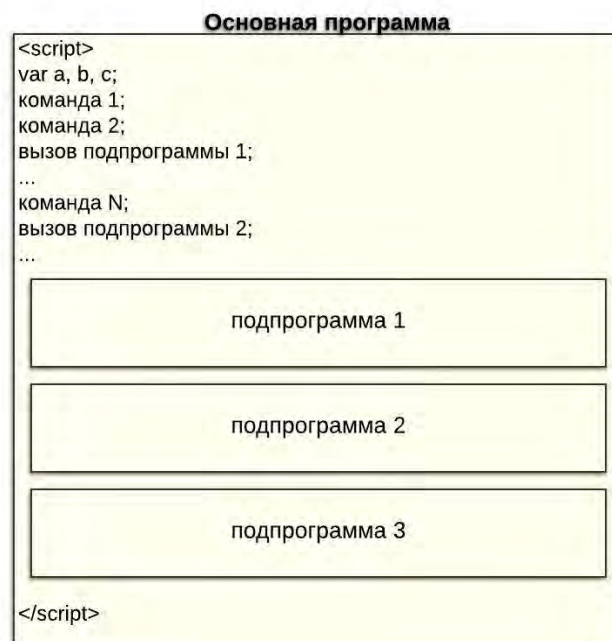
Функции. Рекурсии. Разработка интерактивных игр на сайте.

Функции

Функции — это рабочие лошадки JavaScript. Функции сами по себе играют те роли, которые в других языках исполняются самыми разными средствами: процедурами, методами, конструкторами, классами и модулями. После освоения функций вы овладеете существенной частью JavaScript.

Функция - это подпрограмма

Программисты при написании своих программ постоянно используют функции. Одной из главных целей использования функций является упрощение программирования. Действительно, ведь при написании большой программы часто возникают случаи, когда один и тот же код нужно выполнять много раз. В этом случае повторяющийся код может быть оформлен в виде подпрограммы внутри программы только со своим собственным именем. Такие программы внутри программы в общем случае называются подпрограммами, а в разных языках программирования эти подпрограммы могут носить разное название: функции, процедуры, методы - в зависимости от назначения подпрограммы.



На самом деле, когда мы писали `alert` или `document.write` и другие команды, мы уже использовали стандартные функции JavaScript.

Скажем привет функциям

Наберите программу, которая показывает, как описать функцию, и как её вызвать. В отличие от некоторых языков программирования, описание функции может идти после её вызова.

```
<script>
  SayHello();
  function SayHello()
  {
    alert("Hello!");
  }
</script>
```

`SayHello()` - это вызов функции. Отличительной особенностью функций от переменных является наличие скобок после её названия.

```
function SayHello()
{
  alert("Hello!");
}
```

Это описание функции. Описание функции начинается с ключевого слова **function**, за которым идет название функции, скобки, в которых могут быть параметры функции, и фигурные скобки, в которых заключено тело функции.

Параметры функции

Мы написали простую функцию, которая выводит на экран приветствие. Но что если нам нужно, чтобы в приветствии так же было имя человека с которым мы хотим поздороваться? Писать функции с разными названиями? Для этих целей служат параметры функции.

```
<script>
  SayHello("Fill");
  SayHello("Fred");
  function SayHello(name)
  {
    document.write("Hello!" + name + "<br>");
  }
</script>
```

Если мы хотим, чтобы была возможность передать данные из основной программы в функцию, то можно при описании функции в скобках перечислить названия параметров функции. Теперь при вызове функции мы можем в скобках писать данные, которые будут переданы через параметры внутрь функции. Можно воспринимать параметры как специальные переменные, необходимые для передачи данных внутрь функции. Можно описать сколько угодно много параметров, перечисляя их через запятую.

Функции вычисляют и возвращают значения

Довольно часто необходимо, чтобы главная программа могла получить результат работы функции. В этом случае можно использовать механизм возврата значения из функции. Тогда сама функция выступает в роли переменной, и мы можем использовать её возвращаемое значение.

Пример функции, возвращающей квадрат числа x :

```
<script>
  alert(Power2(5));
  function Power2(x)
  {
    return x*x;
  }
</script>
```

Упрощение логики программы

Как уже было сказано, функции позволяют сделать программу более читаемой. Давайте рассмотрим это на примере решения следующей задачи.

Написать программу: Определить значение $z = \max(a, 2*b) * \max(2*a - b, b)$, где $\max(x, y)$ – максимальное значение из чисел x, y .

Решение без функции	Решение с функцией
<pre>a=parseInt(prompt("a:")); b=parseInt(prompt("b:")); max1=a; if (2*b>max1) max1=2*b; max2=2*a-b; if (b>max2) max2=b; z=max1*max2; alert(z);</pre>	<pre>a=parseInt(prompt("a:")); b=parseInt(prompt("b:")); z=Max(2*b,a)*Max(2*a-b,b); alert(z); function Max(x,y) { if (x>y) return x; else return y; }</pre>

Не знаю, как вам, но нам кажется, что с функцией программа становится более простой для понимания.

Рекурсии

Рекурсия - это вызов функции самой себя. Рассмотрим простой пример:

```
function Print10(n)
{
  document.write(n+"<br>");
  if (n<10) Print10(n+1);
}
Print10(1);
```

Здесь функция вызывает саму себя, пока $n < 11$. Таким образом на экран выведется последовательность чисел от 1 до 10.

Рекурсия является альтернативой циклам, и с её помощью часто можно написать весьма элегантные алгоритмы решения задач.

Задание

Попробуйте переместить `document.write` после условия. Объясните получившийся результат.

Рекурсивный факториал

Факториал $F(n)$ это произведение чисел от 1 до n . Например, $F(5)=1*2*3*4*5=120$.

Факториал хорошо решается с помощью рекурсии, так как факториал можно описать рекуррентно:

$F(n)=F(n-1)*n, n>1$

$F(0)=1, n=0$. Где n - целые числа

Если мы можем построить рекуррентное соотношение, то это рекуррентное соотношение легко реализуется с помощью рекурсии

```
alert( F(5) );
function F(n)
{
  if (n == 0)
    return 1;
  else
    return F ( n - 1 ) * n;
}
```

При этом вычисления происходят в следующем порядке:

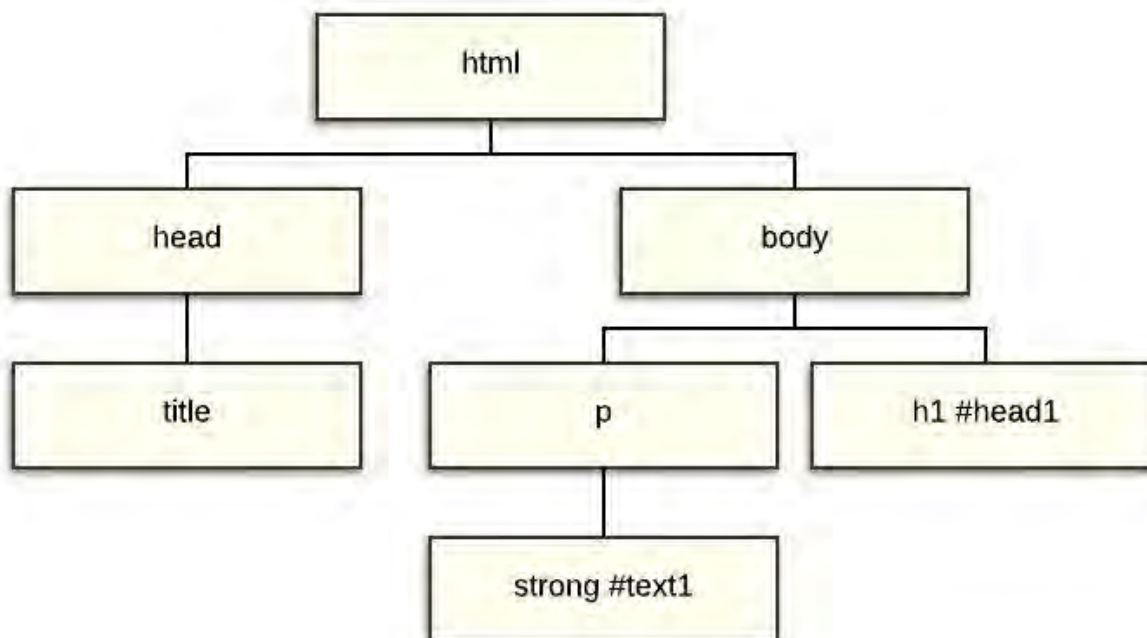
1. $F(5)=F(4)*5$	7. $F(1)=1*1=1$
2. $F(4)=F(3)*4$	8. $F(2)=1*2=2$
3. $F(3)=F(2)*3$	9. $F(3)=2*3=6$
4. $F(2)=F(1)*2$	10. $F(4)=6*4=24$
5. $F(1)=F(0)*1$	11. $F(5)=24*5=120$
6. $F(0)=1$	

DOM, который построим мы

Рассмотрим небольшой html документ:

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Пример документа</title>
  </head>
  <body>
    <h1 id="head1">Заголовок</h1>
    <p>Параграф с <strong id="text1">очень важным </strong>текстом</p>
  </body>
</html>
```

Когда браузер считывает веб-страницу, в памяти строится схема. Для нашей страницы схема будет выглядеть так. Если у элемента есть идентификатор, то он начинается с решетки.



Эта структура называется DOM (Document Object Model - объектная модель документа). Мы можем обращаться к этим элементам с помощью Javascript и считывать или изменять содержимое веб-страницы.

Например:

```
<html>
  <head>
    <title>Пример документа</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1 id="head1">Заголовок до изменения</h1>
    <p>Параграф с <strong id="text1">очень важным </strong>текстом</p>
    <script>
      var h = document.getElementById("head1"); // Находим элемент в DOM
      h.innerHTML = "Измененный заголовок"; // Обращаемся к свойству элемента
    </script>
  </body>
</html>
```

Здесь мы находим с помощью специальной функции `getElementById` элемент с идентификатором `head1`. У большинства элементов есть свойства (переменные, которые описывают состояние элемента), и мы используем свойство `innerHTML`, чтобы изменить внутренний текст элемента.

Страница с загадками

Давайте закрепим полученные знания. Для этого создадим интерактивную страничку с загадками.

Игра в загадки

Отгадай загадки!

Сто одежек, и все без застежек

Зимой и летом одним цветом

Ответить

input

Познакомимся с новым тегом `<input>`. Этот тег позволяет выводить на страницу различные элементы ввода информации.

Например:

```
<html>
  <head>
    <title>Пример документа</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h3>Отгадай загадки!</h3>
    <p>Сто одежек и все без застежек</p>
    <input type="text" id="puzzle1"/>
    <br><br><br>
    <p>Зимой и летом одним цветом </p>
    <input type="text" id="puzzle2"/>
    <br><br><br>
    <input type="button" value="Ответить" />
  </body>
</html>
```

Программа выведет на экран текст загадки, поля, куда пользователь может вводить ответы и кнопку с надписью "Ответить".

Мы сделали возможность вводить ответы, теперь давайте сделаем возможность их проверять. Для этого нам потребуется познакомиться с событиями.

События или events

В Javascript функции играют ещё большую роль, чем в других языках. Дело в том, что они позволяют реализовать механизм событий.

События - это действия, происходящие в браузере: щелчки мышью, нажатия клавиш, перемещения указателя мыши, загрузка рисунка и др. Самые различные события могут влиять на дальнейшее поведение браузера.

Программы, которые позволяют определять события и выполнять соответствующие им действия, называются **обработчиками событий**.

Заставим кнопку реагировать на нажатие.

```
<input type="button" onClick="alert('Событие!)" value="Ответить" />
```

Обратите внимание, что нужно использовать разные кавычки в тексте `onClick="alert('Событие!)"`. Теперь при нажатии на кнопку (возникновения события `click`) будет выполняться команда `alert`.

Но чаще при возникновении событий нужно выполнить несколько команд. В этом случае лучше использовать функции:

```
<input type="button" onclick="Click1()" value="Ответить" />
<script>
  function Click1()
  {
    alert("События удобнее обрабатывать");
    alert("с помощью функций!");
  }
</script>
```

Скрипт с функцией обработки события может быть описан как до, так и после элемента, в котором назначается обработчик события.

Итоговая программа

Функция checkAnswer

Функция `checkAnswer` позволит упростить проверку значений. В функцию мы передаем `id` элемента и правильный ответ. Функция находит элемент по `id` и запоминает значение элемента в переменной `userAnswer`. Далее мы сравниваем `userAnswer` с правильным ответом. Если значения переменных совпадают, то функция возвращает истину, иначе функция возвращает ложь.

```
function checkAnswer(id, trueAnswer)
{
  var userAnswer = document.getElementById(id).value;
  if (userAnswer === trueAnswer)
    return true;
  else
    return false;
}
```

Функция checkAnswers

В этой функции мы множество раз обращаемся к функции `checkAnswer` и передаем ей `id` элемента и правильный ответ. Каждый раз, когда `checkAnswer` возвращает истину, то переменная `goodAnswers` увеличивается на единицу. В конце мы анализируем значение `goodAnswers`.

```
function checkAnswers()
{
  var goodAnswers = 0;
  if (checkAnswer('puzzle1', 'капуста')===true)
    goodAnswers++;
  if (checkAnswer('puzzle2', 'елка'))
    goodAnswers++;
  if (goodAnswers === 0)
    alert("Вы ничего не угадали");
  else
    alert("Количество правильных ответов: " + goodAnswers);
}
```

Возможный вариант текста всей программы веб-страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <script>
      var result = 0;
      function chekAnswer(textBoxId, answer)
      {
        var userAnswer = document.getElementById(textBoxId).value;
        if(userAnswer === answer)
          result++;
      }
    </script>
  </head>
  <body>
    <input type="button" onclick="chekAnswer('textBox1', 'капуста')" value="Проверить" />
  </body>
</html>
```

```

function chekAnswers()
{
  chekAnswer("userAnswer1", "капуста");
  chekAnswer("userAnswer2", "елка");
  if(result == 2)
    alert('Вы угадали все загадки') ;
  else if(result == 1)
    alert('Вы угадали только одну загадку') ;
  else
    alert('Вы ничего не угадали');
}
</script>
</head>
<body>
  <h3>Игра в загадки</h3>
  <h5>Отгадай загадки!</h5>
  <p>Сто одежек, и все без застежек</p>
  <input type="text" id="userAnswer1">
  <p>Зимой и летом одним цветом</p>
  <input type="text" id="userAnswer2">
  <br><br>
  <input type="button" onClick="chekAnswers()" value="Ответить"/>
</body>
</html>

```

Домашнее задание

1. Изменить программу для использования пяти загадок и разместить ее на сайте.

Лабораторная работа №7

Регистрация на хостинге, знакомство с PHP

Регистрация на хостинге, основы PHP: регистрация сайта на хостинге, знакомство с PHP.

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Разместите созданный ранее сайт на хостинге и проверьте его работу.
3. Замените в программных файлах код JavaScript на код PHP и проверьте работу сайта после внесенных изменений.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Листинг разработанной программы.
5. "Скриншоты" рабочих окон программы.
6. Построчное описание разработанной программы.

Регистрация на хостинге и публикация сайта

Пришло время опубликовать наш сайт, чтобы его могли увидеть другие пользователи. Для этого нужно выбрать хостинг-провайдера, то есть компанию, которая возьмет на себя ответственность за хранение вашего сайта у себя на сервере.

Таких компаний довольно много. Существуют как платные, так и бесплатные провайдеры. Бесплатные, как правило, показывают свою рекламу или навязывают собственные системы управления сайтом. В качестве примера мы покажем, как зарегистрироваться и выложить свой сайт на платном хостинге hostland.ru

Инструкция

1. Переходим на сайт <https://www.hostland.ru/>. Щелкаем "Получить хостинг на 30 дней бесплатно"
2. Вводим свой рабочий e-mail и код проверки и нажимаем "Зарегистрироваться и получить 30 дней хостинга бесплатно".
3. После этого для вас будет создан аккаунт и будет предоставлен 30-дневный тестовый период для публикации сайта. На почту придет письмо с параметрами доступа. Там же будет ссылка на панель управления и логин с паролем для входа в панель управления.
4. Заходим в панель управления. На вкладке "Домены" подключаем какое-нибудь доменное имя. Например: testsite2016.ru. Подключение домена происходит до 30 минут, но обычно нужно подождать 2-5 минут, чтобы произошло подключение. Хотя это доменное имя может реально не существовать и по нему ни вы, ни кто-либо ещё не сможет подключиться к вашему сайту, пользователю автоматически будет выдано техническое имя (оно будет отображено в списке доменов). Имя вида: <http://testsite2.ru.hostNNNNNN.servNN.hostland.pro>. По этому техническому домену уже можно работать с сайтом из любого браузера (без регистрации домена testsite2016.ru)
5. Теперь можно переключиться на вкладку Файлы/FTP. Там будет автоматически создана папка с названием вашего домена. Можно просто зайти в папку и перетащить в неё ваши html, css и js файлы. При наборе технического имени домена в адресной строке браузера пользователь будет перенаправлен в эту папку.
6. Перейти по техническому адресу и убедиться, что сайт заработал.

*PHP

Если вы смогли пройти предыдущие этапы, то у вас есть хостинг с поддержкой PHP, и можно познакомиться с этим прекрасным языком программирования.

Как и JavaScript, язык PHP - скриптовый язык. Но, в отличие от JavaScript, скрипты выполняются ещё на стороне сервера и позволяют формировать страницы ещё до отправки их пользователю. Что использовать, JavaScript или PHP, зависит от задач, и часто оба языка используются совместно.

Создадим страницу с загадками, реализованную с помощью PHP.

Страница с загадками на PHP

Для того, что бы сервер мог обработать данные от нашей страницы добавим на нее элемент форма. Форма позволяет задавать с помощью атрибута action адрес программы или документа, который обрабатывает данные формы. Атрибут method позволяет задать способ передачи данных от формы на сервер.

Для демонстрации мы возьмем упрощенный вариант страницы [puzzle](#) и сохраним его под именем [puzzle.html](#)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Личный сайт студента</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h3>Игра в загадки</h3>
    <form action="puzzle.php" method="GET">
      <p>Не ездок, а со шпорами, не будильник, а всех будит.</p>
      <input type="text" name="userAnswer1">
      <p>Сижу верхом, не ведаю на ком.</p>
      <input type="text" name="userAnswer2">
      <br><br>
      <input type="submit" value="Ответить">
    </form>
  </body>
</html>
```

Когда форма отправляется на сервер, управление данными передается программе, заданной атрибутом action тега <form>. Предварительно браузер подготавливает информацию в виде пары «имя=значение», где имя определяется атрибутом name тега <input>, а значение введено пользователем или установлено в поле формы по умолчанию с помощью атрибута value.

Для отправки данных достаточно нажать клавишу Enter в одном из полей формы, но мы разместили элемент input типа submit - кнопку, которая предназначена для отправки данных формы на сервер.

А это веб-страница с PHP кодом **puzzle.php**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Личный сайт студента</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h3>Игра в загадки</h3>
    <?php
      $userAnswer = $_GET['userAnswer1'];
      $score = 0;
      if ($userAnswer == 'петых' || $userAnswer == 'Петых'){
        echo "1 - Правильно!";
        $score++;
      } else {
        echo "1 - Не правильно.";
      }
      echo '<br />';
      $userAnswer = $_GET['userAnswer2'];
      if ($userAnswer == 'шапка' || $userAnswer == 'Шапка'){
        echo "2 - Правильно!";
        $score++;
      } else {
        echo "2 - Не правильно.";
      }
      echo '<br /><br />';
      echo "Вы отгадали: " . $score . " загадок.";
    ?>
  </body>
</html>
```

Как видите, это обыкновенная веб-страница, только часть страницы занимает php-скрипт, который будет формировать содержимое веб-страницы в зависимости от полученных данных.

Давайте разберем содержимое скрипта подробнее:

```
$userAnswer = $_GET['userAnswer1'];
$score = 0;
```

Все переменные в PHP начинаются со знака \$.

\$userAnswer - это переменная, которая запоминает данные из массива \$_GET. \$_GET - это глобальный массив сервера с PHP, в который поместились данные с формы. И мы обратились к элементу массива с ключом 'userAnswer1', так как в нашей форме был элемент с именем 'userAnswer1'. В переменную \$score мы поместили количество угаданных загадок. Далее:

```
if ($userAnswer == 'шапка' || $userAnswer == 'Шапка'){
  echo "2 - Правильно!";
  $score++;
} else {
  echo "2 - Не правильно.";
}
```

echo - команда вывода текста. То есть в этом месте на веб-странице будет вставлен текст, который идет после echo. Текст может содержать теги. Остальное знакомо по JavaScript.

Для того, чтобы этот пример работал, страницы следует размещать на хостинге с поддержкой PHP.

Лабораторная работа №8

Основы PHP

Установка и настройка компонентов веб-сервера (XAMPP), установка редактора веб-страниц Brackets.

Задание для выполнения

1. Изучить теоретическое обоснование.
2. Установить и настроить компоненты веб-сервера XAMPP и редактор веб-страниц Brackets.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

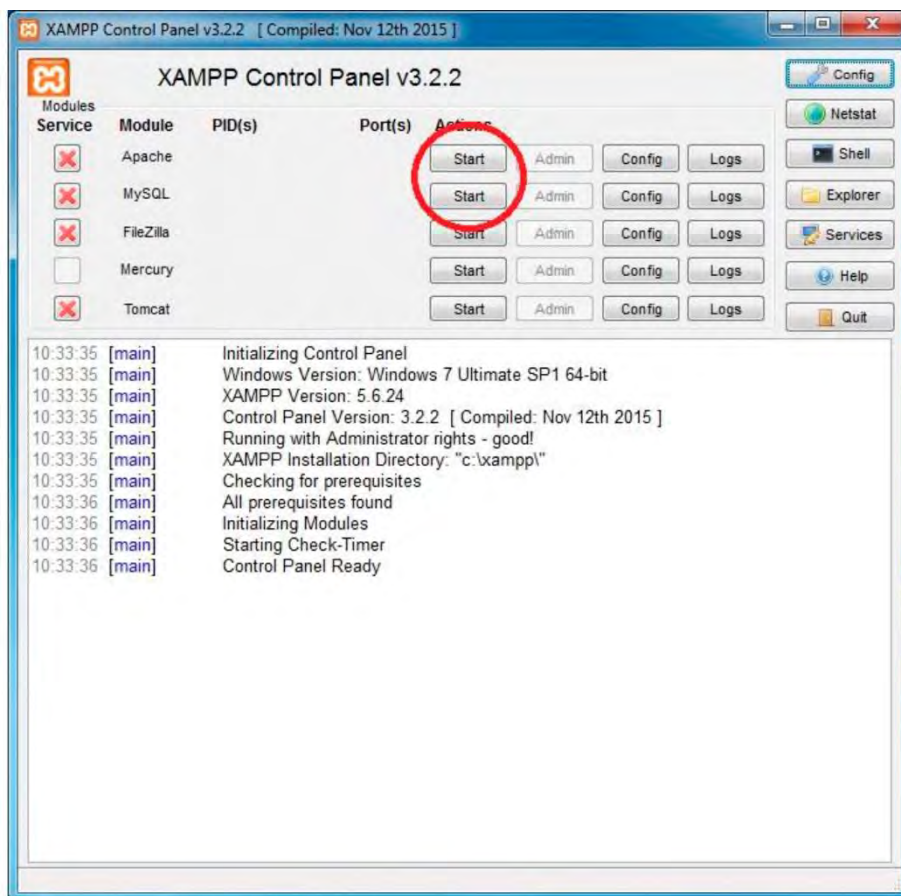


Рисунок – Панель управления веб-сервера XAMPP

ОСНОВЫ PHP

Курс Основы PHP предназначен для создания блога, который можно разместить в сети интернет на каком-либо хосте. Посетители смогут заходить на ваш сайт и читать ваши статьи. Кроме того будет поддерживаться простая авторизация пользователей, с помощью которой будет предоставляться возможность регистрации под своим именем.



Рисунок 1. Образец блога.

Для использования PHP на локальном компьютере устанавливаются и настраиваются необходимые для этого компоненты. Требуется поддержка веб-сервера Apache или Internet Information Server, базы данных MySQL и интерпретатора языка PHP. Для этого нужно мы будем использовать XAMPP, которая уже содержит всё необходимое для работы. Сборку можно скачать по адресу <https://www.apachefriends.org/ru/index.html>, нужно только выбрать сборку для своей операционной системы. В нашем случае будет использоваться система Windows 7.



Рисунок 2. Страница официального сайта XAMP

Установка XAMP

Процедура установки стандартна и не должна вызывать сложностей. При запуске установки, программа может порекомендовать отключить системы защиты. Можно выполнить рекомендации и продолжить установку.

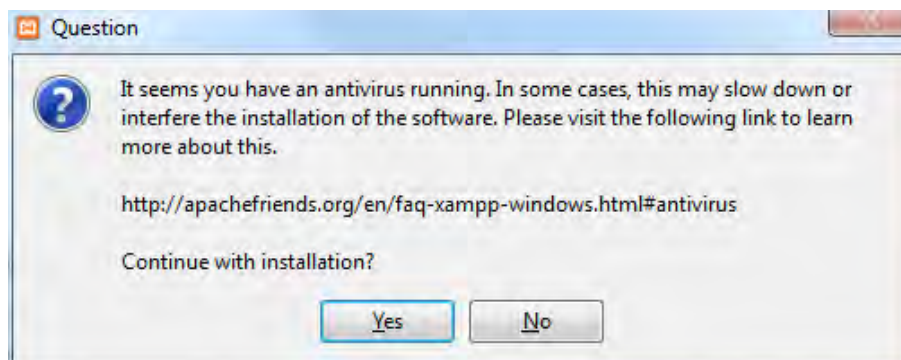


Рисунок 3. Вопрос установщика XAMP

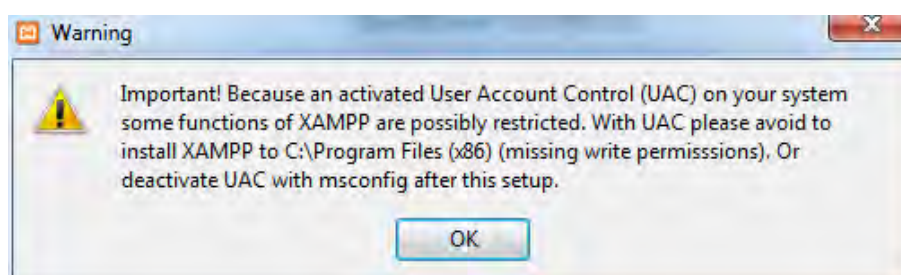


Рисунок 4. Предупреждение.

В окне выбора компонентов оставляем вариант «По умолчанию».

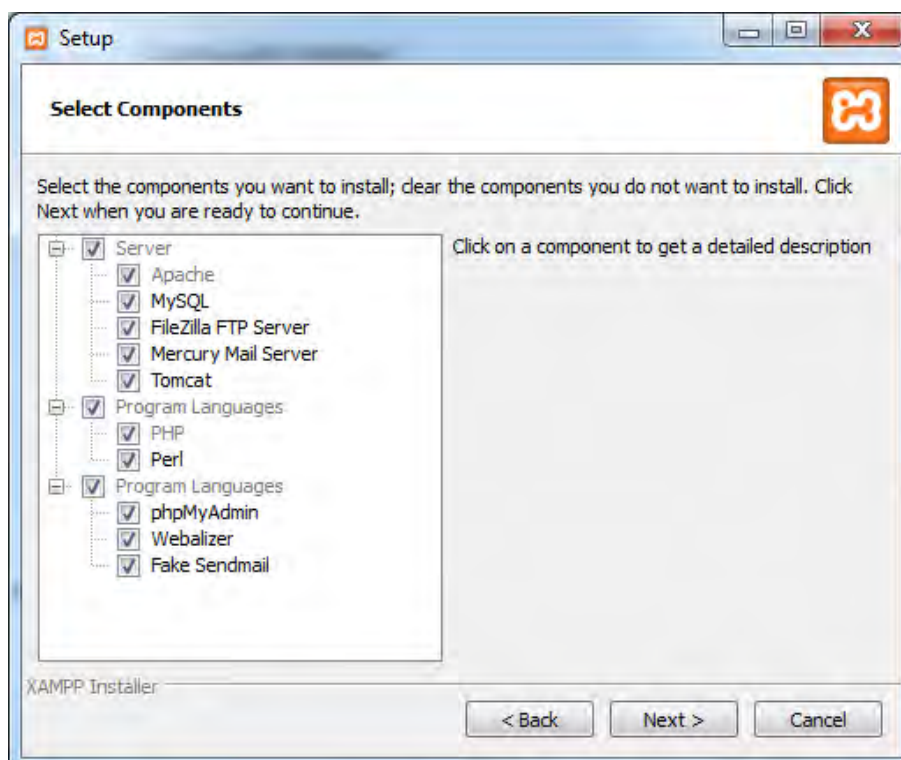


Рисунок 5. Выбор компонентов для установки

При выборе пути для установки желательно установить сервер прямо в корень диска, это обеспечит быстрый доступ к компонентам и к самому сайту.

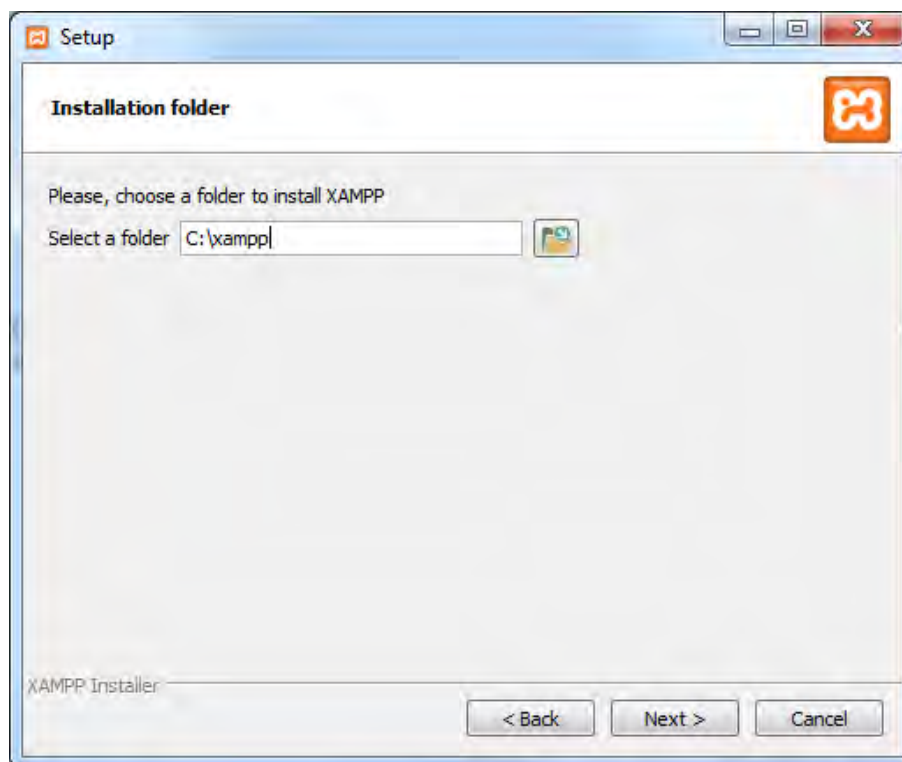


Рисунок 6. Выбор папки для установки

После установки нажимаем “Finish” запустится панель управления сервером.

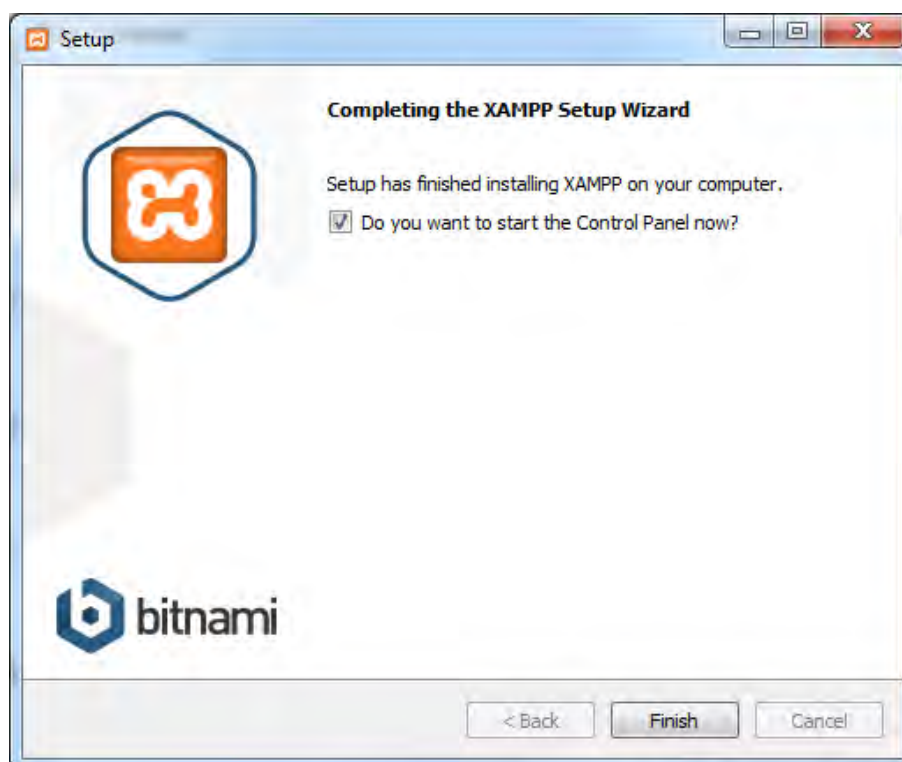


Рисунок 7. Завершение установки

В запущенной панели нужно запустить сервера Apache и MySQL нажатием кнопки “Start”.

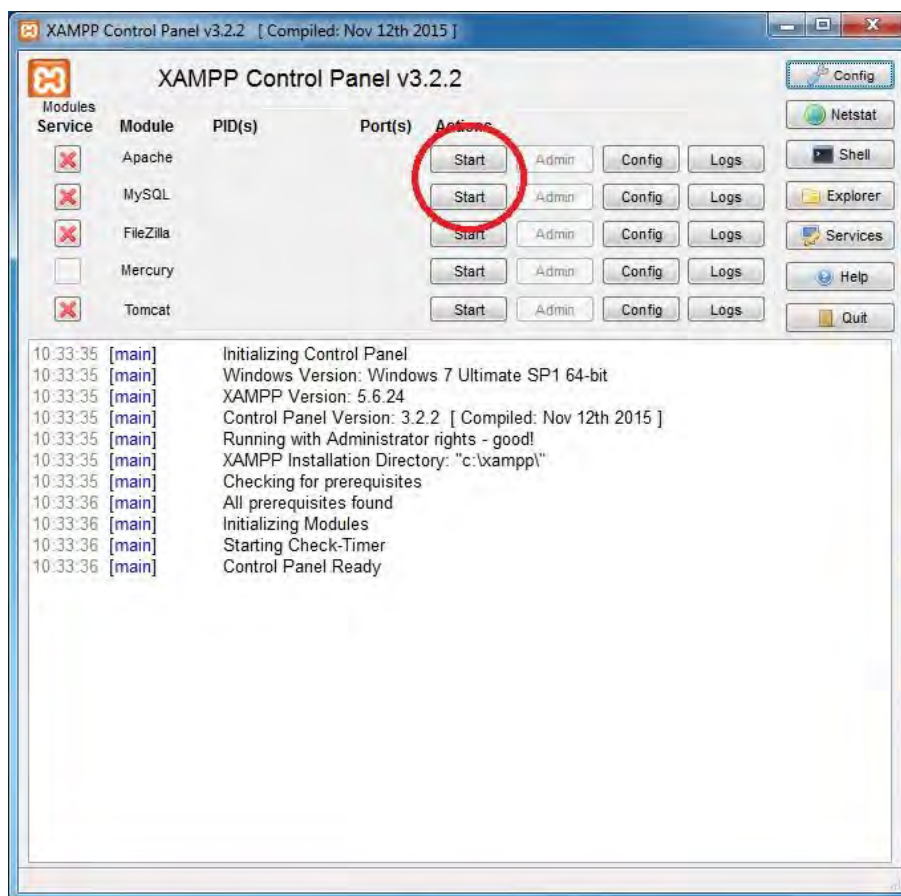


Рисунок 8. Запуск служб сервера

Далее нужно перейти по адресу <C:\xampp\htdocs> и создать отдельно папку `blog`. В этой папке создаём простой текстовый файл `file.txt` с текстом "Hello world!". После его создания открываем браузер и в адресной строке пишем <http://localhost/blog/file.txt>. В том случае, если всё было выполнено правильно, то в браузере должен открыться наш текст, который мы написали в документе.

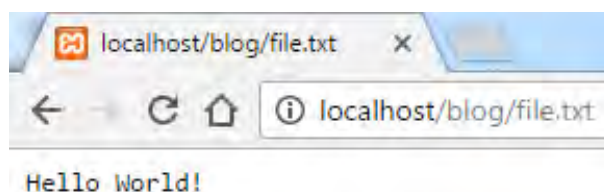


Рисунок 9. Доступ к созданному файлу через браузер

Для более удобного редактирования нашей веб-страницы вместо стандартного Блокнота или используемого ранее `Sublime Text` мы будем использовать сторонний редактор веб-страниц `Brackets`, его можно скачать с сайта <http://brackets.io/>.

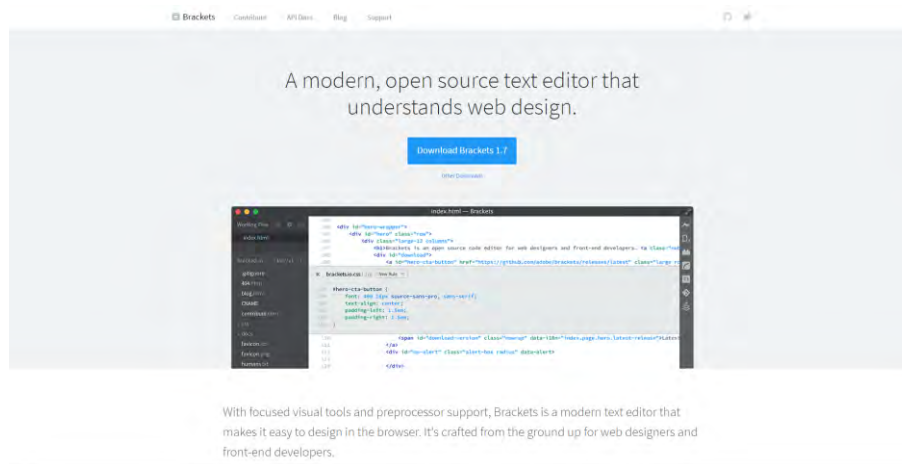


Рисунок 10. Главная страница программы Brackets

Программа устанавливается стандартно и никаких сложностей в процессе установки возникнуть не должно, единственное замечание, состоит в том, что программу нужно устанавливать от имени администратора.

После установки, чтобы открыть наш файл (file.txt) для редактирования, нужно щёлкнуть по нему правой кнопкой мыши и выбрать пункт «Open with Brackets».

Лабораторная работа №9

Шаблон сайта на HTML

Создание шаблона сайта на языке HTML

Задание для выполнения

1. Изучить теоретическое обоснование.
2. Следуя предложенным инструкциям создайте статьи на языке HTML на локальном веб-сервере

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

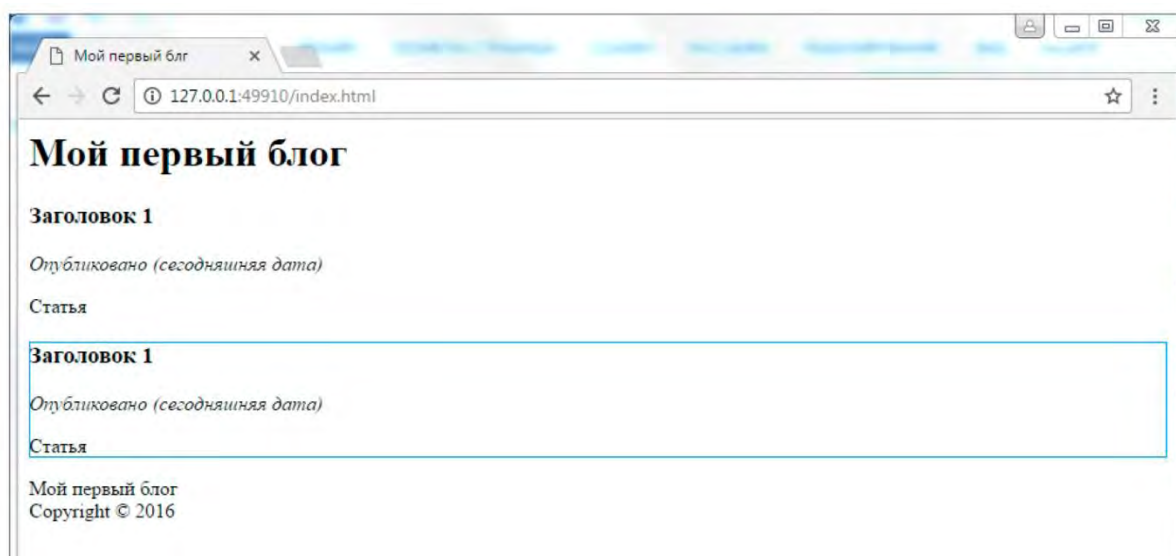


Рисунок – Результат работы в окне браузера

ОСНОВЫ HTML

HTML- это язык гипертекстовой разметки. На этом языке написаны абсолютно все сайты мира. Он максимально прост и требует лишь знания тегов. Для начала работы нужно настроить редактор сайтов Brackets на директорию по умолчанию. В программе, выбираем файл/Открыть директорию. Далее в окне нужно выбрать нашу директорию с документом file.txt (<C:\xampp\htdocs>). Затем нужно создать новый файл с именем index.html, затем сразу же его сохраняем в этой же папке. Для большего удобства можно включить предварительный просмотр проделанной работы. Для этого в правой части окна нужно нажать значок молнии.

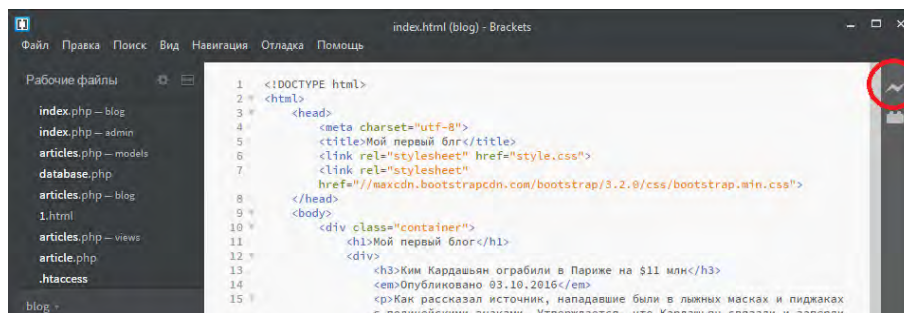


Рисунок 1. Функция Live режима

Напишите следующий код:

```
<!DOCTYPE html> //версия языка html
<html> //начало документа html, обязательный элемент.
  <head> //«голова» документа, обязательный элемент.
    <meta charset="utf8"> //указание кодировки документа
    <title>Мой первый блог</title> //Заголовок страницы сайта
  </head>
  <body> //тело документа, видимая часть для пользователя
    <div> //контейнер элементов, содержащихся в нём
      <h1>Мой первый блог</h1> //заголовок контейнера
    </div>
```

Создадим ещё один контейнер, в котором будут размещаться статьи.

```
<div>
  <h3>Заголовок 1</h3>
  <em>Опубликовано (сегодняшняя дата)</em> //для записи даты под наклоном используется тег <em>
  <p>Статья 1</p> //текст статьи
</div>
<div>
  <h3>Заголовок 2</h3>
  <em>Опубликовано (сегодняшняя дата)</em>
  <p>Статья 2</p>
</div>
</div>
<footer> //подвал страницы, который будер расположен в самом конце страницы
  <p>Мой первый блог <br> Copyright &copy; 2015</p>
</footer>
</div>
</body>
</html>
```

Примечания:

`
`- перенос на новую строку, «©»- знак копирайта.

После написания вышеуказанного кода в окне браузера должен отобразиться следующий результат:

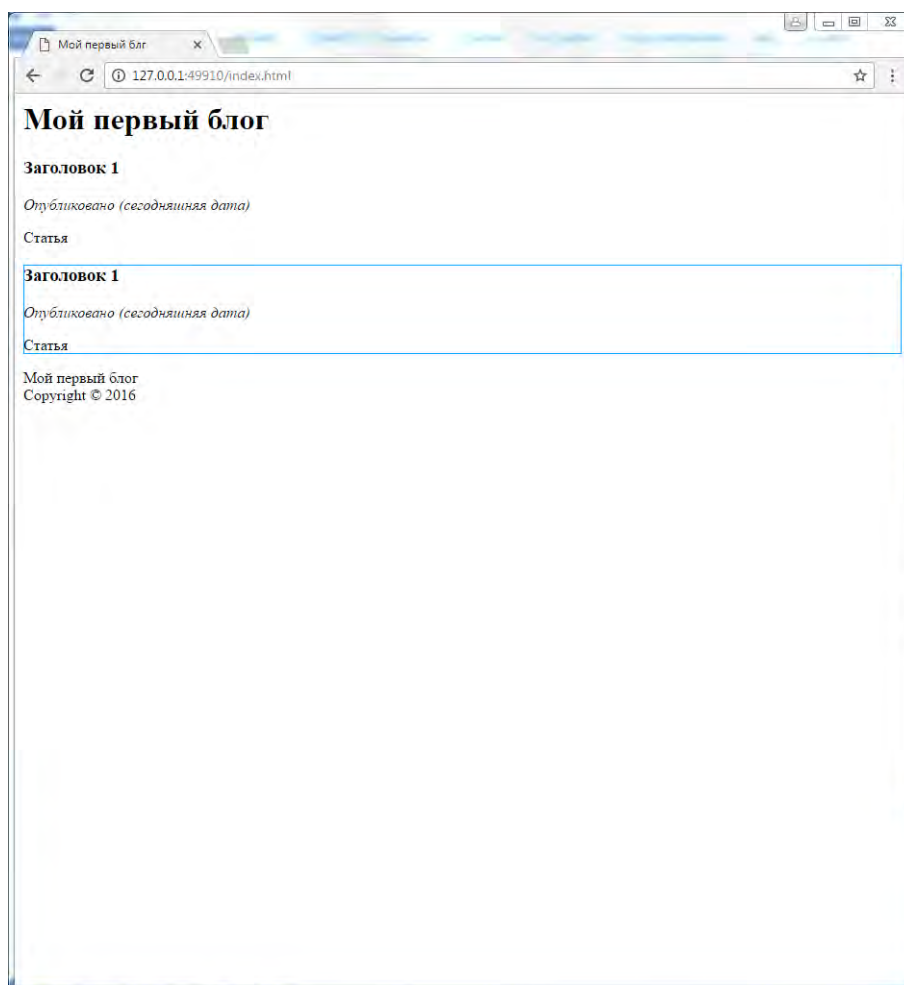


Рисунок 2. Результат работы

Теперь нужно наполнить статью содержимым. Для этого прямо в коде пишем заранее заготовленный текст статьи. Для этого в заголовках 1, 2 пишем заголовки статей, а в статьях 1, 2 пишем текст самой статьи.

В конечном итоге готовые статьи с заголовками будут выглядеть примерно вот так:



Рисунок 3. Страница наполненная статьями

Лабораторная работа №10

Основы CSS

Оформление сайта с помощью таблиц CSS

Задание для выполнения

1. Изучить теоретическое обоснование.
2. Следуя предложенным инструкциям с помощью таблиц CSS оформите сайт.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

Bootstrap CDN

Участники [MaxCDN](#) любезно предоставили поддержку CDN для CSS и JavaScript Bootstrap. Просто воспользуйтесь этими ссылками для [Bootstrap CDN](#).

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">

<!-- Optional theme -->
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->
<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>
```

Установка с помощью Bower

Установите и работайте с Less, CSS, JavaScript и шрифтами Bootstrap с помощью [Bower](#).

```
$ bower install bootstrap
```

Рисунок – Код для использования сторонних таблиц стилей

ОСНОВЫ CSS

CSS или каскадные таблицы стилей, служат для форматирования веб-страниц. Например, можно с помощью CSS выровнять положение текста, сменить межстрочный интервал, поменять шрифт и прочее.

Для подключения этих таблиц нужно в секции `<head>` добавить `<link rel="stylesheet" href="style.css">`. (`href="style.css"`), он указывает, где лежит наш файл с таблицей стилей. После этого нужно будет сохранить изменения и перезапустить функцию Live Preview, которая сразу отображает в браузере введённые изменения (значок молнии в правой части окна редактора Brackets). Теперь нужно создать сам файл с каскадными таблицами. Для этого создаём новый файл и сразу же его сохраняем в ту же папку, где лежит наш `index.html`.

Для примера, допустим, нам нужно выровнять «подвал» с копирайтом выровнять по середине, для этого в только что созданном документе с каскадными стилями нужно написать код:

```
Footer {  
    Text-align: center; //выравнивание текста по середине.  
}
```

Footer- это тег из кода `html`, которым мы обозначили подвал. Сразу же сохраняем и видим, что текст в «подвале» переместился в центр.



Рисунок 1. Выравнивание подвала по середине

Теперь нужно изменить расстояние между статьями, для этого в секции `<div>` перед статьями нужно дописать классы, потому, что у нас секции `<div>` используются во многих местах документа, чтобы текст не развалился, секции со статьями нужно классифицировать, для этого в коде перед статьями в секции `<div>` дописываем `<div class="article">`. Теперь в CSS коде дописываем следующее:

```
<...>
.article {
  Margin-top: 30px; //расстояние между статьями в пикселях
  Margin-bottom: 30px; //расстояние между статьями и «подвалом» в пикселях.
}
```

После этого обновляем окно браузера и мы увидим изменения после проделанной работы.

Использование сторонних CSS

Для форматирования своего сайта можно использовать и сторонние таблицы стилей. Их можно скачать в готов виде и подключить их локально, либо и х можно подключить указав просто ссылку в нашем html коде. В случае подключения с помощью ссылки в интернет потребуется подключение к интернету, так как по этой ссылке скачивается файл оформления. Существует сайт, который поддерживает обе эти возможности, расположен он по ссылке <http://www.oneskyapp.com/ru/docs/bootstrap/getting-started>, затем нужно скопировать первый код из секции Bootstrap CDN:



Рисунок 2. Код для вставки в html документ

После того, как необходимая ссылка скопирована, то её нужно вставить в место, где мы впервые подключаем нашу таблицу CSS, в тело секции `<head>`. После этого страницу браузера нужно будет обновить. Далее, нужно текст выровнять по центру. Для этого в подключённой внешней библиотеке от бутстрапа есть класс, который называется "container". Его нужно дописать рядом с главным div (перед `<h1>Мой первый блог</h1>`). Так будет выглядеть дописанный в секцию класс: `<div class="container">`, в итоге текст в браузере будет выровнен по центру и он будет иметь более приятный внешний вид.

Код, который должен был получиться в результате проделанной работы:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Мой первый блог</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
  </head>
</body>
  <div class="container">
    <h1>Мой первый блог</h1>
    <div>
      <h3>Ким Кардашьян ограбили в Париже на $11 млн</h3>
      <em>Опубликовано 03.10.2016</em>
      <p>Как рассказал источник, нападавшие были в лыжных масках и пиджаках с полицейскими знаками. Утверждается, что Кардашьян связали и заперли в ванной. По имеющимся данным, грабители взяли шкатулку с ювелирными украшениями стоимостью от €5 млн до €6 млн, а также кольцо стоимостью около €4 млн. Кроме того, оказалось, что Кардашьян ограбили не в отеле, а в «тайной» элитной резиденции недалеко от церкви Мадлен. В этих апартаментах часто останавливаются знаменитости и состоятельные клиенты. Кардашьян уже приезжала туда по меньшей мере один раз, в 2014 году. Ранее сообщалось, что Кардашьян угрожали оружием в парижском отеле. Также сообщалось, что Кардашьян ограбили пять вооруженных мужчин в парижском отеле.</p>
    </div>
    <div>
      <h3>В сознании японцев неразделимы Солнце и Хризантемы</h3>
      <em>Опубликовано 29.09.2016</em>
      <p>菊 (кику) – этим иероглифом в Японии обозначается хризантема. А ещё им же – солнце. Поэтому, можно сказать, что в Японии два солнца – одно на небе, другое на земле).
      <p>В одной японской легенде рассказывается как бог Неба Идзанаги, решил искупаться в реке на Земле. Его драгоценности, упав на землю, превратились в цветы: один браслет – в ирис, другой – в цветок лотоса, а ожерелье – в золотую хризантему. Хризанте му в Японии не просто любят – а боготворят.
      <p>9-го числа 9-го лунного месяца в Японии раньше отмечали Праздник Хризантем. Люди катались на «хризантемовых лодках», пили «хризантемное вино», любовались цветущими в садах хризантемами, слагали в их честь песни и стихи. Хризантемные стихи» писали на длинных бумажных полосах тушью с особым старанием и прикрепляли их на деревьях, чтобы ветер разнес славу о красоте хризантем по всему миру.
      <p>Хризантема в Японии является символом солнца и любимым цветком Солнечной Богини Аматэрасу, от которой вели свой род японские императоры.
      <p>Предположительно с VII века, когда рисунок хризантемы украсил клинок микадо, она считается эмблемой японских императоров. Стилизованный золотистый цветок с шестнадцатью двойными лепестками и поныне остается гербом Императорского дома, а иногда выполняет и роль государственного герба: на монетах, печатях и официальных документах.
      <p>Именно такое изображение хризантемы считалось священным, право на него, в частности, на ношение одежды с рисунком 16-лепесткового цветка, принадлежало исключительно членам императорской фамилии.
      <p>Нарушившим этот порядок рядовым японцам грозила смертная казнь.
      <p>Такая же хризантема красуется на Ордене Хризантемы, учрежденном в 1888 году и до нынешнего времени считающимся высшей и самой почетной наградой в стране.
      <p>Однако для японцев солнечный цветок — не просто застывший в металле символ. Это растение в Японии окружено любовью и заботой. Японцам нет равных в промышленном разведении хризантем, в создании новых сортов с разнообразием форм и окрасок.
      <p>Очевидно, благодаря долгому периоду цветения, хризантема олицетворяет счастье и долголетие. Есть поверье, что собранная с хризантемы роса продлевает жизнь.</p>
    </div>
  </div>
  <div class="text">
    <p>Мой первый блог<br>Copyright &copy; 2016</p>
  </div>
</html>
```

</div>
</body>
</html>

Внешний вид веб-страницы, после проделанной работы:

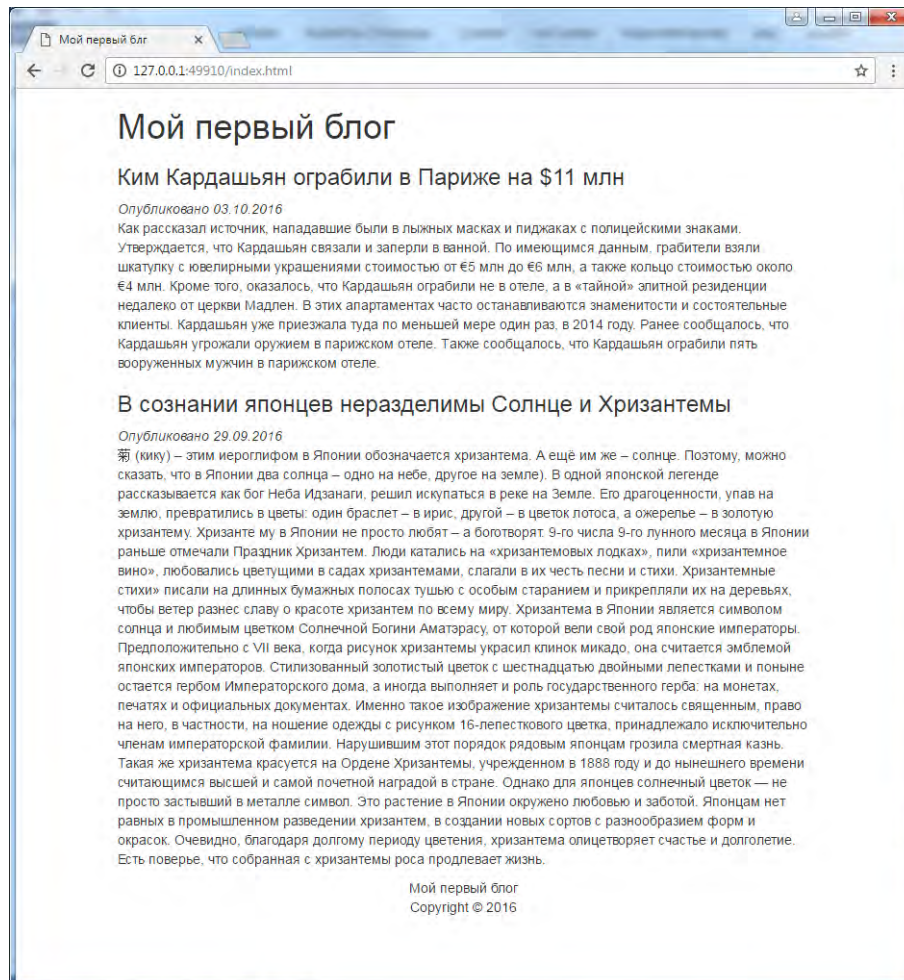


Рисунок 3. Результат работы

Лабораторная работа №11

Функции и циклы языка PHP

Выполнение арифметических вычислений с помощью функций и циклов

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Изучите принципы использования функций и организации циклов на языке PHP.
3. Следуя предложенным инструкциям произведите простые арифметические вычисления с выводом их на экран.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

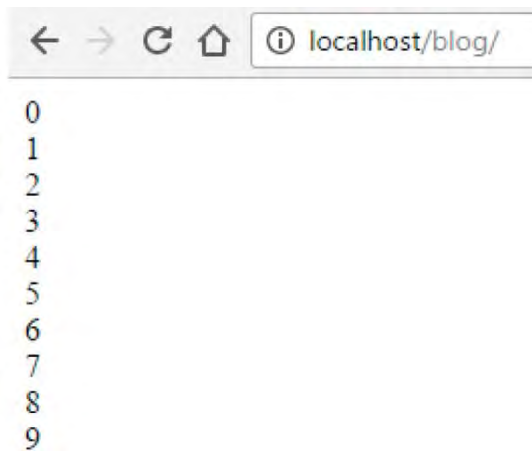


Рисунок – Результат выполнения программы

ОСНОВЫ PHP

PHP- скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических вебсайтов. В примерах ниже будут приведены простые арифметические операции, которые может выполнить язык php. В программе Brackets создаём новый файл и сразу же сохраняем его под именем `index.php` в той же папке, где у нас уже лежат каскадные таблицы и `html` документ.

В только что созданном документе нужно написать код:

```
<?php
    echo "Hello World!";
?>
```

Перейдите по адресу <http://localhost/blog/index.php>. Это путь, к `php` файлу, который только что был создан. После перехода по этому пути, браузер выведет вам сообщение, которое мы написали в коде `php`.

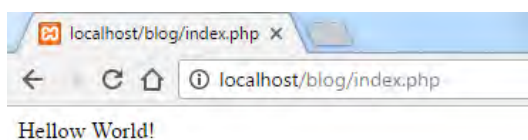


Рисунок 1. Сообщение `php` интерпретатора

В следующем шаге мы выведем время прошедшее с 1 января 1970 года в секундах, для этого мы напишем следующий код:

```
<?php
    echo time();
?>
```

Затем нужно обновить браузер и интерпретатор выведет нам прошедшие с того года секунды. Стоит заметить, что при каждом обновлении страницы, число будет увеличиваться на 1 ежесекундно.

Теперь мы сложим 2 числа, для этого нужно объявить 2 переменные:

```
<?php
    $a=$_GET['a'];
    $b=$_GET['b'];
    //Далее мы пишем то, что хотим вывести в ответе
    Echo "a+b=".(($a+$b));
?>
```

Для подсчёта чисел нужно передать этому скрипту информацию о том, что мы хотим посчитать. Для этого в адресной строке браузера нужно прописать параметры вычисления для `a` и `b`:

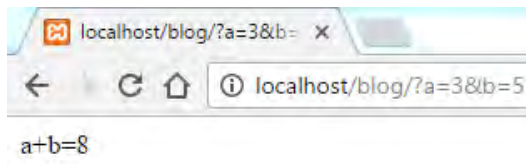


Рисунок 2. Результат подсчётов

Стоит заметить, что `php` язык чувствителен к регистру. Например команды `$_GET` и `$_get` совершенно разные.

Функции.

PHP как и любой другой алгоритмический язык поддерживает работу с функциями. Функции удобны тем, что если во многих участках кода нужно постоянно выполнять одно и то же действие, например, если каждый раз требуется одна и та же процедура для подсчёта или сортировки. В данном случае рассмотрим как можно выполнить простое сложение.

```
<?php
function add($param1, $param2){ //объявляем функцию и называем её add, в скобках пишем её
принимаящие параметры.
return $param1+$param2; //считаем, результат отправляем обратно в команду отображения echo.
}
$a=$_GET['a'];
$b=$_GET['b'];
echo add($a, $b); //передаём данные в функцию, а после подсчёта принимаем результат и отображаем
его.
?>
```

В итоге при введённых в адресной строке данных <http://localhost/blog/?a=3&b=5> на экране высветится число 8.

Циклы.

Циклы выполняют роль счётчика или циклического условия и выполняются до тех пор, пока условия не будет выполнено или наоборот. Например, цикл `for` считает от определённой заданной числовой позиции до конечной числовой позиции. Если цикл достиг своей цели, то он завершает всю работу. Рассмотрим пример ниже:

```
<?php
for ($i=0;$i<10;$i++){ //цикл работает до тех пор, пока значение $i не достигнет 10. Параметр i++ увеличивает
значение $i на 1.
echo $i."<br>"; //отображаем результат и сразу же переходим на новую строку.
}
?>
```


Таким образом результат должен выглядеть вот так:

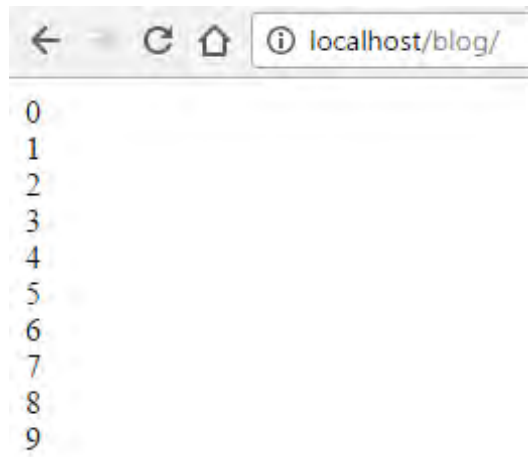


Рисунок 3. Результат работы цикла

Лабораторная работа №12

Проектирование блога на языке PHP

Подготовка и создание файлов и папок для написания блога.

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Следуя предложенным инструкциям создайте файлы и папки, необходимые для организации блога. Внесите необходимые изменения для возможности управления функционалом блога.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

ПРОЕКТИРОВАНИЕ

В этот раз мы создадим некоторые файлы и папки для нашего будущего сайта.

Для этого перейдём в папку <C:\xampp\htdocs\blog> и создадим там папку `admin`, в этой папке создадим файл `index.php`. Уже в этом файле запишем небольшой `php` код:

```
<?php
    echo "Admin Panel" ;
?>
```

Этот код ничего кроме отображения информации пока не делает.

Далее, переходим в папку `blog` и создаём новую папку, которая будет называться `"models"`, в этой папке создадим файл `articles.php`. В этом файле будут функции для работы со статьями. Пишем в этом файле следующее:

```
<?php
function articles_all(){
}
function articles_get($id){ //получение статьи по ид
}
function articles_new($title, $date, $content){ //создание статьи
}
function articles_edit($id, $title, $date, $content){ //редактирование статьи
}
function articles_delete($id){ //удаление статьи
}
?>
```

Стоит напомнить, что значения в скобках у функции являются принимающими, они принимают значения от команд, которые в свою очередь обрабатываются и уже обработанный результат передают обратно.

После этого нужно сохранить наш файл и перейти в папку `blog`. В этой папке создадим файлы `database.php` и `article.php`.

Файл `article.php` будет выводить информацию о наших статьях, в него пишем следующий код:

```
<?php
    echo "Article";
?>
```

Теперь в папке `blog` откройте `index.php`. Удалите всё содержимое документа и напишите следующее:

```
<?php
    require_once("database.php"); //подключения файла базы данных, на этот момент он пуст
    require_once("models/articles.php"); //Подключение файла с функциями управления статьями
    $articles=articles_all(); //Обозначение функции «articles_all», теперь к ней можно обращаться через переменную
    $articles так же как и обычно.
    include("views/articles.php");
?>
```

В папке `blog` создадим ещё папку `views`. В этой папке будут храниться коды `html` страниц, чтобы отделить коды `php` и `html`.

Лабораторная работа №13

Подготовка шаблонов для статей в блоге

Создание шаблона блога на языке PHP

Задание для выполнения

1. Изучите теоретическое обоснование.
2. Следуя предложенным инструкциям создайте шаблон для будущих статей на языке PHP.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

Мой первый блог

Простой заголовок

Опубликовано: 2016-10-06

Статья

Мой первый блог

Copyright © 2016

Рисунок – Результат работы скрипта PHP

ШАБЛОНИЗАЦИЯ

В этот раз мы рассмотрим примеры для работы с шаблонами сайтов, с помощью которых в дальнейшем будет удобнее работать и создавать статьи.

Откройте файл `articles.php` и допишем в секцию `articles_all()` код, который будет возвращать массив из 2 элементов, которые содержат статьи. Для этого пишем массив, который будет называться `$art1`.

Код:

```
<...>
function articles_all(){
    $art1=["id"=>1, "title"=> "Title1", "date"=> "2016-10-06", "content"=> "Content1"]; //запись будем иметь id=1, "title" будем
    "Title1", "date"( календарная дата) будем 2016-10-06, а в контенте будем написано "Content1".
}
//Таким же образом создадим ещё один элемент с названием $art2 сразу после $art1.
$art2=["id"=>2, "title"=> "Title2", "date"=> "2016-10-06", "content"=> "Content2"];
//Теперь помещаем элементы в массивы:
$arr[0]=$art1;
$arr[1]=$art2;
return $arr;
//В конечном счёте код будет выглядеть таким образом:
...
function articles_all(){
    $art1=["id"=>1, "title"=> "Title1", "date"=> "2016-10-06", "content"=> "Content1"];
    $art2=["id"=>2, "title"=> "Title2", "date"=> "2016-10-06", "content"=> "Content2"];
    $arr[0]=$art1; //массив 1 и 2 снизу.
    $arr[1]=$art2;
    return $arr; //возвращаем результат работы
}
}
```

Переходим к файлу `index.php`. Здесь в коде подключаем файл `articles.php`:

```
Include ("views/articles.php");
```

В данный момент такого файла в папке `views` пока нет, его нужно создать. Создаём файл `articles.php` в папке `views`. Теперь открываем файл `index.html`, который хранится в папке `blog` и копируем всё его содержимое и вставляем в `articles.php`, который лежит в папке `views`. Удалите из кода названия статей, дату и сами статьи. После этого нужно составить цикл для отображения статей.

Код:

```
<...>
<h1>Мой первый блог</h1>
<div>
    <?php foreach($articles as $a): ?> //цикл для каждого элемента $articles, а$- название рассматриваемого элемента
        <div class="article">
            <h3>
                <a href="articles.php?id=<?=$a["id"]? (обращение к id заметки, которая будет передаваться в качестве
                параметра)>"><?=$a["title"]?></a> //краткая запись объявления php имеет вид "?"! Этот пункт будет гиперссылкой в
                заголовке статьи, который будет ссылаться на articles.php. Здесь так же используется название элемента массива
                $a, который будет отображать элементы статьи на странице.
            </h3>
            <em>Опубликовано: <?=$a["date"]?></em> //отображение даты публикации через обращение к элементу
            массива date.
            <p><?=$a["content"]?></p> //отображение контента в статье. Здесь так же обращение к элементу
            происходит через а$.
        </div>
```

```
<?php endforeach ?> //конец цикла. Между <?php foreach($articles as $a) и <?php endforeach ?> цикл будет
повторяться определённое количество раз, в зависимости от количества элементов в массиве $articles.
</div>
```

В итоге, если набрать в браузере <http://localhost/blog/>, то должно отобразиться следующее:

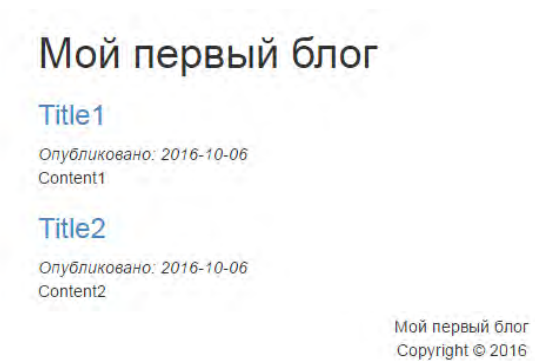


Рисунок 1. Главная страница блога

Теперь нужно открыть файл `article.php`, который лежит в каталоге `blog` и скопировать в него содержимое файла `index.php`, который лежит в корневой папке `blog`. Затем функцию `$articles=articles_all()` нужно переименовать в `$articles=articles_get`, так как нам нужно к ней обратиться для правильной работы нашего блога, затем через параметр `$_GET` мы должны принять параметры из URL строки наш `id` статьи `$articles=articles_get($_GET['id']);`

Теперь в папке `views` создадим `article.php` и копируем всё, что хранится в файле `articles.php`, убираем наш цикл (`<?php foreach($articles as $a):?>`) и его конец, так как нам нужно будет отобразить только статью, так же нужно удалить ссылку, так как она в данном случае будет не нужна. Таким образом код будет выглядеть вот так:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Мой первый блог</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <h1>Мой первый блог</h1>
      <div>
        <div class="article">
          <h3>
            <?=$article['title']?>
          </h3>
          <em>Опубликовано: <?=$article['date']?></em>
          <p><?=$article['content']?></p>
        </div>
      </div>
      <footer>
        <p>Мой первый блог<br>Copyright &copy; 2016</p>
      </footer>
    </div>
  </body>
```

</html>

Открываем `article.php`, который лежит в папке `blog` и меняем переменную `$articles` на `$article`.

Код:

```
<?php
    require_once("database.php");
    require_once("models/articles.php");
    $article=articles_get($_GET['id']);
    include("views/article.php");
?>
```

Теперь нужно открыть файл `articles.php` в папке `models` и в секции `function articles_get($id)` дописать код:

```
return ["id"=>1, "title"=>"Простой заголовок", "date"=> "2016-10-06", "content"=>"Статья"];
```

Он будет отображать заголовок, дату и статью на экране.

Осталось переименовать файл `article.php` в папке `blog` в `articles.php`, в конечном итоге, если на главной странице перейти по ссылкам `Title1` или `Title2` то на экране должны будут отобразиться наши данные о статье.

Мой первый блог

Простой заголовок

Опубликовано: 2016-10-06

Статья

Мой первый блог

Copyright © 2016

Рисунок 2. Результат работы скрипта `php`

Лабораторная работа №14

MySQL, создание базы данных

Создание базы данных и таблиц в phpMyAdmin

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям создайте базу данных и необходимые таблицы.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

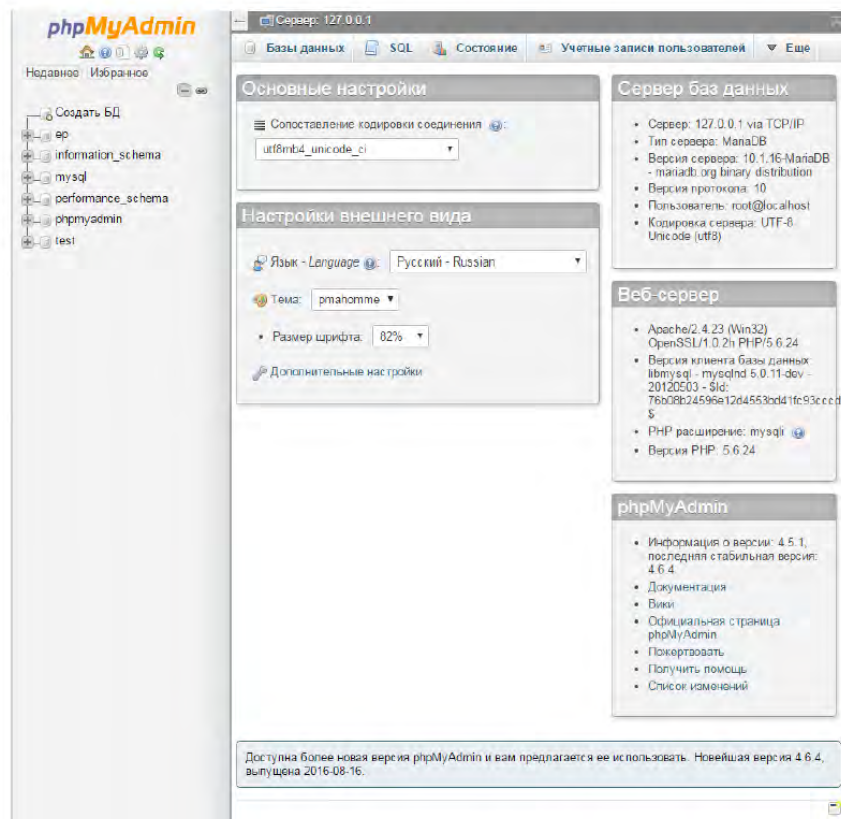


Рисунок – Панель управления phpMyAdmin

MYSQL, СОЗДАНИЕ БАЗЫ ДАННЫХ

Базы данных используются абсолютно на любых крупных сайтах, особенно активно базы используются в различных интернет магазинах и особенно социальных сетях. В них хранится вся информация о каждом зарегистрированном на сайте пользователе. Самая распространённая база на сегодняшний день- MySQL, в ней мы сейчас и будем работать.

Запустите веб-сервер, если он не запущен, нам понадобятся Apache, MySQL. После этого перейдите по адресу <http://localhost/>, в левой части списка найдите ссылку на phpMyAdmin и сразу же переходим по ней.

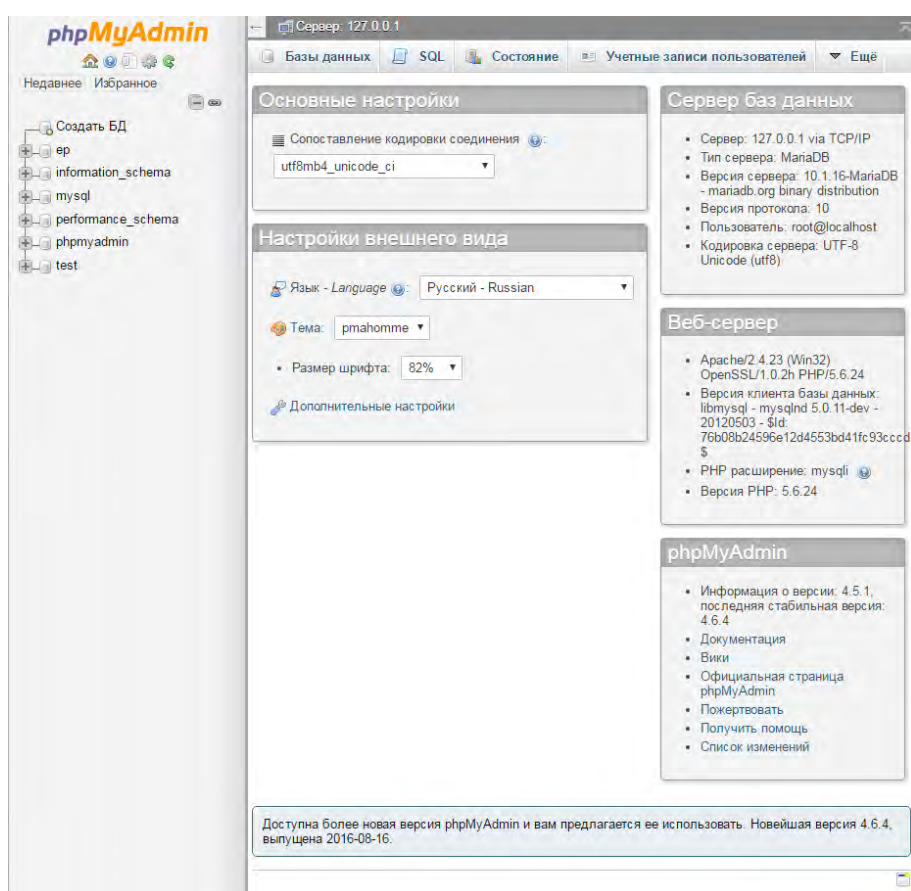


Рисунок 22. Главная страница phpMyAdmin

Данный интерфейс предназначен для управления базой данных через окно браузера. Теперь нам нужно создать базу данных. В левой части окна нажмите ссылку «Создать БД», затем назовите базу данных, например blog, выберете кодировку «utf8_general_ci» и нажмите кнопку создать. В левой части окна выберете вновь созданную базу и в имени базы введите “articles”, количество столбцов- 4 и нажмите вперёд. Заполните таблицы следующим образом как показано на рисунке снизу. После заполнения нажмите на кнопку «сохранить».

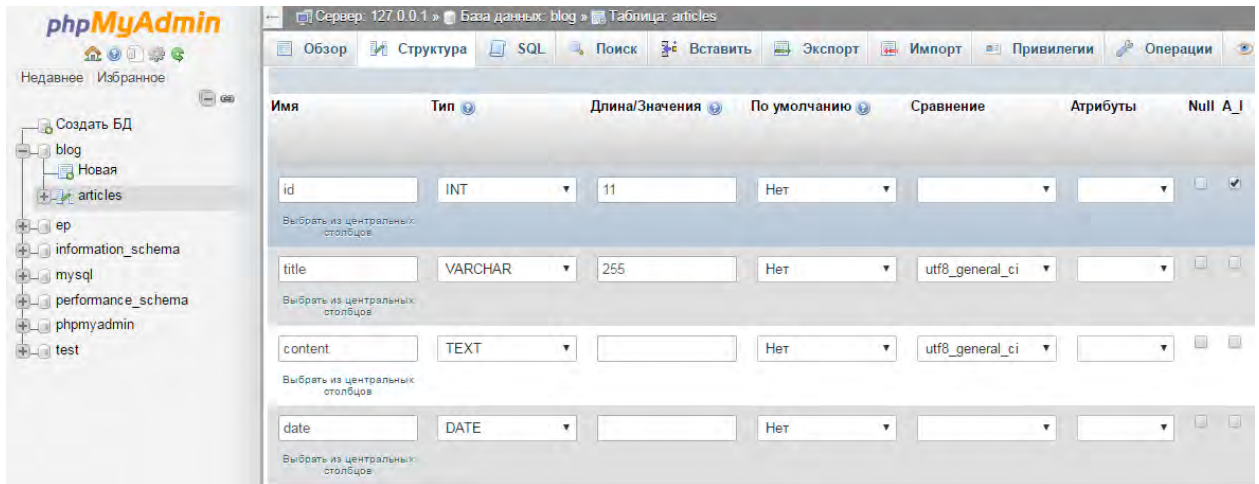


Рисунок 23. Создание структуры базы данных

В итоге должно получиться следующее:

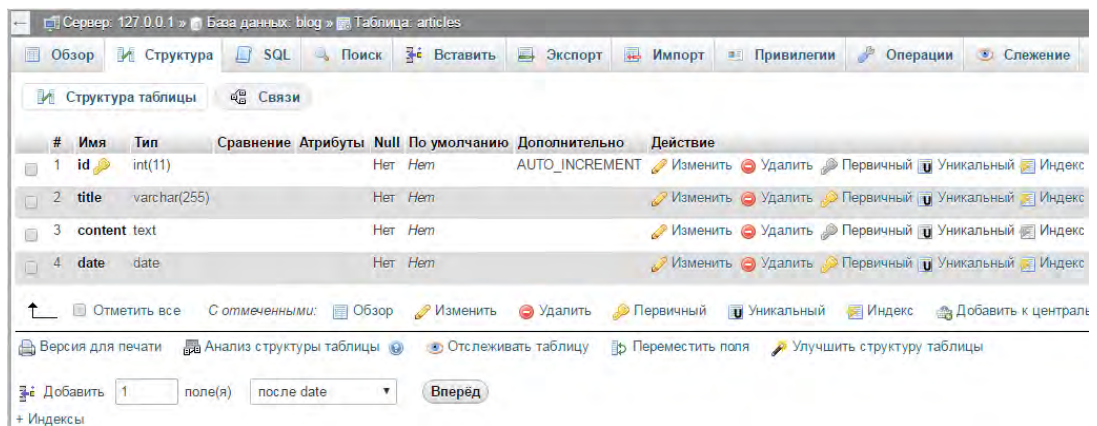
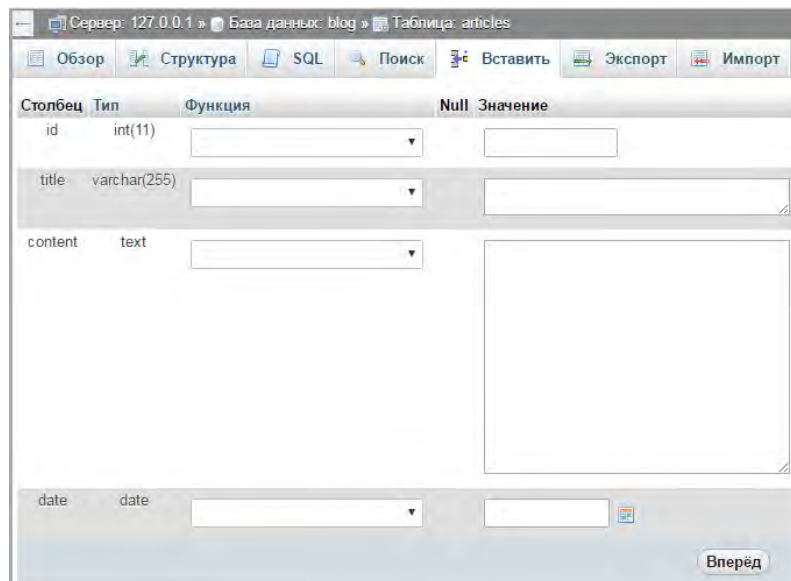


Рисунок 24. Созданная структура базы данных

Теперь нужно назначить записи “id” первичные значения (primary key), для этого в строке int нужно выбрать параметр «первичный». Там же выбираем «изменить», сверху находим параметр «A_I» и устанавливаем на него галочку, затем сохраняем. Переходим во вкладку «Вставить».

Заполните данные в строках как на примере:



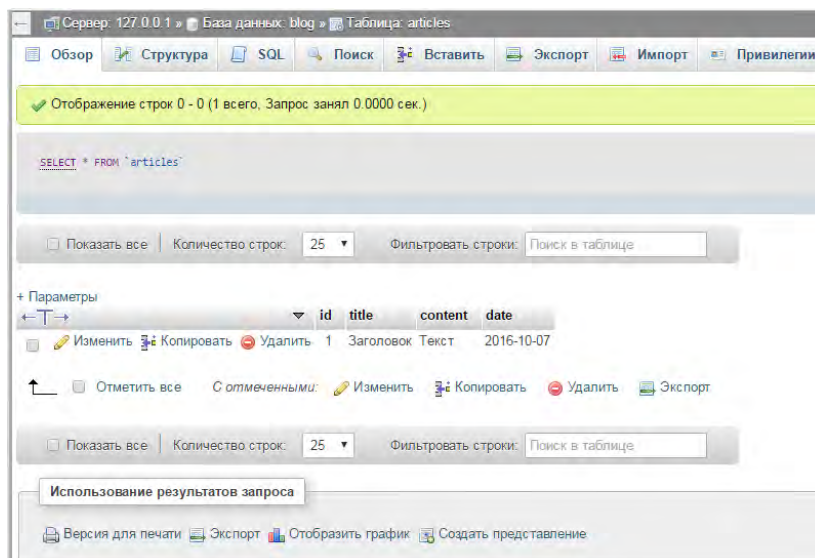
The screenshot shows a database management interface for a table named 'articles'. The table structure is as follows:

Столбец	Тип	Функция	Null	Значение
id	int(11)			
title	varchar(255)			
content	text			
date	date			

Each column has a corresponding input field: a dropdown for 'id', a text box for 'title', a large text area for 'content', and a date picker for 'date'. A 'Вперед' button is located at the bottom right.

Рисунок 25. Заполнение таблицы данными

Теперь, если перейти во вкладку «Обзор», то мы увидим, что в таблице появилась 1 запись.



The screenshot shows the 'Обзор' (Overview) tab of the database management tool. It displays the following information:

- Message: Отображение строк 0 - 0 (1 всего, Запрос занял 0.0000 сек.)
- SQL Query: `SELECT * FROM 'articles'`
- Table Headers: id, title, content, date
- Table Data: 1 row with values: Заголовок, Текст, 2016-10-07
- Actions: Изменить, Копировать, Удалить
- Buttons: Показать все, Количество строк: 25, Фильтровать строки: Поиск в таблице
- Footer: Использование результатов запроса, including options like Версия для печати, Экспорт, and Создать представление.

Рисунок 26. Новые данные в таблице

Лабораторная работа №15

Взаимодействие PHP и MySQL

Вывод содержимого базы на экран при помощи языка PHP

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям с помощью языка PHP выведите на экран информацию из базы данных.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

ВЗАИМОДЕЙСТВИЕ PHP И MYSQL

Мы уже знаем, что все наши статьи можно хранить в *php* файле и оттуда же их отображать пользователю на экран. Однако, это не самый оптимальный способ хранения информации, так как со временем количество информации растёт и всё хранить в *php* файлах становится не удобно. Все эти данные можно хранить в базе данных и при необходимости их оттуда вытащить или изменить.

Откройте файл *database.php*, который лежит в папке *blog* и запишите туда следующий код:

```
<?php
//с 3-6 строки- определения констант
define('MYSQL_SERVER', 'localhost'); // MYSQL_SERVER= localhost
define('MYSQL_USER', 'root'); // MYSQL_USER= root. Root- это самый главный пользователь в базе данных, который
имеет все привилегии.
define('MYSQL_PASSWORD', ''); //пароль по умолчанию не ставится
define('MYSQL_DB', 'blog'); //MYSQL_DB= blog
function db_connect(){ //функция для подключения к базе данных
    $link=mysqli_connect(MYSQL_SERVER, MYSQL_USER, MYSQL_PASSWORD, MYSQL_DB)
    or die("Error: ".mysqli_error($link)); //если по какой-то причине соединение с базой установить не получилось, то
выводится сообщение об ошибке. Данный об ошибке записываются в переменную $link, так же она используется для
взаимодействия с соединением.
    if(!mysqli_set_charset($link, "utf8")){ //выставляем кодировку utf8
        printf("Error: ".mysqli_error($link)); //выводим сообщение об ошибке в случае неудачи
    }
    return $link; //функция db_connect() возвращает дескриптор.
}
?>
```

Теперь откройте *index.php*, который лежит в папке *blog* и впишите туда код:

```
<?php
require_once("database.php");
require_once("models/articles.php");
$link=db_connect(); //функция соединения с базой данных
$articles=articles_all();
include("views/articles.php");
?>
```

Теперь откройте файл *articles* в папке *models*. До этого мы отображали только статичную информацию, которую писали в этом файле. Сейчас мы будем уже вытаскивать нужную нам информацию из базы данных и отобразим её на экране. Для этого в функции *articles_all* дописываем следующий код, всё что было в этой секции раньше нужно удалить.

Код:

```
function articles_all($link){ //на вход этой функции будет принимать дескриптор $link
    $query="SELECT*FROM articles ORDER BY id DESC"; //производится выбор из всех колонок таблицы articles, затем
производится сортировка по колонке id в убывающем порядке
    $result=mysqli_query($link, $query); //отправка SQL запроса
    if (!$result) //условие ошибки, в следствии неудачного соединения
    die(mysqli_error($link)); //отображение результата ошибки
    $n=mysqli_num_rows($result); //получение количества строк базы в случае удачного соединения
    $articles=array(); //создание пустого массива
    for ($i=0; $i<$n; $i++) //цикл проходит по каждой строке нашей таблицы
```

```

    {
        $row=mysqli_fetch_assoc($result);
        $articles[]=$row; //создание ассоциативного массива строки в таблице и записываем его в $articles[]
    }
return $articles; //возвращение результата
}

```

Используемые команды SQL:

SELECT- выбор

*- все колонки

FROM- из какой таблицы производится выбор

ORDER BY- команда сортировки колонок

DESC- убывающий порядок

Теперь откройте `index.php` в папке `blog` и измените функцию `$articles` следующим образом:

```
$articles=articles_all($link);
```

В файле `articles` в папке `models`, допишите в функцию `articles_get` следующий код:

```

function articles_get($link, $id_article){
    $query=sprintf("SELECT*FROM articles WHERE id=%d", (int)$id_article); // производится выбор из всех колонок таблицы
articles, где id должен быть равен определённому значению
    $result=mysqli_query($link, $query); //запрос SQL
    if (!$result) //условие в случае ошибки
        die(mysqli_error($link)); //вывод ошибки на экран
    $article=mysqli_fetch_assoc($result); //получение результата из базы в виде массива
    return $article; //возвращение этого массива
}

```

Теперь откройте файл `articles.php`, который лежит в папке `blog` и допишите следующее:

```

<?php
    require_once("database.php");
    require_once("models/articles.php");
    $link=db_connect(); //соединение с базой данных
    $article=articles_get($link, $_GET['id']); //передача в качестве 1-го параметра дескриптор соединения.
    include("views/article.php");
?>

```

Лабораторная работа №16

Авторизация в панели администратора

Создание возможности входа по логину и паролю в панель администратора

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям добавьте функцию авторизации для администраторской панели с логином и паролем..

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

АДМИН ПАНЕЛЬ, АВТОРИЗАЦИЯ

Каждый блог имеет панель администратора, где администратор ресурса может добавлять, редактировать и удалять статьи. В этот раз мы сделаем панель авторизации для администратора.

У нас в имеется папка `admin`, которая лежит в папке `blog`, если набрать адрес <http://localhost/blog/admin/>, то у нас высветится сообщение «Admin Panel». Нам нужно сделать так, чтобы при обращении к этой папке из браузера выводилось окно авторизации.

Для этого сначала сделаем ссылку на эту панель на главной странице блога, откройте `articles.php` в папке `views`, затем допишите следующее в код:

```
<...>
<h1>Мой первый блог</h1>
<a href="admin">Панель администратора</a>
<div>
<...>
```

Теперь в редакторе `Brackets` создаём новый файл и называем его «`.htaccess`» и сразу же его сохраняем в папке `admin`. В самом файле пишем следующее:

```
AuthType Basic //Тип авторизации
AuthName "Admin Panel" //Сообщение выводимое на экране для пользователя
AuthUserFile C:/xampp/htdocs/blog/admin/.htpasswd //путь для файла с паролем
Require valid-user //необходимость вводить данные существующего пользователя
```

Теперь необходимо создать файл с паролем. Для этого запустите командную строку и вводите следующее:

```
C:\Users\Администратор>cd ../xampp/apache/bin //переход в папку с программой для генерации файла с паролем.
C:\xampp\apache\bin>htpasswd.exe -c .htpasswd admin //генерация файла с паролем для пользователя admin
```

После того, как вы запустите программу генерации, вас попросят 2 раза ввести пароль. Мы зададим несложный пароль и логин `admin`. Теперь перейдите в папку <C:\xampp\apache\bin> и скопируйте файл «`.htpasswd`» в папку `admin`.

Теперь, если на главной странице нажать на ссылку «Панель администратора» и ввести «Логин: `admin`», «Пароль: `admin`», то мы увидим сообщение «Admin Panel».

Лабораторная работа №17

Вывод статей в панели администратора

Создание интерфейса управления блогом в панели администратора

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям реализуйте возможность вывода всех статей из базы данных в панели администратора. Добавьте ссылки для редактирования, добавления и удаления статей.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

АДМИН ПАНЕЛЬ, ВЫВОД СТАТЕЙ

В этот раз мы выведем список всех статей, которые есть у нас в блоге, а так же добавим функциональные ссылки на редактирование, добавление и удаление статей.

Перейдите в папку `admin` и в файле `index.php` напишите следующий код:

```
<?php
    require_once("../database.php"); //подключаем файл, который служит для обработки
базы данных. Так как файл на уровне выше, нужно ставить двоеточие.
    require_once("../models/articles.php"); //файл для работы со статьями
    $link=db_connect(); //сохранение результата соединения в переменной
    $articles=articles_all($link); //запрос в базу для отображения статей, $link- принимающий
дескриптор.
    Include("../views/articles_admin.php"); //подключение шаблона
?>
```

Теперь в папке `views` создайте файл `articles_admin.php` и запишите туда такой код:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Мой первый блог</title>
        <link rel="stylesheet" href="../style.css">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    </head>
    <body>
        <div class="container">
            <h1>Мой первый блог</h1>
            <div>
                <a href="index.php?action=add">Добавить статью</a>
                <table class="admin-table"> //объявление класса таблицы admin-table
                    <tr>
                        <th>Дата</th>
                        <th>Заголовок</th>
                        <th></th>
                        <th></th>
                    </tr>
                    <?php foreach($articles as $a): ?> //цикл, который будет отображать
статьи
                    <tr>
                        <th><?=$a['date']?></th> //отображение даты публикации из базы данных
                        <th><?=$a['title']?></th> //отображение заголовка статьи из базы данных
                        <td>
```

```

    <a href="index.php?action=edit&id=<?=$a['id']?>">Редактировать</a>
//редактирование статей. Параметр action будет создан позже. Параметр
«<?=$a['id']?>» отвечает за идентификатор статьи, к которому мы обращаемся
    </td>
    <td>
        <a href="index.php?action=delete&id=<?=$a['id']?>">Удалить</a>
    </td>
</tr>
<?php endforeach ?>
</table>
</div>
<footer>
    <p>Мой первый блог<br>Copyright &copy; 2016</p>
</footer>
</div>
</body>
</html>

```

Теперь нужно открыть файл `style.css`, который лежит в папке `blog` и дописать следующее:

```

footer {
    text-align: center;
}

.article {
    margin-top: 50px;
}

.admin-table{
    width: 100%; //выравнивание на всю ширину таблицы класса admin-table.
}

```

Теперь, если обновить страницу, то можно увидеть, что панель стала шире.

Лабораторная работа №18

Добавление статей в панели администратора

Добавление возможности создания новых статей в блоге

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям реализуйте на языке PHP возможность добавления данных в базу используя панель администратора.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

АДМИН ПАНЕЛЬ, ДОБАВЛЕНИЕ СТАТЕЙ

В прошлый раз мы создали панель управления статьями, но не придали ей функциональности. В этот раз мы сделаем функцию для добавления статей в базу данных с последующим её отображением на экран.

Перейдите в папку `views`, создайте и сохраните там файл `article_admin.php`, позже мы отредактируем его. Откройте файл `index.php` в папке `admin` и дополните его следующим кодом:

```
<?php
require_once("../database.php");
require_once("../models/articles.php");
$link=db_connect();
if(isset($_GET['action'])) //если от функции action есть данные для приёма, то приравниваем их к переменной $action
    $action=$_GET['action']; //входящий параметр get action
else
    $action=""; //в случае, если данных для приёма нет, то переменная будет пустой.
if($action=="add"){ //в случае вызова функции добавления статьи $action будет равен add
    if(!empty($_POST)){ //условие для добавления статей
        articles_new($link, $_POST['title'], $_POST['date'], $_POST['content']); //передача данных функции articles_new (она
находится в ../models/articles.php) для добавления в базу.
        header("Location: index.php"); //перенаправление на страницу панели, в случае удачного добавления в базу
    }
    include("../views/article_admin.php"); // в случае вызова функции добавления статьи подгружаем шаблон управления
статьями
}
else{ //в случае, если $action не равно add (функции добавления статьи), то просто выводим список всех статей.
    $articles=articles_all($link);
    include("../views/articles_admin.php");
}
?>
```

Откройте файл `articles.php` в папке `models` и в секцию функции `function articles_new` допишите код:

```
function articles_new($link, $title, $date, $content){ //$link- соединение с базой, $title- заголовок статьи, $date- дата
публикации, $content- текст статьи. Все эти входящие параметры будут обрабатываться этой функцией.
    $title=trim($title); //убираем пробелы у заголовка статьи слева и справа.
    $content=trim($content); //убираем пробелы статьи слева и справа.
    if($title=="") //если заголовок статьи пустой, то будет параметр false
        return false;
    $t="INSERT INTO articles (title, date, content) VALUES ('%s', '%s', '%s')"; //если заголовок не пустой, то соединяемся с
базой данных и записываем в неё заголовок, дату, контент. (%s означает, что тип передаваемых данных является
строкой.)
    $query=sprintf($t, mysqli_real_escape_string($link, $title), //функция подставляет из строки $t данные из $title
mysqli_real_escape_string($link, $date), //$date
mysqli_real_escape_string($link, $content)); //$content
    // mysqli_real_escape_string функция экранирует специальные символы строки, принимая во внимание кодировку
соединения, таким образом, что результат можно безопасно использовать в SQL-запросе в функции
    $result=mysqli_query($link, $query); //выполнение SQL запроса
    if (!$result)
        die(mysqli_error($link)); //если запрос не удался, то вывести ошибку.
    return true;
}
```

Теперь откроем файл `article_admin.php` и запишем туда следующее:

```

<...>
<h1>Мой первый блог</h1>
<div>
  <form method="post"(метод ввода данных) action="index.php?action=add"(передача введённых данных скрипту)>
//форма для ввода данных
  <label>Название<input type="text" (тип вводимых данных- текст) name="title"(имя для поля ввода) value=""
class="form-item"(класс для поля ввода. Пригодится для придания стиля в css) autofocus (сразу после открытия
страницы поле будет готово для ввода) required> //метка с полем ввода.
  </label>
  <label>Дата<input type="date"(выпадающий календарик для выбора даты) name="date" value="" class="form-item"
required>
  </label>
  <label>Содержимое<textarea class="form-item" name="content" required></textarea> //многострочное поле для ввода
данных
  </label>
  <input type="submit" value="Сохранить" class="btn"> //кнопка «сохранить» для отправки данных в базу
</form>
</div>
<...>

```

Теперь, если в панели админа нажать на ссылку «добавить статью», то появится форма для добавления статей. После её заполнения и нажатия кнопки «сохранить» в базу у нас запишется наша статья.

Лабораторная работа №19

Редактирование статей в панели администратора

Создание возможности редактирования статей в панели администратора

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям реализуйте возможность редактирования статей в панели администратора.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.

АДМИН ПАНЕЛЬ, РЕДАКТИРОВАНИЕ СТАТЕЙ

В этот раз мы добавим возможность редактирования существующей статьи у нас в базе.

Откройте файл `index.php` в папке `admin` и допишите следующий код:

```
<?php
require_once("../database.php");
require_once("../models/articles.php");
$link=db_connect();
if(isset($_GET['action']))
    $action=$_GET['action'];
else
    $action="";
if($action=="add"){
    if(!empty($_POST)){
        articles_new($link, $_POST['title'], $_POST['date'], $_POST['content']);
        header("Location: index.php");
    }
    include("../views/admin_add.php");
}
//дополнить новым кодом
else if($action=="edit"){ //в случае вызова функции редактирования
    if(!isset($_GET['id'])) //если id не установлен, то перенаправляем на страницу админской панели.
        header("Location: index.php");
    $id=(int)$_GET['id']; //если id задан, то мы переводим его в целочисленный тип (int) и сразу же присваиваем его
    переменной id.
    if(!empty($_POST)&&$id>0) //если параметры POST не пустые и id больше 0, то передаём данные для сохранения
    функции articles_edit, которая лежит в папке models/articles.php
        articles_edit($link, $id, $_POST['title'], $_POST['date'], $_POST['content']);
        header('Location: index.php'); //в случае удачного сохранения авто переход в панель админа
    }
    $article=articles_get($link, $id); //в случае если данные POST пустые, то получаем список статей для отображения
    include("../views/article_admin.php"); //подключаем шаблон для отображения таблиц со статьями
}
//конец нового кода
else{
    $articles=articles_all($link);
    include("../views/articles_admin.php");
}
?>
```

Теперь, чтобы при добавлении новой статьи у нас поля отображались корректно нужно создать отдельный файл для добавления статей, назовём его `admin_add.php` и лежать он будет в папке `views`. Перед изменением в файле `article_admin.php` всё его содержимое скопируйте в файл `admin_add.php`

Теперь откройте файл `articles_admin.php` в папке `views` и запишите следующие изменения в коде:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Мой первый блог</title>
        <link rel="stylesheet" href="../style.css">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    </head>
```



```

<body>
  <div class="container">
    <h1>Мой первый блог</h1>
    <div>
      <form method="post" action="index.php?action=<?=$_GET['action']?>&id=<?=$_GET['id']?>">
        <label>Название
          <input type="text" name="title" value="<?=$_article['title']?>" (обращаемся к переменной $article, к полю
title) class="form-item" autofocus required>
        </label>
        <label>Дата
          <input type="date" name="date" value="<?=$_article['date']?>" (обращаемся к переменной $article, к полю
date) class="form-item" required>
        </label>
        <label>Содержимое
          <textarea class="form-item" name="content" required><?=$_article['content']?></textarea> //мак как форма
контента имеет тип тега не input, а textarea, то команду для обращения к базе мы поставим между тегами.
        </label>
        <input type="submit" value="Сохранить" class="btn">
      </form>
    </div>
    <footer>
      <p>Мой первый блог<br>Copyright &copy; 2016</p>
    </footer>
  </div>
</body>
</html>

```

В разделе объявления формы нужно добавить переменную для передачи ей типа команды:

```

<form method="post" action="index.php?action=<?=$_GET['action']?>&id=<?=$_GET['id']?>">

```

В этом случае, мы упрощаем работу и не нагружаем код лишней информацией, а лишь заменяем принимающим параметром одной строкой. Т.е. при выборе того или иного элемента в панели администратора, эта функция примет параметр и запустит определённую функцию. Например, мы выбрали команду редактирования, эта команда передаётся этой функции и она запускает раздел редактирования, если мы выберем команду для добавления статьи, то она передаётся этой функции и она в свою очередь запустит форму для добавления статьи.

Теперь нужно дописать одну функцию в файл `articles.php` в папке `models`:

```

function articles_edit($link, (функция для соединения с базой) $id, $title, $date, $content). {
  $title=trim($title);
  $content=trim($content);
  $date=trim($date);
  $id=(int)$id;
  if($title=="")
    return false;
  $sql="UPDATE articles SET title='%s', content='%s', date='%s' WHERE id=%d";
  $query=sprintf($sql, mysqli_real_escape_string($link, $title),
    mysqli_real_escape_string($link, $content),
    mysqli_real_escape_string($link, $date), $id);
  $result=mysqli_query($link, $query);
  if(!$result)
    die(mysqli_error($link));
  return mysqli_affected_rows($link);
}

```

Теперь, если нажать на ссылку редактирования статьи то в полях для ввода данных можно отредактировать уже имеющиеся данные.

Лабораторная работа №20

Удаление статей в панели администратора

Создание возможности удаления статей из базы данных и блога

Задание для выполнения

1. Изучите предложенный теоретический материал.
2. Следуя предложенным инструкциям реализуйте возможность удаления статей из базы данных через панель администратора. Создайте функцию для предварительного просмотра статей на главной странице.
3. Готовый сайт следует запаковать в архив и выложить на сайт.

Содержание отчета:

1. Название работы.
2. Цель выполнения работы.
3. Используемое программное обеспечение.
4. Описание этапов выполнения работы.
5. Готовый сайт, упакованный в архив.

АДМИН ПАНЕЛЬ, УДАЛЕНИЕ СТАТЕЙ

В этот раз мы реализуем возможность удаления статей из базы данных из админской панели управления, так же напишем функцию, которая сокращает текст статьи на определённое количество символов, а полный текст можно будет увидеть только при переходе к определённой статье.

Откройте файл `index.php` в папке `admin` и запишем туда ещё одну функцию, которая будет вызвана при нажатии ссылки удаления:

```
<...>
if(!empty($_POST)&&$id>0){
    articles_edit($link, $id, $_POST['title'], $_POST['date'], $_POST['content']);
    header('Location: index.php');
}
$article=articles_get($link, $id);
include("../views/article_admin.php");
}
else if($action=="delete"){ //в случае вызова функции удаления
    $id=$_GET['id']; //присвоение функции $id параметра $_GET['id']
    $article=articles_delete($link, $id); //соединяемся с базой и передаём id статьи, который нужно удалить.
    header("Location: index.php"); //после успешного удаления переходим в панель администратора.
    }
    else{
        $articles=articles_all($link);
        include("../views/articles_admin.php");
    }
}
?>
```

Открываем файл `articles.php` в папке `models` и в секцию `function articles_delete` дописываем:

```
function articles_delete($link, $id){ //соединение с базой данных и получение id статьи
    $id=(int)$id; //переводим id в целочисленное значение
    if($id==0) //в случае, если id не число или равен 0, то возвращается false
        return false;
    $query=sprintf("DELETE FROM articles WHERE id=%d", $id); //в случае, если условие не проходит, то
    формируем sql запрос в формате: «Удалить строку из таблицы articles с переданным id. Id обозначен как
    передающая параметры переменная $id.
    $result=mysqli_query($link, $query); //выполнение самого запроса
    if(!$result) //в случае, если нет результата, то возвращаем количество строк статей.
        die(mysqli_error($link));
    return mysqli_affected_rows($link); //так как id уникален, то значение этой функции будет либо 0, либо 1.
}
}
```

Теперь создадим функцию по укорачиванию статей на главной странице блога. Для этого после функции `articles_delete` допишем следующие строки:

```
function articles_intro($text, $len=500){ //сначала передаётся текст, а потом количество нужных для
    отображения символов
    return mb_substr($text, 0, $len); //функция копирует текст начиная с 0 позиции до 500. Приставка mb
    означает мультибайтовую кодировку, так как у нас кодировка в utf8.
}
}
```

Теперь откройте `articles.php` в папке `views` и внесём следующие изменения:

```
<...>
<?php foreach($articles as $a): ?>
<div class="article">
  <h3>
    <a href="articles.php?id=<?=$a['id']?>"><?=$a['title']?></a>
  </h3>
  <em>Опубликовано: <?=$a['date']?></em>
  <p><?=$a['content']?></p> //вызываем функцию для сокращения статей на главной
  <span>странице
</div>
<?php endforeach ?>
<...>
```

Теперь, если нажать на ссылку для удаления статей в админской панели, то статья сразу же удалится из базы данных и больше не будет видна, так же, теперь у нас высвечиваются укороченные статьи размером в 500 символов и чтобы прочесть всю статью целиком нужно щёлкнуть по заголовку статьи.

Список используемых источников

1. Голицына О.Л., Попов И.И., Основы алгоритмизации и программирования: Учеб. пособие. – М.: ФОРУМ: ИНФРА-М, 2012. – 432 с.
2. Климов А., JavaScript на примерах, 2-е изд.: СПб.: БХВ-Петербург, 2013. – 336 с.
3. Электронный ресурс удаленного доступа (Internet): – Режим доступа: <https://geekbrains.ru/>
4. Электронный ресурс удаленного доступа (Internet): – Режим доступа: <https://ru.hexlet.io/>