

МИНОБРНАУКИ РОССИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО
ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет *романо-германской филологии*

*Кафедра теории и методики преподавания иностранных
культур*

Проект

**Цифровая среда. Форматы данных и кодировки. Notepad+
+. Geany. Регулярные выражения. Спецсимволы:
допетровская кириллица. Макросы в Sublime Text.**

Выполнила студентка 1го курса

Ельчанинова А.Р.

Преподаватель: Дони́на О.В.

Содержание:

1. Цифровая среда;
2. Форматы данных и кодировки;
3. Notepad++;
4. Geany;
5. Регулярные выражения;
6. Спецсимволы: допетровская кириллица;
7. Макросы в Sublime Text;
8. Задания на закрепление материала;
9. Используемые ресурсы.

№1 «Цифровая среда»

Понятие цифровой среды (пространства) во многом идентично мировому пониманию информационного пространства, определяемого Стратегией развития информационного общества как совокупность информационных ресурсов, созданных субъектами информационной среды, средств взаимодействия таких субъектов, их информационных систем и необходимой информационной инфраструктуры.

В основе любых вычислений лежат операции с цифрами. Поэтому буквально в последние годы и в официальных выступлениях, и в публикациях, и в терминологии различных профессиональных сообществ, включая политиков, военных, стратегистов и т.п., и в повседневном языке все чаще используется термин **«цифровая среда»**.

Цифровая среда представляет собой инфосферу, где содержатся воспринимаемые человеком прямые и скрытые смыслы, выраженные в текстах, таблицах, видео и аудиоконтенте. Ультраструктура включает в себя, во-первых, общедоступные сетевые ресурсы типа сайтов, блогов, порталов, соц. сетей и т.п., во-вторых, защищенные, доступные только для определенных категорий пользователей, информационные ресурсы с платным контентом.

MATLAB — это интерактивная среда для научных и инженерных вычислений. В состав MATLAB входят основная программа (ядро) и специализированные пакеты прикладных программ (toolboxes), состоящие из так называемых М-файлов, расширяющих функциональные возможности основной программы. Один из этих пакетов, Control System Toolbox, в сочетании с основной программой дает возможность использовать MATLAB для анализа и синтеза систем управления. Везде в этой книге, где имеется ссылка на MATLAB,

подразумевается, что решение той или иной задачи производится с помощью основной программы и пакета Control System Toolbox.

Лабораторная работа №1

Цель работы – ознакомление с системой MatLab, правилами создания числовых массивов и приобретение практических навыков по использованию средств системы для работы с ними.

В интерфейс системы MatLab входят следующие панели:

- Command Window (Окно Команд), где проводятся все расчеты и операции;
- Launch Pad (Окно Разделов), где можно получить доступ к различным модулям ToolBox;
-
- Workspace (Рабочее пространство), где отображается текущий набор переменных, введенных пользователем в командном окне;
-
- Current Directory (Текущий каталог), где можно установить текущий каталог;
-
- Command History (История команд), где хранятся команды, набираемые пользователями.

Матричная система MatLab выделяется из других систем тем, что ее операторы и функции имеют операнды в виде векторов и матриц. Так как операции с матрицами могут быть как поэлементными, так и матричными, то в поэлементные операторы добавляется точка. Например, символы точка, звездочка '*' определяют поэлементное умножение массивов, символ звездочка '*' – матричное умножение (табл. 1). Набор любой команды должен заканчиваться нажатием клавиши . Действие, выполняемое функцией, применяется ко всем элементам массива, передаваемым в списке входных аргументов.

Получить справочную информацию можно следующими операторами:

helpwin– справка о разделах и функциях системы MatLab;

helpdesk– общая справка о системе MatLab;

doc <имя_функции> –вывод описания функции в окнеHelp ; help
<имя_функции> –краткая информация о функции;

type <имя _функции> –вывод текст-файла функции; demo –команда вызова тестовых примеров.

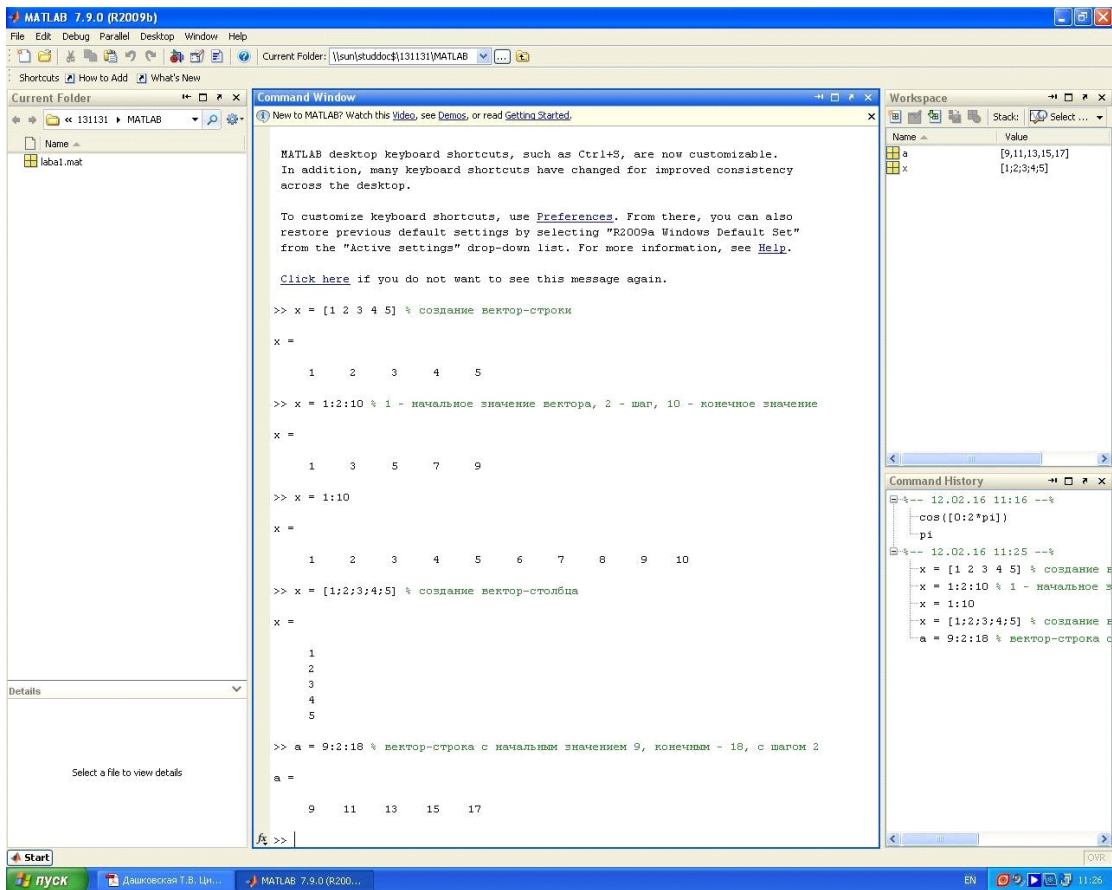
Для создания вектор-строки используются квадратные скобки с перечислением элементов строки через пробел или запятую и специальная конструкция j:i:k с указанием начального значения вектора – j, шага – i и конечного значения вектора – k через двоеточие (если значение шага равно 1, его можно не указывать).

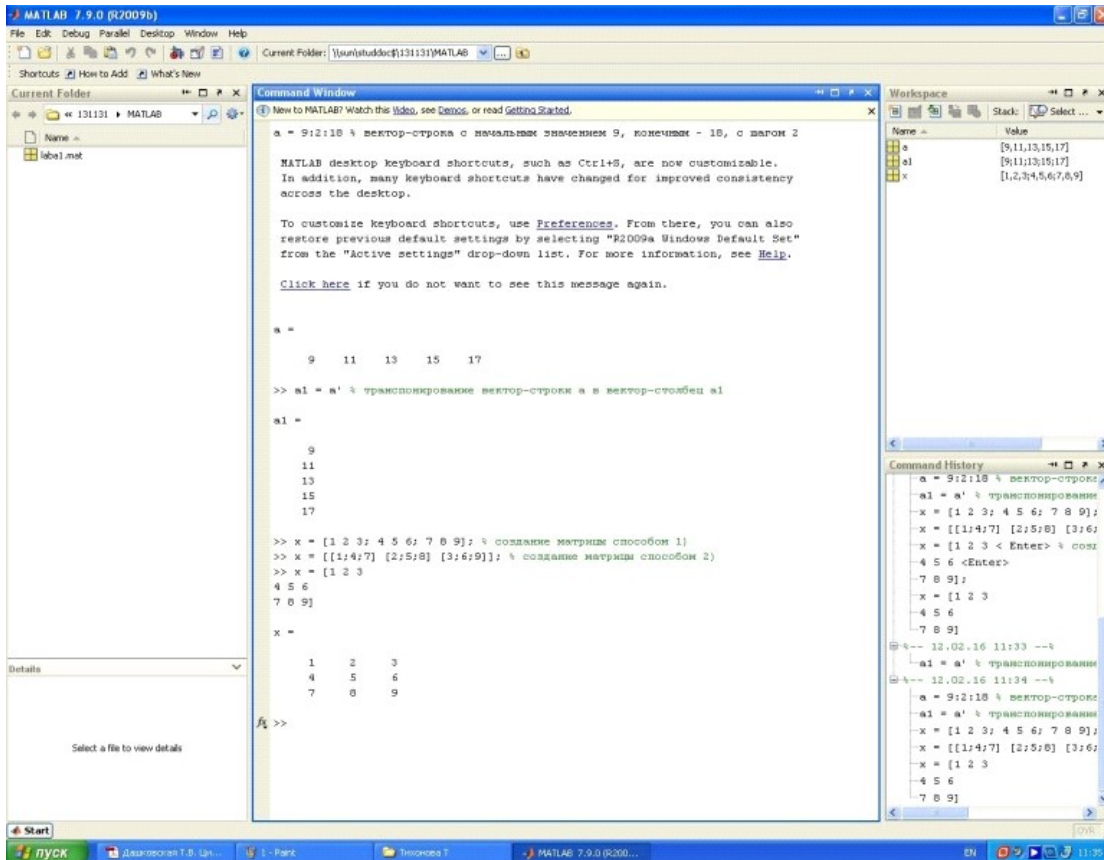
Для создания вектор-столбца элементы вектора перечисляются через точку с запятой в квадратных скобках или транспонируется полученный ранее вектор-строка. Для выполнения операции транспонирования используется одиночная кавычка ('), которая ставится после идентификатора, определяющего транспонируемую структуру. Для комплексных матриц транспонирование дополняется сопряжением матрицы. Точка с одиночной кавычкой (.'') используется для транспонирования массива без операции сопряжения для комплексных матриц.

Для создания матрицы можно использовать следующие способы ввода элементов в квадратных скобках:

- По строкам, разделяющимся точкой с запятой;
- По столбцам, заданным в квадратных скобках;
- По строкам в интерактивном режиме.

Задание 1. Создать вектор-строку, вектор-столбец и матрицы





Конкатенация (объединение) массивов

С помощью операции конкатенации можно формировать новые массивы из ранее созданных – векторов, матриц, используя эти массивы в качестве своих элементов. Объединять массивы можно по горизонтали и по вертикали.

При горизонтальной конкатенации в качестве разделителя массивов в квадратных скобках используется запятая или пробел, например, если В и А – матрицы, то $M = [A, B]$ – горизонтальная конкатенация матриц А и В. А и В должны иметь одинаковое число строк. Горизонтальная конкатенация может быть применена для любого числа матриц в пределах одних скобок $[A, B, C]$.

При вертикальной конкатенации в качестве разделителя массивов в квадратных скобках используется точка с запятой, например, если С и D – матрицы, то $M = [C; D]$ – вертикальная конкатенация матриц С и D. С и D должны иметь одинаковое число столбцов. Вертикальная

конкатенация может быть применена для любого числа матриц в пределах одних скобок [C; D; E]. Вертикальная и горизонтальная конкатенация может быть применена в одной операции.

Задание 2. Создать матрицу, используя вертикальную и горизонтальную конкатенацию

The screenshot shows the MATLAB 7.9.0 (R2009b) environment. The Command Window displays the following code and its output:

```
>> b = [1;2;3]

b =

     1
     2
     3

>> A = [b,b,b]

A =

     1     1     1
     2     2     2
     3     3     3

>> c = [1 4 8];
>> B = [c;c;c]

B =

     1     4     8
     1     4     8
     1     4     8

>> A1 = [0 0;0 1]; % создание матриц A1 и A2 размерностью 2 2
A2 = [5 6;1 8];
A3 = [1 8 6 9]; % создание вектор-строки из четырех элементов
H = [A1,A2;A3]

H =

     0     0     5     6
     0     1     1     8
     1     8     6     9

>> x = zeros(2,3)

x =

     0     0     0
     0     0     0

>>
```

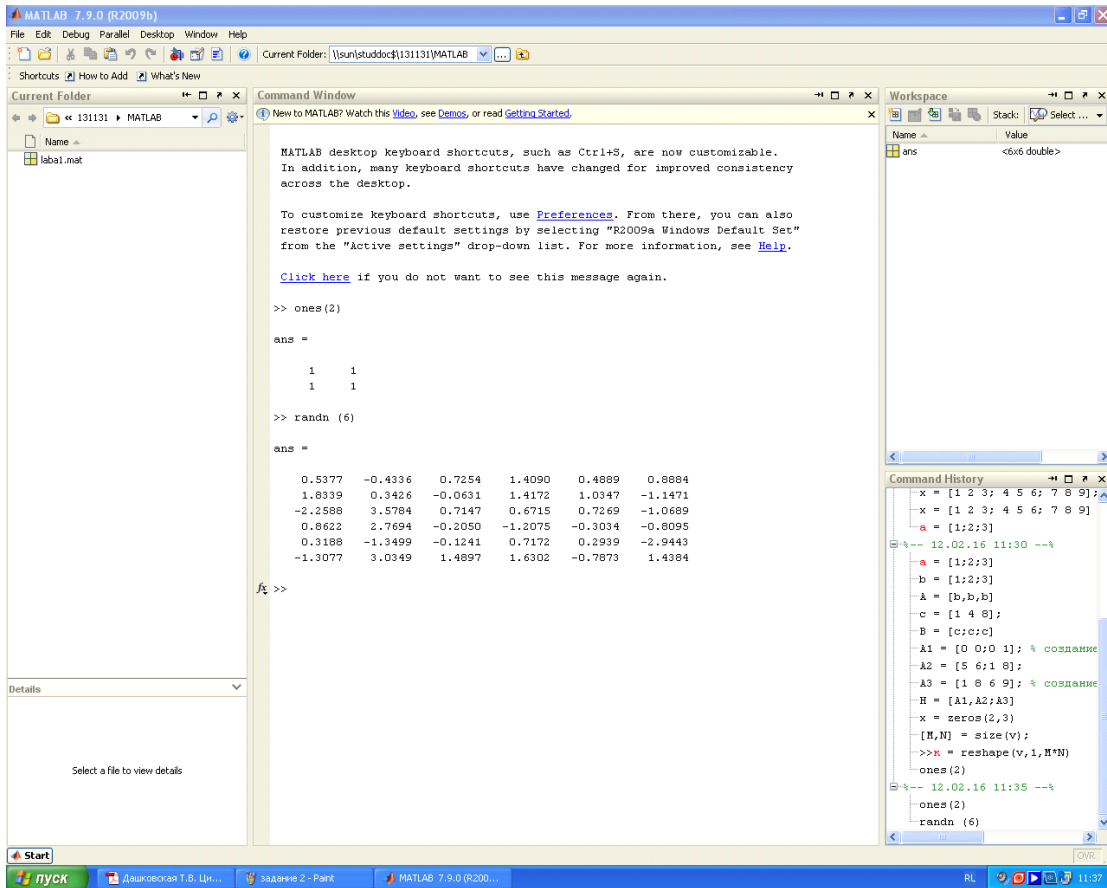
The Workspace window shows the following variables and their values:

Name	Value
A	[1,1,2,2,3,3,3]
A1	[0,0,0,1]
A2	[5,6,1,8]
A3	[1,8,6,9]
B	[1,4,8;1,4,8;1,4,8]
H	<3x4 double>
b	[1;2;3]
c	[1,4,8]
x	[0,0,0,0,0,0]

The Command History window shows the following commands:

```
x = [1 2 3 4 5];
x = 1:2:10;
x = 1:10;
x = [1;2;3;4;5];
a = 9:2:18;
a1 = a';
x = [1 2 3; 4 5 6; 7 8 9];
x = [1 2 3; 4 5 6; 7 8 9];
a = [1;2;3];
-- 12.02.16 11:30 --%
a = [1;2;3];
b = [1;2;3];
A = [b,b,b];
c = [1 4 8];
B = [c;c;c];
A1 = [0 0;0 1]; % создание
A2 = [5 6;1 8];
A3 = [1 8 6 9]; % создание
H = [A1,A2;A3];
x = zeros(2,3);
```

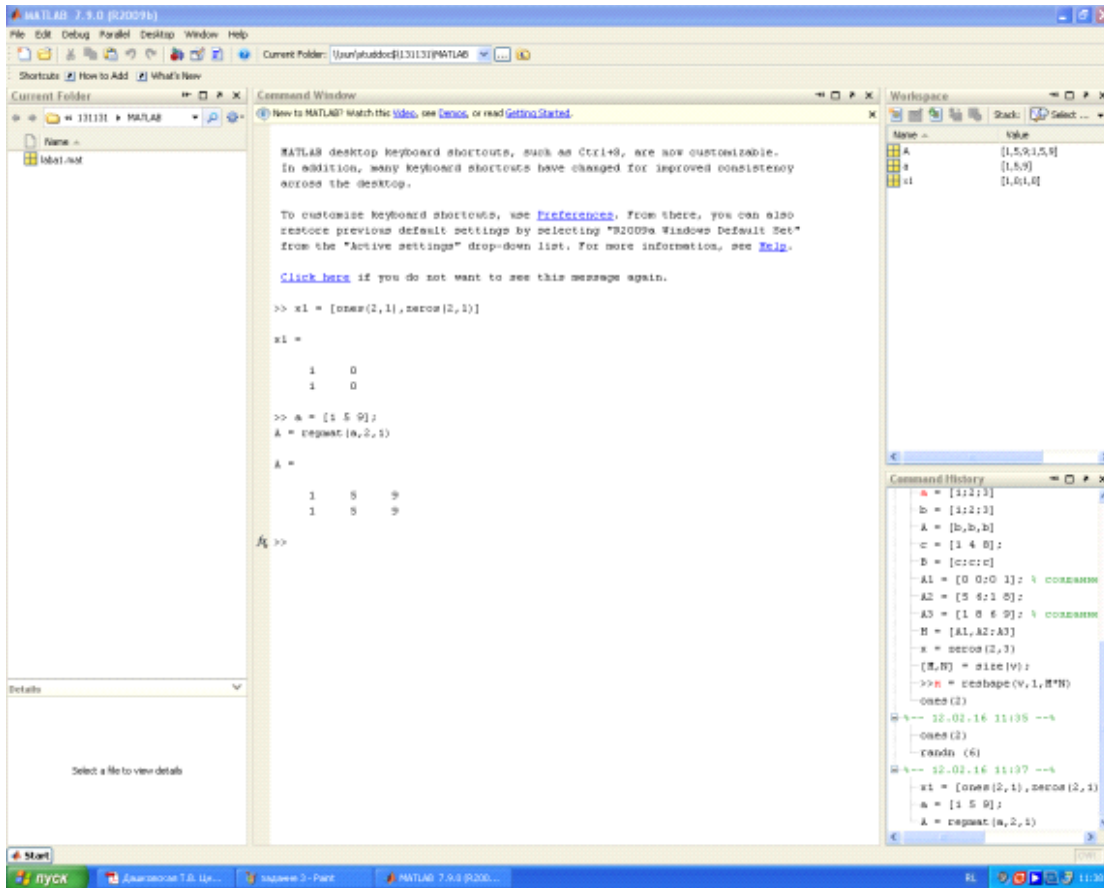
Задание 3. Создать квадратную единичную матрицу размерностью 2



Задание 4. Создать вектор-столбец с помощью вертикальной конкатенации



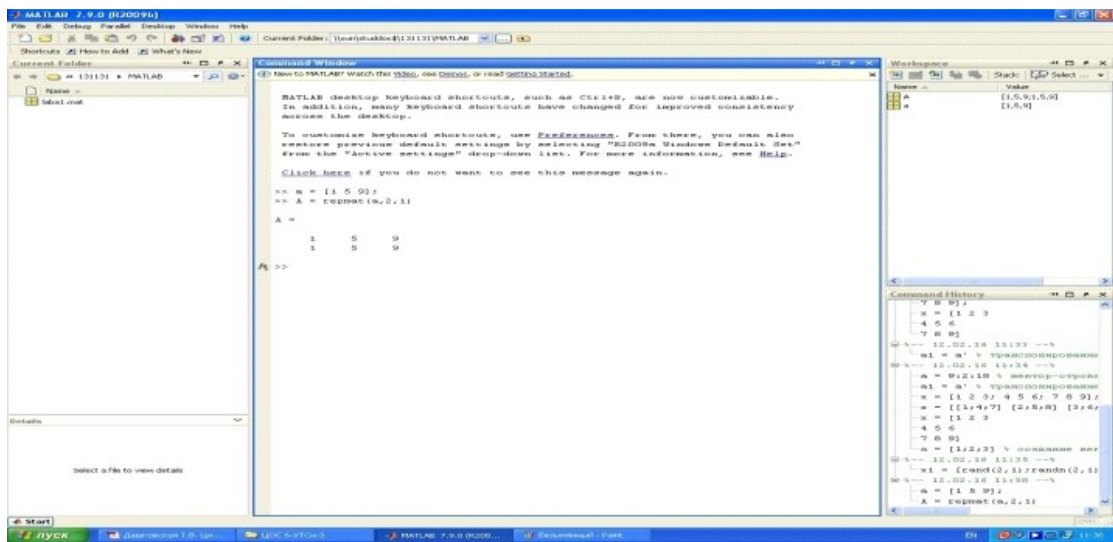
Задание 5. Создать матрицу с помощью



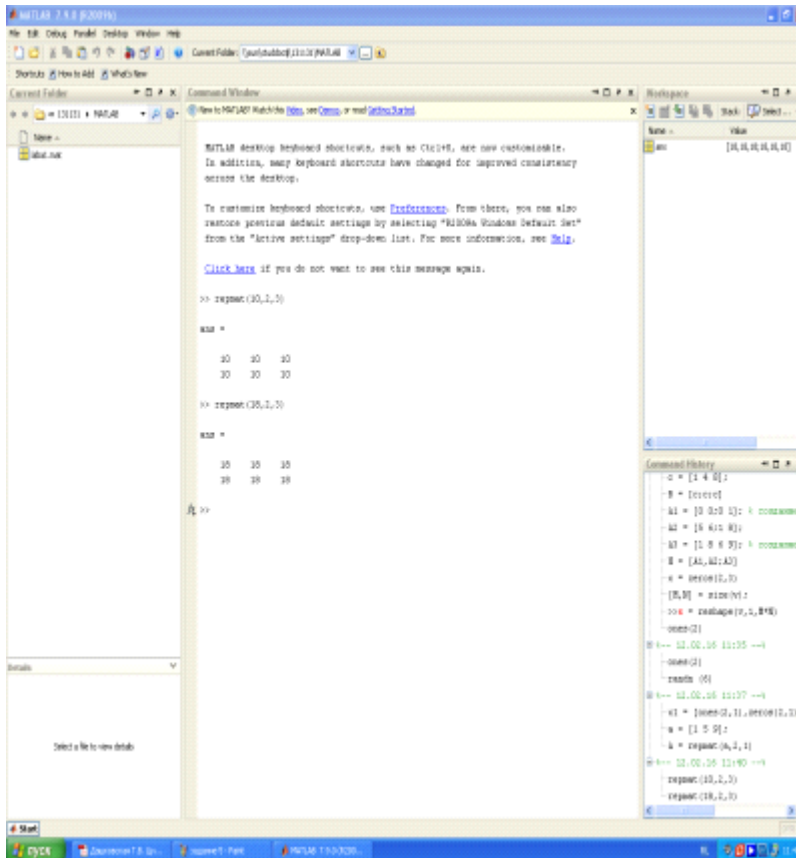
Функция `repmat()` создает матрицу, копируя исходный массив заданное число раз по вертикали и горизонтали.

$B = \text{repmat}(A, M, N)$ – функция создает матрицу B , состоящую из M копий A по вертикали и N копий A по горизонтали, то есть $M \cdot N$ копий массива A (если A – число, функция формирует матрицу размером $M \cdot N$ со значением элементов, равных A).

Задание 6. Сформировать матрицу с использованием вектор-строки из трех элементов



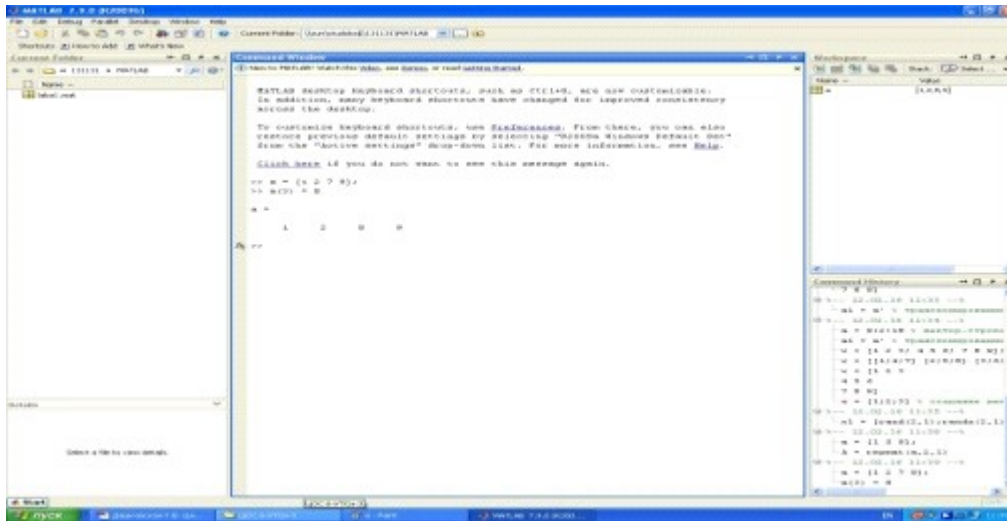
Задание 7. Сформировать матрицу размерностью 2 3, все элементы которой равны десяти



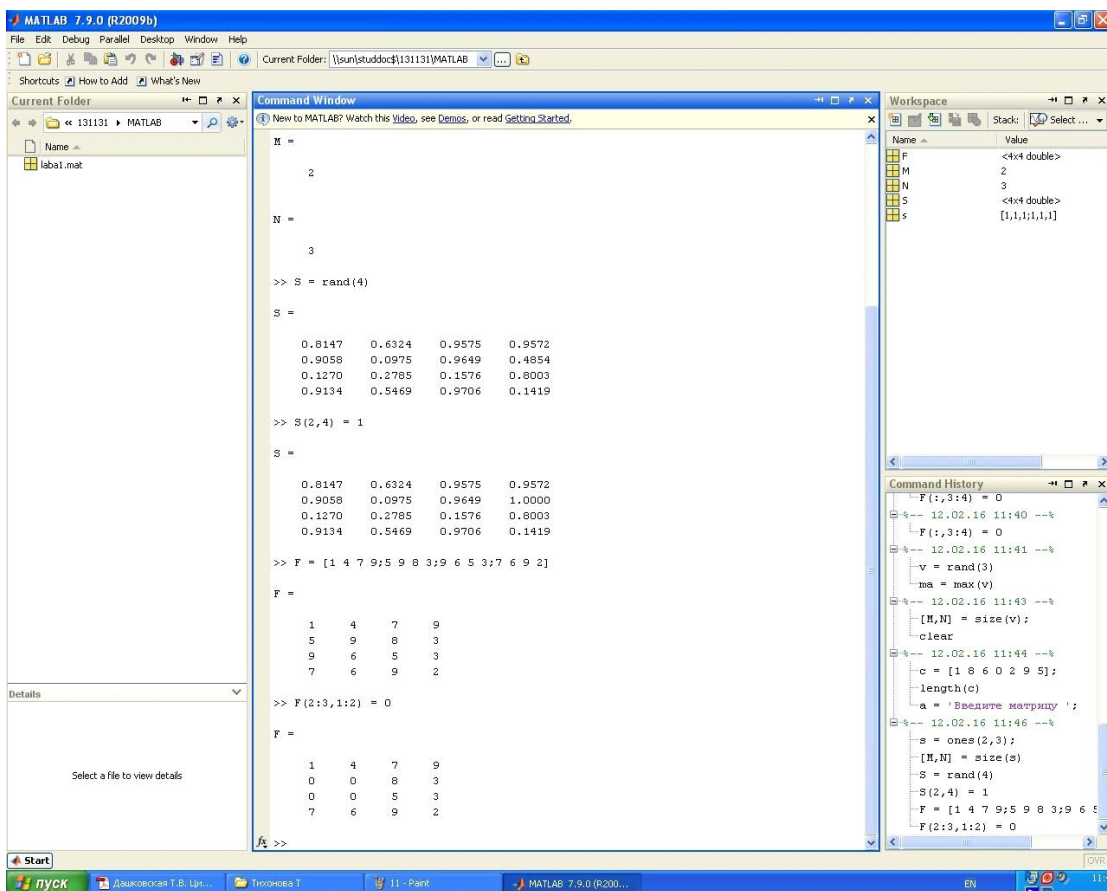
Индексация массивов

Элементы массивов обладают двумя свойствами: порядковым номером (индексом) в массиве и собственно значением. Нумерация элементов в системе MatLab начинается с единицы. Для указания индексов элементов массивов используются круглые скобки (ошибка при индексации массива генерируется в том случае, если индекс элемента меньше единицы или больше размера массива).

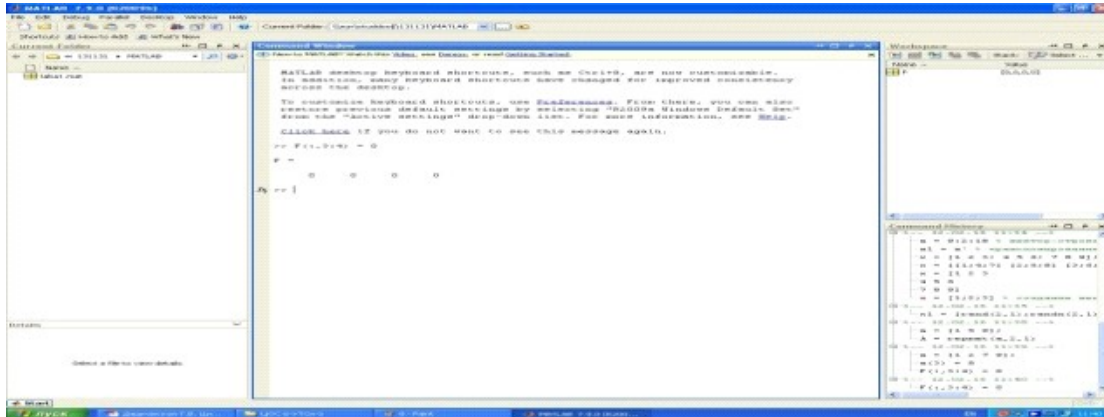
Задание 8. Задать вектор-строку из четырех элементов и изменить третий элемент на значение 8



Задание 9. Изменить значение элемента матрицы случайных чисел S , находящегося во второй строке и в четвертом столбце, на 1



Задание 10. Обнулить третий и четвертый столбец из предыдущего примера



Пустые квадратные скобки удаляют информацию из индексированной структуры.

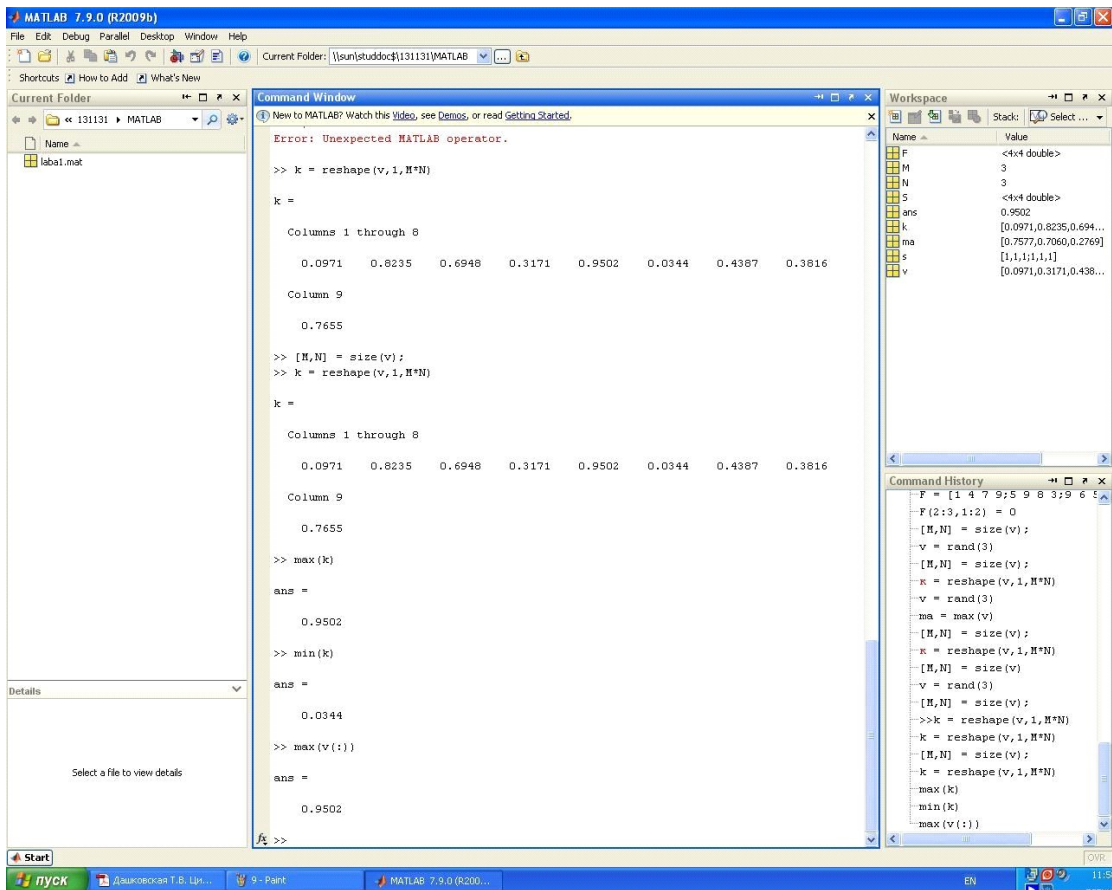
$A(m,:) = []$ – удаляет строку m из матрицы A . $A(:,n) = []$ – удаляет столбец n из матрицы A .

Сервисные функции

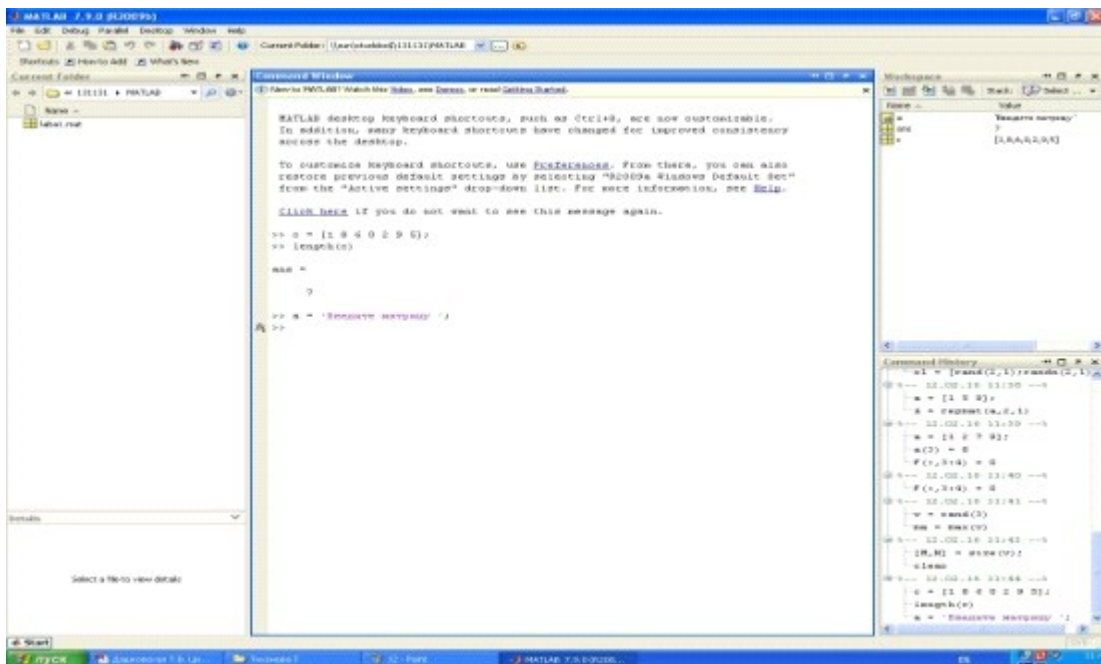
Ниже приведены некоторые функции, необходимые при работе с массивами:

$[M,N] = \text{size}(\text{идентификатор_массива})$ – возвращает размер массива, где M – число строк; N – число столбцов.

Задание 11. Определить размерность единичной матрицы



Задание 14. Определить длину заданного вектора



№2 «Форматы данных и кодировки»

Кодирование текстовых данных и форматы текстовых файлов. Если каждому символу алфавита сопоставить определенное целое число, то с помощью двоичного кода можно кодировать и текстовую информацию. 8 разрядов – 256 символов: английские и русские буквы строчные и прописные, знаки препинания, арифметических действий и некоторые общепринятые специальные символы (% , № , "). Для того, чтобы весь мир одинаково кодировал текстовые данные, нужны единые таблицы кодирования, а это пока невозможно из-за противоречий между символами национальных алфавитов и противоречий корпоративного характера. Для английского языка противоречия уже сняты. Институт стандартизации США ввел в действие систему кодирования ASCII (American Standard Code for Information Interchange – стандартный код информационного обмена США). ASCII: базовая (0 – 127) и расширенная (128 – 255) таблицы кодирования; – с 0 по 31 код отданы производителям аппаратных средств (компьютеров и печатающих устройств), это так называемые управляющие коды, которым не соответствуют никакие символы языка (эти коды не выводятся ни на экран, ни на печать), но они могут управлять выводом других данных; – 32 – 127: коды символов английского алфавита, цифр и др. Аналогичные системы кодирования были разработаны и в других странах. Поддержка производителей оборудования и программ вывела американский код ASCII на уровень международного стандарта, и национальным системам кодирования пришлось отступить на вторую, расширенную часть системы кодирования (128-255 коды). В России: – кодировка Windows-1251, введенная компанией Microsoft, – ввиду большого распространения

программ этой компании; – КОИ-8 (код обмена информацией, восьмизначный): произошла в период действия Совета Экономической Взаимопомощи (СЭВ) государств Восточной Европы; имеет широкое распространение в компьютерных сетях на территории России и в российском секторе Интернета. Международный стандарт, в котором предусмотрена кодировка символов русского алфавита, – кодировка ISO (International Standard Organization – Международный институт стандартизации), на практике используется редко. На компьютерах, работающих в MS-DOS, могут действовать кодировки ГОСТ(устаревшая) и ГОСТ-альтернативная (используется и сейчас). Универсальная система кодирования текстовых данных. Трудности с созданием единой системы кодирования связаны с ограниченным набором кодов (256 – 8 разрядов). Система, основанная на 16-разрядном кодировании символов, получила название универсальной – Unicode. 16 разрядов позволяют обеспечить уникальные коды для 65 536 различных символов.

ASCII - это код для представления символов английского алфавита в виде чисел, каждой букве сопоставлено число от 0 до 127. В большинстве компьютеров код ASCII используется для представления текста, что позволяет передавать данные от одного компьютера на другой.

Текстовый файл, запомненный в формате ASCII, иногда называют ASCII-файлом. Текстовые редакторы и текстовые процессоры обычно могут сохранять данные в формате ASCII. Большинство файлов данных, особенно, если они содержат числовые данные, сохраняются не в ASCII формате. Исполняемые программы никогда не сохраняются в формате ASCII.

Стандартный набор символов ASCII использует только 7 битов для каждого символа. Есть несколько наборов символов, которые используют 8 бит, что дает дополнительно 128 символов.

Дополнительные символы используются для отображения символов не-английского алфавита, графических и математических символов. В операционной системе DOS используется надмножество ASCII, называемое расширенный ASCII. Более универсальным является набор символов ISO Latin 1, который используется во многих операционных системах и в браузерах.

Юникод, или Уникод (англ. Unicode) — стандарт кодирования символов, позволяющий представить знаки практически всех письменных языков. Стандарт предложен в 1991 году некоммерческой организацией «Консорциум Юникода» (англ. Unicode Consortium), объединяющей крупнейшие IT-корпорации. Стандарт состоит из двух основных разделов: универсальный набор символов (UCS, Universal Character Set) и семейство кодировок (UTF, Unicode Transformation Format).

Коды в стандарте Unicode разделены на несколько областей. Область с кодами от U+0000 до U+007F содержит символы набора ASCII с соответствующими кодами. Далее расположены области знаков различных письменностей, знаки пунктуации и технические символы. Часть кодов зарезервирована для использования в будущем. Под символы кириллицы выделены коды от U+0400 до U+052F

Хотя формы записи UTF-8 и UTF-32 позволяют кодировать до 2³¹ (2 147 483 648) кодовых позиций, было принято решение использовать лишь 2²⁰+2¹⁶ (1 114 112) для совместимости с UTF-16. Впрочем, даже и этого более чем достаточно — сегодня (в версии 5.0) используется чуть больше 99 000 кодовых позиций.

Кодовое пространство разбито на 17 плоскостей по 2¹⁶ (65536) символов. Нулевая плоскость называется базовой, в ней расположены символы наиболее употребительных письменностей. Первая плоскость используется, в основном, для исторических письменностей. Плоскости 16 и 17 выделены для частного употребления.

Для обозначения символов Unicode используется запись вида «U+xxxx» (для кодов 0...FFFF) или «U+xxxxx» (для кодов 10000...FFFFF) или «U+xxxxxx» (для кодов 100000...10FFFF), где xxx — шестнадцатеричные цифры. Например, символ «я» (U+044F) имеет код 044F16 = 110310.

Юникод имеет несколько форм представления (англ. Unicode Transformation Format, UTF): UTF-8, UTF-16 (UTF-16BE, UTF-16LE) и UTF-32 (UTF-32BE, UTF-32LE)

Кодировка цвета.

Любой цвет можно представить в виде комбинации трёх основных цветов: красного, зелёного и синего (их называют цветовыми составляющими). Если закодировать цвет точки с помощью трёх байтов (24 бита), то первый байт будет нести информацию о красной составляющей, второй - зелёной, а третий - синей. Чем больше значение байта цветовой составляющей, тем ярче этот цвет. Задавая любые значения (от 0 до 255) для каждого из трёх байтов, с помощью которых кодируется цвет, можно закодировать любой из 16,5 миллионов цветов.

Форматы графических файлов.

RAW- В переводе с английского — сырой. Формат использующийся в процессе обработки фотографий, содержит необработанную информацию, поступающую напрямую с матрицы фотокамеры и не имеющий чёткой спецификации. Эти файлы не обрабатываются процессором камеры (в отличие от JPG) и содержат оригинальную информацию о съемке. RAW может быть сжат без потери качества. В отличие от JPG, который был обработан в камере и уже сохранен с сжатием данных – RAW дает широчайшие возможности по обработке фотографии и сохраняет максимальное качество.

JPEG (или JPG)-самый распространенный формат графических файлов. Свою популярность JPEG заслужил гибкой возможностью сжатия данных. При необходимости изображение можно сохранить с максимальным качеством. Либо сжать его до минимального размера файла для передачи по сети. При сохранении JPEG-файла можно указать степень качества, а значит и степень сжатия, которую обычно задают в некоторых условных единицах, например, от 1 до 100 или от 1 до 10. Большее число соответствует лучшему качеству, но при этом увеличивается размер файла. Обыкновенно, разница в качестве между 90 и 100 на глаз уже практически не воспринимается.

На практике, сохранение фотографии с минимальной степенью сжатия не дает видимого ухудшения качества изображения. Именно поэтому JPG – самый распространенный и популярный формат хранения графических файлов.

TIFF (Tagged Image File Format) Формат TIFF — Он позволяет сохранять фотографии в различных цветовых пространствах (RGB, CMYK, YCbCr, CIE Lab и пр.) и с большой глубиной цвета (8, 16, 32 и 64 бит). TIFF используется при сканировании, отправке факсов, распознавании текста, в полиграфии, широко поддерживается графическими приложениями. Имеется возможность сохранять изображение в файле формата TIFF со сжатием и без сжатия. Степени сжатия зависят от особенностей самого сохраняемого изображения, а также от используемого алгоритма. В отличие от JPG, изображение в TIFF не будет терять в качестве после каждого сохранения файла. Но, к сожалению, именно из-за этого TIFF файлы весят в разы больше JPG.

PSD (Photoshop Document)— оригинальный растровый формат хранения графической информации, использующий сжатие без потерь, созданный специально для программы Adobe Photoshop и поддерживающий все его возможности. Он позволяет сохранять растровое изображение со многими слоями, любой глубиной цвета и в любом цветовом пространстве. Чаще всего формат используется для

сохранения промежуточных или итоговых результатов сложной обработки с возможностью изменения отдельных элементов. Так же PSD поддерживает сжатие без потери качества. Но обилие информации, которое может содержать PSD файл, сильно увеличивает его вес.

BMP (Bit MaP image)— универсальный формат растровых графических файлов, используется в операционной системе Windows. Этот формат поддерживается многими графическими редакторами, в том числе редактором Paint. Рекомендуется для хранения и обмена данными с другими приложениями. Формат BMP один из первых графических форматов. Его распознает любая программа работающая с графикой. BMP хранит данные с глубиной цвета в данном формате от 1 до 48 бит на пиксель, максимальные размеры изображения 65535×65535 пикселей. На данный момент формат BMP практически не используется ни в интернете (JPG весит в разы меньше), ни в полиграфии (TIFF справляется с этой задачей лучше).

GIF (Graphics Interchange Format) - способен хранить сжатые данные без потери качества в формате до 256 цветов. Включает алгоритм сжатия без потерь информации, позволяющий уменьшить объем файла в несколько раз. Изображение в формате GIF хранится построчно, поддерживается только формат с индексированной палитрой цветов. Рекомендуется для хранения; изображений, создаваемых программным путем (диаграмм, графиков и так далее) и рисунков (типа аппликации) с ограниченным количеством цветов (до 256). Используется для размещения графических изображений на Web-страницах в Интернете.

PNG (Portable network graphics)-

Растровый формат хранения графической информации, использующий сжатие без потерь. PNG был создан как для улучшения, так и для замены формата GIF графическим форматом, не требующим лицензии для использования. В отличие от GIF, у PNG есть поддержка альфа-

канала и возможность хранить неограниченное количество цветов. PNG сжимает данные без потерь, что делает его очень удобным для хранения промежуточных версий обработки изображений. Используется для размещения графических изображений на Web-страницах в Интернете.

JPEG 2000 (или jp2)

Графический формат, который вместо дискретного косинусного преобразования, характерного для JPEG, использует технологию вейвлет-преобразования, основывающуюся на представлении сигнала в виде суперпозиции некоторых базовых функций — волновых пакетов. В результате такой компрессии изображение получается более гладким и чётким, а размер файла по сравнению с JPEG при одинаковом качестве уменьшается ещё на 30%. Говоря простым языком, при одинаковом качестве размер файла в формате JPEG 2000 на 30% меньше, чем JPG. При сильном сжатии JPEG 2000 не разбивает изображение на квадраты, характерные формату JPEG. К сожалению, на данный момент этот формат мало распространён и поддерживается только браузерами Safari и Mozilla/Firefox (через Quicktime).

WMF (Windows MetaFile)

Универсальный формат векторных графических файлов для Windows-приложений. Используется для хранения коллекции графических изображений Microsoft Clip Gallery.

CDR (CorelDraw files)

Оригинальный формат векторных графических файлов, используемый в системе обработки векторной графики CorelDraw.

AI (Adobe Illustrator files)

Оригинальный формат векторных графических файлов, используемый в системе обработки векторной графики Adobe Illustrator.

Формат векторных графических файлов, поддерживается программами для различных операционных систем.

Лабораторная работа №2

Цель работы: изучение понятия – «машинное представление информации», машинное представление действительных чисел, машинное представление текстовой информации (ASCII – коды).

2. Используемые технические средства: персональный компьютер, ОС Windows 9x/XP.

3. Программа работы.

- 3.1. Ознакомиться с базовыми положениями «машинного представления действительных чисел», используя лекционный материал и **Приложение 1**.
- 3.2. Ознакомиться с двоичным кодированием текстовой информации с помощью ASCII -кодов, используя **Приложение 2**.
- 3.4. Выполнить задания своего варианта.
- 3.5. Сделать выводы и оформить отчет о проделанной лабораторной работе.

Приложение 1

Для унификации представления чисел с плавающей точкой институт инженеров по электротехнике и радиоэлектронике (IEEE) разработал стандарт IEEE 754 (формат с плавающей точкой обычной точности).

. В последнее десятилетие практически все процессоры проектируются с учетом этого стандарта. Рассмотренный в лекции вариант представления числа с плавающей является форматом IEEE 754 с плавающей точкой обычной точности.

Таким образом, для представления числа в формате IEEE 754 с плавающей точкой с обычной точностью необходимо придерживаться следующего алгоритма:

1. представить число в двоичном виде – $10.625_{10} = 1010.101$;
2. записать число в научной нотации, мантисса должна быть нормализована – $1010.101 = 0.1010101 \cdot 2^4 = 0.1010101 \cdot 2^{100}$;
3. записать знак числа – 0;
4. записать порядок – $4_{10} + 127_{10} = 100 + 0111\ 1111 = 1000\ 0011$;

5. привести окончательную запись числа – 0 10000011
010101000000000000000000.

Приложение 2

Форматы представления текста в ЭВМ

Для представления текстовой информации используется алфавитное кодирование, т.е. каждому символу – значку, цифре или букве, ставится в соответствии его двоичный код. Американский национальный институт стандартов (ANSI) принял 8 – битный код для текстовой информации ASCII – American Standard Code for Information Interchange. Изначально предполагалось использовать только 7 бит (127 – вариантов символов), а восьмой оставить для контроля. Поэтому в стандарте строго регламентированы только первые 128 (0 – 127) позиций. Широкое распространение ASCII – кодов привело к тому, что их стало не хватать для символов различных языков и других целей.

Поэтому оставшиеся возможности (128 – 255) стали использовать под различные национальные кодировки. В связи с этим первые 128 позиций строго регламентированы, а оставшиеся позиции используются по мере необходимости. В России для отображения кириллицы широкое распространение получили следующие варианты кодировок: КОИ8, 866-MS DOS, 1251-Windows.

Таблица 3.1

0 (nul)	16 ► (dle)	32 (sp)	48 0	64 @	80 P	96 `	112 p
1 ☺ (soh)	17 ◀ (dc1)	33 !	49 1	65 A	81 Q	97 a	113 q
2 ● (stx)	18 ↑ (dc2)	34 "	50 2	66 B	82 R	98 b	114 r
3 ♥ (etx)	19 !! (dc3)	35 #	51 3	67 C	83 S	99 c	115 s
4 ♦ (eot)	20 ¶ (dc4)	36 \$	52 4	68 D	84 T	100 d	116 t
5 ♣ (enq)	21 § (nak)	37 %	53 5	69 E	85 U	101 e	117 u
6 ♠ (ack)	22 — (syn)	38 &	54 6	70 F	86 V	102 f	118 v
7 • (bel)	23 ↓ (etb)	39 '	55 7	71 G	87 W	103 g	119 w
8 ▣ (bs)	24 ↑ (can)	40 (56 8	72 H	88 X	104 h	120 x
9 (tab)	25 ↓ (em)	41)	57 9	73 I	89 Y	105 i	121 y
10 (lf)	26 → (eof)	42 *	58 :	74 J	90 Z	106 j	122 z
11 ♂ (vt)	27 ← (esc)	43 +	59 ;	75 K	91 [107 k	123 {
12 ♀ (np)	28 ⊞ (fs)	44 ,	60 <	76 L	92 \	108 l	124
13 (cr)	29 ↔ (gs)	45 -	61 =	77 M	93]	109 m	125 }
14 ♪ (so)	30 ▲ (rs)	46 .	62 >	78 N	94 ^	110 n	126 ~
15 ☼ (si)	31 ▼ (us)	47 /	63 ?	79 O	95 _	111 o	127 △

Далее приводятся ASCII – коды кириллицы в варианте 866-MS DOS.

Таблица 3.2

128 A	144 P	160 a	176 ☐	192 L	208 Ш	224 p	240 Ё
-------	-------	-------	-------	-------	-------	-------	-------

129 Б	145 С	161 б	177	193 ⊥	209 ⊥	225 с	241 ё
130 В	146 Т	162 в	178	194 ⊥	210 ⊥	226 т	242 €
131 Г	147 У	163 г	179	195 ⊥	211 ⊥	227 у	243 €
132 Д	148 Ф	164 д	180	196 —	212 ⊥	228 ф	244 İ
133 Е	149 Х	165 е	181	197 ⊥	213 ⊥	229 х	245 İ
134 Ж	150 Ц	166 ж	182	198 ⊥	214 ⊥	230 ц	246 Ÿ
135 З	151 Ч	167 з	183	199 ⊥	215 ⊥	231 ч	247 Ÿ
136 И	152 Ш	168 и	184	200 ⊥	216 ⊥	232 ш	248 °
137 Й	153 Щ	169 й	185	201 ⊥	217 ⊥	233 щ	249 ·
138 К	154 Ъ	170 к	186	202 ⊥	218 ⊥	234 ъ	250 ·
139 Л	155 Ы	171 л	187	203 ⊥	219	235 ы	251 √
140 М	156 Ь	172 м	188	204 ⊥	220	236 ь	252 №
141 Н	157 Э	173 н	189	205 =	221	237 э	253 №
142 О	158 Ю	174 о	190	206 ⊥	222	238 ю	254 ■
143 П	159 Я	175 п	191	207 ⊥	223	239 я	255

Задание

6. Представить числа в формате с плавающей точкой обычной точности.
7. Перевести число из двоичного формата с плавающей точкой обычной точности в десятичное представление.
8. Используя ASCII кодировку в варианте 866 (MS-DOS), представить в двоичном виде текст.
9. Расшифровать ASCII код.

Варианты заданий к лабораторной работе

№1

1) 32128. 25; – 20.0056 2) 0 11100010 011110000011100000000000
1 11100010 011110000011100000000000 3) «Привет student № 1» 4) 48 65 6C
6C 6F 20 4E 31

№2

1) 10128. 025; – 320.054 2) 0 11000010 011010000011100000000000
1 11000010 011010000011100000000000 3) «Привет student № 2» 4) 48 65 6C
6C 6F 20 4E 32

№3

1) 9711. 34; – 120.059 2) 0 10100010 001110000011100000000000

1 10100010 00111000001110000000000 3) «Привет student № 3» 4) 48 65 6C
6C 6F 20 4E 33

№3 «Notepad++»

Notepad++ - свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки. Поддерживает открытие более 100 форматов.

В чем основные особенности программы?

1. Программа с открытым исходным кодом, абсолютно бесплатна.
2. Расширяемый функционал за счет плагинов.
3. Поддержка большинства популярных языков программирования.
4. Notepad++ умеет автоматически закрывать теги.
5. Полностью русифицирована.

Что умеет Notepad++?

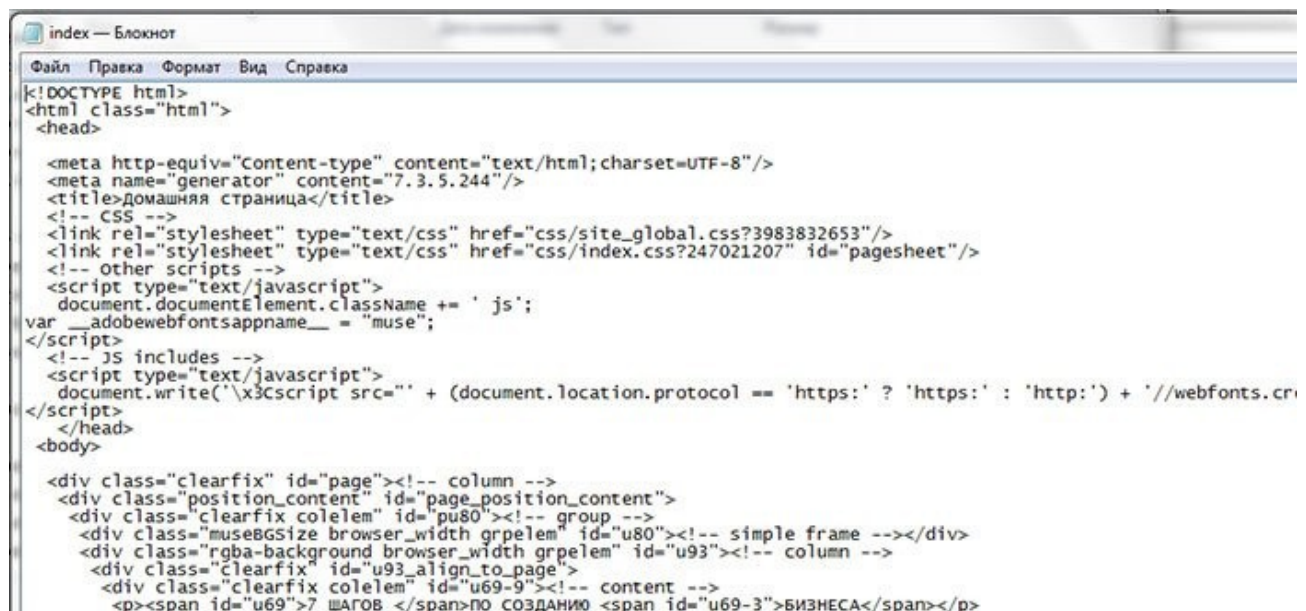


На свете существует просто огромное количество как платных, так и бесплатных текстовых редакторов, но статистика показывает, что более 70% программистов и веб-мастеров, верстают свои творения именно в Notepad.

С чем это связано, так и не понятно, наверное потому что он один из самых удобных, и простейших в управлении. Такое же дело обстоит и с FileZilla.

Часто приходится править, редактировать файлы форматов html, css, php и прочие. Если открыть такой файл в обычном текстовом документе, то там будет все сплошным одинаковым текстом, что для не профессионала сложно воспринимать и найти там нужный участок кода сложнее. Решением данной проблемы и является программа NotePad++ она подсвечивает код документа, что намного удобнее и в таком формате легче работать с документом.

Так выглядит код в обычном документе:



```
index — Блокнот
Файл Правка Формат Вид Справка
<!DOCTYPE html>
<html class="html">
<head>

<meta http-equiv="Content-type" content="text/html; charset=UTF-8"/>
<meta name="generator" content="7.3.5.244"/>
<title>домашняя страница</title>
<!-- CSS -->
<link rel="stylesheet" type="text/css" href="css/site_global.css?3983832653"/>
<link rel="stylesheet" type="text/css" href="css/index.css?247021207" id="pagesheet"/>
<!-- Other scripts -->
<script type="text/javascript">
document.documentElement.className += ' js';
var __adobewebfontsappname__ = "muse";
</script>
<!-- JS includes -->
<script type="text/javascript">
document.write('\x3Cscript src="' + (document.location.protocol == 'https:' ? 'https:' : 'http:') + '//webfonts.cri
</script>
</head>
<body>

<div class="clearfix" id="page"><!-- column -->
<div class="position_content" id="page_position_content">
<div class="clearfix colelem" id="pu80"><!-- group -->
<div class="museBGSize browser_width grpelem" id="u80"><!-- simple frame --></div>
<div class="rgba-background browser_width grpelem" id="u93"><!-- column -->
<div class="clearfix" id="u93_align_to_page">
<div class="clearfix colelem" id="u69-9"><!-- content -->
<p><span id="u69">? ШАГОВ </span>ПО СОЗДАНИЮ <span id="u69-3">БИЗНЕСА</span></p>
```

Так в NotePad++:

```
index.html
1 <!DOCTYPE html>
2 <html class="html">
3 <head>
4
5     <meta http-equiv="Content-type" content="text/html; charset=UTF-8"/>
6     <meta name="generator" content="7.3.5.244"/>
7     <title>Домашняя страница</title>
8     <!-- CSS -->
9     <link rel="stylesheet" type="text/css" href="css/site_global.css?3983832653"/>
10    <link rel="stylesheet" type="text/css" href="css/index.css?247021207" id="pagesheet"/>
11    <!-- Other scripts -->
12    <script type="text/javascript">
13        document.documentElement.className += ' js';
14        var __adobewebfontsappname__ = "muse";
15    </script>
16    <!-- JS includes -->
17    <script type="text/javascript">
18        document.write('\x3Cscript src="' + (document.location.protocol == 'https:' ? 'https:' : 'http:')
19    </script>
20 </head>
21 <body>
```

Плюсы программирования в Notepad++:

- Минималистичный интерфейс
- Широкий спектр работы с текстом (поиск/замена по регуляркам)
- Минимальные системные требования
- Простота в установке и использовании

Минусы прогания в Notepad++:

- Отсутствие дебагера (звоночек начинающим)

Лабораторная работа №3

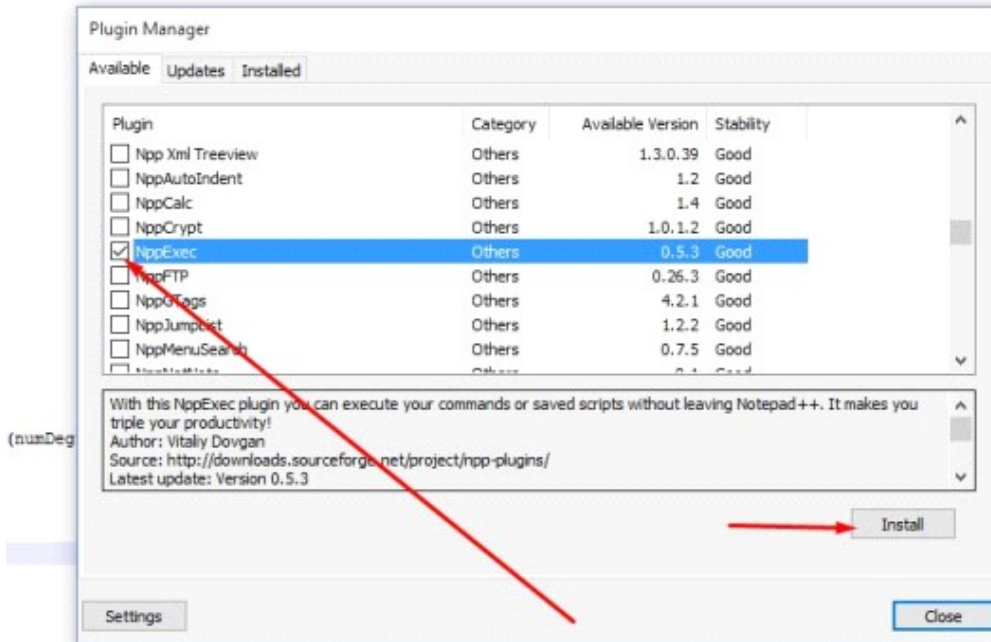
Процесс установки:

Программа бесплатная, скачать ее можете на оф.сайте www.notepad-plus-plus.org

1) Заходим: plugin



2) Далее находим NppExec: Ехес.jpg



3) Соглашаемся на перезагрузку Notepad++

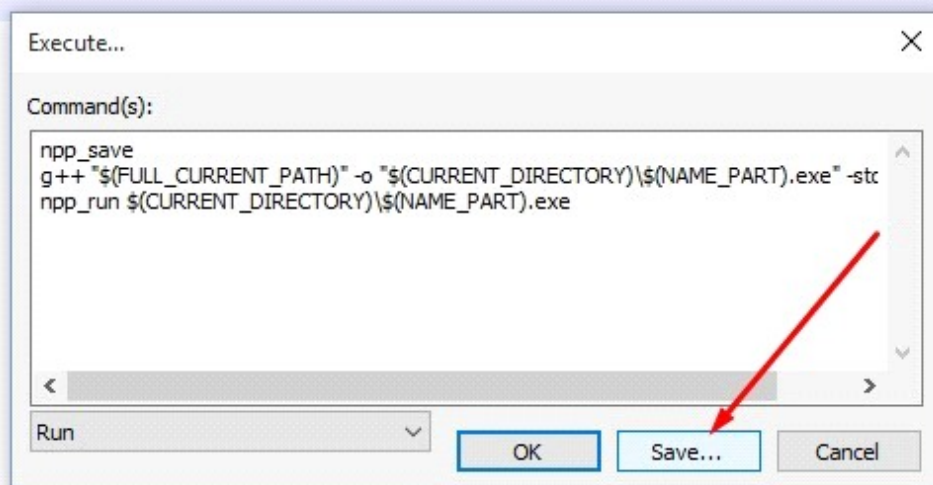
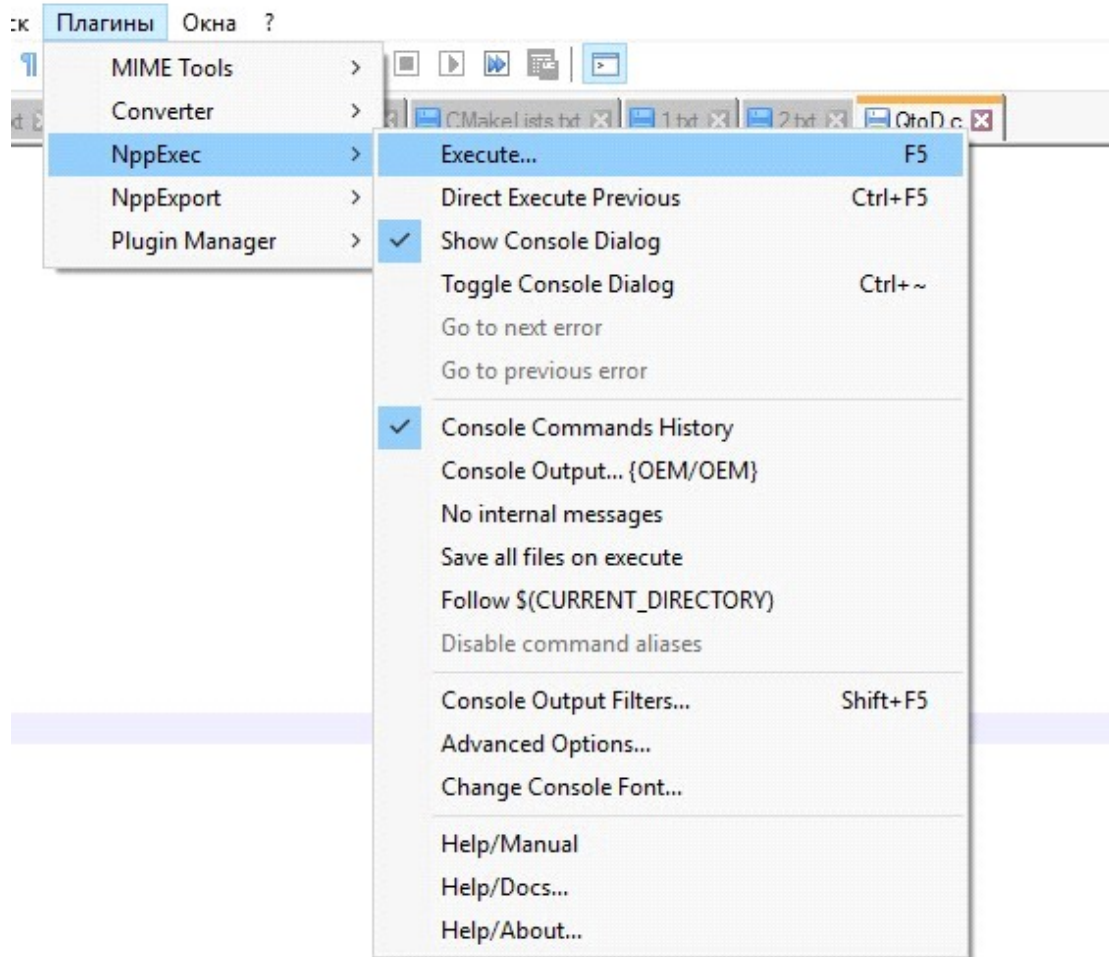
4) Далее необходимо прописать путь до компилятора в переменную среду Path (внизу инструкция)

5) Написать скрипт: npp_save

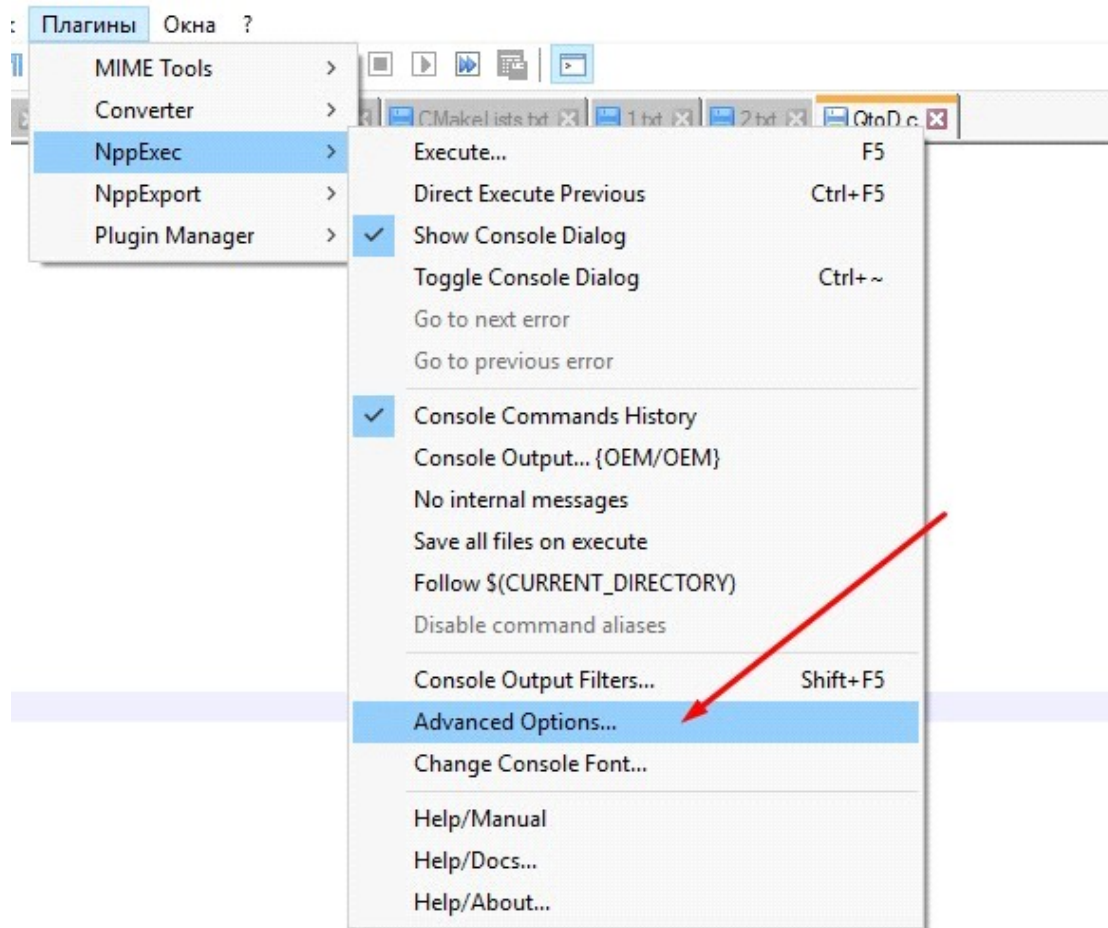
g++ «\$(FULL_CURRENT_PATH)» -o «\$(CURRENT_DIRECTORY)\\$

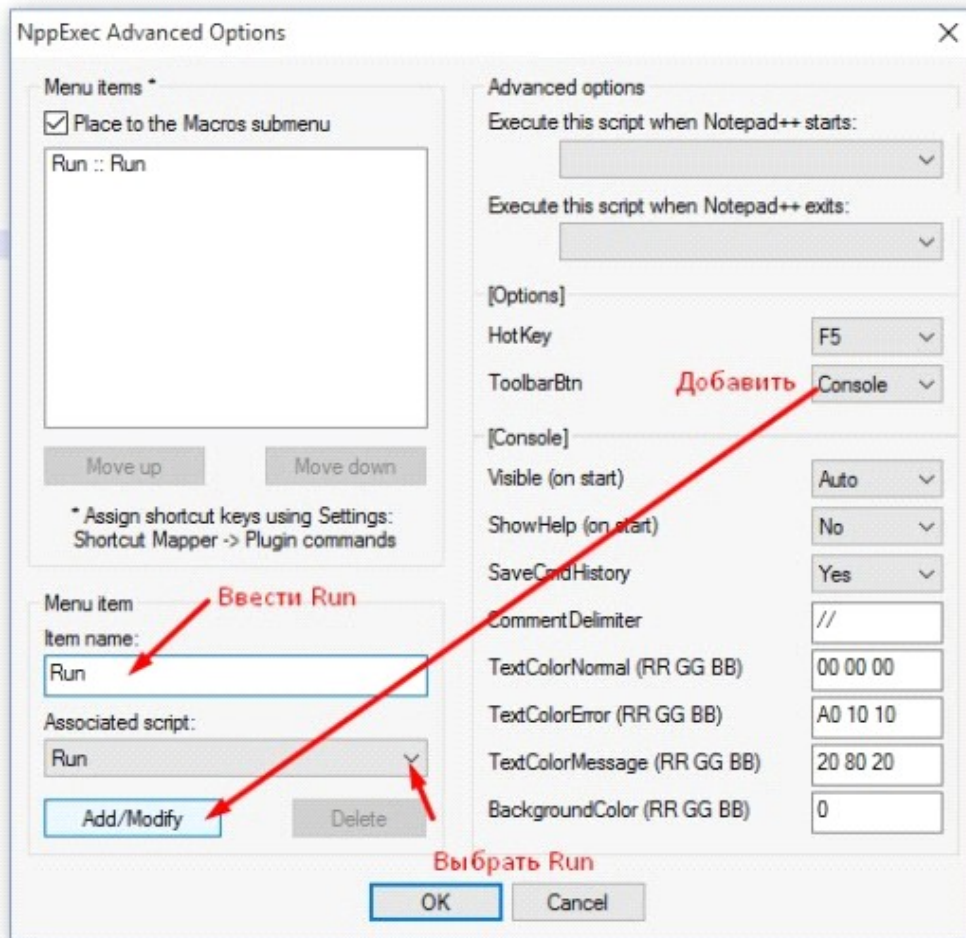
(NAME_PART).exe» -std=c++11

npp_run \$(CURRENT_DIRECTORY)\\$(NAME_PART).exe

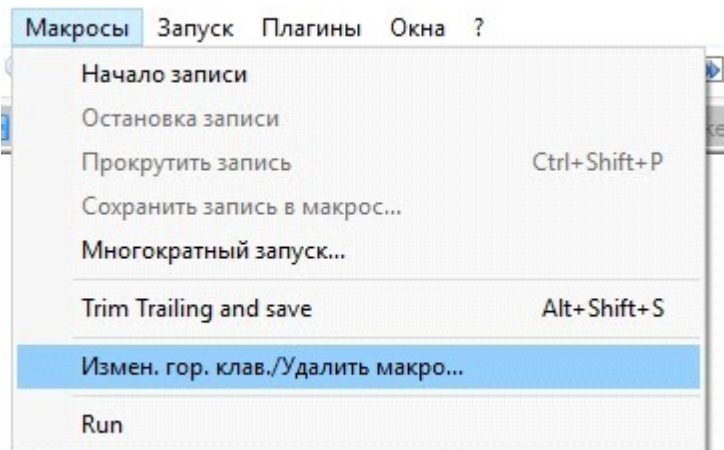


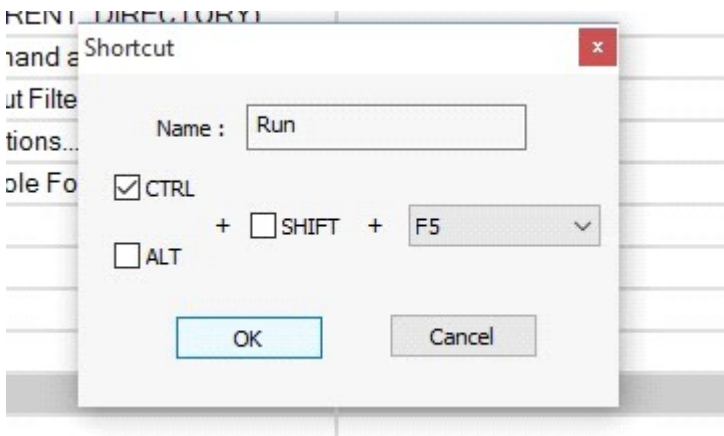
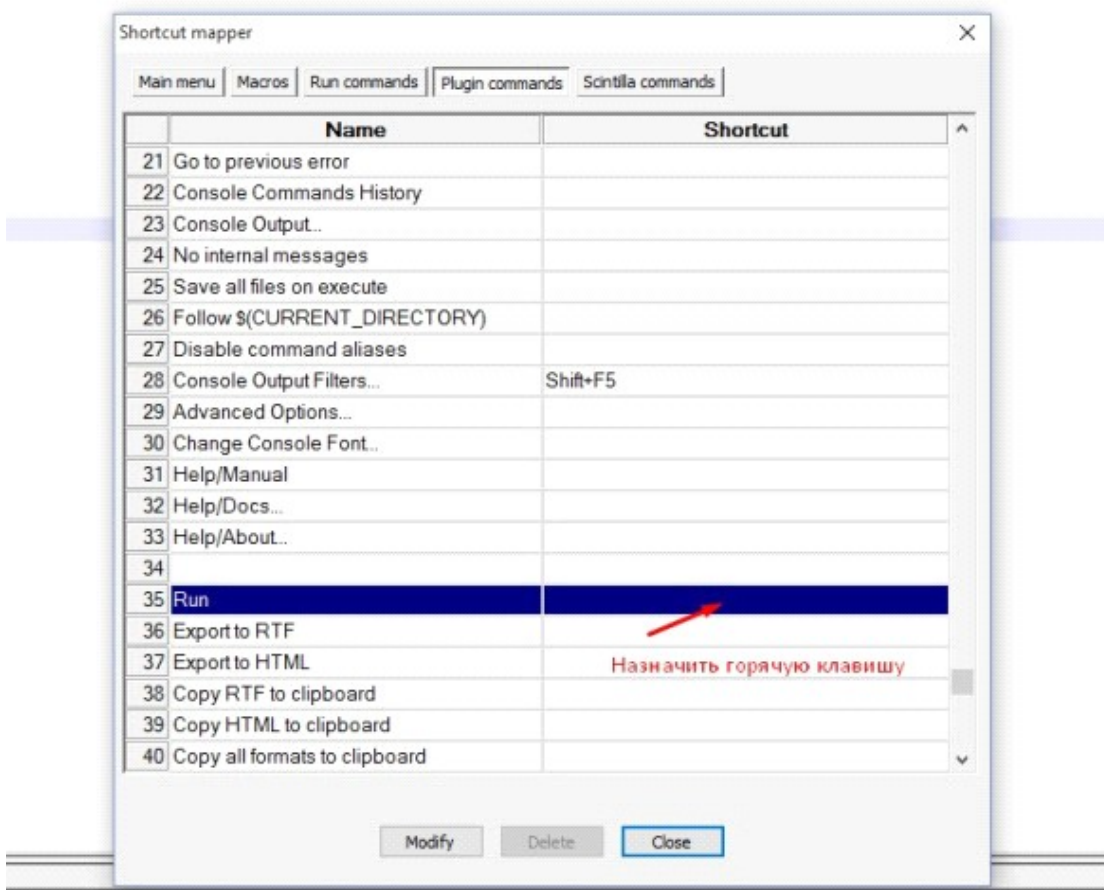
Добавить скрипт в общий пул





Добавить гор. клавишу для данного скрипта (Во вкладке *Plugin commands*)

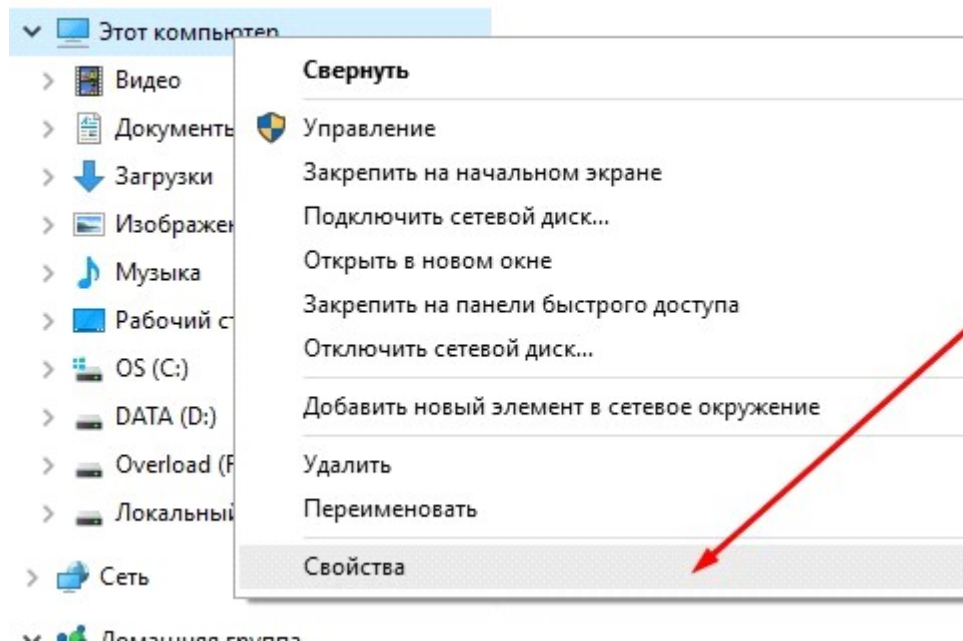




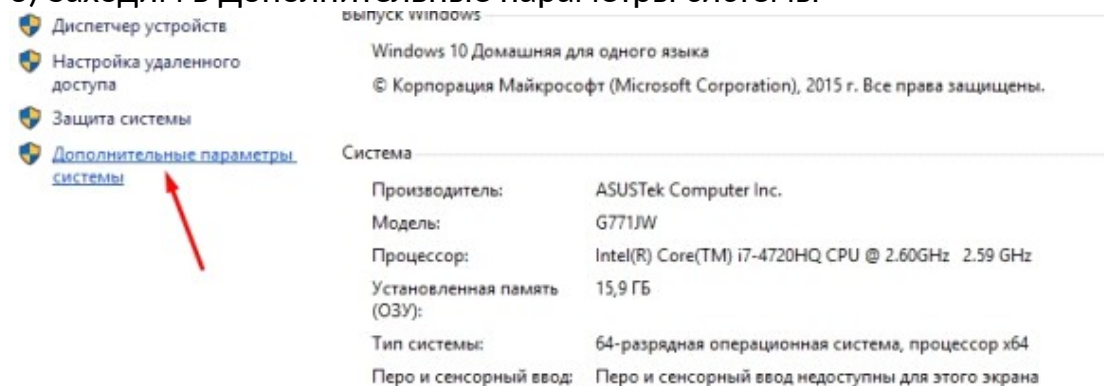
9) Все, Mission Complete, открываем любой *.c/*.cpp исходник, нажимаем Ctrl+F5 (или вашу горячую клавишу) и она благополучно Сохраняет, Компилирует, Запускает вашу прогу.

Как добавить в переменную среду:

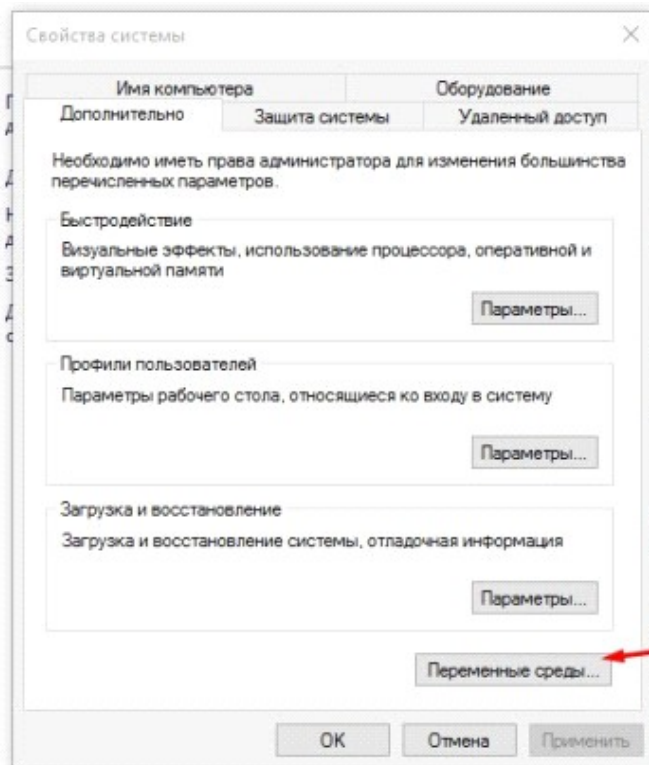
- 1) Открыть Проводник
- 2) Пкм «Мой компьютер»/»Этот компьютер»-> Свойства



3) Заходим в Дополнительные параметры системы



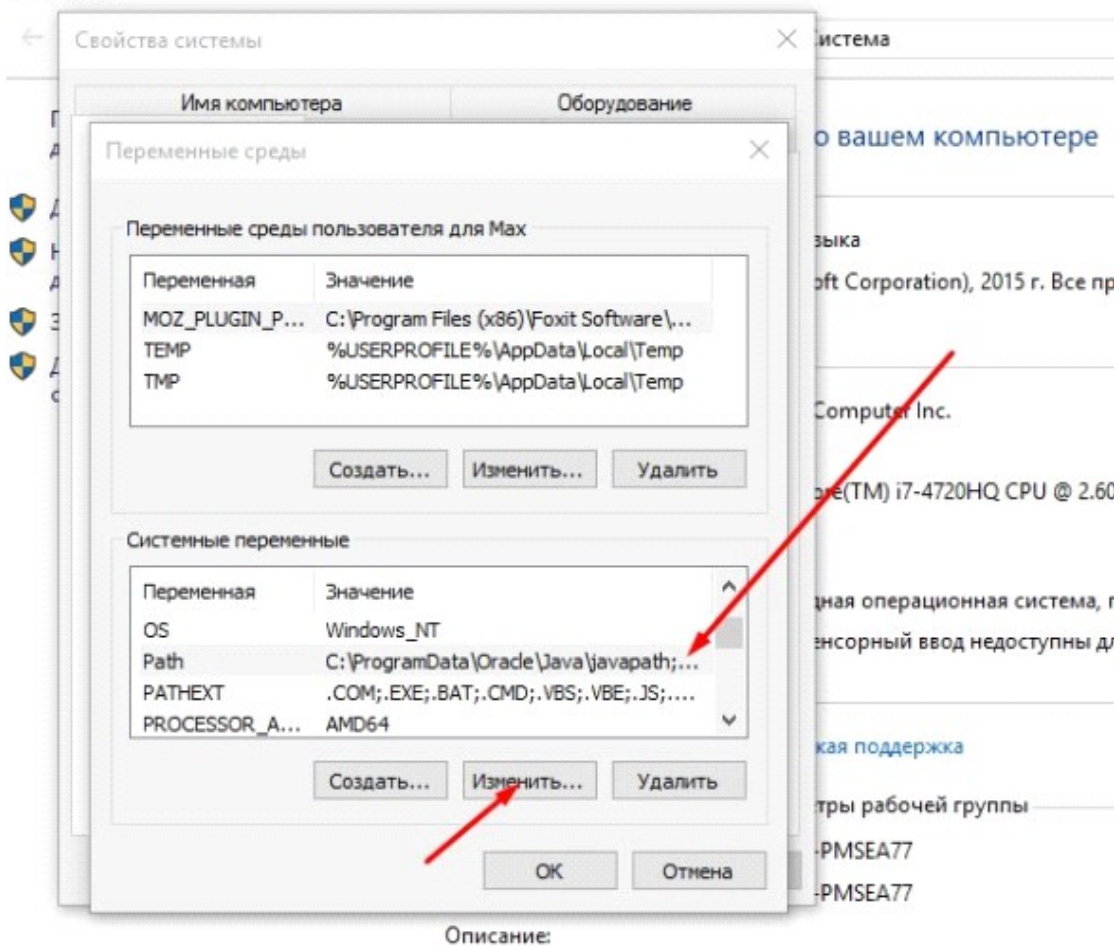
4) Находим Path



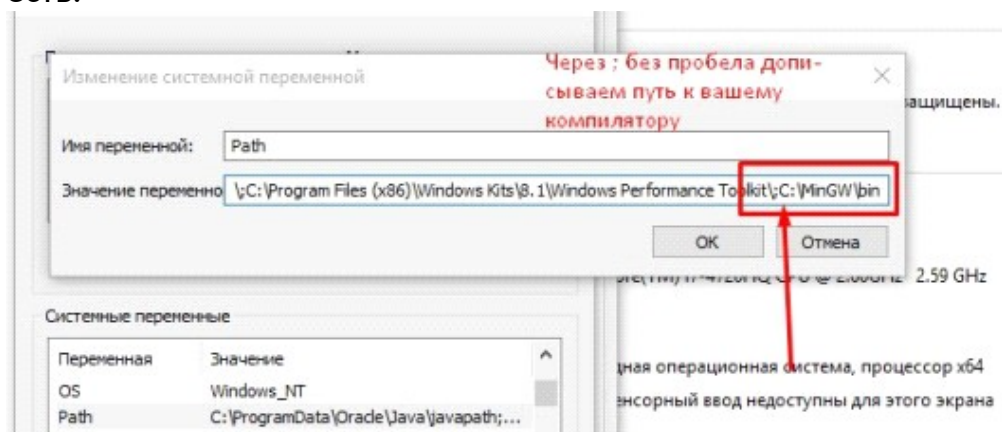
Описание:

Рабочая группа: WORKGROUP

Активация Windows



5) Дописываем путь к компилятору через «;» после того, что там уже есть.



№4 «Geany»

Geany — среда разработки программного обеспечения, написанная с использованием библиотеки GTK+. Доступна для следующих операционных систем: BSD, Linux, Mac OS X, Solaris и Windows. Geany распространяется согласно GNU General Public License.

Geany не включает в свой состав компилятор. Для создания исполняемого кода используется GNU Compiler Collection или, при необходимости, любой другой компилятор.

Функции:

1. Подсветка исходного кода с учётом синтаксиса используемого языка программирования (язык определяется автоматически по расширению файла);
2. Автозавершение слов;
3. Автоматическая подстановка закрывающих тегов HTML / XML. Автоподстановка стандартных и существующих в открытых файлах функций;
4. Простой менеджер проектов;
5. Поддержка плагинов;
6. Встроенный эмулятор терминала;
7. Поддержка большого количества кодировок;

8. Гибкий интерфейс;

9. Возможность использования и создания сниппетов. Для этого используется специальный файл `snippets.conf` в каталоге[4] `/home/user/.config/geany` позволяющий создавать свои сниппеты;

10. Возможность использования и создания шаблонов файлов. Шаблоны должны быть расположены в каталоге[4] `/home/user/.config/geany/templates/files`;

11. Отладка кода с помощью модуля (плагина) `GeanyGDB` (использует отладчик `GDB`);

12. Использование контекстной документации `man`, `Devhelp`; Можно использовать свои;

13. С версии 1.24 в дистрибутив под `windows` включены цветовые схемы редактора.

Создание консольных приложений в Geany

Кто-то называет `Geany` текстовым редактором для программистов, кто-то – интегрированной средой разработки (IDE). Но `Geany` “умеет” больше, чем обычный текстовый редактор, а по поводу классификации в качестве IDE лучше всего узнать мнение автора и ведущего разработчика этой программы – Энрико Трёгера (`Enrico Tröger`): “`Geany` – это компактная и простая среда разработки. Она была создана для того, чтобы предоставить программистам небольшую и быструю IDE, которая имеет зависимости всего лишь от нескольких сторонних пакетов. Другой целью было

обеспечение как можно большей независимости от конкретных рабочих сред (рабочих столов), таких как KDE или GNOME, – для Geany требуются только runtime-библиотеки GTK2”.

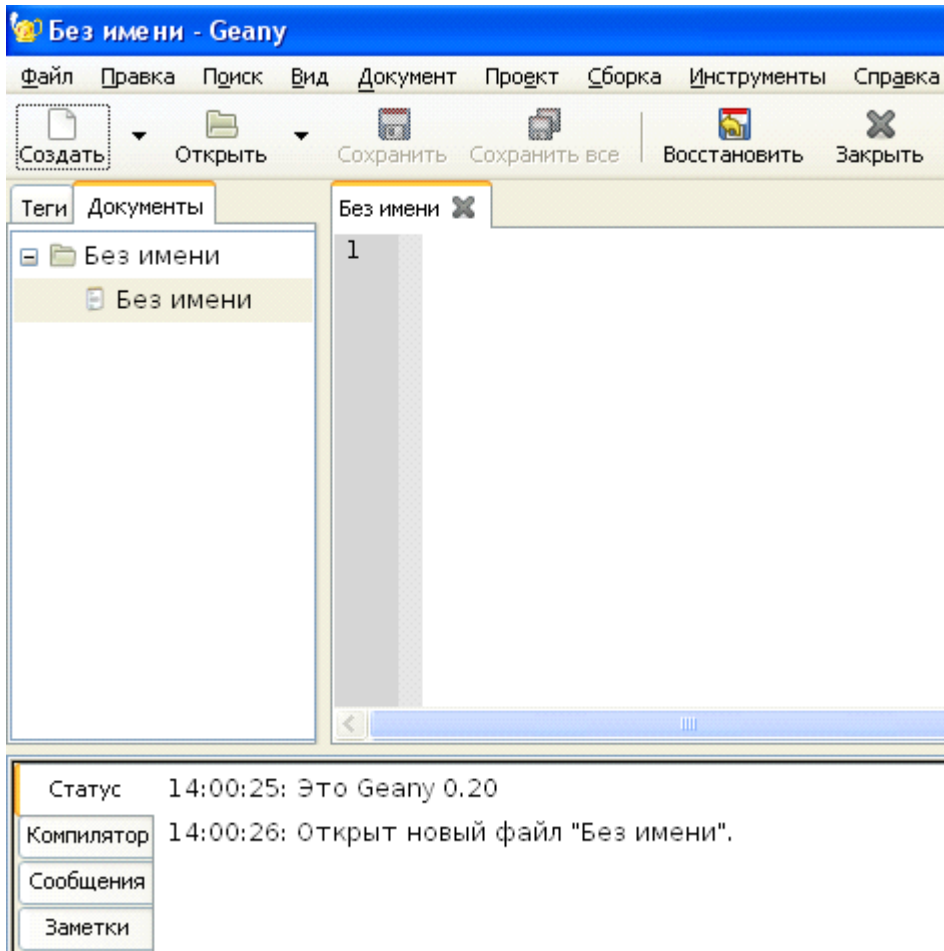
Geany распознаёт и выполняет подсветку синтаксиса для более чем 40 (!) языков программирования, разметки и скриптовых языков, таких как C/C++, Python, Java, PHP, HTML, Pascal, Perl, Basic и др. Кроме того, в ней поддерживается свёртка фрагментов кода, автозавершение символьных имён и языковых конструкций, автозакрывание тэгов XML и HTML, списки символов, здесь имеется подсистема сборки для компиляции и выполнения кода, а также простые средства управления проектами.

Лабораторная работа №4

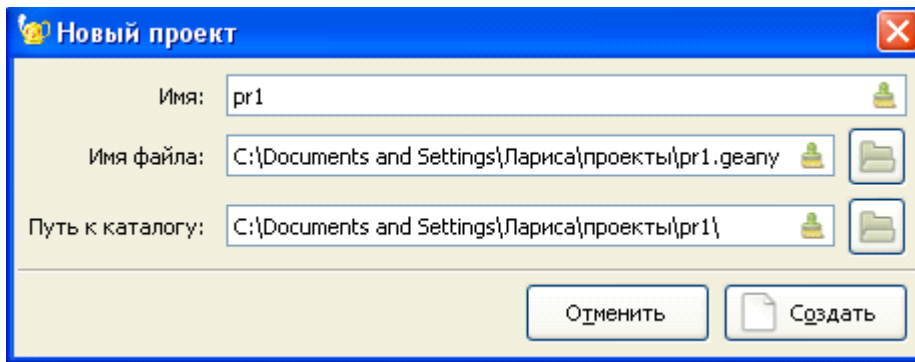
Скачиваем программу:

(<http://www.geany.org/Download/Releases>).

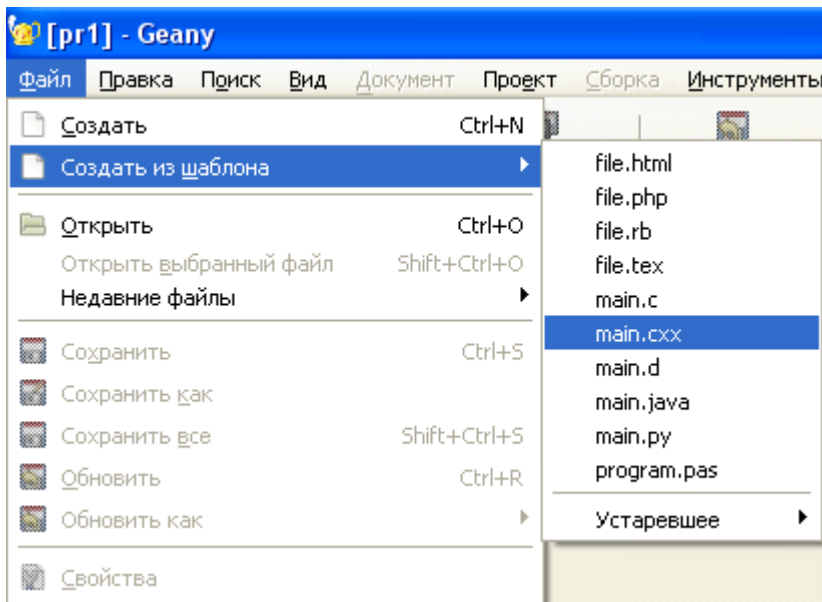
После установки и запуска на рабочем столе появляется окно Geany, показанное на рисунке 1.1



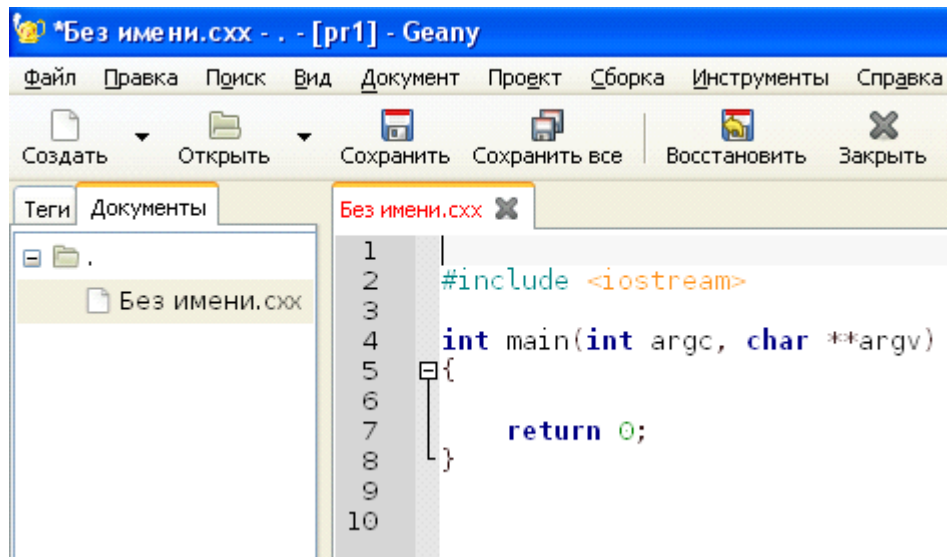
Создадим новый проект: выберем в меню пункт “Проект” и далее в открывшемся меню пункт “Новый”. После этого выводится диалоговое окно “Новый проект”. Имя проекту дадим pr1. По умолчанию Geany предлагает размещать файлы с исходными кодами в каталоге с именем “проекты”, расположенным в домашнем каталоге текущего пользователя. Можно изменить расположение проекта, отредактировав поля “Имя файла” и “Путь к каталогу” или нажав кнопки справа от этих полей и выбрав требуемый каталог. Теперь можно щёлкнуть по кнопке “Создать”.



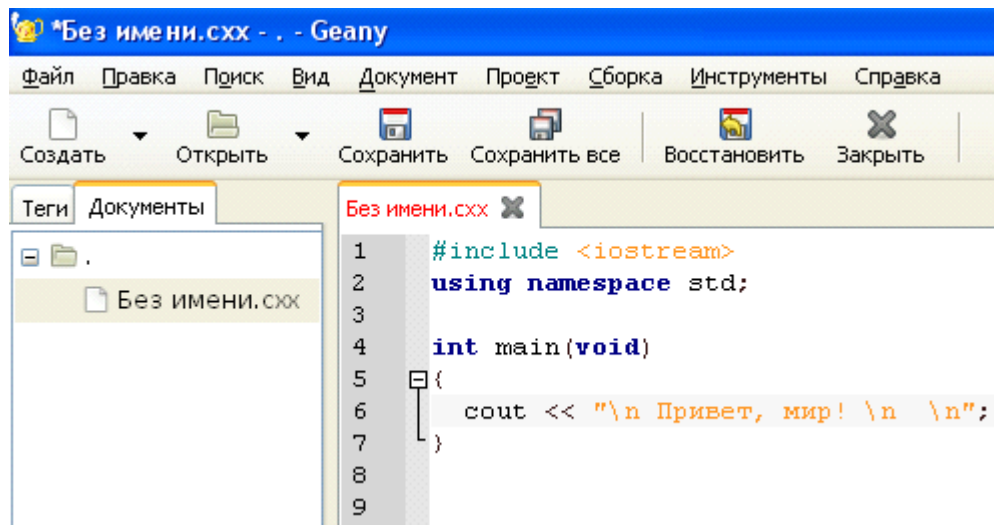
Теперь создадим файл с программой. Это можно сделать следующими способами. Если выбрать в меню “Файл” пункт “Создать”, то создается “Безымянный” файл, тип его определён как “Никакой”, и вводимый текст не подсвечивается. Это можно исправить, выбрав в меню “Документ” пункт “Установить тип файла” и далее в подменю “Языки программирования” и “Исходный файл C++”. Другой способ – сразу воспользоваться шаблоном файла C++



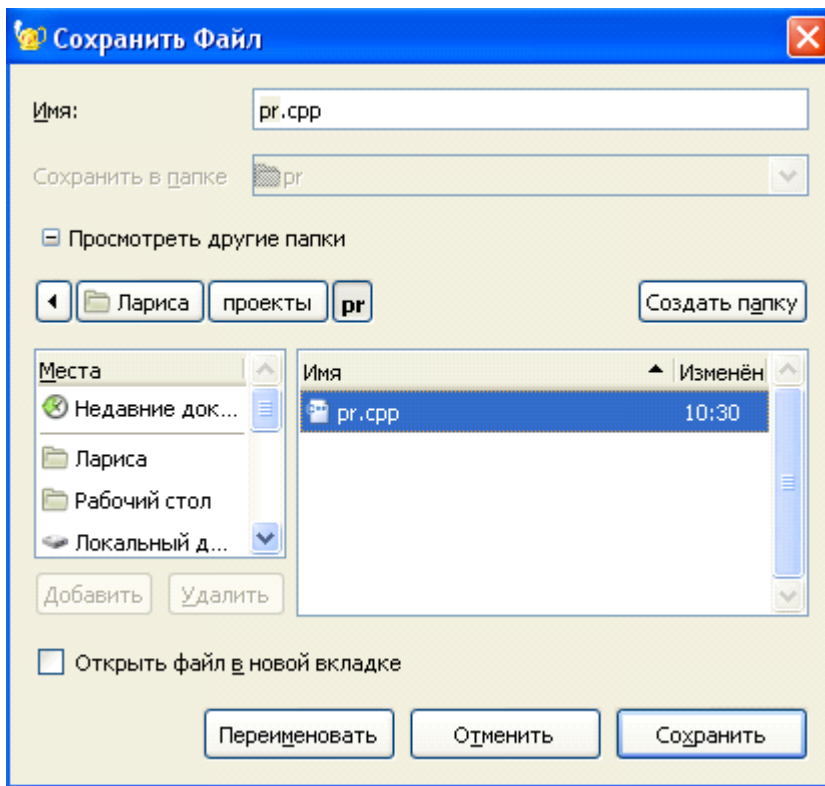
Шаблон файла программы для C++:



Теперь можно приступить к вводу исходного кода программы. Рассмотрим программу, которая выводит на экран сообщение «Привет, мир!».




Сохраним программу. Как вы помните, мы уже определили каталог проекта, в котором будут храниться все наши файлы. Осталось только выполнить команду сохранения и дать имя всё ещё “безымянному” файлу. Для этого воспользуемся кнопкой сохранения текущего файла в панели инструментов (значок дискеты) или обратимся к меню “Файл” -> “Сохранить”.



Теперь мы можем создать выполняемую программу и проверить её функционирование. Для этого воспользуемся

пунктом меню “Сборка”. В выпадающем подменю первые два пункта – “Скомпилировать” и “Сборка”. В результате выполнения первой команды мы получим из файла исходного кода объектный файл, который, хотя и является двоичным кодом, всё же не готов к самостоятельному выполнению. Подобная команда необходима в том случае, когда проект состоит из многих исходных файлов, зачастую написанных на различных языках программирования. Для нашего проекта больше подходит пункт “Сборка”, позволяющий сразу получить требуемый результат. Сборку можно выполнить и одним нажатием функциональной клавиши F9 (она указана справа от данного пункта подменю). После выполнения этой операции (если, конечно, в исходном коде не было ошибок и опечаток) нижняя панель автоматически переключается на вкладку “Компилятор” и сообщает об успешном завершении процесса сборки программы.

Остаётся лишь посмотреть, как работает наша программа. Запустить её можно различными способами: из меню “Сборка” –> “Выполнить”, функциональной клавишей F5 или щелчком по значку шестерёнки с треугольником внутри на панели инструментов.



```
Привет, мир!  
Для продолжения нажмите любую клавишу . . .
```

Результат выполнения программы

№5 «Регулярные выражения»

Регулярные выражения — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Возможности:

Набор утилит (включая редактор sed и фильтр grep), поставляемых в дистрибутивах UNIX, одним из первых способствовал популяризации регулярных выражений для обработки текстов. Многие современные языки программирования имеют встроенную поддержку регулярных выражений. Среди них ActionScript, Perl, Java[1], PHP, JavaScript, языки платформы .NET Framework[2], Python, Tcl, Ruby, Lua, Gambas, C++ (стандарт 2011 года), Delphi, D, Нахе и другие.

Регулярные выражения используются некоторыми текстовыми редакторами и утилитами для поиска и подстановки текста. Например, при помощи регулярных выражений можно задать шаблоны, позволяющие:

- 1. найти все последовательности символов «кот» в любом контексте, как то: «кот», «котлета», «терракотовый»;

- 2. найти отдельно стоящее слово «кот» и заменить его на «кошка»;
- 3. найти слово «кот», которому предшествует слово «персидский» или «чеширский»;
- 4. убрать из текста все предложения, в которых упоминается слово кот или кошка.

Регулярные выражения позволяют задавать и гораздо более сложные шаблоны поиска или замены.

Результатом работы с регулярным выражением может быть:

1. проверка наличия искомого образца в заданном тексте;
2. определение подстроки текста, которая сопоставляется образцу;
3. определение групп символов, соответствующих отдельным частям образца.

Если регулярное выражение используется для замены текста, то результатом работы будет новая текстовая строка, представляющая из себя исходный текст, из которого удалены найденные подстроки (сопоставленные образцу), а вместо них подставлены строки замены (возможно, модифицированные запомненными при разборе группами символов из исходного текста). Частным случаем модификации текста является удаление всех вхождений найденного образца — для чего строка замены указывается пустой.

Лабораторная работа №5

Цель работы: Изучить приемы работы с регулярными выражениями в PHP.

Очень часто регулярные выражения используются для того, чтобы проверить, является ли данная строка строкой в необходимом формате. Например следующий regex предназначен для проверки того, что строка содержит корректный e-mail адрес: `/^\w+([\.\w]+)*\w@\w([\.\w]+)\w+)*\.\w{2,3}$/`

Регулярные выражения пришли к нам из Unix и Perl. В PHP существует два различных механизма для обработки регулярных выражений: POSIX-совместимые и Perl-совместимые. Их синтаксис во многом похож, однако Perl-совместимые регулярные выражения более мощные и, к тому же, работают намного быстрее (в некоторых случаях до 10 раз быстрее). Поэтому здесь мы будем вести речь только о Perl-совместимых регулярных выражениях.

Кстати, необходимо заметить, что полное описание синтаксиса регулярных выражений, имеющееся в PHP Manual, занимает более 50 килобайт и, естественно, здесь мы не будем рассматривать весь синтаксис. Нам необходимы только основы, которые помогут вам понять, как именно пишутся регулярные выражения.

Сутью механизма регулярных выражений является то, что они позволяют задать шаблон для нечеткого поиска по тексту. Например, если перед вами стоит задача найти в тексте

определенное слово, то с этой задачей хорошо справляются и обычные функции работы со строками. Однако если вам нужно найти "то, не знаю что", о чем вы можете сказать только то, как приблизительно это должно выглядеть - то здесь без регулярных выражений просто не обойтись. Например, вам необходимо найти в тексте информацию, про которую вам известно только то, что это "3 или 4 цифры после которых через пробел идет 5 заглавных латинских букв", то вы сможете сделать это очень просто, воспользовавшись следующим регулярным выражением: $\wedge d\{3,4\} \backslash s[A-Z]\{5\} /$

Синтаксис регулярных выражений

Регулярные выражения, как уже было сказано выше, представляют собой строку. Строка всегда начинается с символа разделителя, за которым следует непосредственно регулярное выражение, затем еще один символ разделителя и потом необязательный список модификаторов. В качестве символа разделителя обычно используется слэш ('/'). Таким образом в следующем регулярном выражении: $\wedge d\{3\} - \backslash d\{2\} / m$, символ '/' является разделителем, строка ' $\backslash d\{3\} - \backslash d\{2\}$ ' - непосредственно регулярным выражением, а символ 'm', расположенный после второго разделителя - это модификатор.

Основой синтаксиса регулярных выражений является тот факт, что некоторые символы, встречающиеся в строке рассматриваются не как обычные символы, а как имеющие специальное значение (т.н. метасимволы). Именно это решение позволяет работать всему механизму регулярных

выражений. Каждый метасимвол имеет свою собственную роль в синтаксисе регулярных выражений. Далее мы рассмотрим все эти метасимволы.

Одним из самых важных метасимволов является символ обратного слэша ('\'). Если в строке встречается этот символ, то парсер рассматривает символ, непосредственно следующий за ним двояко:

- **а)** если следующий символ в обычном режиме имеет какое-либо специальное значение, то он теряет это свое специальное значение и рассматривается как обычный символ. Это совершенно необходимо для того, чтобы иметь возможность вставлять в строку специальные символы, как обычные. Например метасимвол '.', в обычном режиме означает "любой единичный символ", а '\.' означает просто точку. Также можно лишить специального значения и сам этот символ: '\\'.
- **б)** если следующий символ в обычном режиме не имеет никакого специального значения, то он может получить такое значение, будучи соединенным с символом '\'. К примеру символ 'd' в обычном режиме воспринимается просто как буква, однако, будучи соединенной с обратным слэшем ('\d') становится метасимволом, означающим "любая цифра".

Существует множество символов, которые образуют метасимволы в паре с обратным слэшем. Как правило подобные пары используются для того, чтобы показать, что на

этом месте в строке должен находиться символ, с кодом, который не имеет соответствующего ему изображения или же символ, принадлежащий какой-то определенной группе символов. Ниже приведены некоторые наиболее употребительные:

<u>Метасимвол</u>	<u>Значение</u>
-------------------	-----------------

Метасимволы для задания символов, не имеющих изображения

\n	Символ перевода строки(код0x0A)
\r	Символ возврата каретки (код0x0D)
\t	Символ табуляции (код 0x09)

Метасимволы для задания групп символов

\d	Цифра (0-9)
\D	Не цифра
\s	Пустой
\S	Непустой символ
\w	"Словесный"
\W	Все, кроме символов, определяемых метасимволом \w

Несколько простейших примеров.

<u>Regexp</u>	<u>Комментарии</u>
<code>\d\d\d/</code>	Любое трехзначное число
<code>\w\s\d\d/</code>	Буква, пробел
<code>\d and \d/</code>	Любая из следующих строк: '1 and 2', '9 and 5', '3 and 4'.

Синтаксис регулярных выражений имеет средства для определения собственных подмножеств символов. Например вам может понадобиться задать условие, что в этом месте строки должна находиться шестнадцатиричная цифра или еще что-то подобное. Для описания таких подмножеств применяются символы квадратных скобок '[]'. Квадратные скобки, встреченные внутри регулярного выражения считаются одним символом, который может принимать значения, перечисленные внутри этих скобок.

Есть небольшая тонкость в том, как работают метасимволы внутри квадратных скобок. Дело в том, что в синтаксисе регулярных выражений существует еще множество метасимволов, но практически все они работают только вне секций описаний подмножеств. Единственные метасимволы, которые работают внутри этих секций это:

Обратный слэш ('\'). Т.е. все метасимволы из приведенной ранее таблицы будут работать.

Минус ('-'). Используется для задания набора символов из одного промежутка (например все цифры могут быть заданы как '0-9')

Символ '^'. Если этот символ стоит первым в секции задания подмножества символов (и только в этом случае!) он будет рассматриваться как символ отрицания. Т.о. можно задать все символы, которые не описаны в данной секции.

Допустим, у нас есть текст:

12 aaa bbb

aaa 27 ccc

aaa aaa 45

И регулярное выражение для поиска чисел в этом тексте: `/\d\d/m` (не обращайтесь пока внимания на модификатор). Поиск по этому регулярному выражению вернет нам 3 значения: '12', '27', '45'. Теперь ограничим поиск, указав, где именно внутри строки должен располагаться текст: `/^\d\d/m`. Здесь результат будет только один - '12', потому что только это число располагается в начале строки. Аналогично, регулярное выражение `/\d\d$/m` вернет результат '45'.

Символ точки '.'. Этот метасимвол указывает, что на данном месте в строке может находиться любой символ (за исключением символа перевода строки). Очень удобно использовать его, если вам нужно "пропустить" какую-нибудь

букву в слове при проверке. Например регулярное выражение `/.bc/` найдет в тексте и `'abc'` и `'Abc'` и `'Zbc'` и `'5bc'`.

Символ вертикальной черты `'|'`. Используется для задания списка альтернатив. Например регулярное выражение:

`/(красное|зеленое) яблоко/`

Найдет в тексте все словосочетания `'красное яблоко'` и `'зеленое яблоко'`.

Допустим, нам необходимо проверить, является ли строка семизначным телефонным номером с указанием кода города и получить из нее код города и номер телефона:

`^((\d{3,5})\s+(\d{3}-\d{2}-\d{2}))/`

Давайте рассмотрим это регулярное выражение подробнее:

- 1) Первая круглая скобка здесь теряет свое специальное значение и будет рассматриваться как обычный символ: `\(`
- 2) Далее идет регулярное выражение в скобках (проверка кода города): `(\d{3,5})`
- 3) После этого идет закрывающая круглая скобка, которая также лишена своего специального значения из-за символа обратного слэша, стоящего перед ней: `\)`
- 4) Затем идет пропуск пустого места: `\s+`

5) И еще одно регулярное выражение в скобках, которое проверяет номер телефона: `(\d{3}-\d{2}-\d{2})`

Как видите, здесь есть 3 регулярных выражения - основное и два внутренних. При этом основное выражение позволяет нам проверить, имеет ли строка необходимый нам формат, а два внутренних - получить соответственно код города и номер телефона.

Посмотрим, как работает это регулярное выражение. Пусть у нас есть строка: "My phone is (095) 123-45-67". Результатами поиска будут 3 строки: '(095) 123-45-67', '095' и '123-45-67'.

Задания к лабораторной работе:

Найти в документации описание следующих функций работы с регулярными выражениями:

`preg_match()`

`preg_match_all()`

`preg_replace()`

`preg_replace_callback()`

`preg_split()`

`preg_quote()`

`preg_grep()`

`preg_quote()`

Реализовать примеры их использования.

Составьте регулярные выражения для маскирования тегов HTML (20-30 на выбор).

Составьте регулярное выражение для проверки значения переменной:

-Быть целым числом.

-Быть вещественным числом.

-Быть идентификатором.

-Быть правильным телефонным номером (например 37-81-40).

-Быть правильным телефонным номером с кодом города (например (231) 5-94-00).

-Быть не числом.

-Не содержать цифр.

-Не содержать букв.

Построить регулярное выражение, возвращающее значение параметров тегов html-документа, содержащих URL.

№6 «Спецсимволы: допетровская кириллица»

Итак, представим себе девятый век нашей эры, когда солунские братья Константин, позже принявший монашество под именем Кирилла, и Мефодий работали над богоугодным делом — составляли азбуку для диких славянских племён, не имевших собственной грамоты, и при помощи новых диковинных закорючек писали перевод Священного Писания. Окончив труд, длившийся месяцы, а то и годы, братья пошли в Моравию с миссией просвещения. После смерти Константина-Кирилла и Мефодия, ученики равноапостольных братьев принесли книги своих учителей в соседскую Болгарию, откуда свет Евангелия пришёл и на территорию Древней Руси. Как это ни странно может показаться, первой азбукой славян была глаголица. Существует множество разных версий того, на что опирались солунские братья, когда создавали данный алфавит, однако каждый может убедиться, что, удобство написания букв явно не была основополагающей идеей, движимой равноапостольными святыми. Собственно, именно это и сыграло злую шутку с новым алфавитом. Мы не знаем, кто точно решил совместить уже сформированный на то время греческий алфавит с модифицированными буквами глаголицы, означающими звуки, которые отсутствовали в греческом (преимущественно шипящие), но результатом данной трансформации мы пользуемся и сейчас, пускай и в очень искажённом виде.

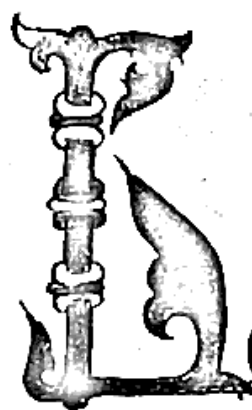
6.1. Устав

Устав — это, собственно, и есть тот самый греческий алфавит с добавленными буквами.

Как мы можем видеть буквы «ша» и «шта» (впоследствии ставшая нашим «ща») выглядят также, как и в глаголице. *Для сравнения можно посмотреть, как в то время писали греки:*

ѲПРОТЪ ХУГЕНИ

ЕКЪ МАТ



ІВЛОСГЕНЕСЕ

ДѡСИУХУ ВУДАД

УУАВРААМЪ АВРА

АМЪ ЕГЕНННСЕ

ТОИІСААКЪ ІСА

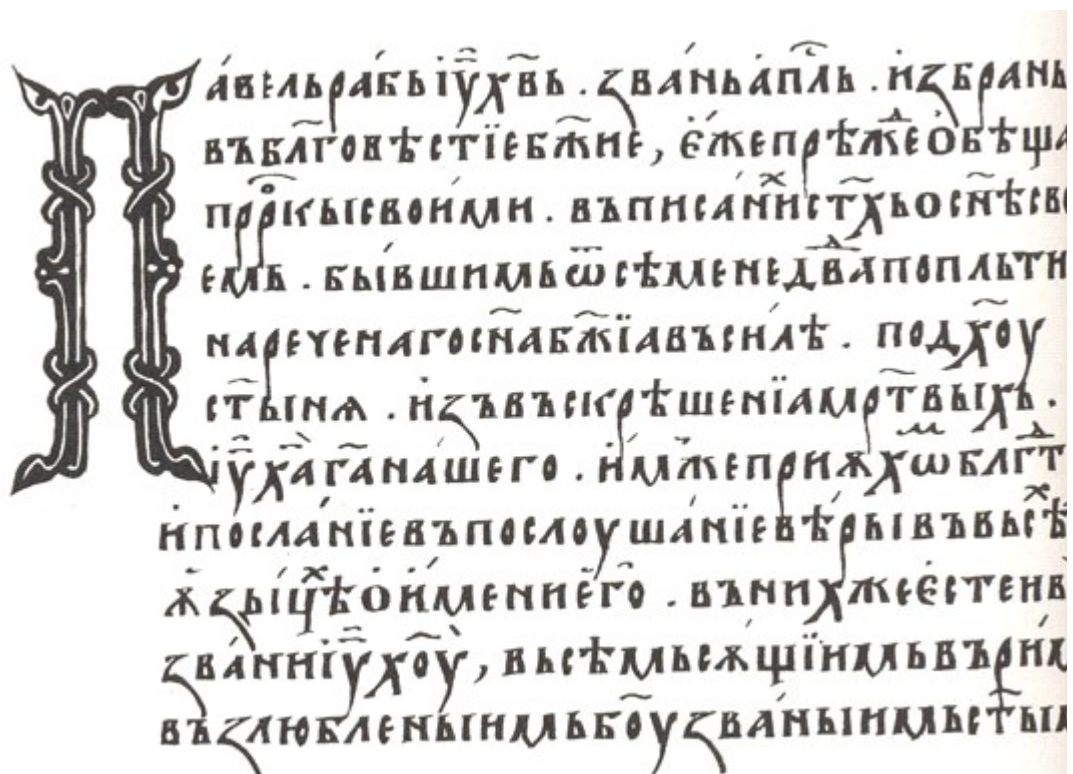
АКДѢ ЕГЕНННСЕ

ТОИІАКѡВЪ ІАКЪ

ДѢ ЕГЕНННСЕ

Из рисунка видно, что кириллица представляла собой модифицированный греческий алфавит для записи речи славян. Фактически, если бы тогда появились бы компьютеры и некое подобие таблицы шрифтов и Юникода, я думаю, кириллицу не стали бы выделять отдельно, а просто добавили символов в греческий. Но ни компьютеров, ни Юникода тогда не было, и кириллица отправляется в вольное плавание самостоятельно.

6.2.Полуустав



По

луустав возник уже в XIV веке. Фактически, это была, как бы мы сейчас сказали, курсивная версия устава. Надо отметить,

что славяне на тот период времени очень чётко следили за всеми модными тенденциями, которые проявлялись в греческом. В полууставе появились сокращения — титла, знаки ударения, придыхания, первое время, отсутствовавшие в славянском уставе, но присутствовавшие в греческом. К XVII веку сложилась определённая церковная орфография, ориентированная на написание греческих слов согласно оригиналу, так как все греческие буквы по наследству перешли кириллице.

6.3.Вязь



Н

о при всём желании копировать греческий оригинал были и существенные отличия. Одним из таких отличий была вязь. Надо отметить, что и в греческом языке она тоже была, но фактически, она умерла, практически в зародыше. Славянская же на территории Руси получила огромное развитие.



Арабская вязь

ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ

ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ

ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ

ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ ᲛᲗᲚᲘᲗᲚ

Грузинская вязь

ᲙᲗᲚᲘᲗᲚ ᲙᲗᲚᲘᲗᲚ

Вязанные руны с о. Валаам

ᲙᲗᲚᲘᲗᲚ ᲙᲗᲚᲘᲗᲚ

Коптская вязь

ᲙᲗᲚᲘᲗᲚ ᲙᲗᲚᲘᲗᲚ

Зачатки византийской вязи

ᲙᲗᲚᲘᲗᲚ ᲙᲗᲚᲘᲗᲚ

Глаголическая вязь

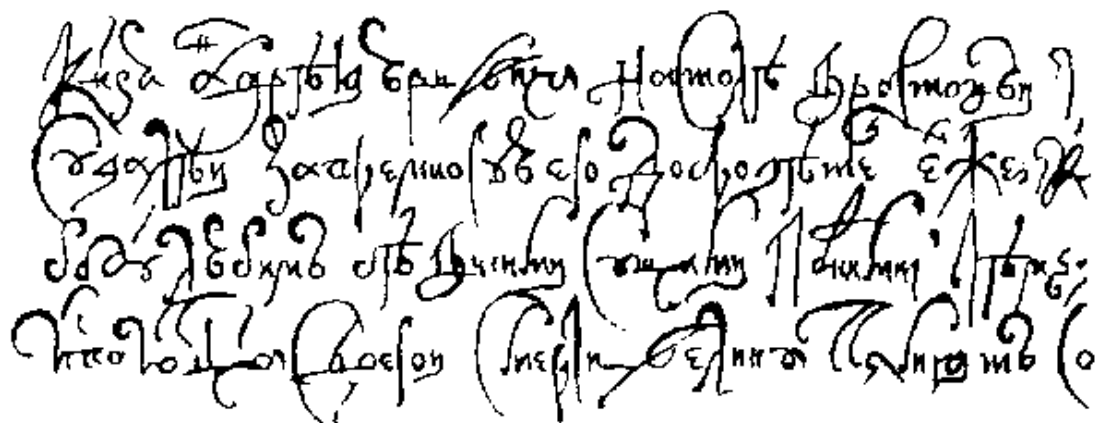
На представленной выше картинке мы можем видеть многие виды вязи: от арабской до глаголической (да, несмотря на то, что глаголический алфавит практически везде был заменён кириллическим, в Хорватии он продолжал жить своей жизнью ещё очень долго). Греческая вязь похожа на зародыш того

обилия красок, которые можно будет впоследствии видеть в кириллической:



Вязь на Руси использовалась для украшения заголовков книг, колоколов, икон.

6.4.Скоропись



Скоропись — это фактически тот вид письма, из которого можно было бы создать весьма отличающуюся от сегодняшней кириллицу. Разнообразие буквенных форм, различные лигатуры, плетёнка подобная арабской — всё это могло служить тому, что из кириллицы того периода времени при грамотной обработке можно было бы создать очень самобытное и красивое письмо. Единственное, что этому помешало — Пётр I. Именно он сделал революцию в кириллице и одел её в заморский шрифт, что не было губительным, конечно, но перевернуло всю эволюционную составляющую нашего письма. И если бороды боярам нужно было рубить, то так относиться к 600-летнему развитию кириллицы на Руси было, по-моему, большой ошибкой. Сейчас даже известные шрифтовики и каллиграфы признают, что такой революционный подход не стоил свеч. К латинской графике мы так и не пришли, но проблемы современной кириллицы не решены. Например, одна из главных проблем — 75% одинаковых строчных и прописных символов, что определённым образом влияет и на чтение, и на общий вид слов и текста, а в последнем современная кириллица проигрывает латинице, где этот недостаток устранён как раз эволюционным ходом развития письма. Остаётся только пожалеть, что, используя современную кириллицу, мы сейчас пишем в определённой мере суррогатом.

Лабораторная работа №6

Вопрос № 1. Дайте определение берестяной грамоте как историческому источнику.

Вопрос № 2. Где и когда была обнаружена первая берестяная грамота?

Вопрос № 3. Что из себя представляла берестяная грамота? Как она создавалась?

Вопрос № 4. Назовите формы берестяных грамот.

Вопрос № 5. Берестяное письмо. О чем оно свидетельствует? О чем писали на бересте?

Вопрос № 6. «Велесова книга». Что она из себя представляла?

Вопрос № 7. С кем из проповедников связана история славянской письменности? Что Вы о них знаете?

Вопрос № 8. Каково место берестяных грамот в истории документа?

Вывод: в результате проделанной работы мы закрепили необходимый минимум теоретических знаний в области отечественной и всеобщей истории (ПК-1), источниковедения. Также были закреплены наши знаниевые компетенции, сформированные в рамках дисциплины «Источниковедение», и освоена элементарная деятельностная компетенция: понимать, критически анализировать и использовать полученную историческую информацию (ПК-6).

№7 «Макросы в Sublime Text»

Sublime Text — проприетарный текстовый редактор. Поддерживает плагины на языке программирования Python.

Что такое macros?

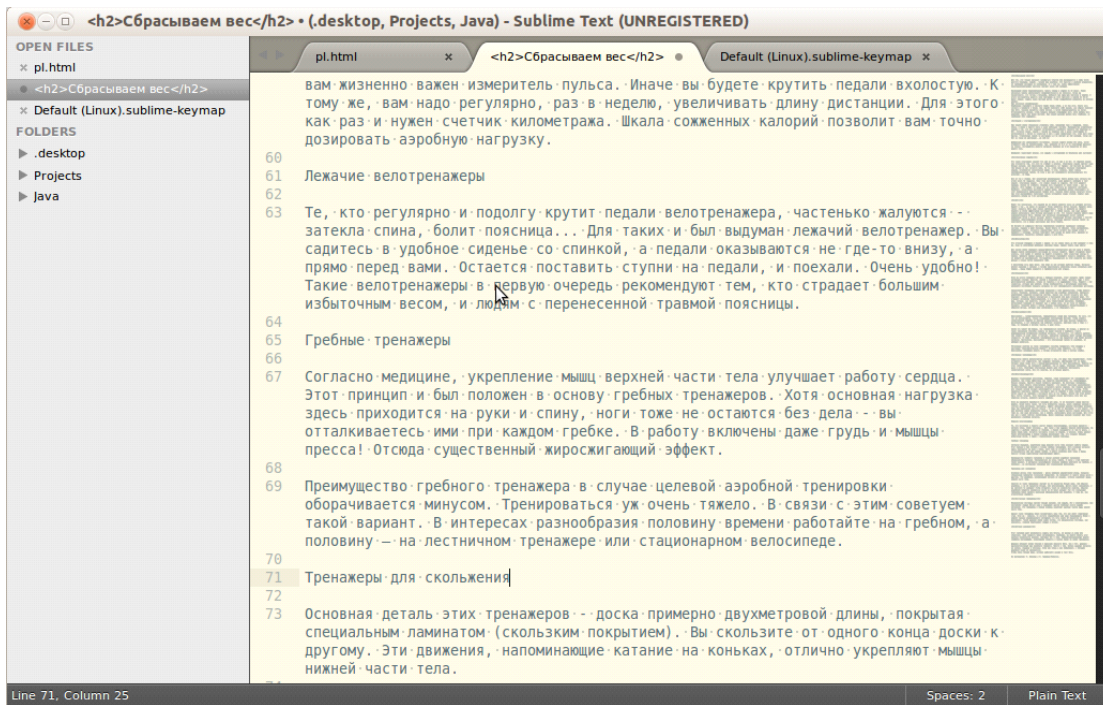
Для начала неплохо было бы вспомнить, что такое макрос (macros)? В любой программе, будь то MS Excel, Adobe Photoshop, Notepad++ или же Sublime Text макрос (macros) - *это записанная этой программой последовательность действий, выполненная пользователем. Образно говоря, пользователь говорит программе - “смотри, запоминай и повторяй за мной”.*

Как записать макрос?

В моем ST (ST3, under Ubuntu Linux 12.04) запись нового макроса можно инициировать комбинацией C(Control)+M(Meta = Alt)+Q, или перейти в меню “Tools -> Record Macro”.

Что записываем?

У нас есть текст, в котором отдельные строки должны стать заголовками.



Пример ситуации

Теперь мы ставим курсор на строку 65 и иницилируем запись нового макроса. Жмем следующую комбинацию клавиш: **Home<h2>End</h2>**

В итоге строка 65 должна выглядеть так: **<h2>Гребные тренажеры</h2>**

Сохраняем макрос

Макрос записан, сохраняем.

Для этого делаем следующую последовательность действий:

-Останавливаем запись макроса – C+M+Q

-Сохраняем макрос – “Tools -> Save Macro...”

-При сохранении используйте папку

\$HOME_OF_SUBLIME_CONFIG/Packages/User/, которая в общем-то предлагается по умолчанию.

Для своего макроса я выбрал имя файла Insert H2.sublime-macro.

Назначаем горячие клавиши

Сам по себе макрос не очень удобная штука, но если назначить его на какую-либо комбинацию клавиш, ценность его повышается в разы.

Для того чтобы добавить свой шорткат, нужно зайти в “Preferences -> Key Bindings User”. Нам откроется файл конфигурации, в котором мы можем прописать наши кастомные шорткаты:

```
[ { "keys": ["ctrl+2"], "command": "run_macro_file", "args": { "file":  
"Packages/User/Insert H2.sublime-macro"}} ]
```

Внимание! Не редактируйте файл “Key Bindings Default”, все свои правки вносите в “Key Bindings User”.

Используем

Теперь мы можем перейти на строку 71, нажать “C+2” и наша строка магическим образом обернется в заголовок второго уровня. Ура!

Syntax Specific Preferences

Если вас не устраивает тип и длина таба для некоторых языков в ST, то это недоразумение можно быстро исправить.

Для этого выставляем текущий язык на нужный(C+Shift+P -> Set Syntax \$LANG_NAME), и заходим в “Preferences -> Settings – More -> Syntax

Specific – User”, тут отрывается привычный нам конфиг в виде JSON файла.

№8 Задания на закрепление материала

ЗАДАНИЕ 1.

Закодируйте слова (1 –5), используя ключ к заданию.

Ключ к заданию:

1 2 3 4 5 6 7 8 1 9 10 4 11 1

МЫ С БРАТОМ ЛЮБИМ

3 1 8 7 5 12 7 13 1 14 9 13 7 15 11 9 13 1 2

СМОТРЕТЬ МУЛЬТФИЛЬМЫ

Задания:

1) 11 10 9 13

2) 4 8 5 7

3) 6 11 3 7

4) 4 14 3 2

5) 4 5 14 3

ЗАДАНИЕ 2.

1. Сколько бит памяти займет слово «Микропроцессор»?

Слово состоит из 14 букв. Каждая буква – символ компьютерного алфавита, занимает 1 байт памяти. Слово занимает 14 байт $= 14 * 8 = 112$ бит памяти.

Ответ: 112 бит

2. Текст занимает 0, 25 Кбайт памяти компьютера. Сколько символов содержит этот текст? ([1], с.133, №31)

Переведем Кб в байты: $0, 25 \text{ Кб} * 1024 = 256$ байт. Так как текст занимает объем 256 байт, а каждый символ – 1 байт, то в тексте 256 символов.

Ответ: 256 символов

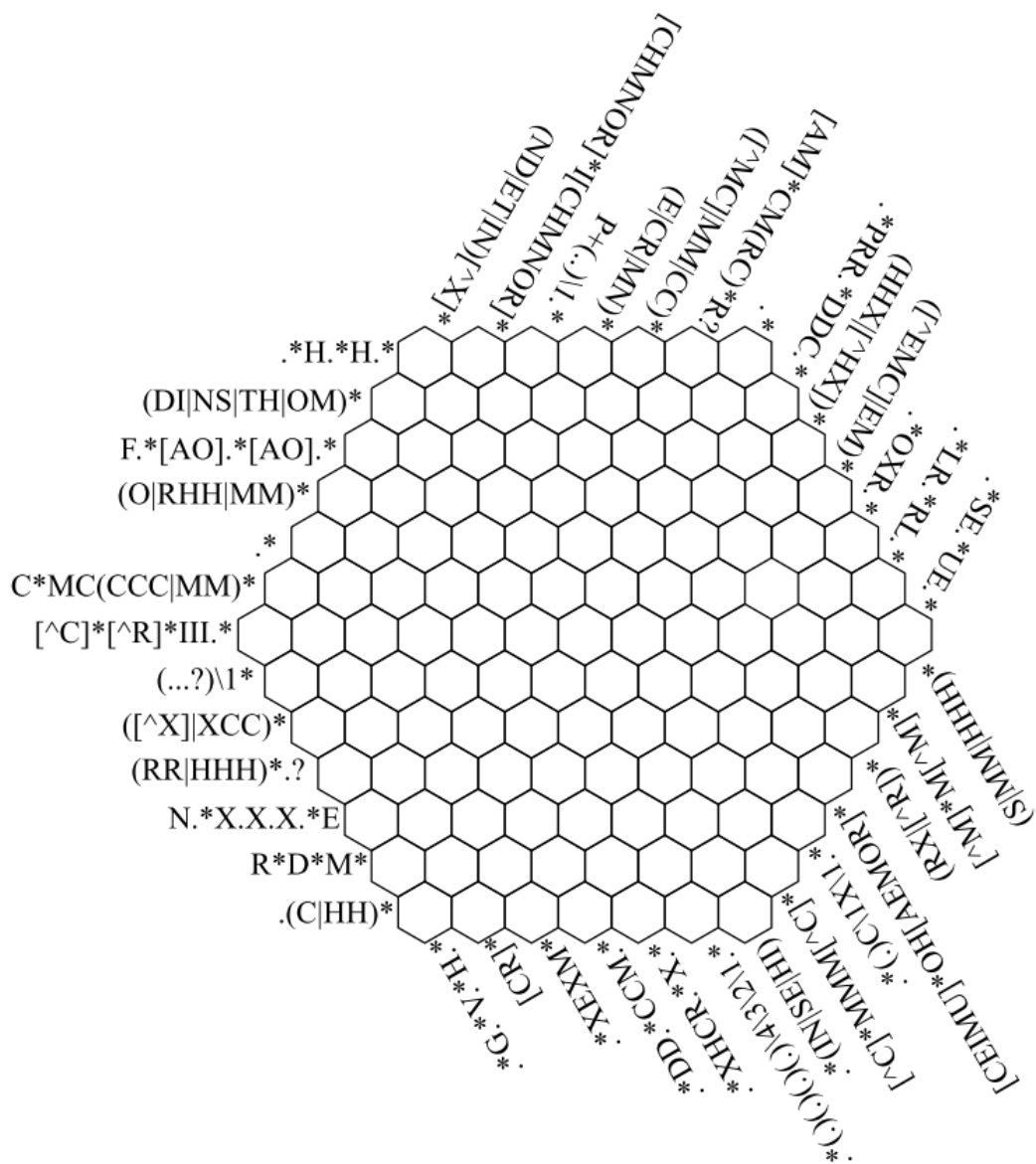
3. Текст занимает полных 5 страниц. На каждой странице размещается 30 строк по 70 символов в строке. Какой объем оперативной памяти (в байтах) займет этот текст? ([1], с.133, №32)

$30 * 70 * 5 = 10500$ символов в тексте на 5 страницах. Текст займет 10500 байт оперативной памяти.

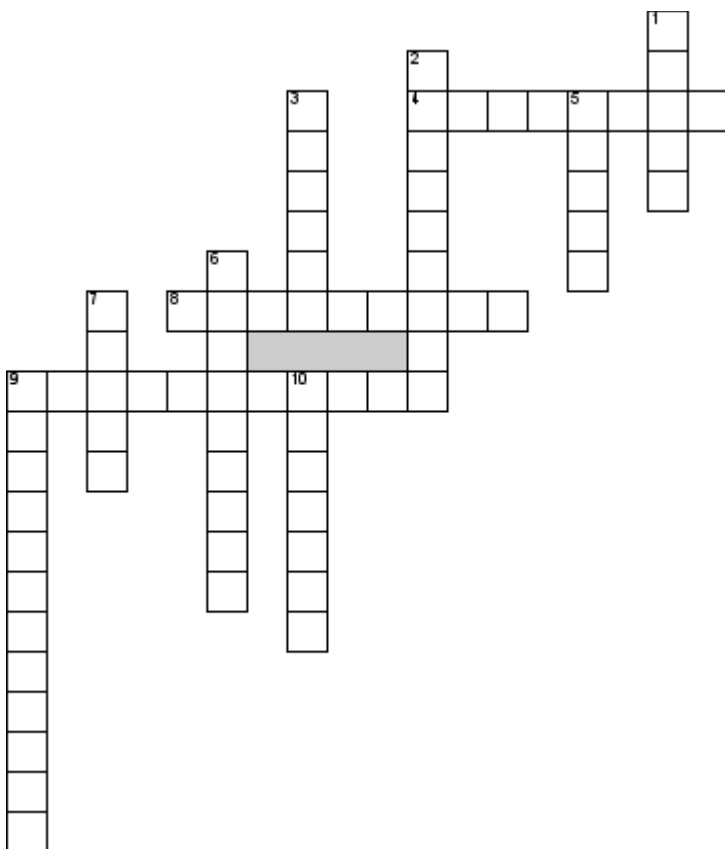
Ответ: 10500 байт

ЗАДАНИЕ 3.

Решите кроссвод из регулярных выражений



ЗАДАНИЕ 4.



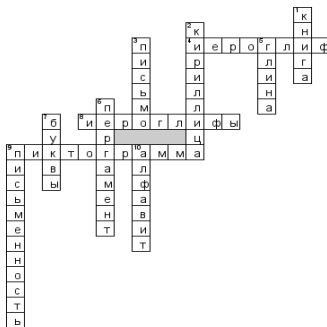
По горизонтали

- 4. Древние рисуночные знаки египетского письма.
- 8. Название письменного знака в некоторых системах письма.
- 9. Знак, отображающий важнейшие узнаваемые черты объекта, предмета или явления.

По вертикали

- 1. Источник знаний.
- 2. Старославянская азбука.

3. Средство закрепления речевой информации при помощи знаков или изображений.
5. Пластичная горная порода.
6. Материал для письма из недубленой сыромятной кожи животных.
7. Отдельный символ какого-либо алфавита
9. Знаковая система, предназначенная для передачи тех или иных данных на расстоянии.
10. Форма письменности, основанная на стандартном наборе знаков.



ЗАДАНИЕ 5.

Контрольные вопросы по основам сетей.

Вопрос №1. Какой уровень OSI отвечает за надежную доставку данных?

(Транспортный)

Вопрос №2. Как называется процесс добавления заголовков к данным при прохождении их от одного уровня OSI к другому?

(Инкапсуляция)

Вопрос №3. Для чего необходима модель OSI?

(Модель OSI была разработана для упрощения взаимодействия различные сетевых устройств друг с другом. Кроме того, данная модель позволяет привести к единой стандартизации сетевых протоколов, работающих на оборудовании различных производителей. Каждый уровень описывает определенные процессы и действия и такое разбиение позволяет лучше и быстрее изучить работу протоколов и устройств)

Вопрос №4. Какой уровень формирует сегмент?

(Транспортный)

Вопрос №5. Какой уровень формирует кадр?

(Канальный)

№9 Ресурсы

<http://www.oboznik.ru/?p=56735>

https://studopedia.ru/3_56361_lektsiya--kodirovki-i-formati-dannih-ispolzuemie-v-internet.html

<https://megalektsii.ru/s1683t8.html>

<https://ichigarev.ru/sozдание-saita/notepad-chto-eto-za-programma.html>

<http://wpmen.ru/znakomstvo-i-instrukciya-po-rabote-s-notepad.html>

<https://ru.wikipedia.org/wiki/Notepad%2B%2B>

<https://maxspblogt.wordpress.com/2016/10/05/программирование-в-notepad/>

<https://studfiles.net/preview/3816140/>

<https://notepad-plus-plus.org>

<https://ru.wikipedia.org/wiki/Notepad%2B%2B>

<http://kinf.ucoz.ua/labphp/node109.html>

https://ru.wikipedia.org/wiki/Регулярные_выражения

[https://www.hse.ru/data/2013/06/11/1296162306/
metclabnew2013_11.06_.pdf](https://www.hse.ru/data/2013/06/11/1296162306/metclabnew2013_11.06_.pdf)

<https://studfiles.net/preview/6750024/>

<https://pavel-uszakov.livejournal.com/10231.html>

<http://gearmobile.github.io/sublime/sublime-macros/>

<https://toster.ru/q/397953>

<https://habr.com/post/349076/>

<https://ruslan.ibragimov.by/2014/01/06/sublime-text-zametki/>