

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Омский государственный технический университет»

ОСНОВЫ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2019

Основы профессиональной деятельности : метод. указания / [сост. А. Г. Белик] ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2019.

Приведены цели, задачи и требования к выполнению лабораторных занятий по дисциплине «Основы профессиональной деятельности». Даны основы теоретических знаний и задания для лабораторных занятий с целью приобретения навыков в области современных компьютерных информационных технологий для решения разнообразных прикладных задач с использованием операционных и офисных программных систем, сетевых и веб-технологий, инструментальных сред программирования, программных комплексов схемотехнического и структурного моделирования.

Предназначено для студентов направлений подготовки бакалавриата: 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 27.03.03 «Системный анализ и управление».

*Рекомендовано редакционно-издательским советом
Омского государственного технического университета*

© ОмГТУ, 2019

Зачет по дисциплине «Основы профессиональной деятельности» выставляется только при наличии всех сданных лабораторных работ и выложенного в портфолио отчета по домашнему заданию (СРС) в каждом семестре.

Номер варианта индивидуального задания выдается преподавателем и изменению не подлежит.

В первом семестре студент выполняет лабораторные работы под номером 1.1, 1.2, 1.3.

Во втором семестре студент выполняет лабораторные работы под номером 1.4, 1.5.

Задания для СРС приведено ниже в разделе ДОМАШНЕЕ ЗАДАНИЕ.

ЛАБОРАТОРНЫЕ РАБОТЫ 1 СЕМЕСТР

1.1 Построение графика функции с использованием Ms Excel, создание макроса для построения графика функции

Цель работы: приобретение и закрепление практических навыков по обработке числовых данных в электронных таблицах и основы компьютерной коммуникации.

Задание.

- 1) Построить график функции с использованием Ms Excel (по методическим указаниям).
- 2) Создать макрос для построения графика функции (по методическим указаниям).
- 3) Построить график функции с использованием Ms Excel (по индивидуальному заданию).
- 4) Создать макрос для построения графика функции (по индивидуальному заданию).
- 5) Защитить работу преподавателю, продемонстрировав полученные навыки.

Краткая теория и методические указания

Построение графика функции с использованием Ms Excel, создание макроса для построения графика функции

Построение графиков функций – это, хотя и трудоемкая (при ручном выполнении), но в то же время весьма полезная математическая операция. Часто графики используются как наиболее простое и наглядное средство, позволяющее быстро выявить наиболее важные особенности исследуемых функций [10].

Построение графиков достаточно легко осуществляется с помощью Ms Excel.

Создайте в Вашей папке файл – приложение Excel – с именем «Графики функций» и откройте его. Дайте двум листам документа имена: «График» и «Макрос построения графика».

Перейдите на лист «График». В верхней части листа наберите текст – заголовок листа: «Построение графиков».

Введите в ячейку **B3** текст «Нач. знач-е», в ячейку **C3** – « $f(x)$ », в ячейку **D3** – «Кон. знач-е». В ячейки **B4**, **C4**, **D4** будем вводить исходные данные

задачи: начало интервала, функцию и конец интервала соответственно. Выделите диапазон ячеек **B3:D4** и с помощью кнопки «все границы» обозначьте рамки нашей «таблички» для начальных условий. Соседние с табличкой ячейки залейте серым цветом. Для придания таблице удобочитаемого вида выделите блок ячеек **B4:D4**, отцентрируйте текст и сделайте шрифт полужирным. Можно слегка расширить столбцы, если текст заголовков не помещается в ячейках.

В ячейку **B6** введите текст «шаг =». Выровняйте его по правому краю. В ячейке **C6** будет *вычисляться* шаг построения графика. Соседние с **B6** и **C6** ячейки также залейте серым цветом.

Подготовимся к табулированию функции. Для этого в ячейку **A8** введите текст «№ шага», в ячейку **B8** – «x», в ячейку **C8** – «f(x)». В столбцах **A**, **B** и **C** будут находиться, соответственно, номер строки таблицы, значение аргумента и значение функции в этой строке. Выделите диапазон ячеек **A8:C8** и с помощью кнопки «все границы» сформируйте рамки введенного заголовка таблицы.

Лист готов к построению графика функции (рис. 1).

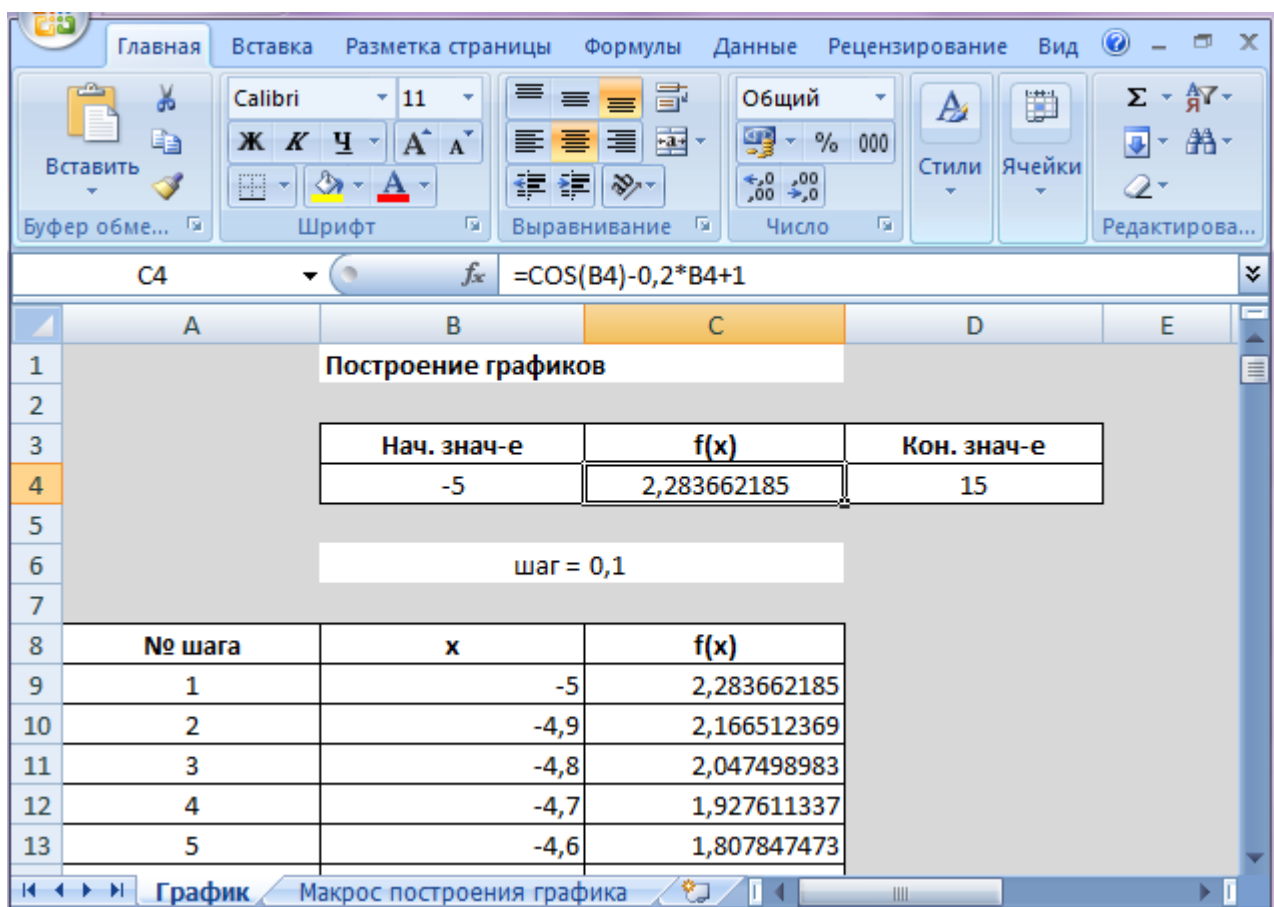


Рисунок 1 – Пример выполнения задания

Теперь решим следующую задачу: построим график функции

$f(x) = \cos(x) - 0,2x + 1$ на интервале $[-5; 15]$.

Внимание! Не вставляйте формулы в MS Excel путем копирования из методических указаний. Формулы необходимо вводить самостоятельно. Адреса ячеек можно вводить в формулы без использования клавиатуры, просто щелкая по ним в нужный момент мышью.

Введем исходные данные задачи, а именно: в ячейку **B4** – значение -5, в ячейку **C4** – формулу **=cos(B4)-0,2*B4+1**, в ячейку **D4** – значение 15.

Для построения графика разобьем заданный интервал на 200 равных отрезков, длину которых занесем в ячейку **C6**. Введите в ячейку **C6** формулу **=(D4-B4)/200**. Проверьте в уме правильность полученного в **C6** результата.

Теперь можно начать табуляцию функции.

Введите в ячейку **A9** значение 1, в ячейку **B9** введите формулу **=B4**, а в ячейку **C9** скопируйте формулу из ячейки **C4**. Для этого выделите ячейку **C4** и с помощью команды меню **Правка/Копировать** скопируйте ее содержимое в буфер обмена. Далее выделите ячейку **C9** и воспользуйтесь командой меню **Правка/Вставить**.

Заполним весь столбец «№ шага». Выделите ячейку **A9**. Воспользуйтесь командой меню **Главная/Редактировать/Заполнить/Прогрессия...** (рис. 2).

В появившемся окне задайте следующие параметры:

- **расположение** по столбцам;
- **тип** арифметическая;
- **шаг** 1;
- предельное значение 201.

Нажмите **ОК**.

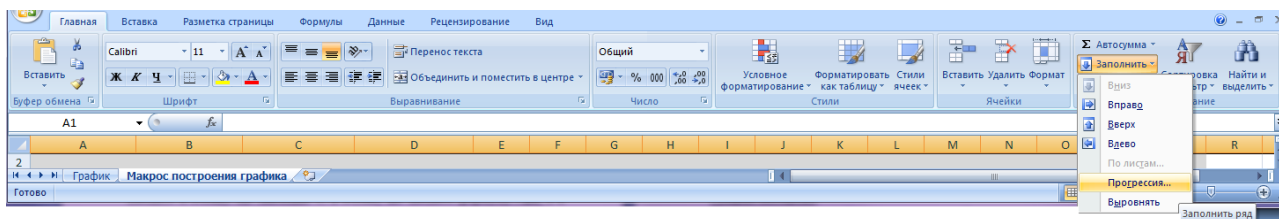


Рисунок 2 – Заполнение шага

Продолжим заполнение таблицы. Введите в ячейку **B10** формулу **=B9+\$C\$6**. В ячейку **C10** также можно ввести формулу **=cos(B10)-0,2*B10+1**, но проще ее просто скопировать из ячейки **C9**. Для этого выделите ячейку **C9**, наведите мышь на *маркер заполнения* – маленький квадрат в правом нижнем углу ячейки. После того, как курсор превратится в черное перекрестие, нажмите левую кнопку мыши и, удерживая ее нажатой, протащите маркер вниз.

Проанализируйте результат ваших действий: вместе с содержимым ячеек (формулами) копируются их форматы. Получается не очень красиво. Позже, при создании макроса, учтем это и выберем другой способ копирования.

Используя методы копирования, сформируем оставшуюся часть таблицы. Выделите блок ячеек **B10, C10** и протащите его маркер заполнения вниз, пока не получите 201 значение (другой вариант – просто щелкните двойным щелчком по маркеру заполнения). Получив таким образом, значения аргумента и функции во всех выбранных точках интервала, мы произвели табулирование функции.

Можно строить график функции. Укажите выделением диапазон ячеек **B9:C209**, по которому нужно построить график. Для этого выделите сначала ячейку **B9**, затем при нажатой комбинации клавиш Ctrl-Shift воспользуйтесь клавишами управления → и ↓.

Для построения графика воспользуемся командой меню **Вставка/Диаграмма...**, или кнопкой **Мастер диаграмм** на панели **Стандартная**. В появившемся окне на вкладке **Стандартные** выберите тип **Точечная**, вид диаграммы – **Точечная диаграмма со значениями, соединенными сглаживающими линиями без маркеров**. Нажмите **Далее**. В открывшемся окне можно указать источник данных, но нужный нам диапазон ячеек уже предложен по умолчанию, т. к. мы его предварительно выделили. Поэтому переходим сразу к следующему шагу построения диаграммы. На вкладке **Заголовки** введите **Название диаграммы** « $f(x)=\cos(x)-0,2*x+1$ ». Введите также обозначение «Y» в микроокно **Ось Y**, и обозначение «X» в микроокно **Ось X**. На вкладке **Легенда** уберите галочку из **Добавить легенду**. Нажмите **Готово**.

График построен (см. рис. 3).

Переместите его вверх страницы.

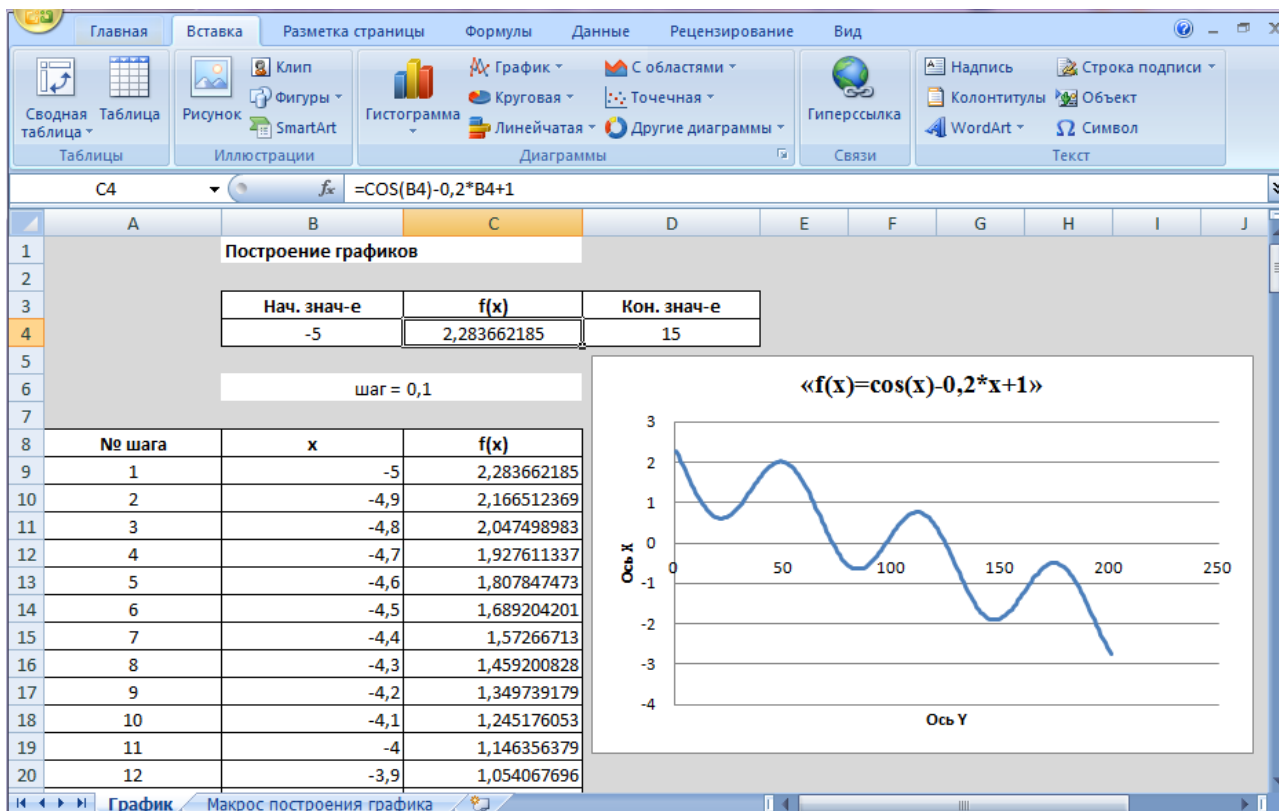


Рисунок 3 – График функции

Если требуется многократное решение типовой задачи в Ms Excel, то ее решение можно автоматизировать с помощью *макроса*.

Макрос – это последовательность команд и функций, которая сохраняется нами под определенным именем [9, 10]. Ее можно вызывать по заданному имени или путем нажатия заданного сочетания клавиш всякий раз, когда необходимо выполнить данную типовую задачу. Например, если нам необходимо строить разные графики функций, то можно создать макрос для построения графика.

Создадим такой макрос.

Перейдите на лист «Макрос построения графика».

Для создания макроса нам нужно повторить некоторые действия, отработанные на предыдущем листе, и добавить указания о записи макроса.

Скопируйте диапазон ячеек **A1:D8** с листа «График» и вставьте его на лист «Макрос построения графика» в соответствующие ячейки. Если скопировался и заголовок листа, то перепишите его как-нибудь иначе, например, так: «Макрос построения графика: Ctrl-q ».

Далее для создания макроса следует воспользоваться командой меню **Вид/Макросы/Запись макроса...** В появившемся окне введите имя макроса и укажите сочетание клавиш, при нажатии которых в будущем будет запускаться его выполнение, например, Ctrl-q. Выберите **сохранить в этой книге** и нажмите **ОК**. С этого момента все выполняемые Вами операции будут

сохранены в виде команд макроса. Поэтому сейчас выполняйте все, что еще остается сделать для построения графика: табулируйте функцию и стройте график, как мы это делали в предыдущей части работы, начиная с фразы «Теперь можно начать табуляцию функции» и до слов «График построен». После этого остановите запись макроса с помощью команды меню **Сервис/Макрос/Остановить запись**.

При копировании формулы из ячейки **C9** в ячейку **C10** не пользуйтесь маркером заполнения, скопируйте ячейку **C9** в буфер, щелкните правой кнопкой мыши по ячейке **C10**, в появившемся контекстном меню выберите опции **Специальная вставка...** и затем – **Формулы**.

Макрос создан (рис. 4).

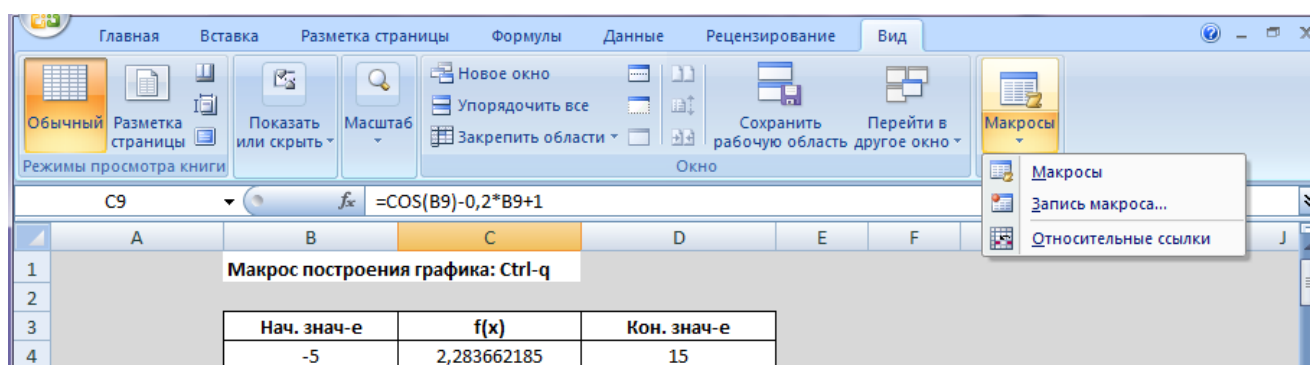


Рисунок 4 – Макрос построения графика

Варианты индивидуальных заданий

№ вар.	Формула	Ограничения
1	$\frac{x^2 + 5}{x^2 + 2};$	$0 \leq x \leq 2.$
2	$x^2 \cdot e^{-x^2};$	$1 \leq x \leq 3$
3	$\sin(x);$	$0 \leq x \leq \pi.$
4	$\frac{1}{\sin\left(\frac{\ln(\sqrt{5x-x^2})}{\cos(3x)}\right)};$	$0,14 \leq x \leq 3,67.$
5	$\frac{2x^2 - 3x + 1}{x + 1}$	$5 \leq x \leq 8.$
6	$\frac{x^2}{x-1};$	$1,5 \leq x \leq 3,5.$
7	$x \cdot \arctg(x);$	$\frac{\sqrt{3}}{3} \leq x \leq \text{до } \sqrt{3}.$
8	$3 \cdot \sqrt{x};$	$4,3 \leq x \leq \text{до } 6,8.$
9	$\cos(x);$	$\frac{\pi}{2} \leq x \leq 2\pi$

10	$\ln(x)$;	$0,5 \leq x \leq 2,5$.
11	$\frac{x}{1+x^2}$;	$0,5 \leq x \leq 2,5$.
12	$\frac{1}{x^2}$;	$\frac{1}{\sqrt{6}} \leq x \leq \frac{2,5}{\sqrt{6}}$
13	$(x + \frac{1}{x})^2$;	$4,3 \leq x \leq 6,8$
14	$\frac{1}{\sqrt{50-x^2}}$;	$2 \leq x \leq 4$.
15	$6x^2 - 3x + 5$;	$5 \leq x \leq 8$.
16	$1 + \sin(x^2)$;	$\frac{\pi}{4} \leq x \leq 5\frac{\pi}{4}$.
17	$\frac{x}{x^3+16}$;	$-2 \leq x \leq 1$.
18	$x^{1/2} - x^{1/3}$;	$0 \leq x \leq 4$.
19	$\frac{16x^2+2x+7}{x^2-8} + 1$;	$5 \leq x \leq 8$.
20	$\frac{(e^x-1) \cdot (e^{2x}+1)}{e^x}$;	$5 \leq x \leq 8$.
21	$\frac{x^3}{3-x^2}$;	$2,5 \leq x \leq 5,5$.
22	$\sin(x)$;	$\pi \leq x \leq 2\pi$.
23	$x^{1/3} + x^{1/4}$;	$0 \leq x \leq 1$.
24	$\ln(x) + x$;	$\pi \leq x \leq 3\pi$.
25	$\cos(x) / x$;	$\frac{\pi}{2} \leq x \leq 2\pi$
26	$\frac{x^3+10}{x^2-5}$;	$0 \leq x \leq \pi$.
27	$\frac{5x^2+2x+3}{x+3}$	$1 \leq x \leq 5$.
28	$\operatorname{tg}(x)/x$;	$\frac{1}{\sqrt{6}} \leq x \leq \frac{2,5}{\sqrt{6}}$
29	$x^3 \cdot e^x$;	$1 \leq x \leq 2$
30	$\frac{10}{\sqrt{100-x^3}}$;	$0 \leq x \leq 2$.

1.2 Перевод чисел из одних систем счисления в другие с использованием Ms Excel

Цель работы: приобретение практических навыков по обработке числовых данных в электронных таблицах и основы компьютерной коммуникации.

Задание.

- 1) Построить машину счисления согласно методических указаний.
- 2) Реализовать с использованием Ms Excel перевод чисел в десятичную систему счисления (по индивидуальному варианту задания).
- 3) Реализовать с использованием Ms Excel перевод чисел из десятичной системы счисления (по индивидуальному варианту задания).
- 4) Реализовать с использованием Ms Excel сложение чисел в позиционных системах счисления (по индивидуальному варианту задания).
- 5) Заполнить таблицу 1 со стр. 18, защитить работу преподавателю, продемонстрировав полученные знания.

Перевод чисел из одних систем счисления в другие с использованием Ms Excel

Системой счисления (нумерацией) обычно называют способ записи чисел с помощью специальных знаков или цифр. Так, общепринятая десятичная система использует 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. В вычислительных машинах чаще применяется двоичная система, использующая только две цифры: 0 и 1.

Все системы счисления принято делить на две группы: позиционные системы счисления и непозиционные системы счисления. В позиционной системе значение каждой цифры зависит от места (позиции), которое она занимает в последовательности цифр, обозначающей число. В непозиционной системе значение знака (цифры или какого-либо другого символа) не находится в зависимости от его места в записи числа [9, 10].

Для выполнения этой работы создайте в Вашей папке файл – приложение Excel – с именем «Системы счисления» и откройте его. Дайте трем листам имена: «В десятичную» «Из десятичной» и «Сложение».

Все три листа нужно сделать «в клеточку», т. е. задать ширину их столбцов примерно равной ширине строк. Это можно сделать одновременно для всех трех листов следующим образом.

На любом (например, первом) листе щелкните правой кнопкой мыши по ярлычку и выберите в открывшемся контекстном меню «Выделить все листы». Ярлычки всех листов станут «активными», белыми. Теперь щелкните кнопку

выделения всего листа (серый прямоугольник в левом верхнем углу листа на пересечении заголовков строк и столбцов). Весь лист станет выделенным. Установите курсор между заголовками любых из двух столбцов и левой кнопкой мыши «стяните» ширину столбца до 2.00 (19 пиксел). Щелкните теперь любую ячейку, чтобы этим закончить операцию. Все столбцы сделались одной ширины, и ячейки приобрели форму квадратных клеточек. То же самое (проверьте) произошло с другими двумя листами. Книга подготовлена к работе. Перейдем к листу «В десятичную».

На листе «В десятичную» создадим «машину» для перевода в десятичную систему счисления чисел, записываемых в системах с произвольным основанием $B > 1$.

С этой целью разметим предварительно лист так, как показано на рис. 5.

В верхней части листа наберите текст - заголовок листа: «Перевод чисел из системы счисления с основанием B в десятичную систему счисления». Растяните в ширину столбец **B**, в котором будем записывать основание системы счисления (в ячейку **B8**) и получать десятичное число – результат перевода (ячейка **B15**). С помощью команды-кнопки меню «внешние границы» разметьте диапазоны для записи цифр целой части числа (**D8:W8**) и дробной части числа (**Y8:AR8**). Сделайте необходимые текстовые заголовки-пояснения к этим размеченным диапазонам ячеек. Соседние с этими диапазонами ячейки можно залить серым цветом для улучшения дизайна нашей «машины» (рис. 5). Если вся «машина» не помещается на экране, то можно слегка уменьшить масштаб листа.

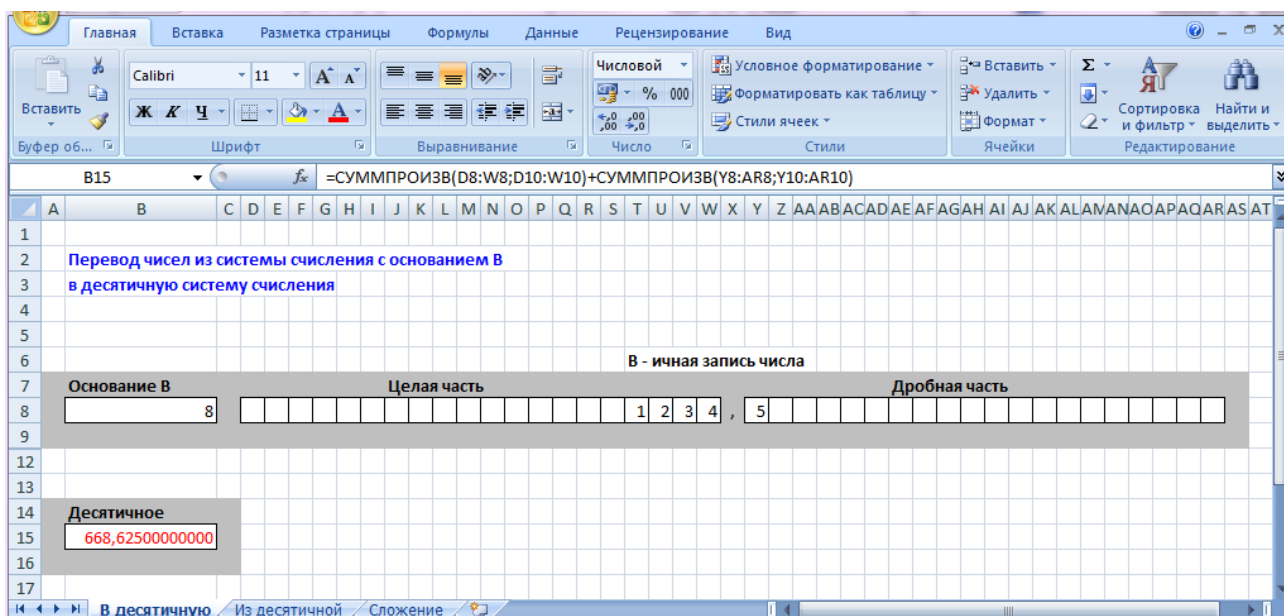


Рисунок 5 – Примерный вид экрана с листом «В десятичную»

Строка 8 листа готова для записи в ячейку **B8** основания системы счисления и для записи в ячейки **D8:AR8** цифр числа, представленного в этой системе

счисления. При этом как целая, так и дробная части числа могут содержать до 20 цифр. Остается сделать так, чтобы в ячейке **B15** появлялось *десятичное* число, записанное в строке **8** в системе счисления с основанием *B*. Для этого нужно лишь правильно ввести в ячейки несколько простых формул.

Во всех этих формулах нам, разумеется, нужно будет ссылаться на одну и ту же ячейку **B8**. И, чтобы при копировании формул ссылка на эту ячейку не изменялась, эту ссылку нужно писать как абсолютную, т. е. в виде **\$B\$8**. Однако наличие ссылок такого вида в формулах делает их не очень удобными для чтения. Поэтому воспользуемся другим механизмом присвоения абсолютного имени ячейкам и диапазонам. Чтобы присвоить ячейке **B8** имя, например, *a*, - выделим ее левой кнопкой мыши. Взгляните на *поле имени* - оно расположено на верхней панели инструментов, слева от строки формул. Введите в *поле имени* букву *a* и нажмите Enter. Имя, присвоенное сейчас выделенной ячейке, является ее абсолютным адресом, уникальным в пределах всей книги. Если, например, сейчас вы перейдете на другой лист и в поле имени выберете *из списка* имя *a*, то тут же окажетесь вновь на листе «В десятичную», где будет выделена ячейка **B8**. Итак, основание системы счисления будет именоваться на листе как *a*, и в формулах вместо **\$B\$8** мы будем писать просто букву *a*.

Введите в ячейку **B8** в качестве основания системы счисления число 3. Для перевода чисел в десятичную систему зададим веса разрядов. В ячейку **W10** (т.е. под младшим разрядом целой части) введите 1 – это *вес младшего разряда*. В соседнюю слева ячейку введите формулу **=W10*a**. В ней появится число 3. Скопируйте эту ячейку с формулой ее «растягиванием» за маркер заполнения влево, до старшего разряда. Теперь мы имеем все *веса целой части* числа. Многие их значения не входят в клеточки и выводятся как символ решетки. Но нам и не нужно их видеть. Скоро мы скроем всю эту строку.

Аналогично задайте *веса для дробной части* числа: в клетку **Y10** введите формулу **=1/a**. Результат из-за его округления до одной цифры будет выглядеть как 0. В соседнюю справа ячейку введите формулу **=Y10/a**. Затем «растяните» - скопируйте ее до младшего разряда дробной части. Теперь в ячейку **B15** введите формулу:

=СУММПРОИЗВ(D8:W8;D10:W10)+СУММПРОИЗВ(Y8:AR8;Y10:AR10).

Программа для перевода чисел в десятичную систему счисления готова. Формулу для вычисления суммы произведений цифр разрядов на их веса пришлось разбить на две части, т. к., из-за дизайнерских соображений, в колонке **X** между двумя диапазонами разрядов нам пришлось в голову поместить запятую. Если запятую оттуда удалить, то формулу в ячейке **B15** можно записать сразу для двух сплошных диапазонов в виде

=СУММПРОИЗВ(D8:AR8;D10:AR10).

Запишите в ячейки **W8** и **Y8** цифры 1 и 1. Остальные разряды троичного числа пусть будут нули (или пустые ячейки - это то же самое). Таким образом, мы ввели троичное число $1,1_3$. В ячейке **B15** появится десятичное изображение $1,3333\dots$ этого троичного числа. Увеличьте ширину столбца B, как на рис.5 или еще больше.

Выделите все ячейки с *входными данными*: для этого щелкните ячейку **B8** и при нажатой клавише Ctrl выделите мышкой диапазон **D8:AR8**. После выделения войдите в меню **Формат/Ячейки...**, в открывшемся окне на вкладке **Защита** снимите флажок **Защищаемая ячейка**, и подтвердите это действие клавишей **ОК**. Теперь выделенные нами ячейки останутся доступными для ввода данных после защиты листа. Затем выделите строки **10** и **11**, щелкните по ним правой клавишей мыши и выберите из контекстного меню «**Скрыть**». Чтобы защитить лист, войдите в меню **Сервис/Защита/Защитить лист** и нажмите **ОК**, не задавая пароль. Снимать защиту листа можно будет без пароля, через те же пункты меню **Сервис/Защита**.

Так мы застраховались от случайного изменения всех ячеек, кроме тех, где нужно будет вводить данные при решении задач. Попробуйте выполнить какие-нибудь непредусмотренные изменения. Попробуйте, например, отобразить скрытые строки, или изменить формулы. Необходимые для этого действия окажутся недоступны. Можно приступать к экспериментам. Сохраните файл, но не закрывайте его.

Первый проверочный опыт можно провести сейчас. Запишите в строке **8** в качестве основания исходной системы счисления число 10. Запишите в этой же строке цифры целой и дробной частей числа 1234,5. Эта запись автоматически преобразуется и отобразится в ячейке **B15** в виде десятичного числа 1234,5. Теперь в ячейке **B8** запишите 8. При *таком* основании исходной системы счисления число, записанное в строке **8**, отобразится в виде десятичного числа 668,625. Это изображено на рис. 5. «Машина» работает правильно.

Перейдите на лист «Из десятичной».

Чтобы построить «машину» для перевода десятичных чисел в системы счисления с любыми основаниями, разметьте ячейки: ячейку **K7** – для ввода основания системы счисления B , ячейку **B10** – для ввода целой части N исходного десятичного числа, и ячейку **B18** – для ввода его дробной части Z (рис. 6).

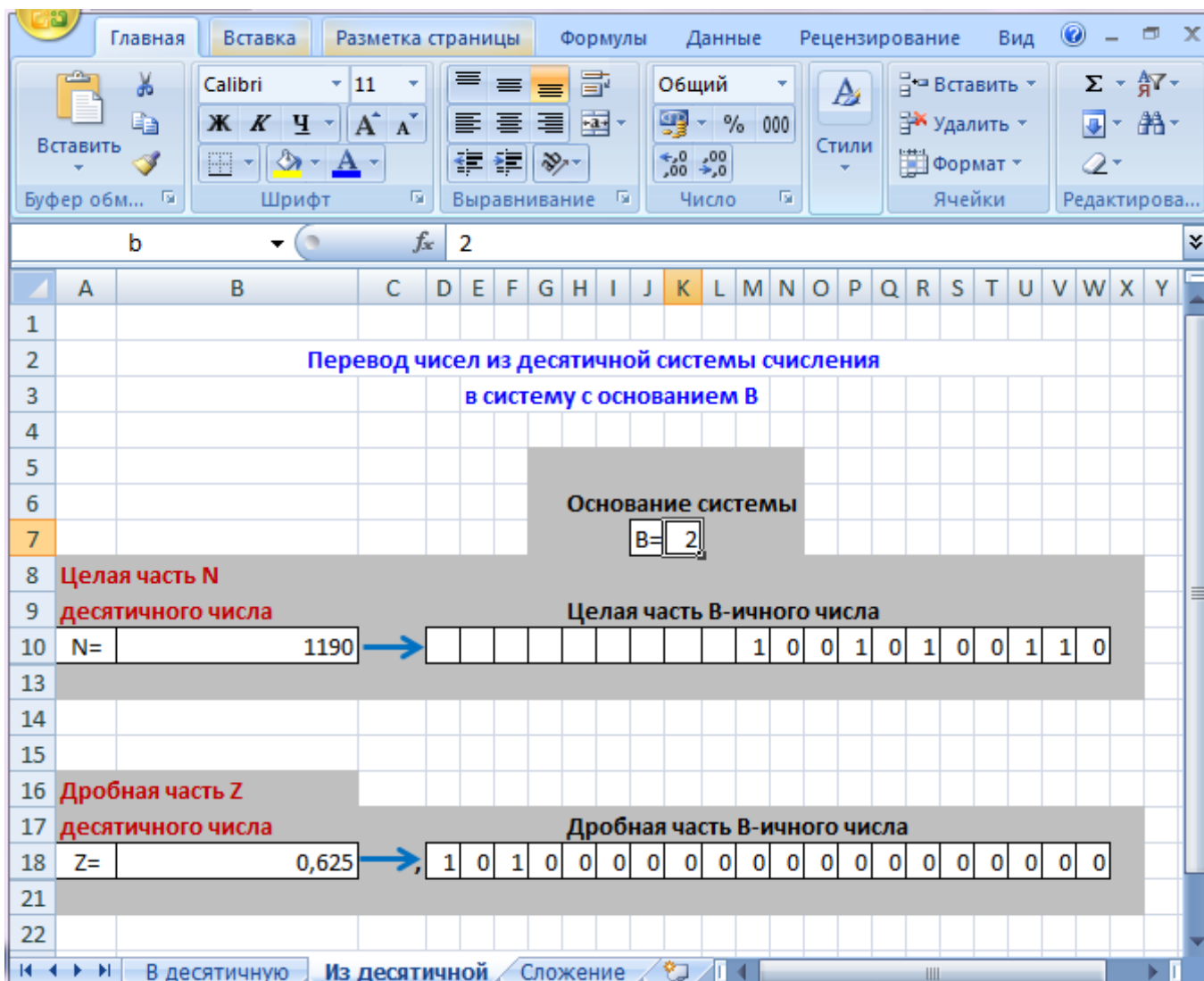


Рисунок 6 – Примерная организация листа «Из десятичной»

Присвойте выбранным ячейкам имена b , N и Z , используя окно имен. Обратите внимание: на рис. 6 выделена ячейка **K7** с записанным в ней числом 2, а в окне имен (вверху слева) высвечено имя b этой ячейки. Кроме того, строки 11, 12 и 19, 20 (с промежуточными результатами) на листе уже скрыты, нам же эти строки пока еще нужны для работы.

Ширину столбца **B** увеличьте.

Сделайте с помощью форматирования границ разметку диапазонов **D10:W10** и **D18:W18** для отображения цифр переведенного числа. Позднее можно будет поработать и над дизайном листа в целом. В ячейки для записи чисел B , N и Z запишите 2, 1190 и 0,625 соответственно; сейчас мы начнем перевод в двоичную систему счисления десятичного числа 1190,625.

В строках 11 и 12 разместим формулы для пересчета целой части N десятичного числа в заданную систему счисления с любым основанием B , реализуя правило последовательного деления N на B и вычисления остатков. Формулы для определения *частных* от деления разместим в диапазоне ячеек

D11:W11, под разрядами B -ичного числа. Строкой ниже введем формулы для вычисления последовательных *остатков* от деления N на B . Эти остатки и будут цифрами B -ичного числа.

Итак, введем в ячейку **W11**, под младшим разрядом, формулу **=ЦЕЛОЕ(N/b)**. Поскольку результат деления – частное – не помещается в ячейку, то в ней выводится символ «решетка». В соседнюю слева ячейку **V11** введем формулу **= ЦЕЛОЕ(W11/b)**. Скопируем эту ячейку влево до старшего разряда, т. е. на весь диапазон **D11:V11**.

В ячейку **W12** запишем формулу **=ОСТАТ(N;b)** – по которой определяется младшая цифра числа. В соседнюю слева ячейку **V12** запишем **=ОСТАТ(W11;b)**, т. к. далее мы должны получать остатки от деления каждого предыдущего частного на b . Скопируем эту формулу влево, до старшего разряда, т. е. в диапазон **D12:V12**.

Мы видим в строке **12** все цифры B -ичного (в данном случае – двоичного) числа, и на этом можно было бы остановиться. Однако потребуем еще, чтобы незначащие нули в старших разрядах не выводились. Тогда переносить полученные цифры в приготовленную для них «разрядную сетку» **D10:W10** нужно с помощью условной функции «ЕСЛИ». Сделаем это так.

В клетку **W10** введем формулу **=W12**. Младший разряд переносится в отображаемые цифры безусловно. Левее, в клетку **V10**, вводим формулу **=ЕСЛИ(СУММ(\$D\$12:V12)=0;"";V12)**. Смысл этой формулы таков: если слева от разряда **V12** (включая и сам этот разряд) *все нули*, то в ячейку записывается результат «пусто» (пустой текст между апострофами), иначе пишется *цифра* из ячейки **V12**. Ячейку **V10** с введенной формулой копируем влево до конца разрядной сетки. Если все сделано правильно, то результат перевода соответствует рис. 6.

Дробная часть Z десятичного числа переводится аналогично, только при этом применяется не деление, а умножение Z на основание системы счисления. После каждого умножения целая часть результата забирается (вычитается) из него и переносится в качестве очередной цифры в состав дробной части перевода числа. При этом цифры дробной части появляются в порядке слева направо.

В строках **19** и **20** для такого перевода выполните следующие действия:

- в ячейку **D19** введите **=b*Z**;
- в ячейку **D18** введите **=ЦЕЛОЕ(D19)** – это первая цифра дробной части;
- в **D20** вводите формулу **=D19 - D18** – вычитаем целую часть из результата;
- в **E19** – формулу **=b*D20**, и скопируйте эту ячейку вправо до **W19**;
- скопируйте ячейку **D18** вправо вплоть до **W18**;

– скопируйте ячейку **D20** вправо вплоть до **W20**.

Если все сделано правильно, то цифры дробной части будут такими, как на рис. 6.

Теперь скройте строки **11, 12** и **19, 20**. Выделите ячейки с исходными данными (**K7, B10, B18**), через меню формата ячеек снимите с них флажок «защищаемая ячейка» и защитите лист, не задавая пароля. Лист готов к экспериментам и к решению задач. Сохраните файл, «Машина» работает правильно.

Устройство листа «Сложение» – наиболее простое в данной работе. Перейдите на этот лист. Разметив ячейки и диапазоны для ввода основания системы счисления, ввода цифр слагаемых и отображения суммы слагаемых (рис. 7), введите в диапазон ячеек **D12:AC12** формулы для вычисления *переносов*. Для этого введите в ячейку **B8** число 8, а в ячейку **AC12** – формулу **=ЦЕЛОЕ((AC9+AC8+AD12)/\$B\$8)**. Смысл формулы в том, что, когда сумма двух разрядов и переноса из предыдущего разряда будет больше основания системы счисления, то сформируется перенос в следующий разряд. Скопируйте ячейку **AC12** с формулой влево до ячейки **D12**.

Аналогично можно задать формулы для вычисления разрядов суммы. В ячейку **AC10** занесите формулу **=ОСТАТ(AD12+AC8+AC9;\$B\$8)** – это часть той же суммы, остающаяся в данном разряде. Скопируйте эту ячейку влево до конца разрядной сетки.

Скройте строку **12**. Введите цифры семеричных слагаемых как на рис. 7. Если Вы не ошибались, то получится и соответствующая этому рисунку сумма.

На рис. 7 видно, что группа разрядов чисел «разделена» на листе на две части стрелками. Так можно условно отмечать положение разделительной запятой, когда нужно интерпретировать суммирование как операцию над дробными числами. На рис. 7 сложены дробные числа из последнего варианта индивидуальных заданий к данной работе.

Выделите ячейку **B8** и, прижимая клавишу **Ctrl**, выделите одновременно с ячейкой диапазон **D8:AC9**. Снимите через меню формата флажок защиты выделенных ячеек, т. к. эти ячейки нужно оставить доступными для ввода данных после защиты всего листа. Защитите лист и сохраните файл.

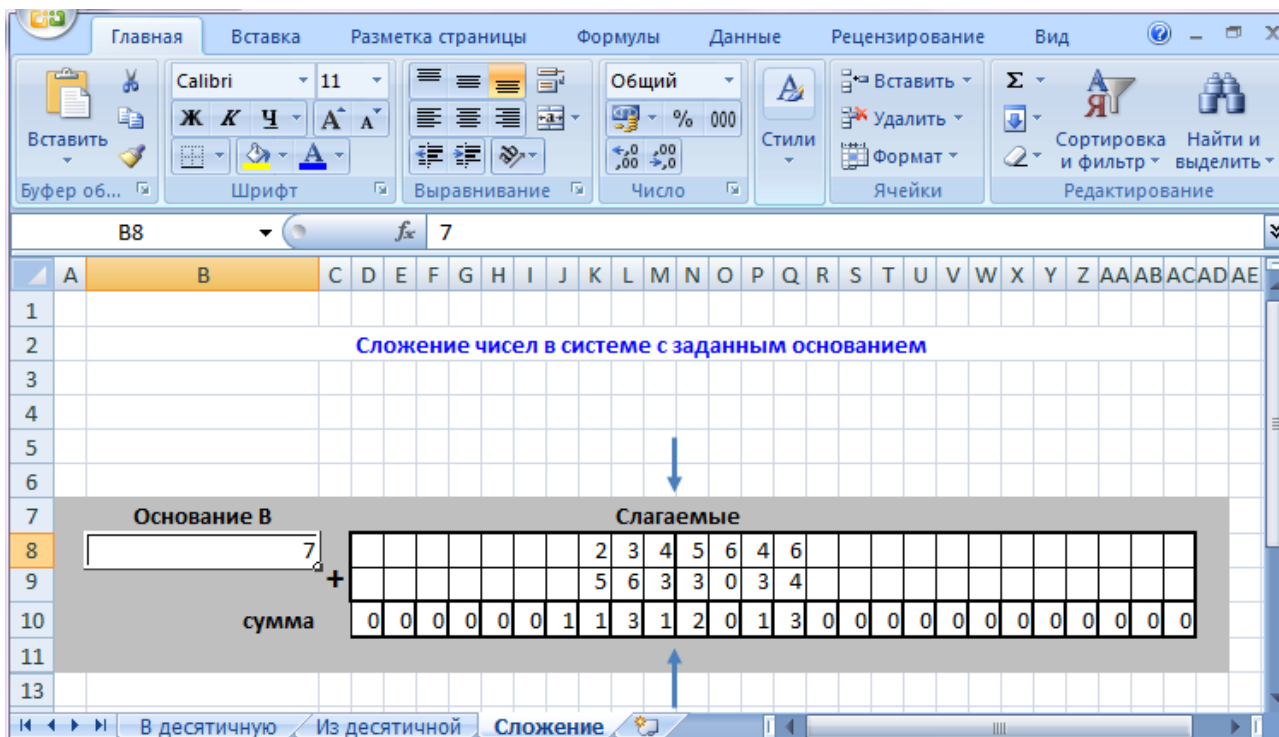


Рисунок 7 – Примерный вид листа «Сложение»

Далее осуществите перевод чисел, согласно Вашего варианта задания. Результаты перевода необходимо внести в табл. 1.

Таблица 1 – Результаты перевода чисел

Число X с основанием B исходной системы	Число X в десятичной системе счисления	Число X с основанием C дублирующей системы
Число Y с основанием B исходной системы	Число Y в десятичной системе счисления	Число Y с основанием C дублирующей системы
Сумма X_B и Y_B с основанием B исходной системы	Сумма X и Y в десятичной системе счисления	Сумма X_C и Y_C с основанием C дублирующей системы
Сумма X_B и Y_B в десятичной системе счисления		Сумма X_C и Y_C в десятичной системе счисления

Варианты индивидуальных заданий

№ вар.	Основание <i>B</i> исходной системы	Число X_B	Число Y_B	Основание <i>C</i> дублирующей системы
20	2	100,0001	100,0111	7
18	2	010,0110	110,1001	9
16	6	432,1414	042,5220	2
14	2	110,0010	111,0110	9
12	9	315,5087	111,5834	2
10	9	073,4821	632,4888	3
8	3	020,1222	221,2020	5
6	16	B42,AB0B	65F,8903	2
4	16	8AB,3F7F	5DB,02D2	4
2	8	452,0632	123,0016	6
1	16	30D,4A89	AAD,DC1E	7
3	2	100,1010	010,0010	7
5	6	300,0414	545,2315	16
7	9	772,4022	706,1470	4
9	8	141,0246	435,3044	2
11	9	855,8063	661,1871	7
13	7	431,0632	222,1003	4
15	6	125,3320	431,1235	5
17	6	012,4012	001,4020	9
19	7	234,5646	563,3034	2
21	16	B61,AB0B	65F,890A	3
22	16	3AB,4F5F	7DB,01B2	2
23	8	456,0332	23,0014	4
24	16	3FD,4A8	AAF,DCF1	8
25	2	100,1110	10,10010	3
26	9	325,5077	111,5734	2
27	9	083,4721	732,4788	3
28	3	022,0222	221,020	5
29	16	E42,AC0B	35F,8903	2
30	16	8AC,3D7F	5DE,0D2	4

1.3 Установка системного и прикладного программного обеспечения

Цель работы: приобретение студентами практических навыков и знаний по основным видам операционных систем, их установке, мобильности программного обеспечения (ПО), по знанию современных средств трассировки и отладки программ, по мониторингу показателей производительности оборудования.

Задание.

- 1) Установить виртуальную машину на свой компьютер.
- 2) Выполнить индивидуальное задание на виртуальной машине.
- 3) Подготовить теоретическую часть по индивидуальному заданию, с кратким описанием ПО, его назначение, функции, задачи, особенности применения, существующие современные версии, достоинства и недостатки заданного ПО.
- 4) Создать презентацию с пошаговым описанием и приведением картинок по установке виртуальной машины, теоретической части вопроса, а так же ход установки заданного ПО.
- 5) Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

Системное программное обеспечение – комплекс программ, которые обеспечивают управление компонентами компьютерной системы, такими как процессор, оперативная память, устройства ввода-вывода, сетевое оборудование, выступая как «межслойный интерфейс», с одной стороны которого аппаратура, а с другой – приложения пользователя [2, 18].

В отличие от прикладного программного обеспечения, системное не решает конкретные практические задачи, а лишь обеспечивает работу других программ, предоставляя им сервисные функции, абстрагирующие детали аппаратной и микропрограммной реализации вычислительной системы, управляет аппаратными ресурсами вычислительной системы.

Отнесение того или иного программного обеспечения к системному условно, и зависит от соглашений, используемых в конкретном контексте. Как правило, к системному программному обеспечению относятся операционные системы, утилиты, системы программирования, системы управления базами данных, широкий класс связующего программного обеспечения.

Операционная система – комплекс системных программ, расширяющий возможности вычислительной системы, а также обеспечивающий управление её ресурсами, загрузку и выполнение прикладных программ, взаимодействие с

пользователями. В большинстве вычислительных систем операционные системы являются основной, наиболее важной (а иногда единственной) частью системного программного обеспечения.

Основные функции операционных систем перечислены ниже.

- Исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
- Загрузка программ в оперативную память и их выполнение.
- Стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).
- Управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
- Управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, оптические диски и др.), организованным в той или иной файловой системе.
- Обеспечение пользовательского интерфейса.
- Сохранение информации об ошибках системы.

К дополнительным функциям относят параллельное или псевдопараллельное выполнение задач (многозадачность), эффективное распределение ресурсов вычислительной системы между процессами, разграничение доступа различных процессов к ресурсам, организация надёжных вычислений, взаимодействие между процессами: обмен данными, взаимная синхронизация, защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений, многопользовательский режим работы и разграничение прав доступа.

Существуют две группы определений операционных систем: «совокупность программ, управляющих оборудованием» и «совокупность программ, управляющих другими программами».

Есть приложения вычислительной техники, для которых операционные системы излишни. Например, встроенные микрокомпьютеры содержатся сегодня во многих бытовых приборах, автомобилях, сотовых телефонах и т. п. Зачастую такой компьютер постоянно исполняет лишь одну программу, запускающуюся по включении. И простые игровые приставки – также представляющие собой специализированные микрокомпьютеры – могут обходиться без операционной системы, запуская при включении программу, записанную на вставленном в устройство «картридже» или компакт-диске. Тем не менее, некоторые микрокомпьютеры и игровые приставки всё же работают под управлением особых собственных операционных систем. В большинстве

случаев, это UNIX-подобные системы.

Встроенные программы или *firmware* – это программы, «зашитые» в цифровые электронные устройства. В ряде случаев (например, BIOS IBM-PC совместимых компьютеров) являются, по сути, частью операционной системы, хранящейся в постоянной памяти. В достаточно простых устройствах вся операционная система может быть встроенной. Многие устройства современных компьютеров имеют собственные «прошивки», осуществляющие управление этими устройствами и упрощающие взаимодействие с ними.

Утилиты – программы, предназначенные для решения узкого круга вспомогательных задач. Иногда утилиты относят к классу сервисного программного обеспечения.

Утилиты используются для мониторинга показателей датчиков и производительности оборудования (например, мониторинга температур процессора или видеоадаптера), управления параметрами оборудования (ограничение максимальной скорости вращения CD-привода; изменение скорости вращения вентиляторов), контроля показателей (проверка ссылочной целостности; правильности записи данных), расширения возможностей (форматирование или переразметка диска с сохранением данных, удаление без возможности восстановления).

Виды утилит следующие: диспетчеры файлов, архиваторы (с возможным сжатием данных), просмотрщики, утилиты для диагностики аппаратного или программного обеспечения, утилиты восстановления после сбоев, оптимизатор диска (вид утилиты для оптимизации размещения файлов на дисковом накопителе, например, путём дефрагментации диска), утилиты – шредеры файлов, деинсталлятор – программа для удаления программного обеспечения, утилиты управления процессами.

Системы программирования – к этой категории относятся системные программы, предназначенные для разработки программного обеспечения:

- ассемблеры – компьютерные программы, осуществляющие преобразование программы в форме исходного текста на языке ассемблера в машинные команды в виде объектного кода;
- трансляторы – программы или технические средства, выполняющее трансляцию программы;
- компиляторы – программы, переводящие текст программы на языке высокого уровня, в эквивалентную программу на машинном языке;
- интерпретаторы – программы (иногда аппаратные средства), анализирующие команды или операторы программы и тут же выполняющие их;
- компоновщики (редакторы связей) – программы, которые производят компоновку – принимают на вход один или несколько объектных модулей и

собирают по ним исполнимый модуль;

- препроцессоры исходных текстов – это компьютерные программы, принимающие данные на входе, и выдающие данные, предназначенные для входа другой программы, например, такой, как компилятор;

- отладчики – модули среды разработки или отдельные программы, предназначенные для поиска ошибок в программах;

- текстовые редакторы – компьютерные программы, предназначенные для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.;

- специализированные редакторы исходных текстов – текстовые редакторы для создания и редактирования исходного кода программ. Специализированный редактор исходных текстов может быть отдельным приложением, или быть встроен в интегрированную среду разработки;

- библиотеки подпрограмм – сборники подпрограмм или объектов, используемых для разработки программного обеспечения;

- редакторы графического интерфейса.

Система управления базами данных (СУБД) – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных [11].

Так как системы управления базами данных не являются обязательным компонентом вычислительной системы, зачастую их не относят к системному программному обеспечению. Часто СУБД осуществляют лишь служебную функцию при работе других видов программ (веб-серверы, серверы приложений), поэтому их не всегда можно отнести к прикладному программному обеспечению. Поэтому СУБД иногда относят к промежуточному программному обеспечению (Middleware).

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;

- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;

- поддержка языков баз данных (язык определения данных, язык манипулирования данными).

Классификация СУБД по способу доступа к базе данных следующая.

- Файл-серверные, в которых файлы данных располагаются централизованно на файл-сервере, а программная реализация СУБД располагается на каждом клиентском компьютере целиком. Доступ к данным

осуществляется через локальную сеть.

- Клиент-серверные СУБД состоят из клиентской части (которая входит в состав прикладной программы) и сервера.

- Встраиваемые – программные библиотеки, которые позволяют унифицированным образом хранить большие объёмы данных на локальной машине.

Варианты индивидуальных заданий

№	Задание
1	Проверка и диагностика жесткого диска с помощью утилит.
2	Настройка подключения к Интернету через роутер.
3	Проверка и диагностика жесткого диска сторонними программами.
4	Работа и настройка почтового клиента.
5	Работа с iTunes.
6	Работа с утилитами для диагностики аппаратного или программного
7	Работа с программой для torrent-сетей.
8	Работа с утилитой мониторинга температуры процессора.
9	Установка драйверов для Windows 7 вручную.
10	Установка Open Office.
11	Установка Micro-Cap 11 Evaluation.
12	Установка и настройка сетевой игры.
13	Установка драйверов и настройка принтера.
14	Форматирование или переразметка диска с сохранением данных.
15	Установка и настройка MySQL.
16	Установка антивируса Касперского.
17	Установка антивируса Avast.
18	Установка операционной системы Windows 8.
19	Установка MS Access.
20	Установка операционной системы Linux.
21	Установка текстовых процессоров.
22	Установка электронных таблиц.
23	Установка графических редакторов.
24	Установка просмотрщика изображений.
25	Установка медиаплеера.
26	Установка просмотрщика pdf-файлов.
27	Установка переводчика текстов.

28	Установка домашнего видеоредактора (видеомонтаж).
29	Установка редактора трехмерной графики.
30	Установка электронной энциклопедии современной техники.

Правила оформления презентаций с использованием Ms PowerPoint

Доклад – это публикация вашей работы. Опыт подготовки доклада окажется вам полезным на протяжении всего учебного процесса. Он будет развиваться и совершенствоваться во время вашего участия в студенческих научных конференциях, при подготовке и защите курсовых и выпускной квалификационной работы [8, 10].

Подготовка к докладу начинается с написания текста доклада (выступления). Ваш доклад – это реферативное сообщение, в котором нужно кратко раскрыть теоретическую сущность и прикладное значение изученного вопроса. Такой доклад можно рассматривать как короткую лекцию по заданной теме. Поэтому рекомендуется придерживаться достаточно простого и ясного плана доклада, который включает следующие части.

- 1) Тема доклада. Постановка задачи, ее научное и практическое значение.
- 2) Современное состояние вопроса.
- 3) Цель доклада.
- 4) Источники информации, привлеченные для изучения вопроса.
- 5) Основная, содержательная часть сообщения по теме доклада.
- 6) Заключение и выводы.
- 7) Личный вклад автора.

Доклады на 5-7 минут должны быть подготовлены с применением программы для представления презентаций Ms PowerPoint, с включением различных видов настройки анимации.

Первая страница презентаций должна содержать название доклада, название вуза, факультета и кафедры, ФИО автора и ФИО преподавателя.

Как и весь доклад в целом, презентация должна иметь четкую структуру, внутренние озаглавленные разделы (подразделы).

Иллюстрации к докладу (слайды) во время выступления служат вашим путеводителем. Они должны быть яркими, лаконичными и легко воспринимаемыми, количество надписей на них - минимальным.

На слайде необходимо поместить только самое необходимое. Не пишите длинных подрисуночных подписей и определений, пользуйтесь общеизвестными сокращениями. Помните, что картинка показывается на экране короткое время и восприятие помещенной на ней информации должно быть быстрым.

Особое внимание уделите разделу заключений и выводов - концентрату всей работы. Обычно он содержит заключения сразу двух видов.

Заключение первого вида - *констатирующего* - служит логическим замыканием постановочной части доклада. В нем вы показываете, что поставленные задачи решены, и цель работы достигнута. Здесь можно подчеркнуть те особенности методики вашей работы, которые обеспечили ее успех и позволили получить новые (для вас) знания.

Заключение второго вида - *результативно-аналитическое*. В нем вы перечисляете и комментируете результаты работы, их научное и практическое значение, оцениваете важность полученных сведений для вашей учебной специальности. Отдельные фразы из предыдущих разделов доклада в Заключении можно повторять дословно.

Приведите в конце текста доклада и в конце презентаций список литературы (библиографию), включая ссылки на использованные электронные ресурсы.

Последний этап работы над текстом доклада состоит в том, чтобы выкинуть все лишние слова, повторения, упростить длинные фразы, расставить знаки препинания, вымести «словесный мусор».

Во время выступления не превышайте установленного для доклада отрезка времени. Всегда строго соблюдайте регламент. Ничто так не раздражает, как докладчик, не соблюдающий регламент. Как правило, ведущий занятия преподаватель по истечении отведенного на доклад времени или предлагает одну – две минуты для его завершения, или сразу останавливает доклад и переходит к следующим вопросам.

Выступление рекомендуется завершать выводами: *«следовательно...»*, *«таким образом...»*. Заключительная фраза выступления почти стандартна: *«Мой доклад закончен, благодарю за внимание»*.

ЛАБОРАТОРНЫЕ РАБОТЫ 2 СЕМЕСТР

1.4 Изучение программы визуального моделирования Micro-Cap


Цель работы: освоение PSpice – технологии (симуляция аналоговой и цифровой логики), на примере программы визуального моделирования Micro-Cap, с целью проведения анализа электромагнитных процессов в энергетических системах широкого применения.

Задание.

- 1) Изучить интерфейс программного продукта *MicroCap*.
- 2) По методическим указаниям разработать простейшие схемы электрических цепей и провести их анализ.
- 3) Придумать схему цепи из не менее 6 элементов, задать параметры, провести их анализ (**разрешено группироваться по 5 студентов на одну схему**).
- 4) Продемонстрировать проект преподавателю и защитить работу. **При этом защищается каждый студент индивидуально, меняя параметры в заданной цепи у элементов, с пояснением полученных данных проведенного анализа.**

Краткая теория и методические указания

Программа *MicroCap 9.0 5.0 Evaluation version* является свободно распространяемой демоверсией профессиональной программы машинного моделирования электронных схем (www.spectrum-soft.com), но она обладает практически всеми качественными возможностями полнофункциональной, а ограничения носят по большей части количественный характер (демоверсия позволяет моделировать схемы, число компонентов в которых не превышает 50, расчеты ряда схем проходят несколько медленнее, чем в полнофункциональной версии, ограничена библиотека компонентов, нет встроенной программы подготовки собственных моделей и некоторых других дополнительных функций) [17].

В программу вставлен достаточно подробный раздел HELP , а на сайте разработчика можно получить дополнительные материалы, например файл «DemoRead.doc», дополнительная литература по MicroCap [1]. Интерфейс программного продукта представлен на рис. 20.

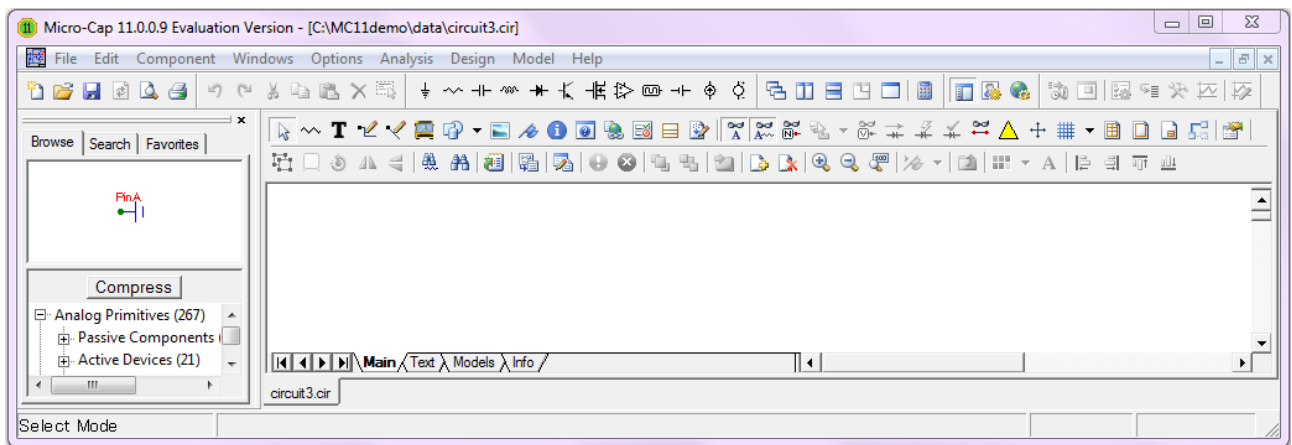


Рисунок 20 – Интерфейс Micro Cap

Программный продукт позволяет начать моделирование электрических цепей новичку даже без глубокого ее изучения. Для наших целей нет необходимости досконального изучения программы, поэтому знакомство с необходимыми функциями мы будем осуществлять непосредственно при выполнении конкретных заданий.

Интерфейс программы является стандартным для программ ОС Windows. Как обычно, все команды можно вызвать через меню, часть наиболее употребимых выведена на инструментальные панели в виде ярлычков (пиктограмм). Назначение стандартных пиктограмм (📁, 💾, и т. п.) не рассматриваем, т. к. они достаточно хорошо известны даже неопытному пользователю.

Пользователь составляет электрическую цепь непосредственно в удобном графическом редакторе (*Circuit editor*), затем задаёт параметры анализа цепи (*Analysis*) и изучает графики с данными.

Программа автоматически составляет уравнения для данной цепи и производит их математический расчёт.


При загрузке программы появляется главное окно *MAIN*, готовое для рисования электрической схемы в новом файле, получающим название по умолчанию *circuit1.cir*.

Выпадающие заметки дня (*Tip of the day*) можно убрать после ознакомления.

В разделе *FAIL* мы видим обычные для ОС Windows команды для работы с файлами.

Созданные файлы электрических цепей мы будем сохранять в типе Schematic (*.cir). Остальные команды нас пока не интересуют.

В окне *EDIT* нам нужна команда *Copy to Clipboard* с её четырьмя возможностями сохранения видимого окна или его части. Наиболее востребовано окно *Component*. По сути это большая библиотека элементов

электрической цепи. В разделе *Analog primitives/Passive Components* мы легко находим три наших главных аналоговых линейных элемента: резистор R , конденсатор C , индуктивность L . На первой инструментальной линейке также можно найти их условно-графические изображения (УГО) в виде пиктограмм, которое также будет появляться в небольшом вспомогательном окне при вызове элемента из базы. Чтобы внести УГО этих элементов (а также, в дальнейшем, и других) в составляемую электрическую цепь, нужно находиться в *Component mode* (кнопка , открывается автоматически с главным окном).

Кликнув курсором на обозначении выбранного элемента, мы переводим его в графическое окно, при этом обозначение курсора примет вид УГО выбранного элемента. Установив элемент в нужном месте графического окна и кликнув в этом месте левой кнопкой мышки мы вызовем окно установки его параметров. Необходимо отметить, что клики правой кнопкой мышки при нажатой левой поворачивают УГО на 90° по часовой оси.

При установке УГО элемента на место открывается довольно обширное окно параметров этого элемента. Но для первых шагов нам потребуется установить в строке *PART* обозначение элемента в цепи (обычно это делается автоматически с каждым вводом подобного элемента) его номинал. Все значения для выделяемых параметров элемента вводятся в окне *Value*. Так номинал элемента вводится в окне *Value* при выделенной строке с названием элемента и затем дублируется в этой строке через знак равенства.

Значения компонентов задаются либо непосредственно (2600), либо в показательной форме (2.6E3), либо условными буквенными обозначениями (2.6K).

Следует обратить внимание, что в MicroCap:

- целая часть чисел отделяется от дробной не запятой, а точкой. Например, 1.3K или 1.3E3;
- буквенные обозначения следует вводить в английском алфавите.

При желании можно выбрать вид УГО в окне *Value* при выделенной строке *Shapigroup*, поскольку по умолчанию вызываются УГО по американским стандартам. Для этого нужно просто вызвать в окне *Value* возможные УГО. Особенно это рекомендуется при вводе УГО резистора, где выбираем *euro* вместо *default*.

Остальные параметры нас пока не интересуют. При нажатии ОК окно параметров исчезает и элемент со своими атрибутами оказывается в окне редактора.

Задание 1: попробуйте ввести элементы $R=1\text{кОм}$, конденсатор $C=1\text{мкФ}$, индуктивность $L=1\text{мГн}$ (рис. 21).

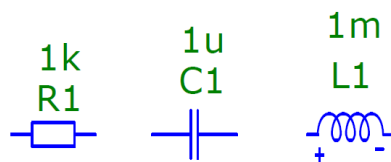





Рисунок 21 – Базовые элементы

Отметим, что рядом с УГО элемента появились и его атрибуты. Этого можно избежать, если при введении параметра отменить его показ (снять флажок *show* рядом с окном *Value*). Перетаскивать УГО и его атрибуты можно нажатой левой кнопкой мышки, но для этого следует перейти в режим *Select mode*, кнопка .

Соединения элементов в цепь производятся с помощью проводников, которые вызываются через кнопки с их пиктограммами (кнопки  и  на второй инструментальной панели).

Задание 2: составьте простейшую схему с параллельным соединением 2 элементов (рис. 22).

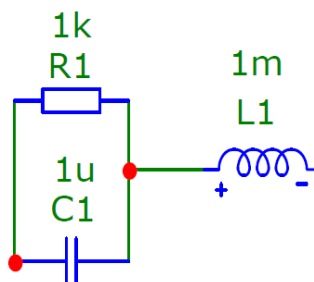


Рисунок 22 – Соединение двух элементов

Отметим что:

- пересечение проводников с замыканием обозначается красной точкой;
- при перемещении элементов проводники не отсоединяются.

Для анализа электрической цепи нам нужно знать, как изменяются её энергетические характеристики. Но для этого в цепь нужно вставить источник энергии (сигналов).

В окне *Component/Analog primitives* можно найти 5 групп источников.

Для знакомства выберем такой хорошо известный источник постоянного тока, как батарейка. Для этого находим *Component/Analog primitives/Waveform Sources/Battery* или на инструментальной панели соответствующую пиктограмму и переносим этот элемент в окно редактора. В окне параметров нужно ввести его значение в Вольтах и желательно изменить УГО на *euro*.

Задание 3: введите электрическую цепь (схему, рис. 23) с источником постоянного тока 5В и тремя резисторами по 1кОм, два из которых включены параллельно.

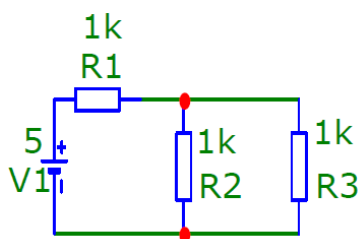


Рисунок 23 – Электрическая цепь

Для проведения анализа нам необходимо разметить схему.

Прежде всего, нужно обозначить узел с потенциалом, равным 0, т.е. обозначить точку заземления (рис. 24). Для этого на инструментальной панели находим пиктограмму заземления и подключаем этот элемент к выбранному узлу схемы. Тоже самое можно сделать через окно *Component/Analog primitives/Connectors/Ground*.

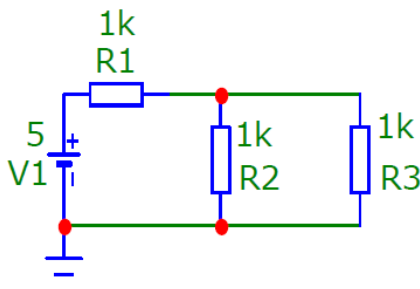


Рисунок 24 – Заземление схемы

Далее выведем номера узлов (рис. 25). Для этого включаем кнопку

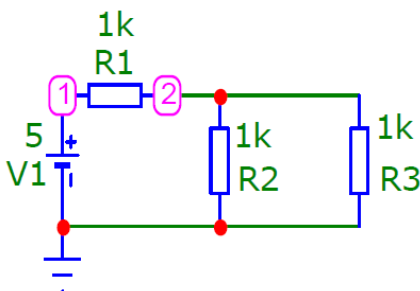


Рисунок 25 – Вывод узлов

А теперь можно приступать к анализу. В окне *Analysis* можно увидеть много видов анализа электрической цепи.

- Transient Analysis...(Alt+1) - анализ переходных процессов.
- AC Analysis...(Alt+2) - анализ частотных характеристик.
- DC Analysis...(Alt+3) - анализ передаточных функций по постоянному току.
- Probe Transient Analysis... - анализ переходных процессов и отображение их результатов в режиме Probe.
- Probe AC Analysis... - Анализ частотных характеристик и отображение

их результатов в режиме Probe.

– Probe DC Analysis... - Анализ передаточных функций по постоянному току и отображение их результатов в режиме Probe.

Для первого знакомства выберем анализ по постоянному току (DC...).

В режиме DC... рассчитываются передаточные характеристики по постоянному току. К входам цепи подключаются один или два независимых источника постоянного напряжения или тока. В качестве выходного сигнала может рассматриваться разность узловых потенциалов или ток через ветвь, в которую включен резистор. При расчете режима DC... - программа закорачивает индуктивности, исключает из схемы конденсаторы и затем рассчитывает режим по постоянному току при нескольких значениях входных сигналов.

После перехода в режим DC... программа проверяет правильность построения схемы. При отсутствии ошибок составляется топологическое описание, выполняется подготовка к численному расчету нелинейных уравнений итерационным методом Ньютона-Рафсона и открывается окно задания параметров моделирования *DC Analysis Limits*.

Прежде, чем начинать любой анализ, не поленитесь оценить поведение схемы в грубых приближениях. Это позволит провести анализ цепи более осмысленно и выявить те грубые ошибки, которые могут быть допущены, но не выявлены программой.

В нашей схеме мы видим два делителя:




– **напряжения** в отношении 1:3 между резисторами $R1=1k$ и параллельно соединёнными резисторами $R2, R3$ с общим сопротивлением $R0=R1*R2/R1+R2=0.5k$; при этом напряжение узла 2 можно найти по правилу делителя напряжения $V2= V1*R0/R1+R0$. Для нашей схемы при $V1=10V$ напряжение $V2=10/3=3,3$.

– **тока** в отношении 1:2 между резисторами $R1$ и $R2$ или $R3$; при этом ток в ветви элемента $R2(R3)$ можно найти из правила делителя тока $I2*R2=I3*R3=V2$. Отсюда $I1= 6,6/1000=6,6$ мА, а $I2=6,6/2= 3,3$ мА.

Итак, кликнув левой кнопкой мышки команду *Analysis/DC...* мы вызовем очень мощное и нужное диалоговое окно *DC Analysis Limits*. Познакомимся с ним подробно.

В этом окне можно установить пределы изменения переменных, вид выводимых графиков и собственно команду *Run*, которая позволяет машине начать анализ. Всё окно разделено на пять областей: кнопки управления, числовые значения, представление графиков, выражения и дополнительные функции (опции).

Кнопки управления

Run: По этой команде начинается анализ схемы. Эту команду можно вызвать также через кнопку F2 на клавиатуре или кликнуть кнопку  на появляющейся вместе с графиками новой инструментальной панели. Моделирование может быть остановлено в любой момент времени нажатием на пиктограмму  или клавишу Esc. Последовательные нажатия на пиктограмму  прерывают и затем продолжают моделирование.

Add: добавление еще одной строки спецификации вывода результатов после строки, отмеченной курсором. На этой строке устанавливается способ отображения результатов и аналитические выражения для построения графиков. При наличии большого количества строк, не уместяющихся на экране, появляется линейка прокрутки.

Delete: удаление выделенной курсором строки в поле представления графиков.

Expand: по этой команде выводится окно, позволяющее расширить поле записей в области представления графиков. Для этого курсором нужно выделить соответствующее текстовое поле.

Stepping: по этой команде переходят в диалоговое окно *Stepping*, которое позволяет выводить данные для нескольких пошагово меняющихся значений элементов схемы.

Properties: по этой команде переходят в диалоговое окно *Properties dialog box*, которое позволяет управлять окном вывода графиков и видом самих графиков.

Help: по этой команде переходят к файлам Help topic для диалогового окна DC Analysis Limits.

Числовые параметры

Variable 1 – задание первой изменяемой переменной. В качестве переменной используется напряжение источника постоянного тока вне зависимости от его первоначально установленных параметров. Таких источников в схеме может быть 2. В графе *Method* выбирается метод изменения напряжения источника (переменной).

– *Auto* – шаг расчёта выбирается автоматически с целью достижения изменения выходного параметра (в %) не более, чем указано в позиции *Maximum Change*. Пределы расчёта (<конец>, <начало>) по умолчанию устанавливаются 10, 0, но можно установить свои. Это очень полезно при непредсказуемо изменяющихся выходных значениях, так как позволяет рисовать плавные аналоговые кривые.

– *Linear* – линейный масштаб, задаваемый в графе *Range* по формату <конец>, <начало>, <шаг>. Если опустить параметр *Step* (шаг), то шаг будет принят равным (конец-начало)/50. Если опустить параметр <начало>, то начальное значение будет приравнено к нулю.

– *Log u List* – в демоверсии не работает.

В графе *Name* указывает имя варьируемой переменной. Это могут быть: величины источника постоянного напряжения или тока; температура; значения одного из параметров модели компонентов, имеющих математические модели; значения символической переменной (определенной директивой *.Define*). Имя варьируемой переменной может выбираться из вложенного в окно списка.

В графе *Range*. Указывает диапазон изменения варьируемой переменной и шаг, зависящий от метода изменения переменной.

Строка **Variable 2** определяет поля *Method*, *Name* и *Range* для второй варьируемой переменной. Для значений, указываемых в этих полях, используются те же правила, что и перечисленные выше для переменной *Variable 1*, за исключением опций в списке *Method*. Здесь исключена опция *Auto*, но появилась и дополнительная опция *None*, выбираемая в том случае, если изменяется только одна переменная. По умолчанию шаг *Step* принимается равным (<конец> - <начало>)/10. Каждое значение переменной *Variable 2* приводит к построению отдельного графика. Метод двух источников очень удобен для построения семейства вольтамперных характеристик транзисторов.

Temperature - диапазон изменения температуры в градусах Цельсия.

Number of Points - количество точек данных, по которым осуществляется интерполяция при построении графиков, или количество строк в таблице вывода результатов (*numeric output*). По умолчанию устанавливается равным 51. Нас пока это вполне устраивает.

Maximum change, %. Действует только при выборе метода *Auto* изменения переменной. Представляет собой максимально допустимое приращение графика первой функции на одном шаге (в процентах от полной шкалы). Если график функции изменяется быстрее, то шаг приращения первой переменной автоматически уменьшается.

Опции окна DC Analysis Limits

Run Options - управление выдачей результатов расчетов. Нас вполне удовлетворит появляющееся по умолчанию значение *Normal* – при этом результаты расчетов не будут сохраняться на диске.







Auto Scale Ranges - присвоение признака автоматического масштабирования *Auto* по осям X, Y для каждого нового варианта расчетов. Если эта опция выключена, то принимаются во внимание масштабы, указанные в графах X

Range, Y Range.

Accumulate plots – позволяет аккумулировать результаты расчётов на одном графике при редактировании схемы (нам пока не потребуется).

Параметры вывода результатов моделирования

Слева внизу мы видим группу пиктограмм. Нажатие каждой пиктограммы определяет характер вывода данных, задаваемых в той же строке. Имеются следующие возможности.

-   переключение между логарифмической и линейной шкалой по оси X. При выборе логарифмической шкалы диапазон изменения переменной должен быть положительным.
-   переключение между логарифмической и линейной шкалой по оси Y. При выборе логарифмической шкалы диапазон изменения переменной должен быть положительным.
-  вызов меню для выбора одного из 64 цветов для окрашивания графиков. График окрашивается в цвет кнопки. Очень удобно, когда вы выводите на один график несколько параметров.
-  при нажатии этой кнопки в текстовый выходной файл заносится таблица отсчетов функции, заданной в графе Y Expression (нам пока не надо). Число строк в таблице задается параметром *Number of Points* в разделе «Числовые параметры». Нам это пока не понадобится.

Page – указываются номера (наименование) окон (страниц), на которые выводятся графики.

Plot Group - в графе *P* числом от 1 до 9 указывается номер графического окна, в котором должна быть построена данная функция. Все функции, помеченные одним и тем же номером, выводятся в одном окне (на одной странице). Если это поле пусто, график функции не строится.

Форматы выражений для DC-анализа

Поля *X Expression* (математическое выражение переменной, откладываемой по оси X) и *Y Expression* (математическое выражение переменной, откладываемой по оси Y) используются для спецификации масштабов и переменных, откладываемых по горизонтальной (X) и вертикальной (Y) осям.

Программа даёт очень широкий выбор возможностей представления переменных, откладываемых по осям графиков: от конкретных токов и напряжений до сложных формул с их участием. В этом можно убедиться, кликнув правой кнопкой мыши в этих полях. Но пока ограничимся выводом на оси X, Y напряжений в узлах – v (<номер узла>) или между узлами – v (<номер одного узла>, <номер другого узла>), а также токов в ветвях (между узлами) – i

(<номер одного узла>, <номер другого узла>). По оси X по умолчанию выводится напряжение источника, которое, как мы знаем, в DC...- анализе является переменной величиной *Variable 1*.

Поля *X Range* (масштаб по горизонтальной оси X) и *Y Range* (масштаб по вертикальной оси Y) имеют несколько возможностей, которые можно вызвать, кликнув правой кнопкой мышки в одном из этих полей.

Auto - приводит к однократному автоматическому масштабированию по соответствующей оси и заполнения полей полученными значениями масштабов.

Установка флага *Auto Scale Ranges* приводит к автоматическому расчету масштабов всех графиков по всем осям при каждом повторении расчетов и соответствующему обновлению полей *X Range* и *Y Range*.

Auto always - приводит к постоянному автоматическому масштабированию по соответствующей оси.

Можно ввести свои пределы значений переменных по осям в формате <high>, <low>. <low> по умолчанию устанавливается в нулевое значение.

Меню режимов расчета передаточных функций DC

После перехода в режим расчета передаточных функций в строке меню появляется новое меню DC, содержащее пункты RUN, Limits, Stepping, Exit, State Variables Editor, OPTIMIZE, Watch, Breakpoints, 3D Windows, Numeric Output, Reduce Data Points. Состав этих команд одинаков для всех видов анализа.

Вернёмся к нашей схеме. Выбрав режим расчёта передаточных функций **DC...**, в окне **DC Analysis Limits** определим, что переменная **Variable 1** будет меняться в режиме *Auto* и это будет напряжение в узле 1 ($V(1)$), т. е. напряжение источника. Оно будет меняться в пределах от 0 до 10В с шагом 0,5В. Второй переменной у нас нет, т.к. нет второго источника. Изменение температуры нас не интересует, также как и 5% в **Maximum change** нас вполне устроит. В *Run Options* нас также устроит значение Normal.

Выведем 4 графика: для напряжений в узлах 1 и 2 $\{v(1), v(2)\}$ и токов через резисторы R1, R2 $\{i(1,2)$ или $i(R1)$, $i(2,0)$ или $i(R2)\}$. Это мы укажем в разделе *Expression*. По оси X (*X Expression*) выведем изменяющееся напряжение источника (DCINPUT1 или $v(1)$).

Все графики выполняем в линейном масштабе по осям X и Y. Желательно придать им разные цвета. Предлагается вывести графики напряжения на первый лист (page 1), а графики токов на второй лист (page 2). При этом на каждом листе обе зависимости расположить на одном графике (в разделе P везде указать 1).

В разделах *X Range* и *Y Range* везде укажем *Auto always*, т. к. другие варианты нас пока не интересуют.


Проверив ещё раз введённые данные в окне **DC Analysis Limits** кликаем *Run*.

На экране открывается окно (рис. 26) с графиком зависимости напряжений $v(1)$, $v(2)$ от $v(1)$.

На нижней линейке можно увидеть значки 1 и 2. Это наши страницы.

Переходим на страницу 2 и получаем (рис. 27) графики зависимости токов $i(1,2)$ и $i(R2)$ от $v(1)$.

Простое рассмотрение полученных графиков подтверждает наши расчёты. Подводя курсор к любой точке графика в выпадающем окошке можно увидеть числовые данные, относящиеся к данной точке.

Обратите внимание на появившуюся вторую инструментальную линейку, которая содержит очень много функций для работы с графиками. Например, очень полезной является функция курсора  (можно также вызвать клавишей F8). Она позволяет вызвать окно с предельными числовыми значениями, а также кликнув левую и правую кнопки мышки можно вызвать левый и правый динамическую курсорную вертикаль.

Они очень удобно передвигаются при нажатой соответствующей кнопке мышки. При этом соответственно меняются показания в разделе числовых данных (рис. 27).

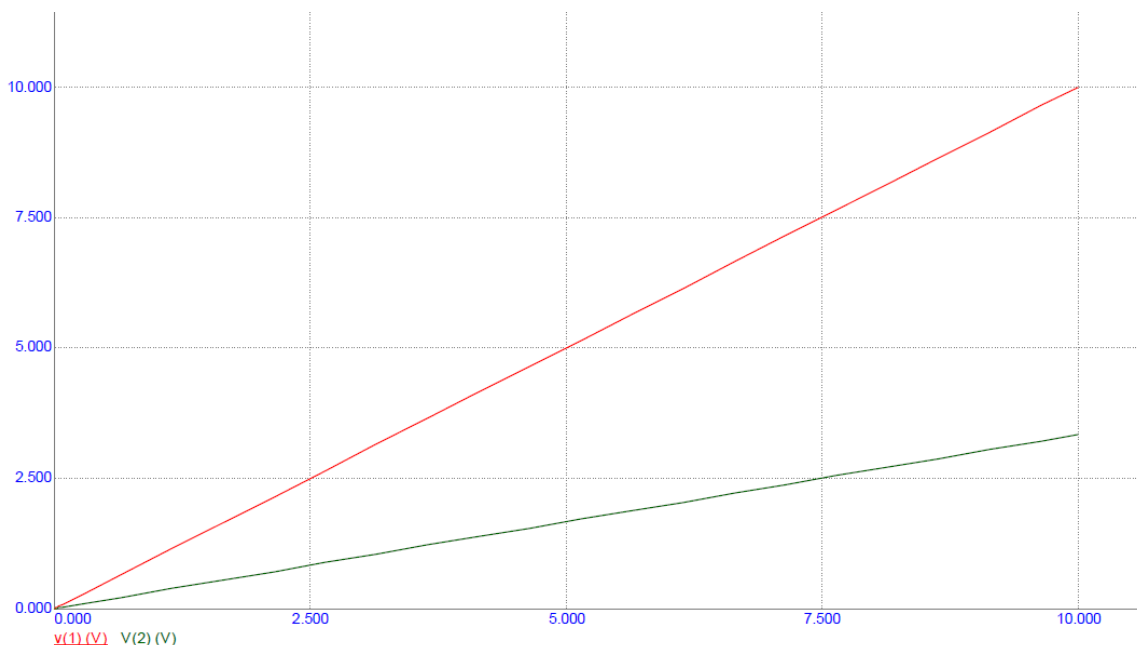


Рисунок 26 – График зависимости напряжений

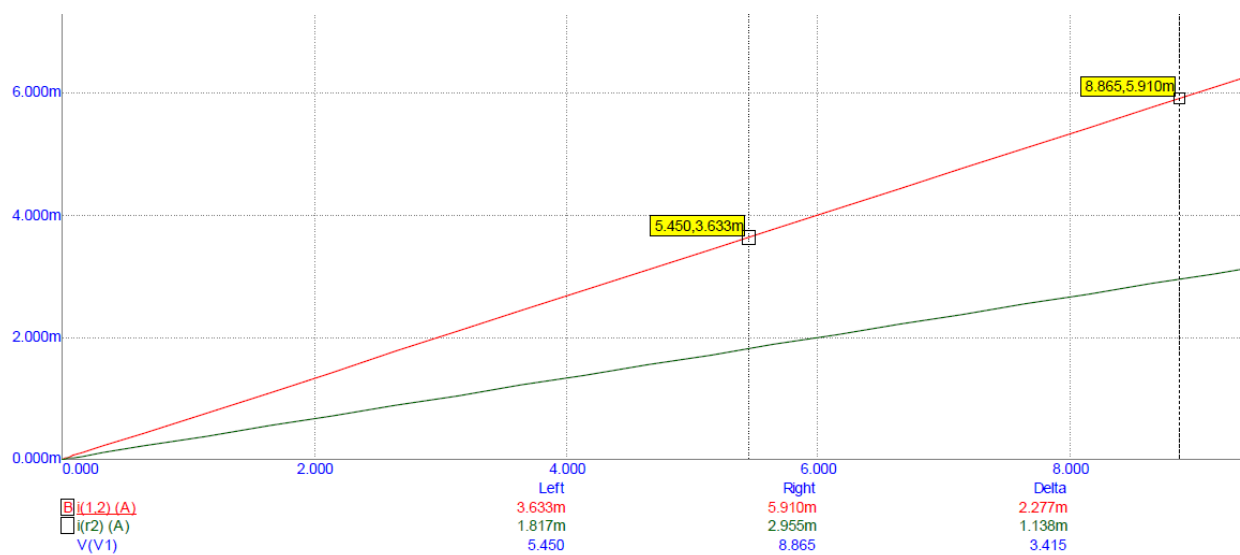


Рисунок 27 – График зависимости токов

1.5 Работа с проектами в среде Microsoft Visual Studio и использование визуальных компонент при программировании приложений

Цель работы: научиться работать с проектами в среде Microsoft Visual Studio и использовать визуальные компоненты при программировании приложений.

Задание.

1) Написать программу калькулятор... (с действиями по варианту). **Разрешено группироваться в одну разработку студентам с одинаковыми действиями для калькулятора, согласно варианта задания.**

2) Разработать тесты для проверки программного продукта (ввод символов, букв, чисел, запятых, точек, множества полей, деление на ноль и т.д.).

3) Продемонстрировать проект преподавателю и защитить работу. **При этом защищается каждый студент индивидуально, внося при преподавателе изменения в программу (смена цвета формы, смена размера кнопок, смена расположения кнопок, добавление нового элемента на форму и т.д.).** Задание на внесение конкретных изменений указывает преподаватель.

Краткая теория и методические указания

В современных средах программирования реализована новая парадигма - объектный подход, что позволило резко повысить производительность труда программистов. Подход был основан на понятии объекта, типа данных, в котором сочетаются как свойства, сгруппированные данные (пример – поля в записи), так и методы их обработки (подпрограммы). Фактически объект стал отражать реальные и даже абстрактные понятия окружающего мира. Благодаря этому теперь удастся выполнить проектирование программ, основываясь на понятии объекта, что значительно проще и быстрее, чем раньше. Работать с привычными понятиями человеку легче, нежели с абстрактными числами. При этом специалистам удалось выделить большой набор объектов, которые нужны при создании самых разных программ. Эти объекты используются повторно без расходования времени на их программирование [4].

Данная лабораторная работа предназначена для обретения первичных навыков при работе в среде программирования Microsoft Visual Studio с использованием языка программирования высокого уровня C#. Этот язык как средство обучения программированию обладает рядом достоинств. Он хорошо организован, строг, большинство его конструкций логичны и понятны, а развитые средства диагностики и редактирования кода делают процесс программирования эффективным.

Среда разработки Microsoft Visual Studio (Visual Studio.NET) предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки. Корпорация Microsoft включила в платформу средства разработки для четырех языков: Visual C#, Visual J#, Visual Basic и Visual C++. Платформа .NET является открытой средой. Это означает, что компиляторы для нее могут поставляться и сторонними разработчиками. К настоящему времени разработаны десятки компиляторов для .NET, например, Ada, Perl, Delphi, Lisp и другие. Все .NET-совместимые языки должны отвечать требованиям CLS (Common Language Specification – общезыковая спецификация), в которой описывается набор общих для всех них характеристик. Это позволяет использовать для разработки приложения несколько языков программирования и вести полноценную межъязыковую отладку. Все программы, созданные в Visual Studio.NET, независимо от языка программирования, на котором они написаны, используют одни и те же базовые классы библиотеки .NET.

Приложение в процессе разработки называется проектом. Проект включает в себя все необходимое для создания приложения: файлы, папки, ссылки и прочие ресурсы. Среда Visual Studio.NET позволяет создавать проекты различных типов, таких как:

- 1) windows-приложение использует элементы интерфейса ОС Windows, включая формы, кнопки, флажки и т.п.;
- 2) консольное приложение;
- 3) библиотеки классов. Они объединяют в себе классы, которые предназначены для использования в других приложениях;
- 4) web-приложение. Это приложение, доступ к которому осуществляется через браузер, и которое по запросу формирует web-страницу и отправляет ее клиенту по сети;
- 5) web-сервис. Это компонент, методы которого могут вызываться через Интернет.

Несколько проектов можно объединить в решение (solution), что облегчает их совместную разработку. Скомпилированная программа будет выполняться ОС Windows, только если будет установлена исполняющая среда .NET (dotnetfx.exe). Этот пакет поставляется совместно с Visual Studio и его можно установить в любой версии ОС Windows. Исполняемая среда контролирует исполнение программ (управляемый код), сглаживая некоторые ошибки, наличие которых в обычных программах могло бы вызвать крах всего приложения. Единственным языком, с помощью которого можно создавать и управляемый код и обычные приложения для Windows (неуправляемый код), является язык Visual C++. Несмотря на присутствие в его названии слова Visual

при разработке программ он не позволяет использовать визуальные компоненты, что замедляет процесс разработки прикладных программ. Остальные три языка, входящие в состав пакета, при разработке программ позволяют использовать готовые визуальные компоненты.

После запуска Visual Studio на экране возникает окно, стартовая страница (Start page) которого содержит список недавно использовавшихся проектов (Recent Projects). Щелчок левой кнопкой мыши по названию одного из проектов в этом списке приведет к его открытию. Также открыть проект можно при помощи меню, выбрав команду File→Open→Project/Solution. Для создания нового проекта следует выбрать в меню пункт File→New→Project. При создании нового проекта появится окно New Project, изображенное на рис. 15.

В древовидном списке слева Project Types (типы проектов) можно выбрать проекты для четырех основных языков программирования, представленных в Visual Studio. При выборе одного из видов проекта в правом списке Templates (Шаблоны) открывается перечень шаблонов, которые могут быть использованы мастером для создания проекта с минимальной функциональностью. Ниже списков в поле Name (Имя) следует ввести имя проекта. В каталоге, который следует указать в комбинированном списке Location (Расположение), будет создан еще один каталог с именем проекта, в котором и расположатся все его файлы. Интегрированная среда Visual Studio оперирует таким понятием как рабочее пространство – Solution (буквально «решение»). Рабочее пространство может быть пустым, но может и содержать несколько проектов.

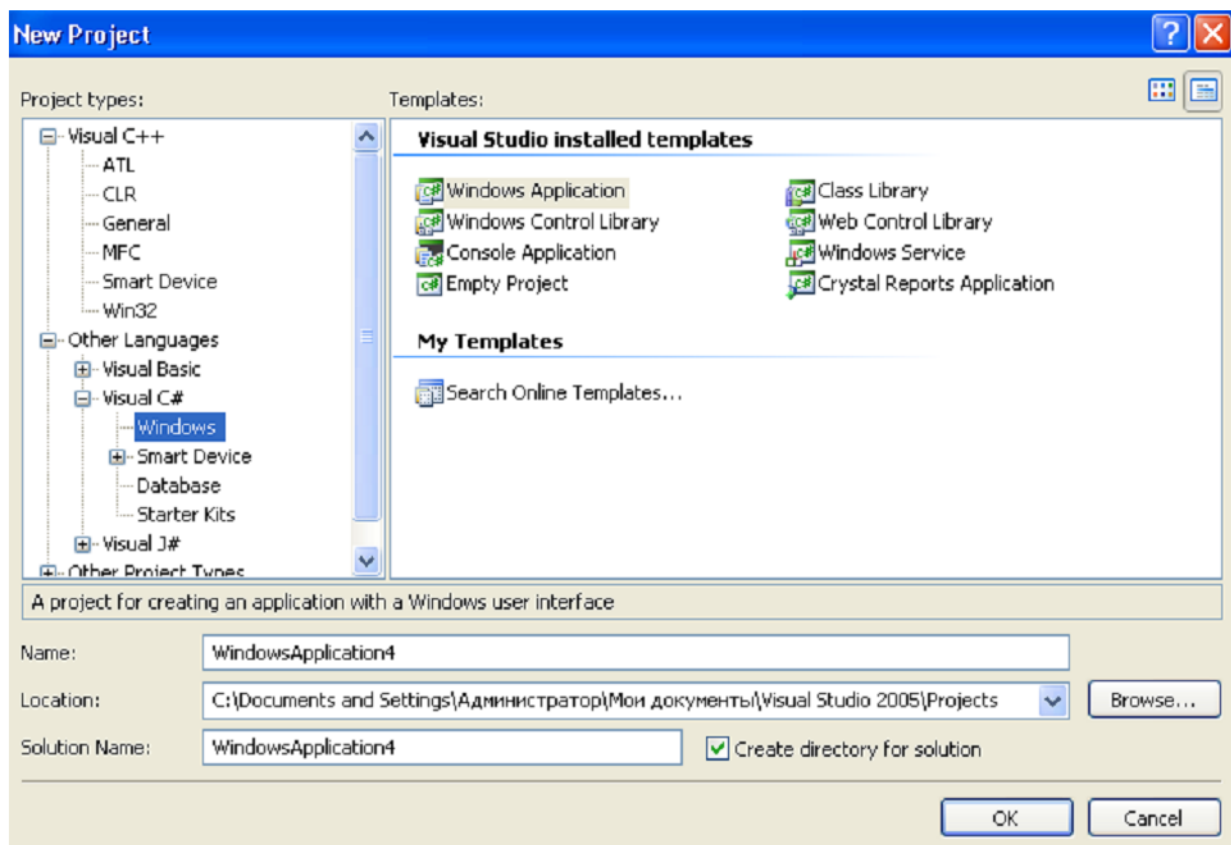


Рисунок 15 – Внешний вид окна «Новый проект»

Обычно рабочее пространство создается вместе с проектом и помещается в один каталог. Имя рабочего пространства задается в поле Solution Name, для его изменения необходимо чтобы был отмечен флаг Create directory for solution (создать каталог для рабочего пространства). При указании имени рабочего пространства, отличного от имени проекта, в каталоге, имя которого записано в поле Location, сначала будет создан подкаталог для рабочего пространства, а в нем подкаталог для проекта. Информация о рабочем пространстве помещается в файл с расширением *.sln, а информация о проекте – в файл с расширением *.vcproj.

В основу ОС Windows положен принцип событийного управления. Это означает, что и сама ОС, и приложения после запуска ожидают действия пользователя и реагируют на них заранее заданным образом. Любое действие пользователя (нажатие клавиши на клавиатуре, щелчок кнопкой мыши, перемещение мыши) называется событием. Событие воспринимается ОС Windows и преобразуется в сообщение – запись, содержащую необходимую информацию о событии (например, какая клавиша была нажата, в каком месте экрана произошел щелчок мышью и т.д.). Сообщения могут поступать не только от пользователя, но и от самой ОС, а также других приложений. Определен достаточно широкий круг стандартных сообщений, образующих иерархию, кроме того, можно определять собственные сообщения.

Сообщения поступают в общую очередь, откуда распределяются по очередям сообщений. Каждое приложение содержит цикл обработки очереди сообщений, в котором выбирается сообщение из очереди и вызывается подпрограмма, предназначенная для его обработки. Таким образом, Windows-приложение состоит из главной программы, обеспечивающий инициализацию и завершение приложения, цикла обработки сообщений и набора обработчиков событий. Более подробную информацию о функционировании Windows-приложений можно найти в [21, 22], а справочную информацию о функциях Win-dows API (Application Program Interface – программный интерфейс приложения) в [5].

Среда Visual Studio содержит удобные средства разработки Windows-приложений, выполняющие вместо программиста рутинную работу – создание шаблонов приложений и форм, заготовок обработчиков событий, организацию цикла обработки сообщений и т.д. При создании нового проекта Visual C# и выбора шаблона Windows Application среда Visual Studio сформирует готовый шаблон Windows-приложения. В этом шаблоне имеется вкладка заготовки формы Form1.cs[Design], которая располагается в основной части экрана.

Форма представляет собой окно и предназначена для размещения компонентов (элементов управления) – меню, текста, кнопок, списков, изображений и т.д.

Среда создает не только заготовку формы, но и шаблон текста приложения. Перейти к нему можно, щелкнув в окне Solution Explorer (View→Solution Explorer) правой кнопкой мыши на файле Form1.cs и выбрав в контекстном меню команду View Code. При этом откроется вкладка с кодом формы, листинг которого имеет примерно следующий вид.

Листинг 1. Шаблон Windows-приложения.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace windowsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

    }
  }
}

```

Приложение начинается с директив использования пространств имен библиотеки .NET. Для пустой формы, не содержащей ни одного компонента, необходимыми являются только две директивы:

```

using System;
using System.Windows.Forms;

```

Остальные директивы добавлены средой на «вырост». Все классы библиотеки .NET, а также все классы, который создает программист в среде .NET, имеют одного общего предка – класс **Object** и организованы в единую иерархическую структуру. Внутри нее классы логически сгруппированы в так называемые пространства имен, которые служат для упорядочивания имен классов и предотвращения конфликтов имен: в разных пространствах имена могут совпадать. Пространства имен могут быть и вложенными. Любая программа, создаваемая в .NET использует пространство имен **System**. В нем определены классы, которые обеспечивают базовую функциональность, например, поддерживают выполнение математических операций, управление памятью и ввод-вывод. Пространство имен **System.Windows.Forms** содержит элементы графического интерфейса пользователя, такие как формы, кнопки и т.д. Также в листинге программы содержится описание части класса **Form1**, порожденного от класса **Form**:

```

public partial class Form1 : Form

```

Для улучшения читабельности программ введена возможность разбивать описание типа на части и хранить их в разных физических файлах, создавая так называемые частичные типы (**partial types**). Это может потребоваться при описании классов, содержащих большой объем исходных текстов или, что более актуально, для отделения части кода, сгенерированной средствами среды от написанной программистом вручную. Кроме того, такая возможность облегчает отладку программы, позволяя отделить отлаженные части класса от новых. Для описания отдельной части класса используется модификатор **partial**. Он может применять к классам, структурам и интерфейсам, например:

```

public partial class A
{
    ...
}
public partial class A
{

```

```
...  
}
```

После совместной компиляции этих двух частей получается такой же код, как если бы класс был описан обычным образом. Все части одного и того же частичного типа должны компилироваться одновременно, иными словами добавление новой части к уже скомпилированной не допускается. Модификатор `partial` не является ключевым словом и должен стоять непосредственно перед одним из ключевых слов `class`, `struct` или `interface` в каждой из частей. Все части определения одного класса должны быть описаны в одном и том же пространстве имен. Если модификатор `partial` указывается для типа, описание которого состоит только из одной части, то это не является ошибкой.

Поскольку, как следует из листинга 1, класс `Form1` описан как частичный тип, то для получения полного исходного описания этого класса, находясь на вкладке `Form1.Designer.cs`, либо `Form1.cs*` (рис. 16) следует выбрать из левого раскрывающегося списка либо пункт `components`, либо `Dispose (bool disposing)`, либо `InitializeComponent()`. При выборе одного из этих пунктов откроется текст, представленный в листинге 2.

Листинг 2. Шаблон Windows-приложения (продолжение).

```
namespace windowsApplication4  
{  
    partial class Form1  
    {  
        private System.ComponentModel.IContainer  
        components = null;  
        protected override void Dispose(bool disposing)  
        {  
            if (disposing && (components != null))  
            {  
                components.Dispose();  
            }  
            base.Dispose(disposing);  
        }  
  
        #region windows Form Designer generated code  
        private void InitializeComponent()  
        {  
            this.components = new  
                System.ComponentModel.Container();  
            this.AutoScaleMode =  
                System.Windows.Forms.AutoScaleMode.Font;  
        }  
    }  
}
```

```

        this.Text = "Form1";
    }
    #endregion
}
}

```

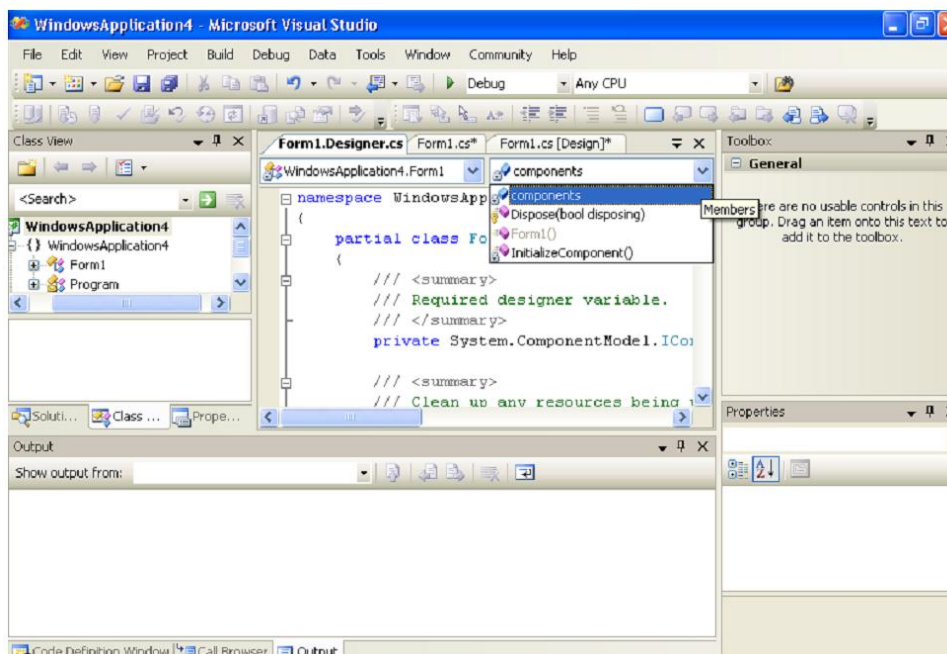


Рисунок 16 – Окно, используемое для доступа к различным частям класса

Класс **Form1** наследует от своего предка множество элементов, а в нем самом, как видно из листинга 2, описано новое закрытое поле **components** – контейнер для хранения компонентов, которые можно добавить в класс формы.

Поскольку во вновь созданной форме компонентов не имеется, то значение этого поля равно **null**. Конструктор формы вызывает закрытый метод **InitializeComponent**, автоматически формируемый средой (код этого метода скрыт между директивами препроцессора **#region** и **#endregion**).

Этот метод обновляется средой при добавлении элементов управления на форму, а также при изменении свойств формы и свойств содержащихся на ней элементов управления. Например, если изменить цвет фона формы с помощью окна свойств (Properties), в методе появится примерно такая строка:

```

this.BackColor=System.Drawing.SystemColors.ControlDarkDark

```

Метод освобождения ресурсов **Dispose** вызывается автоматически при закрытии формы. Если в режиме отладки произвести пошаговый запуск программы (Debug → Step Over или нажав клавишу F10), то появится еще одна вкладка **Program.cs**, которая содержит текст, представленный в листинге 3.

Листинг 3. Шаблон Windows-приложения (продолжение).

```

using System;

```

```

using System.Collections.Generic;
using System.Windows.Forms;

namespace windowsApplication4
{
    static class Program
    {
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

Точка входа в приложение, метод **Main**, содержит вызов трех статических методов класса **Application**. Последний из них – метод **Run**, запускает цикл обработки сообщений и выводит на экран форму, новый экземпляр которой передается ему в качестве параметра.

Процесс создания Windows-приложения состоит из двух основных этапов:

- 1) визуальное проектирование, т.е. задание внешнего облика приложения;
- 2) определения поведения приложения путем написания процедур обработки событий.

На этапе визуального проектирования на форму помещаются компоненты (элементы управления) и задаются их свойства и свойства самой формы при помощи окна свойств (рис. 17). Если его не видно, то можно воспользоваться командой меню **View→Properties Window**. Свойства отображаются либо в алфавитном порядке, либо сгруппированными по категориям. Способ отображения выбирается с помощью кнопок, расположенных в верхней части окна свойств (рис. 17).

Для размещения элементов управления на форме следует воспользоваться палитрой компонентов **Toolbox**. Если ее не видно, то следует воспользоваться командой меню **View →Toolbox**. Поместить элемент управления на форму можно либо дважды щелкнув по нему, либо при нажатой левой кнопки мыши перетащить компонент на форму. Можно также сделать один щелчок на палитре и еще один щелчок в том месте формы, куда планируется поместить элемент управления. Задание свойств выполняется либо выбором имеющихся в списке вариантов, либо вводом требуемого значения с клавиатуры.

Определение поведения программы начинается с принятия решений, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе

пунктов меню и т.д., иными словами, по каким событиям будут выполняться действия, реализующие функциональность программы. Заготовка шаблона обработчика события формируется двойным щелчком на поле, расположенном справа от имени соответствующего события на вкладке Events окна свойств, при этом появляется вкладка окна редактора кода с заготовкой соответствующего обработчика.

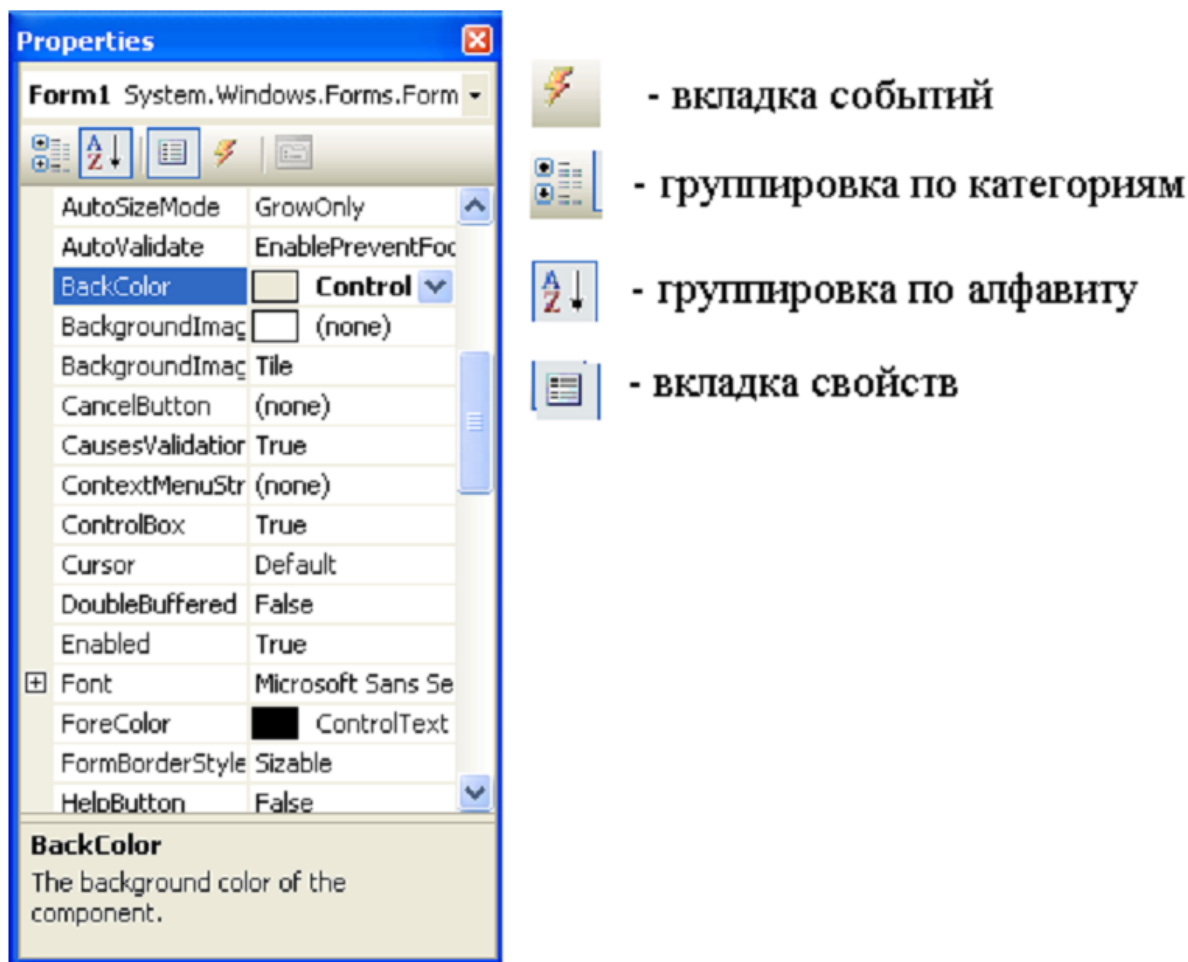


Рисунок 17 – Окно свойств

Для каждого класса определяется свой набор событий, на которые он может реагировать. Наиболее часто используемые события перечислены ниже.

- 1) **Activated** – получение формой фокуса ввода.
- 2) **Click, DoubleClick** – одинарный и двойной щелчки мышью.
- 3) **Close** – закрытие формы.
- 4) **Load** – загрузка формы.
- 5) **KeyDown, KeyUp** – нажатие и отпускание любой клавиши и их сочетаний.
- 6) **KeyPress** – нажатие клавиши, имеющей ASCII-код.
- 7) **MouseDown, MouseUp** – нажатие и отпускание кнопки мыши.

8) **MouseMove** – перемещение мыши.

9) **Paint** – возникает при необходимости прорисовки формы.

Имя обработчика события формируется средой автоматически из имени компонента и имени события (первым идет имя компонента, далее символ подчеркивания и после имя события). Пусть, например, на форме размещена кнопка с именем **button1**, тогда имя обработчика щелчка мышью по этой кнопке будет **button1_Click**, а заготовка обработчика события, также автоматически формируемого средой **Visual Studio**, выглядит следующим образом:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

Как следует из этого текста обработчику события передаются два параметра: **sender** – объект-источник события и **e** – запись, соответствующая типу события. Часто в обработчиках событий используется оператор приведения объектных типов **as**. Для обеспечения совместимости в 99% случаев источник события **sender** имеет тип **object**, хотя в тех же 99% случаев им является форма или другие компоненты. Поэтому, чтобы иметь возможность пользоваться их свойствами, применяют оператор **as**. Например, создается программа калькулятор (рис. 18).

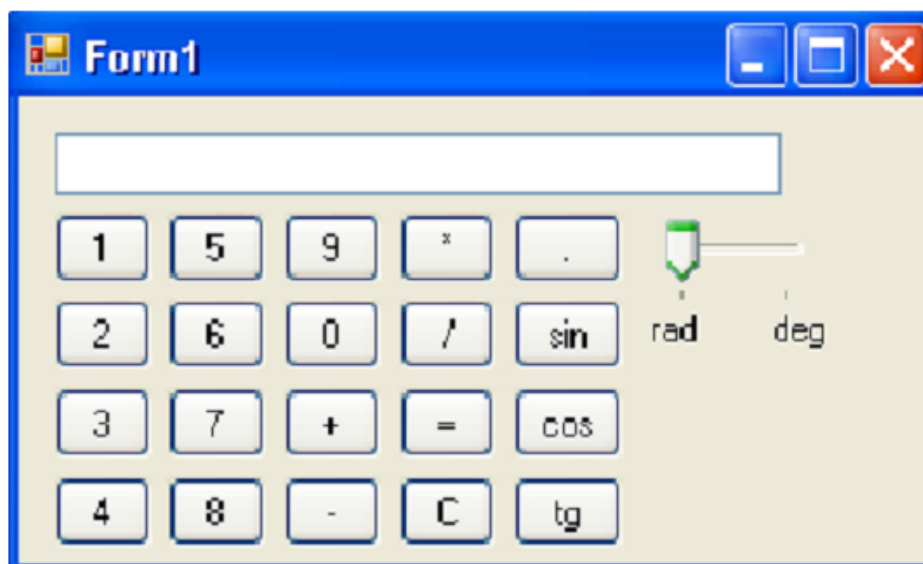


Рисунок 18 – Внешний вид программы калькулятор

При нажатии на цифровые клавиши формируется число – к содержимому поля редактирования (компонент **textBox1**) справа добавляется цифра, отображенная на кнопке. Поскольку при нажатии на любую цифровую клавишу происходит однотипное действие, то и обработчик события для всех этих клавиш будет одним и тем же. Но в нем придется в этом случае определять

объект – источник события и использовать оператор **as**:

```
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text+(sender as Button).Text;
}
```

При нажатии на клавишу, выполняющую одну из четырех арифметических операций, необходимо в обработчике события сохранить число, введенное в поле редактирования и сохранить код нажатой клавиши. Поскольку данные, вводимые в поле редактирования имеют строковый тип (тип **string**), то для выполнения арифметических действий необходимо выполнить преобразование из строки в переменную соответствующего типа. Преобразование можно выполнить либо при помощи специального класса **Convert**, определенного в пространстве имен **System**, либо с помощью метода **Parse**, имеющегося в каждом стандартном арифметическом классе. Например, преобразовать строку в вещественное число типа **double** можно при помощи следующих синтаксических конструкций – **Convert.ToDouble(s)**, либо **double.Parse(s)** (здесь **s** – это переменная типа **string**). Для сохранения кода нажатой клавиши в классе формы можно описать дополнительное поле. С другой стороны у каждого компонента наследника формы и у самой формы имеется поле **Tag**, значением которого должно быть целое число. Тогда на этапе визуального проектирования свойству **Tag** клавиш '+', '-', '*', '/' можно, например, присвоить соответственно значения 0, 1, 2 и 3. Тогда в обработчике события нажатием на одну из этих клавиш следует выполнить присваивание свойства **Tag** кнопки свойству **Tag** формы:

Tag = (sender as Button).Tag

Далее в обработчике события нажатием на клавишу '=' следует использовать свойство **Tag** формы как выражение для оператора **switch** (переключатель).

В выражениях часто используются математические функции, например косинус или возведение в степень. Они реализованы в классе **Math**, определенном в пространстве имен **System**. Например, чтобы вычислить синус некоторой величины **x**, следует записать **Math.sin(x)**. Также у класса **Math** имеются два поля, в которых хранится число π и число **e**, доступ к которым можно получить, соответственно, использовав записи **Math.PI** и **Math.E**.

Более подробную информацию по программированию на языке Visual C# можно получить из [20], а о синтаксических конструкциях языка, унаследованных им из языков C и C++ из [19].

Варианты индивидуальных заданий

Написать программу «Калькулятор» ... (по вариантам).

Студенты с вариантом 1-4: ... выполняющую два основных арифметических действия («+», «-»).

Студенты с вариантом 5-8: ... вычисляющую значения \sin для аргументов заданных как в градусах, так и в радианах.

Студенты с вариантом 9-12: ... выполняющую преобразование целых чисел из двоичной системы счисления в десятичную.

Студенты с вариантом 13-16: ... выполняющую два основных арифметических действия («:», «*»).

Студенты с вариантом 17-21: ... вычисляющую значения \cos для аргументов заданных как в градусах, так и в радианах.

Студенты с вариантом 22-25: ... выполняющую преобразование целых чисел из десятичной системы счисления в двоичную.

Студенты с вариантом 26-30: ... вычисляющую значения tg для аргументов заданных как в градусах, так и в радианах.

ДОМАШНЕЕ ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ **СТУДЕНТА**

Домашнее задание в 1 семестре включает создание технического задания (ТЗ) на разработку программного продукта (стандарт на ТЗ по ГОСТ 34.602-89 «Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы»).

Выполненное задание оформляется в виде отчета. Отчет оформлен согласно ГОСТ 7.32-2017.

Студент сдает отчет преподавателю ТОЛЬКО в электронном виде, с распечатанным титульным листом, на котором стоит подпись студента и дата сдачи отчета.

Преподаватель принимает отчет, соответствующий правилам оформления, согласно ГОСТ.

В случае наличия ошибок оформления – отчет отдается студенту для исправления.

В случае правильности, преподаватель подписывает титульный лист.

Студент сканирует титульный лист, заменяет в word-файле пустой титульный на подписанный, конвертирует весь отчет с последними правильными правками в формат pdf. Выкладывает отчет в портфолио студента в личном кабинете.

Варианты индивидуальной темы создания программного продукта, представлены в таблице. Студент имеет право предложить свой вариант темы, связанной с его интересами в области разработки игр, мобильных предложений, программных продуктов широкого потребления, согласовав с преподавателем индивидуальную тему.

№ п/п	Тема
1	<i>Автоматизированная система анализа успеваемости группы</i>
2	<i>Автоматизированная система учета и выдачи лекарств в аптеке</i>
3	<i>Автоматизированная система выдачи книг в библиотеке</i>
4	<i>Автоматизированная система учета пациентов в больнице</i>
5	<i>Автоматизированная система регистрации проживающих в гостинице</i>
6	<i>Автоматизированная система составления меню</i>
7	<i>Автоматизированная система учета животных в зоопарке</i>
8	<i>Автоматизированная система выдачи билетов в кинотеатре</i>
9	<i>Автоматизированная система учета коллекции</i>
10	<i>Автоматизированная система учета клиентов в мобильной компании</i>
11	<i>Автоматизированная система учета и продажи цветов в магазине</i>
12	<i>Автоматизированная система учета и продажи обуви в магазине</i>
13	<i>Автоматизированная система выдачи талонов в поликлинике</i>
14	<i>Автоматизированная система приема заказов в ателье</i>
15	<i>Автоматизированная система составления расписания в ВУЗе</i>
16	<i>Автоматизированная система оказания услуг по ремонту телевизоров</i>
17	<i>Автоматизированная система учета клиентов и оказанных услуг в салоне красоты</i>
18	<i>Автоматизированная система изготовления мебели</i>
19	<i>Автоматизированная система ведения учета на складе</i>
20	<i>Автоматизированная система учета породистых животных в клубе</i>
21	<i>Автоматизированная система фитнес-центра</i>
22	<i>Автоматизированная система учета топлива на автозаправочной станции</i>

23	<i>Автоматизированная система учета клиентов в стоматологической клинике</i>
24	<i>Автоматизированная система учета и продаж строительных материалов</i>
25	<i>Автоматизированная система ведения учета соревнований по боксу</i>
26	<i>Автоматизированная система выпуска на линию автобусов</i>
27	<i>Автоматизированная система учета платежей домохозяйки</i>
28	<i>Автоматизированная система учета спортсменов мотоклуба</i>
29	<i>Автоматизированная система учета видеокассет в видео-салоне</i>
30	<i>Игра «Шашки»</i>

Во втором семестре студент дополняет ранее созданный отчет второй главой, где приводит анализ существующих методов проектирования программного продукта и обоснование выбора методов проектирования.

Создание отчетов с использованием Ms Word

Все отчеты, выполняемые в процессе обучения, студенты кафедры «Автоматизированные системы обработки информации и управления» должны оформлять согласно ГОСТ 7.32-2017. «Отчёт о научно-исследовательской работе. Структура и правила оформления».

Ниже приведены лишь основные требования для составления отчета.

Отчет должен быть выполнен в электронном виде для размера бумаги формата А4.

Отчет необходимо оформлять с соблюдением следующих размеров полей: верхнее – 20 мм; левое – 30 мм; нижнее – 20 мм.; правое – 15 мм.

Шрифт, используемый в отчете – Times New Roman. Цвет шрифта должен быть черным, высота букв, цифр и других значков – 12 или 14 пт. Разрешается понижать шрифт до 12 пт в таблицах. Также разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры.

Величина красной строки – абзацный **отступ** должен быть 1,25 см.

Межстрочный интервал всего документа должен быть 1,5 строки.

Все формулы, встречаемые в отчете – оформляются через «Редактор формул».

Формула располагается с новой строчки, с красной строки. Формулы следует нумеровать арабскими цифрами в пределах всего текста в круглых скобках в крайнем правом положении на строке, например:

$$f(x) = \cos(x) - 0,2x + 1 \quad (1)$$

Рисунки и подрисуночная подпись размещаются по центру страницы, без красных строк (примером оформления рисунков является любой рисунок данного документа). Нумерация рисунков – сквозная. Ссылки по тексту документа на рисунки обязательны, они имеют вид: (... на рис. 3 изображено...).

Таблицы, как и рисунки, следует располагать непосредственно после текста, в котором она впервые упоминается. Каждая таблица должна иметь порядковый номер, изображаемый арабскими цифрами, нумерация должна быть сквозной в пределах всего документа. Название таблицы следует помещать над таблицей слева без красной строки, с её номером через тире (без точки в конце названия). Примером оформления таблицы является, например, табл. 1 данного документа. Ссылка на таблицу по тексту документа также обязательна, как и у рисунков.

Структура отчета и порядок расположения структурных элементов отчета следующий:

Титульный лист (см. образец ниже)

Реферат

Содержание

Термины и определения

Перечень сокращений и обозначений

Введение

1 Создание технического задания согласно ГОСТ 34

1.1 Назначение и цель автоматизированной системы

1.2 Характеристика объекта автоматизации (в разделе указывается для

кого может быть предназначена данная АС, предприятия, сотрудники каких профессий могут быть задействованы при работе указанной АС)

1.3 Разработка требований к системе (указываются требования к функциям, к задачам АС, к программному обеспечению, к среде и языку разработки, к безопасности данных, к надежности системы, к эргономике интерфейса, к защите информации от несанкционированного доступа)

1.4 Плановые сроки и этапы для создания системы

2 Теоретический анализ существующих методов проектирования программного продукта

3 Обоснование выбора метода проектирования

Заключение

Список использованных источников

Приложение А. ГОСТ 34

Приложение Б. ГОСТ 7.32-2017

Структурные элементы, расположенные ниже «Содержания», **подлежат отображению при создании автоматического оглавления**, и их необходимо задавать заголовками в самой пояснительной записке.

Структурные элементы «Введение», «Заключение», «Список использованных источников», «Приложение» – это заголовки первого уровня.

Главы 1, 2, 3 – заголовки второго уровня.

Подглавы 1.1, 1.2 и т.д. – заголовки третьего уровня.

В конце заголовка или номера у главы – точки не ставят.

Структурные элементы, расположенные выше «Содержания» – в автоматически созданном оглавлении отображаться не должны.

Структурные элементы, подлежащие отображению, размещаются в «Содержании» с указанием номеров их начальных страниц.

Образец структуры «Содержания» приведен на рис. 8.

СОДЕРЖАНИЕ	
ВВЕДЕНИЕ.....	5
1 Построение графиков функций.....	6
1.1 Задание.....	6
1.2 Практическая часть.....	6
2 Вычисление определенных интегралов.....	10
2.1 Задание.....	10
2.2 Практическая часть.....	10
3 Технологии будущего.....	12
3.1 Русские технологии будущего	12
3.2 Зарубежные разработки.....	12
ЗАКЛЮЧЕНИЕ.....	15
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	16
ПРИЛОЖЕНИЕ А.....	17

Рисунок 8 – Образец структуры содержания

В отчете заголовки первого уровня выравниваются по центру страницы, не нумеруются и не имеют красной строки, например:

Введение

Заключение

Список использованных источников

Заголовки второго и третьего уровня размещаются по левому краю с красной строки – отступ 1,25 см. При необходимости их можно выделять жирным шрифтом, курсивом, или использовать все прописные буквы, например:

1 СОЗДАНИЕ МАКРОСА ДЛЯ ПОСТРОЕНИЯ ГРАФИКА ФУНКЦИИ

1.1 Задание по варианту

1.2 Описание хода выполненной работы

2 ПЕРЕВОД СИСТЕМ СЧИСЛЕНИЯ

2.1 Задание по варианту

2.2 Описание хода выполненной работы

Структурные элементы – заголовки первого и второго уровня должны начинаться с нового листа.

«Термины и определения», а так же «Перечень сокращений и обозначений»

содержат уточнения терминов, перечень обозначений и сокращений, применяемых в тексте. Их запись приводят в алфавитном порядке (порядок алфавитов - русский, английский) с необходимой расшифровкой и пояснениями. Не допускается сокращение слов в тексте отчета, кроме общепринятых сокращений.

Список использованных источников оформляется в соответствии с требованиями ГОСТ 7.1-2003 [6].

Материал, дополняющий текст отчета, необходимо помещать в приложениях. Приложения обозначают заглавными буквами русского алфавита.

Нумерация страниц – арабскими цифрами в центре нижней части листа без точки. Нумерация страниц в текстовом документе должна быть сквозной. На титульном листе и листах до «Содержания» включительно – номер страницы не ставят (все листы, начиная с титульного, считаются, а номер страницы проставляется на листах после содержания).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Амелина, М. А. Программа схемотехнического моделирования Micro-Cap 8 [Текст] / М. А. Амелина, С. А. Амелин. – М. : Горячая линия – Телеком, 2007. – 464 с.
2. Аристов, В. В. Программное управление автоматизированными производственными системами [Текст] : конспект лекций / В. В. Аристов. – Омск : ОмГТУ, 2009. – 98 с.
3. Бекаревич, Ю. Б. Самоучитель Access 2010 [Текст] / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб. : БХВ-Петербург, 2011. – 432 с.
4. Белик, А. Г. Теория и технология программирования [Текст] : конспект лекций / А. Г. Белик, В. Н. Цыганенко. – Омск : ОмГТУ, 2013. – 85 с.
5. Верма, Р. Д. Справочник по функциям Win32 API [Текст] / Р. Д. Верма. – М. : Горячая линия-Телеком, 2005. – 551 с.
6. ГОСТ 7.1-2003. Библиографическая запись. Общие требования и правила составления [Текст]. – М. : ИПК Изд-во стандартов, 2004. – 47 с.
7. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления [Текст]. – М. : ИПК Изд-во стандартов, 2001. – 16 с.
8. Гуменюк, А. С. Информатика [Текст] / А. С. Гуменюк, И. В. Потапов. – Омск : ОмГМА, 2012. – 175 с.
9. Задорожный, В. Н. Информатика [Текст] : конспект лекций / В. Н. Задорожный, О. Н. Канева. – Омск : ОмГТУ, 2005. – Ч. 1. – 43 с.
10. Задорожный, В. Н. Информатика [Текст] : метод. указания к лаб. работам / сост. : В. Н. Задорожный, О. Н. Канева. – Омск : ОмГТУ, 2005. – 54 с.
11. Илюшечкин, В. М. Основы использования и проектирования баз данных [Электронный ресурс]: учеб. пособие для вузов по направлению «Информатика и вычислительная техника» / В. М. Илюшечкин. – М. : Юрайт, 2011. – 1 эл. опт. диск (DVD-ROM) (гриф).
12. Крылов, Е. В. Техника разработки программ [Текст] : учеб. для вузов по направлениям «Информатика и вычислительная техника» и «Техника и технологии»: в 2 кн. Кн. 2 : Технология, надежность и качество программного обеспечения. – М. : Высш. шк., 2008. – 468 с.
13. Малков, О. Б. Базы данных [Текст]: метод. указания к выполнению лабораторных работ / О. Б. Малков, Е. Т. Гегечкори. – Омск : ОмГТУ, 2007. – 112 с.
14. Малков, О. Б. Базы данных [Текст]: учеб. пособие для студентов заочной формы обучения / О. Б. Малков, Е. Т. Гегечкори. – Омск : ОмГТУ, 2007. – 64 с.
15. Малков, О. Б. Работа с СУБД MySQL [Текст]: учеб. пособие / О. Б. Малков, М. В. Девятерикова. – Омск : ОмГТУ, 2010. – 84 с.

16. Морозов, М. А. Информационные технологии в социально-культурном сервисе и туризме. Оргтехника [Текст]: учебник для студ. высш. учеб. заведений / М. А. Морозов, Н. С. Морозова. – М. : Издательский центр «Академия», 2008. – 240 с.

17. Никонов, А. В. Электротехника и электроника [Электронный ресурс] : учеб. электрон. изд. локального распространения : конспект лекций / А. В. Никонов. – Омск : ОмГТУ, 2014. – 1 эл. опт. диск (CD-ROM).

18. Новожилов, О. П. Архитектура ЭВМ и систем [Электронный ресурс] : учеб. пособие для бакалавров вузов по направлению 230100 «Информатика и вычислительная техника» / О. П. Новожилов. – М. : Юрайт, 2012. – 1 эл. опт. диск (CD-ROM).

19. Павловская, Т. А. C/C++. Программирование на языке высокого уровня [Текст] / Т. А. Павловская. – СПб. : Питер, 2002. – 464 с.

20. Павловская, Т. А. C#. Программирование на языке высокого уровня [Текст] : учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2009. – 432 с. 4

21. Пирогов, В. Ю. Ассемблер для Windows [Текст] / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 864 с.

22. Пирогов, В. Ю. Ассемблер на примерах [Текст] / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 416 с.

23. Силаенков, А. Н. Информационные технологии [Электронный ресурс] : учеб. электрон. изд. локального распространения : учеб. пособие / А. Н. Силаенков. – Омск : ОмГТУ, 2014. – 1 эл. опт. диск (CD-ROM).

ПРИЛОЖЕНИЕ А

(обязательное)

Образец титульного листа

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Омский государственный технический университет»
Факультет информационных технологий и компьютерных систем
Кафедра «Автоматизированные системы обработки информации и управления»

ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Основы профессиональной деятельности»

на тему: «Автоматизированная система»

Выполнил:
студент гр. ИВТ – 119
И.И. Иванов

(подпись, дата)

Принял:
асс. кафедры АСОИУ
Ю.Г. Лагунова

(подпись, дата)

Омск 2019