

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ИНФОРМАТИКА

*Методические рекомендации к лабораторным работам
для студентов специальности
1-54 01 02 «Методы и приборы контроля качества
и диагностики состояния объектов»
очной и заочной форм обучения*



Могилев 2019

УДК 004
ББК 32.973
И 74

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления»
«3» сентября 2019 г., протокол № 1

Составитель канд. техн. наук, доц. В. М. Ковальчук

Рецензент Ю. С. Романович

Методические рекомендации к лабораторным работам предназначены для студентов специальности 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов» очной и заочной форм обучения.

Учебно-методическое издание

ИНФОРМАТИКА

Ответственный за выпуск	А. И. Якимов
Технический редактор	А. А. Подошевка
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд.л. . Тираж 56 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».

Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.

Пр-т Мира, 43, 212022, Могилев.

© Белорусско-Российский
университет, 2019



Содержание

Введение.....	4
1 Лабораторная работа № 1. Изучение систем счисления и способов представления информации в памяти ПК.....	5
2 Лабораторная работа № 2. Текстовый процессор Microsoft Word. Редактор формул, диаграммы, рисунки и таблицы.....	8
3 Лабораторная работа № 3. Электронные таблицы и табличный процессор Excel. Использование встроенных функций. Диаграммы.....	9
4 Лабораторная работа № 4. Создание макросов в MS Excel.....	10
5 Лабораторная работа № 5. Структура математического процессора MathCad. Вычисление функций и построение графиков в MathCad	13
6 Лабораторная работа № 6. Решение алгебраических уравнений в MathCad	16
7 Лабораторная работа № 7. Решение дифференциальных уравнений и систем в MathCad	18
8 Лабораторные работы № 8–10. Система моделирования Multisim.....	22
9 Лабораторная работа № 11. Программирование на Си. Интерфейс Microsoft Visual Studio 2010. Выполнение программ	28
10 Лабораторная работа № 12. Программирование линейных вычислительных процессов на Си.....	32
11 Лабораторная работа № 13. Программирование разветвляющихся процессов	35
12 Лабораторная работа № 14. Программирование циклических процессов с использованием счетчиков на Си.....	37
13 Лабораторная работа № 15. Использование подпрограмм и процедур.....	39
14 Лабораторная работа № 16. Использование функций.....	43
15 Лабораторная работа № 17. Задание структурированных типов данных на языке Си.....	45
Список литературы.....	47



Введение

Цель методических рекомендаций к лабораторным работам по дисциплине «Информатика» заключается в овладении и закреплении студентами практических навыков работы в среде приложений MS Office.

Целью преподавания дисциплины является изучение основных современных операционных систем и программных сред, пакетов прикладных программ для научных и инженерных расчетов, компьютерного проектирования и конструирования, основ программирования, методов математического моделирования, общих вопросов алгоритмизации и приобретение навыков решения задач с применением средств вычислительной техники.

Дисциплина «Информатика» является неотъемлемой частью современных инженерных знаний и входит в состав естественнонаучных дисциплин, компонент учреждения высшего образования.

Методические рекомендации предназначены для изучения основных принципов работы в текстовом процессоре Word и табличном процессоре Excel. Выполнение заданий позволит студентам выработать практические навыки в создании текстовых документов и сохранении их на диске, редактировании и форматировании текстов, подготовке простых и сложных таблиц, использовании графических возможностей программ, создании диаграмм и математических формул.

Полученные при изучении дисциплины знания и навыки будут востребованы при изучении специальных дисциплин инженерной направленности и станут инструментом для грамотного выполнения и оформления рефератов, курсовых и дипломных работ.

Каждая работа рассчитана на 2 часа.

Выполнение каждой работы производится в следующем порядке:

- 1) ознакомиться с теоретическими положениями работы;
- 2) согласно варианту, указанному преподавателем, выбрать задание и исходные данные, выполнить их и оформить рукописный отчет.

Отчет содержит: название и цель работы; постановку задачи; исходные данные; использованные технологии; результаты выполнения и анализ полученных результатов; выводы. В отчете можно привести также ответы на наиболее сложные вопросы, имеющиеся в конце каждой работы.

1 Лабораторная работа № 1. Изучение систем счисления и способов представления информации в памяти ПК

Цель работы: приобрести навыки перевода чисел в позиционных системах исчисления и ознакомиться со способами представления информации в памяти ПК.

Методические указания

Совокупность приемов именованности и обозначения чисел называется системой счисления. Системы счисления бывают позиционные, в которых значение цифры зависит от ее положения в ряду цифр, изображающих число, и непозиционные, в которых такой зависимости нет. В качестве знаков для записи чисел используются цифры.

Примером непозиционной системы счисления является римская система, в которой роль цифр играют буквы латинского алфавита: I – один, V – пять, X – десять, C – сто, L – пятьдесят, D – пятьсот, M – тысяча. Например, 3222 = CCCXXII. В непозиционной системе счисления арифметические операции выполнять неудобно и сложно. К позиционным относятся системы счисления: десятичная, двоичная, восьмеричная, шестнадцатеричная, которые имеют основания соответственно 10, 2, 8, 16 (таблица 1.1).

Таблица 1.1 – Числа в различных системах счисления

10	2	8	16	10	2	8	16
0	000	0	0	8	1000	10	8
1	001	1	1	9	1001	11	9
2	010	2	2	10	1010	12	A
3	011	3	3	11	1011	13	B
4	100	4	4	12	1100	14	C
5	101	5	5	13	1101	15	D
6	110	6	6	14	1110	16	E
7	111	7	7	15	1111	17	F

Любое десятичное число можно разложить по степеням основания его системы счисления. Например, число 130678 можно представить в виде

$$130678 = 8 \cdot 100 + 7 \cdot 101 + 6 \cdot 102 + 0 \cdot 103 + 3 \cdot 104 + 1 \cdot 105.$$

Здесь показатель степени – это номер позиции цифры в записи числа. Причем нумерация позиции цифр ведется справа налево, начиная с нуля.

Для представления чисел в памяти компьютера применяется двоичная система счисления, в которой используются две цифры: 0 и 1. Для сокращения записи адресов и содержимого байтов оперативной памяти компьютера



применяют шестнадцатеричную и восьмеричную системы счисления. В таблице 1.1 приведены первые 16 натуральных чисел, записанных в десятичной, двоичной, восьмеричной и шестнадцатеричной системах счисления.

Правила перевода чисел в позиционных системах счисления. Целые числа кодируются двоичным кодом просто – достаточно взять целое число и делить его пополам до тех пор, пока частное не будет равно единице. Совокупность остатков от каждого деления, записанная справа налево вместе с последним частным, и образует двоичный аналог десятичного числа.

При переводе из двоичной и шестнадцатеричной систем счисления в десятичную рассчитывается полное значение числа по формуле

$$13_{16} = 1 \cdot 16^1 + 3 \cdot 16^0 = 16 + 3 = 19.$$

Таким образом, $13_{16} = 19$.

$$10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 0 + 0 + 2 + 1 = 19.$$

Перевод из двоичной системы счисления в шестнадцатеричную.

1 Исходное число разбивается на тетрады (т. е. четыре цифры), начиная с младших разрядов. Если количество цифр исходного двоичного числа не кратно 4, оно дополняется слева незначащими нулями до достижения кратности 4.

2 Каждая тетрада заменяется соответствующей шестнадцатеричной цифрой в соответствии с таблицей.

Выполнить перевод числа 10011_2 в шестнадцатеричную систему счисления.

Поскольку в исходном двоичном числе количество цифр не кратно 4, дополняем его слева незначащими нулями до достижения кратности 4 числа цифр. Имеем $10011_2 = \underbrace{0001}_{16} \underbrace{0011}_B$. В соответствии с таблицей $0011_2 = 11_2 = 3_{16}$ и $0001_2 = 1_2 = 1_{16}$. Тогда $10011_2 = 13_{16}$.

Правила перевода правильных дробей. Результатом является всегда правильная дробь. Из десятичной системы счисления в двоичную и шестнадцатеричную:

1) исходная дробь умножается на основание системы счисления, в которую переводится (2 или 16);

2) в полученном произведении целая часть преобразуется в соответствии с таблицей в цифру нужной системы счисления и отбрасывается – она является старшей цифрой получаемой дроби;

3) оставшаяся дробная часть вновь умножается на нужное основание системы счисления с последующей обработкой полученного произведения в соответствии с шагами 1 и 2;

4) процедура умножения продолжается до тех пор, пока не будет получен нулевой результат в дробной части произведения или не будет достигнуто требуемое количество цифр в результате;

5) формируется результат: последовательно отброшенные в шаге 2 цифры составляют дробную часть результата, причем в порядке уменьшения старшинства.



Пример 1 – Перевести число $0,625_{10}$ в двоичную систему счисления (рисунок 1.1).



Рисунок 1.1 – Пример перевода дробной части десятичного числа

Правило перевода дробных чисел. Отдельно переводится целая часть числа, отдельно – дробная. Результаты складываются.

Пример 2 – Выполнить перевод из десятичной системы счисления в шестнадцатеричную числа $19,847$. Перевод выполнять до трех значащих цифр после запятой.

Представим исходное число как сумму целого числа и правильной дроби:

$$19,847 = 19 + 0,847.$$

Как следует из примера, $19 = 13_{16}$, а в соответствии с примером $0,847 = 0,D8D_{16}$. Тогда имеем

$$19 + 0,847 = 13_{16} + 0,D8D_{16} = 13,D8D_{16}.$$

Таким образом, $19,847 = 13,D8D_{16}$.

Порядок выполнения работы.

- 1 По указанию преподавателя выбрать вариант задания из таблицы 1.2.
- 2 Выполнить операции с числами.

Таблица 1.2 – Варианты заданий

Вариант	Числа для перевода	Вариант	Числа для перевода	Вариант	Числа для перевода
1	$231_{10} \rightarrow 16,8,2$ $48,55_{10} \rightarrow 2$	5	$132_{10} \rightarrow 16,8,2$ $68,44_{10} \rightarrow 2$	9	$443_{10} \rightarrow 16,8,2$ $67,88_{10} \rightarrow 2$
2	$564_{10} \rightarrow 16,8,2$ $72,43_{10} \rightarrow 2$	6	$434_{10} \rightarrow 16,8,2$ $44,73_{10} \rightarrow 2$	10	$333_{10} \rightarrow 16,8,2$ $43,75_{10} \rightarrow 2$
3	$322_{10} \rightarrow 16,8,2$ $55,87_{10} \rightarrow 2$	7	$711_{10} \rightarrow 16,8,2$ $46,55_{10} \rightarrow 2$	11	$811_{10} \rightarrow 16,8,2$ $40,55_{10} \rightarrow 2$
4	$987_{10} \rightarrow 16,8,2$ $23,98_{10} \rightarrow 2$	8	$459_{10} \rightarrow 16,8,2$ $44,98_{10} \rightarrow 2$	12	$422_{10} \rightarrow 16,8,2$ $75,71_{10} \rightarrow 2$

Контрольные вопросы

- 1 Какие системы счисления Вы знаете?
- 2 Как перевести целое двоичное число в восьмеричную и шестнадцатеричную системы счисления?



2 Лабораторная работа № 2. Текстовый процессор Microsoft Word. Редактор формул, диаграммы, рисунки и таблицы

Цель работы: приобрести практические навыки по работе с таблицами, набору формул, созданию диаграмм и рисунков в редакторе Microsoft Word.

Методические указания

Таблицы предназначены для упорядочивания данных и создания интересных макетов страницы с последовательно расположенными столбцами текста или графики. Для работы с таблицами используется пункт меню *Таблица*. Наиболее быстрый путь создания простой таблицы – например такой, которая имеет одинаковое количество строк и столбцов – с помощью команды *Добавить таблицу*.

Редактор математических формул *Microsoft Equation Editor* вызывается по *OLE*-технологии и ориентирован на создание сложных формул, как правило, содержащих матрицы, дроби, знаки суммирования и другие математические конструкции.

Порядок выполнения работы.

1 Создать новый документ, который должен содержать формулы из задания 1 и таблицу из рисунка 2.1.

Задание 1

$$1 \int_1^2 \frac{dx}{x} \quad (n = 10).$$

$$2 \int_0^1 \frac{dx}{1+x^2} \quad (n = 10).$$

$$3 \int_{0,7}^{1,3} \frac{dx}{\sqrt{2x^2 + 0,3}} \quad (n = 17).$$

$$4 \int_0^{1,2} \ln(1+x^2) dx \quad (n = 6).$$

$$5 \int_1^3 \frac{dx}{1+x} \quad (n = 4).$$

$$6 \int_1^9 \sqrt{6x-5} dx \quad (n = 8).$$

$$7 \int_4^{5,2} \ln x dx \quad (n = 6).$$

$$8 \int_0^1 e^{-x^2} dx \quad (n = 10).$$

ИНТЕРНЕТ ДЛЯ ВАС!

ТАРИФЫ НА УСЛУГИ	ЦЕНЫ УКАЗАНЫ В РУБ.							
	базовый		экономик		фанат		бизнес	
МИНИМАЛЬНАЯ ПРЕДОПЛАТА	122,00		91,00		304,00		304,00	
АБОНЕНТ/ПЛАТА В МЕСЯЦ	нет		18,20		91,00		91,00	
E-MAIL	1/МБ		бесплатно					
ДНЕВНОЕ ВРЕМЯ (в час)	с08 до11	с11 до17	с08 до11	с11 до17	с08 до11	с11 до17	с08 до11	с11 до17
	11,00	14,60	7,30	14,60	9,10	11,00	7,30	9,10
ВЕЧЕРНЕЕ ВРЕМЯ (в час)	14,60		11,00		11,00		11,00	
НОЧНОЕ ВРЕМЯ (в час)	7,30		5,40		бесплатно		5,40	
ДОМАШНЯЯ СТРАНИЧКА	18,20/мес. до 2 МБ		бесплатно до 2 МБ					

Рисунок 2.1 – Пример таблицы



Контрольные вопросы

- 1 Какой пункт меню Word используется для работы с таблицами?
- 2 Как удалить строку или столбец из таблицы?
- 3 Как объединить несколько ячеек?
- 4 Как добавить в текстовый документ математическую формулу?
- 5 Как сгруппировать несколько нарисованных элементов?

3 Лабораторная работа № 3. Электронные таблицы и табличный процессор Excel. Использование встроенных функций. Диаграммы

Цель работы: изучить технологию работы с функциями в среде MS Excel.

Методические указания

При построении расчетной таблицы в MS Excel имеется возможность использования большого количества встроенных функций, которые сосредоточены в удобном инструменте их ввода – мастере функций, объединяющем девять следующих категорий: финансовые; дата и время; математические, статистические; ссылки и массивы, работа с базой данных; текстовые, логические, проверки свойств и значений.

Любая функция имеет вид: *имя(список аргументов)*, где *имя* – это символическое имя функции, а *список аргументов* – величины, над которыми функция выполняет операции.

Аргументами функции могут быть адреса ячеек, константы, формулы, а также другие функции. Например, функция *КОРЕНЬ(ABS(A2))* вычисляет квадратный корень из абсолютного значения числа из ячейки *A2*.

Для использования мастера функций необходимо кликнуть на кнопке *fx* в строке ввода окна Excel, в открывшемся окне *Мастер функций* выбрать категорию, содержащую вызываемую функцию, выбрать требуемую функцию и нажать кнопку *OK*. Откроется окно мастера вызываемой функции, в которое следует ввести требуемые параметры и нажать *OK*.

При решении многих задач в Excel требуется использование матриц, например, решение систем линейных алгебраических уравнений, задач линейного программирования и других. Для работы с матрицами эффективно использование формул специального вида, называемых формулами массива, которые отличаются от обычных формул тем, что их аргументами и результатом является не одно число, а набор величин – матрица. При завершении таких операций требуется специальное подтверждение – вместо клавиши *ENTER* применяется комбинация из трех клавиш – *Ctrl + Shift + Enter*.



Порядок выполнения работы.

- 1 Получить у преподавателя задание на выполнение работы (таблица 3.1).
- 2 Разработать электронную таблицу «Формулы массива».
- 3 Разработать электронную таблицу «Встроенные функции».

Таблица 3.1 – Варианты заданий для выполнения работы

Номер варианта	Вычислить, используя формулы массива	Вычислить, используя встроенные функции
1	Матрицу В, обратную матрице С размерности 6×6	$y = x^2 + \cos x$
2	$ C = A \cdot B $, размерность $ A - (4, 4)$, $ B $ – выбрать самостоятельно	$y = \operatorname{tg} x - x$
3	Таблицу умножения от 5 до 14	$y = \operatorname{tg} x - x$
4	$ C = A \cdot B $, размерность $ A - (4, 2)$, $ B $ – выбрать самостоятельно	$y = \operatorname{ctg} x - x^2$
5	Таблицу умножения от 1 до 12	$y = \ln x - \sin x$
6	Матрицу В, обратную матрице С размерности 4×4	$y = x - \sin x$
7	Таблицу умножения от 1 до 15	$y = \sqrt{x} + \sin x$
8	$ C = A \cdot B $, размерность $ A - (3, 5)$, $ B $ – выбрать самостоятельно	$y = \sqrt{x^2 + 3} + \cos x$
9	Матрицу В, обратную матрице С размерности 5×5	$y = x^2 + \sin x$

Контрольные вопросы

- 1 Какие категории встроенных функций программы MS Excel Вы знаете?
- 2 Как ознакомиться с технологией использования встроенной функции?
- 3 Какие функции содержатся в категории *Логические*?
- 4 Что такое формулы массива и для чего они используются?
- 5 Как завершается ввод формулы массива?

4 Лабораторная работа № 4. Создание макросов в MS Excel

Цель работы: изучить технологию создания макросов с помощью инструментов MS Excel.

Методические указания

В качестве первоначального знакомства с VBA попытаемся решить следующую задачу. Допустим, вы решили вести учет своих расходов, и с этой целью в конце каждого месяца намерены составлять таблицу (рисунок 4.1) и строить диаграмму для более наглядного отображения доли каждой статьи расходов вашего бюджета. Составлять ежемесячно одну и ту же таблицу с одновременным построением диаграммы – довольно непроизводительная трата



времени. Более разумно один раз научить компьютер создавать таблицу, а потом по мере необходимости лишь отдавать команду подготовки таблицы, чтобы осталось только внести в нее данные.

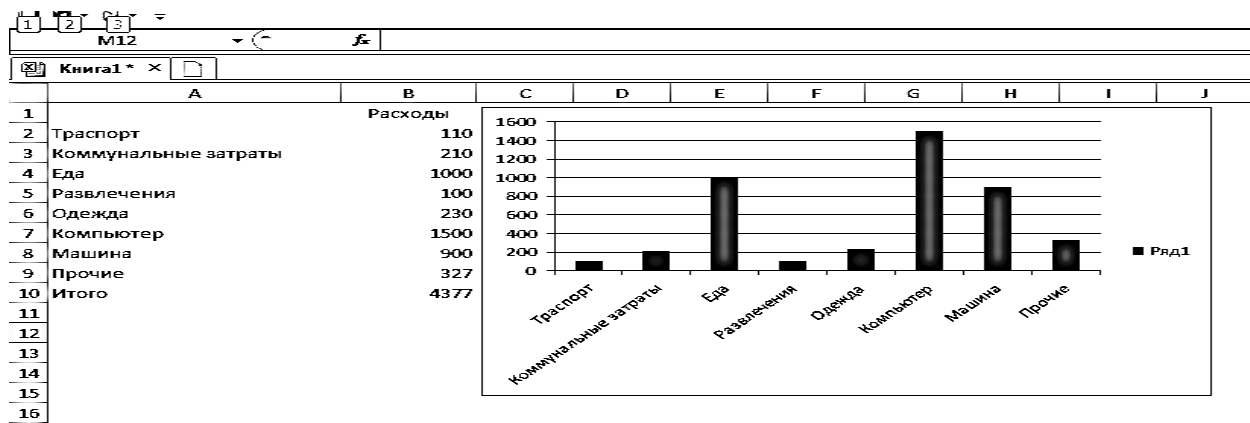


Рисунок 4.1–Таблица ежемесячных расходов

Для обучения компьютера отлично подходит Macro Recorder – транслятор, создающий программу (макрос) на языке VBA, которая является результатом перевода на язык VBA действий пользователя с момента запуска Macro Recorder до окончания записи макроса.

Для активизации Macro Recorder выберите закладку *Вид* → *Макросы* → *Запись макроса*. Появится диалоговое окно *Запись макроса*. Это диалоговое окно позволяет задать параметры макроса. В диалоговом окне *Запись макроса* в поле *Имя макроса* введем *Расходы*, а в поле *Описание* – *Расчет месячных расходов*.

Присвоим макросу комбинацию клавиш быстрого вызова:

- *сочетанием клавиш* в расположенном рядом поле введем букву, например, *r* (вызов макроса будет осуществляться при нажатии клавиш *Ctrl + r*);
- сохраним макрос в текущей рабочей книге, установим в разделе *Сохранить в Эта книга*.

Нажмем кнопку *ОК*. Появится кнопка *Остановить запись*. Теперь все производимые действия будут записываться до тех пор, пока не будет нажата эта кнопка. Построим шаблон таблицы расходов.

Рабочий лист теперь будет выглядеть так, как показано на рисунке 4.1. Остановим запись макроса, нажав кнопку *Остановить запись* (Stop Recording). Заполним ячейки таблицы исходными данными, расчет суммарных расходов и построение диаграммы теперь будет происходить автоматически.

Для запуска созданного макроса необходимо перейти на новый рабочий лист или очистить текущий, затем выбрать закладку *Вид* → *Макросы*, которая вызовет диалоговое окно *Макрос*. В этом окне в списке выделим исходный макрос и нажмем кнопку *Выполнить*. Диалоговое окно закроется и выполнится процедура, создающая на активном рабочем листе шаблон таблицы. Теперь в нее остается ввести новые данные, а расчет суммарных расходов и построение диаграммы будет происходить автоматически.

Также для запуска созданного макроса можно нажать комбинацию клавиш *Ctrl + r*. Excel запустит макрос, который последовательно выполнит все записанные действия.

Пользователю предоставляется возможность отредактировать существующий макрос. Для изменения макроса используют закладку *Вид* → *Макросы* → *Макросы* → *Изменить* непосредственно на листе модуля.

Порядок выполнения работы.

1 Проанализировать поставленную задачу (см. варианты заданий) и определить, данные какого типа содержит данная таблица.

2 Разработать макрос для создания заголовка таблицы и для задания формата ячеек каждого типа данных. Текстовая информация должна вводиться в ячейку в несколько строк, стоимость или цена – с указанием единиц.

3 Рассмотреть различные варианты возможностей вызова макроса.

Варианты заданий

1 Название фирмы, дата создания, годовой доход в долларах США, число сотрудников, телефон с кодом города.

2 Фамилия студента, номер зачетной книжки, дата рождения, адрес, телефон.

3 Заболевание, код заболевания, количество заболевших в текущем году, количество заболевших в прошедшем году, процент роста.

4 Фамилия абонента, телефон, дата разговора, тариф, сумма в рублях.

5 Наименование товара, код товара (страна-код), дата использования, цена в долларах США.

Контрольные вопросы

1 Что такое макрос?

2 Как начать запись макроса?

3 Можно ли просмотреть или изменить записанный макрос?

4 Перечислите способы запуска макроса.

5 Какие параметры макроса позволяет задать диалоговое окно *Запись макроса*?



5 Лабораторная работа № 5. Структура математического процессора MathCad. Вычисление функций и построение графиков в MathCad

Цель работы: ознакомиться со структурой математического процессора MathCad, освоить ввод данных, вычисление математических выражений.

Методические указания

Система MathCad – одна из самых мощных и эффективных систем математического направления, которая ориентирована на широкий круг пользователей и позволяет выполнять математические расчеты, как в численном, так и в символьном виде. Причем описание решения задач задается с помощью привычных математических формул и знаков.

Документ системы MathCad строится из областей, которые делятся на вычислительные, графические, текстовые.

В вычислительных областях можно задавать данные, выражения, операторы и управляющие структуры. Все данные системы можно разделить на простые и структурированные. Простые данные представлены константами и переменными; структурированные – дискретными переменными, массивами и файлами.

Константы – элементы данных, хранящие некоторые значения, которые не могут быть изменены.

Переменные – поименованные объекты, имеющие некоторые значения, которые могут изменяться в процессе выполнения документа.

Имена переменных в системе MathCad могут содержать любые латинские и греческие буквы, а также цифры, они должны начинаться только с буквы. Строчные и прописные буквы в именах различаются. Имена должны быть уникальными, т. е. они не должны совпадать с именами встроенных или определенных пользователем функций.

Выражение – это совокупность данных, функций и математических объектов, связанных знаками операций. Выражения могут содержать скобки.

Операции, используемые в выражениях, можно разделить на арифметические и логические. Арифметические операции представлены в палитре арифметических операторов; логические операции и операции отношения – в палитре логических операторов.

К базовым операторам системы относятся:

- $:=$ – оператор локального присваивания;
- \equiv – оператор глобального присваивания;
- $=$ – оператор вычисления и вывода.

Оператор локального присваивания ($:=$) распространяет свое действие на область документа, расположенную в строке и ниже места присваивания. Этот оператор выполняется так: данному, стоящему в левой части оператора, присваивается вычисленное значение выражения, стоящего в правой части оператора.



Оператор глобального присваивания (\equiv) не зависит от места присвоения и распространяет свое действие на весь документ. Этот оператор выполняется точно так же, как и оператор локального присваивания.

Оператор вычисления и вывода ($=$) выводит вычисленное значение выражения, стоящего в его левой части, на экран.

Функция – выражение, согласно которому проводятся некоторые вычисления с ее аргументами и определяется числовое значение. Функция имеет имя и может иметь список параметров. Различают стандартные и пользовательские функции.

Общий вид описания функции следующий:

$$\text{ИМЯ (СФП)} := \text{выражение},$$

где *ИМЯ* – имя функции;

СФП – список формальных параметров функции.

При обращении к функции формальные параметры заменяются на фактические, т.е. на выражения, имеющие числовые значения. Например,

$$z(m,n) := m^2 + n^2 \text{ – описание функции;}$$

$$z(2,3) = 13 \text{ – обращение к функции.}$$

MathCad позволяет обрабатывать различные виды графической информации. Возможности системы по работе с графикой таковы:

- построение двумерных графиков в декартовой и полярной системах координат;
- построение трехмерных поверхностных графиков;
- внесение рисунков, созданных другими компьютерными системами;
- создание анимационных клипов.

Для построения графиков используется палитра графиков. Перечень возможных типов графиков приведен в основном меню ***Insert – Graph***.

При построении двумерных графиков после нажатия соответствующей кнопки на панели графических инструментов появляется шаблон (рисунок 5.1).

В шаблоне графика по вертикали задаются через запятую функции, а по горизонтали – аргументы. Крайние шаблоны данных служат для указания предельных значений абсцисс и ординат, т.е. они задают масштабы графика. Если оставить эти шаблоны незаполненными, то масштабы по осям графика будут устанавливаться автоматически.

В полярной системе координат каждая точка задается углом a , радиусом и длиной радиус-вектора $R(a)$. График функции обычно строится при изменении угла a в пределах от 0 до 2π .

После построения график может быть отформатирован по следующим направлениям: форматирование осей графика, форматирование линий графика, форматирование надписей на графике (рисунок 5.2).



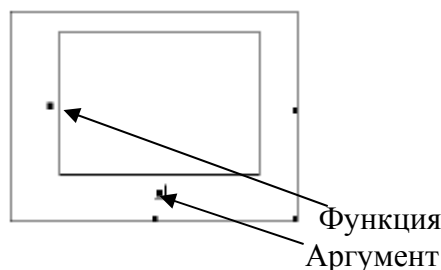


Рисунок 5.1– Шаблон двумерных графиков в декартовой системе координат.

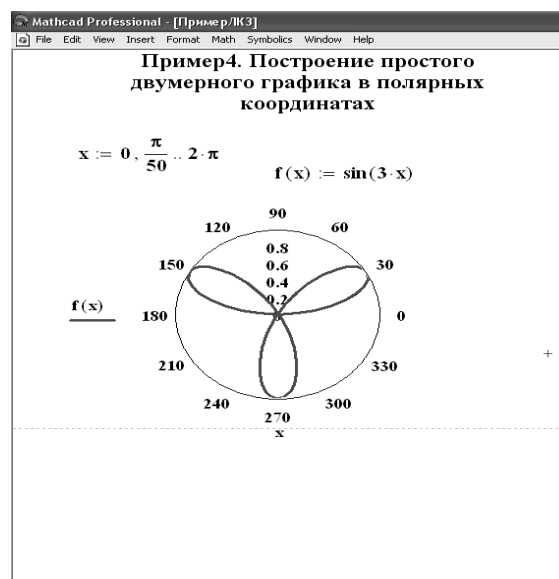
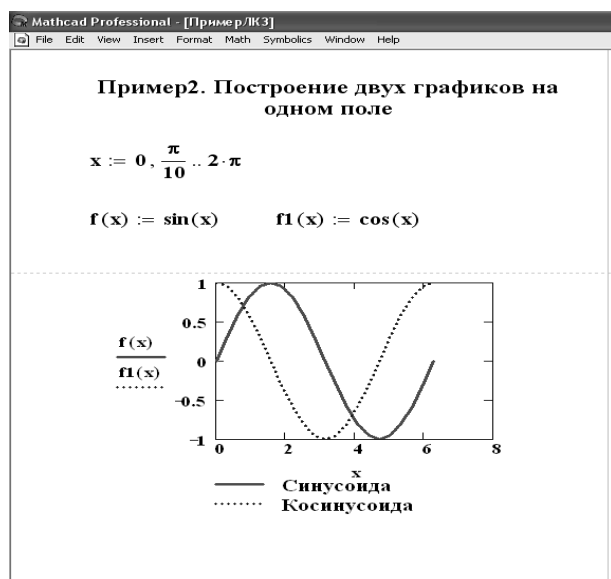


Рисунок 5.2 – Примеры

Порядок выполнения работы.

- 1 Задать дискретную переменную x от 1 до 10 с шагом 0,5 и по заданию преподавателя построить графики функций в декартовых координатах.
- 2 Задать дискретную переменную x от 0 до 2π с шагом $\pi/50$ и по заданию преподавателя построить графики функций в полярных координатах.

Контрольные вопросы

- 1 На какие типы задач ориентирована система MathCad?
- 2 Какие указатели присутствуют в окне системы MathCad?
- 3 Чем отличаются команды :=, =, ≡?
- 4 Как классифицируются функции MathCad?
- 5 Какой формат имеют пользовательские функции?
- 6 Как создается массив дискретных переменных аргумента функции?
- 7 Какую графическую информацию обрабатывает MathCad?
- 8 Как можно разместить на одном графике несколько функций?

6 Лабораторная работа № 6. Решение алгебраических уравнений в MathCad

Цель работы: ознакомиться с методами решения алгебраических уравнений в MathCad.

Методические указания

Для алгебраических уравнений вида $f(x) = 0$ решение в MathCad находится с помощью функции *root*.

Общий вид функции следующий:

$$\text{root}(f(x), x),$$

где $f(x)$ – функция, описывающая левую часть выражения вида $f(x) = 0$;

x – имя переменной, относительно которой решается уравнение.

Функция *root* реализует алгоритм поиска корня численным методом и требует предварительного задания начального приближения искомой переменной x . Поиск корня будет производиться вблизи этого числа. Таким образом, присвоение начального значения требует предварительной информации о примерной локализации корня.

Если после многих итераций MathCad не находит подходящего приближения, то появится сообщение «отсутствует сходимость».

Эта ошибка может быть вызвана следующими причинами:

- уравнение не имеет корней;
- корни уравнения расположены далеко от начального приближения;
- выражение $f(x)$ имеет разрывы между начальным приближением и корнем;
- выражение имеет комплексный корень, но начальное приближение было вещественным и наоборот.

В зависимости от типа задачи функция *root* может включать либо два, либо четыре аргумента и, соответственно, использует разные алгоритмы поиска корней:

$$\text{root}(f(x), x),$$

$$\text{root}(f(x), x, a, b),$$

где $f(x)$ – скалярная функция, определяющая уравнение $f(x) = 0$;

x – имя скалярной переменной, относительно которой решается уравнение;

a, b – границы интервала, внутри которого происходит поиск корня.

Первый тип функции *root* требует дополнительного задания начального значения переменной x (примерная локализация корня, поиск корня будет производиться вблизи этого числа).



Для задания начального значения или интервала удобно предварительно построить график функции. Чем точнее выбрано начальное приближение корня, тем быстрее будет *root* сходиться.

Пример – Найти корень уравнения $\sin(2x) + \cos(2x) = \sqrt{x \sin(3x)}$ при заданном начальном значении $x \approx 5$. Выполнить графическую интерпретацию результата.

Реализация в MathCad:

- привести уравнение к виду $f(x) = 0$, если это необходимо, например, уравнение $\sin(2x) + \cos(2x) = \sqrt{x \sin(3x)}$ преобразуется в $\sin(2x) + \cos(2x) - \sqrt{x \sin(3x)} = 0$;
- установить курсор в свободное место рабочего окна документа MathCad;
- определить функцию, содержащуюся в левой части уравнения вида $f(x) = 0$, например, $f(x) := \sin(2x) + \cos(2x) - \sqrt{x \sin(3x)}$;
- построить график функции $f(x)$ (рисунок 6.1);
- установить оси графика в виде креста;
- задать начальное приближение искомой переменной x ;
- подставляя в качестве параметров функции *root* имя функции левой части уравнения и переменную, содержащую начальное приближение, вывести значение корня уравнения при заданном начальном приближении с помощью оператора « \Rightarrow »;
- задать аргумент функции $f(x)$ в виде дискретной переменной, при этом диапазон изменения аргумента должен включать найденный корень уравнения.

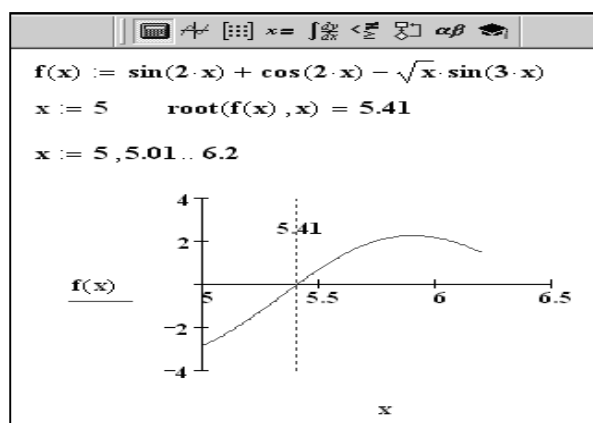


Рисунок 6.1 – Пример построения графика функции

Порядок выполнения работы.

1 Ознакомиться с методом решения алгебраических уравнений с помощью функции *root* и выполнить вышеразобранный пример.

2 С помощью функции *root* решить уравнения:

$$\cos(x) - x - 0,2 = 0;$$

$$x^3 - 10x + 2 = 0;$$

$$4x^4 - 10x^3 + 2x - 1 = 0 \text{ (} x \text{ от } -5 \text{ до } 15\text{)}.$$

Контрольные вопросы

- 1 Какие функции имеются в пакете MathCad для решения уравнений?
- 2 Какие алгоритмы поиска корней уравнений реализует функция *root*?
- 3 С какой целью желательно предварительно построить график функции?
- 4 По каким причинам может появиться сообщение «отсутствует сходимость»?

7 Лабораторная работа № 7. Решение дифференциальных уравнений и систем в MathCad

Цель работы: ознакомиться с методами решения дифференциальных уравнений в MathCad.

Методические указания

У MathCad очень небольшие возможности символьного решения обыкновенных дифференциальных уравнений (ОДУ). В частности, для решения ОДУ первого порядка можно записать готовую формулу для задачи Коши и символично ее проинтегрировать.

Для решения ОДУ и их систем с начальными условиями система Mathcad имеет ряд встроенных функций, предназначенных для численного решения ОДУ:

- **Odesolve(x,b,[step])** – функция, решающая задачу блочным методом;
- **rkfixed(y, x1, x2, npoints, D)** – решение на отрезке методом Рунге–Кутты с постоянным шагом;
- **Rkadapt(y, x1, x2, npoints, D)** – решение задачи на отрезке методом Рунге–Кутты с автоматическим выбором шага;
- **rkadapt(y, x1, x2, acc, npoints, D, kmax, save)** – решения задачи в заданной точке методом Рунге–Кутты с автоматическим выбором шага;
- **Bulstoer(y, x1, x2, npoints, D)** – решение задачи на отрезке методом Булирша–Штера;
- **bulstoer(y, x1, x2, acc, npoints, D, kmax, save)** – решение задачи в заданной точке методом Булирша–Штера;
- **StiffR(y, x1, x2, acc, D, J)** – решение задачи для жестких систем на отрезке с использованием алгоритма Розенброка;
- **stiffR(y, x1, x2, acc, D, J, kmax, save)** – решения задач для жестких систем на отрезке с использованием алгоритма Розенброка;
- **StiffB(y, x1, x2, acc, D, J)** – решение задачи для жестких систем на отрезке с использованием алгоритма Булирша–Штера;



– **stiffb(y, x1, x2, acc, D, J, kmax, save)** — решение задач для жестких систем в заданной точке с использованием алгоритма Булирша–Штера.

Несмотря на различные методы поиска решения, каждая из этих функций требует, чтобы были заданы, по крайней мере, следующие величины:

- начальные или граничные условия;
- набор точек, в которых нужно найти решение;
- само дифференциальное уравнение, записанное в некотором специальном виде (символ ‘, обозначающий производную, ставится с использованием клавиш <Ctrl + F7>).

Для решения дифференциальных уравнений с начальными условиями система MathCad имеет ряд встроенных функций:

rkfixed – функция для решения ОДУ и систем ОДУ методом Рунге–Кутты четвертого порядка с постоянным шагом;

Rkadapt – функция решения ОДУ и систем ОДУ методом Рунге–Кутты с переменным шагом.

Например, стандартная функция **rkfixed(y, x1, x2, p, D)** имеет аргументы:

y – вектор начальных условий из **k** элементов (**k** – количество уравнений в системе);

x1 и **x2** – левая и правая границы интервала, на котором ищется решение ОДУ или системы ОДУ;

p – число точек внутри интервала (**x1, x2**), в которых ищется решение;

D – вектор, состоящий из **k**-элементов, который содержит первую производную искомой функции или первые производные искомым функций, если речь идет о решении системы.

Результатом работы функции является матрица из $p + 1$ строк, первый столбец которой содержит точки, в которых получено решение, а остальные столбцы – сами решения.

При решении дифференциального уравнения первого порядка нужно создать вектор начальных условий из одного элемента Y_1 , который затем используется при формировании вектора-функции правой части дифференциального уравнения. При обращении к функции **rkfixed** указываются имя вектора **Y**, границы интервала, на котором ищется решение уравнения, например (0; 2), количество точек, в которых ищется решение, – 100, вектор-функция, описывающая правую часть дифференциального уравнения, – **D**. В результате получается матрица **z**, в первом столбце которой содержатся значения аргумента искомой функции, во втором – значения самой результирующей функции. При построении графика функции первый столбец полученной матрицы указывается как аргумент, второй – как функция.

Технология решения дифференциальных уравнений первого порядка, используя функцию **rkfixed**:

– сформировать вектор начальных условий из одного элемента, присвоив начальное значение искомой функции переменной с индексом, например: $Y_1 :=$ или $Y_0 :=$ (в зависимости от значения переменной **ORIGIN**);

– определить вектор-функцию из одного элемента, которая содержит



первую производную неизвестной функции:

а) набрать имя функции с двумя параметрами: первый параметр – аргумент искомой функции (независимая переменная), второй – имя вектора, содержащего искомую функцию (можно использовать имя вектора начальных условий), например, $D(x, Y)$;

б) набрать оператор «:=» и выражение для первой производной (выразить из дифференциального уравнения), в котором вместо имени искомой функции подставлен первый элемент вектора-параметра, например, для уравнения $y'(3 \cdot x - y) = y$ вектор-функция будет определяться следующим

образом: $D(x, Y) := \frac{Y_1}{3 \cdot x - Y_1}$ (если **ORIGIN** = **0**, подставлять Y_0);

– присвоить некоторой переменной значение функции *rkfixed*, указав в скобках следующие параметры:

а) имя вектора начальных условий;

б) левая граница интервала, на котором ищется решение, в виде числовой константы;

в) правая граница интервала, на котором ищется решение, в виде числовой константы;

г) количество точек, в которых ищется решение;

д) имя вектора-функции, описывающего первую производную, без параметров.

Например: $Z := rkfixed(Y, 0.2, 5, 1000, D)$ (в результате получится матрица Z , в первом столбце которой содержатся значения аргумента искомой функции, во втором – значения самой функции);

– вывести матрицу, содержащую решение ДУ с помощью оператора «=», например: $Z =$;

– построить график найденной функции, указав в качестве аргумента по оси абсцисс столбец $Z^{<1>}$, а в качестве значения функции по оси ординат столбец $Z^{<2>}$ (если **ORIGIN** = **0**, набирать соответственно $Z^{<0>}$ и $Z^{<1>}$).

Пример 1 – Найти численное решение дифференциального уравнения первого порядка $y'(3 \cdot x - y) = y$ на интервале от 0,2 до 5 в 1000 точек при начальном условии $y(0) = 0,1$, выполнить графическую интерпретацию результатов (рисунок 7.1).

Технология решения дифференциальных уравнений второго порядка и выше, используя функции *rkfixed*:

– представить ОДУ в виде системы ОДУ первого порядка (в стандартной форме: обозначив значение первой производной $Y_1 :=$ и выразив вторую производную через значение первой производной и искомой функции Y_0 (значение переменной **ORIGIN** = **0**));

– сформировать вектор-столбец правых частей системы уравнений – D ;

– сформировать вектор начальных условий;

– присвоить некоторой переменной значение функции *rkfixed*, указав в скобках необходимые параметры (в результате получится матрица Z , в первом



столбце которой содержатся значения аргумента искомой функции, во втором – соответствующие приближенные значения самой функции, а в третьем – значения производной);

– вывести матрицу, содержащую решение ДУ с помощью оператора «=», например: $Z =$;

– построить график найденной функции, указав в качестве аргумента по оси абсцисс столбец $Z^{<0>}$, а в качестве значения функции по оси ординат столбцы $Z^{<1>}$ и $Z^{<2>}$.

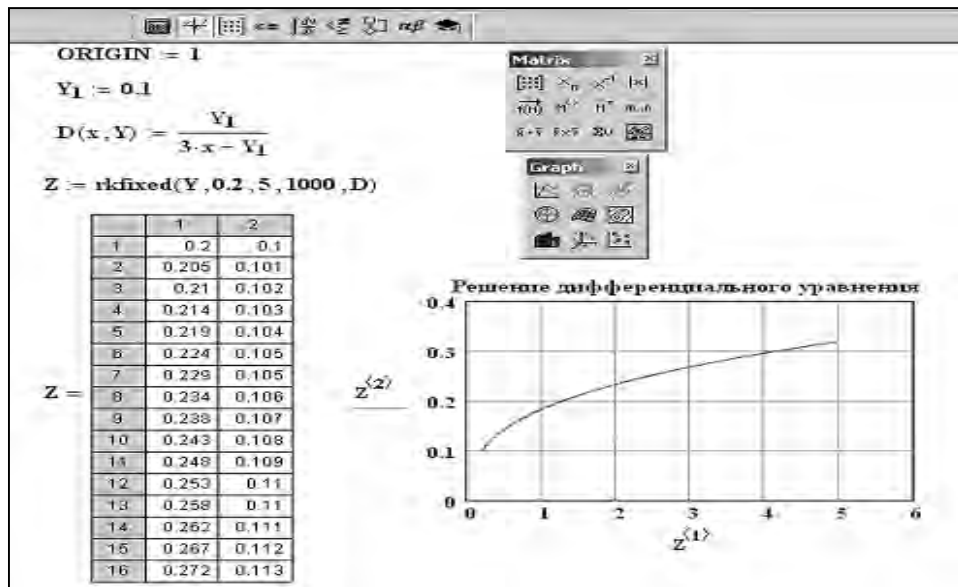


Рисунок 7.1 – Реализация примера 1

Пример 2 – Найти численное решение ОДУ второго порядка $y'' + 5y' - 1.7y = 5x^2$ при начальных условиях $y(0) = 0$ и $y'(0) = 1$, используя функции **rkfixed** (рисунок 7.2).

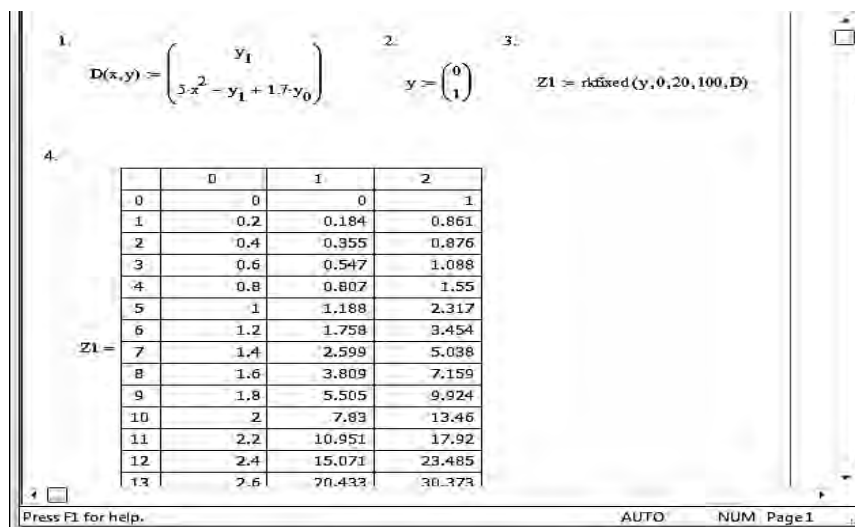


Рисунок 7.2 – Реализация примера 2

Порядок выполнения работы.

- 1 Выполнить вышеразобранные примеры.
- 2 По заданию преподавателя решить уравнения.

Контрольные вопросы

- 1 Какие функции имеются в пакете MathCad для решения дифференциальных уравнений и их систем?
- 2 Формат стандартной функции *rkfixed*.
- 3 Технология решения дифференциального уравнения первого порядка.
- 4 Технология решения дифференциального уравнения второго порядка и выше.

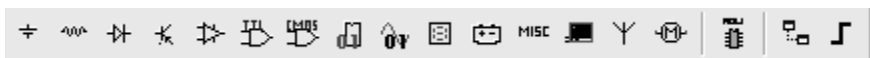
8 Лабораторные работы № 8–10. Система моделирования Multisim

Цель работы: научиться использовать возможности программы Multisim для изучения свойств электронных приборов и электронных аналоговых устройств.

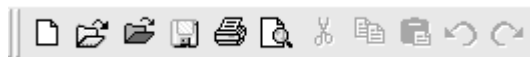
Методические указания

Multisim – это программа моделирования и расчета электронных и электрических схем устройств. Широкий набор приборов позволяет задавать входные воздействия, производить измерения различных величин и строить графики. Все приборы изображаются максимально приближенными к реальным. Интерфейс Multisim состоит из следующих базовых элементов.

Панель компонентов схем – ряд кнопок с графическим обозначением (пиктограммами) электронных приборов и микросхем.



Стандартная панель – ряд кнопок, позволяющий выполнять стандартные операции с файлом, например, печать.



Панель Вид – ряд кнопок, позволяющий увеличивать или уменьшать изображение электрической схемы в **Окне схемы**.



Строка меню – позволяет выполнять различные действия с файлом, загруженным в **Окно схемы**.

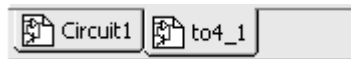
Панель инструментов – размещена справа от окна схемы, на ней расположены пиктограммы приборов, необходимых для исследования различных электронных устройств (например, мультиметр, осциллограф).



Строка состояния - находится в нижней части рабочего окна.

Run / resume simulation

Закладка активной схемы – расположена под окном схемы, на ней отображается название загруженного в окне схемы проекта.



В **Окно схемы** загружается схема готового проекта или создается новая схема из компонентов, которые находятся в **проводнике компонентов**, измерительных и анализирующих приборов. Схемы исследования электронных приборов обязательно содержат один или два источника питания (DC Supply), источники переменного напряжения, полупроводниковые приборы, резисторы, конденсаторы и измерительные приборы: вольтметры, миллиамперметры, мультиметр, осциллограф, характериограф.

Multisim оперирует с двумя категориями **компонентов: виртуальными (virtual) и реальными (real)**.

У реальных компонентов схемы в программе **Multisim** есть определенные и неизменяемые параметры, у виртуальных – можно назначать свои параметры.

Название компонента, марка, графическое обозначение отображаются в **Проводнике компонентов** (рисунок 8.1).

Вызывается **Проводник компонентов** щелчком ПК мыши на элементе, в окне **Group** находится закладка с нужным видом элементов и выделяется марка из списка **Component**, после чего нажимается кнопка **ОК**. Элемент выбранной марки загружается в **Окно схемы**.

Подобным же образом можно изменять марку любых компонентов схемы.

Словарь терминов



BJT – биполярный транзистор ;	MOSFET – МОП – транзистор;
Capacitor – конденсатор;	Inductor – индуктивность;
LED – светодиод;	Diode – диод;
Potentiomete – потенциометр;	Resistor – резистор;
Value – величина;	Frequency – частота;
Supply – источник питания;	Source – источник;
АС – переменное напряжение/ток;	DC – постоянное напряжение/ток.



Рисунок 8.1 – Проводник компонентов

Для исследования свойств электронных приборов и электронных устройств используются источники постоянного напряжения (источники питания – DC Supply) и источники переменного напряжения (AC Voltage Source). В процессе выполнения лабораторных работ возникает необходимость устанавливать величину питающего напряжения, амплитуду и частоту переменного напряжения. Для этого нужно двойным щелчком ЛК мыши на источнике войти в меню источника напряжения.

В закладке Voltage (параметры) в меню источника питания устанавливается требуемое постоянное напряжение. В аналогичной закладке меню источника переменного напряжения (AC Voltage Source) устанавливается значение амплитуды переменного напряжения и частоты.

Двойным щелчком ЛК мыши на графическом изображении элемента можно войти в его меню и установить свой или выбрать номинальный параметр. Например, для изменения сопротивления резистора необходимо войти в его меню, в закладке Resistance развернуть перечень номинальных значений сопротивлений и выбрать требуемый номинал сопротивления. После чего нажать кнопку **ОК**. Аналогично устанавливается и величина емкости конденсатора и индуктивности катушки. Меню электронных приборов позволяет увидеть числовые значения основных параметров данной модели. Для присоединения элемента проводниками в схему необходимо навести курсор на вывод компонента, зажать ЛК мыши и тянуть к предполагаемому месту соединения. В месте соединения щелкнуть клавишей мыши для установления контакта, отображаемого на схеме точкой. Для отсоединения компонента от схемы необходимо выделить ЛК мыши соединительный проводник, нажатием клавиши «delete» удалить его. Для проведения процесса моделирования (эмуляции) и получения показаний измерительных приборов нажать на значок «выключатель», переведя его из позиции 0 в позицию 1 . Аналогично запуск моделирования можно производить клавишей с пиктограммой в виде зеленого треугольника .

Выключение – клавишей с пиктограммой в виде красного квадрата. Есть возможность останавливать процесс моделирования, нажав на паузу.

Примечание – Для произведения каких-либо изменений в схеме необходимо выключить процесс эмуляции.

Наиболее часто при выполнении лабораторных работ в программе Multisim будут использоваться следующие виртуальные измерительные приборы.

Мультиметр – многофункциональный виртуальный измерительный прибор, который можно использовать как амперметр, вольтметр и омметр в цепи постоянного или переменного тока.

Функциональный генератор – виртуальный прибор для получения сигналов гармонических и импульсных (режим прямоугольных и треугольных импульсов) с заданными параметрами. Дважды щелкните на «генераторе функций» для настройки его параметров. Панель управления функционального генератора содержит следующие закладки: Frequency (Частота), Duty Cycle (Производительность), Amplitude (Амплитуда), Offset (Постоянная составляющая). Waveform – форма колебаний, выбирается одной из кнопок верхнего ряда на панели управления.

Оциллограф (осциллоскоп) – виртуальный многофункциональный измерительный прибор, который дает возможность визуально наблюдать сигналы, измерять напряжения, разность фаз, период сигнала и временные интервалы. Двойным щелчком мыши войдите в меню **Oscilloscope** (осциллограф). Необходимо установить значение масштаба по временной оси и масштаб амплитуд (**Amplitude**). Далее запускается процесс моделирования и наблюдается сигнал.

Анализатор характеристик (характериограф) – виртуальный прибор для получения статических вольт-амперных характеристик электронных приборов.

Порядок выполнения работы.

1 Собрать при помощи программного пакета Multisim схему, приведенную на рисунке 8.2, выбрав необходимый вариант из таблицы 8.1.

2 Произвести расчет указанных в таблице 8.2 необходимых параметров схемы.

3 Включить моделирование схемы и произвести измерения параметров схемы, указанных в таблице 8.2.

4 Сравнить расчетные и экспериментальные данные.

5 Выполнить аналогичные действия для схем на рисунке 8.3 (таблицы 8.1 и 8.3) и рисунке 8.4 (таблицы 8.4 и 8.5).

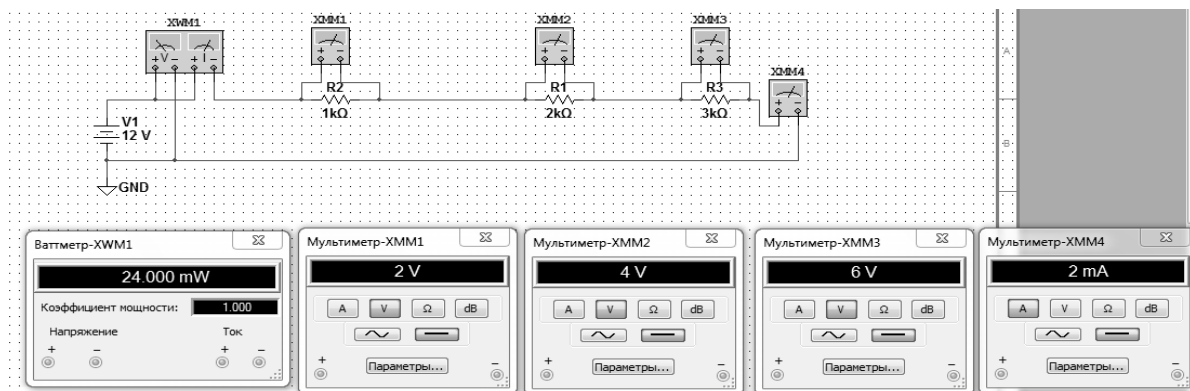


Рисунок 8.2 – Последовательное соединение сопротивлений

Таблица 8.1 – Варианты параметров схемы первого и второго заданий

Номер варианта	Напряжение питания U , В	Сопротивление резистора R_1 , Ом	Сопротивление резистора R_2 , Ом	Сопротивление резистора R_3 , Ом
1	10	10	20	30
2	11	11	22	33
3	12	12	24	36
4	13	13	27	39
5	14	15	30	43
6	15	15	33	47
7	16	16	36	51
8	17	18	39	56
9	18	20	43	62
10	19	22	47	68
11	20	24	30	75
12	21	27	33	82
13	22	30	36	91
14	23	33	39	100
15	24	36	43	110

Таблица 8.2 – Расчетные и измеренные данные параметров первой схемы

Параметры	Расчетные данные	Результаты измерений
Общее сопротивление цепи $R_{общ}$, Ом		
Ток I , А/мА		
Напряжение на резисторе R_1 , В		
Напряжение на резисторе R_2 , В		
Напряжение на резисторе R_3 , В		
Общая мощность, потребляемая резисторами R_1, R_2, R_3 , Вт		

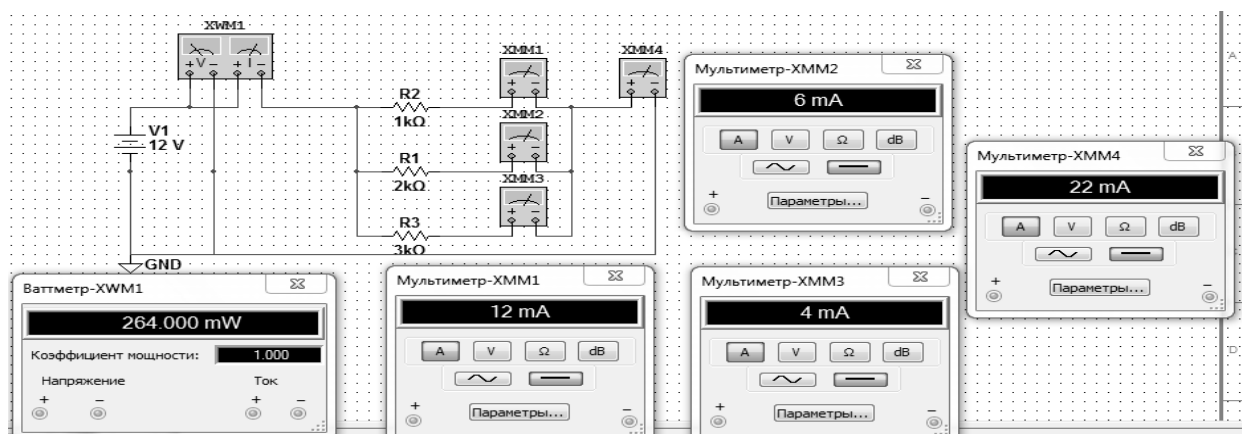


Рисунок 8.3 – Параллельное соединение сопротивлений

Таблица 8.3 – Расчетные и измеренные данные параметров второй схемы

Параметры	Расчетные данные	Результаты измерений
Общее сопротивление цепи $R_{общ}$, Ом		
Ток I , А/мА		
Ток через резистор R_1 , А		
Ток через резистор R_2 , А		
Ток через резистор R_3 , А		
Общая мощность, потребляемая резисторами R_1, R_2, R_3 , Вт		

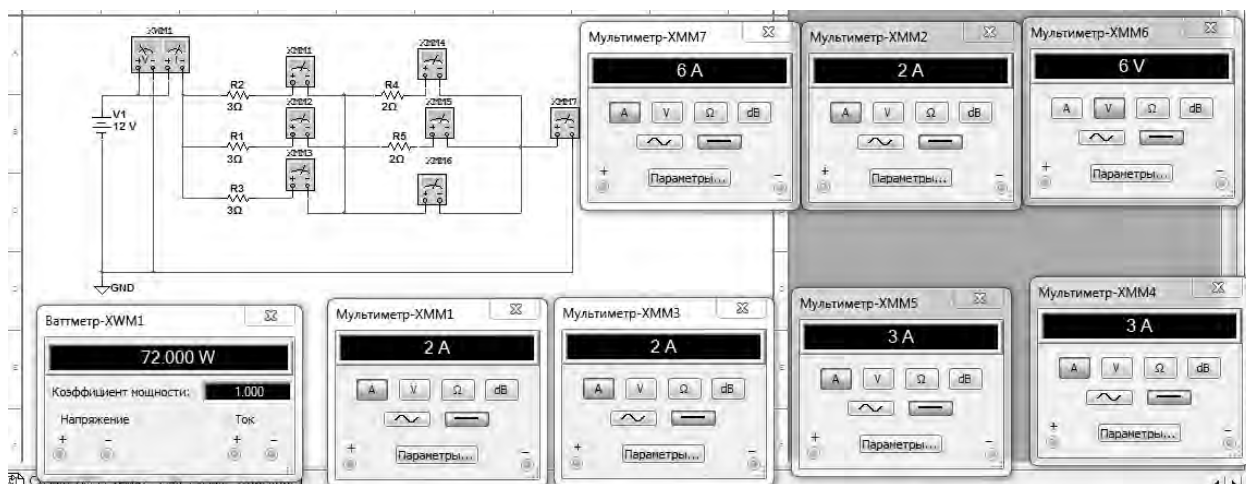


Рисунок 8.4 – Смешанное соединение сопротивлений

Таблица 8.4 – Варианты параметров схемы третьего задания

Номер варианта	Напряжение питания U , В	R_1 , Ом	R_2 , Ом	R_3 , Ом	R_4 , Ом	R_5 , Ом
1	10	10	20	30	20	30
2	11	11	22	33	22	33
3	12	12	24	36	24	36
4	13	13	27	39	27	39
5	14	15	30	43	30	43
6	15	15	33	47	33	47
7	16	16	36	51	36	51
8	17	18	39	56	39	56
9	18	20	43	62	43	62
10	19	22	47	68	47	68
11	20	24	30	75	51	75
12	21	27	33	82	56	82
13	22	30	36	91	62	91
14	23	33	39	100	68	100
15	24	36	43	110	75	110

Таблица 8.5 – Расчетные и измеренные данные параметров третьей схемы

Параметры	Расчетные данные	Результаты измерений
Общее сопротивление цепи $R_{общ}$, Ом		
Ток I , А/мА		
Ток через резистор R1, А		
Ток через резистор R2, А		
Ток через резистор R3, А		
Ток через резистор R4, А		
Ток через резистор R5, А		
Общая мощность, потребляемая резисторами R1, R2, R3, R4, R5, Вт		

Контрольные вопросы

- 1 Какие виртуальные приборы программного пакета Multisim необходимы для выполнения лабораторных работ?
- 2 Какие измерения можно производить при помощи виртуального мультиметра из программного пакета Multisim?
- 3 Сформулируйте закон Ома для участка цепи.
- 4 Как определить общее сопротивление последовательно включенных резисторов?
- 5 Как определить общее сопротивление параллельно включенных резисторов?

9 Лабораторная работа № 11. Программирование на Си. Интерфейс Microsoft Visual Studio 2010. Выполнение программ

Цель работы: ознакомиться со средой разработки Microsoft Visual C++ 2010, научиться создавать, компилировать и отлаживать приложения, разобраться со структурой программы на языке C++.

Методические указания

Среда разработки **Microsoft Visual Studio** – это набор инструментов и средств, предназначенных для разработчиков программ, с широким набором поддерживаемых языков программирования, в том числе и C++.

Назначение среды разработки программ ясно следует из ее названия. Естественно, что любая программа сначала должна быть спроектирована, затем переложена на выбранный разработчиком для ее реализации язык программирования, после чего средствами конкретного языка программирования подвергается преобразованию в код, понятный микропроцессору. Сам по себе микропроцессор «не знает» ни одного из известных нам языков программирования, кроме языка машинных кодов.



Выделим основные этапы создания программы.

- 1 Разработка (проектирование).
- 2 Выбор языка программирования.
- 3 Написание текста программы.
- 4 Перевод в язык машинных кодов (компиляция).
- 5 Отладка.
- 6 Выполнение.

Запуск среды осуществляется через соответствующий пункт меню «Пуск».

При запуске на экране появляется окно среды разработки, так называемая «Начальная страница» (Start Page), которая позволяет получить быстрый доступ к наиболее часто используемым возможностям, таким как, например, открытие недавно созданных проектов или создание нового проекта.

Для начала работы необходимо создать новый проект. В данном случае это будет консольное приложение для платформы Win32, выводящее на экран сообщение «Привет Мир!» (Win32 – это 32-битное приложение для ОС Windows).

Для создания проекта нужно выбрать пункт меню среды разработки **File – New – Project** или нажать комбинацию клавиш **Ctrl+Shift+N**. При этом появится диалоговое окно **New Project** (рисунок 9.1), позволяющее создать все типы проектов Visual Studio.

В начале необходимо выбрать тип проекта. В данном случае создать проект Visual C++. Далее выбираем необходимый подтип Win32 и шаблон приложения Win32 Console Application. После этого вводим имя приложения hello, проверяем месторасположение каталога с файлами проекта (или изменяем его при помощи кнопки «Browse ...»). Далее нажимаем «OK» и попадаем в окно мастера создания приложений Application Wizard, представленного на рисунке 9.2. Нажимаем кнопку Next для уточнения параметров приложения. На рисунке 9.3 представлены возможные варианты модификации параметров приложения Win32.

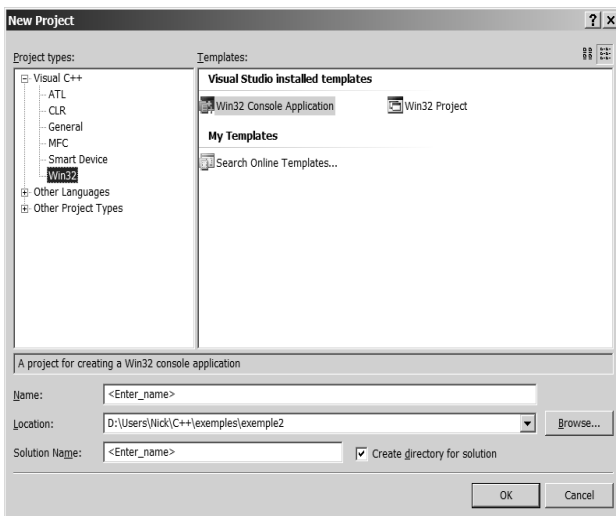


Рисунок 9.1 – Окно New Project



Рисунок 9.2 – Окно мастера создания приложений



В дополнительных опциях проекта **Additional options** щелкаем мышью по текстовому полю **Empty project** для создания пустого проекта и нажимаем на кнопку **Finish**. В результате в проводнике решений (**Solution Explorer**) (рисунок 9.4) появятся все файлы, связанные с данным проектом. Файлы разделены на несколько групп: файлы заголовков (Header Files), файлы ресурсов (Resource Files) и файлы исходного кода (Source Files). Все перечисленные папки пусты.

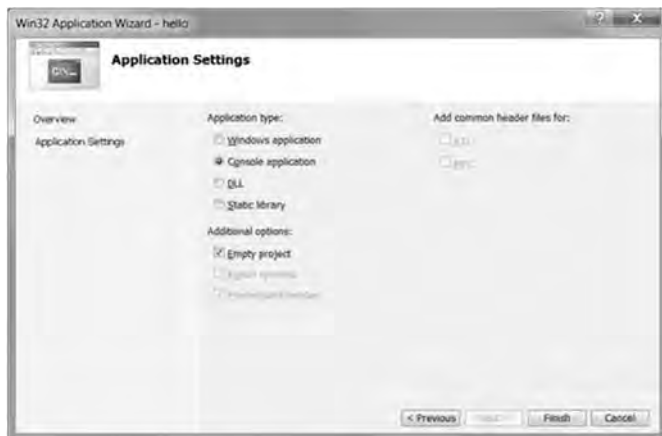


Рисунок 9.3 – Параметры приложения Win32



Рисунок 9.4 – Вид среды после создания консольного приложения

Для работы с консольным приложением необходимо создать новый файл для набора текста программы. С этой целью:

- на проводнике решений (**Solution Explorer**) найти строку с именем проекта (**hello**) и щелкнуть по ней мышью;
- выполнить команду **Project – Add New Item ...**. В появившемся диалоговом окне (рисунок 9.5) в поле **Name** ввести имя файла. Для удобства желательно ввести имя, совпадающее с именем проекта;
- щелкнуть мышью по кнопке **Add**. В результате в папке **Source Files** появится файл **hello.cpp**, а в окне редактора – закладка с именем **hello.cpp**, в которой набирают текст программы.

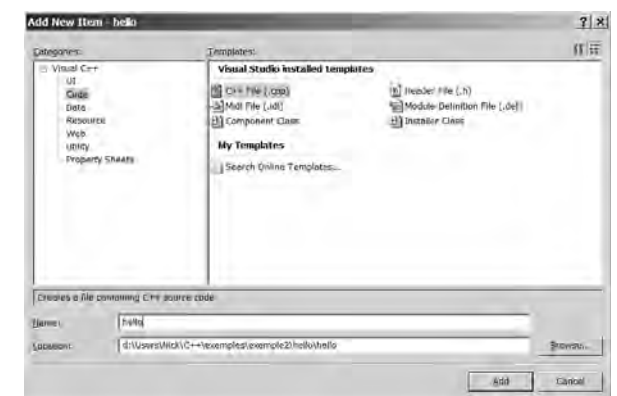


Рисунок 9.5 – Окно создания нового файла, содержащего источник кода C++

Теперь напишем первое приложение на языке C++, скомпилируем его и запустим на выполнение.

Откройте в окне редактора файл `hello.cpp` (дважды кликнув на нем в окне проводника решений) и введите текст программы, представленный далее.

```
/*Первая программа на C++*/
#include "stdafx.h"
#include <conio.h>
#include <stdio.h>
int main()
{
    printf("Hello!!!\n");// вывод на экран текста Hello !!!
    _getch();          //задержка ввыводы текста на экран
}
```

Для компиляции и запуска приложения существует несколько способов. Это и соответствующие пункты меню, и кнопки на панели инструментов, и сочетания горячих клавиш, ускоряющих работу программиста. Рассмотрим доступные варианты подробнее (в скобках указано сочетание клавиш).

Build Solution (<F7>) – собрать проект. При этом перекомпилируются все файлы проекта.

Rebuild Solution (<Ctrl + Alt + F7>) – пересобрать проект.

Clean Solution – очистить проект. При этом удаляются все лишние файлы, необходимые на момент разработки и отладки, но не нужные в конечном продукте.

Compile (<Ctrl + F7>) – скомпилировать проект. При этом перекомпилируются только измененные файлы проекта.

Start Debugging (<F5>) – начать отладку. Запускает программу под отладчиком.

Start without Debugging (<Ctrl + F5>) – запустить без отладчика. Просто осуществляется запуск откомпилированной программы.

Step Into (<F11>) – пошаговое выполнение с заходом в процедуру.

Step Over (<F10>) – пошаговое выполнение без захода в процедуру.

Toggle Breakpoint (<F9>) – установить/снять точку останова.

Breakpoints (<Alt + F9>) – показать текущие точки останова.

Порядок выполнения работы.

1 Изучить возможности интегрированной среды разработки Microsoft Visual C++.

2 Создать новый проект консольного приложения.

3 Набрать, отладить простейшую консольную программу, выводящую несколько символов на экран.

4 Проверить работоспособность программы.

5 Модифицировать программу, изменив, например, текст выводимого сообщения.



6 Отладить и протестировать программу.

7 Оформить отчёт.

Контрольные вопросы

1 Назовите основные этапы создания проекта.

2 Как можно просмотреть вывод Вашей программы?

3 Как компилировать и выполнять программу в IDE MS Visual C++?

4 Как получить контекстно-зависимую подсказку?

10 Лабораторная работа № 12. Программирование линейных вычислительных процессов на Си

Цель работы: освоить простейшую структуру программы на языке Си. Получить навыки в организации ввода-вывода на языке Си.

Методические указания

Если в программе все операторы выполняются последовательно, один за другим, такая программа называется линейной. Рассмотрим в качестве примера программу, вычисляющую результат по заданной формуле.

Например, надо написать программу, которая переводит температуру в градусах по Фаренгейту в градусы Цельсия по формуле $C = 5/9 (F - 32)$, где C – температура по Цельсию, а F – температура по Фаренгейту.

Перед написанием любой программы надо четко определить, что в нее требуется ввести и что мы должны получить в результате.

В данном случае:

– в качестве исходных данных выступает одно вещественное число, представляющее собой температуру по Цельсию;

– в качестве результата — другое вещественное число.

Перед написанием программы откроем интегрированную среду Visual C++: Пуск/Программы/Microsoft Visual Studio/ Microsoft Visual C++.

Далее создадим проект. Для этого:

1) File > New ...;

2) в открывшемся окне:

– выберите тип Win32 Console Application;

– введите имя проекта в текстовом поле Project Name;

– введите имя каталога размещения файлов проекта в текстовом поле

Location, например G:/ASOIZ/;

– щелкните левой кнопкой мыши на кнопке ОК.

Открывается диалоговое окно Win32 Console Application – Step1 of 1 и в нем:

– выберите тип An empty project;

– щелкните на кнопке Finish.



После щелчка появится окно New Project, в котором щелкните на кнопке ОК.
Далее создаем файл:

- 1) File > New В результате откроется диалоговое окно New;
- 2) на вкладке Files:
 - выберите тип файла (в данном случае **C++ Source File**);
 - в текстовом поле File Name введите нужное имя файла;
 - флажок **Add to project** должен быть включен;
 - щелкните на кнопке ОК.

Набираем следующий текст программы:

```
#include "stdafx.h"
#include <iostream> //1
#include <conio.h>
#include <cstdlib>
#include <locale.h>
int main()
{
    using namespace std
    setlocale(LC_ALL, "Rus");
    float fahr,cels; //2
    cout <<"Введите температуру по Фаренгейту" ; //3
    cin>>fahr; //4
    cels=(5.0/9.0)*(fahr-32.0); //5
    cout <<"По Фаренгейту:"<<fahr<<"\nВ градусах Цельсия:"<<cels;//6
    getch();
    return 0;
}
```

Рассмотрим каждую строку программы отдельно.

В начале программы записана директива препроцессора, по которой к исходному тексту программы подключается заголовочный файл **<iostream.h>**. Это файл, который содержит описания операторов ввода-вывода **cin** и **cout**.

Любая программа на C++ состоит из функций, одна из которых должна иметь имя **main**, указывающее, что именно с нее начинается выполнение программы. После круглых скобок в фигурных скобках { } записывается тело функции, т. е. те операторы, которые требуется выполнить.

Любая заготовка при написании программы имеет вид:

```
#include <...>
#include <...>
int main()
{
    объявление переменных;
    ввод исходных данных;
    расчет результата;
    вывод результата;
    return 0;
}
```



Для хранения исходных данных и результатов надо выделить достаточно места в оперативной памяти. Для этого служит оператор 2. В нашей программе требуется хранить два значения: температуру по Цельсию и температуру по Фаренгейту, поэтому в операторе определяются две переменные. Одна, для хранения температуры по Фаренгейту, названа *fahr*, другая (по Цельсию) – *cels*. Имена переменным дает программист исходя из их назначения. Имя может состоять только из латинских букв, цифр и знака подчеркивания и должно начинаться не с цифры. При описании любой переменной нужно указать ее тип. Поскольку температура может принимать не только целые значения, для переменных выбран вещественный тип **float**. Основные типы: *int (short, unsigned)* – целочисленные, *float (double, long double)* – вещественные, *char* – символьный, *bool* – логический

Для того чтобы пользователь программы знал, в какой момент требуется ввести с клавиатуры данные, применяется так называемое приглашение к вводу (оператор 3). На экран выводится указанная в операторе *cout* строка символов.

В операторе 4 выполняется ввод с клавиатуры одного числа в переменную *fahr*. Для этого используется стандартный объект *cin* и операция извлечения (чтения) `>>`. Если требуется ввести несколько величин, используется цепочка операций `>>`.

В операторе 5 вычисляется выражение, записанное справа от операции присваивания (обозначаемой знаком =), и результат присваивается переменной *cels*, то есть заносится в отведенную этой переменной память. Сначала целая константа 5 будет поделена на целую константу 9, затем результат этой операции умножен на результат вычитания числа 32 из переменной *fahr*.

Для вывода результата в операторе 6 применяется объект *cout*. Выводится цепочка, состоящая из пяти элементов. Это строка "По Фаренгейту":, значение переменной *fahr*, и в новой строке "n\В градусах Цельсия:", значение переменной *cels*.

Последний оператор (оператор 7) этой программы предназначен для возврата из нее и передачи значения во внешнюю среду.

Далее компилируем программу. Для этого нажимаем кнопку на панели инструментов либо комбинацию клавиш *Ctrl + F7*.

Для запуска программы нажимаем кнопку на панели инструментов либо комбинацию клавиш *Ctrl + F5*.

Для использования математических функций необходимо подключить к программе заголовочный файл `<math.h>`. Для подключения русского языка надо добавить следующую библиотеку: `#include <locale.h>`, которая отвечает за локализацию, а затем в начале тела кода ввести эту строку: `setlocale(LC_ALL, "Rus");`

Порядок выполнения работы.

- 1 Изучить приведенный пример программы.
- 2 Написать программу для расчета по двум формулам (по заданию преподавателя). Предварительно подготовить тестовые примеры по второй формуле с помощью калькулятора (результат вычисления по первой формуле должен совпадать со второй).



Контрольные вопросы

- 1 Какую структуру имеет программа на языке Си++?
- 2 Назовите основные типы данных языка Си++.
- 3 Дайте определение понятию знак операции и опишите операции, используемые в программах.

11 Лабораторная работа № 13. Программирование разветвляющихся процессов

Цель работы: изучить операции сравнения, логические операции, операторы передачи управления **if**, **switch**, **break**. Написать и отладить программу разветвляющегося алгоритма в консольном приложении.

Методические указания

Оператор условной передачи управления **if** имеет следующий формат:

if (логическое выражение) оператор 1;
else оператор 2;

Если логическое выражение истинно, то выполняется оператор1, иначе – оператор2.

Общая форма оператора множественного выбора **switch** следующая:

```
switch(переменная выбора) {
case const 1: операторы 1 ; break;
...
case const N: операторы N; break;
default: операторы N+1;
}
```

При использовании оператора **switch** сначала анализируется переменная выбора и проверяется, совпадает ли ее значение со значением одной из констант. При совпадении выполняются операторы этого case. Конструкция **default** (может отсутствовать) выполняется, если результат выражения не совпал ни с одной из констант.

Операции сравнения применяются при работе с двумя операндами и возвращают **true** (1), если результат сравнения – истина, и **false** (0), если результат сравнения – ложь. В языке Си определены следующие операции сравнения: < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), != (не равно), = (равно).

Логические операции работают с операндами скалярных типов и возвращают результат булева типа. Существует три логические операции: ! – отрицание, или логическое НЕ; && – логическое И; || – логическое ИЛИ.



Пример – Вычислить значение выражения

$$s = \begin{cases} |f(x)| + \ln(y) & \text{при } |xy| > 10; \\ e^{f(x)+y} & \text{при } 5 < |xy| < 10; \\ \sin(x) + \operatorname{tg}(y) & \text{при } |xy| = 5. \end{cases}$$

При выполнении задания предусмотреть выбор вида функции $f(x)$, $\operatorname{sh}(x)$, x^2 или e^x .

Текст программы:

```
#include "Stdafx.h"
#include <fstream>
#include <iostream>
#include <conio.h>
#include <cstdlib>
#include <math.h>
#include <locale.h>
int main()
{
using namespace std;
setlocale(LC_ALL, "Rus");
double x,y,f,a,s;
int k;
cout<<"Введите x";cin>>x;
cout<<"Введите y";cin>>y;
cout<<"Введите f: 1-sh(x), 2-x^2, 3-exp(x)"; cin>>k;
switch(k)
{
case 1:f=sinh(x);break;
case 2:f=pow(x,2);break;
case 3:f=exp(x);break;
default: cout<<"Не введена функция"; return 1;
}
a=fabs(x*y);
if(a<5)
{
cout<<"Нет результата"<<endl;
return 1;
}
else
if(a>10) s=fabs(f)+log(y);
else
if(a<=10 && a>5) s=exp(f+y);
else s=sin(x)+tan(y);
cout<<"Результат="<<s<<endl;
return 0;
}
```



Порядок выполнения работы.

- 1 Изучить приведенный пример программы.
- 2 Написать и отладить программу по заданию преподавателя.

Контрольные вопросы

- 1 Для чего используется условный оператор *if* ?
- 2 Приведите синтаксис, структурную схему и примеры простого условного оператора.
- 3 Приведите синтаксис, структурную схему и примеры оператора выбора альтернатив.
- 4 Какие действия выполняет управляющий оператор *break*?

12 Лабораторная работа № 14. Программирование циклических процессов с использованием счетчиков на Си

Цель работы: изучить циклические операторы *while*, *do-while*, *for*, научиться использовать циклические алгоритмы.

Методические указания

Оператор цикла *for* имеет вид:

```
for (инициализирующее выражение; условие; инкрементирующее выражение)
{
    тело цикла
}
```

Инициализирующее выражение выполняется только один раз в начале цикла и, как правило, инициализирует счетчик цикла. Условие содержит операцию отношения, которая выполняется в начале каждого цикла. Если условие равно *1* (*true*), то цикл повторяется, иначе выполняется следующий за телом цикла оператор. Инкрементирующее выражение, как правило, предназначено для изменения значения счетчика цикла. Модификация счетчика происходит после каждого выполнения тела цикла.

Оператор цикла *while* с предусловием имеет вид:

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

Организует повторение операторов тела цикла до тех пор, пока условие истинно.

Оператор цикла *do – while* с постусловием

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

Организует повторение операторов тела цикла до тех пор, пока условие истинно.

Далее приведен исходный код программы, считающей сумму всех целых чисел от 1 до 1000.

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    setlocale(0, "");
    int i = 0; // инициализируем счетчик цикла.
    int sum = 0; // инициализируем счетчик суммы.
    while (i < 1000)
    {
        i++;
        sum += i;
    }
    cout << "Сумма чисел от 1 до 1000 = " << sum << endl;
    return 0;
}
```

Рассмотрим по порядку исходный код данной программы. Сначала инициализируется счетчик цикла и переменная, хранящая сумму чисел.

В данном случае обязательно необходимо присвоить счетчику цикла какое-либо значение, т. к. если не инициализировать счетчик цикла, то в него попадет «мусор» и компилятор в лучшем варианте выдаст ошибку.

Затем необходимо задать условие цикла — **«пока переменная i меньше 1000 – выполняй цикл»**. При каждой итерации цикла значение переменной-счетчика i увеличивается на единицу внутри цикла. Когда выполнится 1000 итераций цикла, счетчик станет равным 999 и следующая итерация уже не выполнится, поскольку 1000 не меньше 1000. Выражение $sum += i$ является укороченной записью $sum = sum + i$. После окончания выполнения цикла, выводим сообщение с ответом.

Цикл **do while** очень похож на цикл **while**. Единственное их различие в том, что при выполнении цикла **do while** один проход цикла будет выполнен независимо от условия. Решение задачи на поиск суммы чисел от 1 до 1000 с применением цикла **do while** следующее:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
```



```

setlocale(0, "");
int i = 0; // инициализируем счетчик цикла.
int sum = 0; // инициализируем счетчик суммы.
do { // выполняем цикл.
    i++;
    sum += i;
} while (i < 1000); // пока выполняется условие.
cout << "Сумма чисел от 1 до 1000 = " << sum << endl;
return 0;
}

```

Порядок выполнения работы.

- 1 Изучить приведенный пример программы.
- 2 Написать и отладить программу по заданию преподавателя.

Контрольные вопросы

- 1 Дайте определение понятию цикл.
- 2 Назовите составляющие части цикла.
- 3 Какие разновидности операторов цикла языка Си Вы знаете?

13 Лабораторная работа № 15. Использование подпрограмм и процедур

Цель работы: овладеть навыками программирования с использованием подпрограмм различных видов, овладеть навыками написания программ и обращения к ним, выбора параметров подпрограмм.

Методические указания

В отличие от других языков программирования высокого уровня в языке Си нет деления на процедуры, подпрограммы и функции. В Си программа строится только из функций. Наиболее часто приходится использовать функции ввода и вывода информации.

Вначале рассмотрим функцию, определяющую форматный вывод:

```
printf("управляющая строка", аргумент1, аргумент2, ... );
```

Управляющая строка содержит объекты трех типов: обычные символы, которые просто выводятся на экран дисплея, спецификации преобразования, каждая из которых вызывает вывод на экран значения очередного аргумента из последующего списка, и управляющие символы-константы.

Каждая спецификация преобразования начинается со знака % и заканчивается некоторым символом, задающим преобразования.

Символ преобразования связан с типом переменных. Приведем символы преобразования:



- 1) **d** – значением аргумента является десятичное целое число;
- 2) **o** – значением аргумента является восьмеричное целое число;
- 3) **x** – значением аргумента является шестнадцатеричное целое число;
- 4) **c** – значением аргумента является символ;
- 5) **s** – значением аргумента является строка символов;
- 6) **e** – значением аргумента является вещественное число в экспоненциальной форме;
- 7) **f** – значением аргумента является вещественное десятичное число с плавающей точкой;
- 8) **u** – значением аргумента является беззнаковое целое число;
- 9) **p** – значением аргумента является указатель (адрес).

Если после знака % записан не символ, то он выводится на экран. Функция *printf* использует управляющую строку, чтобы определить, сколько всего аргументов и каковы их типы.

Например, в результате работы программы получены переменная *i*, имеющая значение 100, и переменная *j*, имеющая значение 25. Обе переменные целого типа. Для вывода этих переменных на экран в виде *i = 100 j = 25* необходимо применить функцию

```
printf("i=%d j=%d",i,j);
```

Как было описано ранее, в кавычках задается формат вывода. Перед знаком % записываются символы, которые будут непосредственно выданы на экран. После знака % применена спецификация *d*, т.к. переменные *i* и *j* имеют целый тип. Сами *i* и *j* приведены через запятую в списке аргументов. Если после знака % стоит цифра, то она задает поле, в котором будет выполнен вывод числа. Приведем несколько функций *printf*, которые будут обеспечивать вывод одной и той же переменной *S* целого типа, имеющей значение 336.

Функция *printf*("%2d", *S*); выдает на экран: **336**.

В данном примере ширина поля (она равна двум) меньше, чем число цифр в числе 336, поэтому поле автоматически расширяется до необходимого размера.

Функция *printf*("%6d", *S*); выдаст на экран: **__336** (шесть позиций).

То есть в результате работы функции число сдвинуто к правому краю поля, а лишние позиции перед числом заполнены пробелами.

Функция *printf*("%-6d", *S*); выдаст на экран: **336__** (шесть позиций). Знак «минус» перед спецификацией приводит к сдвигу числа к левому краю поля.

Рассмотрим вывод вещественных чисел. Если перед спецификацией *f* ничего не указано, то выводится число с шестью знаками после запятой. При печати числа с плавающей точкой перед спецификацией *f* тоже могут находиться цифры.

Рассмотрим на конкретном примере три возможные ситуации:

- 1) **%6f** – печать числа с плавающей точкой в поле из шести позиций;
- 2) **%.2f** – печать числа с плавающей точкой с двумя цифрами после десятичной точки;



3) **%6.2f** – печать числа с плавающей точкой в поле из шести позиций и двумя цифрами после десятичной точки.

Например, в результате работы программы получены переменные вещественного типа **a = 3,687** и **b = 10,17**. Если для вывода значений использована функция `printf(“%7f %8f”,a,b);`, то результат будет представлен в виде строки: `__ 3.687 _ __ _10.17` (семь позиций) (восемь позиций).

Как видно из примера, лишние позиции заполняются пробелами. Если для вывода значений использована функция `printf(“%.2f %/2f”, a, b);`, то результатом будет строка: `3.69 10.17`, из которой следует, что в первом числе третья цифра после десятичной точки отброшена с округлением, т. к. указан формат числа с двумя цифрами после десятичной точки.

Если для вывода значений использована функция `printf(“%7.2f e”,a,b);`, то будет выведена строка: `__ _ 3.68 1.010000e+01` (семь позиций).

Поскольку для вывода значения переменной *b* применена спецификация *e*, то результат выдан в экспоненциальной форме. Следует отметить, что если ширина поля меньше, чем число цифр в числе, то поле автоматически расширяется до необходимого размера.

Как было отмечено ранее, в управляющей строке могут содержаться управляющие символьные константы. Среди управляющих символьных констант наиболее часто используются следующие:

- 1) **\a** – для кратковременной подачи звукового сигнала;
- 2) **\b** – для перевода курсора влево на одну позицию;
- 3) **\n** – для перехода на новую строку;
- 4) **\r** – для перевода курсора в начало текущей строки;
- 5) **\t** – для горизонтальной табуляции;
- 6) **\v** – для вертикальной табуляции.

Предположим, в результате работы программы переменная *i* получила значение 50. В результате записи инструкции вызова функции `printf(“\t ЭВМ\n%d\n”,i);` сначала выполнится горизонтальная табуляция (**\t**), т. е. курсор сместится от края экрана на восемь позиций, затем на экран будет выведено слово «ЭВМ», после этого курсор переместится в начало следующей строки (**\n**), затем будет выведено целое значение *i* по формату *d*, и окончательно курсор перейдет в начало новой строки (**\n**). Таким образом, результат работы данной функции на экране будет иметь вид:

```

----- ЭВМ
50

```

Для форматного ввода данных используется функция

`scanf(«управляющая строка», аргумент1, аргумент2,...);`

Если в качестве аргумента используется переменная, то перед ее именем записывается символ **&**.

Управляющая строка содержит спецификации преобразования и исполь-



зуется для установления количества и типов аргументов. Спецификации для определения типов аргументов такие же, как и для функции *printf*. Перед символами *d*, *o*, *x*, *f* может стоять буква *l*. В первых трех случаях соответствующие переменные должны иметь тип *long*, а в последнем *double*.

Рассмотрим пример. Требуется ввести значения для переменных *i* (целого типа) и *a* (вещественного типа). Эту задачу выполнит функция *scanf* (“%d%f”,&i,&a);

В управляющей строке спецификации трех типов могут быть отделены друг от друга различными знаками, в том числе и пробелом. Следовательно, при занесении значений переменных необходимо использовать указанный разделитель. Если спецификации не отделены одна от другой никакими значениями, то значения переменных заносятся через пробел.

В языке C++ есть две очень удобные функции: *puts* и *gets*, позволяющие вводить и выводить строку символов. Пример их использования показан далее:

```
#include<stdio.h>
main()
{
char q[40]; /*объявление строки символов*/
puts(“Введите строку символов”);
gets(q); /*ввод строки символов*/
puts(q); /*вывод строки символов*/
}
```

В результате работы программы вначале на экране появится текст: **Введите строку символов**, после чего следует ввести какую-либо строку символов. Эта информация при помощи оператора *gets* будет присвоена элементам символьного массива *q*. Оператор *puts* выведет строку символов.

Порядок выполнения работы.

- 1 Изучить приведенные примеры программ.
- 2 Написать и отладить программу по заданию преподавателя.

Контрольные вопросы

- 1 Какие объекты содержит управляющая строка функция, определяющая форматный вывод?
- 2 С какого знака начинается каждая спецификация преобразования управляющей строки функции, определяющей форматный вывод?



14 Лабораторная работа № 16. Использование функций

Цель работы: ознакомиться с особенностями применения функций в языке C++, с понятием прототипа и областью его применения, с понятием автоматических внешних, статических и регистровых переменных и их применением при составлении программ с использованием функций.

Методические указания

Программы на языке C++ обычно состоят из большого числа отдельных функций (подпрограмм). Как правило, они имеют небольшие размеры и могут находиться как в одном, так и в нескольких файлах.

Связь между функциями осуществляется через аргументы, возвращаемые значения и внешние переменные.

Вызов функции осуществляется следующим образом:

<тип функции >(параметр 1, параметр 2, ...);

Если функция имеет переменное число параметров, то вместо последнего из них указывается многоточие.

Передача одного значения из вызванной функции в вызвавшую происходит с помощью оператора возврата, который записывается в следующем виде:

return (выражение);

В этом случае значение выражения (в частном случае может быть просто переменная) передается в основную программу и подставляется вместо обращения к функции.

Пусть вызывающая программа обращается к функции следующим образом:

a = fun(b,c);

Здесь *b* и *c* – аргументы, значения которых передаются в вызываемую подпрограмму.

Если описание функции начинается так:

fun(i,j),

то переменные *i* и *j* получают значения *b* и *c* соответственно.

Пример 1 – Написать программу для вычисления числа вычитаний

$$C_n^m = \frac{n!}{m!(n-m)!}$$



Программа имеет следующий вид:

```
#include "stdafx.h"
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
using namespace std;
int fact(int); //объявление функции
int main()
{
    int n,m,c;
    cin>>n;
    cin>>m;
    cout<<"C="<<fact(n)/(fact(m)*fact(n-m));//вызов функции
    return 0;
}
int fact(int a)//определение функции
{
    int i,p=1;
    if(a==0||a==1)return 1;
    for(i=1;i<=a;i++)
        p=p*i;
    return p;//возврат p в точку вызова
}
```

Порядок выполнения работы.

- 1 Изучить приведенный пример программы.
- 2 Написать и отладить программу по заданию преподавателя.

Контрольные вопросы

- 1 Как осуществляется связь между функциями?
- 2 Как осуществляется вызов функции?
- 3 Как происходит передача одного значения из вызванной функции в вызвавшую?



15 Лабораторная работа № 17. Задание структурированных типов данных на языке Си

Цель работы: изучить правила создания и обработки данных структурного типа в среде Visual C++. Написать и отладить программу по созданию структур в консольном приложении.

Методические указания

Структуры – это составной объект, в который входят элементы любых типов, за исключением функций. В отличие от массива, все элементы которого однотипны, структура может содержать элементы разных типов:

```
struct [ имя_tuna ]
{ mun_1 элемент_1;
  mun_2 элемент_2;
  mun_n элемент_n;
} [ список_описателей ];
```

Элементы структуры называются *полями структуры* и могут иметь любой тип. Если отсутствует имя типа, должен быть указан список описателей переменных, указателей или массивов.

```
struct
{
  int year;
  char month[10];
  int day;
} date, date2;
```

Если список отсутствует, описание структуры определяет новый тип, имя которого можно использовать в дальнейшем наряду со стандартными типами, например:

```
struct student{ // описание нового tuna Worker
  char fio[30];
  long int num_zac;
  double sr_bal;
}; // описание заканчивается точкой с запятой
// определение массива tuna student и указателя на mun student.
student gr[30], *p;
```

Для переменных одного и того же структурного типа определена операция присваивания, при этом происходит поэлементное копирование. Структуру можно передавать в функцию и возвращать в качестве значения функции. Размер структуры не обязательно равен сумме размеров ее элементов, поскольку они могут быть выровнены по границам слова.

Доступ к полям структуры выполняется с помощью операций выбора . (точка) при обращении к полю через имя структуры и -> при обращении через указатель, например:

```
student student 1, gr[30], *p;
student .fio = "Страусенке";
gr[8] .sr_bal=5;
p->num_zac = 012001;
```

Если элементом структуры является другая структура, то доступ к ее элементам выполняется через две операции выбора:

```
struct A {int a, double x;};
struct B {A a, double x,} x[2];
x[0] .a. a = 1;
x[1]. x = 0.1;
```

Как видно из примера, поля разных структур могут иметь одинаковые имена, поскольку у них разная область видимости.

Пример – Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив ГАИ, состоящий из шести элементов типа АВТО;
- вывод на экран информации о владельцах автомобиля, марка которого вводится с клавиатуры;
- если таких авторов нет, то на экран дисплея вывести соответствующее сообщение.

Программа решения задачи будет иметь вид:

```
#include "stdafx.h"
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <locale.h>
using namespace std;
struct avto {
    char fam[25];
    char marka[20];
    char nomer[10];};
void output_gai(avto *,int );
void main(void)
{ setlocale(LC_ALL, "Rus");
    int i,n;
    bool flag;
    cout<<"Введите количество записей: ";
    cin>>n;
    avto *gai = new avto[n], temp;
    char m[20];
    for(i=0;i<n;i++)
    { // Ввод данных
        cout<<"Введите фамилию владельца "<<i+1;
```



```

cout<<"-го автомобиля: ";
cin >> gai[i].fam;
cout<<"Введите марку "<<i+1;
cout<<"-го автомобиля: ";
cin >> gai[i].marka;
cout<<"Введите номер "<<i+1;
cout<<"-го автомобиля: ";
cin >> gai[i].nomer; }
cout << endl <<"Введите марку автомобиля: ";
cin >> m;
cout << endl <<"Искомые автомобили" << endl;
flag = true;
for(i=0;i<n;i++)
    if(!strcmp(m,gai[i].marka))
    {    cout.setf(ios::left);
        cout.width(15);
        cout<<gai[i].fam;// *(gai+i)->fam;
        cout.setf(ios::right);
        cout.width(5);
        cout<<gai[i].nomer<<endl;
        flag = false; }
    if(flag)
    {cout<<"\nМарки "<<m;
      cout<<"нет в списках";}
delete [] gai;
}

```

Порядок выполнения работы.

- 1 Изучить приведенный пример программы.
- 2 Написать и отладить программу по заданию преподавателя.

Контрольные вопросы

- 1 Дайте определение понятию структура.
- 2 Назовите этапы определения объектов типа структуры.
- 3 В каком виде задается структурный тип данных? Приведите примеры.
- 4 Перечислите способы создания структурных переменных. Приведите примеры.

Список литературы

- 1 Информатика. Базовый курс: учебное пособие / Под ред. С. В. Симновича. – 3-е изд. – Санкт-Петербург: Питер, 2013. – 637 с.
- 2 Компьютерные информационные технологии: учебное пособие: в 3 ч. Ч. 1: Программное обеспечение / М. Н. Садовская [и др.]. – Минск: БГЭУ, 2014. – 287 с.

