Министерство образования и науки Республики Марий Эл
Государственное бюджетное профессиональное образовательное учреждение
Республики Марий Эл «Марийский политехнический техникум»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для студентов по выполнению практических работ

по дисциплине **ОП.05 Информационные технологии в профессиональной деятельности**

для специальности 08.02.09 Монтаж, наладка и эксплуатация электрооборудования промышленных и гражданских зданий

Йошкар-Ола, 2022

Методические указания по выполнению практических работ разработаны на основе рабочей программы учебной дисциплины ОП.05 Информационные технологии в профессиональной деятельности для специальности 08.02.09 Монтаж, наладка и эксплуатация электрооборудования промышленных и гражданских зданий

Организация-разработчик: ГБПОУ Республики Марий Эл «Марийский политехнический техникум»

Разработчик-составитель:
Лисин В.С., преподаватель ГБПОУ Республики Марий Эл «Марийский политехнический техникум»

СОДЕРЖАНИЕ

.

Пояснительная записка

Методические указания составлены в соответствии с рабочей программой учебной дисциплины ОП.05 Информационные технологии в профессиональной деятельности по специальности 08.02.09 Монтаж, наладка и эксплуатация электрооборудования промышленных и гражданских зданий и предназначены для выполнения практических работ по ОП.05 Информационные технологии в профессиональной деятельности по специальности всех форм обучения.

Программой учебной дисциплины предусмотрено 34 часа практических работ.

Дидактическая цель практических работ – осмыслить и закрепить материал лекций, сформировать умения применять полученные знания на практике, реализовать единства интеллектуальной и практической деятельности, развивать выработку при решении поставленных задач самостоятельность, ответственность, точность, творческую инициативу.

В данные рекомендации входят 16 практических работ, в каждой работе даются краткие методические указания, которые следует строго выполнять. Далее указаны номер, наименование и количество часов, отведенного на каждую работу.

В результате выполнения практических работ студент должен:

уметь:

– пользоваться пакетами специализированных программ для проектирования, расчета и выбора оптимальных параметров систем электроснабжения;

– выполнять расчеты электрических нагрузок;

– выполнять проектную документацию с учетом персонального компьютера;

знать:

– пакеты специализированных программ для расчета и проектирования систем электроснабжения;

– технические решения по применению микропроцессорной и микроконтроллерной техники в электроэнергетике;

– программирование микроконтроллеров.

Выполнение практических работ способствует формированию общих и профессиональных компетенций:

ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности
ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 07	Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
ОК 09	Использовать информационные технологии в профессиональной деятельности

ОК 10	Пользоваться профессиональной документацией на государственном и иностранных языках.
ПК 1.1.	Организовывать и осуществлять эксплуатацию электроустановок промышленных и гражданских зданий
ПК 2.3.	Организовывать и производить наладку и испытания устройств электрооборудования промышленных и гражданских зданий;
ПК 2.4.	Участвовать в проектировании силового и осветительного электрооборудования
ПК 3.2.	Организовывать и производить наладку и испытания устройств воздушных и кабельных линий;
ПК 3.3.	Организовывать и производить эксплуатацию электрических сетей;
ПК 3.4.	Участвовать в проектировании электрических сетей
ПК 4.3.	Участвовать в расчетах основных технико-экономических показателей.

Правила выполнения практических работ.

1. Практические работы проводятся в компьютерном классе на персональном компьютере.

2. Практические работы №1-№4 проводятся с использованием полнофункциональной или учебной версии среды NI Multisim 10 (MS10).

Прежде чем приступить к моделированию схем устройств, необходимо изучить теоретические сведения по теме практической работы, учебные задания на проведение экспериментов, согласно варианту рассчитать параметры элементов схемы, а в некоторых работах рассчитать схему цепи.

3. Практические работы №5-№8 рассчитаны на выполнение в русифицированной версии программы MathCAD, но их можно выполнять и в более ранних версиях программы.

4. Практические работы №9-№10 рассчитаны на выполнение в Компас-3D и Компас-Электрик.

5. Практические работы №11-№14 выполняются в Microsoft Visual Studio 2008.

6. Практические работы №15-№16 выполняются в среде AVR Studio.

7. При выполнении работ в компьютерном классе необходимо строго следовать методическим указаниям к конкретной практической работе и выполнять вариант задания (на расчет и эксперимент), номер N которого совпадает с номером записи фамилии студента в учебном журнале группы.

Критерии оценки выполнения практических заданий

Оценка «отлично» ставится, если студент выполнил работу в полном объеме с соблюдением необходимой последовательности действий; в ответе правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок.

Оценка «хорошо» ставится, если студент выполнил требования к оценке «отлично», но допущены 2-3 недочета.

Оценка «удовлетворительно» ставится, если студент выполнил работу не полностью, но объем выполненной части таков, что позволяет получить правильные результаты и выводы; в ходе проведения работы были допущены ошибки.

Оценка «неудовлетворительно» ставится, если студент выполнил работу не полностью или объем выполненной части работы не позволяет сделать правильных выводов.

Специальные условия для лиц с ОВЗ

При выполнении практических работ для студентов с нарушением зрения (слабовидящих) применяются следующие технические средства:

- электронный ручной видеоувеличитель Pebble 4.3;
- многофункциональный комплекс для детей с нарушением опорно-двигательного аппарата, слабовидящих и слабослышащих

Раздаточный материал выполнен шрифтом увеличенного размера.

Для слабослышащих используется персональная индукционная система Induction Порт.

Перечень практических работ

№ занятия	Наименование практической работы	Кол-во часов
Тема 1. Моделирование электрических цепей с помощью программы NI Multisim.		**8**
1.	Практическая работа № 1. Измерение электрических величин и параметров элементов электрических цепей в программе NI Multisim.	2
2.	Практическая работа № 2. Моделирование цепи постоянного тока.	2
3.	Практическая работа № 3. Моделирование цепи переменного тока	2
4.	Практическое занятие № 4. Моделирование логических схем.	2
Тема 2. Расчет электрических цепей с помощью программы Mathcad.		**8**
5.	Практическое занятие № 5. Запись математических выражений и вычисление их значений при заданных исходных данных в Mathcad.	2
6.	Практическое занятие № 6. Построение графиков.	2
7.	Практическое занятие № 7. Расчёт цепей постоянного тока. Сравнение результатов расчетов в Mathcad с результатами моделирования в NI Multisim.	2
8.	Практическое занятие № 8. Расчёт цепей переменного тока. Сравнение результатов расчетов в Mathcad с результатами моделирования в NI Multisim.	2
Тема 3. Построение схем в программе Компас-3D.		**6**
9.	Практическое занятие № 9. Основы работы с графическим редактором КОМПАС 3D. Выполнение плана освещения цеха.	3
10.	Практическое занятие № 10. Выполнение плана силовой сети цеха в графическом редакторе КОМПАС 3D.	3
Тема 4. Микропроцессоры и микроконтроллеры в электроэнергетике. Программирование микроконтроллеров.		**12**
11.	Практическое занятие № 11. Язык программирования СС++. Ввод - вывод данных. Первая программа.	2
12.	Практическое занятие № 12. Язык программирования СС++. Условный оператор.	2
13.	Практическое занятие № 13. Язык программирования СС++. Оператор цикла.	2
14.	Практическое занятие № 14. Язык программирования СС++. Массивы.	2
15.	Практическое занятие № 15. Основы программирования микроконтроллеров.	2
16.	Практическое занятие № 16. Средства отладки программного обеспечения.	2
	Всего:	**34**

Практическая работа № 1. Измерение электрических величин и параметров элементов электрических цепей в программе NI Multisim.

Цель занятия:

1. Ознакомиться с измерительными приборами, источниками питания и осциллографом программной среды NI Multisim.
2. Изучить методы и приобрести навыки измерения тока, напряжения, мощности, угла сдвига фаз между синусоидальным напряжением и током, а также сопротивлений резисторов, индуктивностей индуктивных катушек и емкостей конденсаторов.
3. Получить навыки построения электрических схем.

Теоретическое обоснование

Ниже кратко описываются виды и способы измерения электрических величин и параметров компонентов схем электронных устройств с помощью моделей измерительных приборов программной среды интерактивного моделирования и анализа электронных схем NI Multisim 10 (в дальнейшем, для краткости, эту систему будем называть среда MS10). Порядок установки параметров пассивных и активных элементов цепей, измерительных приборов и осциллографа приведен в приложении 1.

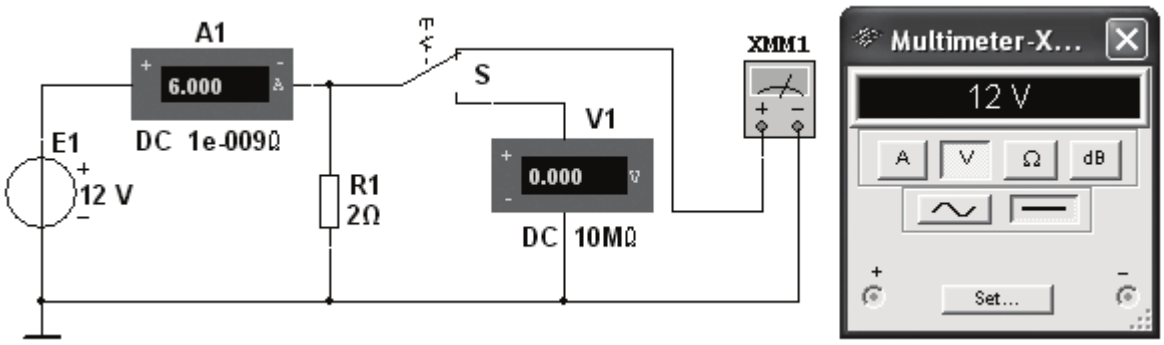
1. Измерение тока и напряжения

Мгновенные значения напряжения и тока можно измерить с помощью двухканального осциллографа **XSC1**, имитируемого программой MS10.

Измерение действующих значений напряжения и тока в ветвях электрической цепи проводится вольтметрами и амперметрами. Амперметр включается последовательно с элементами ветви, а вольтметр – параллельно участку цепи (рис. 1.1а и б), напряжение на котором необходимо измерить. Модели амперметров и вольтметров среды MS10 не требуют установки диапазона измерений.

Для установки режима работы и величин внутренних сопротивлений (Resistance) амперметров **A1**, **A2** и вольтметров **V1**, **V2** нужно дважды нажать на левую клавишу мыши (в дальнейшем дважды щелкнуть мышью) на изображении соответствующего прибора, в открывшемся диалоговом окне свойств прибора установить в команде **Mode** режим работы (постоянный ток **DC** или переменный **AC**), изменить или оставить установленное по умолчанию внутреннее сопротивление прибора (1 нОм для амперметров и 10 МОм для вольтметров) и нажать на кнопку **ОК** (Принять). Внутренние сопротивления 1 нОм для амперметров и 10 МОм для вольтметров, установленные по умолчанию, в большинстве случаев оказывают пренебрежительно малое влияние на работу схем.

а)



б)

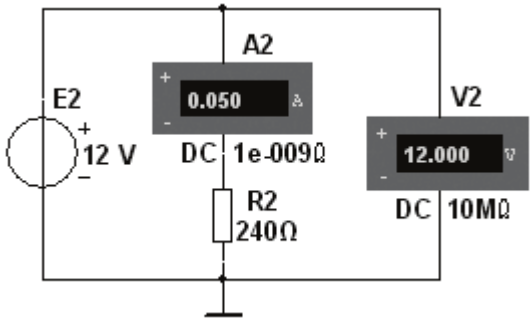


Рисунок 1.1

В библиотеке **Instruments** среды MS10 имеется мультиметр **XMM1** (рис. 1.1а), используемый для измерения тока, напряжения и сопротивления. В схеме (рис. 1.1а) мультиметр, работающий в режиме измерения напряжения, подключается к зажимам резистора **R1** с помощью ключа **S**, управляемого клавишей **S** клавиатуры. В модели мультиметра **XMM1** устанавливают род тока (постоянный «–» или переменный «~»), измеряемую величину по единице измерения: **A** – ток, **V** – напряжение, **Ω** – сопротивление, **dB** – уровень напряжения в децибелах и другие параметры (**Settings**) (см. рис. 1.2 справа).

2. Измерение сопротивлений

Для прямого измерения сопротивления резистивного элемента (резистора в том числе) будем использовать мультиметр **XMM2**, в диалоговом окне которого нужно установить режим работы «–» (постоянный ток), измеряемую величину **Ω**, значение тока, например 10 nA (10 нА) при измерении сопротивления (**Settings**), и подключить прибор к зажимам отдельного резистора (рис. 1.2) или параллельно участку резистивной цепи (без источников энергии). При измерении сопротивления между двумя любыми точками схемы цепи нужно, чтобы хотя бы один из узлов схемы имел соединение с «заземленной» точкой, при этом ветви с идеальными источниками тока должны быть разомкнуты, а идеальные источники напряжения заменены короткозамкнутыми участками (проводниками).

В практике измерения сопротивлений резистивных элементов, кроме прямых и сравнительных методов, широко используется так называемый метод вольтметра-амперметра, в основу которого положен закон Ома для цепей постоянного тока (см. рис. 1.1а и б). Заметим, что этот метод позволяет получить

лишь приближенное значение измеряемого сопротивления **R ≈ U/I**. Так, для схемы, изображенной на рис. 1.1а:

$$R_1 = \frac{U}{I - \frac{U}{R_V}}$$

а для схемы, изображенной на рис. 1.1б:

$$R_2 = \frac{U - R_A I}{I}$$

где R_V и R_A – внутренние сопротивления вольтметра и амперметра соответственно.

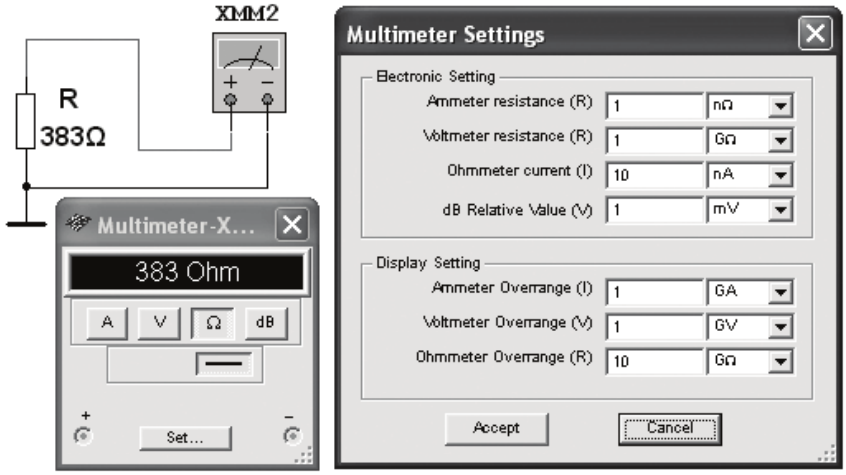


Рисунок 1.2

3. Измерение угла сдвига фаз

Для измерения угла сдвига фаз ϕ между синусоидальным напряжением и током в реальной цепи используют: измерители разности фаз, так называемый метод вольтметра-амперметра-ваттметра, при котором угол ϕ определяют из уравнения $\phi = arccos(P/UI)$, где P – показание ваттметра, а также методы, основанные на измерении временного интервала Δt при помощи электронно-лучевого осциллографа.

Временной интервал $\Delta t = \phi/\omega = \phi/2\pi f$ пропорционален фазовому сдвигу ϕ между синусоидальным напряжением и током в неразветвленной цепи (рис. 1.3а) и обратно пропорционален угловой частоте ω напряжения (тока).

При этом фазовый угол (в электрических градусах) определяют по формуле

$$\phi = 360°\Delta t/T,$$

где T = l/f – период изменения напряжения в секундах (с);

f – частота питающего цепь напряжения в герцах (Гц).

Временной интервал Δt = T2 – T1 обычно измеряют между нулевыми значениями осциллограмм напряжения и тока с помощью визирных линий (визиров), расположенных слева и справа от экрана осциллографа (рис. 1.3а). Угол ϕ берется со знаком «плюс», если ток отстает по фазе от напряжения (см. рис. 1.3а), и со знаком «минус», если ток опережает по фазе напряжение.

а) б)

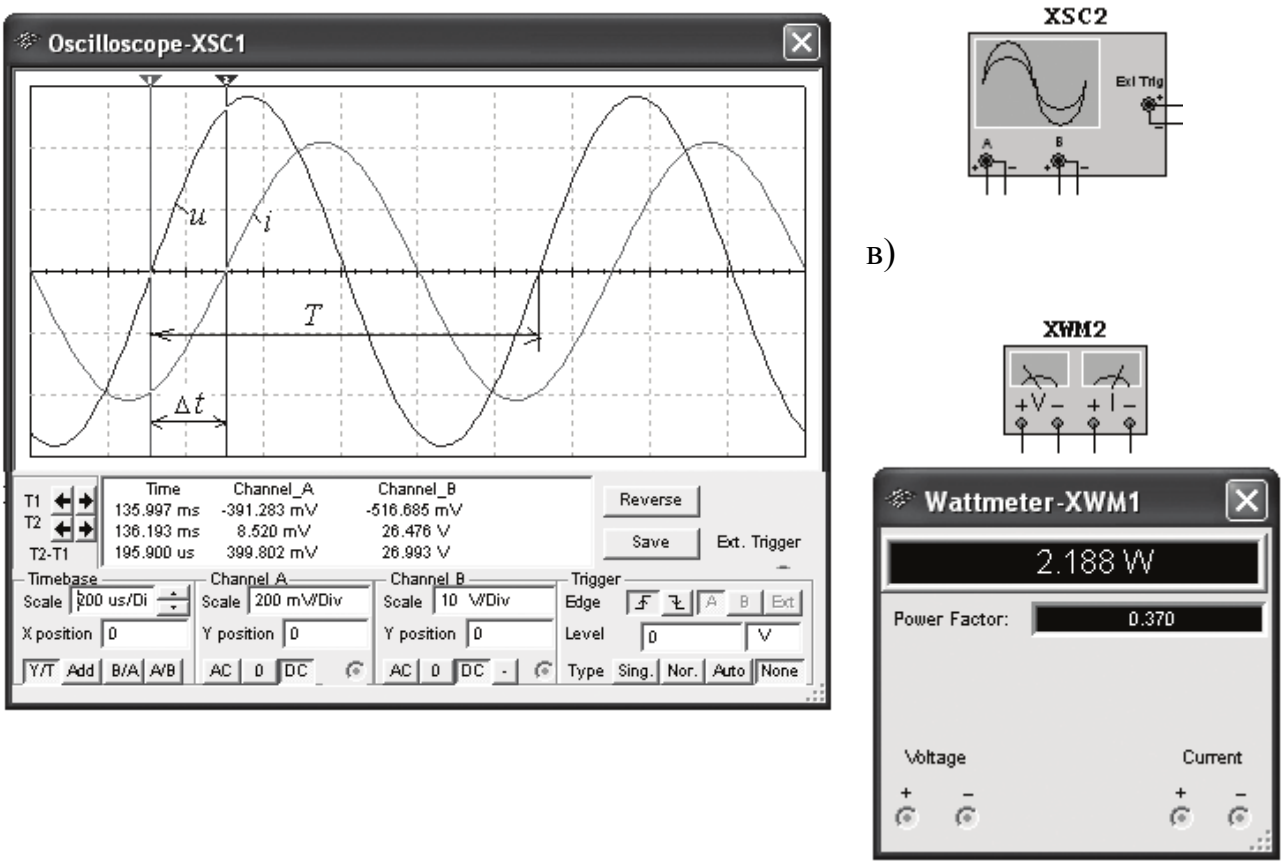


Рисунок 1.3

Установка чувствительности каналов *A* (**Channel A**) и *B* (**Channel B**) и развертки осциллограмм во времени (**Time base**) производится в окне, выводимом ниже поля осциллограмм (см. рис. 1.3а).

При моделировании схем цепей на рабочем поле программы MS10 и их анализе для измерения угла сдвига фаз в цепях переменного тока наряду с осциллографом будем использовать также виртуальный ваттметр **XWM1** (рис. 1.3в), размещенный в библиотеке **Instruments**. Ваттметр непосредственно измеряет активную мощность *P* цепи (ветви) в ваттах и коэффициент мощности cosϕ (**Power Factor**).

Задания и методические указания к их выполнению

Задание 1. Изучить краткую инструкцию (см. приложение 1) работы с программной средой NI Multisim 10 (MS10). С этой целью щелкните мышью на кнопках **Помощь/Первые шаги в MS10** меню и просмотрите демонстрационные кадры с объяснениями, как открыть библиотеку компонентов Basic, «перетащить» мышью компоненты на рабочее поле среды MS10, соединить их проводниками и установить параметры, изменить цвет проводников, как измерить с помощью визирных линий осциллографа временной интервал (сдвиг) Δt меж-

ду двумя синусоидальными величинами – напряжением и током и рассчитать угол сдвига фаз ϕ между ними.

Задание 2. Открыть библиотеку источников энергии **Source** и «перетащить» на рабочее поле среды MS10 идеальный источник **E1** постоянного напряжения, затем из библиотеки базовых компонентов **Basic** «перетащить» четыре резистора **R1**, …, **R4**, из библиотеки индикаторов **Indicator** – амперметр **A** и четыре вольтметра (**V1, …, V4**), из панели приборов **Instruments** – мультиметр **XMM1**.

После двойного щелчка мышью на изображении элемента или прибора в открывающихся диалоговых окнах:

• задать ЭДС источника напряжения $E_1 = N$ (в вольтах), где N – номер записи фамилии студента в учебном журнале группы;

• обозначить (щелкая на кнопках **Label** и **Value**) резисторы и установить значения их сопротивлений: $R_1 = N$; $R_2 = 2N$; $R_3 = 3N$; $R_4 = 4N$;

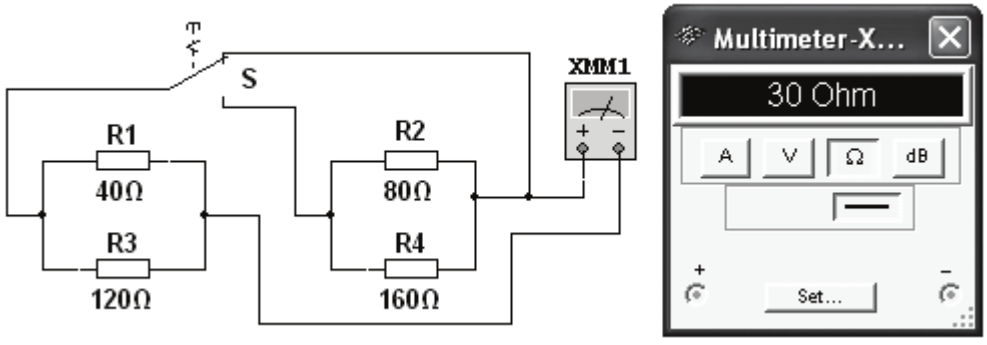
• задать или оставить установленный по умолчанию режим **DC** функционирования измерительных приборов и их внутренние сопротивления: 1 нОм для амперметра и 10 МОм для вольтметров;

• задать измеряемую величину Ω мультиметра и режим его работы (постоянный ток).

Соединить параллельно между собой сопротивления R_1 и R_3 ; R_2 и R_4 и измерить с помощью мультиметра **XMM1** сопротивления разветвлений резисторов (см. рис. 1.4а для варианта 40). Полученные значения сопротивлений занести в табл. 1.1 и сравнить со значениями, вычисленными по формулам

$R_{13} = R_1R_3 /(R_1 + R_3)$ и $R_{24} = R_2 R_4 /(R_2 + R_4)$.

а)



б)

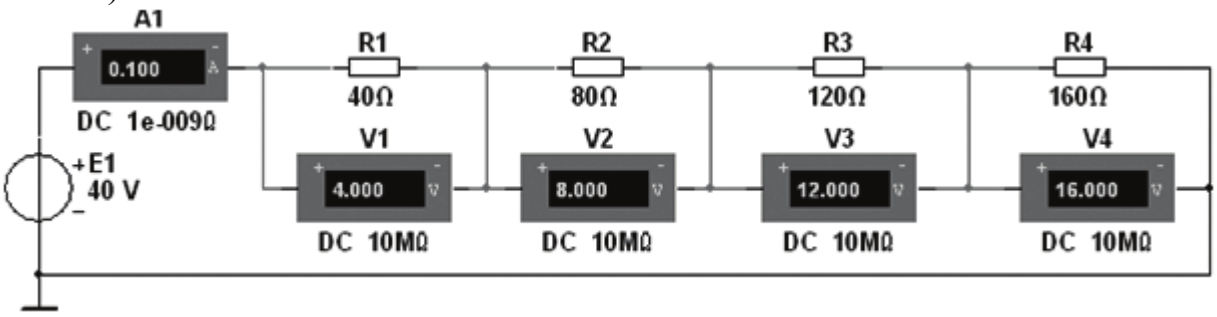


Рисунок 1.4

Таблица 1.1

Измерено	R_{13}, Ом	R_{24}, Ом	U_1, мВ	U_2, мВ	U_3, мВ	U_4, мВ
			$I_1=I$, мА	$I_2=I$, мА	$I_3=I$, мА	$I_4=I$, мА
Вычислено	R_{13}, Ом	R_{24}, Ом	R_1, Ом	R_2, Ом	R_3, Ом	R_4, Ом

Собрать схему (см. рис. 1.4б для варианта 40) и, согласно варианту, установить значения параметров элементов и приборов. Запустить программу MS10 (щелкнуть мышью на кнопке  меню среды MS10) и занести показания приборов (значение тока и значения напряжений на зажимах резисторов) в табл. 1.1.

Рассчитать сопротивления резисторов и занести их значения в табл. 1.1.

Задание 3. Измерить индуктивность катушки и емкость конденсатора косвенным методом по результатам прямых измерений напряжения, тока и мощности *RL*- или *RC*-ветви и косвенного измерения угла сдвига фаз ϕ_k.

С этой целью собрать на рабочем поле программы MS10 схему цепи (см. рис. 1.5), и установить:

• параметры идеального источника синусоидального напряжения $e = E_m \sin(\omega t + \Psi_u) = \sqrt{2}E\sin(2\pi f t + \Psi_u)$: действующее значение ЭДС $E = 5 + N$, В; частоту $f = 1$ *кГц* при измерении индуктивности L катушки и частоту $f = 10$ *кГц* при измерении емкости C конденсатора; начальную фазу напряжения $\Psi_u = 0$;

• режим работы **AC** (переменный ток) амперметра **A** (R_A = 1 нОм) и вольтметра **V** (R_V = 10 МОм);

• значение сопротивления R_1 = 25 Ом резистора **R1** (имитирующего активное сопротивление катушки) и сопротивления R_2 = 10 Ом резистора **R2**;

• значение индуктивности катушки $L = 5 + \text{int}(N/5)$, мГн, и емкости конденсатора $C = 1 + \text{int}(N/10)$, мкФ, где int(a/b) – целая часть операции a/b;

• красный цвет провода, соединенного с каналом **A**, и синий цвет провода, соединенного с каналом **B**, двухканального осциллографа **XSC1**;

• задать параметры осциллографа **XSC1**. При этом на вход канала **A** подано напряжение с источника **INУТ**, пропорциональное входному току *i*, а на вход канала **B** подано напряжение *u* с зажимов источника напряжения *e*. Цветовая окраска осциллограмм (см. рис. 1.3а) соответствует установленным цветам проводов, соединенных с соответствующими входами каналов прибора **XSC1**;

• чувствительность 200 мВ/дел (mV/div) канала **A** осциллографа и 5 или 10 В/дел (V/div) канала **B**; длительность развертки (**TIME BASE**) в режиме *Y/T* – 0,2 мс/дел (2 ms/div). При измерениях указанные цены делений рекомендуется изменять таким образом, чтобы амплитуды напряжений были бы равны не менее 0,5–0,75 высоты экрана осциллографа (в режиме **Espand**), а по оси времени укладывалось два-три периода колебания напряжений;

• управляющую переключателем клавишу **S** клавиатуры;

• значение коэффициента передачи **INUT** k = 1 Ом;

• управляемый контакт переключателя **S** в нижнее положение, то есть подключить R_1L - ветвь к источнику *e*.

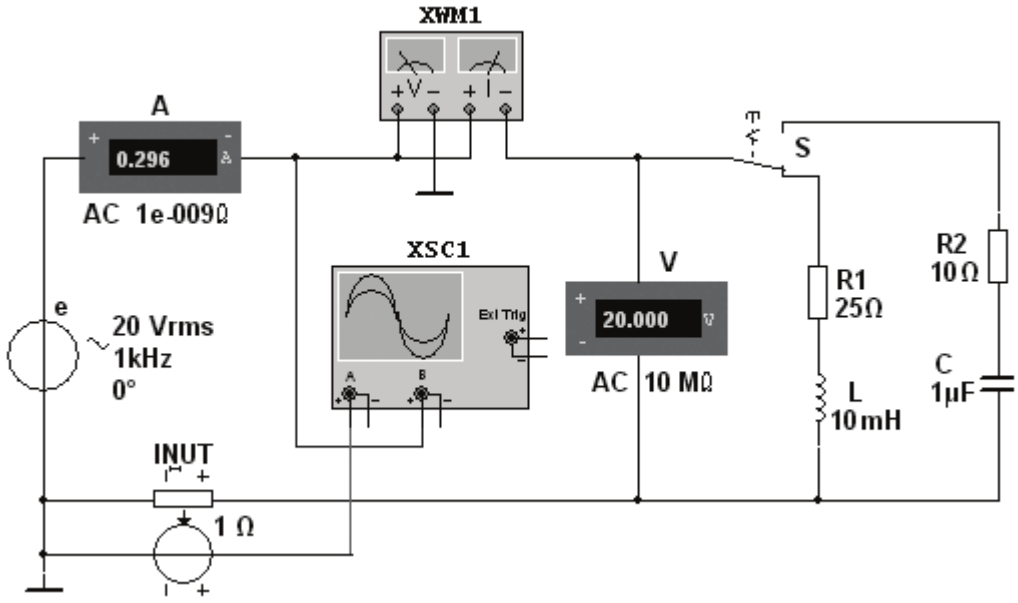


Рисунок 1.5

Запустить программу MS10 (щелкнуть мышью на кнопке  меню среды MS10), снять показания приборов и занести их в табл. 1.2 Методика определения угла сдвига фаз ϕ между напряжением и током описана в п. 3 раздела «Теоретические сведения…». Убедиться (см. рис. 1.3а), что ток i_1 в R_1 *L*-ветви отстает по фазе от напряжения *u* на угол $\phi_1 = arctg(X_L/R_1) = arccos(P_1/UI_1)$, где P_1 – показание ваттметра.

Установить частоту ЭДС f = 10 кГц источника *e(t)* и с помощью переключателя **S** подключить R_2C-*ветвь* к источнику *e*. Показания приборов занести в табл. 1.2. Убедиться (анализируя расположение осциллограмм на экране осциллографа), что ток i_2 в R_2C-ветви опережает по фазе напряжение *u* на угол $\phi_2 = arctg(-X_C/R_2) = -arccos(P_2/UI_2)$, где P_2 – показание ваттметра.

Рассчитать полное $Z = U/I$, активное $R = Z\cos\phi$ и реактивное $X = Z\sin\phi$ сопротивления R_1L- и R_2C-ветвей и занести их в табл. 1.2.

Таблица 1.2

Ветвь	Установлено		Измерено				Вычислено				
	E, В	*f*, кГц	*U*, В	*I*, мА	*P*, Вт	*φ*, град	*Z*, Ом	*R*, Ом	*X*, Ом	*L*, мГн	*C*, мкФ
R_1 *L*											
R_2C											

Так как индуктивное сопротивление катушки $X_L = \omega L = 2\pi fL$, Ом, а емкостное сопротивление конденсатора $X_C = 1/\omega C = 1/2\pi fC$, Ом, то:

• индуктивность катушки, включенной в R_1L-ветвь:

$L = X_L/\omega = X_L/2\pi f$, Гн;

• емкость конденсатора, включенного в R_2C-ветвь:

$C = 1/(\omega X_C) = 1/(2\pi f \cdot X_C)$, Ф.

Вычисленные значения индуктивности L катушки и емкости C конденсатора занести в табл. 1.2. Сравнить полученные значения R, L и C с установленными их значениями в схеме цепи.

Содержание отчёта

1. Наименование и цель работы.

2. Перечень приборов, использованных в экспериментах, с их краткими характеристиками (тип прибора, назначение, род тока, измеряемые величины, пределы измерения или выходные параметры).

3. Электрические схемы измерения сопротивлений резисторов, индуктивности катушки и емкости конденсатора, и копии рисунков осциллограмм напряжения и тока.

4. Таблицы результатов измерений и расчетов.

Практическая работа № 2. Моделирование цепи постоянного тока

Цель занятия:

Смоделировать цепь постоянного тока и опытным путем проверить основные законы для цепи постоянного тока со смешанным соединением приемников электрической энергии.

Задания и методические указания к их выполнению

1. Собрать на рабочем поле экрана электрическую цепь постоянного тока (рис. 2.1) при разомкнутом ключе.

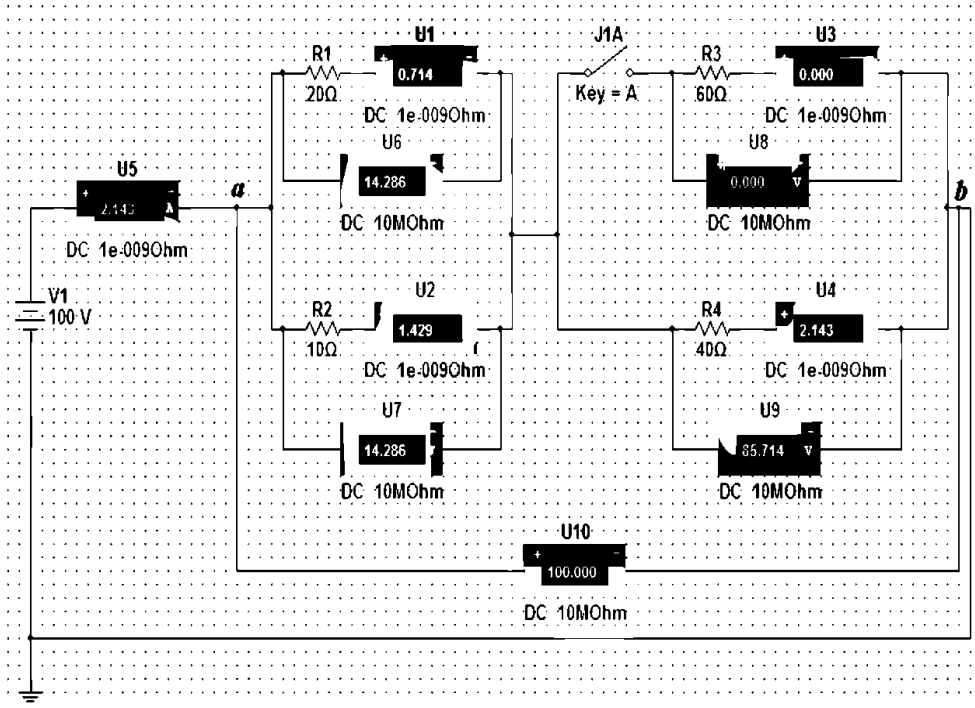


Рисунок 2.1 – Схема цепи

2. Задать параметры элементов цепи согласно варианту (табл. 2.1).

Таблица 2.1

Вариант	1	2	3	4	5	6	7
E, В	100	200	400	100	200	400	200
R_1, Ом	20	10	50	20	40	20	40
R_2, Ом	10	20	50	40	20	40	20
R_3, Ом	60	40	20	60	40	60	40
R_4, Ом	40	60	40	40	60	20	60

3. Записать в таблицу 2.2 показания приборов.

Таблица 2.2

	Положение ключа	Параметры цепи	Показания приборов
Схема с источником ЭДС	Ключ разомкнут	U_{ab}, В	
		I, А	
		I_{R1}, А	
		U_{R1}, В	
		I_{R2}, А	
		U_{R2}, В	

		I_{R3}, А	
		U_{R3}, В	
		I_{R4}, А	
		U_{R4}, В	
	Ключ замкнут	U_{ab}, В	
		I, А	
		I_{R1}, А	
		U_{R1}, В	
		I_{R2}, А	
		U_{R2}, В	
		I_{R3}, А	
		U_{R3}, В	
		I_{R4}, А	
		U_{R4}, В	

Содержание отчёта

1. Наименование и цель работы.
2. Электрическую схему.
3. Таблицу результатов измерений.

Практическая работа № 3. Моделирование цепи переменного тока

Цель занятия:

Смоделировать цепь переменного тока при последовательном и параллельном соединении резистора, катушки индуктивности и конденсатора и опытным путем проверить основные законы для цепи переменного тока.

Задания и методические указания к их выполнению

Задание 1: Моделирование цепи переменного тока при последовательном соединении резистора, катушки индуктивности и конденсатора.

1. Собрать на рабочем поле экрана электрическую цепь синусоидального тока с последовательным соединением резистора, катушки и конденсатора (рис. 3.1). Катушка индуктивности является реальной, поэтому она обладает активным сопротивлением обмотки.

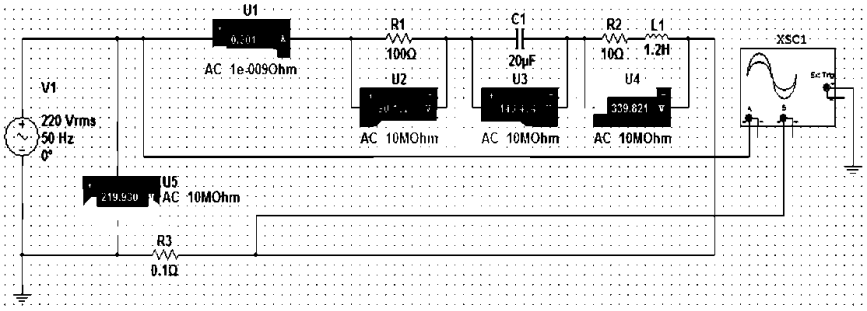


Рис. 3.1 – Схема цепи с последовательным соединением резистора, катушки и конденсатора

Элементы электрической цепи берутся из окон выбора источников и компонентов. Измерительные приборы – с верхней панели индикаторов, осциллограф – из боковой панели инструментов.

Измерительные приборы (амперметр и вольтметр) необходимо переключить в режим измерения переменного тока **AC**.

Для соединения элементов схемы необходимо подвести курсор к подсоединяемому концу одного элемента до появления точки на конце этого элемента, нажать левую кнопку мыши, подвести линию - «провод» к началу следующего элемента до появления точки, снова нажать левую кнопку мыши.

2. Задать параметры элементов цепи согласно варианту (табл. 3.1).

Таблица 3.1

Вариант	1	2	3	4	5	6	7
Напряжение источника питания U, В	220	380	127	220	380	127	220
Частота напряжения источника питания f, кГц	50	50	50	50	50	50	50
Начальная фаза напряжения источника питания	0	0	0	0	0	0	0
Сопротивление резистора R, Ом	100	100	200	300	100	200	50
Индуктивность катушки L, мГн	1200	1500	1400	1500	1700	1000	1400
Активное сопротивление катушки индуктивности R_L, Ом	10	50	20	40	25	50	30
Ёмкость конденсатора C, мкФ	20	30	40	50	35	25	45

Провод, идущий на канал **«A»** осциллографа, необходимо обозначить красным цветом, подведя курсор к проводу, и, нажав, правую кнопку мыши, затем «**Color Segment**» (рис. 3.2), выбрать цвет (красный).

Включить цепь, нажав клавишу  (положение **«I»**) в правом верхнем углу монитора. После появления показаний приборов выключить цепь, нажав ту же клавишу (положение **«O»**).

Для наблюдения осциллограмм напряжения и тока необходимо активизировать осциллограф двойным нажатием левой кнопки мыши.

Можно расширить осциллограф, растянув его по горизонтали и вертикали. В случае необходимости настроить осциллограф: горизонтальная развертка регулируется кнопками «**Time base**» (рис. 3.3), при этом изображение сжимается или расширяется по горизонтали (по оси времени).

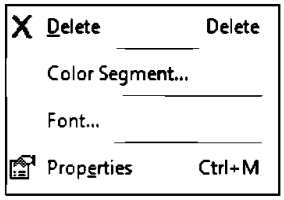


Рисунок 3.2

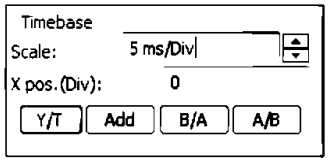


Рисунок 3.3

Масштаб синусоид устанавливается заданием цены деления по вертикальной оси «**V/div**» (рис. 3.4).

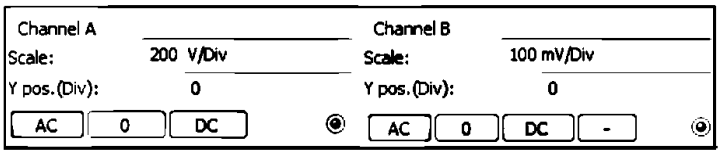


Рисунок 3.4

На канале «А» отображается синусоида напряжения (красная), на канале «В» - синусоида тока (черная).

С помощью осциллографа можно измерить угол сдвига фаз между напряжением и током цепи.

Для измерения угла сдвига фаз φ между напряжением и током необходимо подвести красный курсор к началу синусоиды напряжения (красная синусоида), а синий - к началу синусоиды тока (черная синусоида).

В правом окне осциллографа (рис. 3.5.) значение (T2-T1) необходимо перевести в градусы:

$$\varphi = 360° f(T2-T1),$$

где f – частота напряжения источника питания, Гц.

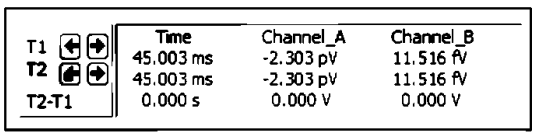


Рисунок 3.5

Обязательно обратить внимание на знак угла φ.

3. Заполнить таблицу согласно снятым показаниям приборов (табл. 3.2).

Таблица 3.2

Элементы цепи	Измеренные величины		Известные величины		
	U, В	*I*, А	*R*, Ом	*L*, Гн	*C*, мкФ
Цепь в целом					
Резистор					
Катушка индуктивности					
Конденсатор					

Задание 2: Моделирование цепи переменного тока при параллельном соединении резистора, катушки индуктивности и конденсатора.

1. Собрать на рабочем поле экрана электрическую цепь синусоидального тока с параллельным соединением резистора, катушки и конденсатора (рис. 3.6).

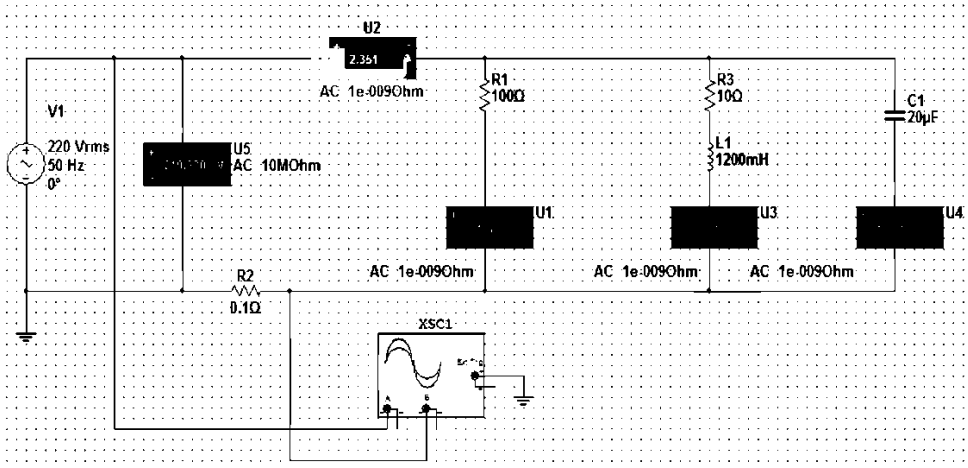


Рис. 3.6. Схема цепи с параллельным соединением резистора, катушки и конденсатора

2. Задать параметры элементов цепи согласно варианту (табл.3.1):
3. Заполнить таблицу согласно снятым показаниям приборов (табл. 3.2).

Содержание отчёта

1. Наименование и цель работы.
2. Электрические схемы.
3. Таблицы результатов измерений.

Практическое занятие № 4. Моделирование логических схем.

Цель занятия:

Приобретение практических навыков моделирования логических схем в программной среде NI Multisim, проверить синтезированные схемы на работоспособность.

Задания и методические указания к их выполнению

1. Изучить экранный интерфейс программы Multisim. Освоить приемы работы с ней. По рис.1 изучить условные обозначения в программе.

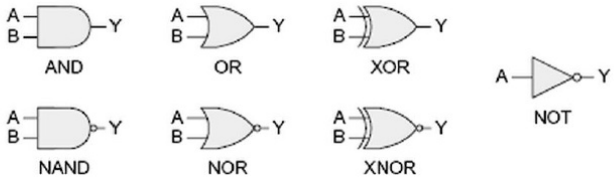


Рисунок 1 – Графические обозначения логических элементов в программе Multisim

2. Собрать схему согласно рис.2 и составить таблицу истинности экспериментальным путем. Записать переключательную функцию в виде СКНФ и СДНФ по таблице истинности .

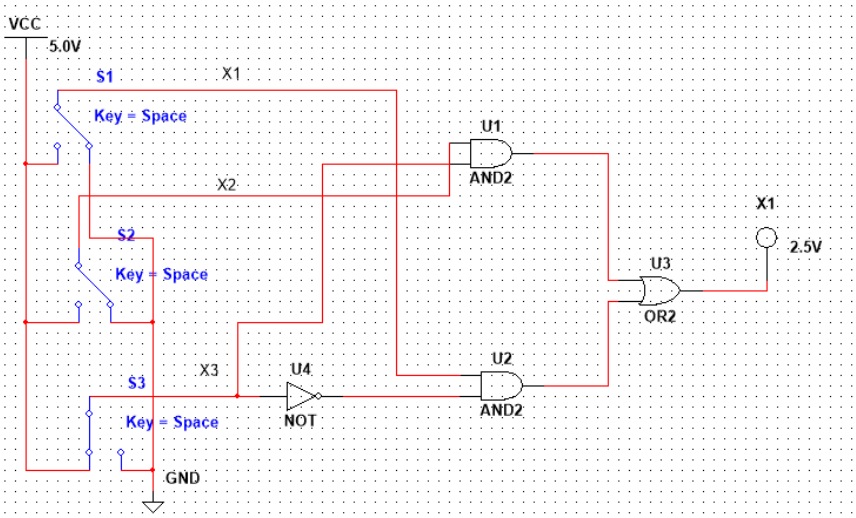


Рисунок 2 - Комбинационная схема

3. Синтезировать устройство и проверить его работоспособность собрав схему в программе Multisim.

	Входные переменные				Номер варианта					
					1	2	3	4	5	6
№ набора	X_4	X_3	X_2	X_1	F	F	F	F	F	F
0	0	0	0	0	1	1	0	1	0	1
1	0	0	0	1	1	1	0	0	0	1
2	0	0	1	0	0	1	1	0	1	1
3	0	0	1	1	0	1	0	0	1	0
4	0	1	0	0	1	1	1	1	0	0
5	0	1	0	1	1	0	0	0	1	0
6	0	1	1	0	0	0	1	0	1	1
7	0	1	1	1	1	0	0	0	1	1
8	1	0	0	0	0	0	0	0	0	1
9	1	0	0	1	0	0	1	0	1	1
10	1	0	1	0	1	0	0	1	0	0
11	1	0	1	1	0	0	1	0	0	0
12	1	1	0	0	0	0	1	1	1	1
13	1	1	0	1	0	0	1	1	0	1
14	1	1	1	0	0	0	0	1	0	0
15	1	1	1	1	0	1	1	0	0	0

Содержание отчёта

1. Наименование и цель работы.
2. Комбинационную схему.

Практическое занятие № 5. Запись математических выражений и вычисление их значений при заданных исходных данных в Mathcad.

Цель занятия:

Приобрести навыки математических вычислений в Mathcad.

Общие сведения

Основное окно приложения имеет ту же структуру, что и большинство приложений Windows. Сверху вниз располагаются заголовок окна, строка меню, панели инструментов (стандартная и форматирования) и рабочий лист, или рабочая область, документа. Новый документ создается автоматически при запуске MathCAD. Файлы документов в MathCAD имеют расширение **.mcd**.

Большинство команд можно выполнить как с помощью меню (верхнего или контекстного), так и панелей инструментов или клавиатуры (рисунок 5.1).

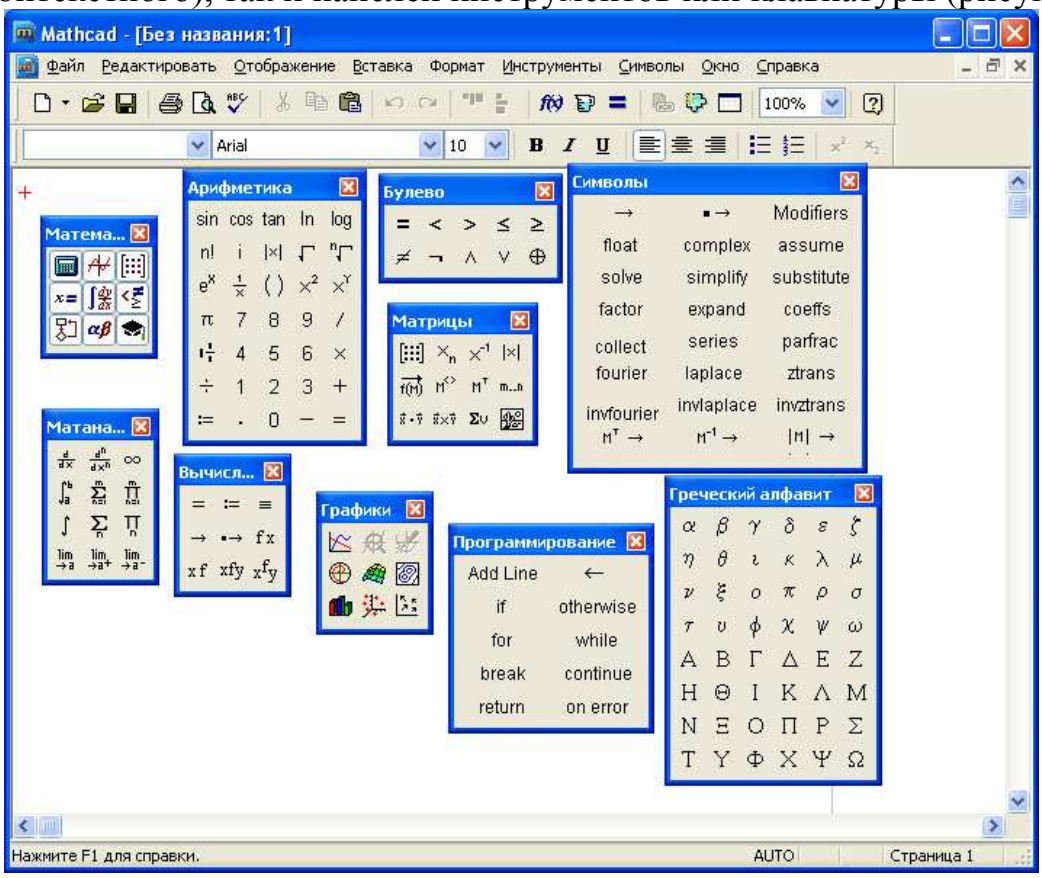
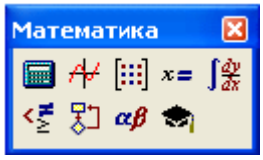


Рисунок 5.1



Панель Math (Математика) предназначена для вызова на экран еще девяти панелей, с помощью которых происходит вставка математических операций в документы. Чтобы вызвать какую-либо из них, нужно нажать соответствующую кнопку на панели Математика.

В окне редактирования формируется документ MathCAD. Новый документ получает имя Untitled (Без названия) и порядковый номер. Одновременно открыто может быть до восьми документов.

Документ состоит из трех видов областей: формульных, текстовых и графических. Расположение нетекстовых блоков в документе имеет принципиальное значение. Области просматриваются системой, интерпретируются и исполняются. Просмотр идет слева направо и сверху вниз.

Для ввода текстового комментария нужно выполнить команду Text Region (Текстовая область) из пункта меню Insert или нажать клавишу с двойной кавычкой (“), или нажать на кнопку текста на панели инструментов. Текстовая область служит для размещения текста между формулами и графиками.

При этом в месте ввода появляется курсор в виде вертикального штриха, на место которого вводятся символы текста. Внутри текста курсор перемещается клавишами перемещения курсора. Переход на новую строку производится нажатием на клавишу Enter. Для окончания ввода нужно щелкнуть мышью вне текстовой области.

Для ввода формулы нужно установить указатель мыши в свободном месте окна редактирования и щелкнуть левой кнопкой мыши. Появится визир в виде красного крестика. Он указывает место, с которого начинается набор формулы.

Константы и переменные

Константами называются поименованные объекты, хранящие некоторые значения, которые не могут быть изменены.

В MathCAD применяются десятичные, восьмеричные и шестнадцатеричные числовые константы. Десятичные константы могут быть целочисленными, вещественными, заданными с фиксированной точкой, и вещественными, заданными в виде мантиссы и порядка.

В MathCAD содержится особый вид констант - размерные. Помимо своего числового значения они характеризуются еще и указанием на то, к какой физической величине они относятся. Для этого указания используется символ умножения. В системе MathCAD заданы следующие основные типы физических величин: time (время), length (длина), mass (масса) и charge (заряд). При необходимости их можно изменить на другие.

Переменные являются поименованными объектами, которым присвоено некоторое значение, которое может изменяться по ходу выполнения программы. Тип переменной определяется ее значением; переменные могут быть числовыми, строковыми, символьными и т. д. Имена констант, переменных и иных объектов называют идентификаторами.

Имя переменной называется идентификатором. MathCAD различает в идентификаторах символы верхнего и нижнего регистров. Например: ABC и AbC имена разных переменных.

Идентификаторы MathCAD должны начинаться с буквы и могут содержать следующие символы:

- латинские буквы любого регистра;
- арабские цифры от 0 до 9;
- символ подчеркивания (_), символ процент (%) и символ (.);
- буквы греческого алфавита (набираются с использованием клавиши Ctrl или применяется палитра греческих букв).

Определение переменных

Переменные должны быть предварительно определены пользователем, т. е. им необходимо хотя бы однажды присвоить значение. В качестве оператора присваивания используется знак :=, тогда как знак = отведен для вывода значения константы или переменной. Попытка использовать неопределенную переменную ведет к выводу сообщения об ошибке.

В MathCAD различают: локальные и глобальные переменные.

Локальные переменные вводятся:

Имя_переменной : выражение

На экране:

Имя_переменной := выражение

Глобальные переменные вводятся:

Имя_переменной ~ выражение

На экране:

Имя_переменной ≡ выражение

Если переменной присваивается начальное значение с помощью оператора :=, такое присваивание называется локальным. До этого присваивания переменная не определена и ее нельзя использовать. MathCAD читает рабочий документ слева направо и сверху вниз, поэтому определив переменную, ее можно использовать в вычислениях везде правее и ниже равенства, в котором она определена. Однако с помощью знака ≡ (три горизонтальные черточки) можно обеспечить глобальное присваивание, т. е. оно может производиться в любом месте документа. К примеру, если переменной присвоено таким образом значение в самом конце документа, то она будет иметь это же значение и в начале документа.

Например:

Ввод с клавиатуры	*Вид на экране*
local:137	*local := 137 локальное определение переменной local;*
global~987.23	*global ≡ 987.23 глобальное определение переменной global.*

Переменные могут использоваться в математических выражениях, быть аргументами функций или операндом операторов.

Переменные могут быть и размерными, т. е. характеризоваться не только своим значением, но и указанием физической величины, значение которой они хранят. Проведение расчетов с размерными величинами и переменными особенно удобно при решении различных физических задач.

Предопределенные переменные

Предопределенные (системные) переменные – особые переменные, которым изначально системой присвоены начальные значения.

Переменная	Ввод	Назначение	Значение по умолчанию
π	Ctrl + Shift + p	Число π	3,14159
e	e	Основание натурального логарифма	2,718
∞	Ctrl + Shift + z	Системная бесконечность	10^{307}

i или *j*	1i или 1j	Мнимая единица	
%		Процент	0,01
TOL		Погрешность численных методов	0,001
ORIGIN		Нижняя граница индексации массивов	0

Операторы

Операторы - элементы языка, с помощью которых можно создавать математические выражения. К ним, например, относятся символы арифметических и логических операций, знаки вычисления сумм, произведений, производной и интеграла и т. д.

Операторы, обозначающие основные арифметические действия, вводятся с панели **Calculator** (Калькулятор, Арифметика).

Вычислительные операторы вставляются в документы при помощи панели инструментов **Calculus** (Матанализ). При нажатии любой из кнопок в документе появляется символ соответствующего математического действия, снабженный несколькими местозаполнителями. Количество и расположение местозаполнителей определяется типом оператора и в точности соответствует их общепринятой математической записи.

Результатом действия логических, или булевых, операторов являются только числа 1 (если логическое выражение, записанное с их помощью, истинно) или 0 (если логическое выражение ложно).

Вычислительные операторы сгруппированы на панели **Evaluation** (Вычисления):

- Численный вывод (Evaluate Numerically) =
- Символьный (аналитический) вывод (Evaluate Symbolically) →
- Присваивание (Definition) :=
- Глобальное присваивание (Global Definition) ≡.

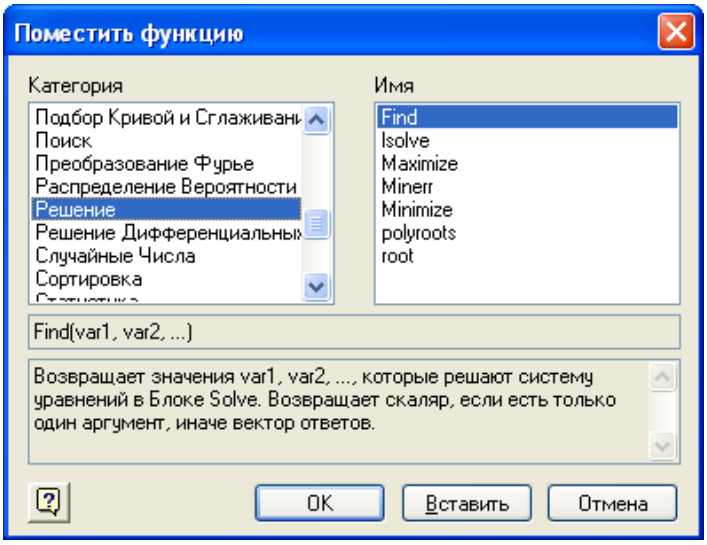
Оператор	Клавиша	Назначение оператора
$X := Y$	*X : Y*	Локальное присваивание *X* значения *Y*
$X \equiv Y$	*X ~ Y*	Глобальное присваивание *X* значения *Y*
$X =$	*X =*	Вывод значения *X*
$X + Y$	*X + Y*	Сложение *X* с *Y*
$X - Y$	*X - Y*	Вычитание из *X* значения *Y*
$X \cdot Y$	*X * Y*	Умножение *X* на *Y*
$\frac{X}{Z}$	*X / Z*	Деление *X* на *Z*
$X \div Y$	*Ctrl + /*	Линейное деление
$a\frac{b}{c}$	*Ctrl + Shift + +*	Дробь (смешанный номер)
z^w	*z ^ w*	Возведение *z* в степень *w*
$\sqrt{z}$	*z *	Вычисление квадратного корня из *z*
$n!$	*n !*	Вычисление факториала
B_n	*B [n*	Ввод нижнего индекса *n*
$A_{n,m}$	*A [n , m*	Ввод двойного нижнего индекса
$A^{<n>}$	*A Ctrl + 6 n*	Ввод верхнего индекса (для векторов)

Определение функций

Функция – выражение, согласно которому проводятся некоторые вычисления с его аргументами и определяется его числовое значение.

Функции в пакете MathCAD могут быть встроенные и определенные пользователем.

В MathCAD имеется множество встроенных функций. Для их ввода используется команда меню Вставка → Функция или кнопка  на панели инструментов. В диалоговом окне нужно выбрать Категорию и соответствующую функцию.



Функция пользователя вначале должна быть определена, а затем к ней может быть произведено обращение. Функция пользователя определяется следующим образом:

Имя_функции(Переменная1, Переменная2, …) := Выражение

Задается имя функции, в скобках указывается список аргументов функции - это перечень используемых в выражении переменных, разделяемых запятыми.

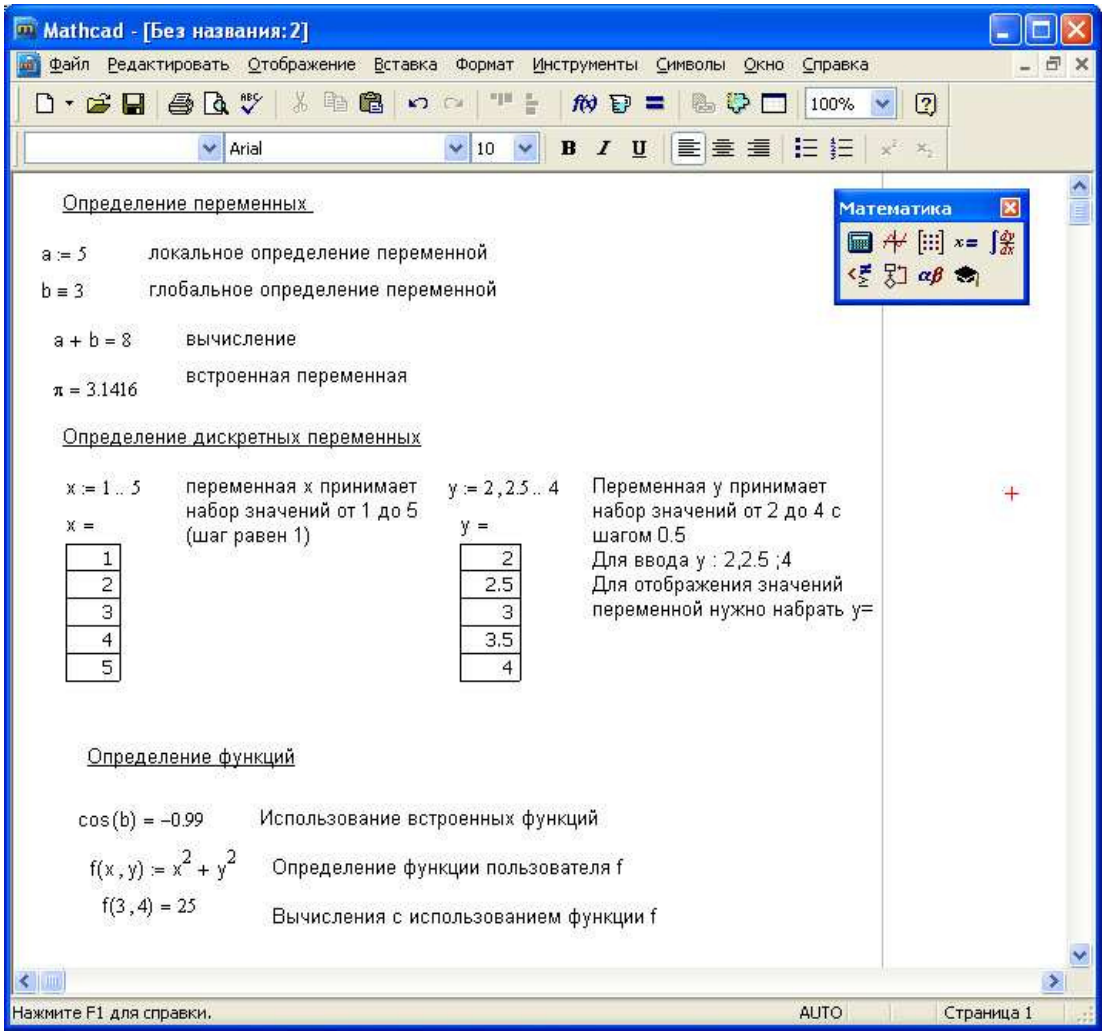
Затем записывается знак присваивания, справа от которого записывается выражение. Выражение - это любое арифметическое выражение, содержащее доступные системе операторы и функции с операндами и аргументами, указанными в списке аргументов.

Примеры задания функций одной и двух переменных:

f(x):= 10 - exp(x)
mult(x, y) := x*y

Обращение к функции осуществляется по ее имени с подстановкой на место аргументов констант, переменных, определенных до обращения к функции, и выражений. Например:

f(3), sin(1), mult(2,3).

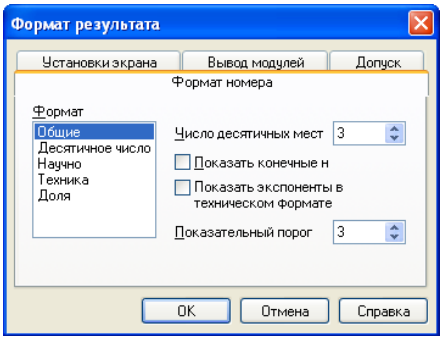


Форматирование результатов

Способ, которым MathCAD выводит числа, называется форматом результата. Формат результата может быть установлен для всего документа (глобальный формат) или для отдельного результата (локальный формат).

Глобальный формат устанавливается командой меню Формат→Результат. В диалоговом окне, появляющемся после выбора этой команды, устанавливается выводимая точность числа, диапазон показателя степени (если вывод чисел нужен в форме с плавающей запятой) и точность нуля. После внесения требуемых изменений нужно нажать кнопку ОК.

Для установки формата отдельного числа нужно: щелкнуть мышью на выражении, результат которого нужно переформатировать; вызвать команду форматирования и проделать вышеописанные действия.



Задания

Задание № 1

Вычислить значение арифметического выражения:

Вариант	Выражение	Вариант	Выражение	Вариант	Выражение
1	$1\frac{1}{4}+\frac{1}{9}$	2	$1\frac{1}{7}+2\frac{1}{5}$	3	$3\frac{3}{4}-\frac{4}{5}$
4	$\frac{5}{7}\div\frac{4}{21}$	5	$\frac{1}{3}\div\frac{5}{12}$	6	$\frac{5}{6}\cdot 2{,}4$
7	$\frac{4}{5}-2{,}5$	8	$3\frac{1}{11}+\frac{1}{3}$	9	$\frac{1}{5}+2\frac{1}{9}$
10	$5\frac{2}{3}\cdot\frac{9}{17}$	11	$8\frac{1}{2}\cdot\frac{7}{14}$	12	$\frac{1}{5}+4\frac{1}{2}$
13	$\frac{1}{35}\div\frac{4}{5}$	14	$\frac{5}{9}\cdot 1{,}8$	15	$\frac{2}{9}\cdot 1{,}8$

Задание № 2

Вычислить значение арифметического выражения:

Вариант	Выражение	Вариант	Выражение
1	$\frac{\left(13{,}75+9\frac{1}{6}\right)\cdot 1{,}2}{\left(10{,}3-8\frac{1}{2}\right)\cdot\frac{5}{9}}+\frac{\left(6{,}8-3\frac{3}{5}\right)\cdot 5\frac{5}{6}}{\left(3\frac{2}{3}-3\frac{1}{6}\right)\cdot 56}-27\frac{1}{6}$	2	$\frac{\left(\frac{1}{6}+0{,}1+\frac{1}{15}\right)\div\left(\frac{1}{6}+0{,}1-\frac{1}{15}\right)\cdot 2{,}52}{\left(0{,}5-\frac{1}{3}+0{,}25-\frac{1}{5}\right)\div\left(0{,}25-\frac{1}{6}\right)\cdot\frac{7}{13}}$
3	$\left(\frac{3\frac{1}{3}+2{,}5}{2{,}5-\frac{1}{3}}\cdot\frac{4{,}6-2\frac{1}{3}}{4{,}6+2\frac{1}{3}}\cdot 5{,}2\right)\div\left(\frac{0{,}05}{\frac{1}{7}-0{,}125}+5{,}7\right)$	4	$\frac{0{,}4+8\cdot\left(5-0{,}8\cdot\frac{5}{8}\right)-5\div 2\frac{1}{2}}{\left(1\frac{7}{8}\cdot 8-\left(8{,}9-2{,}6\div\frac{2}{3}\right)\right)\cdot 34\frac{2}{5}}\cdot 90$
5	$\frac{\left(\frac{3}{5}+0{,}425-0{,}005\right)\div 0{,}1}{30{,}5+\frac{1}{6}+3\frac{1}{3}}+\frac{6\frac{3}{4}+5\frac{1}{2}}{26\div 3\frac{5}{7}}-0{,}05$	6	$\frac{3\frac{1}{3}\cdot 1{,}9+19{,}5\div 4\frac{1}{2}}{\frac{62}{75}-0{,}16}\div\frac{3{,}5+4\frac{2}{3}+2\frac{2}{15}}{\left(1\frac{1}{20}+4{,}1\right)\cdot 0{,}5}$
7	$\frac{\left(1\frac{1}{5}\div\left(\frac{17}{40}+0{,}6-0{,}005\right)\right)\cdot 1{,}7}{\frac{5}{6}+1\frac{1}{3}-1\frac{23}{30}}+\frac{4{,}75+7\frac{1}{2}}{33\div 4\frac{5}{7}}\div 0{,}25$	8	$\frac{\left(4{,}5\cdot 1\frac{2}{3}-6{,}75\right)\cdot\frac{2}{3}}{\left(3\frac{1}{3}\cdot 0{,}3+5\frac{1}{3}\cdot\frac{1}{8}\right)\div 2\frac{2}{3}}+\frac{1\frac{4}{11}\cdot 0{,}22\div 0{,}3-0{,}96}{\left(0{,}2-\frac{3}{40}\right)\cdot 1{,}6}$
9	$\frac{\left(1{,}88+2\frac{3}{25}\right)\cdot\frac{3}{16}}{0{,}625-\frac{13}{18}\div\frac{26}{9}}+\frac{\left(\frac{0{,}216}{0{,}15}+0{,}56\right)\div 0{,}5}{\left(7{,}7\div 24\frac{3}{4}+\frac{2}{15}\right)\cdot 4{,}5}$	10	$\frac{0{,}128\div 3{,}2+0{,}86}{\frac{5}{6}\cdot 1{,}2+0{,}8}\cdot\frac{\left(1\frac{32}{63}-\frac{13}{21}\right)\cdot 3{,}6}{0{,}505\cdot\frac{2}{5}-0{,}002}$
11	$\frac{0{,}125\div 0{,}25+1\frac{9}{16}\div 2{,}5}{(10-22\div 2{,}3)\cdot 0{,}46+1{,}6}+\left(\frac{17}{20}+1{,}9\right)\cdot 0{,}5$	12	$\frac{(3{,}4-1{,}275)\cdot\frac{16}{17}}{\frac{5}{18}\cdot\left(1\frac{7}{85}+6\frac{2}{17}\right)}+0{,}5\cdot\left(2+\frac{12{,}5}{5{,}75+\frac{1}{2}}\right)$

13	$\dfrac{\left(0{,}3275-\left(2\dfrac{15}{88}+\dfrac{4}{33}\right)\div 12\dfrac{2}{9}\right)\div 0{,}07}{\left(13-0{,}416\right)\div 6{,}05+1{,}92}$	14	$\dfrac{3{,}75\cdot 1\dfrac{1}{2}+\left(1{,}5\div 3\dfrac{3}{4}\right)\cdot 2\dfrac{1}{2}+\left(1\dfrac{1}{7}-\dfrac{23}{49}\right)\div\dfrac{22}{147}}{2\div 3\dfrac{1}{5}+\left(3\dfrac{1}{4}\div 13\right)\div\dfrac{2}{3}-\left(2\dfrac{5}{18}-\dfrac{17}{36}\right)\cdot\dfrac{18}{65}}$
15	$\dfrac{\left(\left(3\dfrac{7}{12}-2\dfrac{11}{18}+2\dfrac{1}{24}\right)\cdot 1\dfrac{5}{31}-\dfrac{3}{52}\cdot\left(3\dfrac{1}{2}+\dfrac{5}{6}\right)\right)\cdot 1\dfrac{7}{13}}{\dfrac{19}{84}\div\left(5\dfrac{13}{42}-2\dfrac{13}{28}+\dfrac{5}{24}\right)+1\dfrac{2}{27}-\dfrac{1}{3}\cdot\dfrac{4}{9}}$		

Задание № 3

Вычислить значение арифметического выражения. Результат выведите с 6 знаками после запятой.

Вариант	Значения переменных	Выражение
1	$x = 3{,}981$ $y = 1{,}625$ $c = 0{,}512$	$h=\dfrac{\sqrt{c+x^2\cdot\left(\cos^5 x-c\right)}+\sqrt[5]{\sin x+\ln y}}{c+y}$
2	$x = -6{,}251$ $a = 0{,}827$ $z = 25{,}001$	$b=\dfrac{x^3+z}{\cos^2 x+1}+tgx^2-\sqrt{\sin x-a}+\dfrac{e^x}{3x^2}$
3	$x = 3{,}251$ $y = 3{,}325$ $z = 0{,}466$	$h=\dfrac{\sin z+\cos 2x}{2x^5+tgx}+\sqrt[3]{3x+2y}$
4	$x = 0{,}622$ $y = 3{,}325$ $z = 5{,}541$	$\varphi=\dfrac{\left(\cos x-\sin y\right)^3}{\sqrt{tgz}}+\ln^2\left(x\cdot y\cdot z\right)$
5	$x = 17{,}421$ $b = 10{,}365$ $z = 0{,}828$	$k=\dfrac{1+\sin^3 x}{z^2}+\cos^2 x+\dfrac{\ln^2 x+b}{x^4}$
6	$x = 2.444$ $y = 0{,}869$ $z = -0{,}166$	$g=\left\lvert x^2-\dfrac{1}{e^a+3}\right\rvert-\dfrac{1+\sin^3 x}{a^2}$
7	$x = 0{,}335$ $y = 0{,}025$ $z = 32{,}005$	$t=y^{x+1}+\sqrt{\lvert x\rvert+e^y}-\dfrac{z^{3x}-\sin^2 y}{y+\dfrac{z^2}{e^x}}$
8	$x = 3{,}258$ $r = 4{,}005$ $z = -0{,}666$	$p=\dfrac{e^x-2}{z+3}+\sqrt{\sin^2 x^5}-\dfrac{r^3+1}{\cos^2\left(r-2\right)+1}$
9	$x = 0{,}100$ $y = -8{,}750$ $z = 0{,}765$	$t=\left(\left(1+y\right)\cdot\sqrt{\sin^2 z}-\dfrac{\lvert y-x\rvert}{5}\right)^3$

10	$x = 1{,}542$ $a = 3{,}261$ $z = 8{,}005$	$r = \frac{x^2}{e^a} + \frac{1}{3} \cdot \sin^2 z - \ln\sqrt{2x}$
11	$k = 1{,}426$ $a = -1{,}220$ $p = 3{,}500$	$w = p^{0{,}8} + \frac{a}{a-p} - \sin^2 \frac{k^5}{k^5 - 1}$
12	$x = -4{,}500$ $y = 0{,}750$ $z = 0{,}845$	$k = \sqrt{\left\|\frac{-3 \cdot tgy \cdot \ln\left(x^4 + z\right)}{e^{-x} + 1}\right\|}$
13	$a = 3{,}741$ $x = 0{,}825$ $z = 5{,}160$	$v = tg\frac{\sqrt[3]{a}}{5 + a^3} + \frac{\sin z - tg2x}{e^x}$
14	$x = 0{,}400$ $a = 2{,}875$ $f = -0{,}475$	$d = \frac{\sin^2 x + 1}{x^4} + \cos^3 x + e^{a-1} - \frac{\sqrt{\sin f^2 + f}}{\cos^2 a}$
15	$t = 0{,}750$ $a = 0{,}845$ $m = 2{,}5$	$f = \frac{e^{ma+t}}{\sqrt[5]{\frac{ma}{t} + ma^2}} + \frac{am - e^t}{\sqrt{2} + a^2 - \left\|m^3 - \ln t\right\|}$

Задание № 4

Определить функцию *f(x)*, вычислить ее значение при *x = 2,9* и построить таблицу значений функции для *x* [2; 12] с шагом 1.

Вариант	Функция	Вариант	Функция	Вариант	Функция
1	$\frac{8(x-1)}{(x+2)^2}$	2	$(2x+4)\cdot e^{2x+4}$	3	$\frac{5x}{x^2+3}$
4	$2 - \frac{3x}{x^2+3}$	5	$\frac{x^3 - 27x + 54}{x^3}$	6	$-(x+4)\cdot e^{-x-3}$
7	$-\frac{5x}{x^2+3}$	8	$\left(2 + \frac{1}{x}\right)^2$	9	$\left(\frac{x+2}{x-1}\right)^2$
10	$\frac{2x^2+1}{x^2+3}$	11	$(x+1)\cdot e^{x+2}$	12	$(2x-1)\cdot e^{2x-2}$
13	$\frac{x^2 - 6x + 9}{(x-1)^2}$	14	$\frac{5x^2}{x^2+3}$	15	$\frac{3x-2}{(x+1)^3}$

Содержание отчёта

1. Наименование и цель работы.
2. Результаты вычислений.

Практическое занятие № 6. Построение графиков.

Цель занятия:

Приобрести навыки построения графиков в Mathcad.

Общие сведения

В MathCAD встроено несколько различных типов графиков, которые можно разбить на две большие группы.

Двумерные графики:

• X-Y (декартовый) график (**X-Y Plot**);

• полярный график (**Polar Plot**).

Трехмерные графики:

• график трехмерной поверхности (**Surface Plot**);

• график линий уровня (**Contour Plot**);

• трехмерная гистограмма (**3D Bar Plot**);

• трехмерное множество точек (**3D Scatter Plot**);

• векторное поле (**Vector Field Plot**).

Деление графиков на типы несколько условно, т. к., управляя установками многочисленных параметров, можно создавать комбинации типов графиков, а также новые типы (например, двумерная гистограмма распределения является разновидностью простого X-Y графика).

Для построения графиков используются шаблоны. Их перечень содержится в команде меню Вставка→Графики. Большинство параметров графического процессора, необходимых для построения графиков, по умолчанию задается автоматически. Поэтому для начального построения того или иного вида достаточно задать тип графика. В подменю Graph содержится список из семи основных типов графиков.

X-Y Plot	График в декартовой системе координат
Polar Plot	График в полярных координатах
Surfase Plot	Трехмерный график
Contour Plot	Контурный график трехмерной поверхности
3D Scatter Plot	График в виде точек (фигур) в трехмерном пространстве
3D Bar Chart	График для изображения в виде совокупности столбиков в трехмерном пространстве (гистограмма)
Vector Field Plot	График векторного поля на плоскости

MathCAD представляет пользователю разнообразные средства форматирования графика - изменение толщины и цвета линий, вида осей координат, координатные сетки, текстовые комментарии и др. Для того чтобы изменить вид изображения, нужно щелкнуть дважды по полю графика и установить требуемые параметры в окнах настройки.

Графики любого вида, как любые объекты документа, можно выделять, заносить в буфер обмена, вызывать их оттуда и переносить в любое новое место документа. Их можно и просто перетаскивать с места на место курсором мыши, а также растягивать по горизонтали, по вертикали и по диагонали, цепляясь за специальные маркеры выделенных графиков курсором мыши.

Порядок действий при построении всех графиков одинаков. После выбора шаблона построения графика в рабочем документе открывается поле построения графика с помеченными для ввода позициями, которые нужно заполнить для определения графика.

Когда график определен (заполнены все помеченные позиции), то для построения графика при автоматическом режиме вычислений достаточно щелкнуть мышью вне поля графика.

Заполнение шаблона для разных типов графиков имеет свои особенности.

Можно начертить несколько кривых на одном и том же чертеже. Чтобы представить графически несколько выражений по оси ординат относительно одного выражения по оси абсцисс, введите первое выражение по оси ординат, сопровождаемое запятой. Непосредственно под первым выражением появится пустое поле. Введите туда второе выражение, сопровождаемое другой запятой, чтобы получить пустое поле, и т. д.

Чтобы построить несколько независимых кривых на одном чертеже, введите два или более выражения, отделяемых запятыми по оси абсцисс, и то же самое выражение по оси ординат. MathCAD согласует выражения попарно - первое выражение по оси абсцисс с первым выражением по оси ординат, второе со вторым и т. д. Затем рисуется график каждой пары.

Можно построить до 16 функций по оси ординат в зависимости от одного аргумента по оси абсцисс. Однако если для каждой кривой используется свой аргумент, то можно отобразить только до 10 графиков.

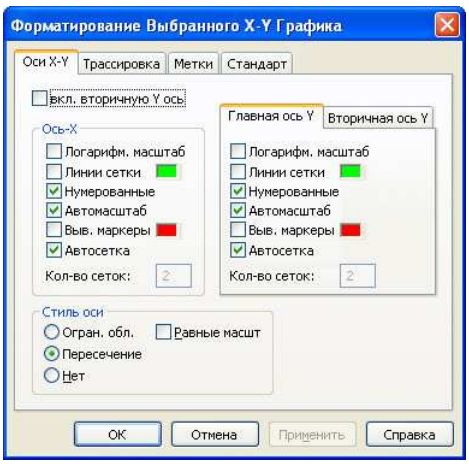
Точно так же можно построить несколько графиков на одном и том же чертеже в полярных координатах, используя эту же технологию заполнения шаблона графика.

Форматирование графиков

Чтобы изменить формат графика, необходимо дважды щелкнуть мышью в области графика.

Если строим график в декартовой системе координат, то появится следующее диалоговое окно для форматирования графика (разные типы графиков имеют разный вид диалоговых окон, но аналогичную технологию форматирования).

Форматирование оси графика можно также произвести, выполнив на ней двойной щелчок.



В MathCAD можно делать следующие надписи на чертеже:

• заголовок выше или ниже графика;

• названия осей, чтобы описать, что отложено на каждой оси;

• имена кривых, идентифицирующих отдельные графики;

• переменные - выражения, определяющие координаты.

Можно использовать эти надписи все вместе или в любой комбинации.

Для того чтобы добавить заголовок к графику в диалоговом окне для форматирования графика, следует щелкнуть по закладке **Labels** (Метки) и напечатать заголовок графика в поле **Title** (Название). Пометить место размещения заголовка: кнопка **Above** (Вверху) или **Below** (Внизу) и удостовериться, что флажок **Show Title** (Выводить) отмечен.

Чтобы надписать одну или обе оси графика, необходимо указать название осей в поле Метки осей.

Можно построить до 16 разных графиков. Каждому графику соответствует строка в прокручивающемся списке, который откроется, если в диалоговом окне для форматирования графика щелкнуть по вкладке **Traces** (Трассировка). На этой вкладке можно изменить параметры: тип, цвет, толщину линии.

По мере появления новых графиков MathCAD ставит в соответствие каждому одну из этих строк.

Задания

Задание

Построить графики функций:

Вариант	Функция одной переменной	Кривая, заданная параметрически	Функция двух переменных
1	$y=\frac{3}{x^3}+\frac{2}{x^2}+\frac{1}{x}$	$x=t^3-3\pi$ $y=t^3-6\cdot arctg(t)$	$z=\sin\left(\frac{x}{y}\right)\cos\left(\frac{y}{x}\right)$
2	$y=\sqrt{x}-\sqrt[3]{x^2}+\sqrt[4]{x^3}$	$x=4\cos^2 t$ $y=4\sin^2 t$	$z=\frac{1}{arctg\frac{y}{x}}$
3	$y=\ln(3x)+\frac{e^{-3x}}{\sqrt{x}}$	$x=sh(t)-t$ $y=ch(t)-1$	$z=x^3y-xy^3$
4	$y=\frac{x^2-\sqrt{x}}{1-x}$	$x=t$ $y=t+2\cdot arctg(t)$	$z=e^{-\frac{x}{y}}$
5	$y=\frac{x^2}{x^3+1}$	$x=2\cdot(3\cos t+\cos(3t))$ $y=2\cdot(3\sin t+\sin(3t))$	$z=4{,}25x\cdot e^{-t}+6t$
6	$y=\sin x-4\cos x$	$x=t^3+3t+1$ $y=t^3-3t+1$	$z=\frac{x^3+y^3}{x^2+y^2}$
7	$y=x^2 tgx$	$x=\frac{3t}{1+t^3}$ $y=\frac{3t^2}{1+t^3}$	$z=\ln\left(x+\sqrt{x^2+y^2}\right)$

8	$y = \frac{\sqrt[3]{x}}{\cos x}$	$x = t \cdot e^{t}$ $y = t \cdot e^{-t}$	$z = \ln\left(tg\frac{x}{y} \right)$
9	$y = \frac{\cos x - \sin x}{\cos x + \sin x}$	$x = 3t + 1$ $y = t^3 + 2t$	$z = \ln\left(x^2 + y^2\right)$
10	$y = \left(1 + x^2\right)\arccos x$	$x = t + e^{-t}$ $y = 2t + e^{-2t}$	$z = x^{xy}$
11	$y = \sqrt{x^3}\, arctg(x)$	$x = \frac{t}{t+3}$ $y = \frac{2t^2 + 3t}{(t+3)^2}$	$z = (1 + \lg x)^{y}$
12	$y = \sin x \cdot \arcsin x$	$x = \ln\left(t + \sqrt{t}\right)$ $y = t \cdot \sqrt{t^2 + 1}$	$z = \frac{x + y}{x - y}$
13	$y = \frac{x^2 - 1}{\lg x}$	$x = 6\cos t - 3\cos(2t)$ $y = 6\sin t - 3\sin(2t)$	$z = \frac{1}{2}\ln\left(x^2 + y^2\right)$
14	$y = x \cdot \cos x \cdot \ln x$	$x = 2\cos t$ $y = 3\sin t$	$z = \frac{x^2 + y^2}{x^2 - y^2}$
15	$y = \ln\left(\sqrt{e^{x}}\right)$	$x = t^3 - 3\pi$ $y = t^3 - 6 \cdot arctg(t)$	$z = x^2 y^4 - x^3 y^3 + x^4 y^2$

Содержание отчёта

1. Наименование и цель работы.
2. Графики функций.

Практическое занятие № 7. Расчёт цепей постоянного тока. Сравнение результатов расчетов в Mathcad с результатами моделирования в NI Multisim.

Цель занятия:

Приобрести навыки расчёта цепей постоянного тока в Mathcad. Сравнить результаты полученные в практической работе №2 с результатами расчётов в Mathcad.

Задания и методические указания к их выполнению

1. Рассчитать значения токов в ветвях схемы и напряжения на участках цепи (Практическая работа №2, рисунок 2.1) при разомкнутом ключе. Данные для расчёта взять из практической работы №2 (таблица 2.1).

Расчет цепи выполнить, используя закон Ома:

1.1. Определить эквивалентное сопротивление цепи:

$$R_{экв} = \frac{R_1 R_2}{R_1 + R_2} + R_4 \tag{7.1}$$

1.2. Определить ток в неразветвленной части цепи I:

$$I = \frac{E}{R_{экв}} \tag{7.2}$$

1.3. Определить напряжения U_{R1} и U_{R2} . Напряжения U_{R1} и U_{R2} равны, так как элементы R_1 и R_2 включены параллельно.

$$U_{R1} = U_{R2} = I\frac{R_1 R_2}{R_1 + R_2} \tag{7.3}$$

1.4. Определить напряжения U_{R3} и U_{R4}:

$U_{R3} = 0$, так как в ветви с сопротивлением R_3 обрыв.

$$U_{R4} = IR_4 \tag{7.4}$$

1.5. Определить токи в ветвях схемы:

$$I_{R1} = \frac{U_{R1}}{R_1} \tag{7.5}$$

$$I_{R2} = \frac{U_{R2}}{R_2} \tag{7.6}$$

$I_{R3} = 0$, так как в ветви с сопротивлением R_3 обрыв;

$$I_{R4} = \frac{U_{R4}}{R_4} \tag{7.7}$$

2. Рассчитать значения токов в ветвях схемы и напряжения на участках цепи при замкнутом ключе.

Расчет цепи выполнить, используя закон Ома:

2.1. Определить эквивалентное сопротивление цепи:

$$R_{экв} = \frac{R_1 R_2}{R_1 + R_2} + \frac{R_3 R_4}{R_3 + R_4} \tag{7.8}$$

2.2. Определить ток в неразветвленной части цепи I по формуле (7.2).

2.3. Определить напряжения U_{R1} и U_{R2} по формуле (7.3).

2.4. Определить напряжения U_{R3} и U_{R4} . Напряжения U_{R3} и U_{R4} равны, так как элементы R_3 и R_4 включены параллельно.

$$U_{R3} = U_{R4} = I\frac{R_3 R_4}{R_3 + R_4} \tag{7.9}$$

2.5. Определить токи в ветвях схемы по формулам (7.5) – (7.7)

$$I_{R3} = \frac{U_{R3}}{R_3} \tag{7.10}$$

3. Полученные результаты записать в таблицу 7.1

Таблица 7.1

	Положение ключа	Параметры цепи	Результаты расчёта
Схема с источником ЭДС	Ключ разомкнут	U_{ab}, В	
		I, А	
		I_{R1}, А	
		U_{R1}, В	
		I_{R2}, А	
		U_{R2}, В	
		I_{R3}, А	
		U_{R3}, В	
		I_{R4}, А	
		U_{R4}, В	
	Ключ замкнут	U_{ab}, В	
		I, А	
		I_{R1}, А	
		U_{R1}, В	
		I_{R2}, А	
		U_{R2}, В	
		I_{R3}, А	
		U_{R3}, В	
		I_{R4}, А	
		U_{R4}, В	

4. Сравнить результаты вычислений с полученными результатами в практической работе №2.

Содержание отчёта

1. Наименование и цель работы.
2. Таблица результатов вычислений.
3. Вывод по работе.

Практическое занятие № 8. Расчёт цепей переменного тока. Сравнение результатов расчетов в Mathcad с результатами моделирования в NI Multisim.

Цель занятия:

Приобрести навыки расчёта параметров цепей переменного тока в Mathcad. Сравнить результаты полученные в практической работе №3 с результатами расчётов в Mathcad.

Задания и методические указания к их выполнению

1. Рассчитать параметры цепи приведённой в практической работе №3 (рисунок 3.1). Исходные данные даны в таблице 3.1 и в результаты измерений из практической работы №3

Полное сопротивление цепи:

$$Z = \frac{U}{I} \tag{8.1}$$

Индуктивное сопротивление катушки:

$$X_L = 2\pi f L \tag{8.2}$$

Емкостное сопротивление конденсатора:

$$X_C = \frac{1}{2\pi f C} \tag{8.3}$$

Реактивное сопротивление:

$$X = X_L - X_C \tag{8.4}$$

Коэффициент мощности:

$$\cos\varphi = \frac{R}{Z} \tag{8.5}$$

Угол сдвига фаз:

$$\varphi = \arcsin\frac{X}{Z} \tag{8.6}$$

Из треугольника мощностей:

$$S = UI \tag{8.7}$$
$$P = S\cos\varphi \tag{8.8}$$
$$Q = S\sin\varphi \tag{8.9}$$

Полученные результаты записать в таблицу 8.1.

Таблица 8.1

Элементы цепи	Вычисленные значения						
	Z, Ом	X, Ом	$cos\varphi$	φ	S, ВА	P, Вт	Q, вар
Цепь в целом							
Резистор							
Катушка индуктивности							
Конденсатор							

2. Рассчитать параметры цепи приведённой в практической работе №3 (рисунок 3.6). Исходные данные даны в таблице 3.1 и в результаты измерений из практической работы №3

Полная проводимость:

$$y = \frac{I}{U} \tag{8.10}$$

Активная проводимость ветви с резистором:

$$g_R = \frac{1}{R} \tag{8.11}$$

Активная проводимость ветви с индуктивностью:

$$g_L = \frac{R_L}{R_L^2 + X_L^2} \tag{8.12}$$

Активная проводимость ветви с конденсатором равна нулю.

Реактивная проводимость ветви с резистором равна нулю.

Реактивная проводимость ветви с индуктивностью:

$$b_L = \frac{X_L}{R_L^2 + X_L^2} \tag{8.13}$$

Реактивная проводимость ветви с конденсатором:

$$b_C = \frac{1}{X_C^2} \tag{8.14}$$

Коэффициент мощности определяется по формуле:

$$\cos\varphi = \frac{g}{y} \tag{8.15}$$

Угол сдвига фаз определяется по формуле^

$$\varphi = \arcsin\frac{b}{y} \tag{8.16}$$

Полная, активная и реактивная мощность определяется по формулам (8.7), (8.8), (8.9), соответственно:

Полученные результаты записать в таблицу 8.2.

Таблица 8.2

Элементы цепи	Вычисленные значения							
	y, См	*G*, См	*b*, См	*cosφ*	*φ*	*S*, ВА	*P*, Вт	*Q*, вар
Цепь в целом								
Резистор								
Катушка индуктивности								
Конденсатор								

Содержание отчёта

1. Наименование и цель работы.
2. Таблица результатов вычислений.
3. Вывод по работе.

Практическое занятие № 9. Основы работы с графическим редактором КОМПАС 3D. Выполнение плана освещения цеха.

Цель занятия:

Изучение программного интерфейса, настроек графического редактора. Изучение условных графических обозначений элементов и буквенно-цифровых обозначений на электрических схемах, выполнение рабочего чертежа осветительной сети цеха.

Общие сведения

1.1 Программный интерфейс графической системы КОМПАС

Для запуска системы необходимо выбрать меню **Пуск / Все программы / АСКОН / КОМПАС-3D V16** . Можно выбрать указателем мыши на поле рабочего стола ярлык программы и дважды щелкнуть левой кнопкой мыши. Чтобы открыть документ, необходимо нажать кнопку **Открыть** на панели инструментов **Стандартная** . Чтобы начать новый документ, нажмите кнопку **Создать** на панели **Стандартная** или выполните команду **Файл/Создать** и в открывшемся диалоговом окне выберите тип создаваемого документа и нажмите ОК .

Для завершения работы следует выбрать меню **Файл / Выход**, комбинацию клавиш **Alt-F4** или щелкнуть на кнопке **Закрыть**.

После запуска программы на экране появится окно с изображением стандартной панели, показанное на рис. 9.1.

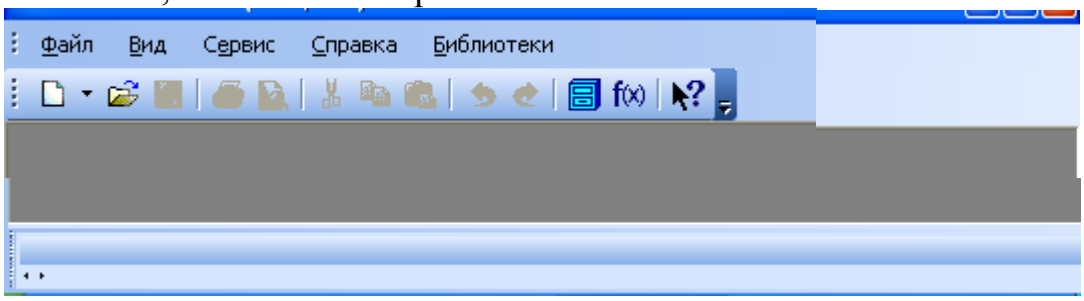


Рисунок 9.1 – Главное окно программы после загрузки системы КОМПАС

Самая верхняя строка служит для вызова выпадающих меню. В середине экрана располагается рабочая область, под которой рациональнее всего располагать место для **Панели свойств**. **Строка сообщений** – самая нижняя строка экрана

Вне зависимости от того, с какими документами приходится работать, на экране всегда рекомендуется отображать панели инструментов **Стандартная**, **Вид**, **Текущее состояние**, **Компактная**. Ниже приведено содержание пунктов **Главного меню.**

1.1.1 Выпадающее меню пункта Файл

В выпадающем меню Файл (рис. 9.2) находятся основные команды работы с файлами документов – **Создать**, **Открыть**, **Сохранить** и т. п. Здесь же находятся команды предварительного просмотра документа, позволяющие оценить, как созданный чертеж будет выглядеть на листе, и команда вывода доку-

мента на печать. В нижней части меню находится список недавно редактированных документов. Можно начать работу с документом, просто выбрав его из этого списка.

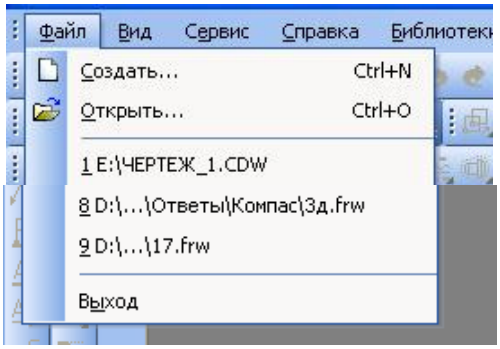


Рисунок 9.2 - Выпадающее меню пункта Файл

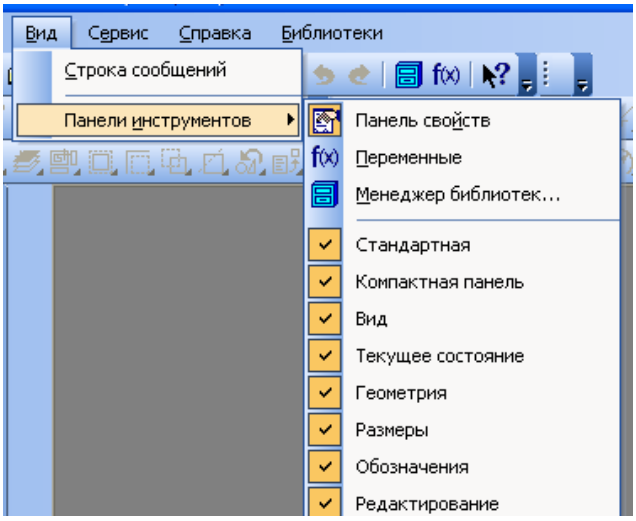


Рисунок 9.3 - Содержание опций меню Вид

1.1.2 Выпадающее меню Вид

Меню **Вид** позволяет активизировать любую панель, воспользовавшись строкой **Панели инструментов**. Для этого нужно щелкнуть левой клавишей мыши в выпадающем меню по пункту **Панели инструментов**.

Появится всплывающее меню, показанное на рис.9.3. Щелкнув мышью по названию нужной панели инструментов во всплывающем меню, увидим, что перед выбранным названием панели появилась галочка в желтом квадрате, а сама панель отображается на экране компьютера.

1.1.3 Выпадающее меню Сервис

В выпадающем меню **Сервис** находятся команды, при помощи которых можно настроить вид документа, тип линий, произвести настройки интерфейса, а также производить различные расчеты (площадь поверхности, объемы и т.д.).

Это меню является контекстно зависимым. Строки этого меню показаны на рис. 9.4.

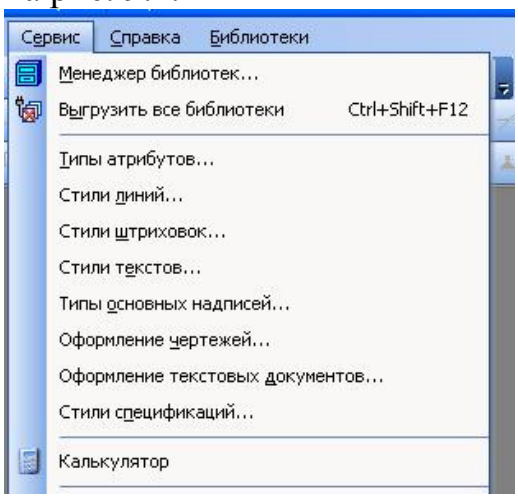


Рисунок 9.4 - Выпадающее меню пункта Сервис

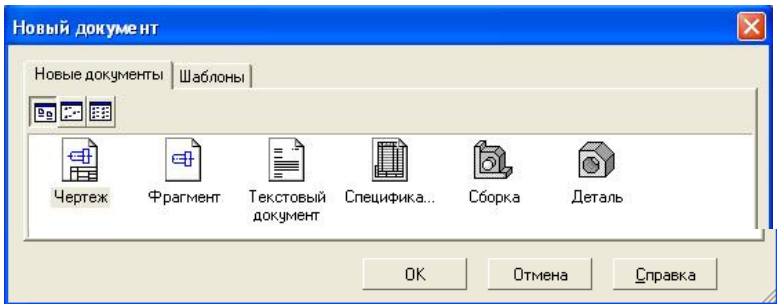


Рисунок 9.5 - Диалоговое окно для выбора типа документа

1.2 Типы документов

Для того чтобы создать новый документ, необходимо щелкнуть по кнопке **Создать** или одновременно нажать сочетание клавиш (**Ctrl+N**). После этого появится диалоговое окно **Новый документ** (рис. 9.5).

Данное окно позволяет выбрать тип создаваемого документа.

1. **Чертеж** – основной тип графического документа в системе КОМПАС. Чертеж содержит графическое изображение изделия в одном или нескольких видах, основную надпись, рамку и всегда содержит один лист заданного пользователем формата (рис. 9.6). Файл чертежа имеет расширение **.cdw**.

2. **Фрагмент** – вспомогательный тип графического документа. Фрагмент отличается от чертежа отсутствием рамки, основной надписи и других объектов оформления конструкторского документа. Во фрагментах хранятся созданные типовые решения для последующего использования в других документах. Файл фрагмента имеет расширение **.frw**.

3. **Текстовый документ** (расширение файла **.kdw**).

4. **Спецификация** (расширение файла **.spw**).

5. **Сборка** (расширение файла **.a3d**).

6. **Деталь** – трехмерное моделирование (расширение файла **.m3d**).

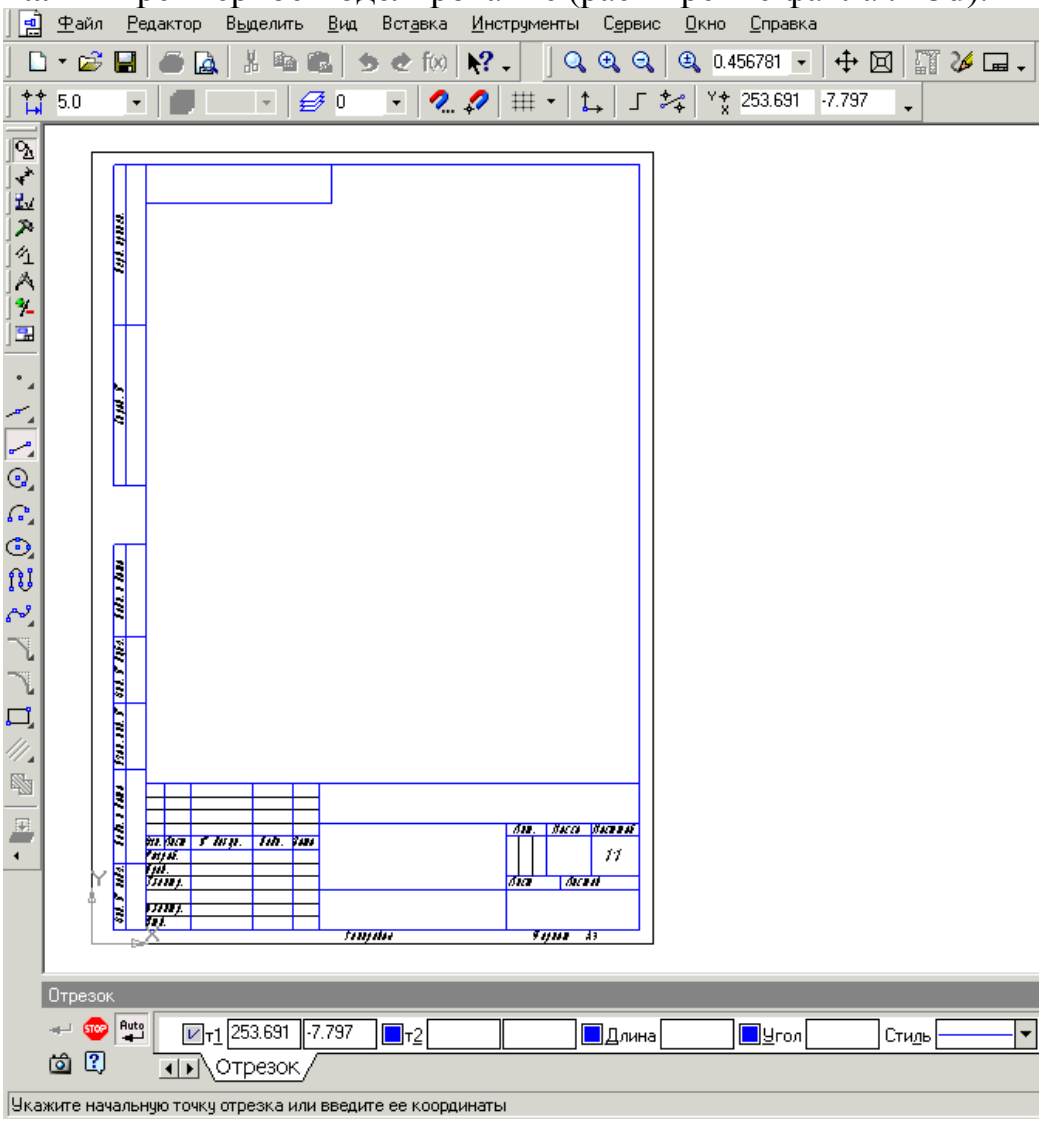


Рисунок 9.6 - Рабочее окно при вызове документа Чертеж

1.3 Панели инструментов

Для удобства работы в системе КОМПАС имеются многочисленные панели инструментов с кнопками, которые соответствуют определенным командам системы. Если указатель мыши задержать на какой-либо кнопке панели инструментов, то через некоторое время появится название этой кнопки, а в строке состояний – краткая расшифровка ее действия. Для активизации кнопки установите на нее указатель мыши и щелкните левой клавишей. Кнопки, имеющие маленький треугольник в правом нижнем углу, могут вызывать расширенную (дополнительную) панель инструментов. Для этого на такой кнопке нужно придержать нажатой левую клавишу мыши, и через некоторое время появится расширенная панель инструментов с кнопками, определяющими различные способы действия.

*1.3.1 Панель инструментов **Стандартная***

Панель инструментов **Стандартная** присутствует практически во всех окнах в различных режимах работы с бóльшим или мéньшим набором кнопок вызова общих команд. Ниже приведена расшифровка кнопок этой панели.

Открыть – вызывает диалоговое окно **Выберите файлы для открытия**.

Сохранить – вызывает диалоговое окно **Выберите файлы для записи**, с помощью которого можно сохранить файл.

Печать – позволяет настроить параметры вывода текстового документа на печать.

Предварительный просмотр – позволяет перейти в режим предварительного просмотра и печати документов.

Вырезать – удаляет выделенные объекты и помещает их в буфер обмена данными.

Копировать – действует так же как и кнопка **Вырезать**, только выделенные объекты остаются на месте.

Вставить – позволяет вставить копию содержимого буфера обмена.

Отменить – отменяет предыдущее действие пользователя, а кнопка **Повторить** – восстанавливает отмененное действие.

Менеджер библиотек – включает или отключает отображение на экране **Менеджера библиотек** – систему управления КОМПАС-библиотеками.

Переменные – включает или отключает отображение на экране диалогового окна **Переменные** для работы с переменными и уравнениями графического документа **Эскиз**

*1.3.2 Панель инструментов **Вид***

Инструментальная панель **Вид** включает кнопки, соответствующие определенным командам:

кнопки, позволяющие управлять масштабом изображения изделия.

Управление ориентацией модели – выводит на экран диалоговое окно **Ориентация вида**. Работает при включении документа **Деталь**.

Сдвинуть – позволяет сдвинуть изображение в активном окне.

Приблизить / отдалить изображение – позволяет плавно менять масштаб, приближая или отдаляя изображение.

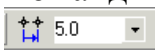
Обновить изображение – позволяет обновить изображение в активном окне. При обновлении масштаб отображения документа в окне не изменяется.

Показать все – изменяет масштаб отображения в активном окне таким образом, чтобы в нем был виден полностью весь документ.

*1.3.3 Панель инструментов **Текущее состояние***

Общий вид панели инструментов Текущее состояние зависит от режима, в котором работает система. Так она выглядит при работе с документом **Чертеж**.

Команды режима работы системы:



Текущий шаг курсора – в поле справа отображает значение шага курсора, то есть расстояние, на которое переместится курсор при однократном нажатии клавиши перемещения.

Состояние видов – выводит на экран диалоговое окно **Состояние видов**, в котором можно изменить параметры существующих видов и создавать новые виды.

Текущий слой – выводит на экран диалоговое окно **Состояние слоев** , в котором можно изменить параметры существующих слоев и создать новые слои.

Настройка глобальных привязок – позволяет включить или отключить какие-либо глобальные привязки и настроить их работу.

Запретить привязки – отключает действие всех глобальных привязок.

Сетка – позволяет включить или отключить отображение вспомогательной сетки в активном окне.

Локальная система координат – позволяет создавать в текущем виде чертежа или фрагмента различные локальные системы координат.

Ортогональное черчение – служит для перехода в режим ортогонального черчения.

Координаты курсора – отображают текущие значения координат курсора по осям в текущей системе координат.

*1.3.4 Панель инструментов «**Компактная**»*

Инструментальная панель **Компактная** облегчает переключение между инструментальными панелями и экономит поле рабочей области. **Панель свойств** предназначена для управления процессом выполнения команды. **Вкладки** Панели свойств содержат поля и переключатели, при помощи которых можно непосредственно определять параметры создаваемых объектов и определять их свойства. Количество вкладок зависит от конкретной команды.

Чтобы перейти на нужную вкладку, необходимо щелкнуть по ней левой клавишей мыши.

*1.3.5 Панель инструментов « **Панель свойств** »*

Панель свойств служит для управления параметрами команды и процессом их выполнения. При этом возможны различные представления одной и той же **Панели свойств**. Например, на рис. 9.7 представлено изображение **Панели свойств** при выполнении команды **Окружность**.

Рисунок 9.7 - Вид Панели свойств при выполнении команды Окружность

Слева от **Панели свойств** расположена **Панель специального управления** (рис. 9.7), которая позволяет контролировать процесс выполнения текущей команды.

В **Панели специального** управления расположены изображения следующих кнопок:

Создать объект – фиксирует создаваемый или редактируемый объект. Используется в том случае, если отключено автоматическое создание объекта.

Прервать команду – завершает выполнение текущей команды ввода или редактирования объекта.

Автоматическое создание объектов – (по умолчанию нажата). Если оставить эту кнопку нажатой, то все объекты будут создаваться немедленно после задания необходимого количества параметров. Если кнопка не нажата – параметры можно варьировать, оценивая их правильность по фантому (контур в тонких линиях) объекта.

Вызов справки – позволяет получить справку по выполнению текущей команды.

Запомнить состояние.

1.4 Последовательность выполнения рабочего чертежа

1.4.1 Создание документов

Для того чтобы создать новый документ, необходимо:

1) из выпадающего меню пункта **Файл** выбрать команду **Создать**;

2) в появившемся диалоговом окне (рис. 9.5) щелкнуть мышью по пиктограмме документа, который нужно создать, – **Чертеж**.

1.4.2 Задание имени чертежа

После создания документа **Чертеж** появится рабочее поле с изображением формата А4 и наименованием чертежа – « **Чертеж без имени** ». Сохраните этот документ, присвоив ему имя – « **План освещения** ». Для этого необходимо:

1) выбрать пункт **Файл / Сохранить как…** ;

2) указать в появившемся окне папку, где будет сохранен данный документ (например, **Мои документы**);

3) в поле **Имя** ввести « **План освещения** »;

4) в появившемся окне **Информация** о документе на вкладке **Общие сведения** введите свою фамилию, имя и отчество и, если необходимо, комментарии к документу в окне с соответствующим названием.

1.4.3 Настройка формата чертежа

При создании чертежа может потребоваться изменить формат листа, шрифт и внешний вид отдельных элементов. Для этого необходимо получить доступ к настройкам формата.

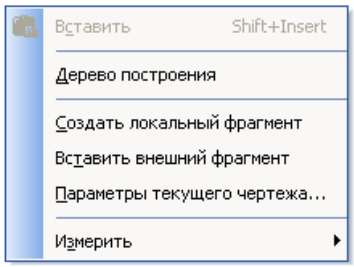


Рисунок 9.8 - Диалоговое окно **Параметры**

Для этого:

1) на рабочем поле чертежа щелкните правой кнопкой мыши и из появившегося меню (рис. 9.8) выберите пункт **Параметры текущего чертежа**;

2) в появившемся диалоговом окне « **Параметры** » выбрать строку **Параметры листа**, а затем строку **Формат** (рис. 9.9);

3) щелкнуть левой кнопкой мыши по строке **Формат**; появится диалоговое окно, в котором необходимо указать требуемый формат листа. Закончив выбор формата и его ориентацию на рабочем поле, необходимо щелкнуть на кнопке **ОК**.

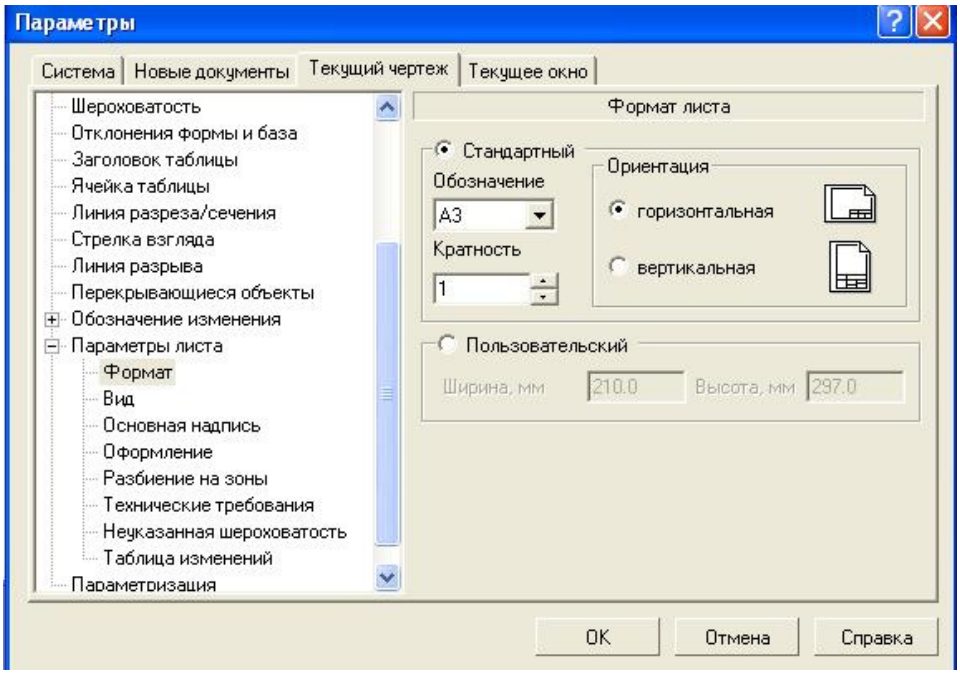


Рисунок 9.9

1.4.4 Масштабирование изображения

На рабочем поле начало отсчета координат ведется от левого нижнего угла чертежа. На панели **Вид** указан масштаб изображения чертежа кнопкой 0.389879 – Текущий масштаб . Для изменения масштаба изображения нужно указателем включить кнопку расширения и выбрать необходимый масштаб.

При необходимости увеличения или уменьшения формата чертежа при работе с изображениями прокрутить колесо мыши. Если формат чертежа неудачно расположен на рабочем поле экрана, можно курсором перемещать его, щелкая по кнопке или полосы прокрутки до тех пор, пока чертеж не займет нужное вам расположение. Можно воспользоваться также кнопкой – **Сдвинуть** на панели **Вид**.

Для увеличения масштаба изображения в окне построений можно воспользоваться инструментом **Увеличить масштаб рамкой**, расположенный на панели инструментов **Вид**. При использовании этой команды нужно графическим курсором указать область экрана, изображение в которой необходимо увеличить.

Для этого необходимо:

1) указать щелчком мыши левый верхний угол ограничивающего прямоугольника;

2) указать правый нижний угол прямоугольника.

После этого выделенная область будет увеличена.

1.4.5 Выбор типов линий построения изображений

При построении геометрических объектов возможно использование различных типов линий, установленных ГОСТ 2.303-68. Для этого используют расширение кнопки **Стиль**, расположенной на **Панели свойств** при включении соответствующей кнопки инструментальной панели **Геометрия** (рис. 9.10).

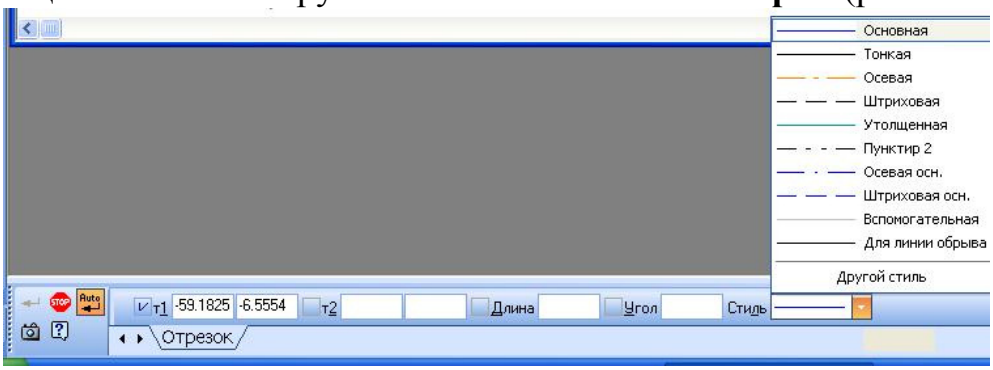


Рисунок 9.10 - Выбор типов линий

1.4.6 Заполнение основной надписи

Команда **Основная надпись** может быть вызвана кнопками **Сервис / Параметры / Параметры листа / Основная надпись**. При помощи этой команды можно заполнять основную надпись автоматически или вручную.

Чтобы основная надпись заполнялась автоматически, необходимо поставить галочку в окошке метки **Синхронизировать основную надпись** (рис. 9.11).

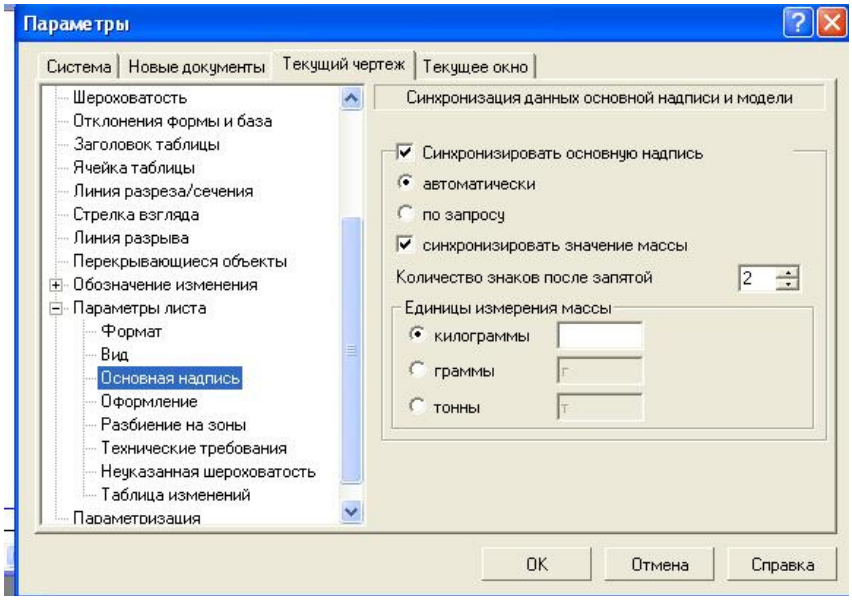


Рисунок 9.11

Пункт **Оформление** позволяет выбрать требуемый стиль оформления листа в соответствии с ГОСТ. По умолчанию используется схема оформления первого листа конструкторского чертежа согласно ГОСТ 2.104-68.

Для заполнения основной надписи необходимо:

1) дважды щелкнуть левой клавишей мыши на поле основной надписи и она примет вид, показанный на рис. 9.12;

2) заполнить требуемые графы основной надписи (размер шрифта выбирается автоматически);

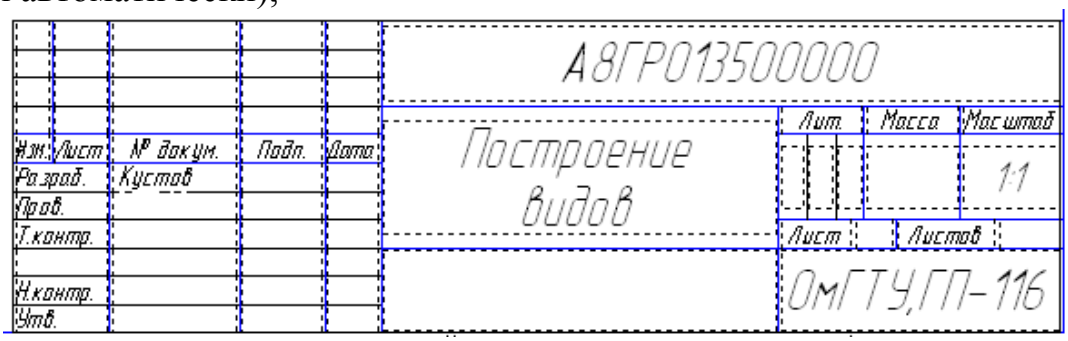


Рисунок 9.12 - Основная надпись в процессе ее заполнения

3) щелкнуть мышью по кнопке **Создать**, расположенной в Панели **специального управления**.

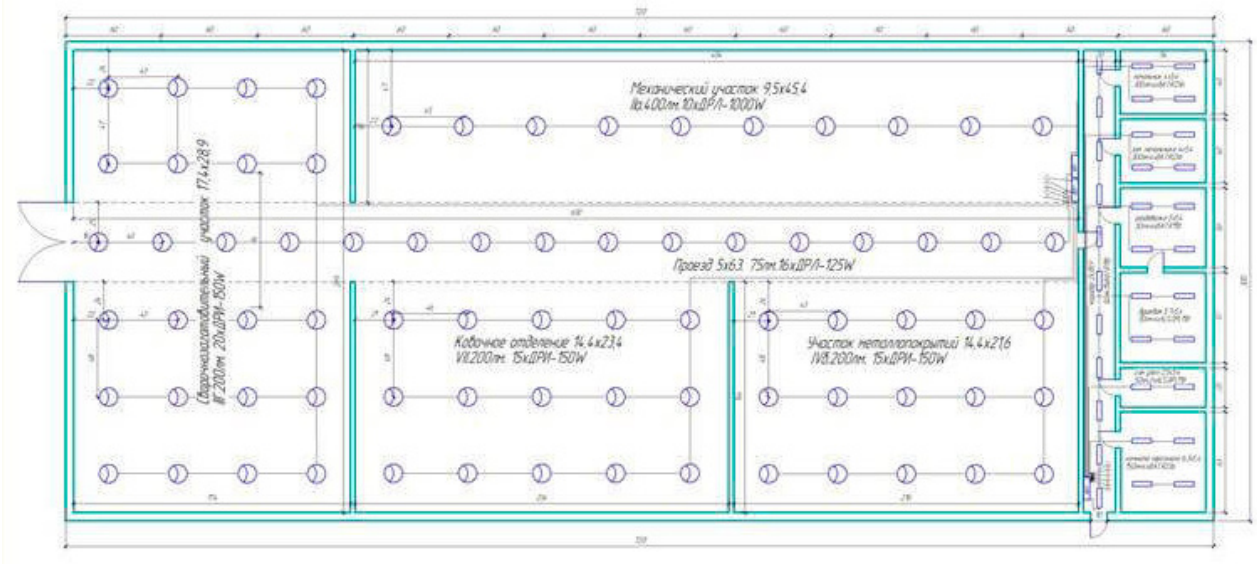
Задание

Согласно своему варианту вычертить на формате А1 графическую работу «План освещения цеха»

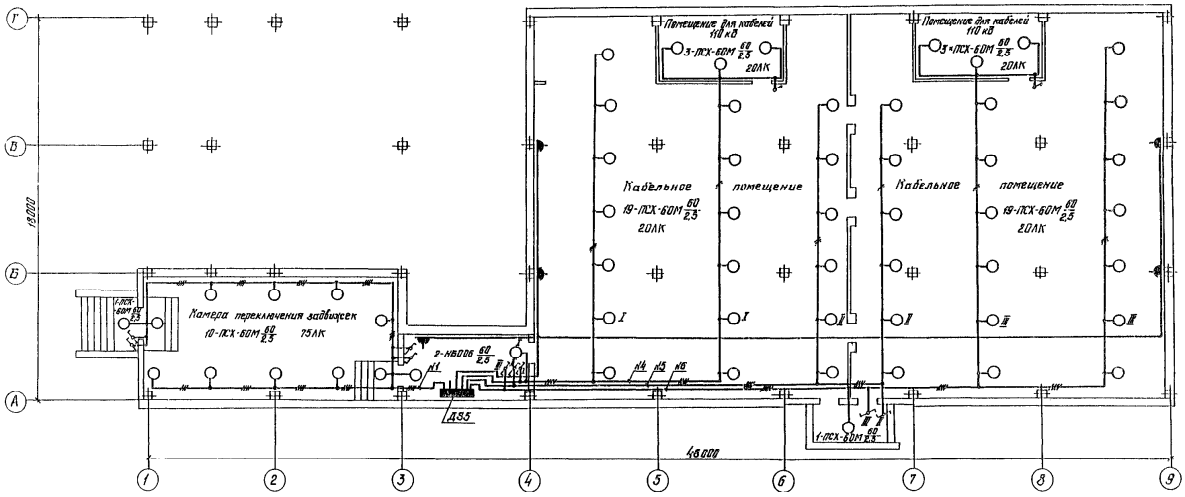
ПОСЛЕДОВАТЕЛЬНОСТЬ РАБОТЫ.

1. Анализ назначения чертежа.
2. Определение элементов системы освещения
3. Определение последовательности построения.
4. Выполнение построений.
5. Заполнение основной надписи.

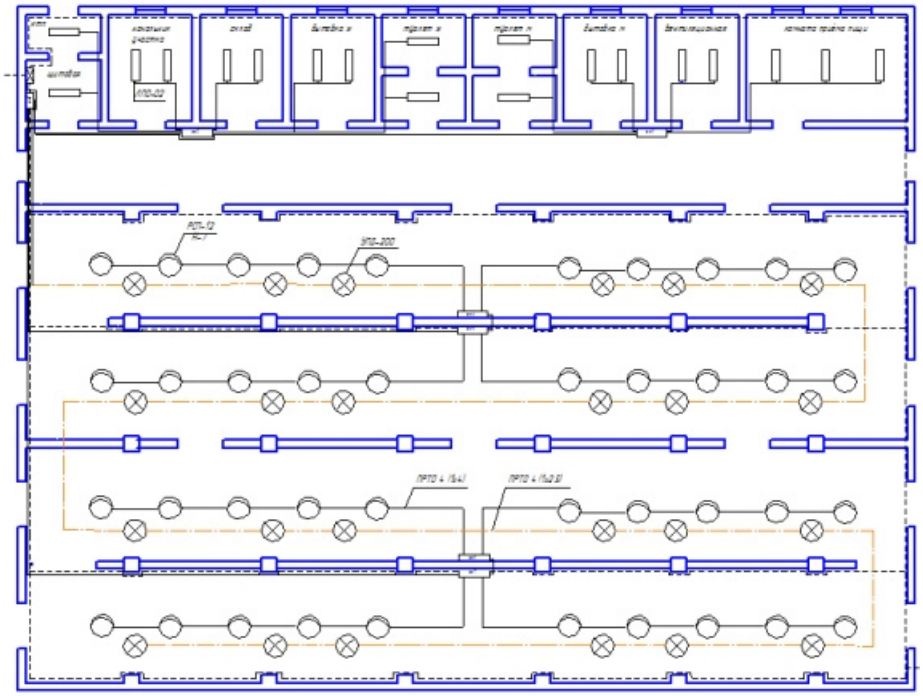
Вариант 1, 9, 17



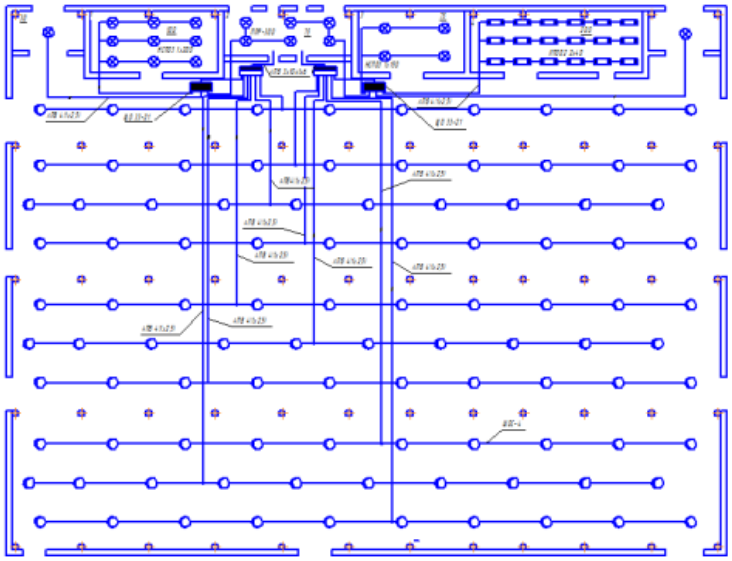
Вариант 2, 10, 18



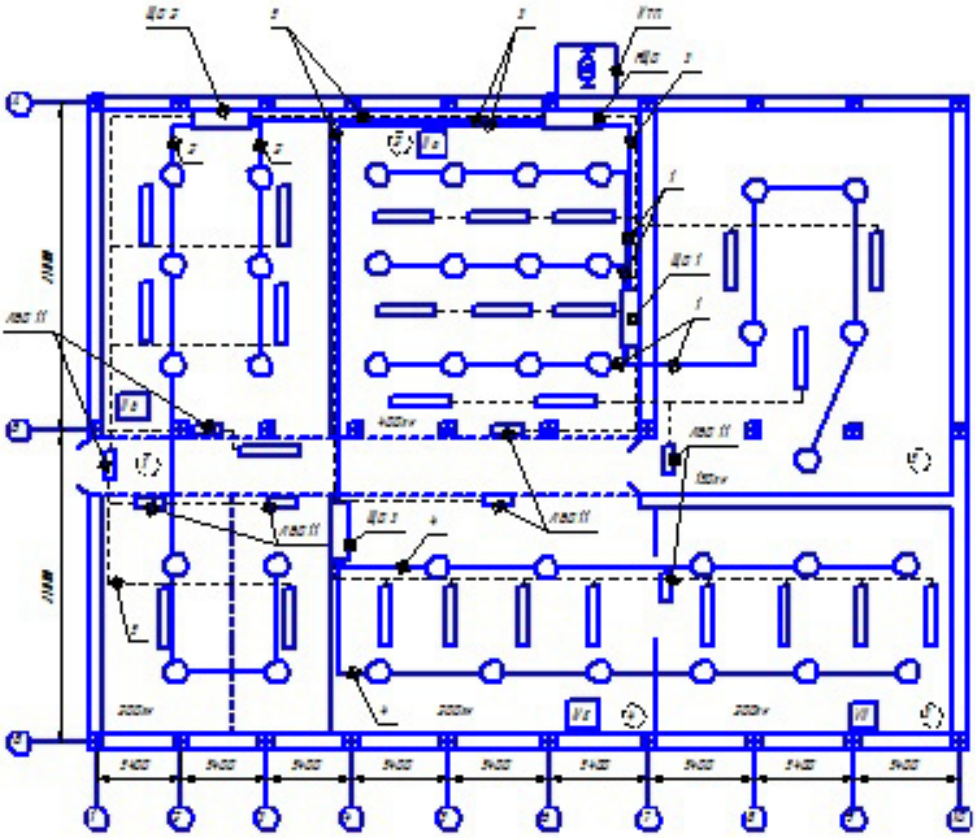
Вариант 3, 11, 19



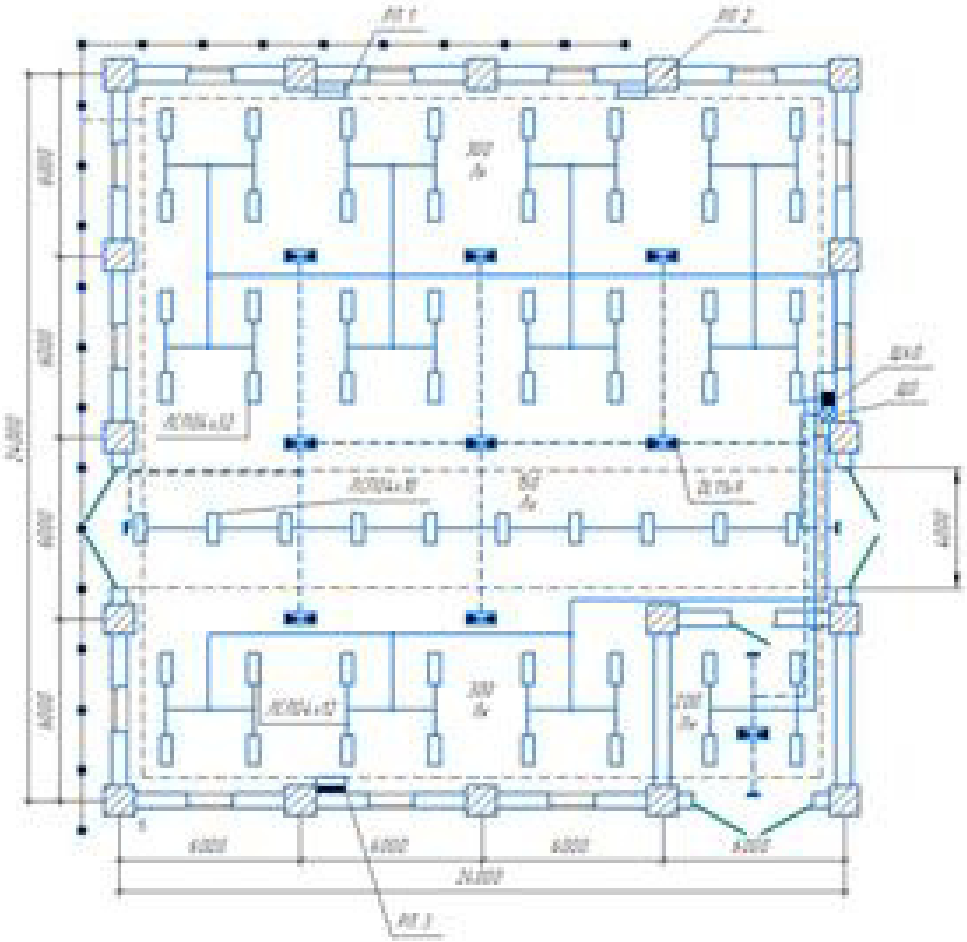
Вариант 4, 12, 20



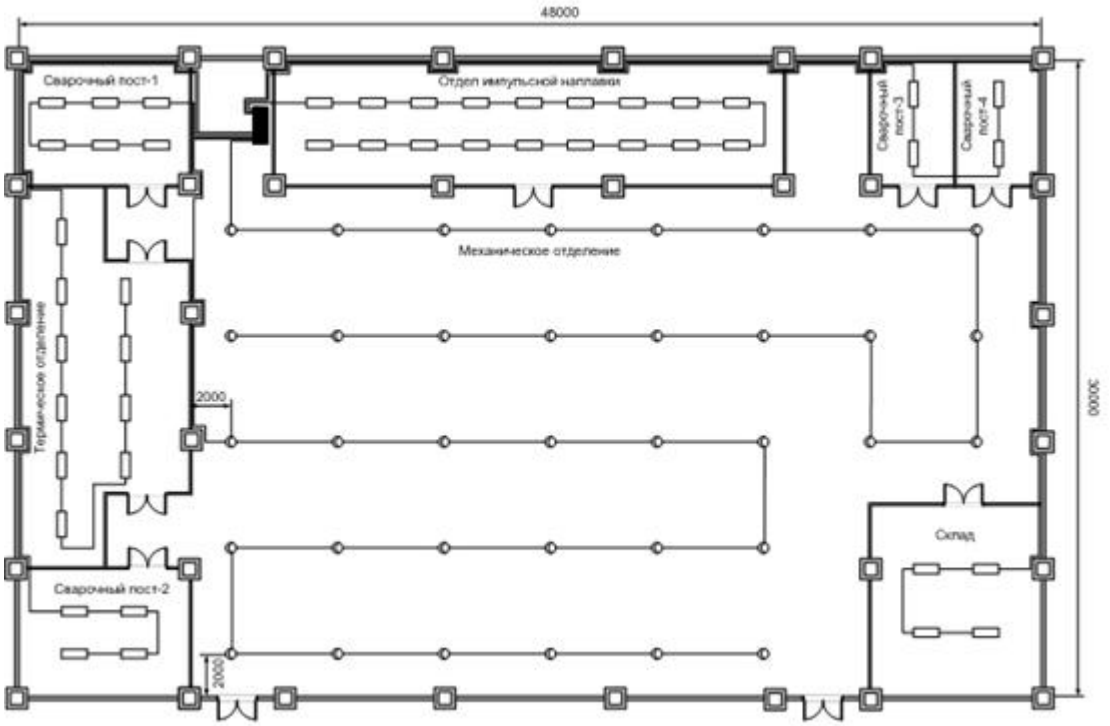
Вариант 5, 13, 21



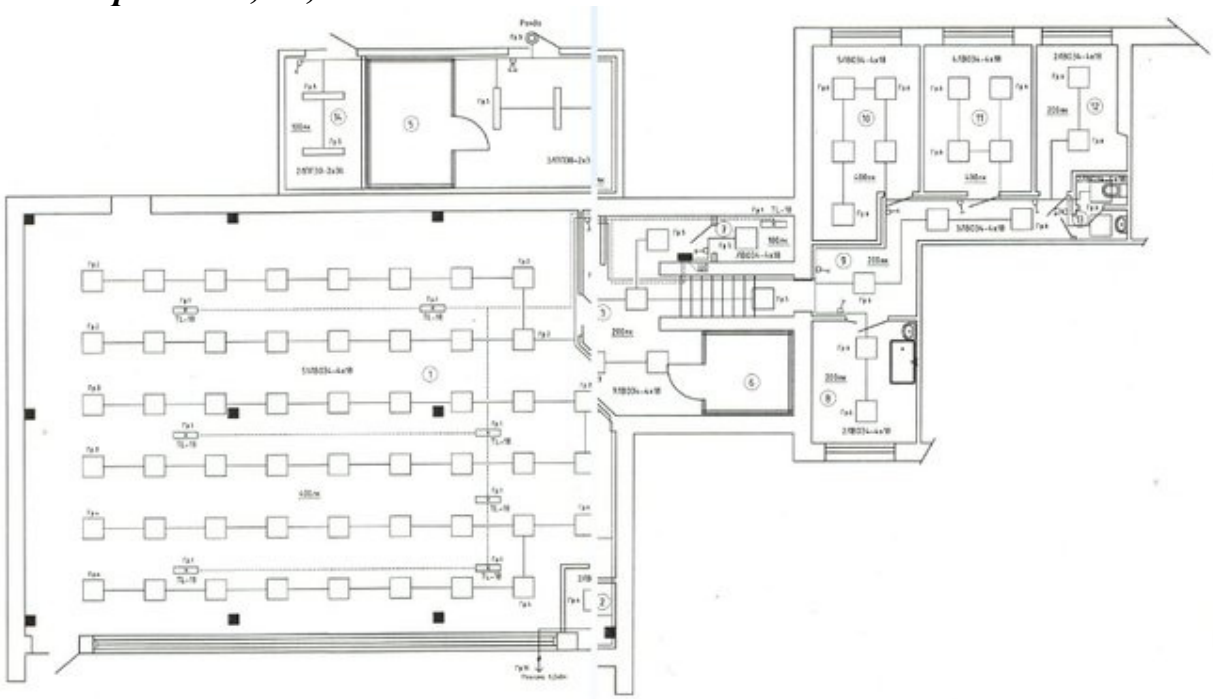
Вариант 6, 14, 22



Вариант 7, 15, 23



Вариант 8, 16, 24



Практическое занятие № 10. Выполнение плана силовой сети цеха в графическом редакторе КОМПАС 3D.

Цель занятия:

Изучение условных графических обозначений элементов и буквенно-цифровых обозначений на электрических схемах, выполнение рабочего чертежа силовой сети цеха.

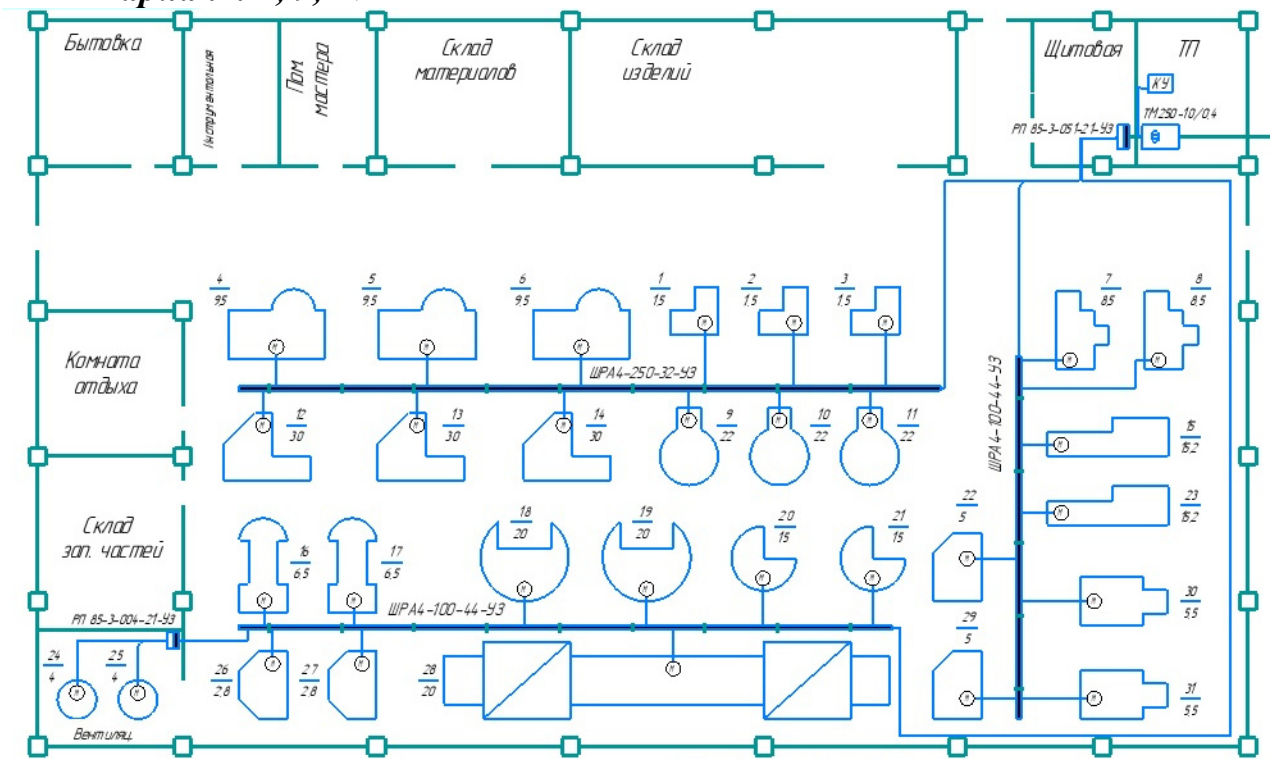
Задание

Согласно своему варианту вычертить на формате А1 графическую работу «План силовой сети цеха»

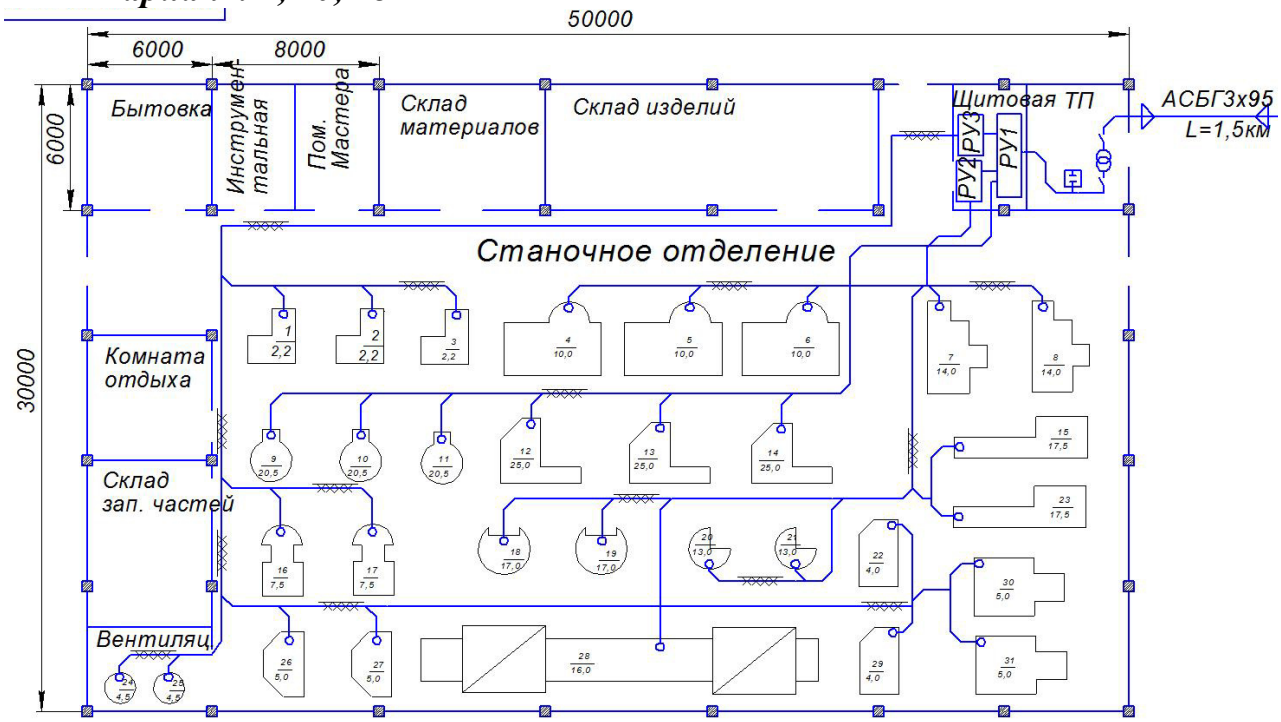
ПОСЛЕДОВАТЕЛЬНОСТЬ РАБОТЫ.

1. Анализ назначения чертежа.
2. Определение элементов системы освещения
3. Определение последовательности построения.
4. Выполнение построений.
5. Заполнение основной надписи.

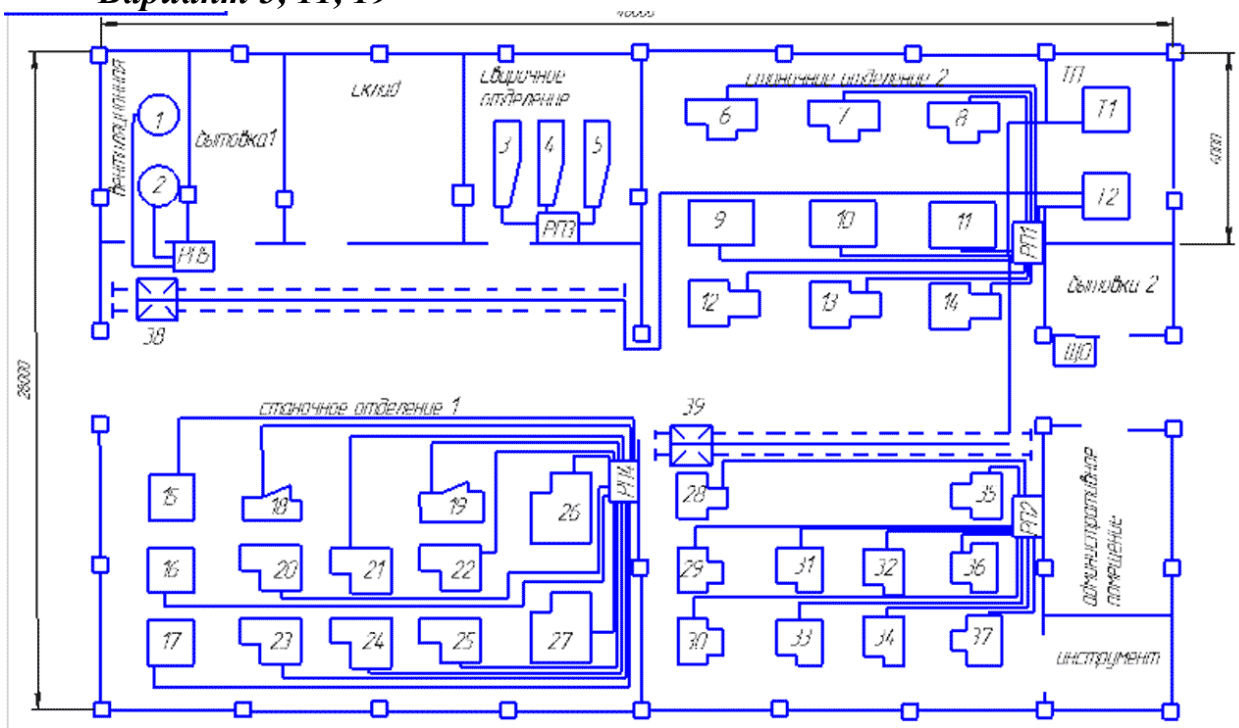
Вариант 1, 9, 17



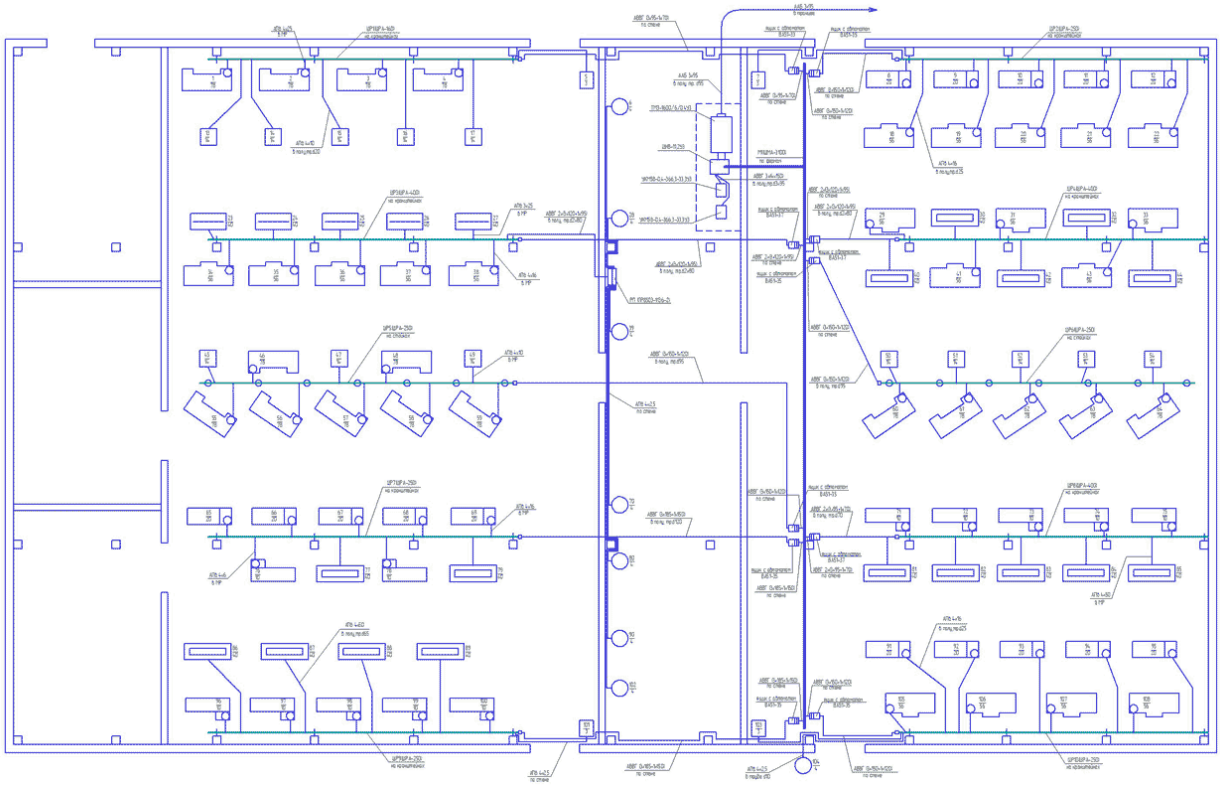
Вариант 2, 10, 18



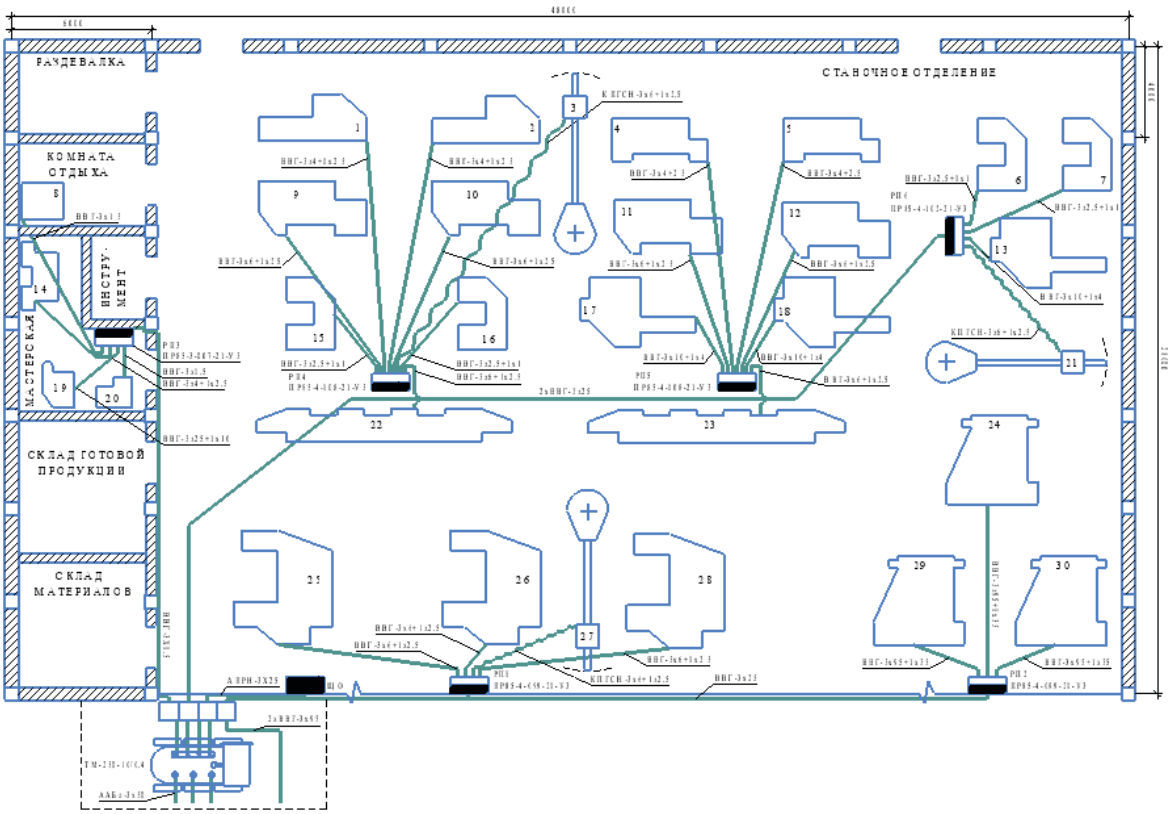
Вариант 3, 11, 19



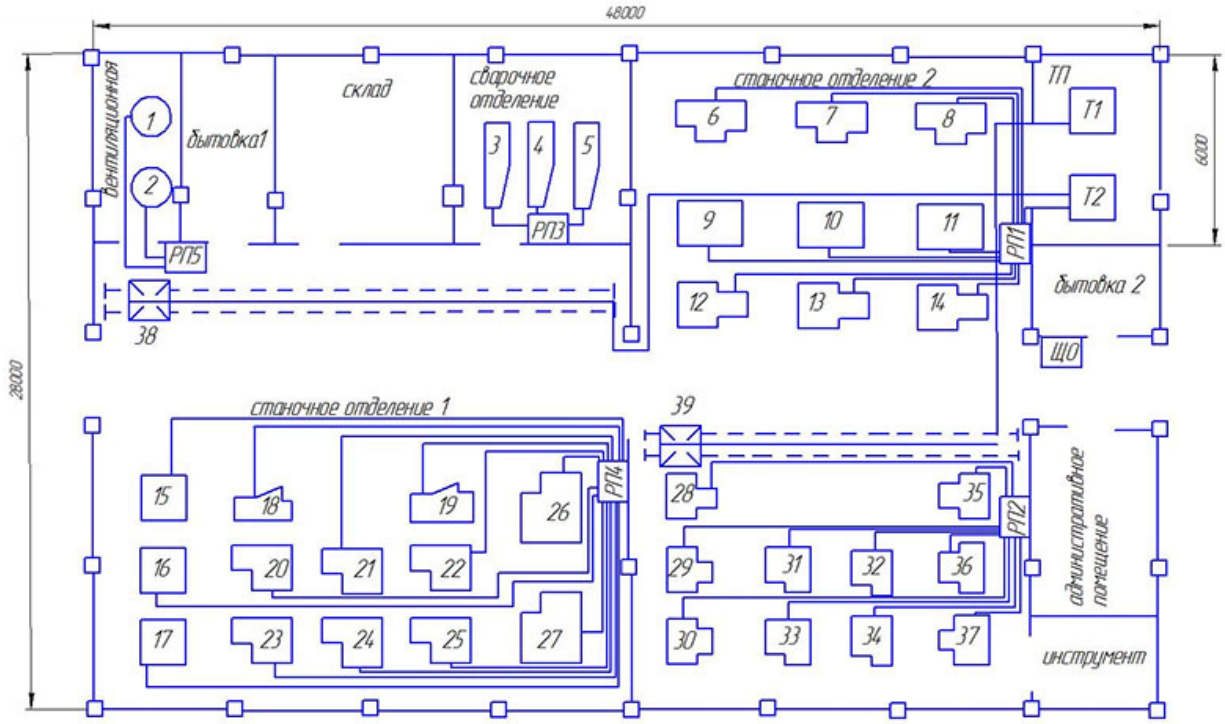
Вариант 4, 12, 20



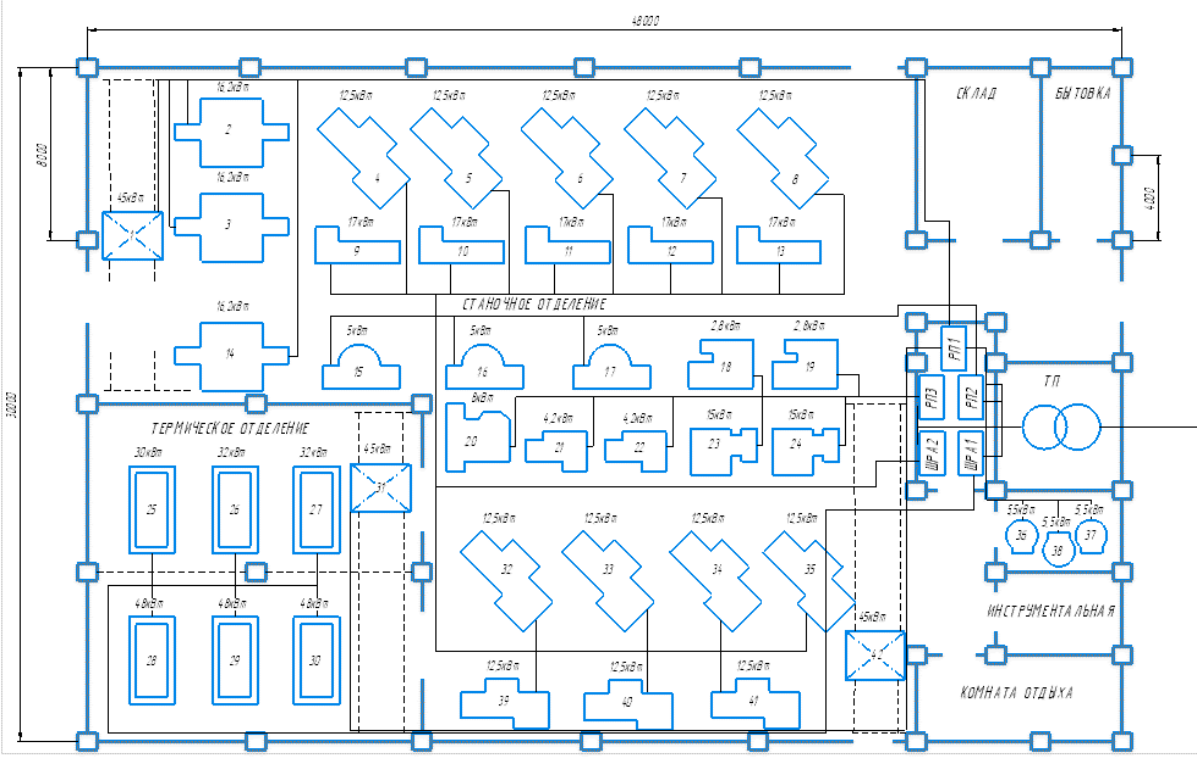
Вариант 5, 13, 21



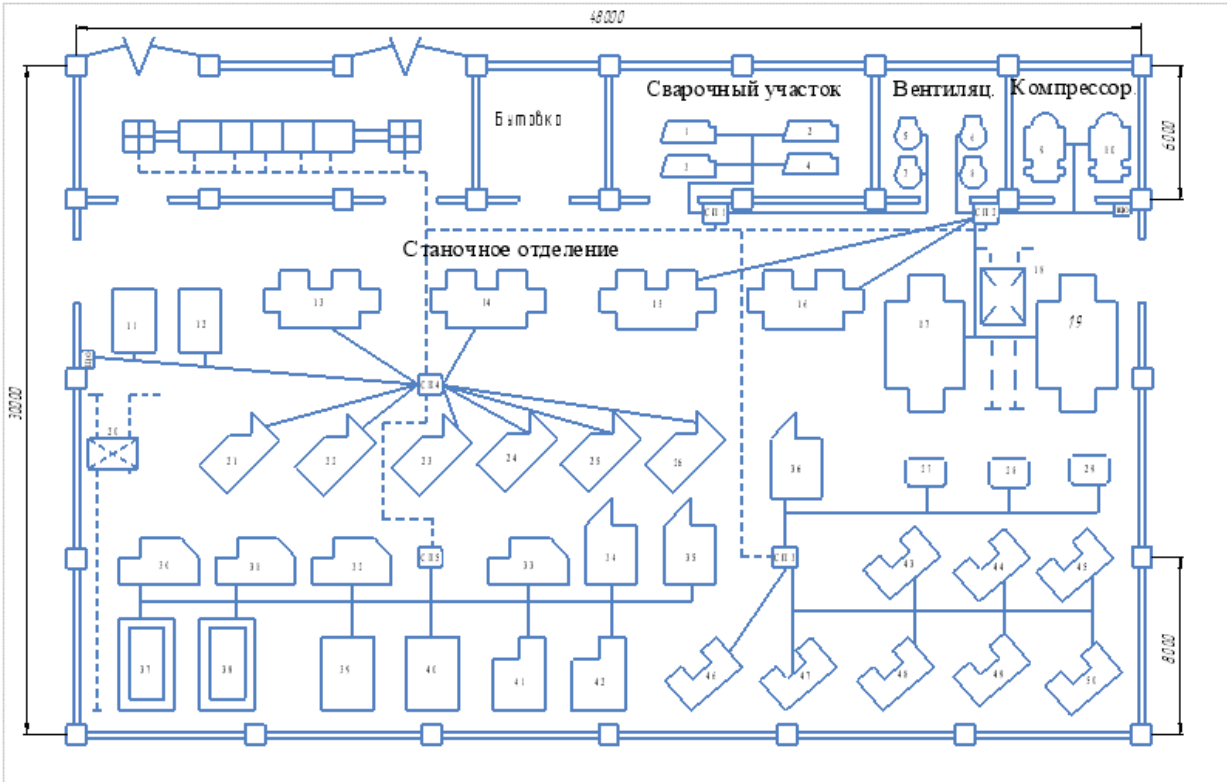
Вариант 6, 14, 22



Вариант 7, 15, 23



Вариант 8, 16, 24



Практическое занятие № 11. Язык программирования СС++. Ввод - вывод данных. Первая программа.

Цель занятия:

Изучение способов ввода текстовых и числовых значений с клавиатуры, вывод данных на консоль

Общие сведения

1. Имена, переменные, константы С++

При решении большинства задач компьютер работает с данными, полученными от пользователя.

Изменяющиеся данные называют – **ПЕРЕМЕННЫМИ**

Постоянные данные называют – **КОНСТАНТАМИ**

Примеры констант:	Примеры переменной величины:
Число дней в неделе - 7 Число месяцев в году -12	Возраст человека Количество учащихся в группе

Для символического обозначения величин, функций и т.п. используются имена. В языке С++ имена переменных, функций, меток называются идентификаторами.

Идентификаторы в языке Си++ – это последовательность знаков, начинающаяся с буквы или знака подчеркивания.

Правила составления имен

Первый символ должен быть буквой или символом подчеркивания, последующие символы должны быть буквами, цифрами или символами подчеркивания. В идентификаторах можно использовать заглавные и строчные латинские буквы, цифры и знак подчеркивания.

Длина идентификаторов произвольная.

Верхние и нижние регистры символов рассматриваются как различные. Следовательно, **count**, **Count** и **COUNT** — это три разных идентификатора.

Идентификатор не может совпадать с ключевым словом С++ или с именем библиотечной функции.

Ниже приведены примеры правильных и неправильных записей идентификаторов:

Правильные	Неправильные
abc A12 NameOfPerson BYTES_PER_WORD high_balance	12X a-b ac_@ zdФ a1.2 az vb

Ключевые слова

Ряд слов в языке Си++ имеет особое значение и не может использоваться в качестве идентификаторов. Такие зарезервированные слова называются ключевыми.

Примеры ключевых слов	
if	void
else	main
case	int
do	bool
enum	float

Переменные

Переменная – это символическое обозначение величины в программе.

Значение переменной (или величина, которую она обозначает) во время выполнения программы может изменяться.

Переменная – это символическое обозначение ячейки оперативной памяти, в которой хранятся данные. Содержимое этой ячейки – это текущее значение переменной.

Объявление переменных

В языке Си++ прежде чем использовать переменную, ее необходимо объявить. Объявить переменную с именем **x** можно так:

Пример: int x

В объявлении первым стоит название типа переменной **int** (целое число), а затем идентификатор **x – имя переменной**.

Тип переменной определяет, какие возможные значения эта переменная может принимать и какие операции можно выполнять над данной переменной.

Тип переменной изменить нельзя.

Пока переменная **x** существует, она всегда будет целого типа.

Константы

Константа – это фиксированное значение, которое не может быть изменено программой. Константа может относиться к любому базовому типу. Способ представления константы определяется ее типом. Константы также называются литералами.

Пример: const float pi = 3.14;

Объявление константы

Синтаксис	Пример
const тип имя;	const bool b;
const тип имя = значение;	const float pi = 3.14;

В С++ принято использовать **символические константы**

Пример: const int dWeek = 7;

Затем в программе в качестве значения количества дней в неделе можно использовать константу **dWeek**:

days_total=4*dWeek;

2. Типы переменных, объявление переменных С++

Язык Си++ – это строго типизированный язык.

Любая величина, используемая в программе, принадлежит к какому-либо типу.

При любом использовании переменных в программе проверяется, применимо ли выражение или операция к типу переменной. Довольно часто смысл выражения зависит от типа участвующих в нем переменных.

Соответствие типов проверяется во время компиляции программы.

Если компилятор обнаруживает несоответствие типа переменной и ее использования, он выдаст ошибку (или предупреждение).

Все переменные перед их использованием должны быть объявлены. Объявление переменной называется также описанием или декларацией.

Общая форма объявления имеет такой вид: **тип список_переменных**;

Здесь тип означает один из базовых типов, а **список_переменных** состоит из одного или более идентификаторов, разделенных запятыми.

Примеры объявлений:

```
int i,j,l;
short int si;
unsigned int ui;
double balance, profit, loss;
```

Описание типов

Тип **char** - Используется для хранения символов, например букв.

Тип **int, long** - Используется для хранения целочисленных значений.

Тип **float** и **double** - Используется для обработки дробных чисел.

Тип **bool** - Предназначен для хранения булевых значений true и false.

Тип **void** - Используется для объявления функции, которая не возвращает значения.

Ошибки использования переменных

Пример несовпадение типа и значения переменной:

1. Объявлена переменная x целого типа

```
int x;
```

2. Переменной присвоено значение (инициализация):

```
x=10.55;
```

Способы объявления переменных

При объявлении переменной она может быть инициализирована.

Общая форма инициализации имеет следующий вид:

тип имя_переменной = константа(значение);

3. Чтение данных с клавиатуры

Выходной поток **cout** позволяет программам записать вывод на экран.

C++ обеспечивает входной поток с именем **cin**, из которого программы могут читать информацию, введенную пользователем с клавиатуры.

Если программы используют выходной поток **cout**, они помещают данные в поток с помощью оператора вставки **(<<).** Подобным образом, если программы применяют **cin** для чтения данных с клавиатуры, они будут использовать оператор извлечения **(>>).**

Оператор ввода

При использовании **cin** для чтения данных с клавиатуры необходимо указывать одну или несколько переменных, которым **cin** будет присваивать входные значения.

Общий вид инструкции ввода данных

cin >>имя_переменной;

Пример программы использует **cin** для чтения числа, введенного с клавиатуры.

Пример использования оператора cin:

```
int k; //объявили переменную
cout <<“Enter number: ”; //вывели подпись
cin >>k; //ввод значения в переменную k
cout<<’Your number: ’<<k;
```

Следующий пример запрашивает два числа. Программа присваивает числа переменным **k** и **m**. Затем программа выводит числа, используя **cout**:

Пример использования оператора cin:

```
int k,m; //объявили переменные
cout <<“Enter numbers: ”; //вывели подпись
cin >>k>>m; //ввод значения в переменную k
cout<<’Your first number: ’<<k<<’\n’;
cout<<’Your second number: ’<<m<<’\n’;
```

Обратите внимание на использование с **cin** двух операторов извлечения:

Пример: cin >> k >> m;

В этом случае **cin** присвоит первое введенное значение переменной **k**, а второе переменной **m**. Если для вашей программы требуется третье значение, вы можете использовать третий оператор извлечения, как показано в примере:

Пример: cin >> k >> m >> d;

4. Операторы. Оператор присваивания. Выражения

Оператор — это часть программы, которая может быть выполнена отдельно. Это означает, что оператор определяет некоторое действие.

В языке C++ существуют следующие группы операторов:

- Операторы присваивания
- Операторы - выражения
- Условные операторы
- Операторы цикла
- Операторы безусловного перехода
- Метки
- Блоки

Оператор присваивания

Оператор присваивания может присутствовать в любом выражении языка C++.

Общая форма оператора присваивания:

Синтаксис оператора присваивания: **имя_переменной=выражение;**

Выражение может быть просто константой или сколь угодно сложным выражением. Адресатом (получателем), т.е. левой частью оператора присваивания должен быть объект, способный получить значение, например, переменная.

Простое присваивание

Операция присваивания устанавливает для переменной текущее значение.

Пример оператора присваивания: **x=10**;

Множественные присваивания

В одном операторе присваивания можно присвоить одно и то же значение многим переменным. Для этого используется оператор множественного присваивания.

Пример оператора присваивания: **x = y = z = 0**;

Следует отметить, что в практике программирования этот прием используется очень часто.

Составное присваивание

Составное присваивание – это разновидность оператора присваивания, в которой запись сокращается и становится более удобной в написании.

Пример 1: **x = x+10**; Пример2**: x += 10**;

Оператор "+=" сообщает компилятору, что к переменной **x** нужно прибавить 10.

Операторы - выражения

Выражения состоят из операторов, констант, функций и переменных. В языке C++ выражением является любая правильная последовательность этих элементов. Большинство выражений в языке C по форме очень похожи на алгебраические, часто их и пишут, руководствуясь правилами алгебры.

Числа в C++

Разделить целой и дробной части в числах – точка!

Задание

Создать проект, где пользователь должен

1. ввести свое имя (текстовое значение) и любое число (числовое данное).
2. результат вывести на экран

Листинг программы (рис.11.1)

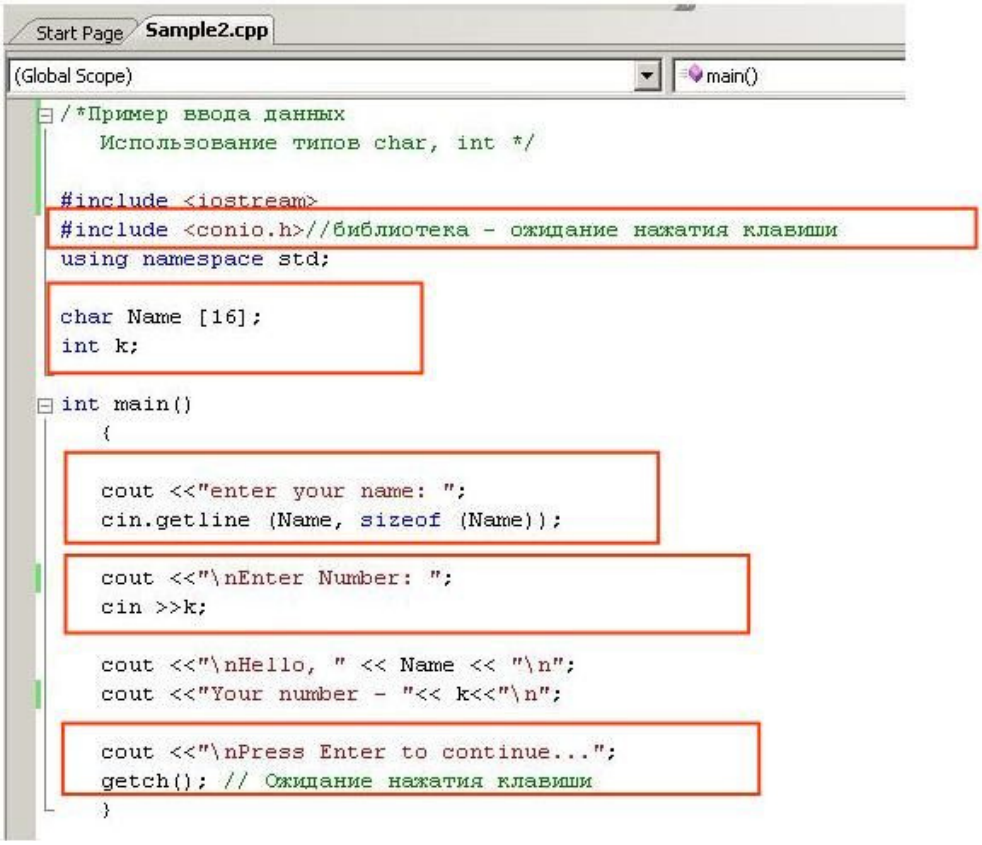


Рисунок 11.1 – Листинг программы

Результат работы программы (рис. 11.2)

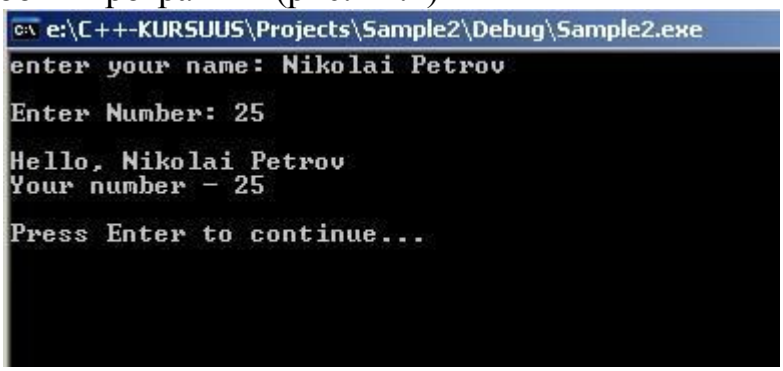


Рисунок 11.2 – Результат работы проекта

Задание:

Вариант 1

1. Задание - проект “Valuta”

Составить проект “**Valuta**” - расчет сумм по курсу валют **евро** и **доллара**.

Сумму обмена в еек вводит пользователь, курс валюты евро и доллара по отношению к рублю – постоянные величины.

Пользователь получает пересчет суммы в евро и долларах.

2. Условия задания

В программе использовать константы **euro=15.70**, **dollar=10.98**

В программе использовать переменные для ввода суммы (в еек) и суммы пересчета в евро и долларах

3. На экране должно быть:

4. Описание вывода данных
   - Продумать и обозначить типы данных для переменных и констант
   - Создать заголовки «**Currency calculators**», «**CURRENCY RATES**»
   - Оформить отступы (табуляция) и межстрочные интервалы
   - Оформить задержку экрана

Вариант 2

1. Задание - проект “**Kaibemaks**”

Составить проект “**Kaibemaks**” - расчет суммы налога с оборота, суммы без налога, суммы по курсу валюты евро из заданной стоимости.

Сумму стоимости товара в еек вводит пользователь. Курс валюты евро, ставка налога с оборота 20%, ставка суммы с налогом 120% – постоянные величины.

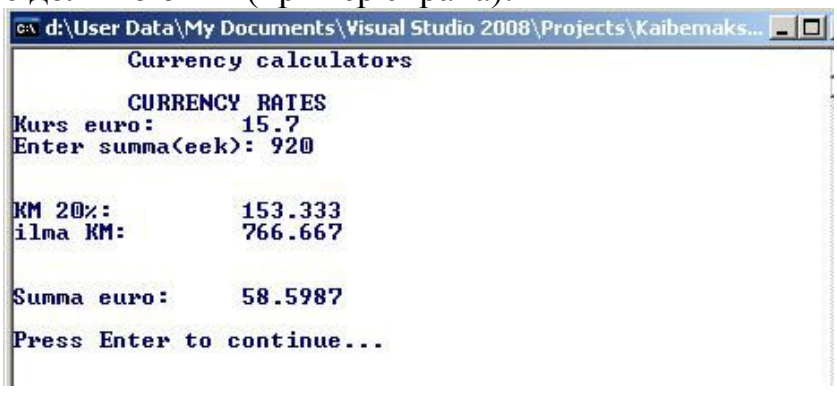
Пользователь получает расчет суммы налога KM, сумму без налога - ilma KM, и пересчет суммы стоимости в евро.

2. Условия задания

В программе использовать константы **euro=15.70**, **percent=20**, **percentKokku=120**.

В программе использовать переменные для ввода суммы (в еек), расчетные величины: налог (KM), сумма без налога (ilma_KM) и сумма пересчета в евро

3. На экране должно быть (пример экрана):



4. Описание вывода данных
   - Продумать и обозначить типы данных для переменных и констант
   - Создать заголовки «**Currency calculators**», «**CURRENCY RATES**»
   - Оформить отступы (табуляция) и межстрочные интервалы
   - Оформить задержку экрана

Практическое занятие № 12. Язык программирования СС++. Условный оператор.

Цель занятия:

Использование условного оператора.

Общие сведения

Операторы управления определяют, в какой последовательности выполняется программа.

Если бы их не было, операторы программы всегда выполнялись бы последовательно, в том порядке, в котором они записаны.

Условные операторы

В языке С существуют два условных оператора: **if** и **switch**. При определенных обстоятельствах оператор **?** является альтернативой оператора **if**.

Условный оператор позволяет выбрать один из вариантов выполнения действий в зависимости от каких-либо условий.

Условие – это логическое выражение, т.е. выражение, результатом которого является логическое значение - **true** (истина) или **false** (ложь).

Общая форма оператора if следующая:

if (условное выражение) оператор;

else оператор;

Здесь оператор может быть только одним оператором, блоком операторов или отсутствовать (пустой оператор).

Фраза **else** может вообще отсутствовать.

Если выражение истинно (т.е. принимает любое значение, отличное от нуля), то выполняется оператор или блок операторов, следующий за **if**.

В противном случае выполняется оператор (или блок операторов), следующий за **else** (если эта фраза присутствует).

Необходимо помнить, что выполняется или оператор, связанный с **if**, или с **else**, но оба — никогда!

Пример:

```
if (x > y)
a = x;
else
a = y;
```

В данном примере переменной **a** присваивается значение максимума из двух величин **x** и **y**.

Конструкция **else** необязательна

Приведенный фрагмент программы изменит значение переменной **x** на его абсолютное значение и присвоит переменной **d** новое значение **x**.

Пример:

```
if (x < 0)
x = -x;
d = x;
```

В примере оператор **x = -x**; выполняется только в том случае, если значение переменной **x** было отрицательным.

Присваивание переменной **d** выполняется в любом случае.

Оператор if - блоки кода

Если в случае истинности условия необходимо выполнить несколько операторов, их необходимо заключить в фигурные скобки:

Пример:

```
if (x < 0)
{
x = -x;
cout << "Изменить значение x на противоположное по знаку";
}
d = x;
```

Теперь, если **x** отрицательно, то не только его значение изменится на противоположное, но и будет выведено соответствующее сообщение.

Фактически, заключая несколько операторов в фигурные скобки, мы сделали из них один сложный оператор или блок.

Блоки кода

Прием заключения нескольких операторов в блок используется везде, где нужно поместить несколько операторов вместо одного.

Блок – это набор логически связанных конструкций.

Конструкции else if

Условный оператор **if** можно расширить для проверки нескольких условий:

Пример:

```
if (x < 0)
cout << "Отрицательная величина";
else if (x > 0)
cout << "Положительная величина";
else
cout << "Ноль";
```

Конструкций **else if** может быть несколько.

Условные выражения. Таблицы условий

Если выполнение программного кода может происходить при выполнении нескольких условий, то в условном выражении используется составное условное выражение.

Условные выражения объединяются логической операцией И (&&) – логическое умножение, ИЛИ (||) – логическое сложение. Возможен вариант использования логического отрицания.

Всё условное выражение становится истинным или ложным в зависимости от результата вычисления каждого отдельного выражения, объединенного логической операцией.

Таблицы условий

Логическое умножение && (и)

A	B	A&&B
True (1)	True (1)	True (1)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	False (0)

Логическое сложение || (или)

A	B	A||B
True (1)	True (1)	True (1)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

Логическое отрицание != (НЕ)

A	!=A
True (1)	False (0)
False (0)	True (1)

Приведенный фрагмент программы изменит значение переменной **x** на его абсолютное значение и присвоит переменной **d** новое значение **x**.

Пример:

```
if (month >=3) &&(month<=5)
cout >>’Это весенние месяцы’;
```

Задания

Задание 1

1. Создать проект - расчет суммы и разницы двух чисел

Пользователь вводит два числа – a, b.

Результат решения задачи:

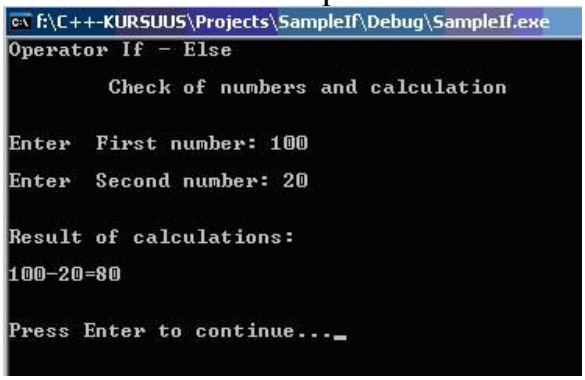
1. Выдать сумму чисел (a+b), если числа равны между собой
2. Выдать разницу (a-b), если число a>b
3. Выдать разницу (b-a), если число a<b

2. Условия задания

Для решения задания использовать операции сравнения данных, оператор вывода результата на экран

3. На экране должно быть:

Пример, если первое число больше второго



4. Описание вывода данных

- Продумать и обозначить типы данных для переменных
- Создать заголовки:
  - ✓ Operator if – else
  - ✓ Check of numbers and calculation
- Оформить отступы (табуляция) и межстрочные интервалы
- Оформить задержку экрана.

Задание 2

1. Создать проект - тест

Составить программу тест из пяти вопросов и трех вариантов ответа на каждый вопрос.

1. Посчитать количество правильных ответов

2. Выдать рекомендации - комментарии по результату количества правильных ответов.

2. Условия задания

- Придумать тему для теста и подобрать вопросы и по три варианта ответа (только один ответ верный)
- Организовать регистрацию учащегося (ввод имени, группы)

3. Описание вывода данных

- Продумать и обозначить типы данных для переменных
- Создать заголовки:
  - ✓ Данные учащегося
  - ✓ Название темы тест
- Оформить отступы (табуляция) и межстрочные интервалы
- Оформить задержку экрана

Задание 3

1. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных пользователем данных и выводить результат на экран. Если числа неверные (делитель равен нулю), выдавать сообщение об ошибке.

2. Пользователь вводит 3 числа, найти максимальное и минимальное из них.

3. Пользователь вводит два числа. Определить и вывести, как соотносится первое и второе число: первое больше второго, или первое меньше второго и числа равны между собой.

Практическое занятие № 13. Язык программирования СС++. Оператор цикла.

Цель занятия:

Использование для решение задачи оператора for, изучение вложенных циклов

Общие сведения

Во всех процедурных языках программирования циклы **for** очень похожи. Однако в С++ этот цикл особенно гибкий и мощный.

Синтаксис оператора for следующий:

for (инициализация; условие; приращение)

{ операторы; }

Цикл **for** может иметь большое количество вариаций. В наиболее общем виде принцип его работы следующий.

1. Инициализация — это присваивание начального значения переменной, которая называется параметром цикла.

2. Условие представляет собой условное выражение, определяющее, следует ли выполнять оператор цикла (часто его называют телом цикла) в очередной раз.

3. Оператор приращение осуществляет изменение параметра цикла при каждой итерации.

Эти три оператора (они называются также секциями оператора **for**) обязательно разделяются точкой с запятой.

Цикл **for** выполняется, если выражение условие принимает значение **ИСТИНА**.

Если оно хотя бы один раз примет значение **ЛОЖЬ**, то программа выходит из цикла и выполняется оператор, следующий за телом цикла **for** .

Пример №1 использования оператора for

Задание: Вычислить сумму всех целых чисел от 0 до 100.

0+1+2+3+4+5+6+….+98+99+100

Решение такой задачи проще всего выполнить с помощью оператора цикла for:

Пример кода for

```
int sum = 0;
int i;
for (i = 0; i <= 100; i = i + 1) // заголовок цикла
{
sum = sum + i; // тело цикла
}
```

4. Количество повторений цикла определяется начальным значением переменной-счетчика, условием выполнения цикл и значением приращения счетчика;

5. Переменная-счетчик должна быть целого (**int**) типа и может быть объявлена непосредственно в инструкции цикла.

Важное для оператора цикла for

1. Оператор **for** реализует фундаментальный принцип вычислений в программировании – итерация.

2. Тело цикла повторяется для разных, в данном случае последовательных, значений переменной **i**.

3. Повторение называется итерацией.

4. Проходя по последовательности значений переменной **i** и выполняя с текущим значением одно и то же действие, тем самым постепенно вычисляем нужное значение.

Пример №2 использования оператора for

Задание: Вывести на экран числа от 1 до 100

Реализация задачи:

Пример программного кода

```
int main()
{ int x;
for(x=1; x <= 100; x++)
cout<<x;
return 0;
}
```

В этом примере параметр цикла **x** инициализирован числом 1, а затем при каждой итерации сравнивается с числом 100. Пока переменная **x** меньше 100, вызывается функция **cout** и цикл повторяется. При этом **x** увеличивается на 1 и опять проверяется условие цикла **x <= 100**. Процесс повторяется, пока переменная **x** не станет больше 100. После этого процесс выходит из цикла, а управление передается оператору, следующему за ним. В этом примере параметром цикла является переменная **x**, при каждой итерации она изменяется и проверяется в секции условия цикла.

Пример №3. В следующем примере в цикле **for** выполняется блок операторов:

Пример программного кода

```
for(x=100; x != 65; x -= 5)
{ z = x*x;
cout<< "Квадрат “<<x<<” равен ”<<z<<“\n“; }
```

Операции возведения переменной **x** в квадрат и вызова оператора вывода данных **cout** повторяются, пока **x** не примет значение 65. Обратите внимание на то, что здесь параметр цикла уменьшается, он инициализирован числом 100 и уменьшается на 5 при каждой итерации.

Бесконечный цикл

Для создания бесконечного цикла можно использовать любой оператор цикла, но чаще всего для этого выбирают оператор **for**. Так как в операторе **for** может отсутствовать любая секция, бесконечный цикл проще всего сделать, оставив пустыми все секции.

Решение задачи:

Пример бесконечного цикла

```
int sum = 0;
for ( ; ; )
{
cout<< "Этот цикл крутится бесконечно.\n";
}
```

Если условие цикла **for** отсутствует, то предполагается, что его значение – **ИСТИНА**. В оператор **for** можно добавить выражения инициализации и приращения, хотя обычно для создания бесконечного цикла используют конструкцию **for(; ;)**.

Задание

Создайте проект SampleFor_SummaNumber

1. Вычислить сумму всех последовательных целых чисел, начиная от 1 до 3
2. Вычислить сумму всех последовательных целых чисел, начиная с единицы до числа, переданного пользователем.
3. Вывести промежуточные суммы.

Описание решения

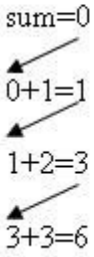
1. Для решения задачи создадим функцию **total()**, назначение которой вывод на экран

данных.

2. В проекте используем переменную **sum**, которая будет накапливать промежуточную сумму цифр.
3. Решение организуем таким образом:
   - сначала выведем сумму цифр от 1 до 3, показывая промежуточные суммы
   - далее запросим от пользователя число и выведем сумму цифр от 1 до числа, заданного пользователем. Вывод предусматривает отображение промежуточных сумм.
4. Алгоритм решения выглядит так:

Алгоритм решения проекта сумма чисел от 1 до 3

Решение будет выглядеть так:



Решение проекта. Программа "Сумма цифр"

1. Листинг программы

Объявление библиотек (рис.13.1)

```
/*Программа for
        1. вывод суммы цифр,
        2. вывод промежуточной суммы*/
//---------------------------------------------------------------------
#include <iostream>
#include <conio.h>//библиотека - ожидание нажатия клавиши
using namespace std;

void total(int x);//функция суммирования цифр

//---------------------------------------------------------------------
```

Рисунок 13.1 - Объявление библиотек

Функция **total()** - расчет сумм цифр (рис.13.2)

```
//---------------------------------------------------------------------
void total(int x)
{
int sum = 0;
int i;

    for (i=1; i<=x;i++)
    {
        sum=sum+i;

        cout <<"Промежуточная сумма - "<< sum<<"\n";//вывод промежуточной суммы
    }
}
//
```

Расчет сумм цифр

Рисунок 13.2 – Функция total()

Функция **main()** (рис.13.3)

```
//---------------------------------------------------------------------
int main()
{
    int n;
    setlocale(LC_CTYPE,""); //использование кирилицы
    cout<<"Сумма цифр";
    cout<<"\n\nЦифр суммирования...3\n";//количество цифр для суммы -3
    n=3;
total(n);
//ввод количество цифр для суммы
    cout<<"\n\nВведите количество цифр для суммирования...";
    cin>>n;
total(n);

//---------------------------------------------------------------------
    cout <<"\n\nНажмите клавишу для продолжения...";
    _getch(); // Ожидание нажатия клавиши

return 0;
}
//---------------------------------------------------------------------
```

Вызов функции total()

Вызов функции total()

Рисунок 13.3 – Функция main()

Измените функцию **total()** (рис.13.4)
Добавьте цикл создания отступов в печати данных

```
//---------------------------------------------------------------------
void total(int x)
{
int sum = 0;
int i,count;

    for (i=1; i<=x;i++)
    {
        sum=sum+i;

            for (count=1;count<=10;count++) cout <<".";//печать отступов

        cout <<"Промежуточная сумма - "<< sum<<"\n";//вывод промежуточной суммы
    }
}
//---------------------------------------------------------------------
```

Рисунок 13.4 – Изменения функции total()

Результат работы программы

```
E:\C++-KURSUUS\Projects\Sample_ForSumma\Debug\ForSumma.exe
Сумма цифр

Цифр суммирования...3
..........Промежуточная сумма - 1
..........Промежуточная сумма - 3
..........Промежуточная сумма - 6


Введите количество цифр для суммирования...5
..........Промежуточная сумма - 1
..........Промежуточная сумма - 3
..........Промежуточная сумма - 6
..........Промежуточная сумма - 10
..........Промежуточная сумма - 15


Нажмите клавишу для продолжения...
```

Вариант 1.

1. Используя оператор цикла for, напечатать в одну строку цифры от 1 до 10

2. Используя вложенный цикл, напечатать 7 строк цифр от 1 до 10

Вариант 2.

1. Используя оператор цикла for, напечатать в одну строку цифры от 10 до 1

2. Используя вложенный цикл, напечатать заданное пользователем количество строк цифр от 10 до 1

Практическое занятие № 14. Язык программирования СС++. Массивы.

Цель занятия:

Познакомиться с возможностью сохранения данных в массиве переменных, практической обработкой массива при помощи цикла, порядком вывода значений массива на экран

Общие сведения

Доступ к конкретному элементу массива осуществляется с помощью индекса. В языке С++ все массивы располагаются в отдельной непрерывной области памяти.

Первый элемент массива располагается по самому меньшему адресу, а последний – по самому большому. Массивы могут быть одномерными и многомерными.

Строка представляет собой массив символьных переменных, заканчивающийся специальным нулевым символом, это наиболее распространенный тип массива.

Под массивом в языке С++ понимают набор данных одного и того же типа, собранных под одним именем.

1. Массив определяется именем массива и порядковым номером каждого своего элемента.
2. Обычно порядковый номер элемента называют индексом.
3. Массив имеет размер – количество элементов в нем.
4. Каждый элемент массива нумеруется, номер называют индексом.
5. Обращение к элементу массива осуществляется путем указания его индекса.
6. Индекс – то есть порядковый номер элемента – всегда целое число.
7. Нумерация элементов массива в языке С++ всегда начинается с 0
8. Индекс последнего элемента массива на единицу меньше размера массива.

Примеры массивов:

```
int numbers[1000]; // массив на 1000 числовых элементов.
float test[10]; // массив на 10 вещественных чисел
```

Различают:

1. Одномерные массивы (один индекс)
2. Многомерные массивы (два и более индексов)

Одномерные массивы

Как и другие переменные, массив должен быть объявлен явно, чтобы компилятор выделил для него определенную область памяти (т.е. разместил массив). Здесь тип обозначает базовый тип массива, являющийся типом каждого элемента.

Объявление массива

Для одномерных массивов синтаксис объявления следующий:

```
тип_элементов идентификатор[размер];
```

Пример: double A[5];

создается 5 элементов массива с именем A типа double:
A[0], A[1], A[2], A[3], A[4]
Инициализация массива
1. Объявлен массив:
int arr[5];
2. Массив инициализирован, если элементам массива присвоены значения:

```
arr[0]=10;          arr[3]=40;
arr[1]=20;          arr[4]=50;
arr[2]=30;
```

Инициализация массива при объявлении

При объявлении массива вы можете сразу указать первоначальные значения элементов массива

Инициализация массива **values**:

```
int values[5] = {100, 200, 300, 400, 500};
```

Доступ к элементам массива

Для доступа к n-ому элементу необходимо написать **имя_массива[номер элемента]** и работать с ним как с обычной переменной.

Примеры доступа к элементам массива:

```
numbers[200]=201; //записать в 200 элемент число 201
test[1]=0,5; // в 1 элемент записать 0,5
test[9]=10; // в 9 элемент записать число 10
```

Сортировка массива

Сортировкой или упорядочением массива называется расположение его элементов по возрастанию (или убыванию).

Если не все элементы различны, то надо говорить о неубывающем (или невозрастающем) порядке.

В этой сложной теме известно много различных алгоритмов. Критерии оценки эффективности этих алгоритмов могут включать следующие параметры:

- количество шагов алгоритма, необходимых для упорядочения;
- количество сравнений элементов;
- количество перестановок, выполняемых при сортировке.

Известные схемы сортировки:

- Метод «пузырька»
- Сортировка вставками
- Сортировка посредством выбора

Мы рассмотрим одну из простейших схем сортировки – метод «пузырька».

Представьте, что массив (таблица) расположен вертикально. Элементы с большим значением всплывают вверх наподобие больших пузырьков. При первом проходе вдоль массива, берется первый элемент и поочередно сравнивается с последующими. При этом, если встречается элемент с меньшим значением, то

они меняются местами; при встрече с элементом с большим значением, последний становится "эталоном" для сравнения, и все следующие сравниваются с ним.

В результате наибольший элемент оказывается в самом верху массива.

Во время второго прохода вдоль массива находится второй по величине элемент, который помещается под элементом, найденным при первом проходе, т.е. на вторую сверху позицию, и т.д.

Перестановка элементов производится всегда через дополнительную промежуточную переменную – **temp**.

Главное, что при втором и последующих проходах, нет необходимости рассматривать ранее "всплывшие" элементы, т.к. они заведомо больше оставшихся. Другими словами, во время **j-го** прохода не проверяются элементы, стоящие на позициях выше **j**.

Алгоритм сортировки методом «пузырька»

Если два соседних элемента расположены не по порядку, то меняем их местами.

Так повторяем до тех пор, пока в очередном проходе не сделаем ни одного обмена, т.е. массив будет упорядоченным.

Для решения задачи используется два цикла:

- Первый цикл – внешний - начинается с 0 элемента и до предпоследнего
- Второй цикл – внутренний - начинается со следующего элемента до последнего

Реализация алгоритма сортировки

Задание: Дан массив из 10 элементов, индексы 0 – 9.

Первый цикл 0 - 8 for (i=0;i<9;i++)

Второй цикл 1 - 9 for (j=i+1;j<10;j++)

Во внутреннем цикле сравниваются значения и, если, условие сравнения выполняется, тогда элементы меняются местами.

- Массив сортируется по возрастанию.
- В результате полного прохода внутреннего цикла, первым встанет элемент с самым большим значением.
- Если поменять знак с **>** на **<**, то массив сортируется по убыванию.

```
//сортировка
for(i=0;i<9;i++)
{
    for(j=i+1;j<10;j++)
    {
        if (a[j]>a[i])
        {
            temp=a[j];
            a[j]=a[i];
            a[i]=temp;
        }
    }
}
```

Задания

Задание 1.

Создайте проект SampleArray1 - Пример одномерного массива

1. Написать программу, в которой с клавиатуры вводят набор из 5 целых чисел в одномерный массив.

2. Перед вводом каждого элемента должна выводиться подсказка с номером элемента.

3. Вывести массив, посчитать и вывести количество ненулевых элементов.

Решение проекта. Программа "Пример одномерного массива"

1. Листинг программы

Первая часть проекта: ввод данных с клавиатуры, данные записываются в массив.

Начало программы (красной точкой отмечены места, где необходимо сделать редакцию кода).

```
/*Программа Массив -ввод,вывод, количество ненулевых*/
#include <iostream>
#include <conio.h>//библиотека - ожидание нажатия клавиши
using namespace std;

int main()
{
//-----------------------------
    int a[5];//Объявление массива
    int i;//счетчик цикла
    int z;//значение элемента массива
```

Обработка ввода данных (продолжение)

```
    setlocale(LC_CTYPE,"");//использование кирилицы
    cout<<"\n\tПример одномерного массива\n";
    cout<<"\n\tВвод массива целых чисел\n";
    cout<<"\n\tПосле ввода каждого числа нажмите <Enter>\n";
//Ввод данных массива
    for (i=0;i<5;i++)
    {
        cout<<"a["<<i<<"] --> ";
        cin>>z;
        a[i]=z;
        cout<<"\n";
    }
```

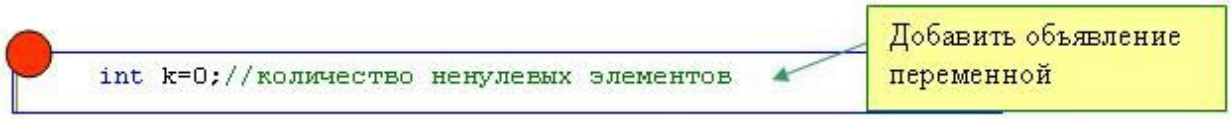
Окончание программы

```
//-------------------------------------------------------------
cout <<"\n\nНажмите клавишу для продолжения...";
_getch(); // Ожидание нажатия клавиши
return 0;
}
```

Скомпилируйте проект и протестируйте программу.

Отредактируйте программный код

1. Объявим переменную - для расчета количества ненулевых элементов



2. Вывод значений и результатов на экран

```
//Вывод массива
cout<< "Массив -> ";
    for (i=0;i<5;i++)
    {
        cout<< a[i]<<" " ;
    //Расчет количества ненулевых элементов
        if (a[i]!=0)k++;
    }
//Вывод результата
    cout<<"\n";
    cout<<"Количество ненулевых элементов - "<<k<<"\n";
```

Скомпилируйте проект еще раз и протестируйте программу

2. Результат работы программы

```
e:\C++-KURSUUS\Projects\SampleArray1\Debug\SampleArray1.exe

        Пример одномерного массива

        Ввод массива целых чисел

        После ввода каждого числа нажмите <Enter>
a[0] --> 5

a[1] --> 0

a[2] --> 0

a[3] --> -2

a[4] --> 3

Массив -> 5 0 0 -2 3
Количество ненулевых элементов - 3


Нажмите клавишу для продолжения...
```

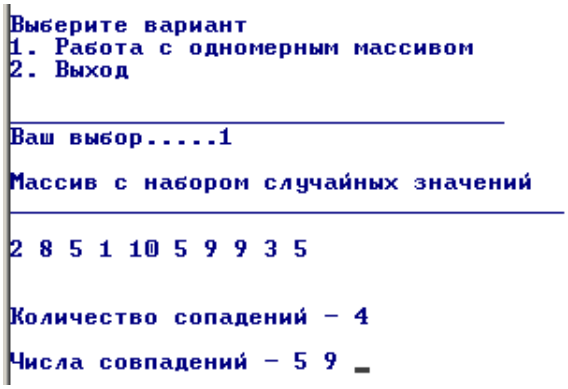
Задание 2. Создать одномерный массив для 10 значений.

- Заполнить случайными числами от 1 – до 10.
- Вывести массив на экран.
- Создать меню для управления проектом.

Посчитать:

- Посчитать количество совпадений цифр и вывести отчет.

Для решения задачи создать массив для хранения выходных значений.
Образец экрана:



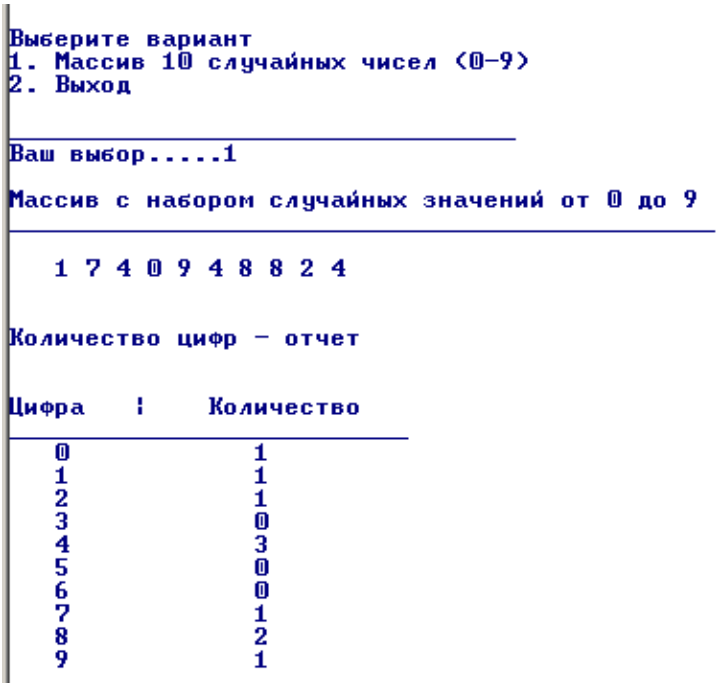
Задание 3. Создать одномерный массив для 10 значений.

- Заполнить матрицу случайными числами от 0 до 9.
- Вывести массив на экран.
- Создать меню для управления проектом

Посчитать:

- Количество каждой цифры и вывести отчет в виде таблицы

Для решения задачи создать массив для хранения выходных значений.
Образец экрана:



Практическое занятие № 15. Основы программирования микроконтроллеров.

Цель занятия:

Изучение назначения и особенностей архитектуры однокристальных микроконтроллеров; ознакомление с архитектурой и программной моделью AVR-микроконтроллеров; изучение этапов разработки ПО для встраиваемых микропроцессоров; приобретение навыков работы в среде AVR Studio.

Общие сведения

Программирование микроконтроллеров. Процесс разработки прикладного ПО устройств на основе однокристальных микроконтроллеров включает следующие этапы:

- разработки алгоритма и структуры программы;
- написания исходного текста программы;
- получения выполняемой программы;
- тестирования и отладки программы;
- получения загрузочной программы.

На этапе разработки алгоритма и структуры программы выбирается метод решения задачи и разрабатывается алгоритм его реализации. Алгоритм – это набор правил или описание последовательности операций для решения определённой задачи или достижения некоторой цели. Графическим изображением алгоритма является *схема алгоритма* (flowchart), выполняемая в соответствии с ГОСТ 19.701–90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

На этапе написания исходного текста программы разработанный алгоритм записывается в виде программы на исходном языке (ассемблере или языке высокого уровня).

Языком ассемблера называется язык программирования, в котором каждой команде процессора или совокупности команд процессора соответствует сокращённая символическая запись (мнемоника). Использование символического обозначения команд, а также адресов регистров и ячеек памяти, переменных, констант и других элементов программы существенно облегчает процесс составления программ по сравнению с программированием на уровне машинных кодов. Символические обозначения элементов обычно отражают их содержательный смысл. Язык ассемблера обеспечивает возможность доступа ко всем ресурсам программируемого микропроцессора (микроконтроллера) и позволяет создавать программы, эффективные как по быстродействию, так и по объёму занимаемой памяти. В этой связи программирование на языке ассемблера предполагает знание архитектуры и свойств микропроцессора, т. е. всего того, что входит в понятие «программная модель». Языки ассемблера различаются для разных типов микропроцессоров, т. е. являются машинно-ориентированными. В ряде ассемблеров допускается оформление повторяющейся последовательно-

сти команд как одной макрокоманды (макроса), такие ассемблеры называют *макроассемблерами*.

Языки высокого уровня (С, Паскаль, Бейсик и др.), как и ассемблер, обеспечивают доступ ко всем ресурсам микроконтроллера, но вместе с тем дают возможность создавать хорошо структурированные программы, снимают с программиста заботу о распределении памяти и содержат большой набор библиотечных функций для выполнения стандартных операций.

На этапе получения выполняемой программы исходный текст программы с помощью специальных средств (трансляторов, компиляторов, компоновщиков и др.) преобразуется в исполняемый код. *Транслятором* (translator) называют программу, служащую для перевода (трансляции) программ на языке ассемблера в машинный код, «понимаемый» процессором. *Компилятор* (compiler) представляет собой программу, преобразующую в эквивалентный машинный код текст программы на языке высокого уровня. Результатом работы транслятора или компилятора может быть как выполняемый загрузочный модуль, так и объектный модуль (программа, команды, переменные и константы которой не «привязаны» к конкретным адресам ячеек памяти). Для построения выполняемой программы из объектных модулей применяется *компоновщик* (редактор связей, linker). В процессе получения выполняемой программы из исходного текста программы устраняются синтаксические ошибки, состоящие в нарушении правил синтаксиса используемого языка программирования.

На этапе тестирования и отладки программы производится поиск, локализация и устранение в ней логических ошибок. *Тестирование* служит для обнаружения в программе ошибок и выполняется с использованием некоторого набора тестовых данных. Тестовые данные должны обеспечивать проверку всех ветвей алгоритма. При тестировании программы могут подвергаться проверке также некоторые показатели системы, связанные с программой (например, объём кодов и данных). После тестирования программа должна быть подвергнута *отладке* (debug), задачей которой является локализация ошибки, т. е. нахождение места в программе, вызывающего ошибку.

Тестирование и отладка программы могут привести (и, как правило, приводят) к необходимости возврата к ранним этапам процесса разработки программы для устранения ошибок в постановке задачи, разработке алгоритма, написании исходного текста и т. д. Таким образом, процесс разработки программы, как и весь процесс проектирования, является итерационным.

На этапе получения загрузочной программы производится «освобождение» программы от лишних фрагментов, использовавшихся для тестирования и отладки. Эти фрагменты увеличивают объём программы и не нужны при нормальном функционировании микропроцессорной системы. Далее полученная загрузочная программа заносится в память микроконтроллера.

По завершении процесса разработки производится *документирование*, т. е. составление комплекта документов, необходимых для эксплуатации и сопровождения программы. Сопровождение программы (program maintenance) – это процесс внесения изменений, исправления оставшихся ошибок и проведения

консультаций по программе, находящейся в эксплуатации. Виды программных документов регламентированы ГОСТ 19.101–77 «Единая система программной документации. Виды программ и программных документов».

Разработка ПО для встраиваемых микропроцессоров производится на персональном компьютере с использованием специальных программных и аппаратных средств. Такой способ создания ПО носит название *кросс-разработки*. Совокупность аппаратных и программных средств, применяемых для разработки и отладки ПО, объединяют общим наименованием *средства поддержки разработки*. В настоящем лабораторном практикуме процесс разработки ПО изучается на примере языка ассемблера **AVR**-микроконтроллеров. Создание исходного текста программы, трансляция и отладка выполняются в интегрированной среде разработки (Integrated Development Environment – IDE) **AVR Studio**.

Работа в среде AVR Studio. В состав среды **AVR Studio** входит редактор исходных текстов, транслятор с языка ассемблера, отладчик и симулятор.

Транслятор работает с исходными программами на языке ассемблера, содержащими метки, директивы, команды и комментарии. Метка представляет собой символическое обозначение адреса (последовательность символов, заканчивающаяся двоеточием). Метки используются для указания места в программе, в которое передается управление при переходах, а также для задания имён переменных. Директивы являются инструкциями для транслятора и не заносятся в исполняемый код программы (список директив приведён в приложении 3). Директивы могут иметь один или несколько параметров. Команды записываются в программе в виде мнемонического обозначения выполняемой операции и могут иметь один или несколько *операндов*, т. е. аргументов, с которыми они вызываются. Транслятор позволяет указывать операнды в различных системах счисления: десятичной (по умолчанию, например, **15**, **154**), шестнадцатеричной (префикс **0x** или **$**, например, **0x0f**, **$0f**, **0x9a**, **$9a**), восьмеричной (префикс – нуль, например, **017**, **0232**) и двоичной (префикс **0b**, например, **0b00001111**, **0b10011010**). Строка программы должна быть не длиннее 120 символов и может иметь одну из четырёх форм:

[метка:] .директива [параметры] [;Комментарий]
[метка:] команда [операнды] [;Комментарий]
[;Комментарий]
[Пустая строка]

Позиции в квадратных скобках необязательны. Текст после точки с запятой и до конца строки является комментарием и транслятором игнорируется. Включение в текст программы комментариев является признаком хорошего стиля программирования и облегчает её сопровождение. Кроме того, улучшению читаемости также способствует форматирование текста программы. При программировании на ассемблере выполнение этих правил особенно важно, так как программы на языке ассемблера неудобочитаемы.

Указать тип микроконтроллера, для которого транслируется программа, позволяет директива **.device**, например:

.device ATmega8535; программа для микроконтроллера ATmega8535

При наличии в программе команд, не поддерживаемых указанным в директиве микроконтроллером, транслятор выдаёт соответствующее предупреждение.

Входным для транслятора является файл **<имя_файла>.asm** с текстом программы на языке ассемблера. Транслятор создаёт четыре новых файла: файл листинга **(<имя_файла>.lst**), объектный файл (**<имя_файла>.obj**), файл-прошивку памяти программ (**<имя_файла>.hex**) и файл-прошивку энергонезависимой памяти данных (**<имя_файла>.eep**).

Файл листинга – это отчёт транслятора о своей работе. На рис. 6 приведена часть листинга трансляции программы, в которой числа 2, 5 и 19 заносятся соответственно в регистры **R17**, **R18** и **R19**; вычисляются произведение и сумма содержимого регистров **R17** и **R18**; из суммы содержимого регистров **R17** и **R18** вычитается содержимое регистра **R19**. Листинг содержит исходный текст транслируемой программы, каждой команде которой поставлены в соответствие машинные коды (правый столбец чисел) и адреса ячеек памяти программ, в которых они будут размещены (левый столбец чисел). Машинные коды и адреса приводятся в шестнадцатеричной системе счисления. Например, строка листинга с командой **ADD** содержит следующую информацию: **0f12** – машинный код команды; **000004** – адрес размещения данной команды в памяти программ.

```
000000 e012 ldi R17, 2 ; загрузка числа 2 в регистр R17
000001 e025 ldi R18, 5 ; загрузка числа 5 в регистр R18
000002 e133 ldi R19, 19 ; загрузка числа 13 в регистр R19
000003 9f12 mul R17, R18 ; умножение R17 на R18, результат в R1:R0
000004 0f12 add R17, R18 ; сложение R17 и R18, результат в R17
000005 1b31 sub R19, R17 ; вычитание R17 из R19, результат в R19
000006 cfff met: rjmp met ; бесконечный цикл (для отладки)
```

Пример листинга трансляции

Объектный файл имеет специальный формат и используется для отладки программы с помощью симулятора-отладчика среды **AVR Studio**. Файл прошивки памяти программ служит для занесения отлаженной программы в память программ микроконтроллера. Файл прошивки EEPROM-памяти данных предназначен для загрузки информации в энергонезависимую память данных. Операции загрузки памяти программ и энергонезависимой памяти данных выполняются с помощью специальных аппаратных средств (программаторов).

Задание

Составить программу вычисления произведения и суммы двух чисел **A** и **B**, находящихся в РОН. Из суммы **A** и **B** вычесть число **C**. За основу взять программу, приведённую на рис. 6. Числа изменить в соответствии с заданным вариантом (табл. 1). В начало программы поместить директиву **.device** для микроконтроллера **ATmega8535** (здесь и далее предполагается использование микроконтроллера **ATmega8535**). В комментариях указать фамилию и номер группы.

№ варианта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86
C	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
№ варианта	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
B	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71
C	65	66	67	68	69	79	71	72	73	74	75	76	77	78	79

Выполнить разработку программы в среде **AVR Studio**[*], проделав следующие операции.

1. Создать новый проект, воспользовавшись командой New Project меню Project. В появившемся диалоговом окне в поле **ProjectName** ввести имя создаваемого проекта без расширения (информация о проекте сохраняется в файле с расширением **.aps**). В поле **Location** указать место размещения файлов проекта на диске (путь); в поле **Project type** выбрать пункт **AVRAssembler** (исходные тексты программ разрабатываются на языке ассемблера). Для создания файла исходной программы установить флажок **CreateinitialFile**. Задать имя файла программы, отличающееся от имени проекта, можно в поле **InitialFile** (расширение **.asm** файлов программ на ассемблере устанавливается автоматически). Создание каталога для хранения файлов проекта обеспечивается установкой флажка **CreateFolder**. Рекомендуется каждый проект размещать в отдельном каталоге. Нажать кнопку «Next».

2. В группе **Selectdebugplatformanddevice** в поле **DebugPlatform** указать способ отладки создаваемого проекта – **AVRSimulator** (симулятор-отладчик), в поле **Device** выбрать тип микроконтроллера, для которого создаётся программа (**ATmega8535**). Нажать кнопку «Finish». На экране появится дерево иерархии проекта (окно **Workspace**, закладка **Project**) и окно редактора исходных текстов программ.

Если при создании проекта не был установлен флажок**Createinitialfile**, создать файл исходного текста программы можно командой New File меню File.

3. Ввести и отредактировать текст программы. Сохранить файл, воспользовавшись командой Save меню File. Если файл с исходным текстом программы уже существует, его можно включить в проект командой Add existing File меню Project или контекстного меню окна иерархии проекта при выделенной группе **Assembler**.

4. Провести трансляцию созданной программы, воспользовавшись командой Build and run меню Project(или сочетанием клавиш **Ctrl**+**F7** на клавиатуре). Перед трансляцией убедиться, что установлен флажок **Listfile** в диалоговом окне **AVRAssembler**, вызов которого осуществляется командой AVR

Assembler Setup меню Project (это необходимо для создания листинга трансляции). По окончании трансляции в окне **Output** на закладке **Build** появится информация о результатах трансляции. Открыть файл листинга можно из дерева иерархии проекта (окно **Workspace**, закладка **Project**). После этого с помощью команды Save Project меню Project сохранить изменения в файле проекта.

Дополнительное задание: ознакомиться с техническим описанием микроконтроллера **ATmega8535** «**ATmega8535 DataSheet**» фирмы **Atmel**.

Примечания: 1. Создаваемые файлы следует размещать ТОЛЬКО в каталогах, специально выделенных для этого администратором дисплейного класса.

2. Некоторые команды программы **AVR Studio** доступны из панели инструментов.

3. Распечатку программ и листингов трансляции обеспечивает команда Print меню File, предварительный просмотр – команда Print Preview меню File, настройку параметров печати – команда Print Setup меню File.

Практическое занятие № 16. Средства отладки программного обеспечения.

Цель занятия:

Ознакомление с аппаратными и программными средствами отладки ПО; изучение команд отладчика среды AVR Studio; приобретение навыков отладки программ под управлением отладчика.

Общие сведения

Особенность отладки ПО устройств на базе встраиваемых МП (в том числе однокристальных микроконтроллеров) состоит в отсутствии в их составе развитых средств для реализации пользовательского интерфейса и ограниченных возможностях системного ПО. В то же время именно для встраиваемых микропроцессорных систем этап отладки является чрезвычайно ответственным, так как для них характерна тесная взаимосвязь работы ПО и аппаратных средств.

Взаимодействие микропроцессора (микроконтроллера) с датчиками и исполнительными устройствами происходит путём передачи данных через регистры периферийных устройств (регистры ввода-вывода). Отдельные разряды таких регистров задают режимы работы периферийных устройств, имеют смысл готовности к обмену, завершения передачи данных и т. п. Состояние этих разрядов может устанавливаться как программно, так и аппаратно. При отладке ПО часто приходится переходить на уровень межрегистровых передач и проверять правильность установки отдельных разрядов. Кроме того, на этапе отладки может производиться оптимизация алгоритма, нахождение критических участков кода и проверка надёжности разработанного ПО.

Для решения указанных задач применяются аппаратные и программные средства отладки ПО (рис. 16.1).

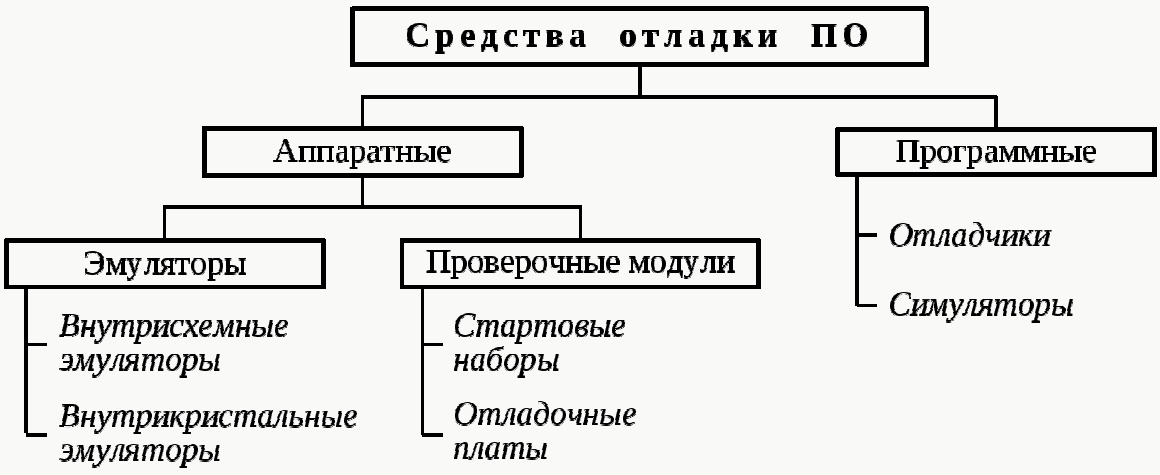


Рисунок 16.1 – Классификация средств отладки программного обеспечения

К **аппаратным средствам отладки** относятся аппаратные эмуляторы и проверочные модули.

Аппаратные эмуляторы предназначены для отладки программного и аппаратного обеспечения микропроцессорных систем в режиме реального времени. Они работают под управлением «ведущего» компьютера, оснащённого спе-

циальным ПО – программами-отладчиками (см. ниже). Основными видами аппаратных эмуляторов являются:

- внутрисхемные эмуляторы или эмуляторы-приставки, замещающие микропроцессор в отлаживаемой системе;
- внутрикристальные эмуляторы, представляющие собой одно из внутренних устройств микропроцессора.

Внутрисхемный эмулятор (In-Circuit Emulator, ICE) – это устройство, содержащее аппаратный имитатор процессора и схему управления имитатором. При отладке с помощью эмулятора микропроцессор извлекается из отлаживаемой системы, на его место подключается контактная колодка, количество и назначение контактов которой идентично выводам замещаемого микропроцессора (рис. 16.2). С помощью гибкого кабеля контактная колодка соединятся с эмулятором. Управление процессом отладки осуществляется с персонального компьютера. Эмуляторам-приставкам присущи следующие недостатки: высокая стоимость, недостаточная надёжность, высокое энергопотребление, влияние на электрические характеристики цепей, к которым подключается эмулятор.

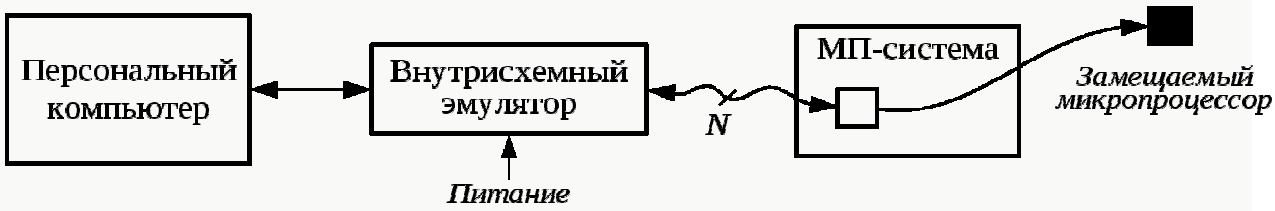


Рисунок 16.2 – Отладка с помощью внутрисхемного эмулятора

Внутрикристальные эмуляторы (On-Chip Emulator) позволяют проводить отладку программ без извлечения микропроцессора из системы. При этом осуществляется непосредственный контроль за выполнением программы, так как средства внутрикристальной отладки обеспечивают прямой доступ к регистрам, памяти и периферии микропроцессора. Наиболее распространённым средством внутрикристальной отладки является последовательный интерфейс **IEEE 1149.1**, известный как **JTAG** (Joint Test Action Group – Объединённая рабочая группа по автоматизации тестирования). Последовательный отладочный порт **JTAG** микропроцессора с помощью специального устройства сопряжения подключается к компьютеру, чем обеспечивается доступ к отладочным средствам процессора (рис. 16.3). Такой способ отладки также называют *сканирующей эмуляцией*. Достоинствами этого способа являются возможность выполнения различных действий на процессоре без его изъятия из системы, использование малого числа выводов процессора и поддержка его максимальной производительности без изменения электрических характеристик системы.

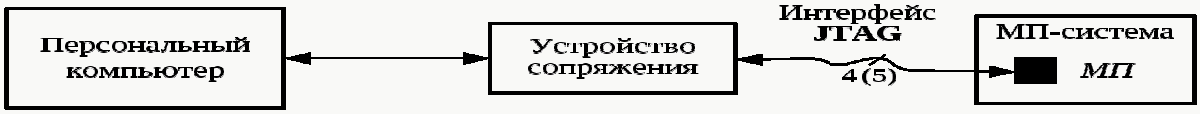


Рисунок 16.3 – Отладка с помощью внутрикристального эмулятора

Проверочные модули предназначены для быстрой отладки программного обеспечения в реальном масштабе времени. Проверочные модули бывают двух видов: стартовые наборы и отладочные платы.

Стартовые наборы (Starter Kit) предназначены для обучения работе с конкретным микропроцессором. Стартовый набор позволяет изучить характеристики микропроцессора, отладить не слишком сложные программы, выполнить несложное макетирование, проверить возможность применения микропроцессора для решения конкретной задачи. В состав стартового набора входят плата, ПО и комплект документации. На плате устанавливаются микропроцессор, устройство загрузки программ, последовательные или параллельные порты, разъёмы для связи с внешними устройствами и другие элементы. Плата подключается к компьютеру через параллельный или последовательный порт. Стартовые наборы удобны на начальном этапе работы с микропроцессором.

Отладочные платы (Evaluation Board) предназначены для проверки разработанного алгоритма в реальных условиях. Они позволяют проводить отладку и оптимизацию алгоритма с использованием установленной на плате периферии, а также изготовить на базе платы законченное устройство. Обычно на плате размещаются микропроцессор, схемы синхронизации, интерфейсы расширения памяти и периферии, схема электропитания и др. Плата подключается к компьютеру через параллельный или последовательный порт или непосредственно устанавливается в слот **PCI**.

Основными **программными средствами отладки** являются симуляторы и отладчики.

Симуляторы (simulator) или *симуляторы системы команд* представляют собой программы, имитирующие работу того или иного процессора на уровне его команд. Симуляторы обычно используются для проверки программы или её отдельных частей перед испытанием на аппаратных средствах.

Отладчики (debugger) представляют собой программы, предназначенные для анализа работы созданного программного обеспечения. Можно указать следующие возможности отладчиков.

1. *Пошаговое выполнение*. Программа выполняется последовательно, команда за командой, с возвратом управления отладчику после каждого шага.
2. *Прогон*. Выполнение программы начинается с указанной команды и осуществляется без остановки до конца программы.
3. *Прогон с контрольными точками*. При выполнении программы происходит останов и передача управления отладчику после выполнения команд с адресами, указанными в списке контрольных точек.
4. *Просмотр и изменение содержимого регистров и ячеек памяти*. Пользователь имеет возможность выводить на экран и изменять (модифицировать) содержимое регистров и ячеек памяти.

Отладчики ПО встраиваемых микропроцессоров обычно используются совместно с внутрисхемными или внутрикристальными эмуляторами, а также могут работать в режиме симулятора. Некоторые отладчики позволяют также выполнять *профилирование*, т. е. определять действительное время выполнения

некоторого участка программы. Иногда функцию профилирования выполняет специальная программа – *профилировщик* (profiler).

Средства отладки ПО AVR-микроконтроллеров. Аппаратные средства отладки программного обеспечения **AVR**-микроконтроллеров представлены внутрисхемным эмулятором **ICE50**, внутрикристальным эмулятором **JTAG ICE**, а также стартовым набором **STK500**.

К программным средствам отладки ПО **AVR**-микроконтроллеров относятся отладчик и симулятор, входящие в состав среды **AVR Studio**. Отладчик среды **AVR Studio** позволяет проводить отладку программ как в исходных кодах (например, ассемблера), так и в кодах дизассемблера (оттранслированной или скомпилированной программы, записанной с помощью мнемоник ассемблера). Вызов окна с кодом дизассемблера производится командой Disassembler меню View или командой Goto Disassembly контекстного меню редактора исходного текста. Обратное переключение в окно исходного текста осуществляется командой Goto Source контекстного меню окна **Disassembler**.

Отладчик среды **AVRStudio** может использоваться с внутрисхемным эмулятором **ICE50**, внутрикристальным эмулятором **JTAGICE**, отладочной платой **STK500** или симулятором. Указание способа отладки производится при создании проекта. Симулятор среды **AVR Studio** предназначен для предварительной отладки программ без применения аппаратных средств. В дальнейшем в настоящем лабораторном практикуме для отладки создаваемых программ предполагается применение отладчика среды **AVR Studio** в режиме симулятора.

Отладка ПО в среде AVR Studio. Команды отладчика в программе **AVR Studio** находятся в меню Debug.

Переход в режим отладчика в среде**AVR Studio** осуществляется автоматически при использовании для трансляции программы команды Build and Run или командой Start Debugging меню Debug при использовании для трансляции команды Build. Выход из режима отладчика производится командой Stop Debugging меню Debug.

Пошаговое выполнение программы задаётся командами Step Into, Step Over меню Debug. Команда Step Into позволяет выполнить одну команду программы (в том числе команду вызова подпрограммы). Для завершения выполнения подпрограммы может использоваться команда Step Out. Команда Step Over также выполняет одну команду программы, но если это команда вызова подпрограммы, последняя полностью выполняется за один шаг. Следующая выполняемая команда (команда, адрес которой содержится в программном счётчике) обозначается символом ⇨ в окне исходного текста программы. Сброс выполнения программы осуществляется с помощью команды Reset.

Прогон (запуск или продолжение выполнения) программы осуществляется командой Run. Для остановки выполнения программы служит команда Break.

Контрольные точки представляют собой специальные маркеры для программы-отладчика и могут быть трёх типов: точки останова, точки трассировки и точки наблюдения.

Точки останова задаются командой Toggle Breakpoint меню Debug или контекстного меню редактора исходного текста программы. Точка останова обозначается в редакторе исходного текста символом слева от помечаемой строки. Просмотреть заданные точки останова можно на закладке **Breakpoints** окна **Output**; там же точки останова могут быть запрещены (путём сброса флажка напротив точки останова) и разрешены (путём установки флажка). При достижении точки останова во время прогона программы её выполнение приостанавливается. Повторный вызов команды установки точки останова на той же строке программы приводит к удалению точки останова. Удалить все заданные точки останова позволяет команда Remove Breakpoints меню Debug или команда Remove all Breakpoints контекстного меню закладки **Breakpoints** окна **Output**. Параметры точки останова задаются в диалоговом окне **BreakpointCondition**, вызов которого осуществляется командой Breakpoints Properties контекстного меню редактора исходного текста программы. Установка флажка **Iterations** позволяет задать количество итераций (повторных выполнений) команды до останова прогона программы. При установке флажка **Watchpoint** по достижению точки останова производится только обновление значений регистров и ячеек памяти в окнах просмотра. Флажки **Iterations** и **Watchpoint** не должны устанавливаться одновременно. Установка флажка **Showmessage** обеспечивает отображение сообщений о достижении точки останова на закладке **Breakpoints** окна **Output**. Вызов диалогового окна задания свойств и удаление точки останова могут быть произведены из контекстного меню закладки **Breakpoints** окна **Output**.

Точки трассировки предназначены для контроля выполнения программы в режиме реального времени. Трассировка позволяет отслеживать так называемую трассу программы – изменение содержимого регистров и ячеек памяти при выполнении определённых команд (команд, по адресам которых заданы точки трассировки). В среде **AVR Studio** функция трассировки может использоваться только при отладке программы с применением внутрисхемного эмулятора; при работе в режиме симулятора функция трассировки недоступна.

Точки наблюдения задаются командой Add to Watch контекстного меню редактора исходного текста программы. Точки наблюдения представляют собой символические имена регистров или ячеек памяти, содержимое которых необходимо отслеживать. При выполнении команды Add Watch на экране появляется окно **Watches**, разделённое на четыре столбца: **Name** (символическое имя точки наблюдения), **Value** (значение), **Type** (тип), **Location** (местонахождение). Новая точка наблюдения может быть также задана в выделенной ячейке столбца **Name** окна **Watches** или командой Quickwatch в окне редактора исходного текста программы (при этом курсор должен находиться на имени регистра или ячейки памяти). Значения, отображаемые в столбце **Value**, обновляются при изменении содержимого соответствующего регистра или ячейки памяти. Удалить заданные точки наблюдения можно из окна **Watches**.

Отладчик среды **AVR Studio** также обеспечивает следующие функции: выполнение до курсора (команда Run to Cursor меню Debug) и последователь-

ное выполнение команд с паузами между ними (команда Auto Step меню Debug).

Для удобства использования в процессе отладки ряд команд отладчика доступен с клавиатуры (табл. 16.1).

Таблица 16.1

Команда отладчика	Клавиша	Команда отладчика	Клавиша
Run	F5	Step Into	F11
Break	Ctrl+F5	Step Out	Shift+F11
Reset	Shift+F5	Step Over	F10
Run to Cursor	Ctrl+F10	Toggle Breakpoint	F9

Для просмотра и изменения содержимого регистрови ячеек памяти служат команды Registers, Memory, Memory 1, Memory 2, Memory 3 меню View.

По команде Registers на экране отображается окно **Registers**, в котором приводятся шестнадцатеричные представления содержимого РОН. Изменение (модификация) содержимого регистров производится путём двойного щелчка мышью. Наблюдение за содержимым РОН может быть также произведено с помощью дерева устройств микроконтроллера, находящегося на закладке **I/O** окна **Workspace**. Для этого необходимо раскрыть объекты **Register 0-15** и **Register 16-31** щелчком мыши по знаку «+».

Команды Memory, Memory 1, Memory 2, Memory 3 обеспечивают вызов окон**Memory**, служащих для отображения содержимого ячеек оперативной и энергонезависимой памяти данных, памяти программ, регистров ввода-вывода и РОН. Выбор типа памяти, отображаемой в окне **Memory**, производится с помощью списка, расположенного в панели управления окна (**Data** – оперативная память данных, **Eeprom** – энергонезависимая память данных, **I/O** – регистры ввода-вывода, **Program** – память программ, **Register** – РОН).

Для наблюдения за состоянием процессора необходимо раскрыть объект **Processor** закладки **I/O** окна **Workspace**. При этом будет отображена следующая информация: содержимое программного счётчика (**ProgramCounter**); содержимое указателя стека (**StackPointer**), количество тактов, прошедших с начала выполнения (**CycleCounter**); содержимое 16-разрядных регистров-указателей **X**, **Y** и **Z**; тактовая частота (**Frequency**); затраченное на выполнение время (**StopWatch**).

Для контроля содержимого регистров ввода-вывода необходимо раскрыть объект **I/O *** закладки **I/O** окна **Workspace**, где ***** – тип микроконтроллера. Регистры ввода-вывода, входящие в объект **I/O**, сгруппированы по типам периферийных устройств.

Модифицированные значения содержимого регистров и ячеек памяти действуют только во время текущего сеанса отладки, в исходный текст программы изменения не заносятся.

Задание

Провести отладку созданной в практической работе № 15 программы с помощью симулятора-отладчика среды **AVR Studio**, проделав следующие операции.

1. Выполнить программу в пошаговом режиме, отслеживая изменение содержимого используемых в программе регистров. Обратить внимание на изменение содержимого программного счётчика. Сравнить содержимое программного счётчика при выполнении команд с их адресами в памяти программ, приведёнными в листинге трансляции и окне памяти программ.

2. Выполнить прогон программы. Проверить правильность результата работы программы.

3. Задать точку останова на команде загрузки в РОН числа **B**. Включить режим отображения сообщений о достижении точки останова. Выполнить прогон программы с контрольными точками. Задать точку останова на команде умножения. Выполнить прогон программы с контрольными точками. Удалить заданные точки останова.

4. Задать точки наблюдения в используемых РОН. Выполнить программу в пошаговом режиме, отслеживая изменение их содержимого.

Литература

1. Компас-Электрик. Руководство пользователя.
2. Моделирование схем в программе Multisim- Интернет - журнал [Электронный ресурс]
3. Руководство по Multisim. Электронный ресурс.
4. М.И. Глотова, О.В. Приходько. Основы работы в среде MathCAD.