|  |
| --- |
| титул |
| **115172, Москва, ул. Б.Каменщики, д. 7;тел., факс: (495) 134 1234; e-mail: spo-54@edu.mos.ru** |

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ**

к лабораторной работе №4:

**Программирование. Переменные. Работа с различными типами значений**

по учебной дисциплине

**МДК 01.01 Разработка программных модулей**

Специальность

**09.02.07 «Информационные системы и программирование»**

**Оглавление**

[Введение 3](#_Toc127967049)

[1. Программирование - терминология, модули, ввод данных 5](#_Toc127967050)

[2. Программирование — переменные 14](#_Toc127967051)

[3. Программирование — работа с типами значений (строка, неявные преобразования) 22](#_Toc127967052)

[4. Программирование — работа с типами значений (дата) 26](#_Toc127967053)

[5. Программирование — работа с типами значений (булево) 29](#_Toc127967054)

[Индивидуальное задание 32](#_Toc127967055)

[Контрольные вопросы 33](#_Toc127967056)

# **Введение**

**Цель работы**: знакомство с конфигуратором платформы 1С: Предприятие для наиболее эффективной работы с информационной базой.

**Время выполнения**: 2 часа

**Задание**:

1. Законспектировать теоретические сведения, при наличии.

2. Выполнить лабораторную работу.

3. Выполнить индивидуальное задания, при наличии.

4. Защитить работу, ответив на контрольные вопросы устно.

В первую очередь введем несколько основных определений.

*При работе с главами цикла «Программирование» после прохождения каждого примера рекомендуется закомментировать код, чтобы его выполнение не нагружало работу персонального компьютера.*

Для того чтобы задать программную логику автоматического расчета суммы по цене и количеству, нужно нашей программе сообщить о том, как производить расчет, то есть написать, что цену в конкретной строке требуется умножить на количество в той же конкретной строке.

Чтобы объяснить программе, как это сделать, нам нужно дать объяснения на ее языке — на языке программирования платформы 1С.

"Поэтому сначала мы изучим базис по программированию: как писать программный код, из чего он состоит, какие есть ключевые термины.

Начнем с терминологии. Первое, это то. где пишется программный код.

**Модуль** — это область, в которой описывается программная логика нашего прикладного решения. В модулях прописываются все переменные, процедуры. выражения и так далее.

**Переменная** — выделенная область оперативной памяти компьютера, в которой хранится значение определённого типа данных: в переменную помещается какое-то конкретное значение.

Переменные могут иметь типы данных. Для компьютера вся информация представляется в виде единиц и нулей, а для себя, как для людей, мы фиксируем информацию в виде текстов, чисел, дат и так далее.

**Типы значений** определяют, что именно представляют собой данные. которые хранятся в переменной, в реквизите и т. д. Примеры типов данных: Булево, Строка, Число и др.

**Выражение** — это совокупность операндов и операторов. Это вычислительные действия в программном коде: например, сравнение двух значений, подсчет суммы по цене или количеству, расчет текущей даты и т.д.

**Операторы** — определённый набор символов, позволяющий описывать различные действия в алгоритмах программы: например, сложение, умножение, присваивание п др.

**Операнды** — элементы, которые описываются в выражении: например, **А + Б + 100.**

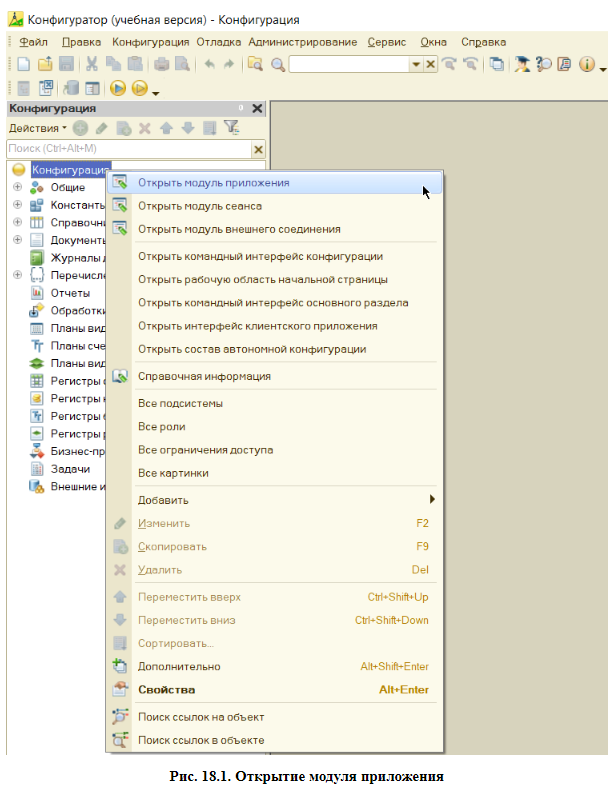
# **Программирование - терминология, модули, ввод данных**

Нам нужно для начала открыть какой-то программный модуль, чтобы в нем прописывать программную логику.

Модулей в платформе 1С достаточно много. но. чтобы было проще начать изучать программирование, мы сделаем все на примере одного модуля, который исполняется при запуске нашего прикладного решения.

Таким модулем называется «Модуль предложения». В нем прописывается логика. которая должна исполниться, когда запускается программа.

Чтобы открыть этот модуль, нужно щелкнуть правой кнопкой мыши по конфигурации и открыть модуль приложения (рис. 18.1).

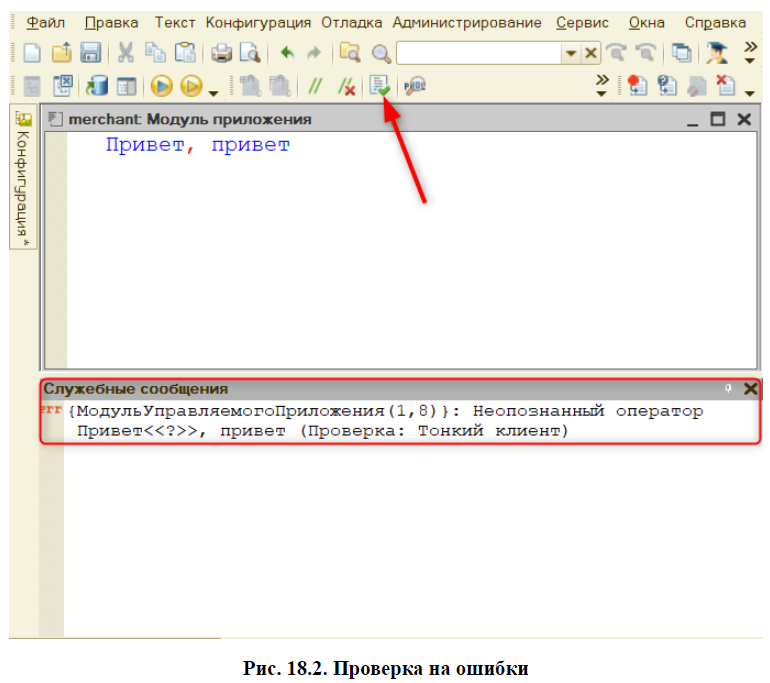
****

**Рисунок 18.1 Открытие модуля приложения**

Есть и другие модули, но про них мы с вами в данном блоке говорить не будем. После открытия области развернем ее на весь экран.

Поначалу область выглядит как текстовый редактор ‚ в котором можно писать какие-то тексты. Но это на самом деле модуль, в котором могут быть прописаны только специальные инструкции на определенном языке — на языке программирования.

Как понять: верно ли написан код в модуле? В конфигураторе есть специальная кнопка, которая позволяет произвести проверку модуля на наличие ошибок. Для этого нужно выбрать в панели главного меню «Текст > Проверка модуля», либо, нажать на кнопку в виде документа с галочкой (рис. 18.2).

****

**Рисунок 18.2 Проверка на ошибки**

Когда мы нажмем на нее, у нас появится ошибка, если мы написали в модуле что-то бессмысленное с точки зрения программирования.

Модуль управляемого приложения — это местоположение ошибки. В скобках пишется номер строки и позиция каретки в этой строке. Далее прописан тип ошибки.

То, что мы написали здесь — неправильно. Поэтому закрываем окно служебных сообщений и убираем написанное.

Первое, с чем мы с вами столкнемся, — это то, что нам нужно выводить информацию о происходящем в нашем модуле. Если сейчас обновить конфигурацию и запустить пользовательский режим, то никаких сообщений или никакого вывода у нас не будет.

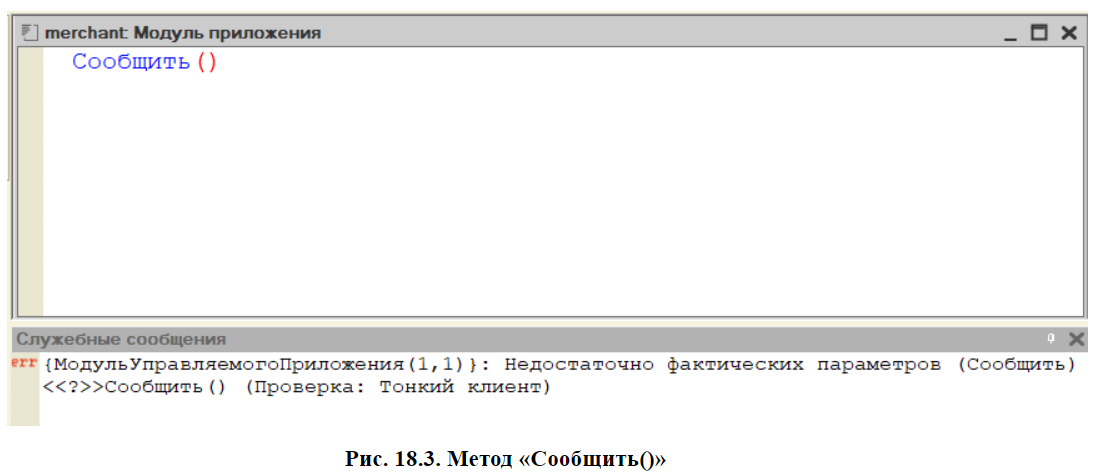
Поэтому познакомимся с первым методом, который позволяет выводить информацию. Это метод «Сообщить».

У всех методов указываются круглые скобки — это их отличительный признак в программном коде.

Итак, мы вызываем наш метод и указываем у него скобки.

Теперь нажмем на кнопку «Проверка синтактических ошибок».

Мы увидим, что вновь возникла ошибка в модуле управляемого приложения. Только теперь в тексте ошибки написано «(1,1)»: это номер строки и номер позиции в строке, где допущена ошибка. Дальше идет текст ошибки: «Недостаточно фактических параметров» (рис. 18.3).

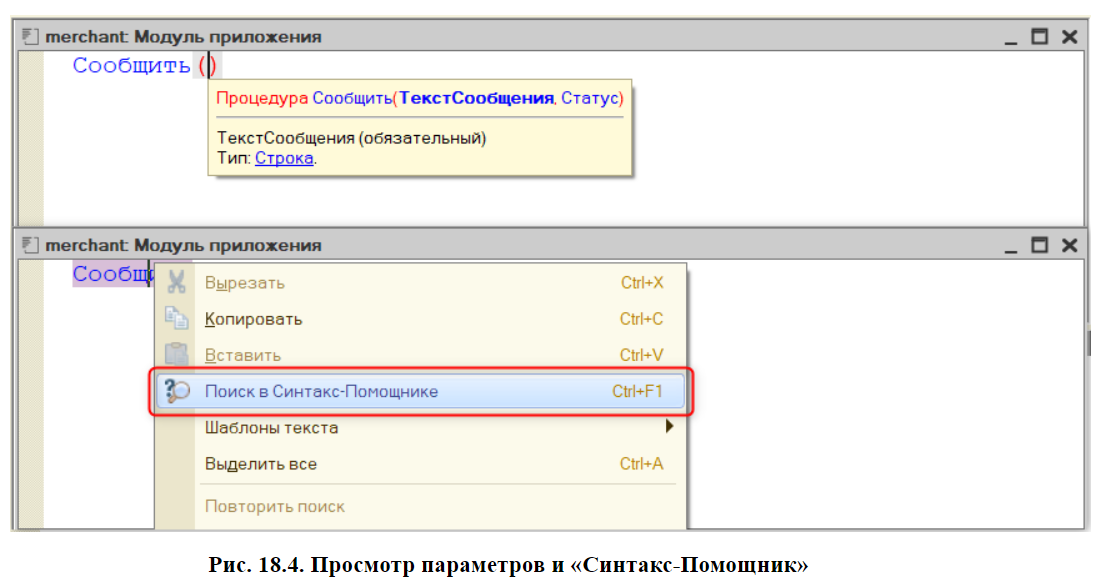
****

**Рисунок 18.3 Метод «Сообщить ()»**

У методов могут быть параметры. Метод «Сообщить» -— не исключение. Чтобы посмотреть список параметров того или иного метода, нужно нажать комбинацию клавиш «Shift + Ctrl+ пробел», поставив курсор в скобках.

Мы увидим все параметры, которыми обладает данный метод.

Параметры метода можно увидеть и другим способом: достаточно щелкнуть правой кнопкой мыши по методу и выбрать пункт «Поиск в «Синтакс-Помощнике» (рис. 18.4).

****

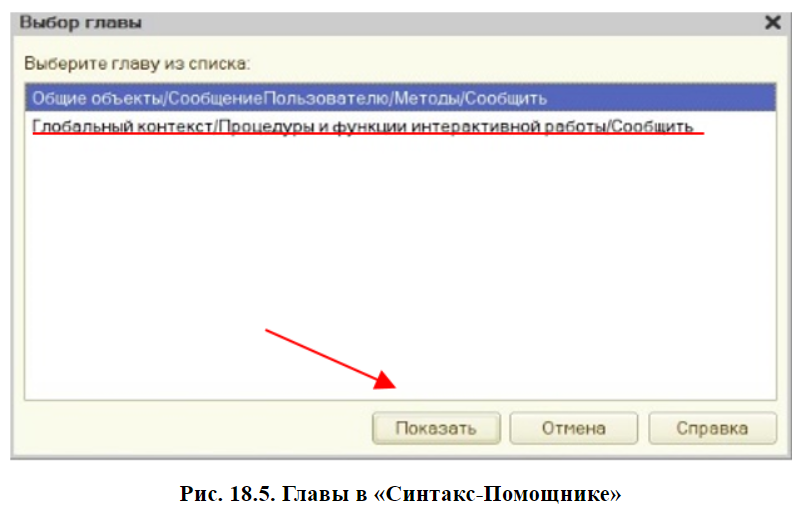
**Рисунок 18.4 Просмотр параметров и «Синтаксис-Помощник»**

«Синтакс-Помощник» — это ваш первый помощник при написании программного кода. Им пользуются даже опытные разработчики, поскольку можно подзабыть параметры или последовательность параметров того или иного метода.

Мы выполним поиск в «Синтакс -Помощнике». У нас появится выбор главы. На самом деле, «Синтакс-Помощник» сейчас осуществляет поиск по всей справке конфигуратора - в конфигураторе есть целая большая книга справки.

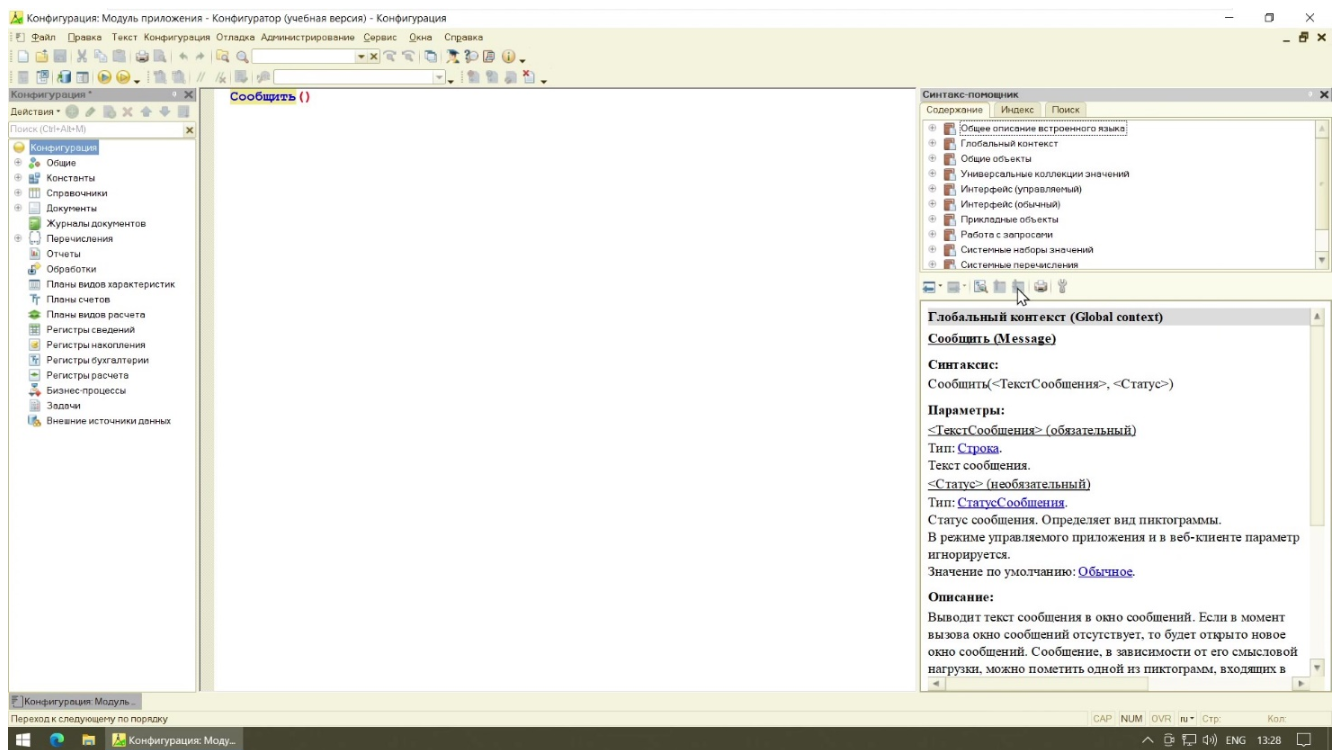
У нас на выбор есть главы «Общие объекты/СообщениеПользователю/Методы/Сообщить» и «Глобальный контекст/Процедуры и функции интерактивной работы/ Сообщить».

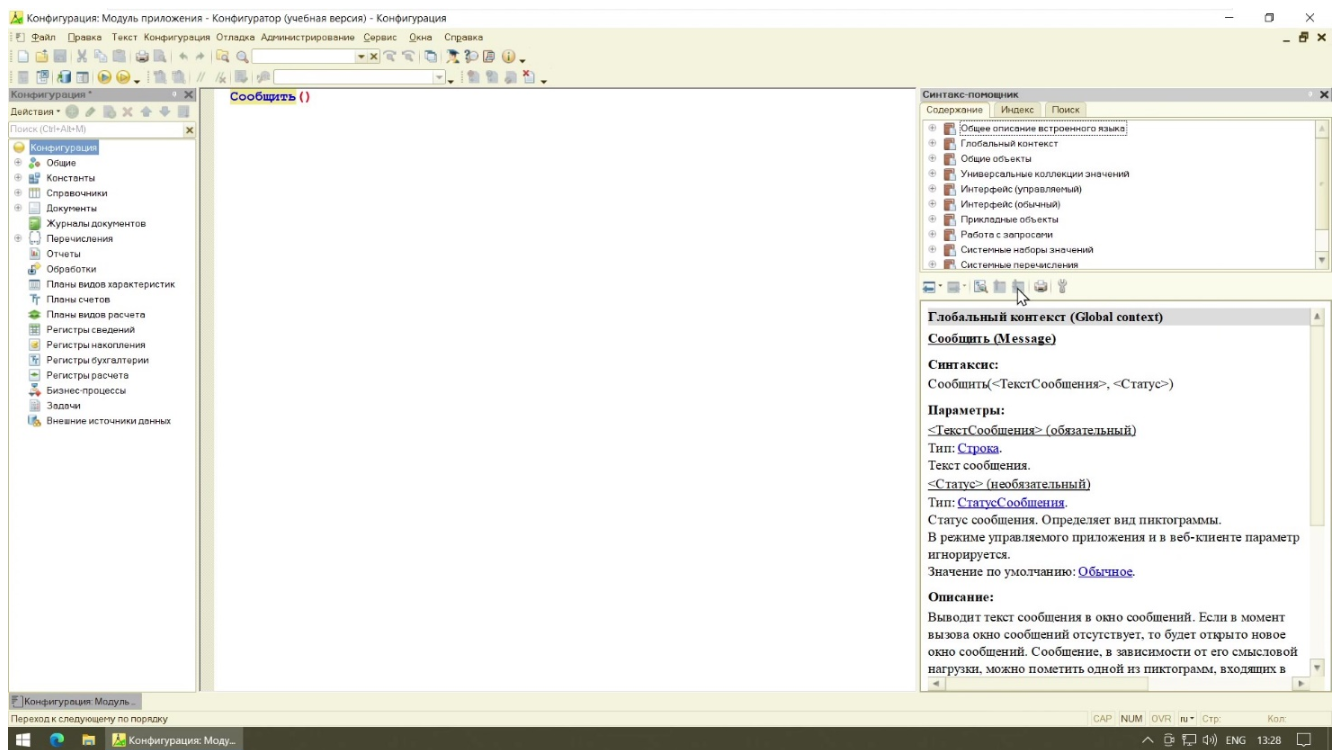
Выберем вторую главу и нажмем «Показать» (рис. 18.5).

****

**Рисунок 18.5 Главы в «Синтаксис - Помощнике»**

Появится подсказка, в которой мы можем ознакомиться с описанием данного метода (рис. 18.6).

****

****

**Рисунок 18.6 «Синтаксис - Помощник»**

Здесь есть вся информация: как пишется данный метод в программном коде, какими параметрами он обладает, что это за параметры, какие типы данных они принимают и так далее.

Описание дает понимание о назначении данного метода: «Выводит текст сообщения в окно сообщений. Если в момент вызова окно сообщений отсутствует, то будет открыто новое окно сообщений». И так далее.

Однако нас сейчас это не сильно интересует, поэтому закрываем «Синтаксис-Помощник» и попробуем написать свой первый программный код.

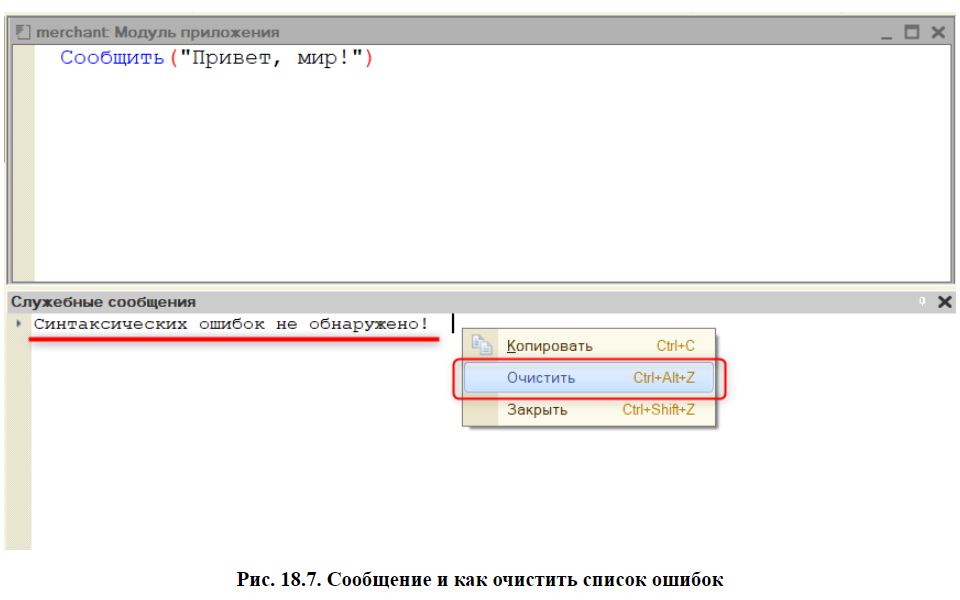
Напишем самые распространенные слова, которые пишутся при изучении вывода информации в программировании: «Привет, мир».

Для того чтобы это сделать, нужно в двойных кавычках написать текст, который хотим вывести. В двойных кавычках нужно писать для того, чтобы обозначать программе, что те или иные строчки относятся к тексту: недостаточно просто написать желаемый текст. Нужно этот текст выделить в двойные кавычки, то есть обозначить границы, где у нас начинается строка, а где она заканчивается.

Строковый тип — это первый тип данных, с которым мы столкнулись.

Проверим наличие синтаксических ошибок. Появилось сообщение «Синтактических ошибок не обнаружено».

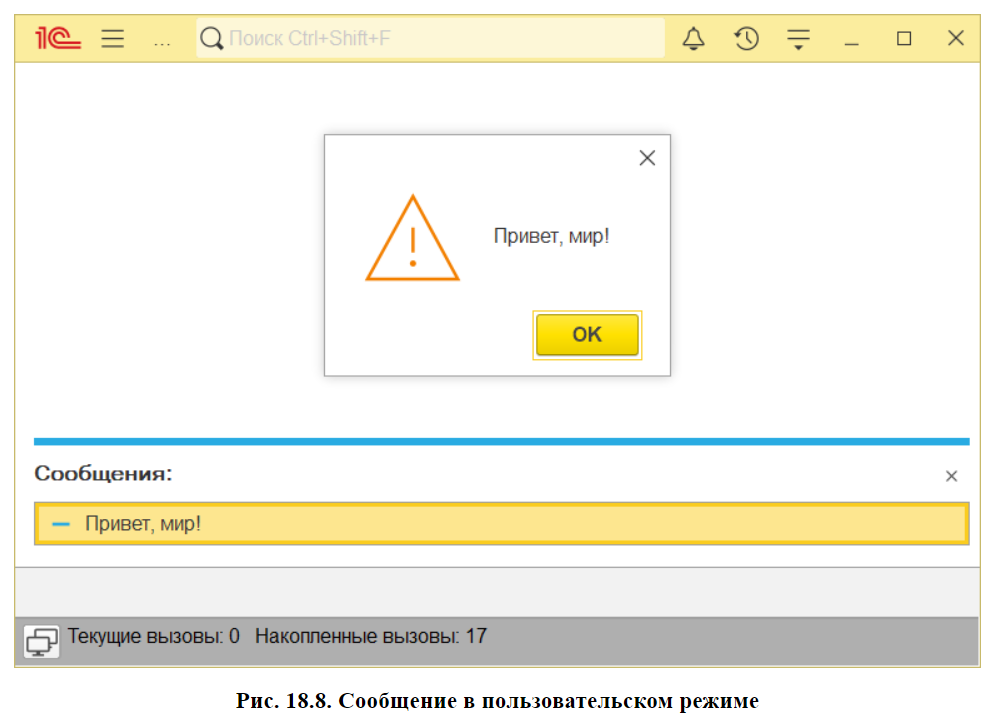
Лишний раз не забывайте очищать данное окно от ошибок. Тогда не будет лишней информации — старые ошибки не затираются автоматически. Также можно каждый раз закрывать окно оповещений об ошибках (рис. 18.7).

****

**Рисунок 18.7 Сообщение и как очистить список ошибок**

Обновляем конфигурацию и запускаем пользовательский режим. Откроется программа, а внизу выведется сообщение.

Если мы хотим более подробно рассмотреть данное сообщение, мы можем два раза левой кнопкой мыши нажать по полю строки: откроется сообщение в отдельном окне (рис. 18.8).

****

**Рисунок 18.8 Сообщение в пользовательском режиме**

Таким образом, мы написали первую строчку программного кода.

Продолжим выводить сообщения. Спросим у мира как у него дела.

Для этого мы сделаем перенос кода на новую строчку. С новой строки еще раз напишем вызов метода «Сообщить».

При написании кода следует запомнить еще одну комбинацию клавиш: «CTRL+Пробел». Работает она следующим образом: платформа пытается предугадать конструкцию, которую мы хотим написать. И платформа либо однозначно угадает и допишет конструкцию, либо предложит варианты, которые начинаются с тех же символов.

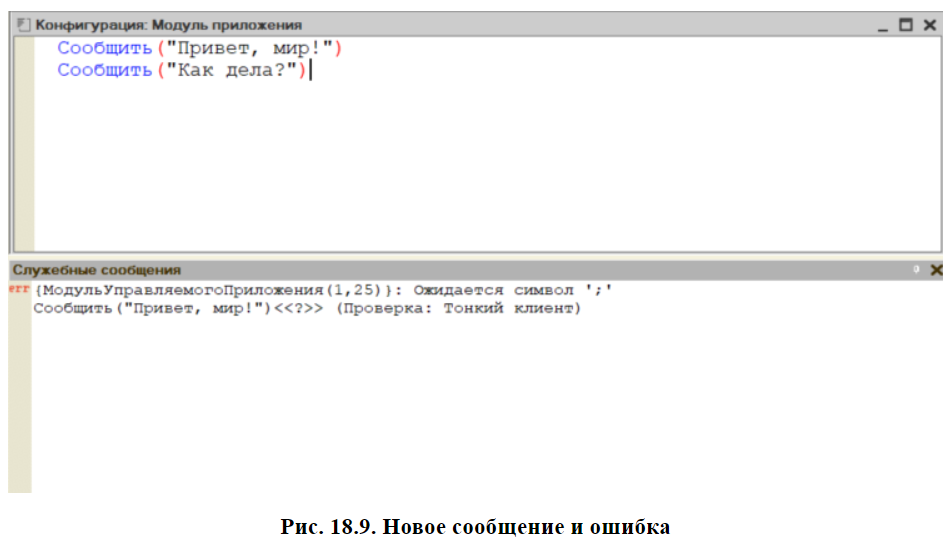
Если написать «Сооб» и нажать комбинацию, то платформа допишет еще четыре буквы: она смогла угадать, что мы имели в виду.

Внутри скобок в двойных кавычках напишем сообщение: «Как дела?».

После появления новой строчки кода нажмем на кнопку проверки модуля.

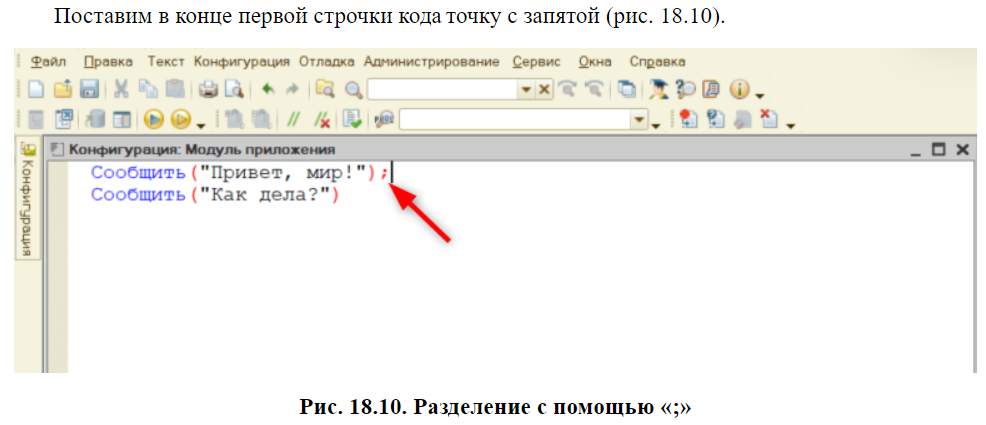
Возникла новая ошибка: ожидается символ «;» с указанием местоположения ошибки.

Для того чтобы не считать вручную номер строки и позицию символа в коде, нажмем два раза левой кнопкой мыши по этой ошибке, и конфигуратор нам автоматически установит курсор на нужной ошибочной строке кода (рис. 18.9).

****

**Рисунок 18.9 Новое сообщение и ошибка**

Поставим в конце первой строчки кода точку с запятой (рис. 18.10).

****

**Рисунок 18.10 Разделение с помощью «;»**

Точка с запятой нужна для разделения различных выражений или команд в коде между собой. Такая логика есть во многих языках программирования.

Нам, людям, понятно, что эти две строчки кода находятся на двух разных строках и это разная логика. Однако для компьютера это все выглядит как одна строка с командами, как кинолента.

Поэтому требуется разделить эти строчки по смыслу, используя символ «;».

Выполним проверку синтактических ошибок — ошибок не обнаружено.

Для удобства чтения кода рекомендуется писать его в таком формате: новое выражение или команда располагается на новой строке и отделяется от предыдущего символом «точка с запятой».

Обновим конфигурацию и запустим пользовательский режим.

Внизу окна отобразятся два сообщения: «Привет, мир!» и «Как дела?» (рис. 18.11).

****

**Рисунок 18.11 Итог в пользовательском режиме**

Мы разобрались с символом точка с запятой и поняли, для чего он нужен. Ознакомились с базисом, от которого можно двигаться дальше и изучать более детально программный код.

# **Программирование — переменные**

Продолжаем изучение синтаксиса языка 1С. Познакомимся с тем, что можно описывать в модуле приложения.

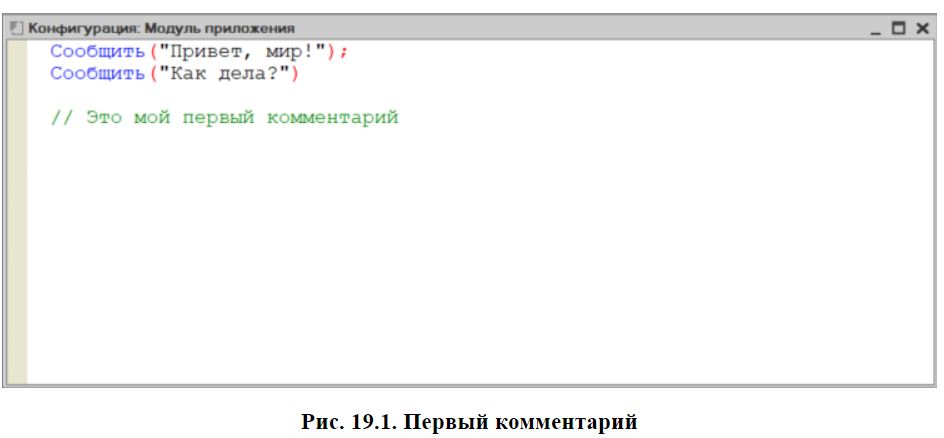
Одна из **важнейших** частей в написании кода — это написание комментариев.

Можно написать любую инструкцию, любой программный код: как он будет работать — зависит от вашего замысла. Но другой разработчик может сходу не разобраться в вашем коде.

Поэтому важно оставлять комментарии к программному коду: они помогают разобраться в назначении конструкций как другим разработчикам, так и вам самим. Поскольку назначение того, что вы программировали раньше, можно забыть.

Комментарии пишутся в программном коде с помощью специального маркера — слэш «/». Двойной правый слэш, или двойная наклонная черта вправо, позволяет в коде написать любой текст, который захочется.

Попробуем написать первый комментарий: «Это мой первый комментарий» (рис. 19.1).

****

**Рисунок 19.1 Первый комментарий**

В комментарии может быть абсолютно любой текст: буквы, цифры, символы и т. д.

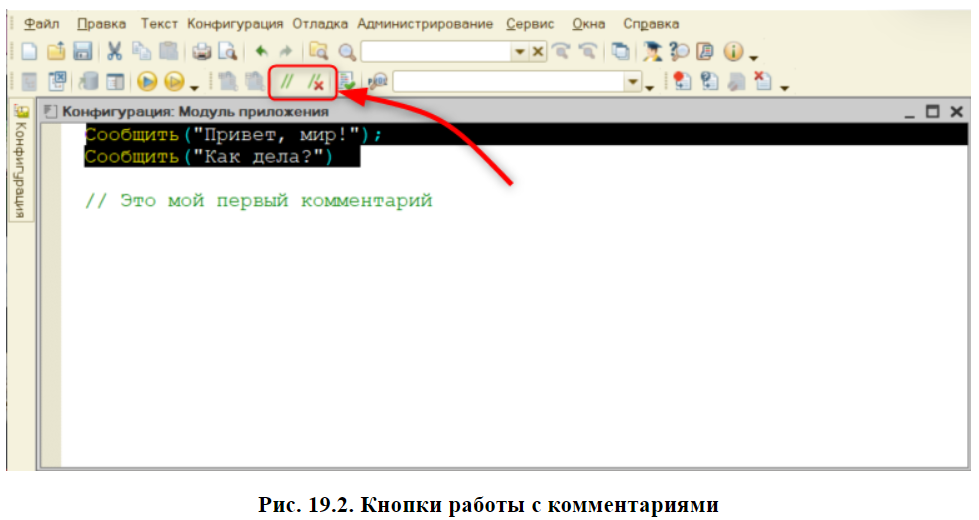
Комментарии не выполняются программой: закомментированные строки будут игнорироваться при исполнении алгоритмов.

Для того чтобы закомментировать строки кода, требуется выделить их и нажать комбинацию клавиш «CTRL + /».

Если нужно убрать комментарии — «CTRL + SHIFT+ /».

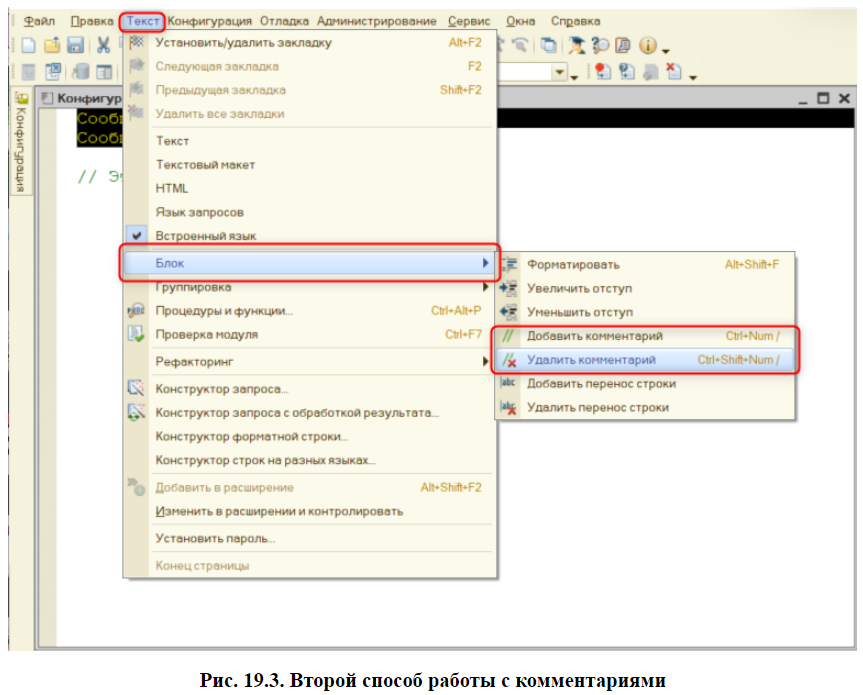
Символ «/» можно найти на цифровом блоке клавиатуры в правой части. Если его нет, то вы можете ставить комментарии с помощью кнопки «Двойной зеленый слэш»

Рядом на панели располагается кнопка для удаления комментариев (рис. 19.2).

****

**Рисунок 19.2 Кнопки работы с комментариями**

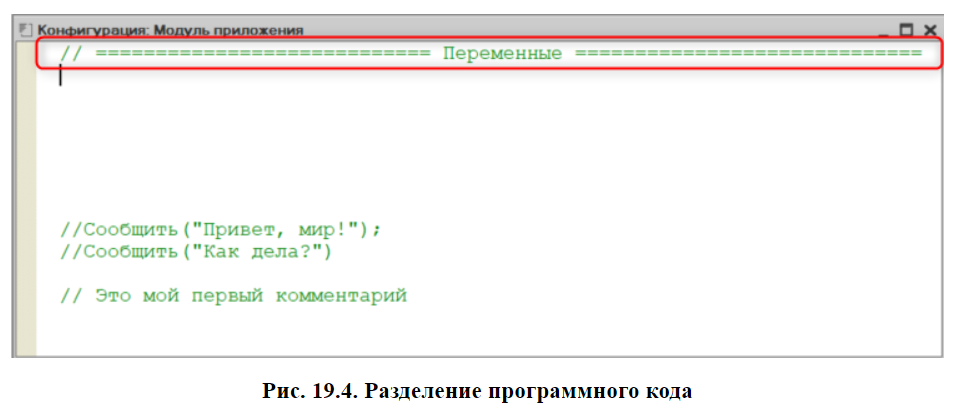
Эти же кнопки можно найти в разделе «Текст>Блок>Добавить комментарий/Удалить комментарий» (рис. 19.3).

****

**Рисунок 19.3 Второй способ работы с комментариями**

Закомментированные строки нам больше не понадобятся, поскольку не будут исполняться программой. Уберем их в самый низ модуля.

Начнем изучение программного кода с основ. Первое, что мы изучим — это переменные. Для удобства мы можем написать комментарий «//===Переменные===», означающий тематические разделы в нашем модуле. С помощью таких комментариев можно разграничить программный код, чтобы было проще и удобнее ориентироваться внутри него (рис. 19.4).

****

**Рисунок 19.4 Разделение программного кода**

Вспомним терминологию программирования.

Переменные - это выделенные области оперативной памяти компьютера, в которых хранится значение определенного типа данных.

В рамках языка 1С нет жесткой типизации. Это означает, что переменной можно задать любой тип значения. Типы значений бывают разные: базовые, примитивные (базовые и еще три типа) и сложные.

Сложные типы — это, например, справочник, ссылка, массив, таблица значений. О некоторых из них мы узнаем позже.

Наша задача сейчас — это создать первые переменные. Переменные должны называться ровно так же, как и прикладные объекты 1С: у них в названии не должно быть специальных символов (кроме нижнего подчеркивания), и они не должны начинаться с цифр.

Также для названия переменных еще добавляется требование о том, что переменные не должны называться как служебные слова. Таких служебных слов на самом деле не очень много, и они будут нам встречаться при написании программного кода. Например, «Возврат».

Служебные слова и конструкции в языке программирования 1С по умолчанию выделяются красным цветом.

Создадим переменную. Ее можно назвать как угодно, например‚ «А». Следующее, что необходимо сделать — это присвоить значение переменной.

Сейчас для платформы это просто символ. Причем платформа не понимает его назначение, и, если выполнить проверку синтактических ошибок, появится ошибка «Неопознанный оператор».

Чтобы ее не было, требуется написать символ «равно» после названия переменной (можно через пробел, можно без него). А дальше присвоить этой переменной какое-то значение.

Символ «=» называется оператором присваивания, то есть он задает значение нашей переменной. Присвоим переменной «А» значение 10. Можно создать еще одну переменную через точку с запятой: «В = 5».

Создадим выражение, которое посчитает сумму этих двух переменных. Для этого потребуется создать еще одну переменную, либо мы можем сразу поместить результат нашего выражения в метод, который выведет информацию.

Метод вывода некоторой информации называется «Сообщить». Это один из методов. Дополним наше выражение и напишем: «А +Б+ 100;».

Если в коде просто написать такое выражение, то при проверке на синтактические ошибки возникнет ошибка.

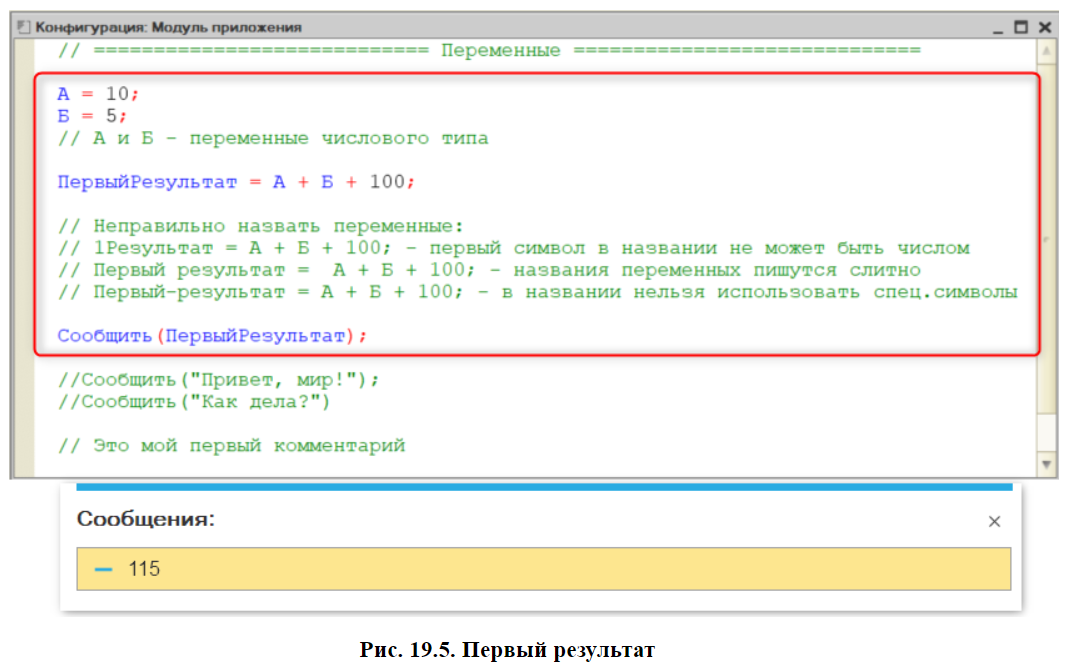
Просто так выражение в коде не может существовать. Нам нужно обязательно значение этого выражения куда-то поместить. Либо в метод «Сообщить»: «Сообщить (А+Б+100)», либо результат этого выражения поместить в какую-то переменную. Например, в переменную «Первый результат».

Стоит обратить внимание, что переменная «Первый Результат», именуется таким же образом, как называются прикладные объекты: нельзя начинать имя переменной с числа.

Теперь выведем переменную: напишем сначала «Сообщить()» . Затем в скобках напишем «перв» и нажмем комбинацию «CTRL + Пробел».

Среда разработки автоматически все допишет. Проверим наличие синтактических ошибок: не обнаружено. Далее обновим конфигурацию и запустим пользовательский режим.

В пользовательском режиме появится сообщение «115» (рис. 19.5).

****

**Рисунок 19.5 Первый результат**

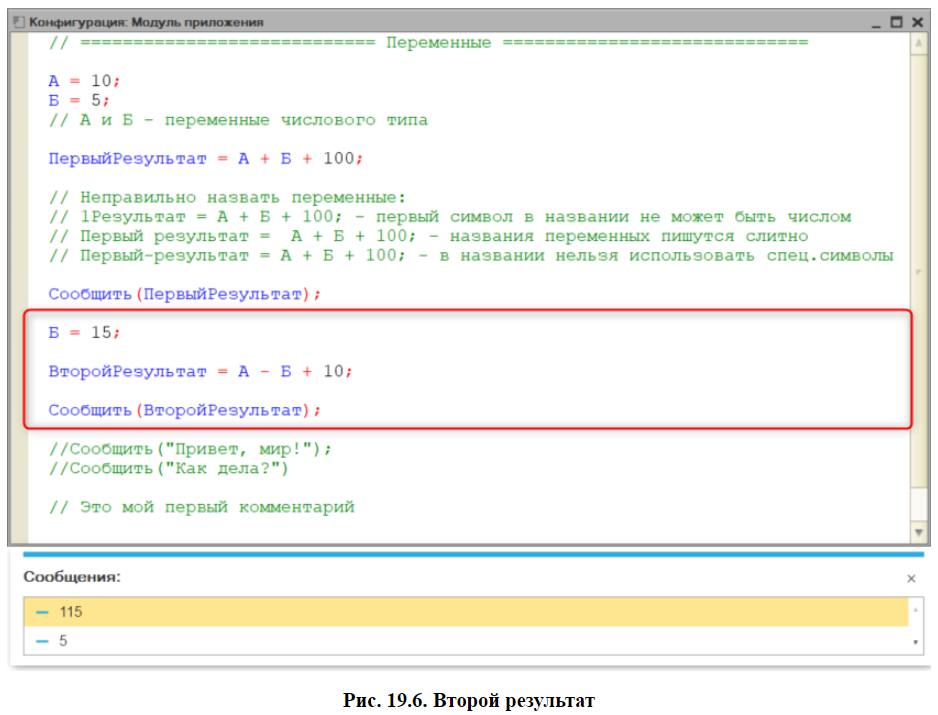
Теперь предположим, что мы хотим получить другой результат, попутно изменив значение переменной «Б».

Присвоим новое значение переменной Б: «Б = 15;» и создадим переменную «ВторойРезультат=А - Б+ 10».

Результат должен быть равен пяти.

Выведем значение переменной «ВторойРезультат» в коде.

Обновим конфигурацию и запустим пользовательский режим. Теперь мы увидим два результата, поскольку в коде срабатывают две команды «Сообщить» (рис. 19.6).

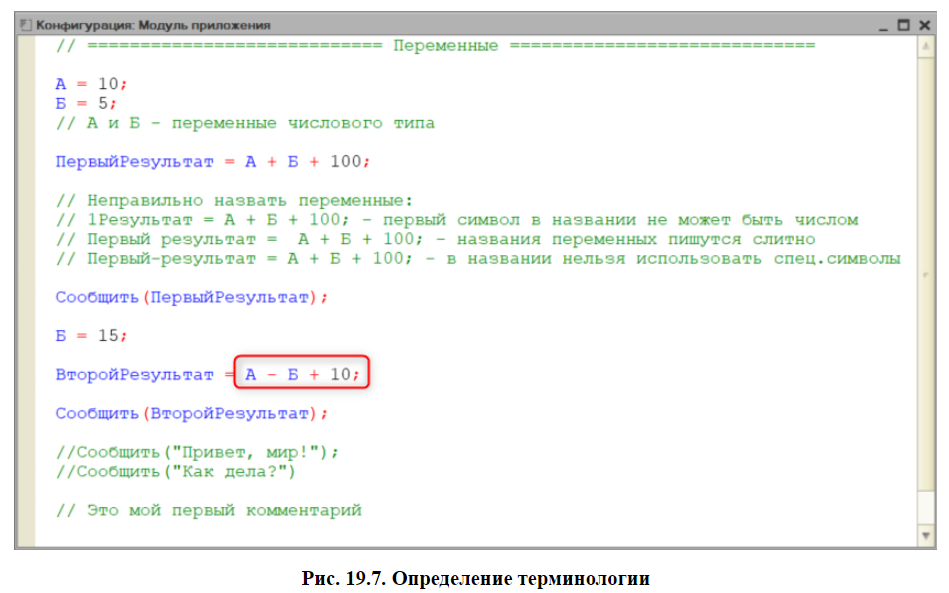
****

**Рисунок 19.6 Второй результат**

Вспомним, что в терминологии есть операторы и операнды.

Операнды — это элементы, которые описываются в выражении. В нашем случае — в выражении второго результата в качестве операндов выступают наши переменные «А», «Б» и число 10.

Оператор — это определенный набор символов, который позволяет описывать различные действия в алгоритмах программы, такие как: +, \*, = и прочие. В нашем случае это «-» и «+» (рис. 19.7).

****

**Рисунок 19.7 Определение терминологии**

Мы рассмотрели тематику переменных: как с ними работать, узнали, что значение переменной не привязывается к конкретному типу данных и т. д.

Если мы в программном коде упомянем переменную с маленькой буквы, а изначально она была названа с большой буквы — ошибки не будет.

В языке 1С можно указывать переменные по-разному. Если мы напишем «Сообщить вТОройРЕЗУЛЬТАТ)», то 1С нас поймет, и при запуске алгоритма никаких ошибок не возникнет (рис. 19.8).

****

**Рисунок 19.8 Дополнительная информация**

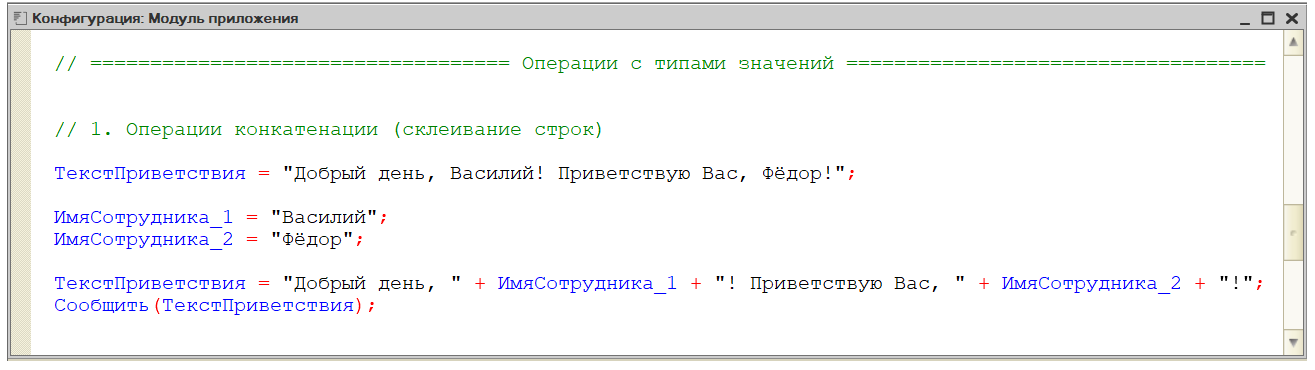
Итак, мы на базовом уровне познакомились с переменными

# **Программирование — работа с типами значений (строка, неявные преобразования)**

**Базовые типы** данных — это строка, число, дата и булево.

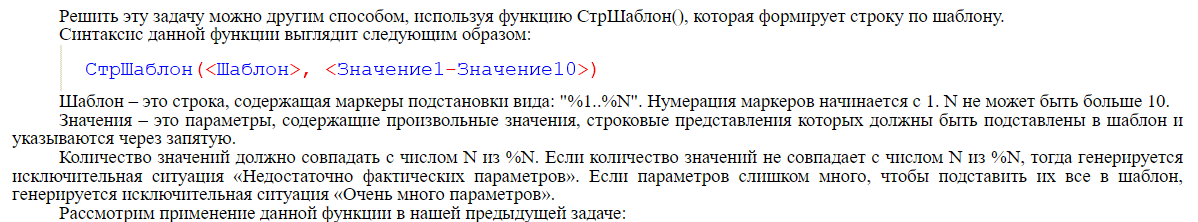
**Конкатенация** — это операция склеивания объектов линейной структуры. В 1С конкатенация используется для того. чтобы присоединить одну строку к другой.

К примеру, необходимо получить фразу: «Добрый день, Василий! Приветствую Вас, Фёдор!». Но нужно учесть, что фамилии сотрудников хранятся в отдельных переменных. Тогда код будет выглядеть следующим образом:



Решить эту задачу можно другим способом, используя функцию СтрШаблон(), которая формирует строку по шаблону.

Синтаксис данной функции выглядит следующим образом:

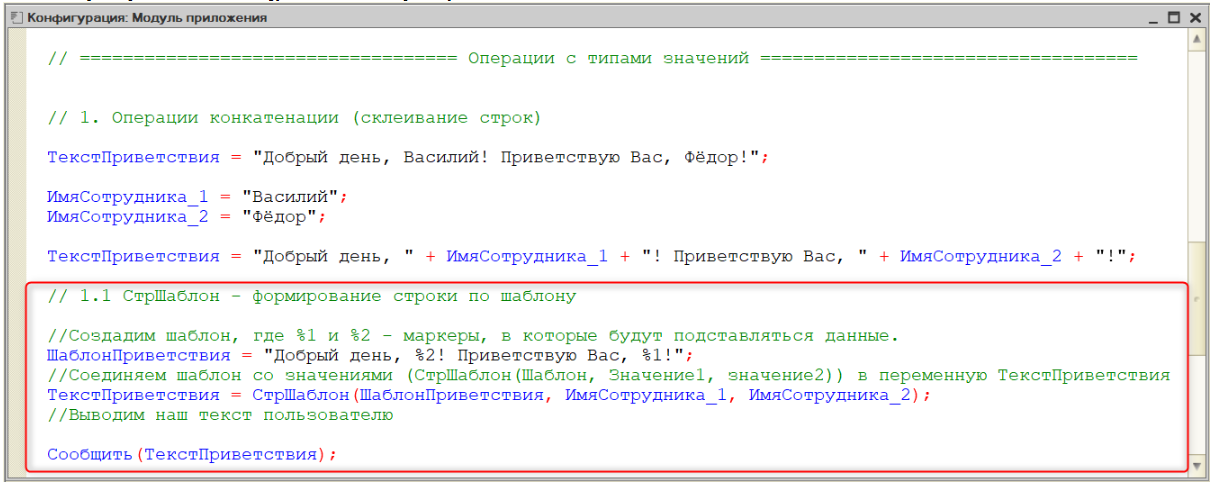


Шаблон — это строка, содержащая маркеры подстановки вида: "%1.. %N". Нумерация маркеров начинается с 1. М не может быть больше 10.

Значения — это параметры, содержащие произвольные значения, строковые представления которых должны быть подставлены в шаблон и указываются через запятую.

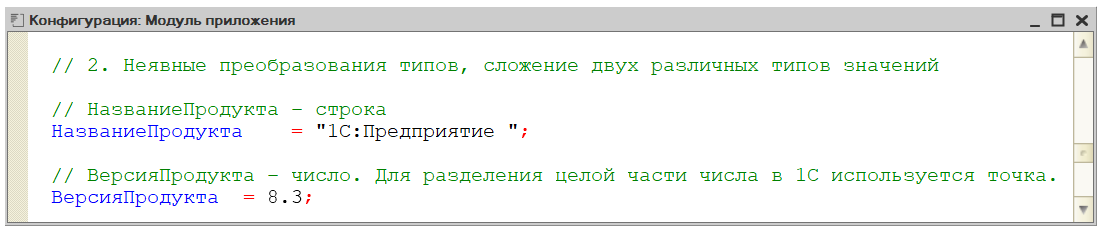
Количество значений должно совпадать с числом N из %N. Если количество значений не совпадает с числом N из %N, тогда генерируется исключительная ситуация «Недостаточно фактических параметров». Если параметров слишком много, чтобы подставить их все в шаблон, генерируется исключительная ситуация «Очень много параметров».

Рассмотрим применение данной функции в нашей предыдущей задаче:

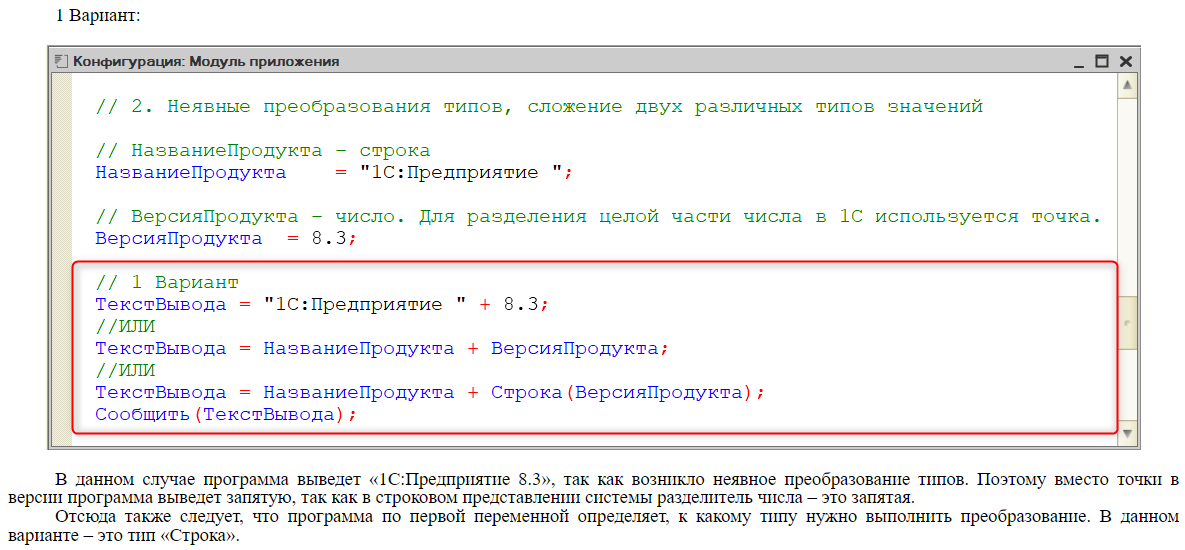


Неявное преобразование типов происходит тогда, когда над переменными выполняются какие-либо операции: сложение, вычитание, конкатенация строк и т. д. При этом типы всех переменных, участвующих в операции, будут преобразовываться в тип первой переменной при работе в «1С: Предприятие».

Рассмотрим несколько видов сложения строки с числом:



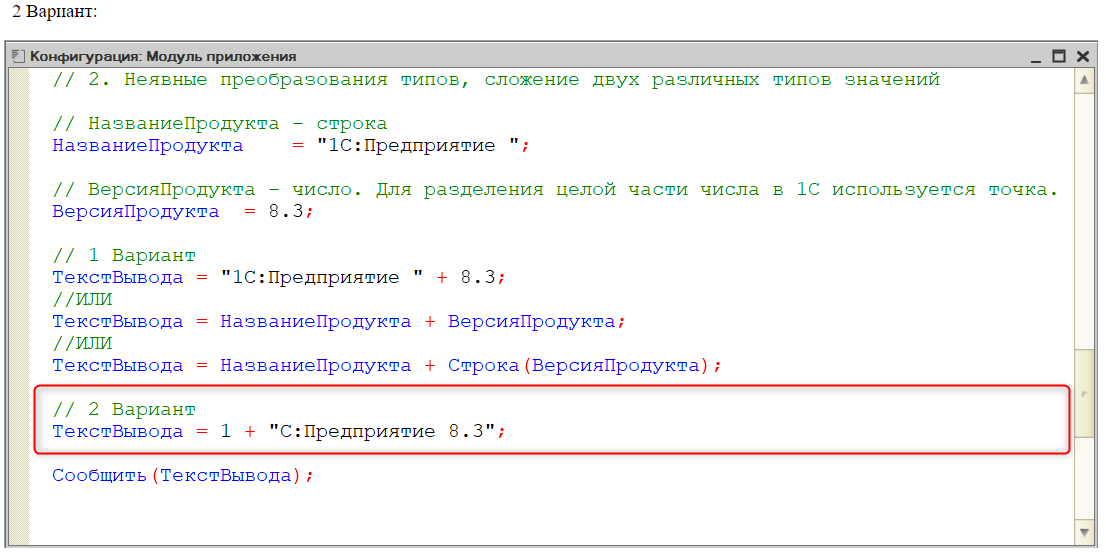
1 Вариант



В данном случае программа выведет «1С:Предприятие 8.3», так как возникло неявное преобразование типов. Поэтому вместо точки в версии программа выведет запятую, так как в строковом представлении системы разделитель числа — это запятая.

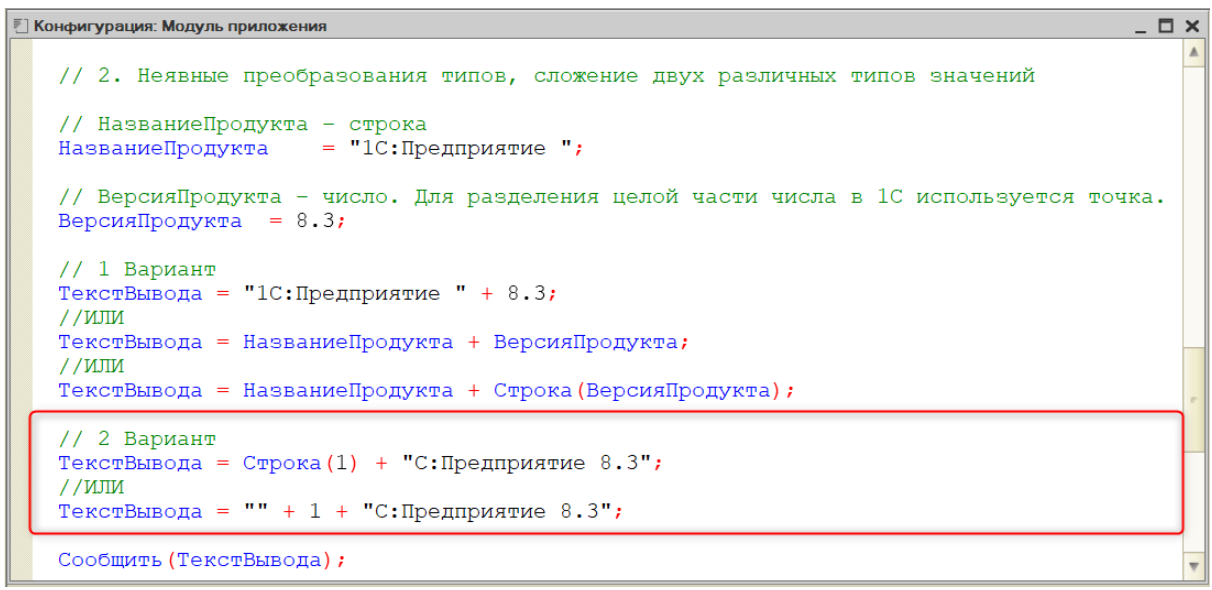
Отсюда также следует, что программа по первой переменной определяет, к какому типу нужно выполнить преобразование. В данном варианте — это тип «Строка».

2 Вариант:



В данном случае программа выдаст ошибку «Преобразование к типу «Число» не может быть выполнено». Это значит, что платформа не смогла преобразовать вторую переменную к типу «Число» и выдала ошибку.

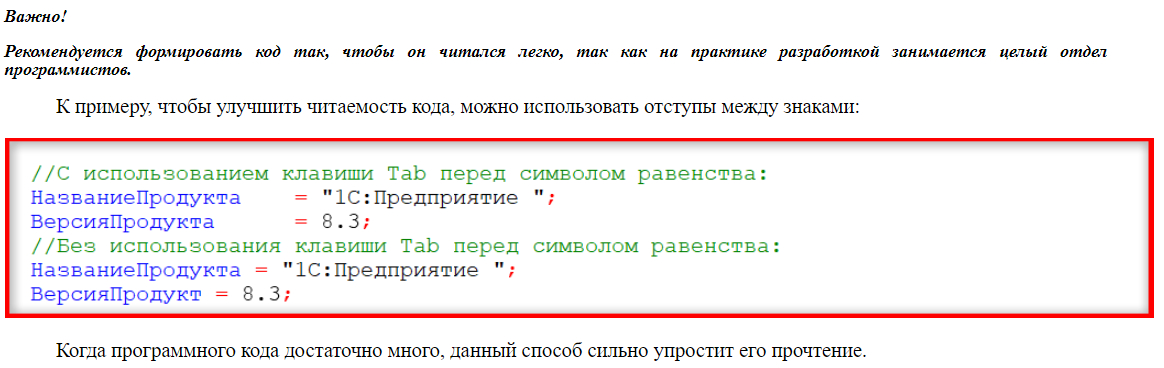
Решить данную ошибку можно, принудительно преобразовав единицу в строковый тип или же добавив в начало пустую строку:



Важно!

Рекомендуется формировать код так, чтобы он читался легко. Так как на практике разработкой занимается целый отдел программистов.

К примеру, чтобы улучшить читаемость кода, можно использовать отступы между знаками:

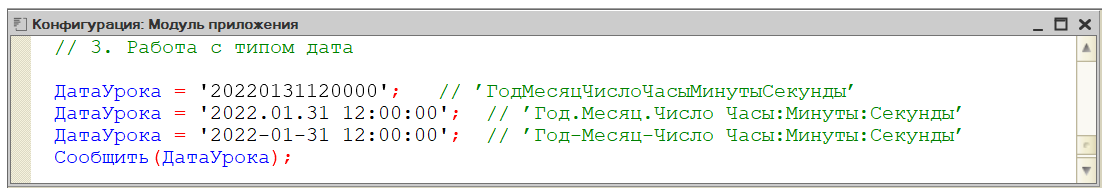


Когда программного кода достаточно много, данный способ сильно упростит его прочтение.

# **Программирование — работа с типами значений (дата)**

Тип «Дата» предназначен для хранения информации о временных интервалах, когда произошло то или иное событие.

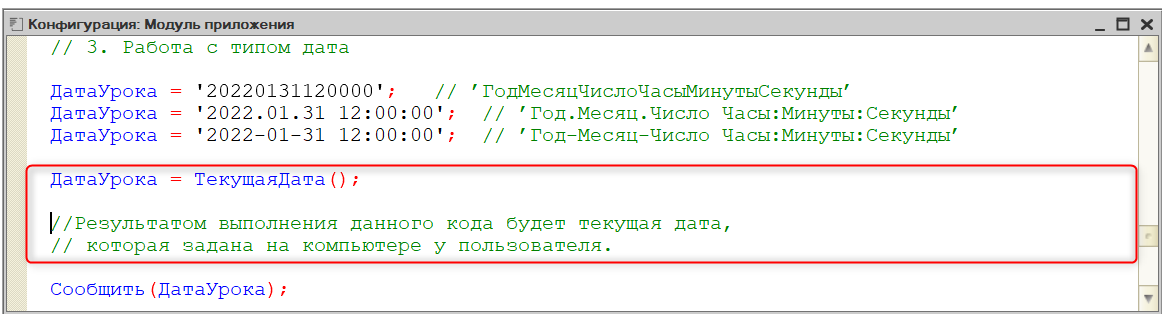
Существует несколько видов записи дат:



Для удобства у типа дата есть разделители: «-», «\_», «/», «\», «:», «.». Пример записи с разделителями описан выше.

Для вывода текущего (системного) времени используется функция ТекущаяДата().

Рассмотрим пример её использования:



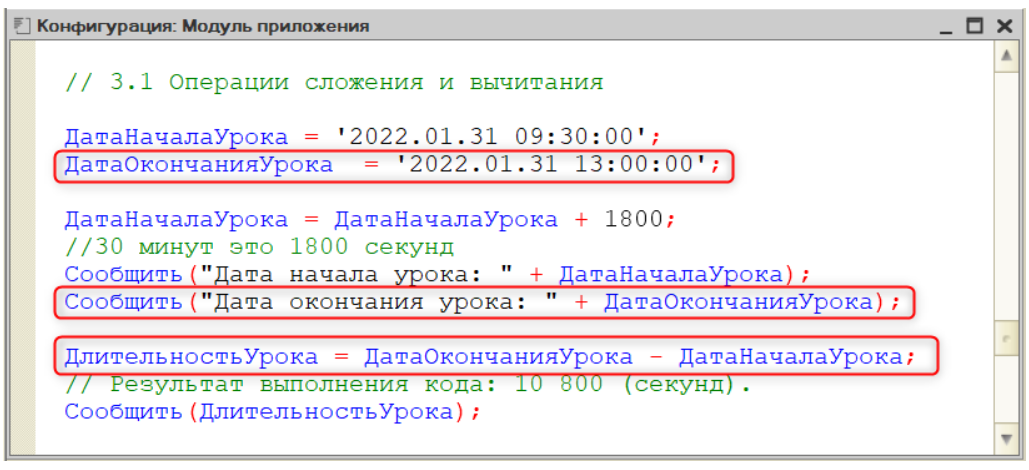
Увеличить значение даты можно с помощью сложения. К примеру, необходимо увеличить время начала урока на 30 минут, тогда код будет выглядеть следующим образом:



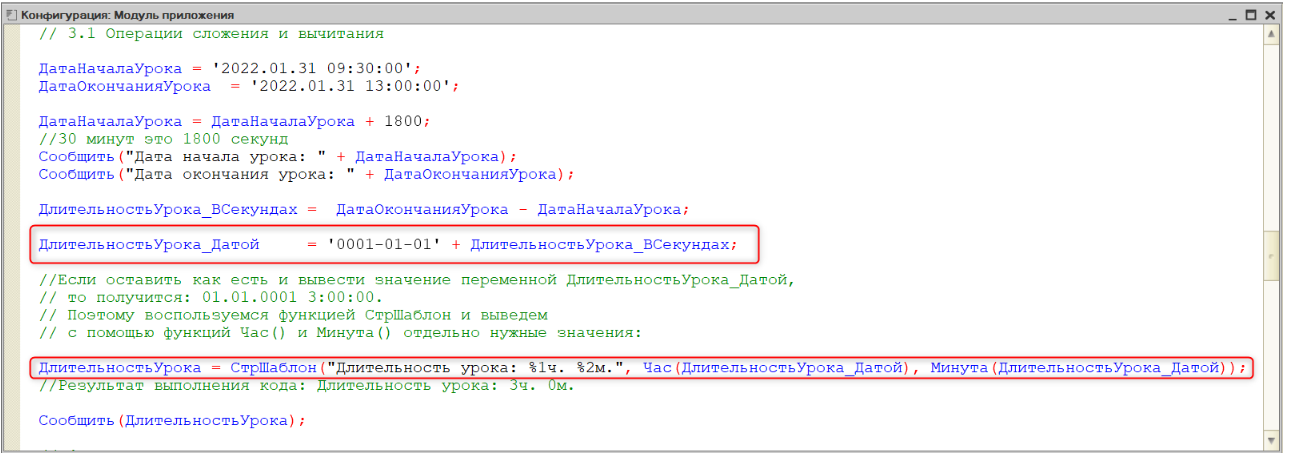
Важно!

Для того чтобы быстро выделить название переменной, необходимо нажать по ней два раза ЛКМ. Для выделения всей строки — три раза.

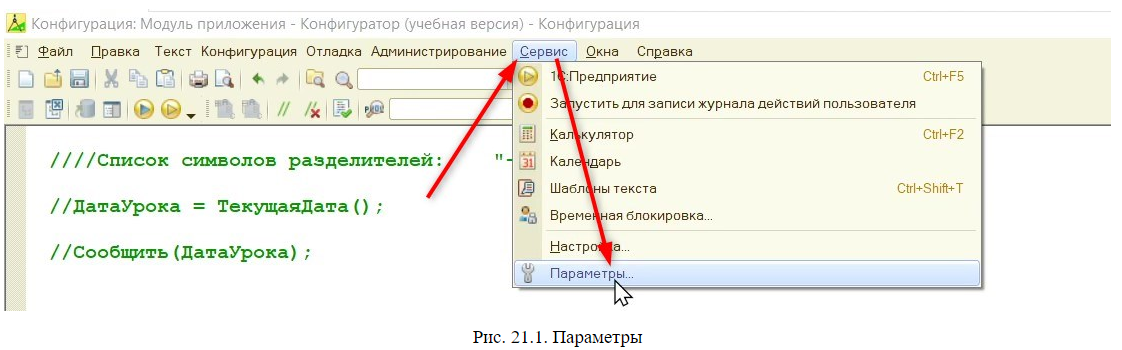
Предположим, необходимо посчитать длительность занятия в секундах, для этого можно воспользоваться операцией вычитания.



Для того чтобы посчитать время урока более корректно, необходимо воспользоваться пустым значением. У строки - это пустая строка, у числа - 0, ау даты - это 0001-01-01. Если прибавить к данному значению полученное время в секундах, мы получим более корректное отображение длительности занятия:

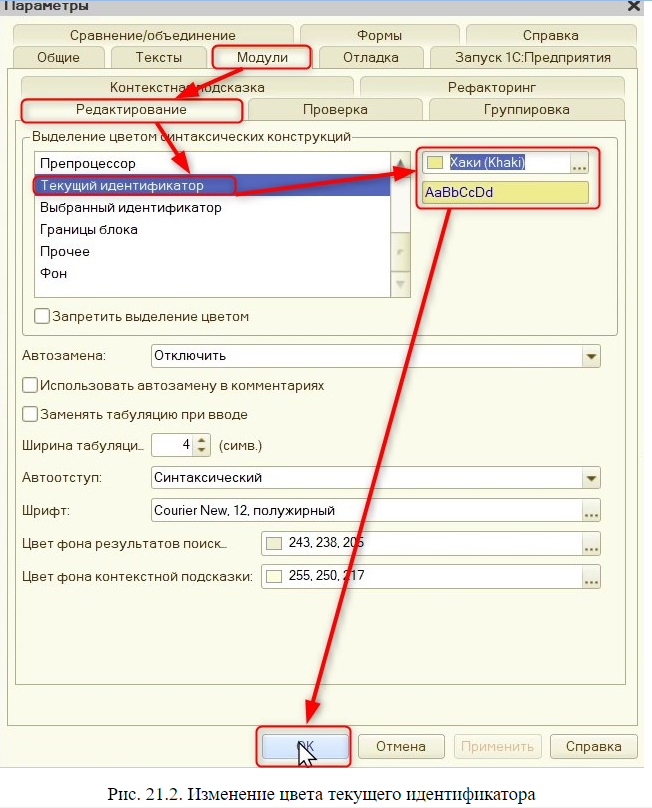


При работе с кодом хотелось бы упомянуть настройку. которая упрощает работу и поиск ошибок, а именно — выделение текущего идентификатора. По умолчанию его цвет белый. Для настройки его цвета откроем параметры через вкладку главного меню «Сервис» (рис. 21.1).

****

**Рисунок 21.1 Параметры**

Далее нужно открыть вкладку «Модули», в ней найти «Редактирование». затем в списке выбрать «Текущий идентификатор» и задать любой удобный цвет (рис. 21.2). Рекомендуется выбрать цвет хаки.

****

**Рисунок 21.2 Изменение цвета текущего идентификатора**

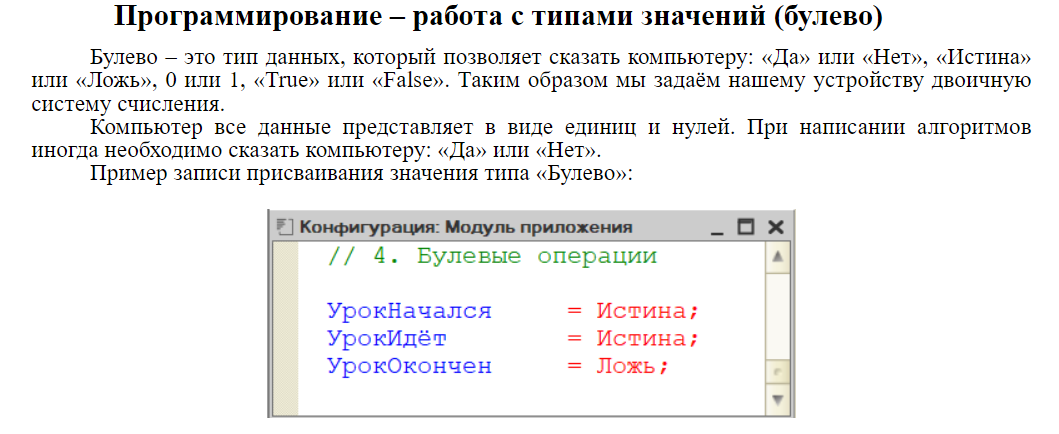
После сохранения данной настройки, если нажать ЛКМ по переменной, в коде будет выделяться все взаимодействия с этой переменной.

# **Программирование — работа с типами значений (булево)**

Булево — это тип данных, который позволяет сказать компьютеру: «Да» или «Нет», «Истина» или «Ложь», 0 или 1, «True» или «False». Таким образом мы задаём нашему устройству двоичную систему счисления.

Компьютер все данные представляет в виде единиц и нулей. При написании алгоритмов иногда необходимо сказать компьютеру: «Да» или «Нет».

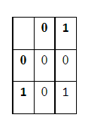
Пример записи присваивания значения типа «Булево»:



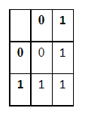
Рассмотрим операции с данным типом, предварительно закомментировав предыдущий код.

Для того чтобы понимать принцип работы с типом «Булево» в вычислительных выражениях, необходимо познакомиться с таблицей истинности.

Таблица истинности — это таблина, описывающая логическую функцию. Простыми словами, данная таблица показывает, что произойдёт, если произвести логическое умножение (рис. 22.1) или логическое сложение (рис. 22.2).

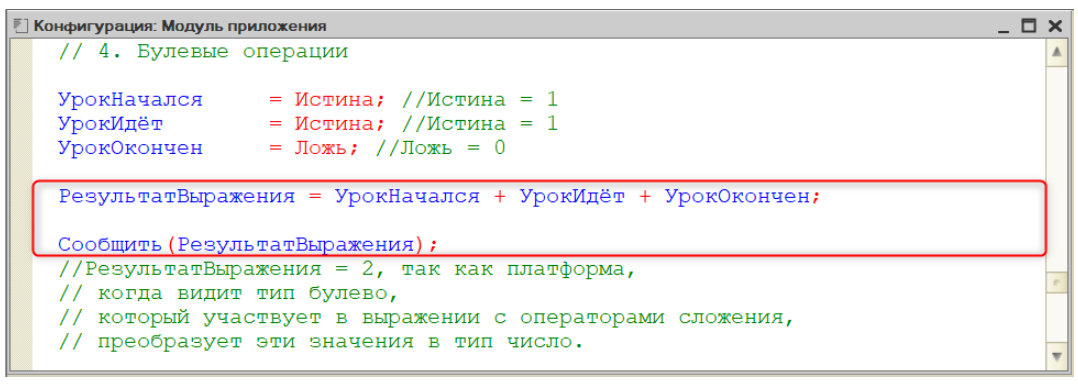
****

**Рисунок 21.1 Таблица истинности – логическое умножение (И)**

****

**Рисунок 21.2 Таблица истинности – логическое умножение (ИЛИ)**

Рассмотрим операции с типом «Булево»:



Для того чтобы производить логические операции, используются другие операторы, а именно:

И - Логическое умножение.

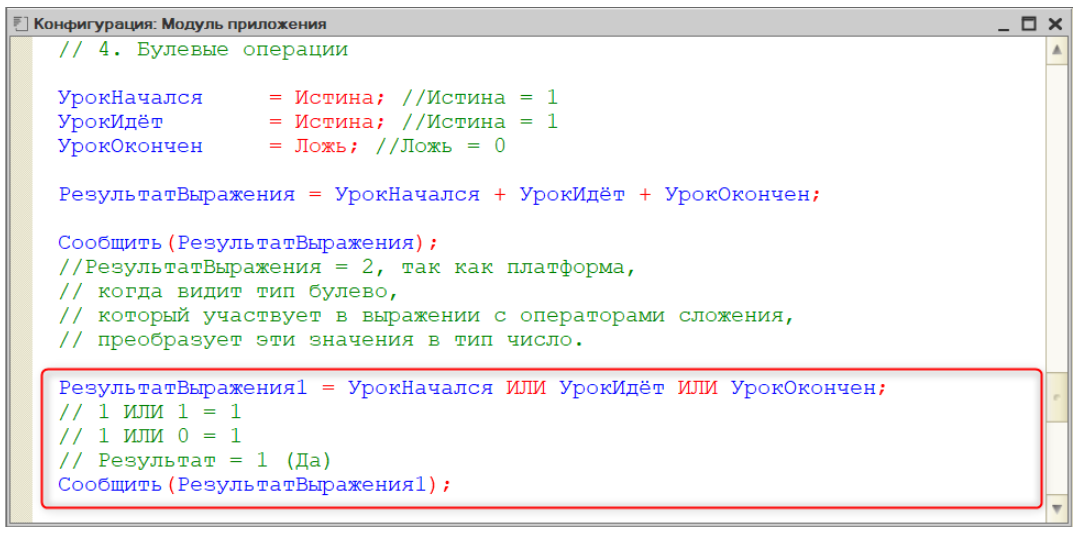
ИЛИ — Логическое сложение.

НЕ - Логическое отрицание (т.е. противоположное значение).

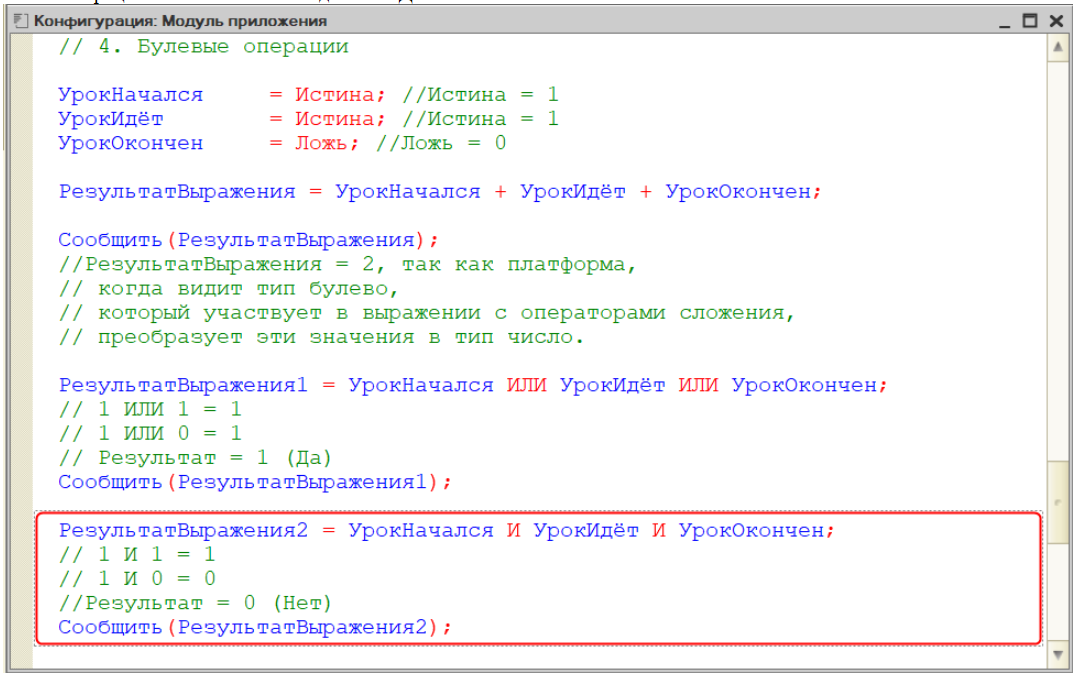
*Важно!*

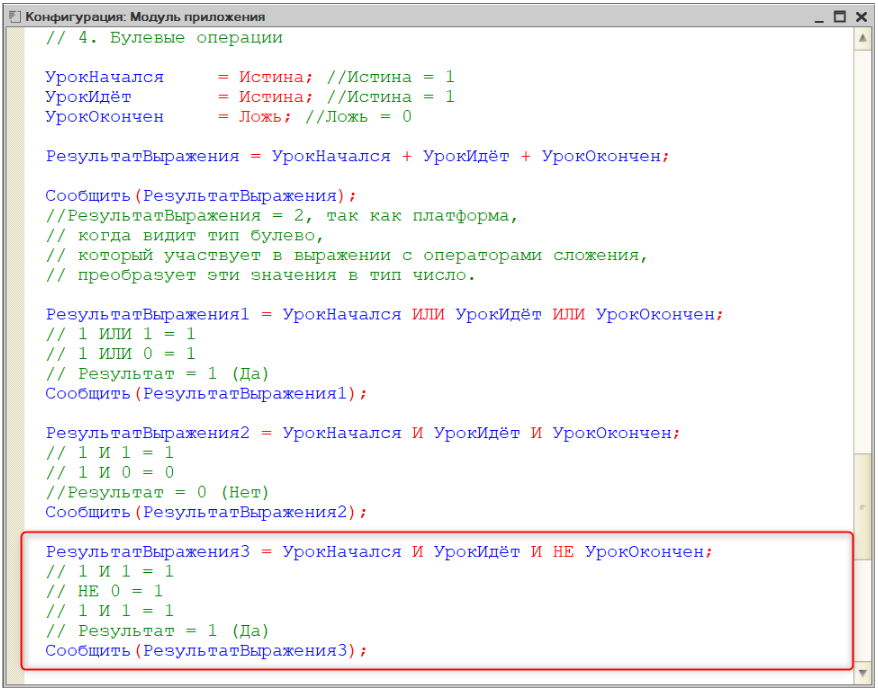
*Нельзя использовать И, ИЛИ, НЕ в качестве имён переменных. Рекомендуется для удобства чтения кода данные операторы писать большими буквами.*

Пример использования логических операций:



В пользовательском режиме высветится «Да» вместо «Истины», так как представление значения типа «Булево» в нашей операционной системе выглядит как «Да» или «Нет»





# **Индивидуальное задание**

**№ 7. Алгоритмы - «Сообщение пользователю»**

Необходимо определить 3 переменные со значениями:

1. Название склада - «Основной»
2. Количество металла - 125
3. Дата подсчёта — 01 апреля 2022 года 10 утра

Сформировать строчку, используя объявленные переменные, для вывода следующего сообщения:

«На дату 01.04.2022 10:00:00 остаток металла на складе «Основной»: 125 шт».

*Примечание:* Для вывода сообщения лучше использовать СтрШаблон().

**№ 8. Алгоритмы — «Расчёт итоговой цены»**

Необходимо определить 3 переменные:

1. Себестоимость товара - 100
2. Наценка — 40 %
3. Ставка НДС - 20%

Требуется вычислить итоговую пену товара для розничного покупателя

*Примечание:*

Наценка = Себестоимость \* ПроцентНаценки / 100:

ЦенаСНаценкой = Себестоимость + Наненка:

НДС = ЦенаСНаценкой \* СтавкаНДС / 100;

`ИтоговаяЦена = ЦенаСНаценкой + НДС.

# **Контрольные вопросы**

1. Что такое «Операнды»?
2. Что такое «Операторы»?
3. Что такое «Переменная»?
4. Какая комбинация клавиш помогает дозаполнить текст в алгоритмах?
5. Для чего нужна конкатенация?
6. Что делает функция «СтрШаблон»?
7. Чему равно пустое значение даты?
8. В какой величине программа считает сложение и вычитание переменных с типом «Дата»?