Code style

Соглашения при написании кода

Валентина Помогаева

О спикере:

- ИП в сфере внедрения программ на основании линейки продуктов 1С
- Экс-декан факультета 1С «Разработчик» университета GeekBrains
- 14 лет в разработке и внедрении программных продуктов 1С
- 6 лет в обучении специалистов 1С

Ошибки в коде

Вопрос: почему в коде появляются ошибки?

Ошибки в коде

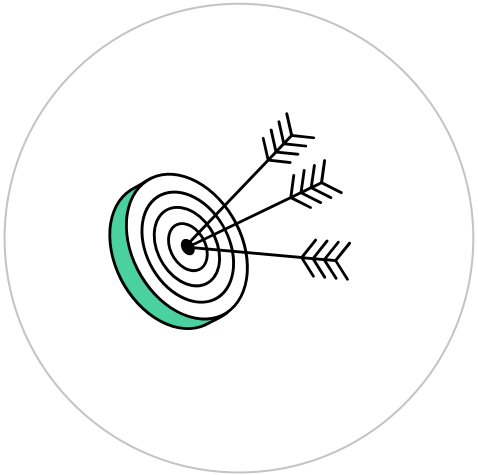
Вопрос: почему в коде появляются ошибки?

Ответ:

- у разработчика мало опыта
- плохой пример для подражания
- опечатки
- проблемы копирования/вставки
- спешка при разработке, обрастание «костылями»

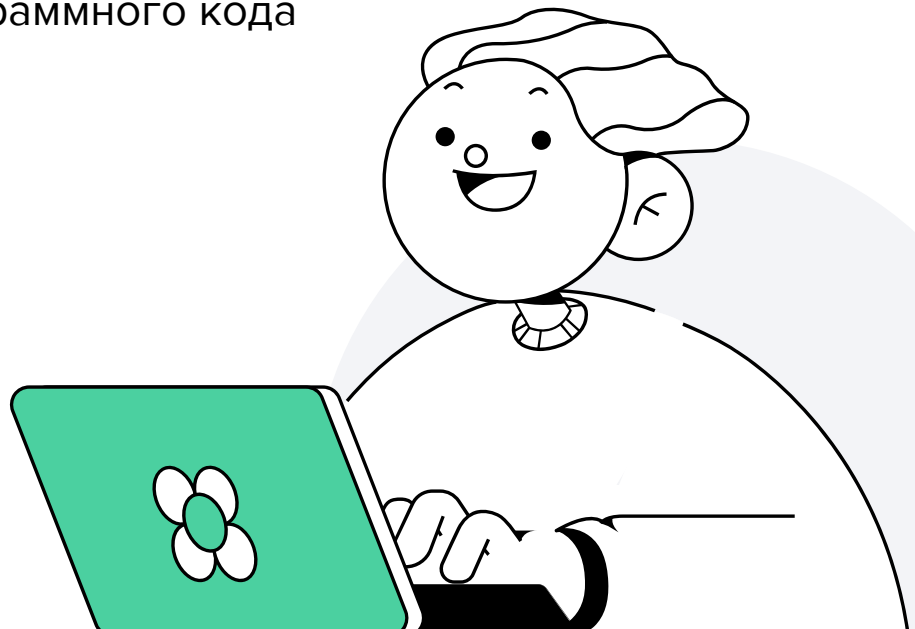
Цели занятия

- Изучить основные правила и типичные ошибки написания программного кода
- Узнать стандарты «1С» по разработке программного кода
- Ознакомиться с инструментами для чтения чужого программного кода



План занятия

1. Стандартизация кода
2. Основные правила написания программного кода
3. Инструменты для чтения чужого программного кода



*Нажми на нужный раздел для перехода

Стандартизация кода

1

Стандартизация программного кода: предпосылка

На заре автомобилестроения возникла потребность упростить обслуживание авто не у поставщика, а в сервисном центре. Поэтому появились общие стандарты на детали.

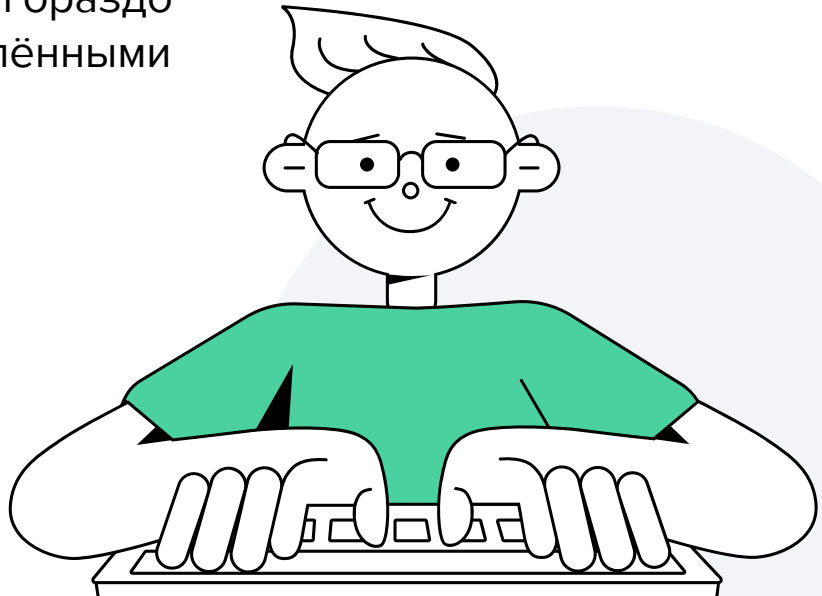
В итоге все производители стали придерживаться стандартов и автомобили стало проще обслуживать

Стандартизация программного кода

Стандартизация актуальна и в разработке программных продуктов.

Мы можем выпускать уникальный продукт, но если его будет невозможно или очень сложно сопровождать, использовать его будет непросто. В итоге стоимость эксплуатации будет высокой.

Поэтому, даже с экономической точки зрения, гораздо выгоднее писать код в соответствии с определёнными в сообществе стандартами



Стандартизация программного кода

Кроме упрощения сопровождения, стандартизованный код гораздо проще читать и понимать.

Можно написать условие в одну строку:

```
Если Делитель = 0 Тогда Результат = 0; Иначе Результат = Делимое / Делитель; КонецЕсли;
```

Однако на то чтобы осознать, что написано в этом условии, придётся потратить гораздо больше сил, чем если бы оно было написано корректно:

```
Если Делитель = 0 Тогда
    Результат = 0;
Иначе
    Результат = Делимое / Делитель;
КонецЕсли;
```

Стандартизация программного кода

Все стандарты разработки, принятые в фирме «1С» можно изучить на портале ИТС.

В каждой команде разработки могут быть приняты свои стандарты, которые, как правило, ужесточают общепринятые.

Правила оформления кода следует учитывать при выполнении домашних работ

Ваши вопросы

Основные правила написания программного кода

Они требуются и при выполнении домашних заданий

2

Основные правила при выполнении домашних заданий

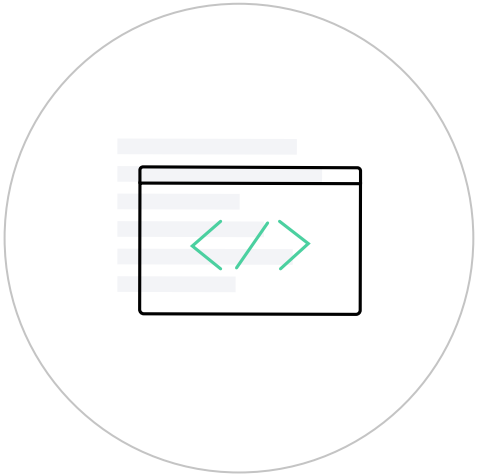
Рекомендуем в первую очередь изучить следующий перечень правил на Github

- Язык модулей
- Комментарии
- Имена переменных
- Имена процедур и функций
- Сложные конструкции
- Условия

Язык модулей

1. Тексты модулей должны быть написаны на **русском языке**. Исключения могут составлять методы SOAP- и HTTP-сервисов, а также идентификаторы внешних информационных систем

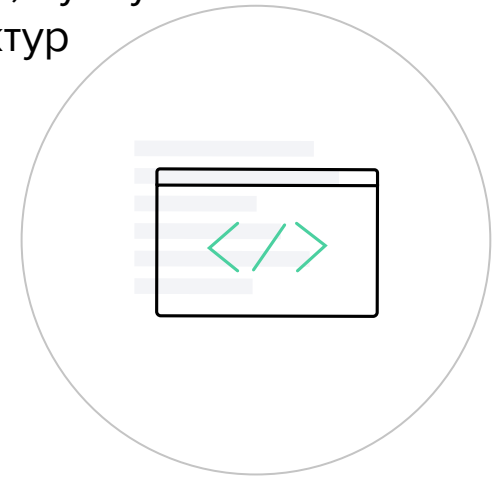
   Хотя допускается двуязычное написание программного кода, следует придерживаться одного языка — того, на котором разрабатывается вся конфигурация



Язык модулей

2 В текстах модулей **не допускается использовать букву «ё»**, за исключением ситуаций, когда она используется в выводимых пользователю сообщениях

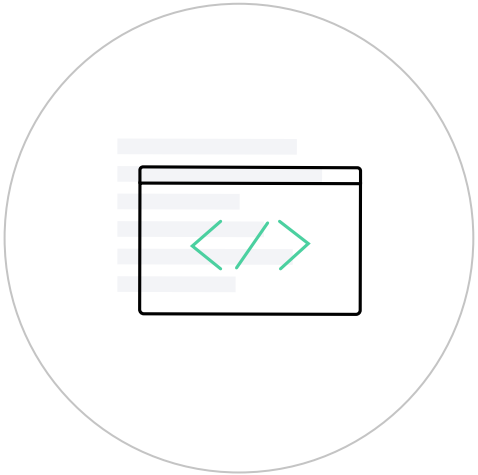
«Е» и «ё» при написании программного кода — это разные символы. Чтобы снизить количество возможных ошибок, букву «ё» не используют в именах переменных и ключах структур



Язык модулей

3 В модулях должны содержаться **только используемые процедуры и функции**, неиспользуемые следует удалять

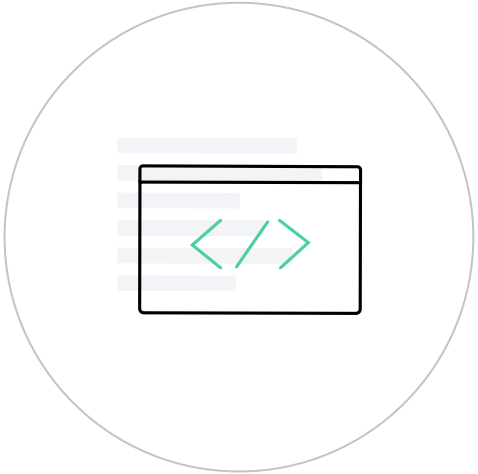
Если вы переписали свой алгоритм так, что процедура или функция больше не используется, лучше удалить её, чтобы код был лаконичнее и понятнее. Также **не рекомендуется оставлять закомментированный код**



Язык модулей

4 Тексты модулей необходимо оформлять по принципу **«Один оператор в одной строке»**

Чтобы упростить читаемость кода, в одной строке должна располагаться только одна команда. Не следует писать несколько действий в одну строку

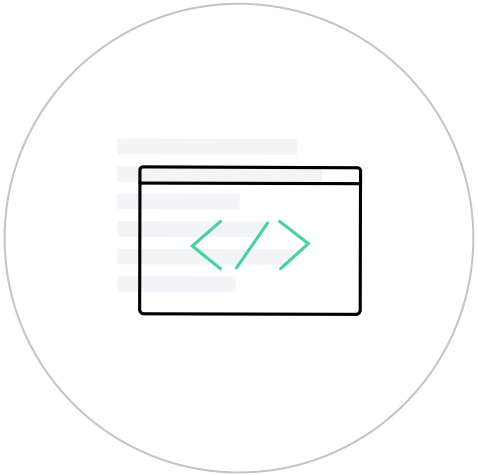


Язык модулей

5 Текст модуля необходимо оформлять синтаксическим отступом. **Для этого используется табуляция**

Соблюдайте разметку модуля так, как предлагает конфигуратор. Тогда за счёт отступов вы сразу будете видеть вероятные ошибки в логике. Кроме того, это упростит для вас чтение программного кода других программистов

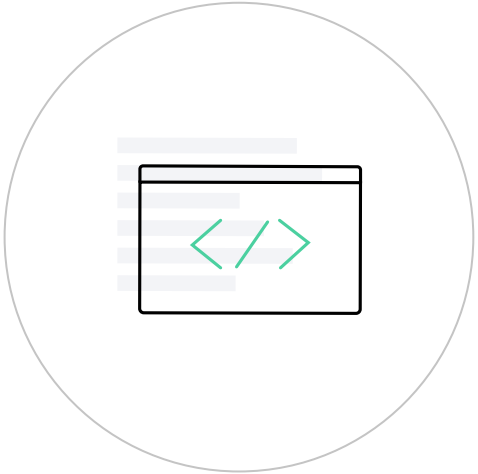
Чтобы отформатировать уже написанный код, его можно выделить и использовать сочетание клавиш **Alt + Shift + F**



Язык модулей

6 Длина строк не должна быть более 120 символов. Исключение — длинные строковые константы, которые должны выводиться без переноса

Это общее правило, которое позволяет просматривать код без горизонтальной прокрутки



Комментарии

В текстах модулей можно оставлять комментарии. Они поясняют работу модуля или комментируемого оператора.

Тексты комментариев должны быть составлены по правилам русского языка, в деловом стиле и содержать только ту информацию, которая относится к функциональности программы.

Между символами комментария «//» и текстом комментария должен быть **пробел**.

В текстах не должно быть модулей закомментированного кода, а также фрагментов, которые связаны с процессом разработки: служебных отметок, TODO, MRG и т. д.

Имена переменных

Имена переменных необходимо образовывать из терминов предметной области, чтобы по имени можно было понять их назначение.

Имена образовываются удалением пробелов между словами, при этом каждое слово пишется с заглавной буквы. Предлоги и местоимения из одной буквы также пишутся заглавными буквами.

Имена переменных не должны начинаться с подчёркивания или состоять из одного символа. Переменные не должны содержать отрицания в имени

Имена процедур и функций

Имена процедур, функций и их параметров следует образовывать от терминов предметной области, чтобы по имени можно было понять их назначение.
Имена должны документировать сами себя

Параметры процедур и функций

Параметры должны идти в логической последовательности. Лучше располагать их от общего к частному.

Необязательные параметры должны располагаться после обязательных.

```
Функция КурсВалютыНаДату(Валюта, Дата = Неопределено) Экспорт
```

Не должно быть более 7 параметров, а в идеале — не более 4.
Если передать в процедуру или функцию большое число параметров всё же необходимо, рекомендуется:

- **группировать** однотипные параметры в один или несколько составных параметров типа «Структура»
- полностью пересмотреть логику работы, **разделить на несколько разных**, более простых процедур и функций

Сложные конструкции

1. Не рекомендуется при передаче параметров в одну функцию применять вложенные вызовы других функций. Лучше разбивать такие вызовы на отдельные строки с помощью вспомогательных локальных переменных

Плохо:

```
СтруктураВложений.Вставить(ПрисоединенныйФайл.Наименование,
    Новый Картинка(ПолучитьИзВременногоХранилища(
        ПрисоединенныеФайлы.ПолучитьДанныеФайла(
            ПрисоединенныйФайл.Ссылка).СсылкаНаДвоичныеДанныеФайла)));
```

Хорошо:

```
ДанныеФайла = ПрисоединенныеФайлы.ПолучитьДанныеФайла(ПрисоединенныйФайл.Ссылка);
АдресФайлаИзображения = ДанныеФайла.СсылкаНаДвоичныеДанныеФайла;
ДанныеИзображения = Новый Картинка(ПолучитьИзВременногоХранилища(АдресФайлаИзображения));
СтруктураВложений.Вставить(ПрисоединенныйФайл.Наименование, ДанныеИзображения);
```

Сложные конструкции

2 Не рекомендуется использовать конструктор структуры с большим количеством свойств (более 2)

Плохо:

```
ПараметрыЗаполнения = Новый Структура("Дата, Организация, Контрагент, Договор",
    ТекущаяДата(), Строка.Организация, Строка.Контрагент, Строка.Договор)
```

Хорошо:

```
ПараметрыЗаполнения = Новый Структура;
ПараметрыЗаполнения.Вставить("Дата", ТекущаяДата());
ПараметрыЗаполнения.Вставить("Организация", Строка.Организация);
ПараметрыЗаполнения.Вставить("Контрагент", Строка.Контрагент);
ПараметрыЗаполнения.Вставить("Договор", Строка.Договор);
```

Условия

1. Вместо тернарного оператора лучше использовать функцию с понятным именем

Плохо:

```
НДС = Сумма * ?(СтавкаНДС = "НДС0", 0, 20);
```

Хорошо:

```
    НДС = Сумма * ПроцентНДС(СтавкаНДС);

...

Функция ПроцентНДС(СтавкаНДС)

    Если СтавкаНДС = "НДС0" Тогда
        Возврат 0;
    Иначе
        Возврат 20;
    КонецЕсли;

КонецФункции
```

Условия

2 Сложные условия, которые содержат 3 конструкции и более, лучше выносить в отдельные методы

Плохо:

```
Если ИдентификаторОбъекта = "АнализСубконто"
    ИЛИ ИдентификаторОбъекта = "АнализСчета"
    ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомость"
    ИЛИ ИдентификаторОбъекта = "ОборотыМеждуСубконто"
    ИЛИ ИдентификаторОбъекта = "ОборотыСчета"
    ИЛИ ИдентификаторОбъекта = "СводныеПроводки"
    ИЛИ ИдентификаторОбъекта = "ГлавнаяКнига"
    ИЛИ ИдентификаторОбъекта = "ШахматнаяВедомость" Тогда
    ПараметрыРасшифровки.Вставить("ОткрытьОбъект", Ложь);

    ЕстьПоказатель   = Ложь;
    ЕстьКорЗначение  = Ложь;
    ЕстьСчет         = Истина;
    Счет             = Неопределено;
    ПервыйЭлемент    = Неопределено;
КонецЕсли;
```

Условия

Хорошо

```
Если ОткрыватьОбъектПриИдентификаторе(ИдентификаторОбъекта) Тогда
    ПараметрыРасшифровки.Вставить("ОткрытьОбъект", Ложь);

    ЕстьПоказатель  = Ложь;
    ЕстьКорЗначение = Ложь;
    ЕстьСчет        = Истина;
    Счет            = Неопределено;
    ПервыйЭлемент   = Неопределено;
КонецЕсли;

Функция ОткрыватьОбъектПриИдентификаторе(ИдентификаторОбъекта)

    Возврат ИдентификаторОбъекта = "АнализСубконто"
        ИЛИ ИдентификаторОбъекта = "АнализСчета"
        ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомость"
        ИЛИ ИдентификаторОбъекта = "ОборотыМеждуСубконто"
        ИЛИ ИдентификаторОбъекта = "ОборотыСчета"
        ИЛИ ИдентификаторОбъекта = "СводныеПроводки"
        ИЛИ ИдентификаторОбъекта = "ГлавнаяКнига"
        ИЛИ ИдентификаторОбъекта = "ШахматнаяВедомость";

КонецФункции
```

Дополнительные материалы

- Красота разработки в 1С, или художественная вёрстка кода
- Принципы для разработки: KISS, DRY, YAGNI, BDUF, SOLID, APO и бритва Оккама

При разработке на 1С не все принципы можно в полной мере применить, но основные тезисы схожи во всех языках программирования

Ваши вопросы

Инструменты для чтения чужого программного кода

3

Как читать чужой код

Часто при работе с приложением вы можете столкнуться с кодом, написанным другим разработчиком. Такой код необходимо научиться читать

Основные инструменты для чтения чужого кода:

1. Точка останова
2. Вычислить выражение и табло
3. Стек вызовов
4. Замер производительности
5. Закладки
6. Переход к определению процедур и функций
7. Синтакс-помощник
8. Подсветка синтаксических конструкций

Точка останова

1 **Точка останова (breakpoint)** останавливает исполнение программы в конкретном месте и позволяет оценить текущее состояние переменных

Горячие клавиши:

- **F5** — продолжить выполнение кода
- **F10** — шагнуть через
- **F11** — шагнуть в
- **Shift + F10** — идти до курсора
- **Shift + F11** — шагнуть из

При необходимости следует использовать точку останова с условием

Вычислить выражение и табло

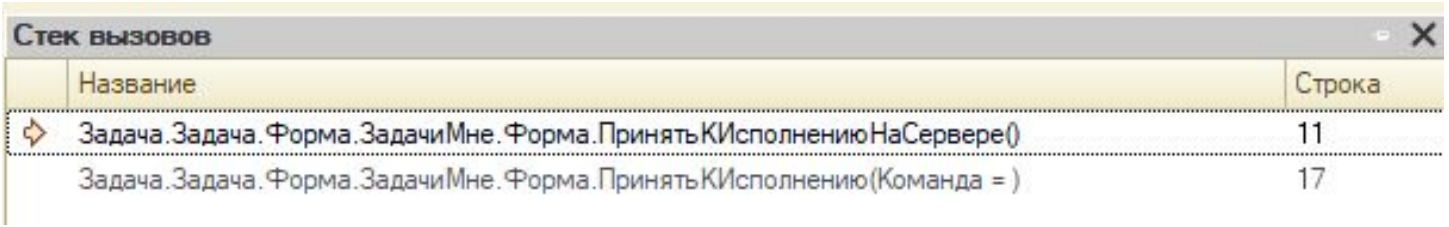
В точке останова мы можем проанализировать переменные
в их текущем состоянии.

2. Для этого можно использовать инструмент **«Вычислить выражение»** и при необходимости поместить переменную в табло для отслеживания её состояния в динамике

   Если же требуется узнать значения всех переменных, можно использовать панель «Локальные переменные»

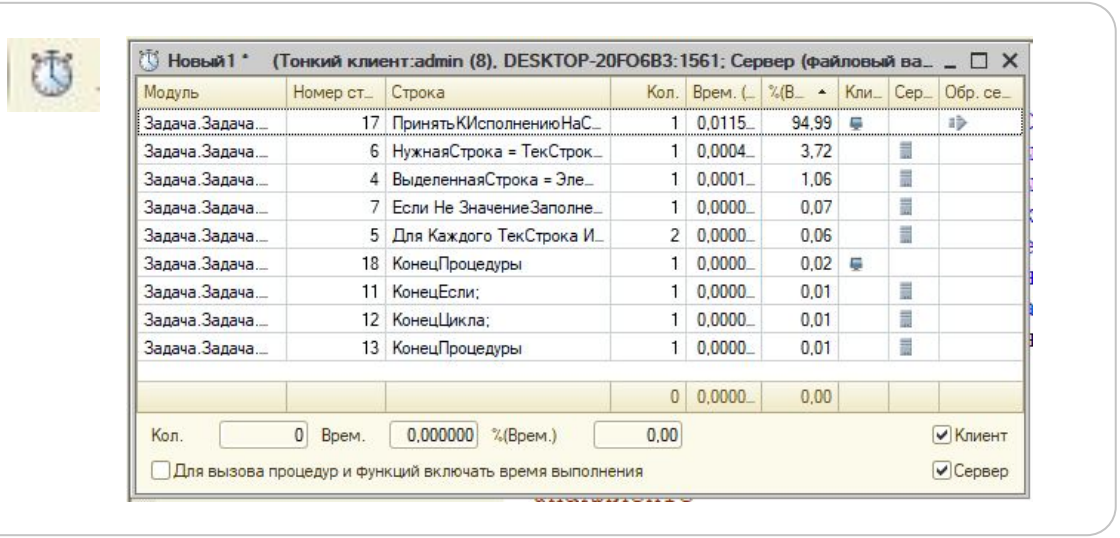
Стек вызовов

3 Если вы точно знаете, что исполнение программы попадает в определённую точку, но необходимо узнать, как исполнение программы попало в эту строку, можно применить стек вызовов. Он покажет цепочку вызова процедур и функций и позволит быстро к ним перейти



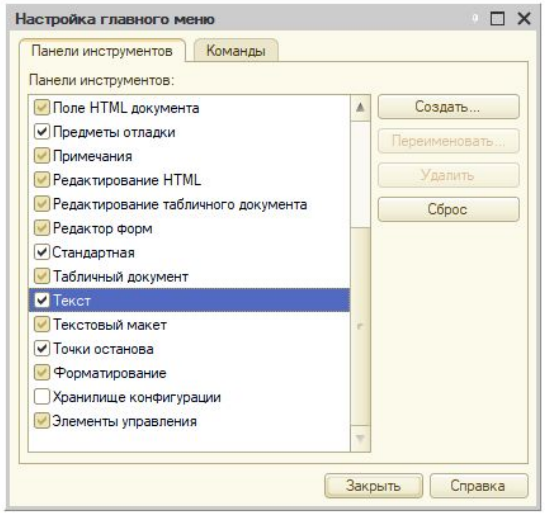
Замер производительности

4 Возможны обратные ситуации, когда вы совсем не понимаете, какой код выполнялся. Здесь на помощь может прийти замер производительности кода. В качестве дополнительного эффекта вы увидите, какие строки программного кода отработали



Закладки

5 Закладки запоминают конкретное место в длинном модуле. При необходимости вы можете быстро к нему перейти
Закладки располагаются в панели работы с текстом, которая по умолчанию может быть отключена. Чтобы включить её, перейдите в «Сервис» -> «Настройка» и установите соответствующий флажок



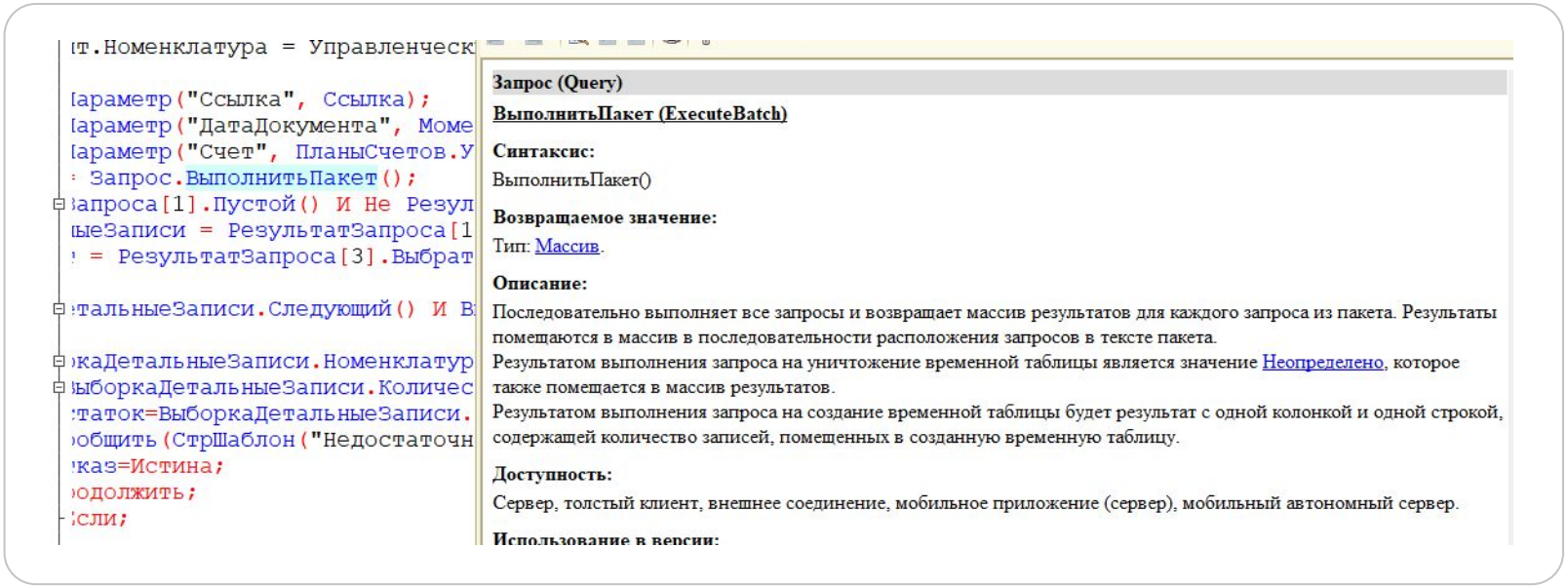
Переход к определению процедур и функций

6 Когда нужно быстро перейти к какой-то процедуре или функции, её можно просто найти в модуле. Но если модуль очень длинный, это может занять много времени.

Для быстрого перехода установите курсор на имя процедуры и нажмите на клавишу **F12**, а для возвращения — комбинацию **Ctrl + –**

Синтакс-помощник

7 Все стандартные типы данных и методы системы описаны в синтакс-помощнике.
Синтакс-помощник — это наиболее полный и актуальный справочник по платформе «1С».
Пожалуй, уступает он только порталу ИТС
Чтобы найти описание платформенного метода, поставьте курсор на него и нажмите **Ctrl + F1**



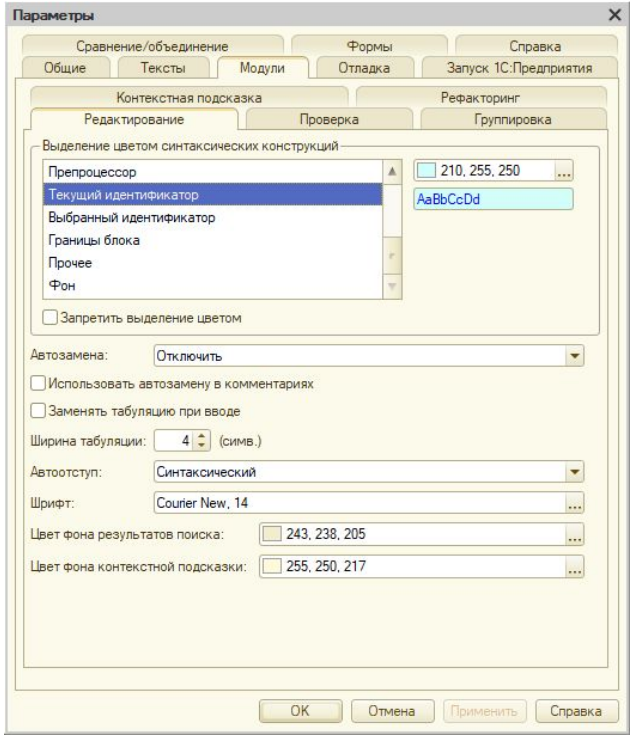
Подсветка синтаксических конструкций

8

Ещё один инструмент, который позволит быстро анализировать программный код, — подсветка синтаксических конструкций.

Чтобы настроить её, перейдите в меню «Сервис» — «Параметры» и переключитесь на вкладку «Модули», а в ней на вкладку «Редактирование». Здесь вы можете настроить внешний вид редактора, в котором работаете.

В первое время отходить от стандартных настроек не рекомендуется, чтобы глаз привыкал к тому, как оформлен код. Но полезно сразу задать цвета для «Текущего идентификатора» и «Выбранного идентификатора»



Дополнительные материалы

Статья от учебного центра фирмы «1С» «Приёмы чтения кода»

Ваши вопросы

Итоги

Сегодня мы:

1. Узнали, какие правила используются 1С-сообществом при разработке программного кода
2. Изучили основные требования, которые предъявляются к коду
3. Разобрали инструменты, предназначенные для анализа чужого программного кода в 1С

Задавайте вопросы и пишите отзыв о лекции