**Практическая работа № 5.**

**Разработка расчетной подсистемы: расчет показателей, перерасчеты**

**Цель работы:** научиться программно рассчитывать результаты начислений в *регистре расчета*, работать с перерасчетами

**Расчет в регистрах расчета**

Предыдущая работа завершилась на том, что мы сформировали движения *по* регистрам накопления и вызвали процедуру общего модуля РассчитатьЗарплату. Эта процедура находится в модуле **РасчетЗарплаты**, ниже приведен ее код:

Процедура РассчитатьЗарплату(Регистратор) Экспорт

//Набор записей регистра расчета ОсновныеНачисления

НаборЗаписейОсн=РегистрыРасчета.ОсновныеНачисления.СоздатьНаборЗаписей();

НаборЗаписейОсн.Отбор.Регистратор.Установить(Регистратор);

НаборЗаписейОсн.Прочитать();

//Набор записей регистра расчета ДополнительныеНачисления

НаборЗаписейДоп

=РегистрыРасчета.ДополнительныеНачисления.СоздатьНаборЗаписей();

НаборЗаписейДоп.Отбор.Регистратор.Установить(Регистратор);

НаборЗаписейДоп.Прочитать();

//Перебор категорий расчета

Для Каждого Категория из Перечисления.КатегорииРасчета Цикл

РассчитатьОсновныеНачисления(Регистратор, НаборЗаписейОсн, Категория);

РассчитатьДополнительныеНачисления(Регистратор, НаборЗаписейДоп, Категория)

КонецЦикла;

Сообщить("Расчет зарплаты по документу "

+Регистратор+" завершен");

КонецПроцедуры

Процедура РассчитатьОсновныеНачисления(Регистратор, НаборЗаписейОсн, Категория)

//Получаем базу

Измерения=Новый Структура;

Измерения.Вставить

("Сотрудник", "ОсновныеНачисления.Сотрудник");

Ресурсы=Новый Массив(2);

Ресурсы[0]="ОсновныеНачисления.Сумма";

Ресурсы[1]="ОсновныеНачисления.ДнейОтработано";

ОтборРегистратор = Новый Структура;

ОтборРегистратор.Вставить("Регистратор", Регистратор);

База=РегистрыРасчета.ОсновныеНачисления.ПолучитьБазу

(ОтборРегистратор, Ресурсы, Измерения);

Для Каждого Запись из НаборЗаписейОсн Цикл

Если Запись.ВидРасчета.КатегорияРасчета = Категория Тогда

//Получаем данные для расчета записей

Индекс=НаборЗаписейОсн.Индекс(Запись);

БазаЗаписи=База[Индекс].Сумма;

МетодРасчета=Запись.ВидРасчета.МетодРасчета;

ДнейФактЗаписи=Запись.ПолучитьДанныеГрафика

(ВидПериодаРегистраРасчета.ФактическийПериодДействия);

//ДнейФакт[Индекс].РабочийДень;

ДнейПланЗаписи=Запись.ПолучитьДанныеГрафика

(ВидПериодаРегистраРасчета.ПериодДействия);

//ДнейПлан[Индекс].РабочийДень;

ДнейБазаЗаписи=База[Индекс].ДнейОтработано;

ИсходныеДанные=Запись.ИсходныеДанные;

ВлияетНаОтработанноеВремя=Запись.ВидРасчета.ВлияетНаОтработанноеВремя;

//Различные алгоритмы расчета, зависящие от метода расчета

Если МетодРасчета=Перечисления.МетодыРасчета.ФиксированнаяСумма Тогда

Запись.Сумма=ИсходныеДанные;

ИначеЕсли МетодРасчета=Перечисления.МетодыРасчета.Процент Тогда

Запись.Сумма=БазаЗаписи\*ИсходныеДанные/100;

ИначеЕсли МетодРасчета=Перечисления.МетодыРасчета.ПоДням Тогда

Запись.Сумма=ДнейФактЗаписи[0].РабочийДень\*ИсходныеДанные;

ИначеЕсли МетодРасчета=Перечисления.МетодыРасчета.ПоСреднему Тогда

Запись.Сумма=БазаЗаписи/ДнейБазаЗаписи\*ДнейФактЗаписи[0].РабочийДень;

ИначеЕсли МетодРасчета=Перечисления.МетодыРасчета.ПоОтработанномуВремени

Тогда Запись.Сумма=ИсходныеДанные\*ДнейФактЗаписи[0].РабочийДень

/ДнейПланЗаписи[0].РабочийДень;

КонецЕсли;

//Записываем отработанное время

Если ВлияетНаОтработанноеВремя Тогда

Запись.ДнейОтработано=ДнейФактЗаписи[0].РабочийДень;

КонецЕсли;

//Если сторно - записываем показатели со знаком "-"

Если Запись.Сторно Тогда

Запись.Сумма=-Запись.Сумма;

Запись.ДнейОтработано=-Запись.ДнейОтработано;

КонецЕсли;

КонецЕсли;

КонецЦикла;

НаборЗаписейОсн.Записать(,,Ложь);

КонецПроцедуры

Процедура РассчитатьДополнительныеНачисления

(Регистратор, НаборЗаписейДоп, Категория)

Измерения=Новый Структура;

Измерения.Вставить("Сотрудник", "ОсновныеНачисления.Сотрудник,

ДополнительныеНачисления.Сотрудник");

Ресурсы=Новый Массив(1);

Ресурсы[0]="ОсновныеНачисления.Сумма, ДополнительныеНачисления.Сумма";

ОтборРегистратор = Новый Структура;

ОтборРегистратор.Вставить("Регистратор", Регистратор);

База=РегистрыРасчета.ДополнительныеНачисления.ПолучитьБазу

(ОтборРегистратор, Ресурсы, Измерения);

Для Каждого Запись из НаборЗаписейДоп Цикл

Если Запись.ВидРасчета.КатегорияРасчета=Категория Тогда

Индекс=НаборЗаписейДоп.Индекс(Запись);

БазаЗаписи=База[Индекс].Сумма;

МетодРасчета=Запись.ВидРасчета.МетодРасчета;

ИсходныеДанные=Запись.ИсходныеДанные;

Если МетодРасчета=Перечисления.МетодыРасчета.ФиксированнаяСумма Тогда

Запись.Сумма=ИсходныеДанные;

ИначеЕсли МетодРасчета=Перечисления.МетодыРасчета.Процент Тогда

Запись.Сумма=БазаЗаписи\*ИсходныеДанные/100;

КонецЕсли;

КонецЕсли;

КонецЦикла;

НаборЗаписейДоп.Записать();

КонецПроцедуры

Рассмотрим этот код. Когда мы завершили формирование (но не расчет) записей *регистров расчета* в модуле документа **НачислениеЗарплаты**, мы вызываем экспортную процедуру РассчитатьЗарплату() из общего модуля **РасчетЗарплаты**. В свою *очередь*, из этой процедуры производится вызов еще двух методов – один – РассчитатьОсновныеНачисления() вызывается для расчета начислений, внесенных в *регистр* **ОсновныеНачисления** документом, из которого была вызвана процедура РассчитатьЗарплату(), вторая – РассчитатьДополнительныеНачисления() – для расчета дополнительных начислений.

В процедуре РассчитатьЗарплату мы создаем копии наборов записей *регистров расчета* **ОсновныеНачисления** и **ДополнительныеНачисления**, отбирая записи *по* параметру Регистратор – то есть – *по* ссылке на документ, из которого была вызвана процедура.

После создания копий необходимых записей, мы открываем цикл перебора перечисления **КатегорииРасчета**. При переборе значения перечисления присваиваются переменной *цикла* Категория в соответствии с порядком их расположения в настройках перечисления в **Конфигураторе**. Таким образом, этот цикл последовательно проходит *по* категориям расчета Первичное, Зависимое первого уровня, Зависимое второго уровня. Смысл этого *цикла* заключается в том, чтобы *виды расчета* с более высоким приоритетом были бы рассчитаны раньше, чем *виды расчета* с более низкими приоритетами. Все дело в том, что, например, *виды расчета*, относящиеся к категории расчета Первичное, могут быть рассчитаны независимо от других *видов расчета*. Их мы рассчитываем в первую *очередь*. *Виды расчета* с категорией Зависимое первого уровня опираются на первичные *виды расчета*, и, в свою *очередь*, Зависимые второго уровня могут опираться на две предыдущие категории.

В цикле мы вызываем процедуры РассчитатьОсновныеНачисления() и РассчитатьДополнительныеНачисления(). При вызове этих процедур мы передаем им схожие наборы параметров. Так, это *параметр* Регистратор – он пригодится нам для дальнейшей работы, это параметры НаборЗаписейОсн и НаборЗаписейДоп – наборы записей регистров, полученные ранее, а так же *параметр* Категория – основываясь на переданной категории расчета мы будем, внутри процедур расчета записей, принимать решения о том, какие записи нужно рассчитать при текущем значении категории расчета.

Рассмотрим метод РассчитатьОсновныеНачисления().

В начале мы получаем параметры базы для расчета начислений, использующих *базовый период* с помощью команды:

База=РегистрыРасчета.ОсновныеНачисления.ПолучитьБазу(ОтборРегистратор, Ресурсы, Измерения);

Перед вызовом данной команды мы формируем структуры данных для получения базы. Во-первых – это структура Измерения, в которой находятся измерения *регистра расчета* для получения базы. Во-вторых – это *массив* Ресурсы – он содержит значения, которые позволяют нам получить, суммовые показатели расчетной базы и показатели количества отработанных дней *по* *базовым видам расчета*. В-третьих – это структура ОтборРегистратор – она позволяет отобрать данные *по* регистратору.

Полученная *переменная* База – это *таблица* значений, содержащая нужные данные *по* *базовым видам расчета*.

Далее, мы открываем цикл перебора записей в полученном ранее наборе записей регистра НаборЗаписейОсн, и, при входе в цикл, проверяем, принадлежит ли *запись* к рассчитываемой при данном вызове метода РассчитатьОсновныеНачисления() категории расчета. Если это так, мы переходим к формированию расчетных показателей, в противном случае переходим к следующей итерации *цикла* перебора записей.

При выполнении условия на проверку категории расчета мы получаем следующие данные:

БазаЗаписи – суммовой показатель *по* расчетной базе. Расчетная база собирается в соответствии с базовым периодом, указанным при вводе записи, и в соответствии с набором *базовых видов расчета*, указанных при настройке *вида расчета*, используемого в записи. Например, если мы рассчитываем Отпуск, базой для которого служит Оклад и *базовый период* для расчета показателя Отпуск равняется трем месяцам, в показатель БазаЗаписи попадут все начисления Оклада за эти три месяца. Если в один из месяцев оклад равнялся 10000 рублей, во втором – 12000, в третьем – 11000, в показателе БазаЗаписи мы получим 10000+12000+11000=33000 рублей.

МетодРасчета – метод расчета, указанный для *вида расчета*, который используется в текущей записи.

ДнейФактЗаписи – фактический период для текущей записи. Например, если мы рассчитываем оклад, с которым связан *график* Пятидневка и в месяце, за который производится оклад, 21 день, причем, за счет вытеснения другими видами расчета (прогулом, например), оказывается, что фактически работник отработал лишь 19 дней, в переменной ДнейФактЗаписи окажется *значение* именно 19 дней.

ДнейПланЗаписи – показатель, характеризующий общее число дней в *периоде действия* записи. В вышеописанном примере сюда попадет именно 21 день.

ДнейБазаЗаписи – в этой переменной окажется количество дней, которое соответствует отработанным дням для *базовых видов расчета*. Например, если *вид расчета* Отпуск рассчитывается за три месяца и базой для этого *вида расчета* является Оклад, причем, в одном из месяцев работник отработал 19 дней, во втором – 11, в третьем – 20 – мы получим в данном показателе 19+11+20=50 отработанных дней.

ИсходныеДанные – это тот показатель, который мы задаем при вводе информации о начислении. Для *вида расчета* Оклад – это сумма оклада, которую сотрудник получил бы, отработав полный месяц, для Оклада *по* дням – это сумма оплаты за один отработанный день.

ВлияетНаОтработанноеВремя – сюда мы записываем *значение* реквизита ВлияетНаОтработанноеВремя *вида расчета*, соответствующего текущей записи. На основе анализа этой переменной мы будем либо увеличивать число отработанных дней, либо не будем этого делать.

Далее следует ряд проверок показателя МетодРасчета на соответствии его значениям перечисления МетодыРасчета. В зависимости от значения этого показателя используются различные методы расчета.

Для метода ФиксированнаяСумма мы, без каких-либо дополнительных изменений, переносим *значение* из свойства ИсходныеДанные в свойство Сумма.

Для метода Процент мы умножаем расчетную базу на *процент*, заданный при вводе записи. Так как проценты мы вводим в виде 10%, 50% и т.д., мы, для получения верного результата, делим ИсходныеДанные на 100.

Для метода По дням мы получаем количество фактически отработанных дней и умножаем полученное число дней на ИсходныеДанные, в которых, в данном случае, хранится сумма оплаты за один день.

Для метода ПоСреднему мы делим суммовой показатель базы (БазаЗаписи) на количество отработанных дней в *базовом периоде* (ДнейБазаЗаписи) и умножаем полученный результат (то есть – размер среднедневной оплаты труда) на количество дней, которое должно быть оплачено *по* начислению, для которого производится расчет.

Для метода ПоОтработанномуВремени мы умножаем ИсходныеДанные на *отношение* фактически отработанного времени к полному времени, которое мог бы отработать сотрудник в периоде, для которого производится расчет.

После завершения расчета в соответствии с методом расчета, мы проверяем, влияет ли рассчитываемая *запись* на количество отработанных дней, если влияет – устанавливаем *параметр* ДнейОтработано в *значение* фактически отработанных дней *по* данной записи.

Далее мы проверяем, является ли обрабатываемая *запись* сторнировочной, то есть – такой, которая отражает в текущем периоде уточнения для записей, выполненных в предыдущем периоде. Если является – инвертируем знак у суммы, которая ранее была записана, и у количества отработанных дней. Если оказалось, что в текущем периоде был введен *вид расчета*, который уменьшает показатели начислений предыдущих периодов (например, в текущем периоде введен Прогул за предыдущий период, вытесняющий Оклад, который уже начислен), сторнировочная *запись* позволит отразить в текущем периоде корректировочный показатель, не затрагивая данных предыдущих периодов.

В итоге мы записываем набор записей командой НаборЗаписейОсн.Записать;

Теперь рассмотрим метод РассчитатьДополнительныеНачисления(). В начале мы получаем расчетую базу (База) такими же методами, как и выше, для записей регистра ОсновныеНачисления. Разница здесь заключается лишь в том, что записи *регистра расчета* ДополнительныеНачисления могут включать расчетную базу, сформированную как в регистре ДополнительныеНачисления, так и в регистре ОсновныеНачисления. Поэтому, создавая структуру отбора *по* измерениям мы указываем измерения двух регистров, создавая *массив* отбора *по* ресурсам мы, так же, указываем два суммовых показателя, на которые нужно ориентироваться системе при формировании расчетной базы.

Далее мы входим в цикл обработки набора записей регистра ДополнительныеНачисления, сверяем категорию расчета текущей записи с категорией расчета, которая передана в процедуру при ее вызове (это, напомним, позволяет нам рассчитать записи в соответствии с порядком их расчета, что позволит правильно получить расчетную базу для зависимых *видов расчета*). Если категория расчета совпадает с переданной, мы переходим к расчету показателей.

Показатели данного регистра не оперируют понятием *периода действия*, они характеризуются датой начисления и суммой (или другим алгоритмом расчета, не учитывающим отработанное время). И в данном регистре мы используем *виды расчета*, которые рассчитываются двумя методами – **процентом** и **фиксированной суммой**. Расчет этих показателей аналогичен вышеописанному.

*Виды расчета* из *плана видов расчета* ДополнительныеНачисления не увеличивают отработанное время, для них мы не формируем сторнировочных записей (*механизмы* вытеснения в данном регистре не используются), в итоге, после расчета показателей мы записываем сформированные записи.

После завершения в процедуре РассчитатьЗарплату *цикла* обхода перечисления КатегорииРасчета мы выводим пользователю сообщение о том, что расчет зарплаты *по* текущему документу завершен.

Нужно отметить, что главная цель вышеприведенного кода – максимально понятная демонстрация достаточно сложных механизмов работы с регистрами расчета. Для повышения быстродействия данного кода его следует проанализировать и вынести как можно больше команд, формирующих запросы к системе (ПолучитьДанныеГрафика, ПолучитьБазу) из циклов, оформив их в виде запросов. Количество запросов так же следует минимизировать. Однако, наш код выполняет возложенные на него функции, поэтому мы не будем заниматься его оптимизацией, перейдем к проверке результатов расчета.

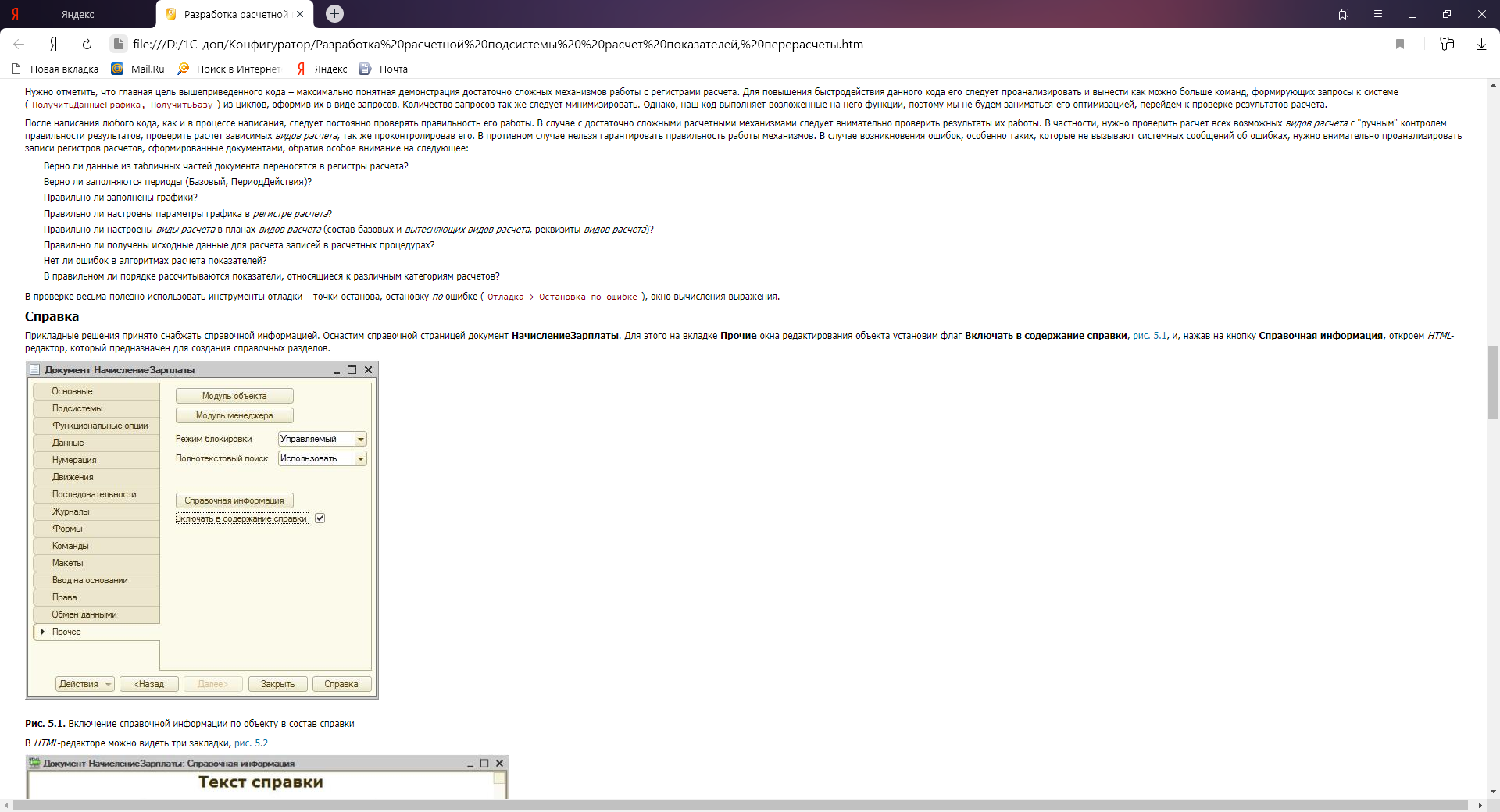
После написания любого кода, как и в процессе написания, следует постоянно проверять правильность его работы. В случае с достаточно сложными расчетными механизмами следует внимательно проверить результаты их работы. В частности, нужно проверить расчет всех возможных *видов расчета* с "ручным" контролем правильности результатов, проверить расчет зависимых *видов расчета*, так же проконтролировав его. В противном случае нельзя гарантировать правильность работы механизмов. В случае возникновения ошибок, особенно таких, которые не вызывают системных сообщений об ошибках, нужно внимательно проанализировать записи регистров расчетов, сформированные документами, обратив особое внимание на следующее:

* Верно ли данные из табличных частей документа переносятся в регистры расчета?
* Верно ли заполняются периоды (Базовый, ПериодДействия)?
* Правильно ли заполнены графики?
* Правильно ли настроены параметры графика в *регистре расчета*?
* Правильно ли настроены *виды расчета* в планах *видов расчета* (состав базовых и *вытесняющих видов расчета*, реквизиты *видов расчета*)?
* Правильно ли получены исходные данные для расчета записей в расчетных процедурах?
* Нет ли ошибок в алгоритмах расчета показателей?
* В правильном ли порядке рассчитываются показатели, относящиеся к различным категориям расчетов?

В проверке весьма полезно использовать инструменты отладки – точки останова, остановку *по* ошибке (Отладка > Остановка по ошибке), окно вычисления выражения.

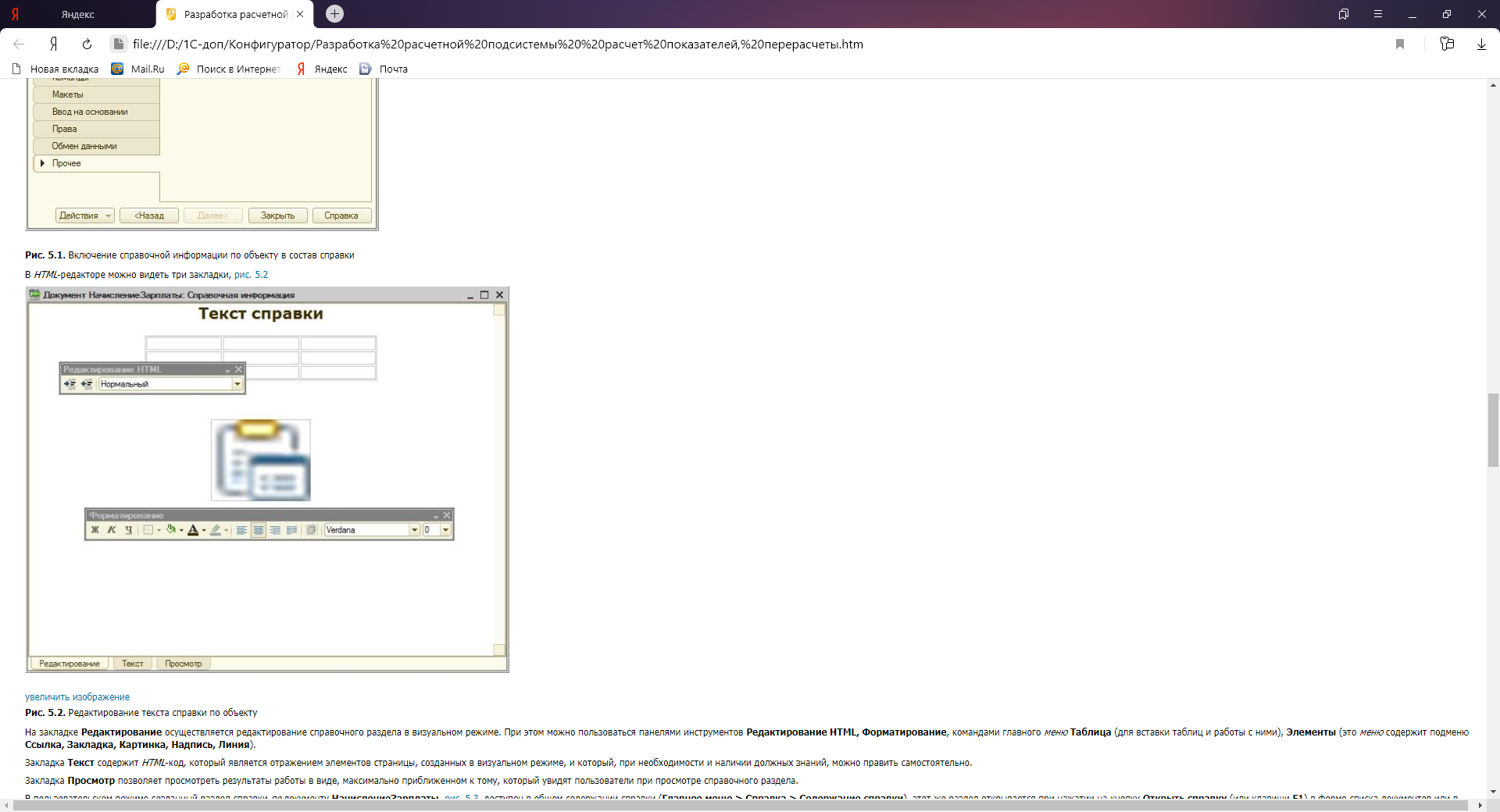
**Справка**

Прикладные решения принято снабжать справочной информацией. Оснастим справочной страницей документ **НачислениеЗарплаты**. Для этого на вкладке **Прочие** окна редактирования объекта установим флаг **Включать в содержание справки**, [рис. 5.1](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=3#image.5.1), и, нажав на кнопку **Справочная информация**, откроем *HTML*-редактор, который предназначен для создания справочных разделов.



**Рис. 5.1.**Включение справочной информации по объекту в состав справки

В *HTML*-редакторе можно видеть три закладки, [рис. 5.2](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=3#image.5.2)

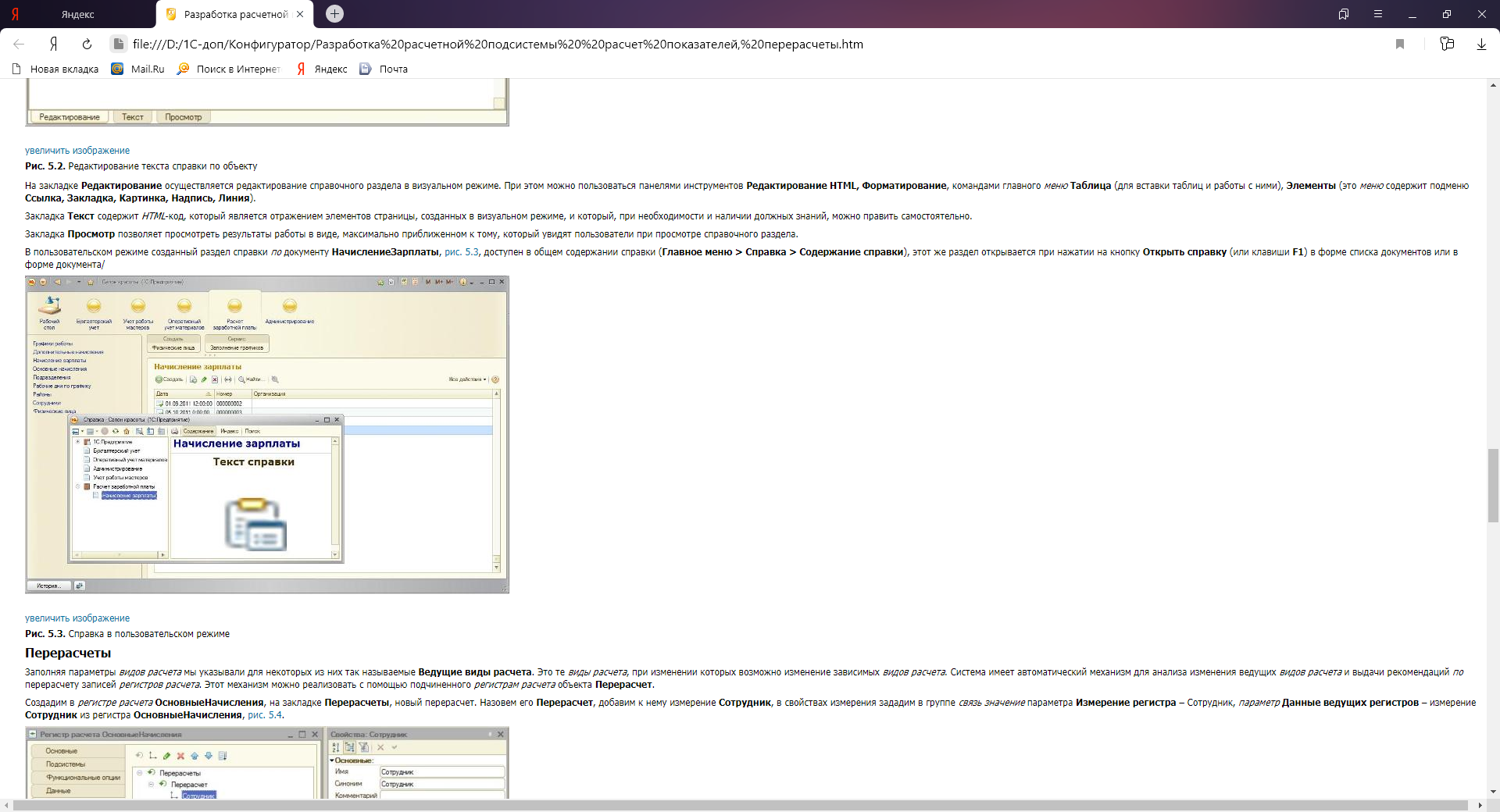
  
  
**Рис. 5.2.**Редактирование текста справки по объекту

На закладке **Редактирование** осуществляется редактирование справочного раздела в визуальном режиме. При этом можно пользоваться панелями инструментов **Редактирование HTML, Форматирование**, командами главного *меню* **Таблица** (для вставки таблиц и работы с ними), **Элементы** (это *меню* содержит подменю **Ссылка, Закладка, Картинка, Надпись, Линия**).

Закладка **Текст** содержит *HTML*-код, который является отражением элементов страницы, созданных в визуальном режиме, и который, при необходимости и наличии должных знаний, можно править самостоятельно.

Закладка **Просмотр** позволяет просмотреть результаты работы в виде, максимально приближенном к тому, который увидят пользователи при просмотре справочного раздела.

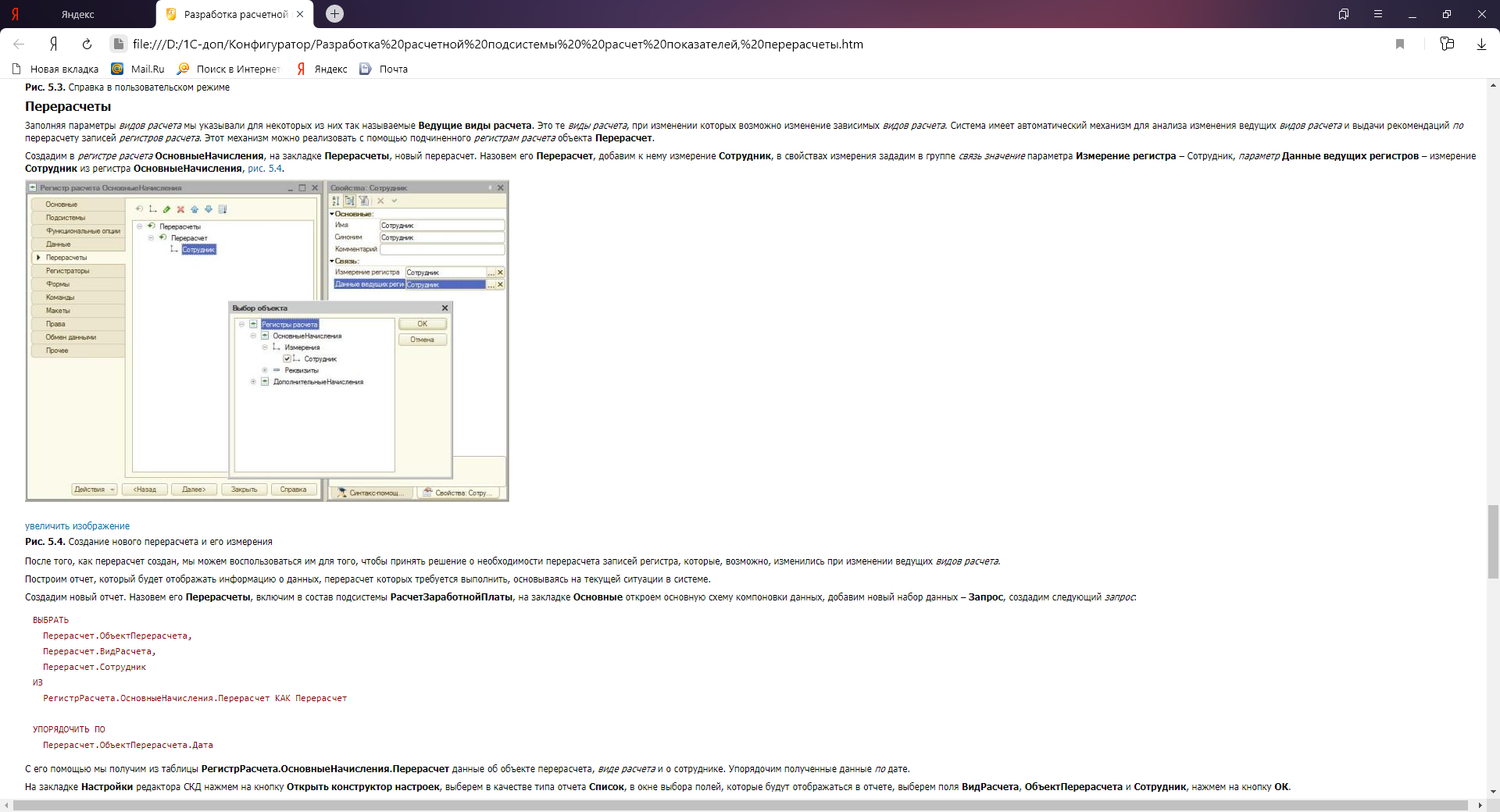
В пользовательском режиме созданный раздел справки *по* документу **НачислениеЗарплаты**, [рис. 5.3](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=3#image.5.3), доступен в общем содержании справки (**Главное меню > Справка > Содержание справки**), этот же раздел открывается при нажатии на кнопку **Открыть справку** (или клавиши **F1**) в форме списка документов или в форме документа/

  
**Рис. 5.3.**Справка в пользовательском режиме

**Перерасчеты**

Заполняя параметры *видов расчета* мы указывали для некоторых из них так называемые **Ведущие виды расчета**. Это те *виды расчета*, при изменении которых возможно изменение зависимых *видов расчета*. Система имеет автоматический механизм для анализа изменения ведущих *видов расчета* и выдачи рекомендаций *по* перерасчету записей *регистров расчета*. Этот механизм можно реализовать с помощью подчиненного *регистрам расчета* объекта **Перерасчет**.

Создадим в *регистре расчета* **ОсновныеНачисления**, на закладке **Перерасчеты**, новый перерасчет. Назовем его **Перерасчет**, добавим к нему измерение **Сотрудник**, в свойствах измерения зададим в группе *связь* *значение* параметра **Измерение регистра** – Сотрудник, *параметр* **Данные ведущих регистров** – измерение **Сотрудник** из регистра **ОсновныеНачисления**, [рис. 5.4](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=3#image.5.4).

  
**Рис. 5.4.**Создание нового перерасчета и его измерения

После того, как перерасчет создан, мы можем воспользоваться им для того, чтобы принять решение о необходимости перерасчета записей регистра, которые, возможно, изменились при изменении ведущих *видов расчета*.

Построим отчет, который будет отображать информацию о данных, перерасчет которых требуется выполнить, основываясь на текущей ситуации в системе.

Создадим новый отчет. Назовем его **Перерасчеты**, включим в состав подсистемы **РасчетЗаработнойПлаты**, на закладке **Основные** откроем основную схему компоновки данных, добавим новый набор данных – **Запрос**, создадим следующий *запрос*:

ВЫБРАТЬ

Перерасчет.ОбъектПерерасчета,

Перерасчет.ВидРасчета,

Перерасчет.Сотрудник

ИЗ

РегистрРасчета.ОсновныеНачисления.Перерасчет КАК Перерасчет

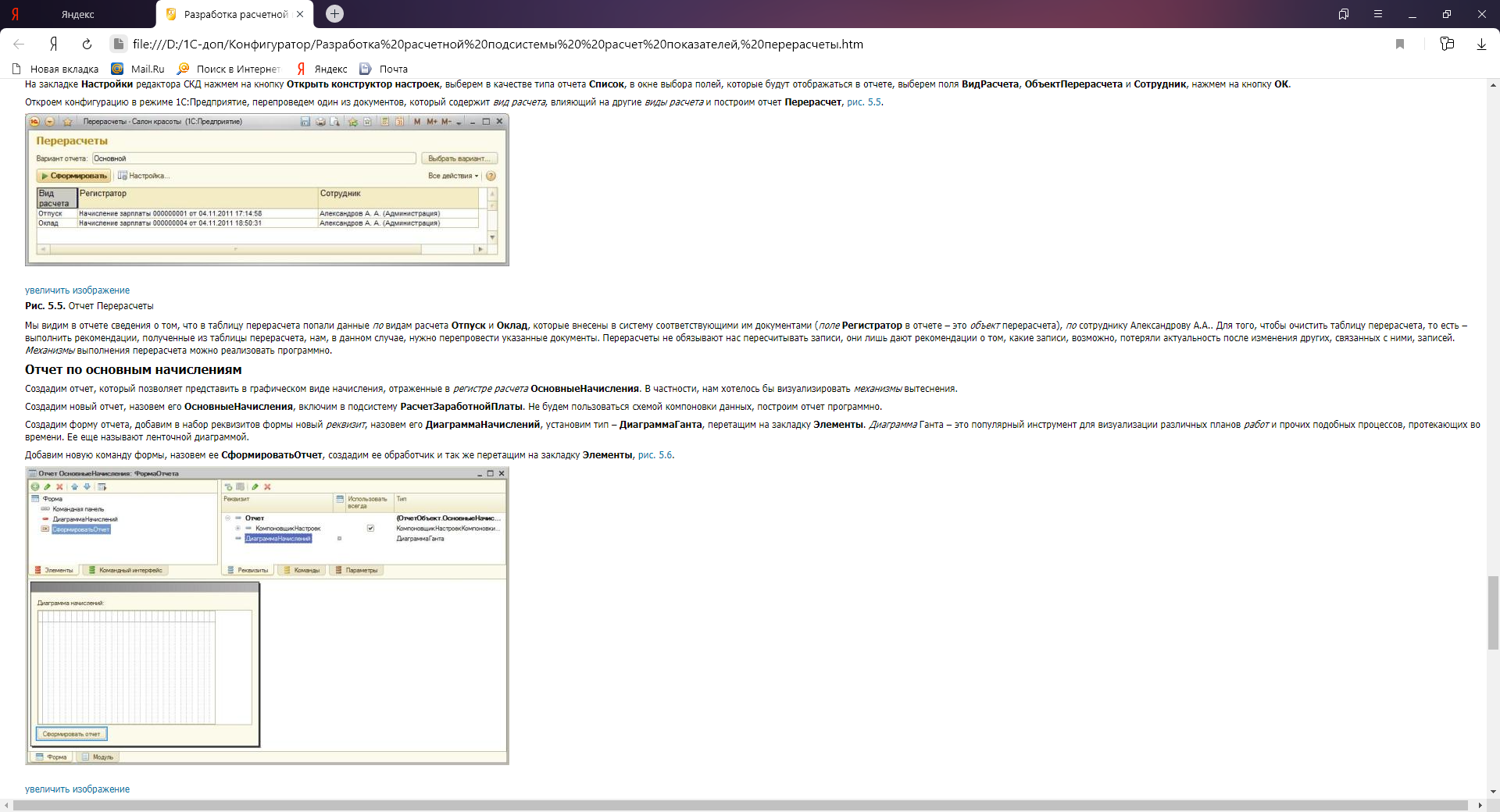
УПОРЯДОЧИТЬ ПО

Перерасчет.ОбъектПерерасчета.Дата

С его помощью мы получим из таблицы **РегистрРасчета.ОсновныеНачисления.Перерасчет** данные об объекте перерасчета, *виде расчета* и о сотруднике. Упорядочим полученные данные *по* дате.

На закладке **Настройки** редактора СКД нажмем на кнопку **Открыть конструктор настроек**, выберем в качестве типа отчета **Список**, в окне выбора полей, которые будут отображаться в отчете, выберем поля **ВидРасчета**, **ОбъектПерерасчета** и **Сотрудник**, нажмем на кнопку **ОК**.

Откроем конфигурацию в режиме 1С:Предприятие, перепроведем один из документов, который содержит *вид расчета*, влияющий на другие *виды расчета* и построим отчет **Перерасчет**, [рис. 5.5](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=3#image.5.5).

  
  
**Рис. 5.5.**Отчет Перерасчеты

Мы видим в отчете сведения о том, что в таблицу перерасчета попали данные *по* видам расчета **Отпуск** и **Оклад**, которые внесены в систему соответствующими им документами (*поле* **Регистратор** в отчете – это *объект* перерасчета), *по* сотруднику Александрову А.А.. Для того, чтобы очистить таблицу перерасчета, то есть – выполнить рекомендации, полученные из таблицы перерасчета, нам, в данном случае, нужно перепровести указанные документы. Перерасчеты не обязывают нас пересчитывать записи, они лишь дают рекомендации о том, какие записи, возможно, потеряли актуальность после изменения других, связанных с ними, записей. *Механизмы* выполнения перерасчета можно реализовать программно.

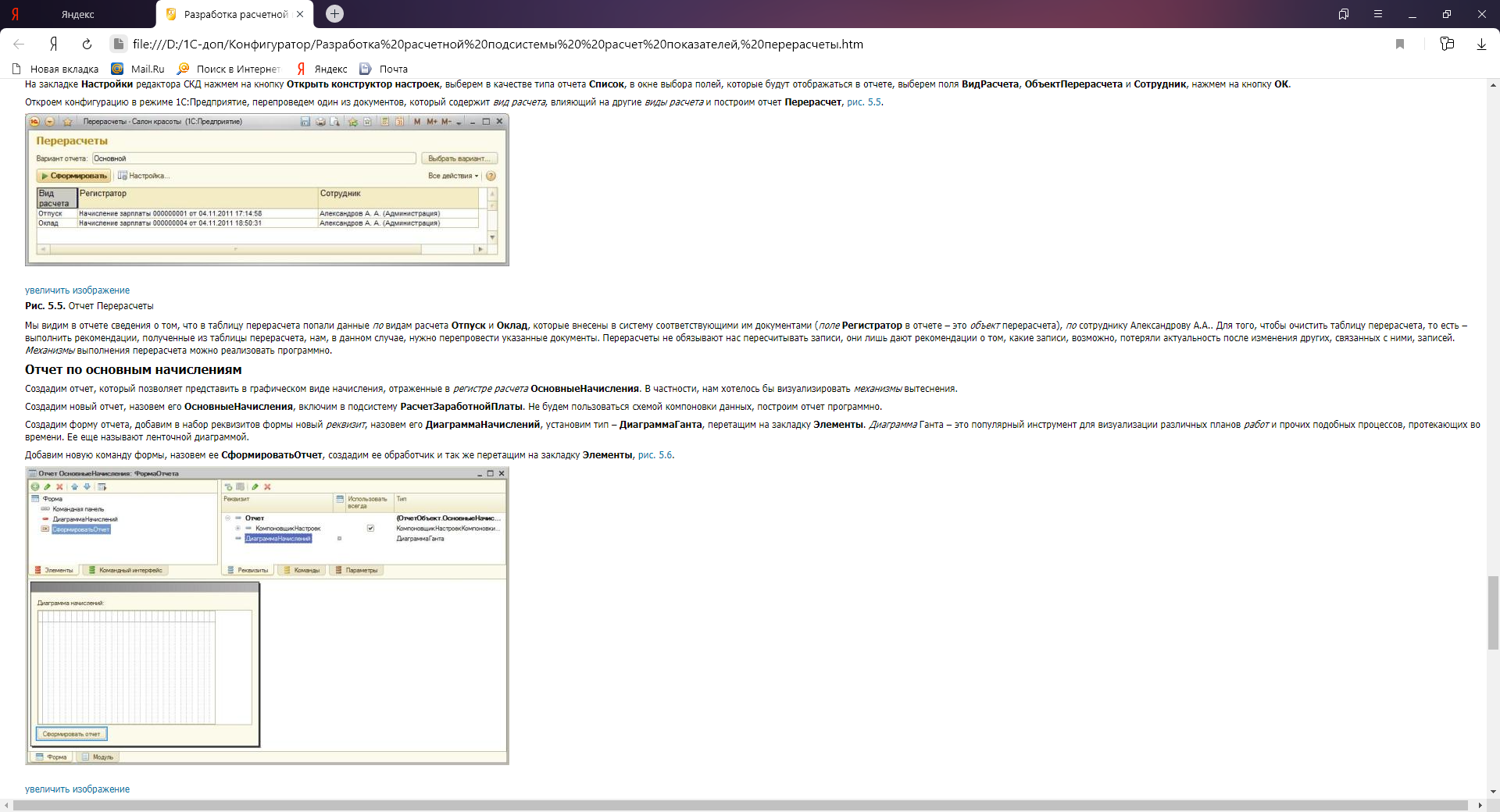
**Отчет по основным начислениям**

Создадим отчет, который позволяет представить в графическом виде начисления, отраженные в *регистре расчета* **ОсновныеНачисления**. В частности, нам хотелось бы визуализировать *механизмы* вытеснения.

Создадим новый отчет, назовем его **ОсновныеНачисления**, включим в подсистему **РасчетЗаработнойПлаты**. Не будем пользоваться схемой компоновки данных, построим отчет программно.

Создадим форму отчета, добавим в набор реквизитов формы новый *реквизит*, назовем его **ДиаграммаНачислений**, установим тип – **ДиаграммаГанта**, перетащим на закладку **Элементы**. *Диаграмма* Ганта – это популярный инструмент для визуализации различных планов *работ* и прочих подобных процессов, протекающих во времени. Ее еще называют ленточной диаграммой.

Добавим новую команду формы, назовем ее **СформироватьОтчет**, создадим ее обработчик и так же перетащим на закладку **Элементы**, [рис. 5.6](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=4#image.5.6).

  
**Рис. 5.6.**Отчет ОсновныеНачисления

Теперь перейдем в *модуль* формы и создадим там следующий код:

&НаКлиенте

Процедура СформироватьОтчет(Команда)

СформироватьНаСервере(ДиаграммаНачислений)

КонецПроцедуры

&НаСервереБезКонтекста

Процедура СформироватьНаСервере(Диаграмма)

Запрос = Новый Запрос;

Запрос.Текст =

"ВЫБРАТЬ

| ОснФакт.Сотрудник,

| ОснФакт.ВидРасчета,

| ОснФакт.Регистратор,

| ОснФакт.ПериодДействияНачало,

| ОснФакт.ПериодДействияКонец

|ИЗ

| РегистрРасчета.ОсновныеНачисления.ФактическийПериодДействия

| КАК ОснФакт";

Результат = Запрос.Выполнить();

ВыборкаДЗ = Результат.Выбрать();

Диаграмма.Обновление = Ложь;

Диаграмма.Очистить();

Пока ВыборкаДЗ.Следующий() Цикл

Точка = Диаграмма.УстановитьТочку(ВыборкаДЗ.Сотрудник);

Серия = Диаграмма.УстановитьСерию(ВыборкаДЗ.ВидРасчета);

Значение = Диаграмма.ПолучитьЗначение(Точка, Серия);

Интервал = Значение.Добавить();

Интервал.Начало = ВыборкаДЗ.ПериодДействияНачало;

Интервал.Конец = ВыборкаДЗ.ПериодДействияКонец;

Интервал.Расшифровка=ВыборкаДЗ.Регистратор;

КонецЦикла;

Диаграмма.Обновление = Истина;

КонецПроцедуры

Здесь мы вызываем из клиентской процедуры обработчика команды СформироватьОтчет серверную внеконтектсную процедуру СформироватьНаСервере, передавая ей диаграмму, которую нужно заполнить.

В серверной процедуре мы, с помощью запроса, получаем сведения о сотруднике, *виде расчета*, о начале и конце *периода действия* *вида расчета* и о регистраторе, сделавшем соответствующую *запись* в регистре. Данные мы получаем из виртуальной таблицы ФактическийПериодДействия *регистра расчета* ОсновныеНачисления.

Далее, мы запрещаем обновление диаграммы для того, чтобы в процессе ее формирования не нагружать систему ненужными вычислениями, и, в цикле перебора полученной из результатов запроса выборки формируем диаграмму.

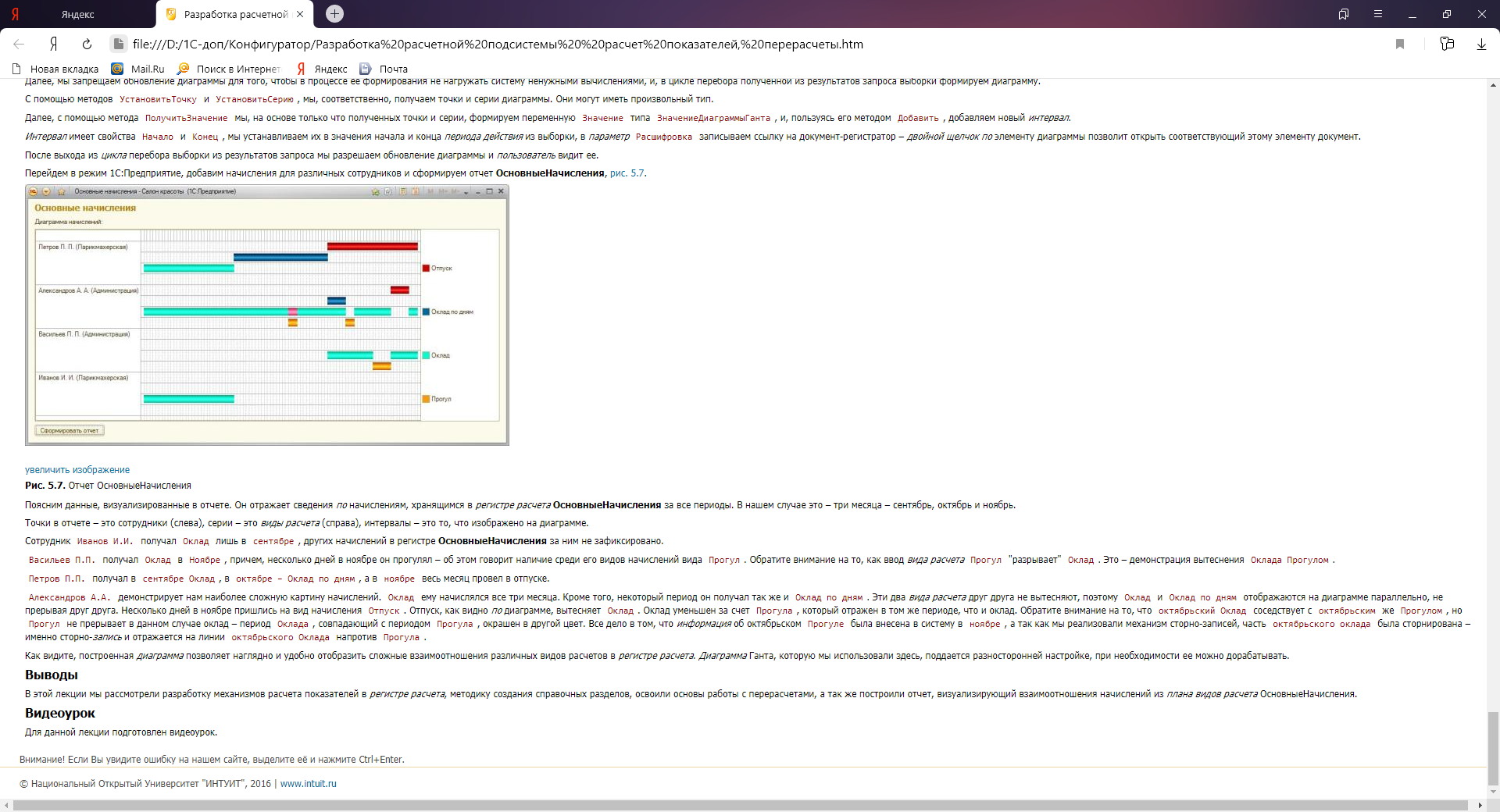
С помощью методов УстановитьТочку и УстановитьСерию, мы, соответственно, получаем точки и серии диаграммы. Они могут иметь произвольный тип.

Далее, с помощью метода ПолучитьЗначение мы, на основе только что полученных точки и серии, формируем переменную Значение типа ЗначениеДиаграммыГанта, и, пользуясь его методом Добавить, добавляем новый *интервал*.

*Интервал* имеет свойства Начало и Конец, мы устанавливаем их в значения начала и конца *периода действия* из выборки, в *параметр* Расшифровка записываем ссылку на документ-регистратор – *двойной щелчок* *по* элементу диаграммы позволит открыть соответствующий этому элементу документ.

После выхода из *цикла* перебора выборки из результатов запроса мы разрешаем обновление диаграммы и *пользователь* видит ее.

Перейдем в режим 1С:Предприятие, добавим начисления для различных сотрудников и сформируем отчет **ОсновныеНачисления**, [рис. 5.7](http://www.intuit.ru/studies/courses/2321/621/lecture/13523?page=4#image.5.7).

  
**Рис. 5.7.**Отчет ОсновныеНачисления

Поясним данные, визуализированные в отчете. Он отражает сведения *по* начислениям, хранящимся в *регистре расчета* **ОсновныеНачисления** за все периоды. В нашем случае это – три месяца – сентябрь, октябрь и ноябрь.

Точки в отчете – это сотрудники (слева), серии – это *виды расчета* (справа), интервалы – это то, что изображено на диаграмме.

Сотрудник Иванов И.И. получал Оклад лишь в сентябре, других начислений в регистре **ОсновныеНачисления** за ним не зафиксировано.

Васильев П.П. получал Оклад в Ноябре, причем, несколько дней в ноябре он прогулял – об этом говорит наличие среди его видов начислений вида Прогул. Обратите внимание на то, как ввод *вида расчета* Прогул "разрывает" Оклад. Это – демонстрация вытеснения Оклада Прогулом.

Петров П.П. получал в сентябре Оклад, в октябре – Оклад по дням, а в ноябре весь месяц провел в отпуске.

Александров А.А. демонстрирует нам наиболее сложную картину начислений. Оклад ему начислялся все три месяца. Кроме того, некоторый период он получал так же и Оклад по дням. Эти два *вида расчета* друг друга не вытесняют, поэтому Оклад и Оклад по дням отображаются на диаграмме параллельно, не прерывая друг друга. Несколько дней в ноябре пришлись на вид начисления Отпуск. Отпуск, как видно *по* диаграмме, вытесняет Оклад. Оклад уменьшен за счет Прогула, который отражен в том же периоде, что и оклад. Обратите внимание на то, что октябрьский Оклад соседствует с октябрьским же Прогулом, но Прогул не прерывает в данном случае оклад – период Оклада, совпадающий с периодом Прогула, окрашен в другой цвет. Все дело в том, что *информация* об октябрьском Прогуле была внесена в систему в ноябре, а так как мы реализовали механизм сторно-записей, часть октябрьского оклада была сторнирована – именно сторно-*запись* и отражается на линии октябрьского Оклада напротив Прогула.

Как видите, построенная *диаграмма* позволяет наглядно и удобно отобразить сложные взаимоотношения различных видов расчетов в *регистре расчета*. *Диаграмма* Ганта, которую мы использовали здесь, поддается разносторонней настройке, при необходимости ее можно дорабатывать.

**Выводы**

В этой работе мы рассмотрели разработку механизмов расчета показателей в *регистре расчета*, методику создания справочных разделов, освоили основы работы с перерасчетами, а так же построили отчет, визуализирующий взаимоотношения начислений из *плана видов расчета* ОсновныеНачисления.