

# Рекомендации для выполнения домашнего задания по курсу «Архитектура компьютеров»

## Общие представления о домашнем задании

В домашнем задании требуется реализовать подпрограмму численного интегрирования некоторой функции. Различаются варианты заданий с: а) вычислением определённого интеграла заданным методом первого порядка (метод средних прямоугольников или метод трапеций), б) решением некоторой задачи динамики точки (метод эйлера, верле или скоростной верле), в) определение длины кривой (замкнутой или отрезка), заданной уравнениями в параметрической форме.

Численное интегрирование предполагает разбиение области интегрирования на отрезки с некоторым шагом, аппроксимацию функции на каждом шаге и суммирование результатов по всем отрезкам. Такой метод интегрирования, естественно, вносит свои погрешности, связанные с точностью подбора аппроксимирующей функции. В домашнем задании используются методы, аппроксимирующие функцию отрезками прямых, погрешность этих методов зависит от второй и более высоких производных подынтегральных функций (т. е. от кривизны функции на каждом отрезке). Предполагается, что в большинстве случаев можно ограничиться постоянным шагом интегрирования и *использовать адаптивный подбор шага не надо*, хотя особого внимания могут требовать кривые с точками возврата.

Должна быть разработана универсальная подпрограмма, способная (интегрировать любую заданную подынтегральную функцию; вычислять длину произвольной кривой, заданной параметрическим виде; определять траекторию движения точки и т.п.) для задаваемого интервала и числа шагов интегрирования:

```
typedef double T; /* в задании указан тип либо double, либо float */

T Integral( T Left, T Right, long N, T (*func)(T) );
/* Left и Right определяют область интегрирования,
   N — ожидаемое число шагов интегрирования,
   T func( T x ); — функция, вычисляющая значение функции в точке x */
```

В задании также указан тип данных, используемый для вещественных чисел (double или float), на первых шагах использование указанного типа обязательно, в последующем возможно использование более сложных типов по согласованию с преподавателем).

Численное интегрирование в общем случае позволяет узнать результат лишь с некоторой точностью. Эта точность зависит от используемого метода интегрирования, качества его реализации, свойств интегрируемой функции, пределов интегрирования и разбиения области. Обычно даже некорректно реализованная программа даёт, тем не менее, некоторый результат, иногда похожий на правильный (а иногда даже совпадающий с правильным, но лишь в некоторых тестах). При разработке программы важно научиться оценивать полученные при тестировании результаты (можно ли их считать подходящим) и составлять набор тестов, «проявляющий» те или иные ошибки и погрешности реализации.

В целом в программе обычно присутствуют погрешности/ошибки нескольких типов:

- Погрешность используемого метода; ожидаемая величина погрешности метода может быть оценена аналитически, формулы для оценки легко найти в интернете; *правильно реализованный метод должен демонстрировать поведение, согласующееся с оценкой погрешности метода*; обычно оценка погрешности метода является весьма пессимистичной и правильно реализованный метод даёт результат несколько лучший оценки до тех пор, пока эта оценка не становится меньше или равна точности представления данных используемого типа.

- ошибки реализации метода; здесь речь идёт не о погрешностях, а именно об ошибках, например, о выходе за заданные границы области интегрирования; такие ошибки должны быть исключены, так как могут привести к фатальным последствиям (например, к попытке вычисления функции вне её области определения);
- погрешности реализации метода; это очень похоже на ошибки реализации, но они не приводят к фатальным последствиям; например, если в результате выполнения заданного числа шагов интегрирования не вторая граница области интегрирования так и не достигнута (скажем, *чуть-чуть* не достигнута); в данном случае можно говорить о неточно реализованном методе, так как тот предполагал покрытие *всей* области интегрирования небольшими интервалами; такие погрешности должны устраняться;
- вычислительные погрешности; в общем виде при использовании чисел с фиксированной или плавающей запятой операции выполняются с некоторой погрешностью; часто эти погрешности являются причиной предыдущего типа погрешностей; избавиться от таких погрешностей в общем случае невозможно, но они должны быть учтены при реализации метода; в самом хорошем случае можно достичь точности, сопоставимой с машинной точностью  $\epsilon$ .

Для устранения погрешностей реализации, обычно, приходится несколько усложнять программу. Как правило, просто «переведённые» формулы из языка математики в выражения в синтаксисе языка программирования содержат значительное число таких погрешностей и их приходится видоизменять для получения более точного результата. Получение подобных навыков также является одной из целей данного домашнего задания.

Задание выполняется в несколько этапов, последовательно, по мере выполнения которых составляется отчёт. Число этапов заранее неизвестно. Сначала разрабатывается *рабочая* программа интегрирования (т. е. не содержащая грубых ошибок реализации), исходный код которой включается в отчёт.

Далее выполняется анализ написанной программы и её тестирование, позволяющие убедиться в её корректной реализации или найти условия, при которых итоговые погрешности вычисления становятся слишком большими. *Критерием может являться соответствие поведения программы оценке погрешности метода*. Для этого надо сравнить полученную погрешность вычисления в наборе тестов с ожидаемым поведением метода, например, зависимость полученной погрешности от числа шагов интегрирования (т. е. выполняется серия тестов, вычисляющая погрешность для различного числа шагов). В принципе, погрешность зависит не только от числа шагов, а от всех возможных факторов, скажем, границ области интегрирования, вида функции и т. п., поэтому может потребоваться вычисление нескольких серий тестов, выявляющих зависимость погрешности от тех или иных параметров.

Интересен вопрос с оценкой погрешности вычислений. Если результатом работы программы является некоторая скалярная величина  $V$  (например, определённый интеграл, длина кривой и т. п.), правильное значение которого  $V_0$  может быть вычислено аналитически (т. е. представлено с машинной точностью  $\epsilon$ ), то тогда целесообразно использовать относительную погрешность  $E$ :

$$E = \frac{V - V_0}{V_0} \quad (1)$$

В таком виде погрешность удобно сопоставлять с машинной точностью. Если  $V_0$  равно нулю, то относительную погрешность так посчитать нельзя, надо использовать либо абсолютную  $V - V_0$ , либо как-либо изменить способ расчёта  $V_0$ , например, использовать вместо него нормированное значение  $V_{\text{норм}}$ , см. (6), либо изменить тест и вычислять, например, вместо определённого интеграла функции определённый интеграл модуля функции и т. п. различные решения удобны в различных случаях. Если же результатом является нечто более сложное, чем скалярная величина (например, траектория), то тогда в качестве погрешности можно использовать величину «промаха» в конце траектории (если траектория имеет началь-

ную и конечную точки) или отклонение от начальной точки в конце периода, если траектория замкнута, или погрешность вычисления в особых точках траектории (апогей, перигей и т. п.), для которых результат может быть несложно аналитически вычислен.

Иногда имеет смысл использовать специальные условия тестов, скажем, подсчитывать интеграл от константной или линейной функции, определять длину кривой такого вида, для которого может быть вычислено точное значение, моделировать какую-либо типичную задачу динамики точки из учебника и т. п. *Во время выполнения работы необходимо обоснованно менять условия задания — область интегрирования, подынтегральную функцию и прочие параметры задачи (но не метод интегрирования).*

Самые разные тесты и различные способы вычисления погрешностей могут быть использованы для того, чтобы сравнить поведение вашей реализации с ожиданиями метода.

Необходимые исходные коды тестов, используемые подынтегральные функции, пределы интегрирования, результаты выполнения тестов и т. п. вместе с обсуждением и анализом результатов тоже включаются в отчёт. По результатам тестов принимается решение либо о модификации кода функции интегрирования, либо о проведении дополнительных тестов.

Анализ полученных результатов и принятые решения *с их обоснованием* включаются также в отчёт. Требуется не только предложить какое-либо изменение в программе, но и *до выполнения новой серии тестов* оценить его влияние на точность вычисления результата, чтобы в дальнейшем можно было проверить соответствие полученных результатов ожиданиям. *Этап заканчивается описанием предлагаемых изменений и предсказанием ожидаемого эффекта в количественном или аналитическом виде.*

*Примечание:* результатом выполнения этапа может быть примерно такое утверждение: если внести такое-то изменение в программу, то в таком-то и таком-то тесте погрешность должна составить столько-то (или получен результат такой-то), а в таком-то тесте, например, мы эффекта изменений не увидим. Причём такое «предсказание» должно быть неочевидным.

Далее наступает следующий этап, на котором реализуются принятые решения, выполняются тесты и сравниваются полученные результаты с ожидаемыми. Если полученный результат подтверждает сделанные ранее предположения, то внесённые изменения в программу принимаются и далее ищем новые пути уменьшения погрешностей.

*Изменения программы, для которых нет подтверждения результатами тестов могут быть ошибочными и не принимаются.* В этом случае необходимо разобраться в причинах: ошибочно сделанная оценка ожидаемого эффекта, неправильная реализация изменений, сами по себе ошибочные изменения и пр., и по результатам предложить либо новые изменения, либо иной способ оценки ожидаемого эффекта.

Процесс выполняется итеративно, пока не будет получена приемлемая функция интегрирования заданным методом, достаточно устойчивая<sup>1</sup> к накоплению погрешностей. *В отчёт входят все выполненные этапы с исходными кодами, тестами, анализом результатов и т. п. Результаты каждого этапа целесообразно обсуждать с преподавателем на консультациях.* Любой результат, даже показывающий ошибочность ранее принятого решения — при условии обнаружения причины ошибки — является шагом в правильном направлении (более того, ошибочный шаг, по результатам которого сделаны правильные выводы, гораздо «ценнее», шага в правильном направлении сразу).

Получение такого навыка нахождения ошибок и является целью работы, с этой точки зрения очень высокая точность достигнутого результата не требуется, достаточно хотя бы одного хорошо проанализированного шага. Дополнительные цели: получение навыков разра-

---

1 Чёткого критерия достаточности не существует; несколько условно можно ориентироваться на точность представления вещественных чисел указанным типом данных: примерно 14..15 десятичных знаков для double и 6..7 десятичных знаков для float. Для большинства функций и пределов интегрирования, задаваемых в условиях домашнего задания, возможно устойчивое достижение точности интегрирования на один-два знака хуже представления (т. е. 12..13 для double и 5..6 для float), но в любом конкретном задании это может быть не так.

ботки программ в вещественных числах, умение учитывать вычислительные погрешности, навыки составления отчётов.

## Справочные сведения о методах численного интегрирования

Ниже приводятся небольшие иллюстрации к указанным методам и формулы оценки погрешности метода. Надо учитывать, что в результате выполнения программы полученный результат содержит погрешности разных типов:

*Погрешность метода*; обусловленная выбранной аппроксимацией подынтегральной функции и шагом интегрирования; эта погрешность часто может быть оценена математически; в домашнем задании функции непрерывны и дифференцируемы в указанной области, что позволяет элементарно оценить величину погрешности в каждом конкретном случае.

*Погрешность реализации метода*; их же можно назвать явными ошибками реализации: например, если число шагов интегрирования окажется не равно заданному, если имеет место вычисление  $x_i$  вне диапазона интегрирования (одна или обе границы диапазона интегрирования могут совпадать с областью определения) и так далее; *эти ошибки являются грубыми и должны быть исключены в программе*; возможно, для этого потребуются проведение специально разработанных тестов;

*Погрешность вычислений*; при использовании вещественных чисел мы всегда имеем дело с приближёнными вычислениями и полученный результат не является точным; программы, в которых выполняется множество операций с вещественными числами, как правило, демонстрируют тенденцию к накоплению погрешностей, способную привести к получению совершенно неправильного результата, либо к появлению фатальных ошибок.

### Метод трапеций

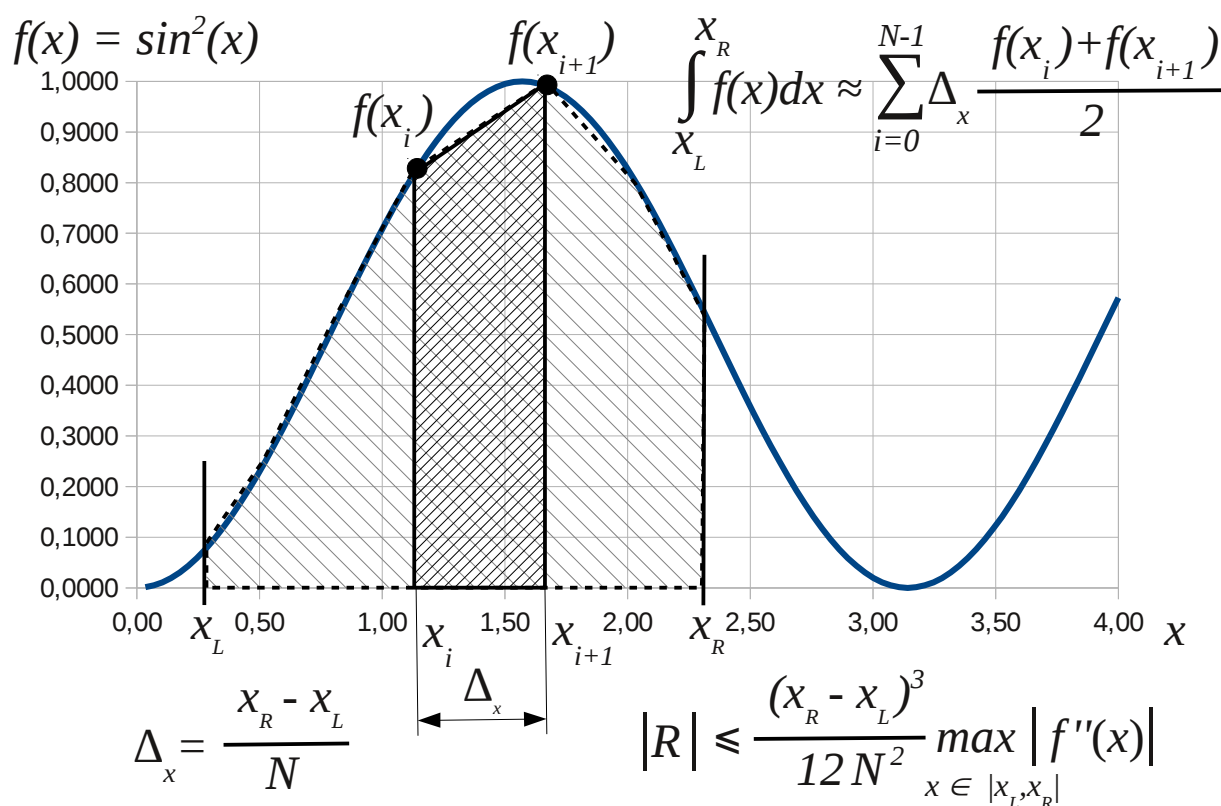


Рисунок 1: Метод трапеций;  $X_L$ ,  $X_R$  - пределы интегрирования;  $N$  - число шагов интегрирования;  $\Delta_x$  - шаг интегрирования;  $R$  - погрешность метода.

## Метод средних прямоугольников

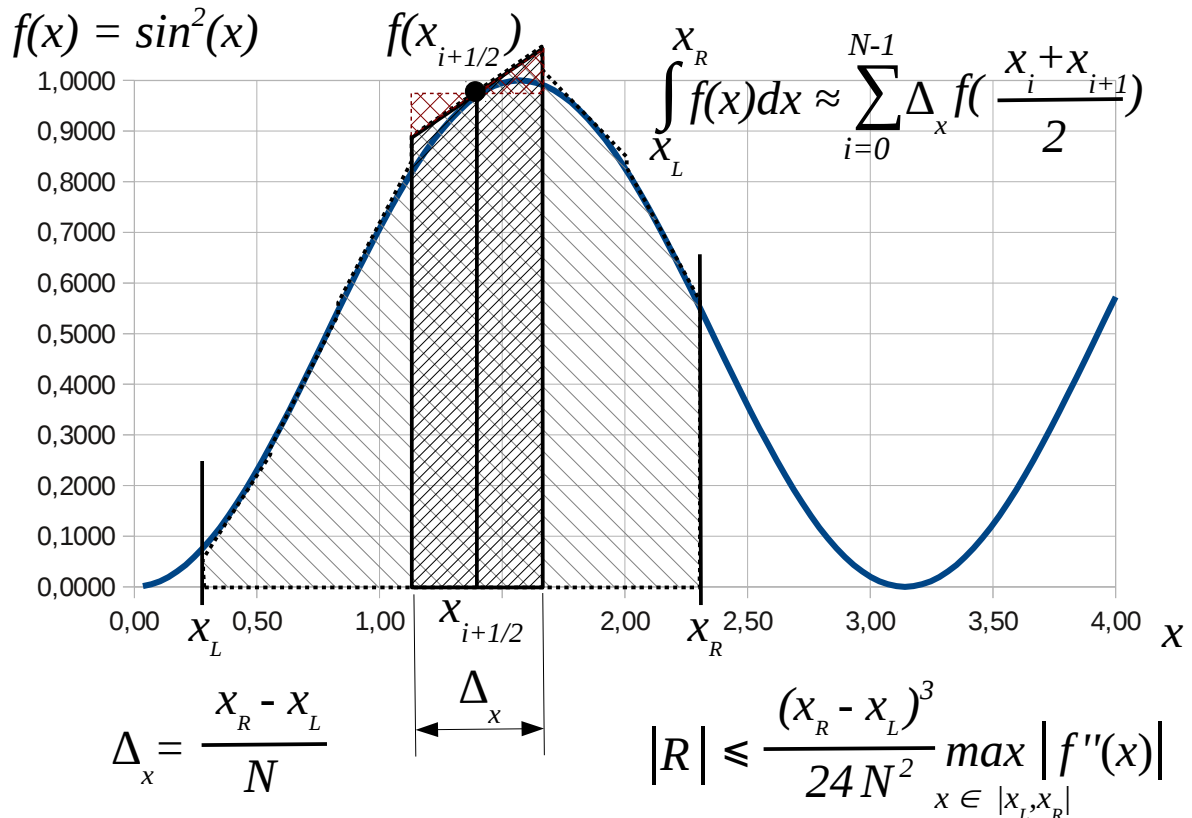


Рисунок 2: Метод средних прямоугольников;  $x_L$ ,  $x_R$  - пределы интегрирования;  $N$  - число шагов интегрирования;  $\Delta_x$  - шаг интегрирования;  $R$  - погрешность метода.

На приведённых выше рисунках (1 и 2) иллюстрируются основные идеи методов численного интегрирования методом трапеций и методом средних прямоугольников. Эти методы относятся к т. н. методам первого порядка, которые учитывают первую производную функции, а погрешность метода определяется второй и более высокими производными. На рисунках приведены аппроксимационные формулы и формулы для оценки ожидаемой погрешности метода  $R$ . При выводе формул для оценки погрешности метода  $R$  предполагается, что ошибка зависит от второй производной (в основном, более высокие производные влияют в меньшей степени), определяющей кривизну функции.

Можно условно считать, что погрешность метода равна суммарной площади всех «горбушек», остающихся между ломаной, аппроксимирующей функцию в методе первого порядка и графиком функции. Для точной оценки погрешности нужно знать значение второй производной на каждом отрезке интегрирования, но, так как в обычных условиях численное интегрирование применяют тогда, когда аналитическое невозможно, то точное значение второй производной неизвестно. Вместо точного значения второй производной используют её максимальное значение на всём интервале интегрирования; обычное это значение если и не известно точно, то его можно хотя бы примерно оценить. Однако, если вторая производная в заданной области интегрирования значительно меняет своё значение, то полученная оценка  $R$  иногда оказывается неадекватно большой. В этом случае вместо максимального значения модуля второй производной можно использовать её среднее значение — формула (2).

$$|R| = \frac{(x_R - x_L)^3}{K \cdot N^2} \text{avg}_{x \in [x_L, x_R]} |f''(x)| \quad (2)$$

$K$  равно 12 или 24 в зависимости от метода интегрирования, а среднее значение второй производной на интервале  $x_L \dots x_R$  вычисляется по формуле (3) ниже. Вообще говоря, все методы интегрирования, использующие аппроксимацию функции отрезками прямых дают

погрешность, зависящую от второй и более высоких производных и их погрешность может быть оценена по этим формулам с точностью до  $K$ . Это справедливо и в других вариантах заданий с задачами динамики или вычисления длины кривой.

$$\text{avg}|f''(x)| = \frac{\int_{x_L}^{x_R} f''(x) dx}{x_R - x_L} = \frac{f'(x_R) - f'(x_L)}{x_R - x_L} \quad (3)$$

Таким образом, уточнённая оценка погрешности может быть вычислена как (4):

$$|R| = \frac{(x_R - x_L)^2}{K \cdot N^2} \cdot |f'(x_R) - f'(x_L)| \quad (4)$$

Стоит заметить, что вычисленная погрешность  $R$  является абсолютной.

Во-первых, погрешность  $|R|$  всегда положительна, хотя разные методы для разных функций дают как завышенные, так и заниженные результаты. Например, метод трапеций для вогнутой (т. е. с положительной второй производной) неотрицательной в области интегрирования функции должен давать завышенный результат, а метод средних прямоугольников в том же случае — заниженный. Знак погрешности стоит самостоятельно приводить к ожидаемому знаку ошибки метода. Если вторая производная или сама функция на выбранном отрезке интегрирования знакопеременная, то для определения стоит разбить область интегрирования на отдельные интервалы и оценить итоговое значение  $R$  для каждого из них.

Во-вторых, если подынтегральная функция имеет нулевую вторую производную (и более высокие), то погрешность метода нулевая — все погрешности, замеченные в итоге, будут только погрешностями реализации или вычисления. С одной стороны это кажется удобным, но всё-таки ограничиваться только линейными функциями не следует, так как некоторые вычислительные ошибки могут проявляться только при наличии второй и более высоких производных; необходимость использования нелинейных функций, равно как и возможность ограничиться только линейными может быть выяснена только при детальном изучении конкретной реализации метода. Иногда даже ничтожные на первый взгляд изменения приводят к заметному изменению поведения функции численного интегрирования.

В-третьих, с точки зрения вычислительных погрешностей можно считать, что для вычислительных систем, удовлетворяющих требованиям стандарта IEEE 754, арифметические операции и вычисление стандартных функций выполняются совершенно точно, но результат каждой операции округляется до ближайшего числа, представимого в заданном формате. Точность представления числа в этом случае определяется числом разрядов в мантиссе (число разрядов в экспоненте определит максимальное и минимальное представимое число). Так, например, для типа float (32-х битовое число с плавающей запятой) длина мантиссы составляет 23 бита и точность представления  $2^{-23} \approx 1.19 \cdot 10^{-7}$ . Важно, однако, понимать, что эта ошибка *относительная*, т. е. абсолютная погрешность представления числа  $x$  (разность между желаемым значением  $x$  и значением  $x^*$ , до которого число будет округлено) будет зависеть от величины этого числа (точнее, от его экспоненты). Можно считать, что (5):

$$x \cdot (1 - \epsilon) < x^* < x \cdot (1 + \epsilon) \quad (5)$$

Здесь  $\epsilon$  обозначает относительную погрешность представления типа ( $2^{-23} \approx 1.19 \cdot 10^{-7}$  для типа float и  $2^{-53} \approx 1.11 \cdot 10^{-16}$  для типа double).

При оценках погрешности функции интегрирования удобнее пользоваться *относительными* погрешностями, а не *абсолютными*, для чего абсолютные погрешности (как результата вычислений, так и оценки погрешности метода) часто можно просто разделить на ожидаемое значение интеграла (оно может быть легко вычислено аналитически). Этот подход, однако, нельзя применять для знакопеременных в области интегрирования функций, особенно если ожидаемое значение интеграла будет равно нулю. В этих случаях можно пользоваться либо

абсолютными погрешностями, либо делить абсолютную погрешность не на итоговое значение интеграла, а на наибольшее по модулю значение, достигаемое в процессе интегрирования ( $V_{\text{norm}}$ ) (6):

$$V_{\text{norm}} = \max_{x \in [x_L, x_R]} \left| \int_{x_L}^x f(x) \cdot dx \right| \quad (6)$$

**Программу рекомендуется оформить примерно таким образом:**

*Листинг 1: Исходный код программы ii-1.c.*

```
#define _USE_MATH_DEFINES      /* для использования констант M_PI и пр. */
#include <math.h>              /* определения констант и мат. функций */
#include <locale.h>            /* прототип setlocale и констант LC_... */
#include <stdio.h>
double f( double x )          /* подынтегральная функция */
{
    return sin(x);
}
double F( double x )          /* первообразная */
{
    return -cos( x );
}
/* функция численного интегрирования */
double Integral( double Left, double Right, long N, double (*func)(double) )
{
    double x, dx, res = 0.0;
    dx = (Right - Left) / N;
    for ( x = Left; x < Right; x+=dx ) res += ... func( ... ) ...;
    res *= dx;
    return res;
}
int main()
{
    long n;
    double L = 0.0, R = M_PI;
    double V, V0 = F( R ) - F( L );                                /* точное решение */
    setlocale( LC_ALL, "" );    /* использовать , или . для отделения дробной части */
    printf("Число шагов;Относительная ошибка\n");
    for ( n = 1; n < 100; n += n/50+1 ) {
        V = Integral( L, R, n, f );                                /* приближенное решение для n шагов */
        printf( "%ld;=%.15G\n", n, (V-V0)/V0 );                    /* n и относительная ошибка */
    }
    return 0;
}
```

В результате её выполнения на консоль выводится два столбца, разделённых точкой с запятой, содержащих в левом столбце число шагов интегрирования и в правом — достигнутую *относительную* ошибку при данном числе шагов.

*Примечание:* ошибку численного интегрирования можно сравнивать с машинной точностью только в случае нормализованных результатов. По сути, машинная точность — это

ошибка представления чисел, близких к 1.0, поэтому и нужно вычислять ошибку, нормированную по точному решению; если точное решение даёт в указанной области интегрирования ноль, что возможно для знакопеременных функций, то, как отмечалось выше, надо либо использовать абсолютную ошибку, а не относительную, либо делить на значение  $V_{\text{ном}}$ , полученное по формуле (6), а не точное значение интеграла (в программе —  $V0$ ).

В данной программе часть кода, реализующего метод средних прямоугольников, не показана; даже и эта часть показана только для того, чтобы продемонстрировать одну из ошибок и то, как эту ошибку надо находить и анализировать, в качестве небольшого примера выполнения самого первого этапа.

### Примерная логика выполнения каждого этапа

Определяем относительную погрешность данной реализации и дополнительно выполняем оценку погрешности метода (см. рис. 4 ниже). То, как технически обработать полученные результаты, получить графики и как оформлять отчёт см. в разделе «Дополнительные рекомендации», начиная со стр. 12.

При вычислении интеграла от функции  $\sin(x)$  на интервале от 0 до  $\pi$  методом средних прямоугольников результат должен быть завышенным по сравнению с аналитическим решением (так как во всей области интегрирования вторая производная не положительна). Так как нам надо убедиться в том, что метод реализован правильно (до того, как начинать поиск возможного накопления погрешностей надо быть уверенным, что программа в целом правильная), то имеет смысл выполнить тест вычисления относительной ошибки для небольшого числа шагов, когда накопление погрешностей ещё не должно себя проявить, а вот ошибки реализации вполне могут проявиться в неадекватно большом отклонении ожидаемого результата от полученного. Более того, для малого числа шагов мы можем легко вычислить ожидаемый результат, который должен быть получен нашим численным методом. Так, например, для функции  $\sin(x)$  на интервале от 0 до  $\pi$  методом средних прямоугольников с единственным шагом интегрирования ( $N = 1$ ) должен быть получен результат, равный  $\pi$  (ширина интервала, умноженная на значение в точке  $\pi/2$ , т. е. на 1). Если наша функция даст для  $N=1$  иной результат, значит есть грубейшая ошибка реализации. Аналогично можно проверить для  $N=2, 3, \dots$ .

*Листинг 2: Результаты запуска программы (исходный код см. лист. 1 стр. 7).*

```
makarov@iu9-makarov:~> gcc -o ii -O3 ii-1.c -lm
```

```
makarov@iu9-makarov:~> ./ii
```

```
Число шагов;Относительная ошибка
```

```
1;=0,570796326794897
```

```
2;=0,110720734539592
```

```
...
```

Для малых значений  $N$  результат правильный, см. лист. 2. Согласно методу для  $N=1$  мы должны были получить результат, равный  $\pi$ , тогда как точное значение равно 2; т. е. относительная ошибка должна быть  $(\pi - 2) / 2 \approx 0,57080\dots$ , что соответствует полученному результату.

Проверять «руками» для большого числа разных  $N$  слишком долго, поэтому можно построить зависимость относительной ошибки от числа шагов интегрирования  $N$  на интервале, например, от 1 до 100 и сравнить полученные данные с ожидаемой оценкой ошибки метода. Мы можем ожидать, что при отсутствии явных ошибок реализации, график относительной ошибки будет в виде некоторой кривой, похожей на график оценки ошибки, но с меньшими значениями. Если на этом интервале числа шагов интегрирования всё хорошо, то тогда имеет смысл проанализировать поведение при большем числе шагов, пока не станут проявляться какие-либо другие ошибки, возможно, скажется накопление погрешностей, а, возможно, обнаружатся более «тонкие» ошибки реализации.



Выполняем тест для  $N=1...100$ , результаты см. рис. 3. В данном тесте явно выделяются отдельные «выбросы», когда функция интегрирования даёт заниженный результат, но по виду данного графика судить сложно; имеет смысл несколько переделать график, чтобы отчётливо проявились ошибки вычислений при малом числе шагов.

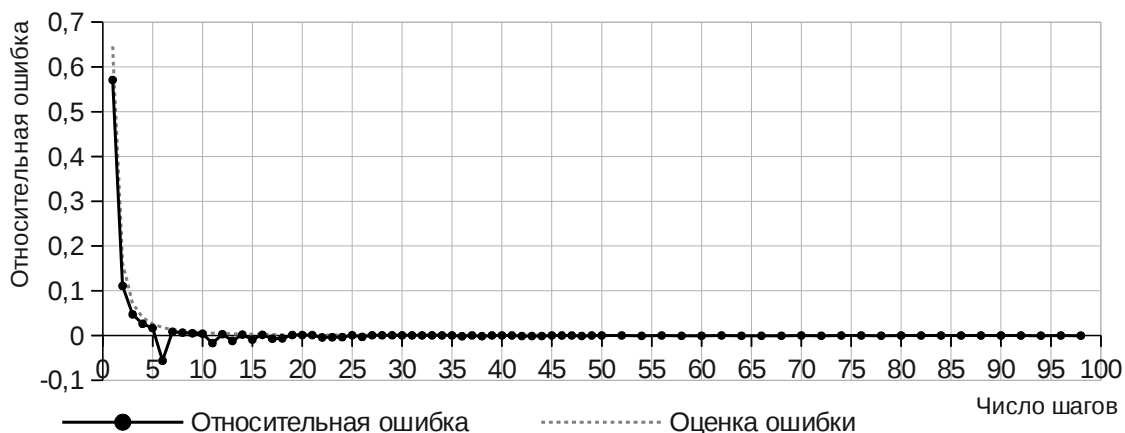


Рисунок 3: Результаты теста для  $N=1...100$ .

Ниже (начиная со стр. 12) рассказано, как получать и вставлять в отчёт графики именно на примере получения диаграммы с графиками относительной ошибки нашей программы и оценки ошибки метода, потребовавшихся на этом этапе анализа результатов.

Итоговый вид диаграммы приведён на рис. 4, который уже достаточно информативен для того, чтобы можно было сделать некоторые выводы.

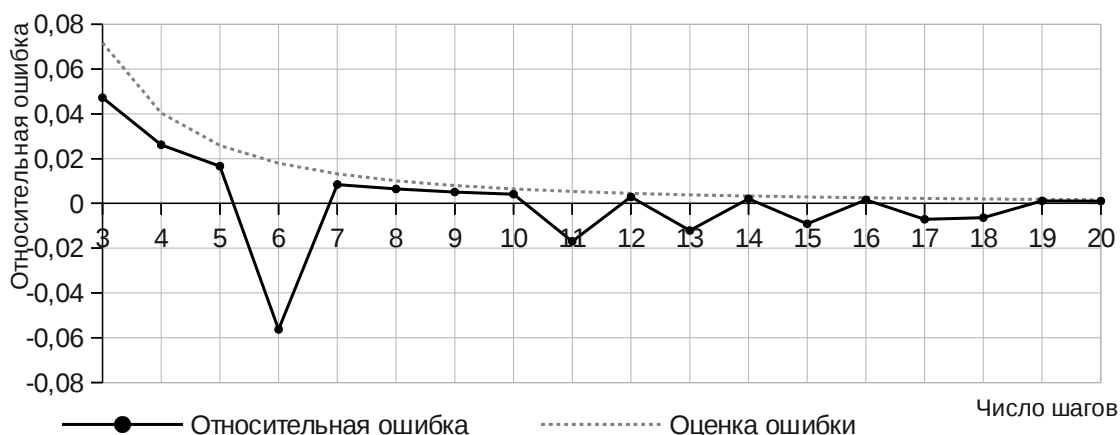


Рисунок 4: Относительная ошибка и оценка погрешности метода в первом варианте функции интегрирования.

Так, например, видно, что а) соблюдается генеральная тенденция с асимптотическим (вероятно, насколько можно судить по видимой части графика) приближением к нулевой относительной ошибке, т. е. нет систематических ошибок; б) для многих точек полученная относительная ошибка имеет правильный знак (для данной функции, метода и интервала интегрирования численное определение интеграла даёт чуть завышенный результат, т. е. положительную относительную ошибку; в) в целом для тех точек, где знак ошибки правильный вычисленная ошибка чуть меньше её оценки, что тоже правильно; г) в некоторых случаях проявляются какие-то ошибки — либо ошибки реализации, либо ошибка накопления погрешностей. Так как они проявляются при очень малом числе шагов, то, скорее всего, их можно считать ошибками реализации. Попробуем проанализировать код функции интегрирования (см. лист. 3), чтобы найти ошибку.

## Попытка анализа

Листинг 3: Функция интегрирования.

```
double Integral( double Left, double Right, long N, double (*func)(double) )
{
    double x, dx, res = 0.0;
    dx = (Right - Left) / N;
    for ( x = Left; x < Right; x+=dx ) res += ... func( ... ) ...;
    res *= dx;
    return res;
}
```

(Пробуем предложить гипотезу) Здесь у нас имеет место вычисление координаты по оси  $x$ , начиная с  $Left$  до  $Right$  с шагом  $dx$ . Так как все операции над вещественными числами выполняются с некоторой точностью, приближённо, то при каждом суммировании  $x += dx$  может появляться некоторая погрешность.

(Любую гипотезу надо проверить) попробуем оценить величину ошибки, которая может возникнуть из-за накопления погрешностей здесь. Возможны два варианта:

а) погрешности в сумме накапливаются в большую сторону, так, что после предпоследнего шага мы оказываемся к правой границе ближе, чем на расстоянии  $dx$ . Тогда во время интегрирования на последнем шаге мы посчитаем ещё некоторый интервал справа от  $Right$ , на величину этой накопленной погрешности. Так как в наших тестах шагов мало, то такая ошибка не может накопиться слишком заметной: для типа `double` относительная точность операций сложения/вычитания примерно  $10^{-16}$ , т. е. даже если предположить строго однонаправленное суммирование всех погрешностей (все «в плюс»), то итоговый промах по оси  $x$  составит не более  $\Delta = 6 \cdot \pi \cdot 10^{-16}$  для  $N=6$ . Такая оценка может быть получена из предположения что абсолютная погрешность каждой операции сложения  $x += dx$  будет менее, чем  $\epsilon \cdot Right$  (у нас во всём интервале  $|x|$  не превысит  $Right$ ). Эту оценку можно уточнить:

$$x_0 = x_L \quad (7)$$

$$x_1 = (x_L + dx)(1+\epsilon)$$

$$x_2 = (x_1 + dx)(1+\epsilon) = ((x_L + dx)(1+\epsilon) + dx)(1+\epsilon) = (x_L + dx)(1+\epsilon)^2 + dx(1+\epsilon)$$

Учитывая, что для  $\epsilon \ll 1$  выполняется (8)

$$(1 \pm \epsilon)^n \approx 1 \pm n \cdot \epsilon \quad (8)$$

Можно упростить последнюю формулу из (7):

$$x_2 \approx (x_L + dx) \cdot (1+2\epsilon) + dx \cdot (1+\epsilon) = x_L \cdot (1+2\epsilon) + dx \cdot (2+3\epsilon) \quad (9)$$

Продолжая по аналогии с (9) получим:

$$x_3 = (x_2 + dx)(1+\epsilon) = (x_L \cdot (1+2\epsilon) + dx \cdot (2+3\epsilon)) \cdot (1+\epsilon) + dx \cdot (1+\epsilon) \approx x_L \cdot (1+3\epsilon) + dx \cdot (3+6\epsilon) \quad (10)$$

Продолжая операции как в (10) будет несложно показать, что для произвольного  $k$  в интервале  $1..N$  выполняется (11):

$$x_k \approx x_L \cdot (1+k \cdot \epsilon) + dx \cdot (k+k \cdot (k+1) \cdot \epsilon) = x_L + k \cdot dx + (x_L \cdot k + dx \cdot k \cdot (k+1)/2) \cdot \epsilon \quad (11)$$

Здесь правое слагаемое является погрешностью вычисления  $x_k$  (12):

$$\Delta = (x_L \cdot k + dx \cdot k \cdot (k+1)/2) \cdot \epsilon \quad (12)$$

У нас  $x_L = Left = 0$ ,  $dx = \pi/6$  и, следовательно, по формуле (12) на правом краю ( $k=N$ ) получаем  $\Delta = \pi/6 \cdot 6 \cdot 7 \cdot 10^{-16} = 7 \cdot \pi \cdot 10^{-16}$ , что очень близко к предыдущей грубой оценке.

Так как функция  $\sin(x)$  в окрестности  $\pi$  имеет первую производную, примерно равную  $-1$  (в зависимости от размеров окрестности), то лишняя посчитанная площадь окажется примерно равной  $\Delta^2/2$  (т. е. порядок  $10^{-30}$ ), что неизмеримо меньше, чем наблюдаемая на практике ошибка (см. рис. 5).

Таким образом, вариант (а) отбрасываем, так как он не может объяснить имеющейся ошибки вычислений для числа шагов  $N=6$  (на рис. 5 величина  $\Delta$  показана для наглядности на много порядков большей, чем на самом деле).

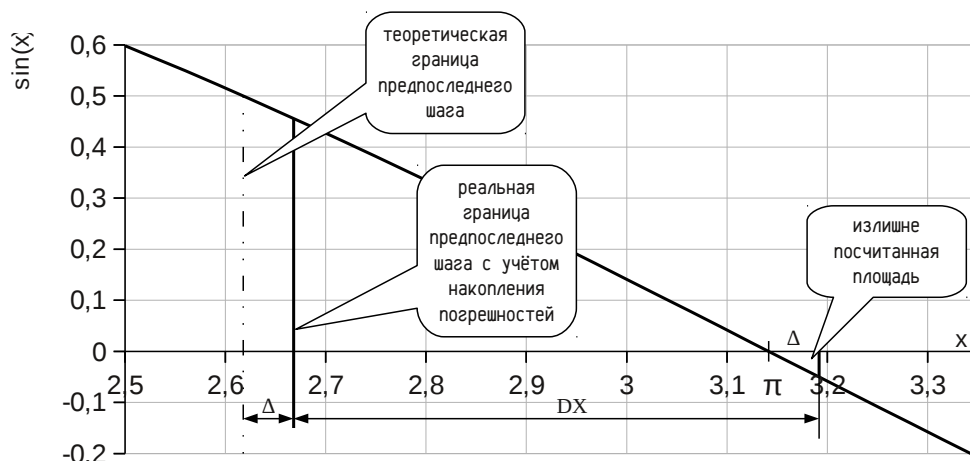


Рисунок 5: Ошибка при накоплении погрешностей "в плюс".

б) погрешности в сумме накапливаются в меньшую сторону, так что предпоследний шаг заканчивается чуть раньше (не более, чем на величину  $\Delta$ , определённую выше). Но тогда последний планируемый шаг закончится тоже примерно на  $\Delta$  не доходя до  $\pi$ , а это значит, что при проверке на завершение цикла условие  $x < \text{Right}$  выполнится и будет сделан ещё один шаг размером  $dx$  сверх ожидаемых  $N$  шагов (см. рис. 6).

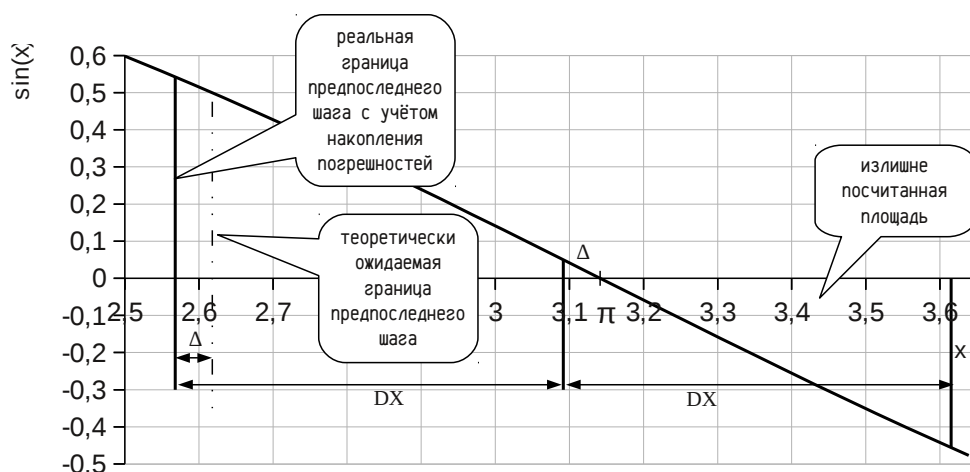


Рисунок 6: Ошибка при накоплении погрешностей "в минус".

Такая ошибка приведёт к ошибке по величине близкой к  $(dx - \Delta)^2/2$ , что, т. к.  $dx \gg \Delta$  примерно равно  $dx^2/2$  или, для  $N=6$ ,  $\pi^2/72 \approx 0,13708$ ; надо учесть, что излишне посчитанная площадь лежит ниже оси абсцисс, т. е. вычисленное с такой ошибкой значение интеграла будет в нашей задаче меньше, чем ожидаемое в соответствии с оценкой метода на величину порядка  $0,13708 \dots$ . Так как эта величина соответствует абсолютной ошибке, то при вычислении относительной мы получим величину  $0,13708/2 = 0,068539$  (т. к. точное значение интеграла равно 2, то нормирование соответствует делению на два). В итоге следует ожидать, что при возникновении такой ошибки при  $N=6$  мы получим результат примерно на  $0,068539$  меньший оценки метода (т. е. около  $-0,050595$ ), тогда как в эксперименте мы получаем относительную ошибку примерно  $-0,0562431$ , что очень похоже. Следует учесть, что при оценке погрешности синусоида была аппроксимирована прямой, а площадь под ней — треугольником, что при больших шагах приводит к заметно неточным вычислениям.

Таким образом, можно принять в качестве рабочей гипотезы, что «выбросы» со значительной ошибкой на графике (см. рис. 4, стр. 9) связаны с накоплением погрешностей «в

минус» при вычислении значения  $x$  для следующего шага:  $x += dx$ . Вероятно, ошибка, связанная с накоплением погрешности в большую сторону, тоже сказывается, но пока её порядок слишком мал, чтобы это увидеть.

...

**ВАЖНО!**

В этом разделе показан пример рассуждений — от попыток проверить работающую программу, до обнаружения факта неточной работы, выдвижение гипотезы о причине этой погрешности, и проверка этой гипотезы.

Далее должно быть сделано (и обосновано) предложение об изменении исходного кода, чтобы эту ошибку устранить. После чего начат второй этап с проверкой и анализом работы исправленной версии программы.

Ключевой момент: выдвигаемые гипотезы должны быть обязательно проверены до того, как начнутся исправления в программе. Просто исправленная программа без этого обоснования (анализ-гипотеза-проверка) не принимается. Основной смысл д.з. именно в подобном осмыслении программы и результатов её работы.

(Рисунки, аналогичные 4 и 5, могут быть представлены в качественно более простой форме, возможно заменены текстом. Всё-таки это д.з. по программированию, а не машинной графике. Однако, для понимания, такие рисунки, хотя бы от руки на бумаге, должны быть; необходимость и вид, в котором их надо вставлять в отчёт надо обсуждать на консультации).

## Дополнительные рекомендации

### Получение результатов на каждом этапе

Имеет смысл включать в имена файлов и результатов номер этапа и, для результатов, номер «версии», так как одинаковый, по сути, код программы может быть скомпилирован и запущен несколько раз с изменениями только в границах интегрирования или числе шагов, или диапазоне изменения числа шагов или в величине приращения числа шагов и т. д., то есть без изменений кода функции интегрирования и без существенных изменений логики теста. Возможно, стоит добавить функцию `main` несколькими строчками, в которых перед циклом выполнения тестов будут выводиться основные сведения об области интегрирования, диапазоне изменения числа шагов и т. п.

Пусть, приведённая выше программа называется `ii-1.c` (см. лист. 1) и выполняет интегрирование методом средних прямоугольников. Умышленно взята примитивная реализация функции численного интегрирования (часть текста функции в листинге заменена многоточием), так что ошибки появляются на самых первых шагах. Компилируем и выполняем эту программу:

*Листинг 4: Компиляция и выполнение программы `ii-1.c`.*

```
makarov@iu9-makarov:~> gcc -o ii -O3 ii-1.c -lm
```

```
makarov@iu9-makarov:~> ./ii
```

Число шагов;Относительная ошибка

1;=0,570796326794897

2;=0,110720734539592

3;=0,0471975511965979

...

94;=-0,000232688926307012

6;=4,46230861568253E-05

98;=-0,000214082632961521

Анализ результатов в числовой форме (лист. 4) возможен и иногда необходим, но зачастую удобнее воспользоваться графическим представлением. Графики можно строить самыми разными способами, включая `gnuplot`, `kalgebra` и т. п., в том числе их можно строить с помощью средств офиса — Microsoft Office разных версий и Libre Office (Open Office). Применение офисных средств и будет рассмотрено ниже. Результаты тестов, графики, их обсуждение надо включать в отчёт; использование «однородных» средств и для написания текста отчёта и для формирования графиков, возможно, имеет смысл.

*Примечание:* жёсткого требования на используемые средства написания отчёта не предъявляется; важно лишь содержание отчёта и соблюдение правил оформления, а не средства, использованные для этого.

Для передачи полученных результатов в офисные средства удобно записать полученные результаты в файл. В листинге 5 ниже приводятся два способа выполнения этой операции.

*Листинг 5: Способы записи результатов, выводимых на стандартный вывод (stdout) в файл. Второй способ позволяет сразу и просматривать результаты и сохранять их в файле.*

```
makarov@iu9-makarov:~> ./ii >ii-1.1.csv
```

```
makarov@iu9-makarov:~> ./ii |tee ii-1.1.csv
```

## Получение графиков

Существует несколько распространённых форматов для экспорта/импорта таблиц с данными в текстовой форме. Основное различие между ними — в способе обозначения границ «столбцов» таблицы. Различают способ с фиксированной шириной каждого столбца и с переменной, когда для отделения одного столбца от другого используется специальный символ-разделитель (пробел, символ табуляции, запятая и т. п.). Один из типичных — т. н. CSV формат (`comma-separated values`), в котором столбцы произвольной ширины и отделяются друг от друга запятой или точкой с запятой.

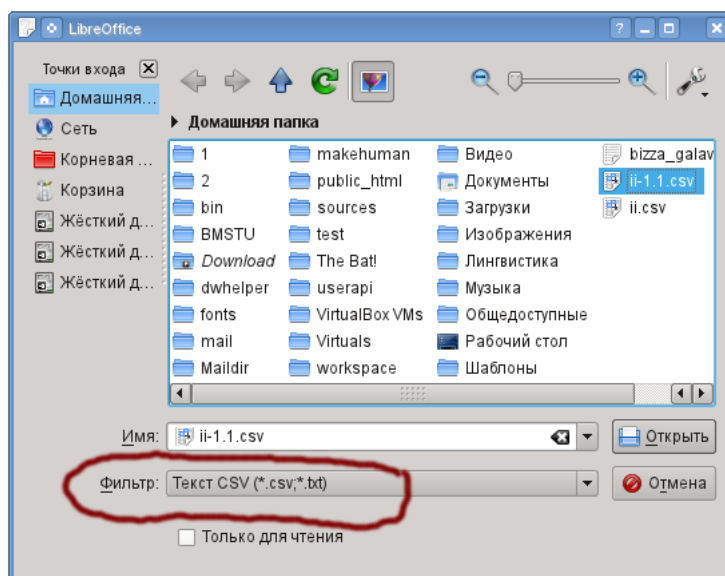


Рисунок 7: Открытие CSV файла.

Офисные средства умеют работать со всеми этими форматами и, обычно, при попытке открытия текстового файла в Microsoft Excel или Libre Office Calc предпринимается попытка автоматического определения формата; если есть «подозрение» на использование подобного формата запускается специальный мастер импорта. Функциональные возможности этих мастеров в разных офисах близки, хотя есть мелкие различия. В дальнейшем будет показан один из способов на примере Libre Office (средства Microsoft Office весьма похожи).

Мастер импорта должен запускаться при любой попытке открытия обычного текстового файла (и имеющего расширение .txt или .csv — последнее обозначает CSV-файл) в Calc (Excel), но если этого не происходит, то имеет смысл в меню выбрать команду «Открыть файл» и в появившемся диалоге явно указать, что вы собираетесь открывать файл CSV (см. рис. 7 выше). Далее появляется диалог мастера (рис. 8):

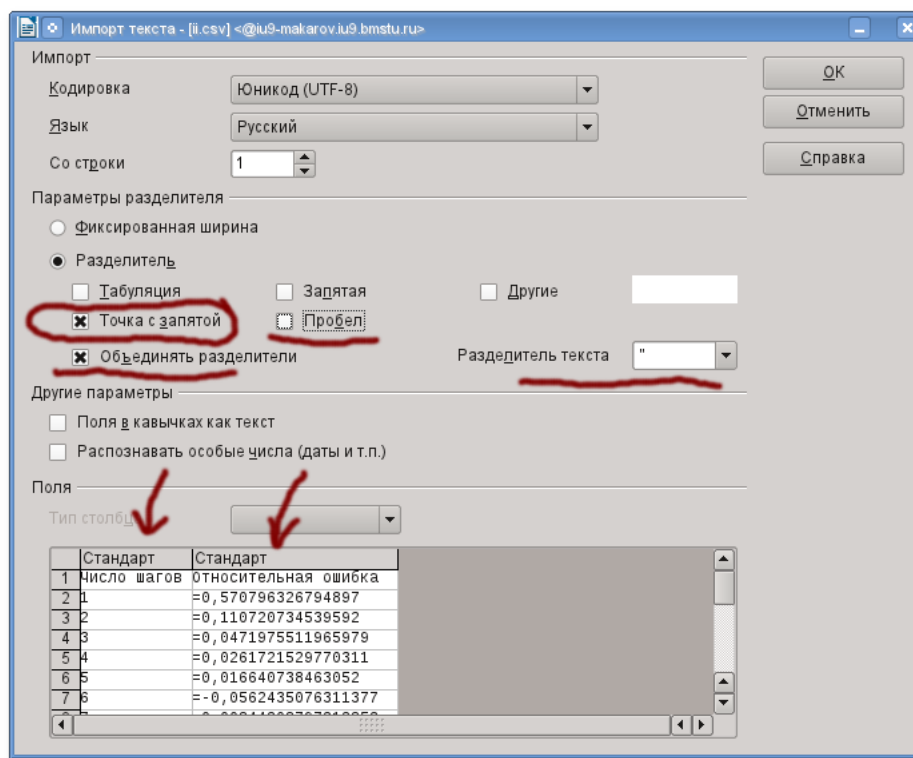


Рисунок 8: Мастер импорта CSV файла.

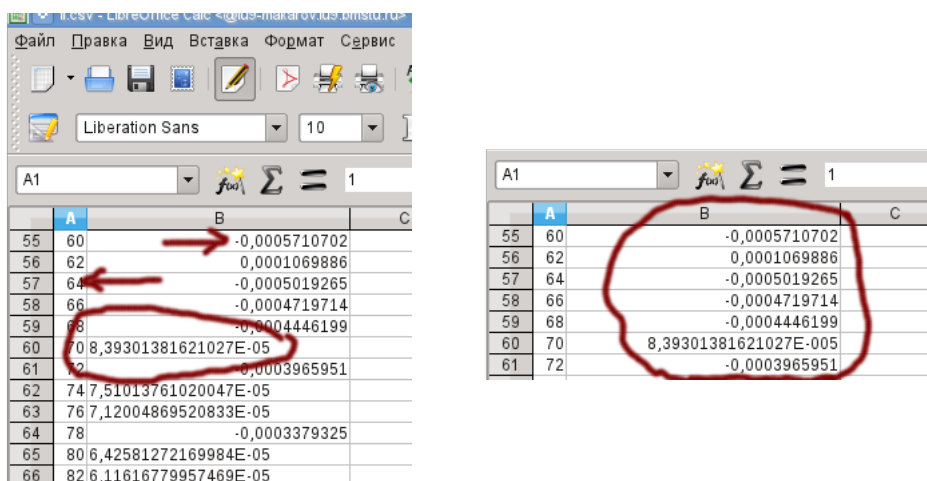
В форматном выражении для функции printf использовался символ «;» для отделения одной колонки от другой. Это целесообразно, так как с символом «,» существует определённая путаница. Дело в том, что в разных странах существуют разные традиции для обозначения дробной части числа, в англоязычных чаще используется точка «.», а вот в русском языке для этой цели используется запятая «,». Офисные продукты используют настройки локализации, указанные в конфигурации операционной системы и, в случае использования в качестве стандартной русской локали они будут ожидать запятую в качестве разделителя между целой и дробной частями вещественного числа. Именно поэтому целесообразно в качестве разделителя в CSV использовать не запятую, а точку с запятой.

*Примечание:* стоит обратить внимание на вызов функции setlocale в main (см. лист. 1 на стр. 7) - она указывает функциям стандартной библиотеки, что надо использовать настройки локали операционной системы — в нашем случае программа на «С» сама будет ставить запятую вместо точки при выводе вещественных чисел.

В нижней части диалога мастера показывается «preview» импортируемого текста с обозначением границ столбцов — по нему удобно ориентироваться в установке опций мастера.

*Примечание:* в нашем случае текст результатов не очень «читаемый», для удобства так и хочется добавить пробелы между столбцами. Это сделать можно, но тогда в мастере стоит указать, что а) пробел также является разделителем, б) смежные разделители, если между ними нет других символов, надо считать одним разделителем и в) при необходимости вывести текст с пробелами (например, названия столбцов «Число шагов», их надо заключать в

кавычки, типа: `printf( "\\Число шагов\\", "\\Относительная ошибка\\\"\\n\" );` и убедиться, что кавычки обозначены в мастере ограничителем текста.



	A	B	C
55	60	-0,0005710702	
56	62	0,0001069886	
57	64	-0,0005019265	
58	66	-0,0004719714	
59	68	-0,0004446199	
60	70	8,39301381621027E-05	
61	72	-0,0003965951	
62	74	7,51013761020047E-05	
63	76	7,12004869520833E-05	
64	78	-0,0003379325	
65	80	6,42581272169984E-05	
66	82	6,11616779957469E-05	

Рисунок 9: Ошибка распознавания чисел в экспоненциальной форме(слева) и правильное распознавание (справа).

После импорта следует убедиться, что числа (вещественные, как с указанием порядка, так и без) были определены мастером правильно. Для этого существует один простой и удобный признак: по умолчанию числа в таблицах выравниваются Calc'ом и Excel'ем по правой границе, а текст — по левой. Если увеличить ширину столбца, то все числа, которые были ошибочно распознаны как текст, окажутся выровнены по левой границе, а все «правильные» числа — по правой, что показано на рис. 9. Будет аналогично, если в дробных числах использован «неправильный» с точки зрения офиса, разделитель: точка вместо запятой или запятая вместо точки, смотря по настройкам системы и офиса.

Логика распознавания чисел в разных офисах чуть различается; в Libre Office числа с порядком, т. е. с буквой «Е» внутри, распознаются как обычный текст, а не число. Однако для Libre Office можно сделать хитрость: поставить (как сделано в лист. 1 на стр. 7) знак присваивания «=» перед вещественным числом. Со знака присваивания в офисах начинаются формулы, так что такую запись Libre Office поймёт как операцию вычисления значения того, что находится после присваивания и записи этого в ячейку таблицы; результат будет правильный. Пример на рис. 9 был получен для форматной строки `"%ld;%.14G\n"` при выводе результатов теста (см. лист. 1 стр. 7). К ошибке определения числа в Libre Office также приводит наличие пробелов перед числом. Именно поэтому пробелы, если вы их вставляете между столбцами для лучшей читаемости, надо объявлять разделителями в мастере (либо в строке с числами пробелы не должны встречаться).

Мастер в Microsoft Office относится к пробелам в начале ячейки и к знакам порядка в числе куда «спокойнее», так что мастер успешно разбирает форматы в большем числе случаев. Правда, он «негативно» реагирует на наличие символа присваивания перед числом — он присвоит ячейке не число, а текст, начинающийся со знака «=» и продолжающийся числом. В целом имеется некоторая несовместимость мастеров (для надёжного распознавания чисел в Libre Office нужно «=», а в Microsoft Office его быть не должно), но приспособиться к конкретному офису совсем несложно.

Если все числа во всех нужных столбцах распознаны правильно, то можно перейти к созданию графика. Для этого стоит выделить всю область чисел (удобно указать «мышкой» в верхнюю левую ячейку и нажать `Ctrl+Shift+Стрелка_Вниз` затем, не отпуская `Ctrl+Shift`, нажать `Стрелка_Вправо`) и выбрать в меню пункт «Вставка | Диаграмма...», как показано на рис. 10 ниже.



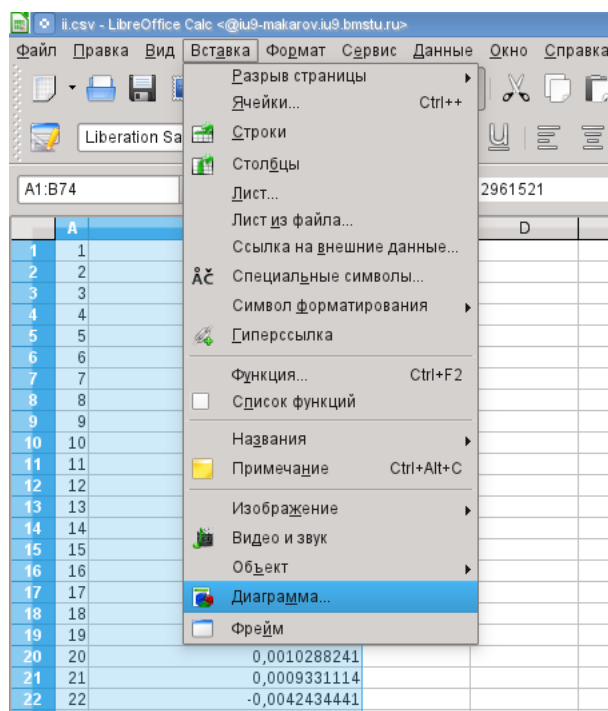


Рисунок 10: Вставка диаграммы.

При этом запускается мастер диаграмм, с помощью которого нужно правильно указать основные характеристики диаграммы и, в дальнейшем, отредактировать её представление. В большинстве случаев мастера создают диаграмму, удобную для презентаций и демонстрации заказчикам, но не для научных работ и публикаций. Созданную мастером диаграмму надо будет потом отредактировать, чтобы привести к некоторому приемлемому виду.

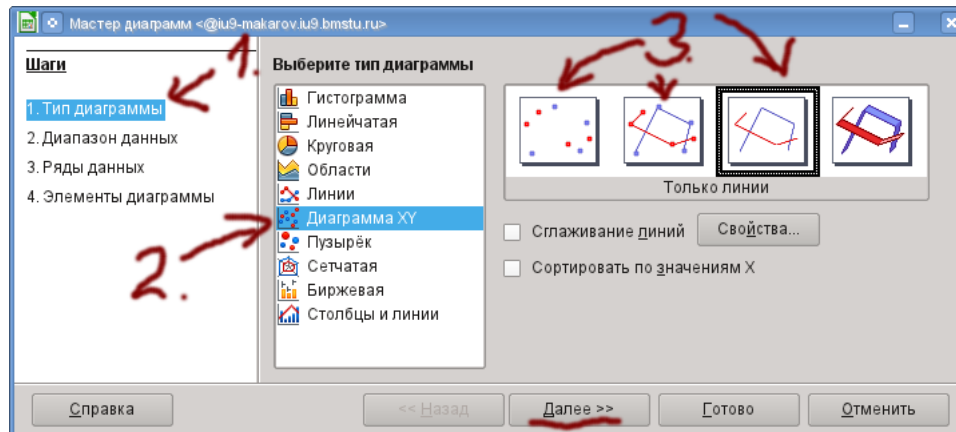


Рисунок 11: Выбор типа диаграммы.

На первом шаге мастера (см. рис. 11 на стр. 16) нужно выбрать тип диаграммы. В нашем случае нужна т. н. «XY» диаграмма, в которой для каждой точки графика задаётся пара чисел: абсцисса и ордината. Такой вид графика позволяет точно воспроизводить внешний вид зависимости при нерегулярном шаге точек по оси абсцисс. Среди множества подвидов «XY» диаграмм надо использовать один из трёх возможных вариантов: только точки; точки и соединяющие их ломанные или ломаная линия без обозначения точек. Важно учитывать, что по результатам численного эксперимента мы получаем только лишь множество точек и у нас обычно нет никаких достоверных оснований, чтобы мы могли предполагать вид линии между этими точками. Поэтому любые методы сглаживания запрещены категорически (сглаженная линия будет показывать наличие экстремумов между реально полученными точками, что просто-напросто является необоснованной фантазией), впрочем, ломанную линию тоже можно рассматривать лишь в качестве некоторой зрительной подсказки, облегчающей нахождение экспериментальных точек, но не как действительную зависимость.



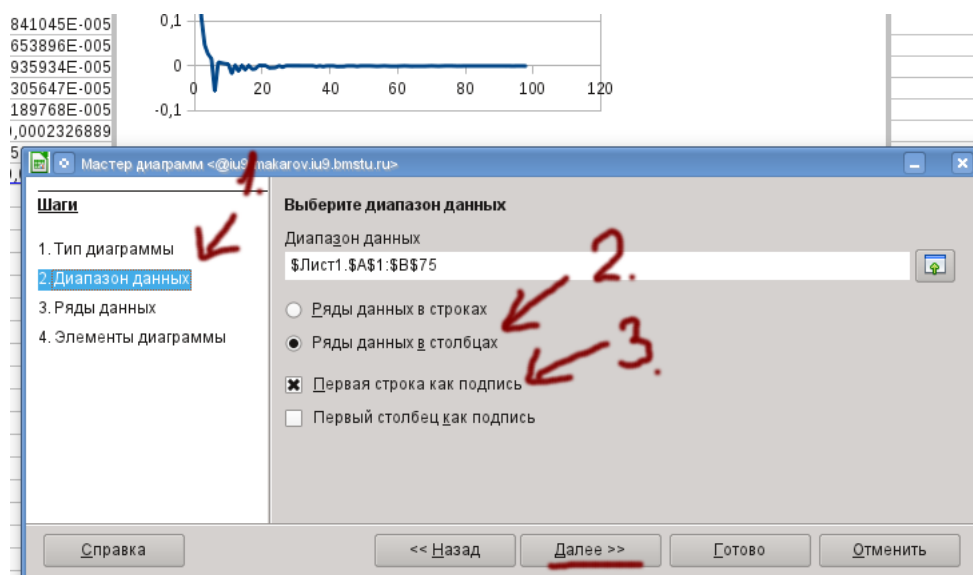


Рисунок 12: Уточнение типа диаграммы.

Далее полезно убедиться в правильной установке остальных опций мастера; на следующей странице (рис. 12) следует проверить установку переключателя «ряды данных в столбцах» и, если в первой строке файла содержатся названия столбцов («Число шагов» и пр.), то пункт «Первая строка как подпись» должен быть отмечен.

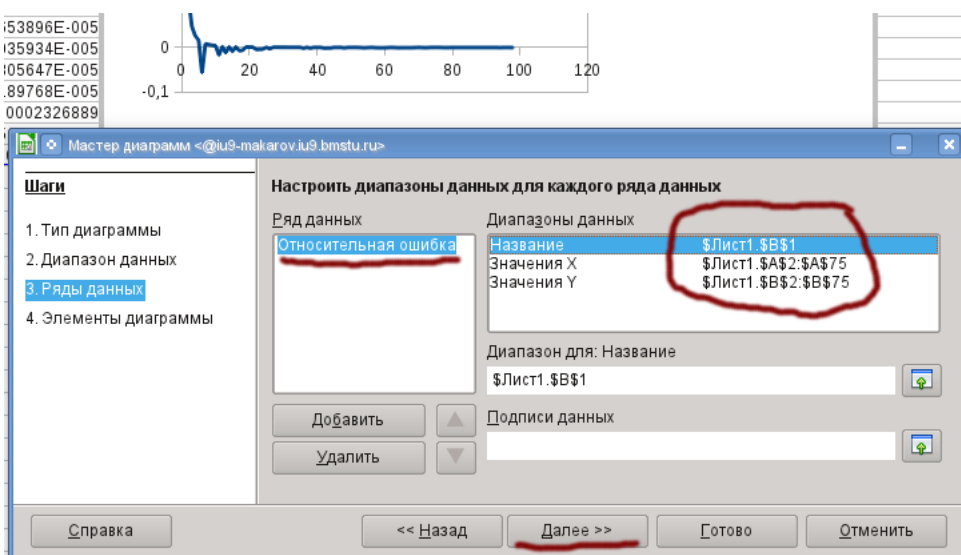


Рисунок 13: Проверка свойств ряда данных.

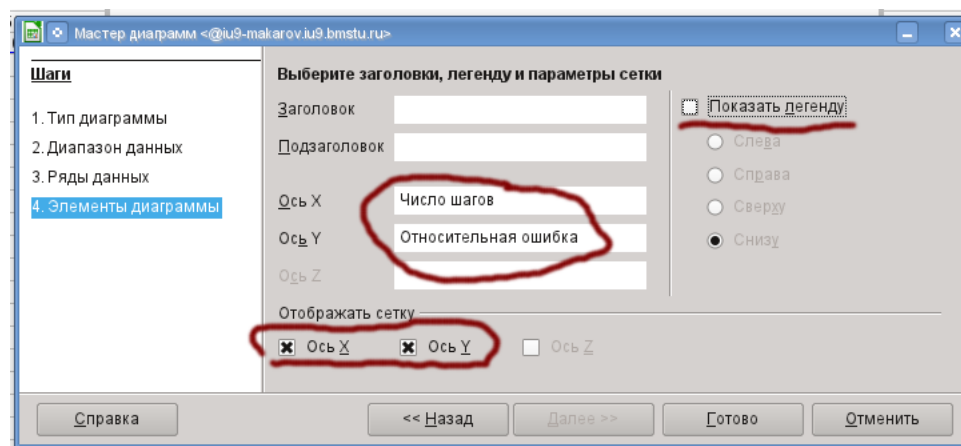


Рисунок 14: Основные элементы оформления диаграммы.

На третьей странице мастера (рис. 13) стоит проверить правильность определения названия ряда (рядов, если в таблице много столбцов) и диапазонов ячеек со значениями; при необходимости их можно исправить руками.

Последняя страница мастера (см. рис. 14 выше) позволяет указать названия диаграммы, надписи рядом с осями, место отображения легенды. В публикациях названия (заголовок и подзаголовок диаграммы) не указываются: для этого предназначены обязательные подписи под рисунками (Например, «Рисунок 13: Проверка свойств ряда данных.»), на самой диаграмме названия быть не должно. А вот *названия осей с обозначением размерности* (для величин с размерностью), наоборот, обязательны. Также целесообразно указать на необходимость рисования сетки по обеим осям (шаг сетки и стиль линии будет настраиваться позже). Легенда не нужна, если на графике представлена только одна зависимость или если зависимостей несколько, но подпись под рисунком достаточна для понимания где какая зависимость приведена. Впрочем, если на одном графике показано две и более зависимостей, то гораздо чаще в график включают легенду, на которой указано какой линией и какими значками обозначена та или иная зависимость.

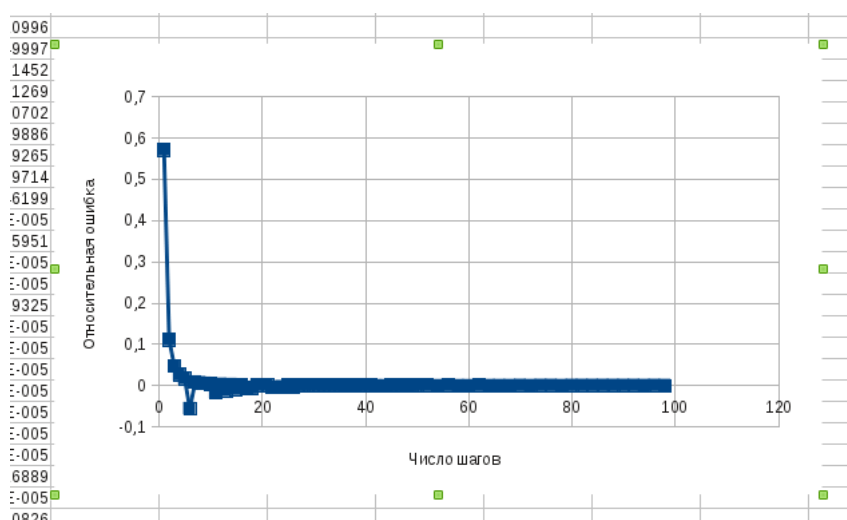


Рисунок 15: Внешний вид графика после завершения работы мастера.

У сформированного мастером графика (рис. 15) остаётся ещё множество недостатков: а) автоматическое определение границ обычно несколько ошибается (справа много пустого пространства от 100 до 120); б) сам график занимает далеко не всю отведённую ему область, поля излишне широки; в) оси координат и сетка визуально не отличаются (в данном случае ось абсцисс — вторая снизу линия); г) подписи к осям удобнее размещать около концов осей, а не по центру; д) линия графика достаточно толста, а использованные для обозначения точек маркеры чересчур огромны и сливаются друг с другом.

Такой график следует отредактировать, чтобы избавиться от этих недостатков. Начиная редактирование (см. рис. 16 ниже) с изменения свойств осей: сначала ординат, потом абсцисс. Для этого после двойного «щелчка» по графику и «щелчка» правой кнопкой мыши на оси ординат выбираем в контекстном меню пункт «Свойства оси». Как минимум, нас интересуют вкладки:

«Линия», позволяющая задать стиль линии для рисования оси. Сетку удобно оставить серой, а вот для осей задать тонкую (менее 0.5 мм, в данном случае 0.2мм) сплошную линию чёрного цвета. Для второй оси позже стоит задать такие же значения.

«Масштабирование», позволяющее указать диапазон изменения значений по оси. На рисунке показано, что «галочки» при выборе максимального и минимального значений по оси ординат сняты; но так следует делать лишь в самом конце — первое время удобно их задавать автоматически, и отказаться от автоматического задания только если в чистовом варианте они неудачно определяются. Дело в том, что меняя левую и правую границу оси абсцисс удобно исследовать отдельные части графика; чтобы его было хорошо видно удобнее

оставить выбор масштаба по оси ординат автоматическим и лишь когда станет ясно, какой именно диапазон значений по оси абсцисс нас интересует, то тогда станут ясны и диапазон значений оси ординат. В «чистовом» варианте стоит подобрать диапазон значений, шаг сетки и число дополнительных интервалов на шкале (и, возможно, сетке) вручную для лучшего внешнего вида.

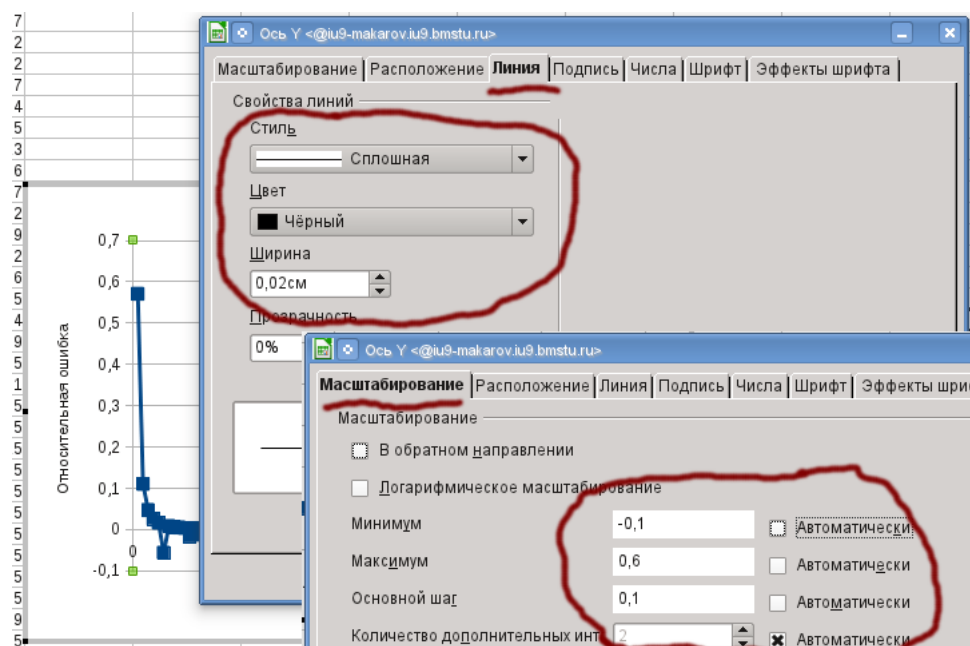


Рисунок 16: Изменение свойств оси ординат.

После задания свойств осей можно пододвинуть их названия и подправить размер самой диаграммы (см. рис. 17). При этом следует постараться обеспечить такой внешний размер диаграммы, чтобы при вставке в отчёт не происходило его масштабирования. Для этого стоит посмотреть на разметку страницы отчёта — размер листа, ориентацию и ширину полей и установить такие же значения для формата страницы, содержащей график. В дальнейшем можно иногда делать «print preview», чтобы убедиться, что диаграмма умещается на листе (возможно, её для этого придётся пододвинуть на листе с таблицей так, чтобы на печати она располагалась у левого края листа).

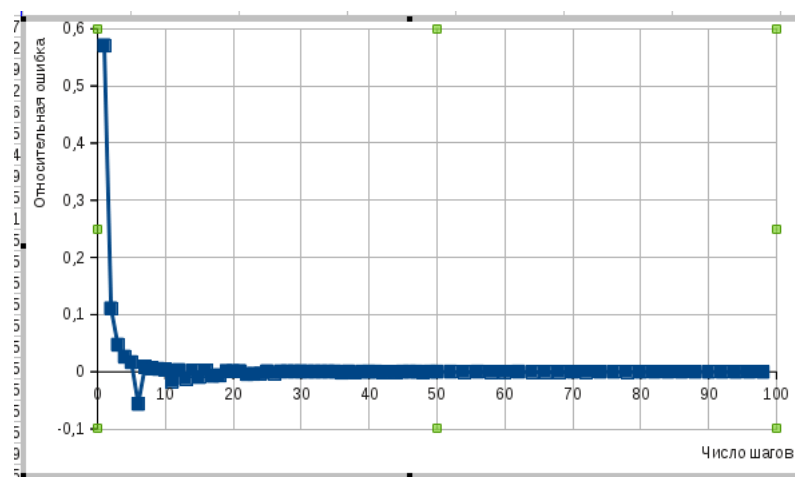


Рисунок 17: Улучшенный вид графика.

Окончательно надо исправить стиль линии, использованной для отображения графика. Её тоже надо сделать либо чёрной, либо серой (цветные допустимы только в случае презентаций на цветном устройстве или цветной печати; но даже при этом чёрно-серые зачастую лучше воспринимаются) и достаточно тонкой (менее 0,5 мм, но не менее 0,2 мм — на некоторых устройствах, способных выводить очень тонкие линии, линия «нулевой» толщины или

менее 0.2 мм может оказаться плохо различимой). Размер значков, изображающих точки, тоже следует сделать небольшим — 1-1.5 мм. См. рис. 18 ниже.

Возможно, надо будет вернуться к свойствам осей и изменить шрифт и формат представления чисел на осях — это может существенно изменить «читаемость» графика.

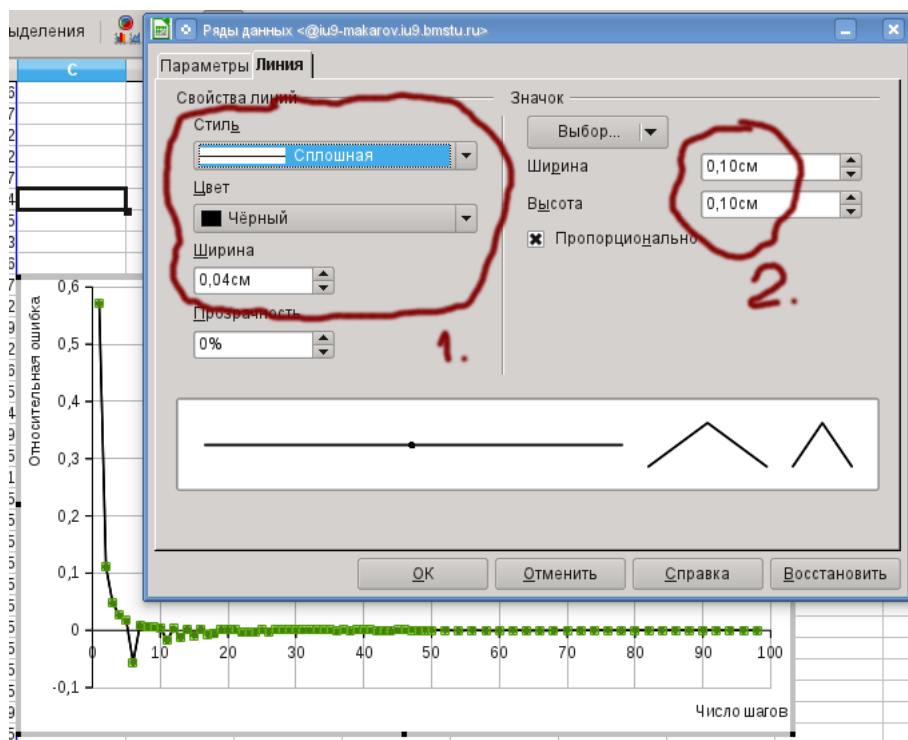


Рисунок 18: Исправление стиля графика.

Полученный таким способом график выглядит (см. рис. 19) так себе, хотя даже в таком виде извлечь из него информацию легче: отдельные точки не сливаются друг с другом, точнее видны странные «выбросы» некоторых точек и т. п. В чистовом варианте график будет смотреться лучше хотя бы потому, что здесь приводится растровая картинка, снятая с экрана, а в отчёт должны вставляться векторные изображения, качество воспроизведения которых заметно выше (об этом позже).

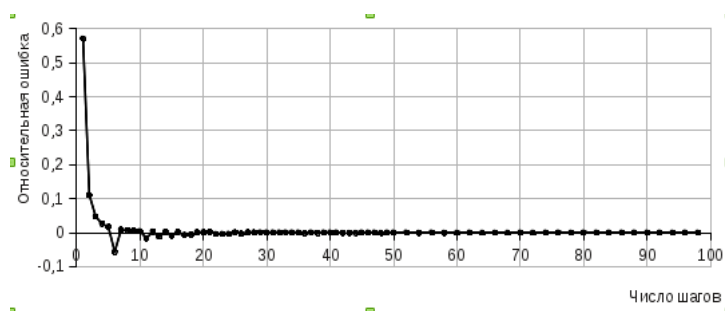


Рисунок 19: Примерный вид полученного графика.

То, что полученные результаты теста явно требуют исправлений в программе, в общем, понятно даже и по такому графику; но дополнительно попробуем добавить сюда график оценки погрешности (см. формулу для вычисления  $|R|$  на рис. 2, стр. 5, она же (13) ниже):

$$|R| < (X_R - X_L)^3 / 24N^2 * \max |f''(x)| \quad (13)$$

В нашей задаче  $X_L = 0$ ,  $X_R = \pi$ , т. е.  $X_R - X_L = \pi$ , а максимальное значение модуля второй производной на интервале интегрирования равно 1.0, т. к. подынтегральная функция в примере (лист. 1, стр. 7) - это  $\sin(x)$ . Т. е. упрощённо мы можем оценить ошибку метода (14):

$$|R| < \pi^3 / 24N^2 * 1.0 \quad (14)$$

Надо, однако, учитывать, что так определённая ошибка — это оценка абсолютной погрешности, т. е. нам надо её ещё нормировать до относительной (15):

$$|R_{\text{отн}}| < \pi^3 / 24 N^2 * 1.0 / 2.0 \quad (15)$$

Здесь 2.0 — точное значение интеграла на нашем интервале интегрирования.

Для добавления этого графика посчитаем значения оценки ошибки метода непосредственно в Libre Office Calc. Для этого сначала добавим столбец с вычисленными по формуле (15) значениями, а потом добавим ещё один график в эту диаграмму.

	A	B	C	D	E
1	Число шагов	Относительная ошибка			
2	1	0,5707963268	=PI()*PI()*PI()*1/24/\$A2/\$A2/2		
3	2	0,1107207345			
4	3	0,0471975512			
5	4	0,026172153			
6	5	0,0166407385			

Рисунок 20: Задание формулы для вычисления оценки погрешности.

На рис. 20 показано задание формулы для вычисления оценки погрешности в ячейке C2 (т. е. для случая с одним шагом интегрирования). PI() -функция, возвращающая значение  $\pi$ , \$A2 — ссылка на ячейку A2 таблицы (если эту формулу вставить в другую ячейку в другой строке, например, в 3, то цифра 2 изменится в соответствии с новым размещением — 3, 4 — если в строке 4 и т. п., а буква даже при вставке в другой столбец изменяться не будет — об этом говорит знак \$ перед буквой<sup>2</sup>). Далее, если эту формулу скопировать в буфер обмена (clipboard) и затем вставить в правую колонку (Ctrl+C, выделить необходимый диапазон, Ctrl-V), то будут вычислены значения оценки во всех соответствующих строках.

Далее надо добавить ещё один график в уже существующую диаграмму. Для этого проще всего начать редактирование диаграммы (двойной щелчок на диаграмме или пункт «Правка» в контекстном меню), далее правой кнопкой мыши вызвать контекстное меню в области диаграммы (см. рис. 21) и выбрать пункт «Диапазоны данных...».

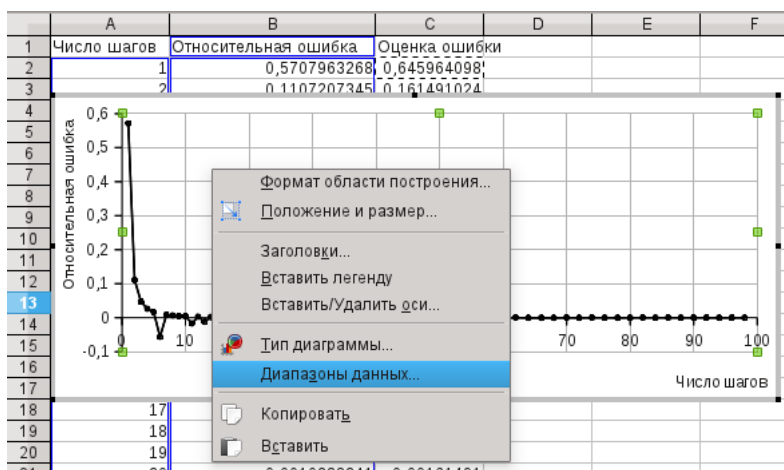


Рисунок 21: Изменение рядов данных существующей диаграммы.

В появившемся диалоге надо открыть вкладку «Ряды данных» (см. рис. 22 на стр. 22); в списке рядов слева должен быть выделен (пока единственный) ряд, в списке свойств ряда

<sup>2</sup> Если ссылка имеет вид A1, то при вставке в новую ячейку изменяется и буква и цифра; \$A\$1 — не изменится ничего; \$A1 — изменится только цифра, но не буква; а если A\$1, от изменяться будет только буква. Изменение, если разрешено, буквы/цифры выполняется так, что сохраняется относительное расположение между источником и целью ссылки.

Чтобы воспользоваться этим способом запомним в буфере обмена диапазон для значений X или Y (в нашем случае строку «\$Лист1.\$A\$2:\$A\$75»), после чего надо нажать кнопку «Добавить» и последовательно вставить и отредактировать значения атрибутов «название», «Значения X» и «Значения Y» по аналогии с уже существующим рядом.



Диапазоны данных <@iu9-makarov.iu9.bmstu.ru>

Диапазон данных Ряды данных

Ряд данных

Относительная ошибка

Оценка ошибки

Диапазоны данных

Название \$Лист1.\$C\$1

Значения X \$Лист1.\$A\$2:\$A\$75

Значения Y \$Лист1.\$C\$2:\$C\$75

Диапазон для: Значения Y

\$Лист1.\$C\$2:\$C\$75

Добавить

Удалить

Подписи данных

OK Отменить Справка

Рисунок 23: Новый ряд данных и его свойства.

Полученная диаграмма в результате выглядит примерно так (рис. 24). С точки зрения оформления это уже вполне устраивает, но воспользоваться ей для анализа результатов затруднительно — оба графика почти точно наложены друг на друга, да и разобраться в длинном почти горизонтальном ряду из точек тоже затруднительно. Очень важно так подобрать параметры диаграммы, чтобы проблемные места стали видны.

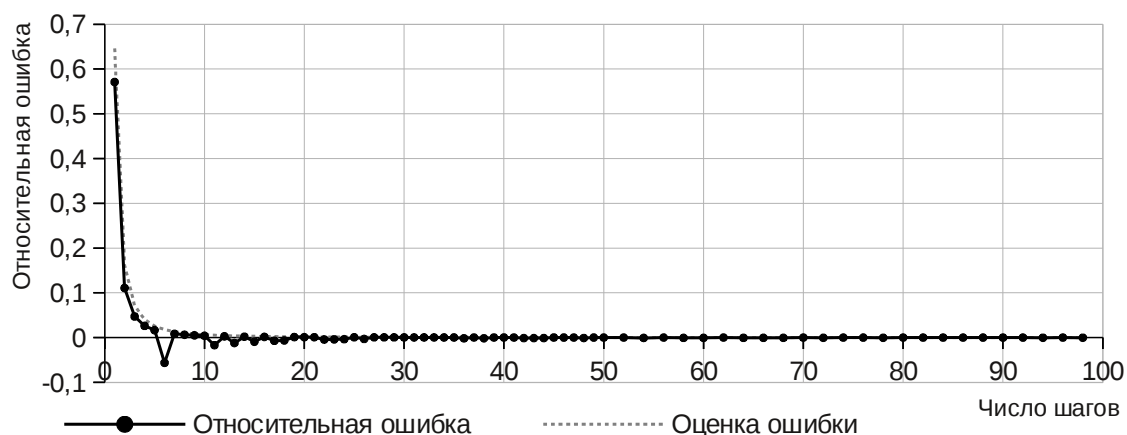


Рисунок 24: Диаграмма с двумя графиками.

В данном случае, например, можно пропустить несколько первых точек, т. е. начинать с 3..4 точки, и для начала «обрезать» длинный хвост справа: ограничиться рассмотрением первых 15..20 точек. Проще всего не менять диапазонов данных просто изменить масштаб по оси абсцисс, аналогично тому, как это было показано для оси ординат на рис. 16 см. стр. 19. При таком «поисковом» подборе масштаба по оси абсцисс стоит установку максимального и минимального значений и шаг по оси ординат сделать автоматическим.

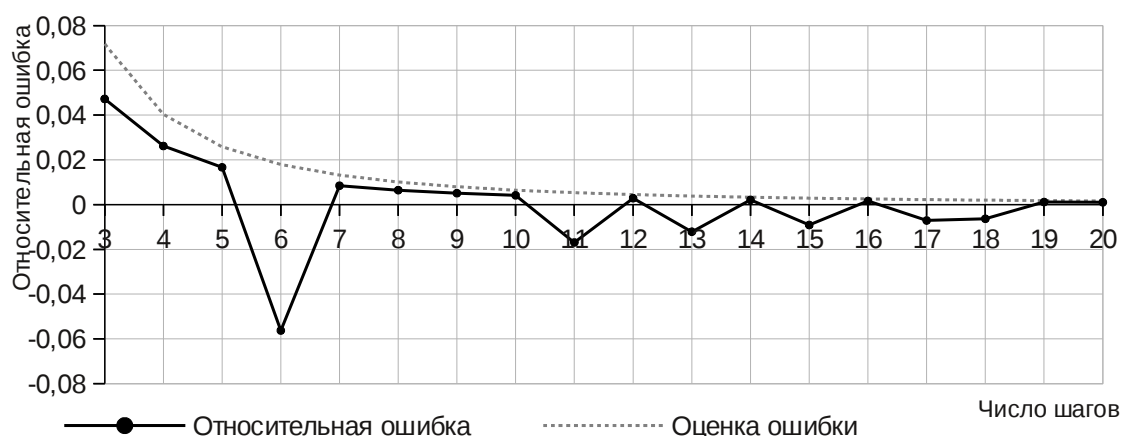


Рисунок 25: Уточнённый вид диаграммы (он же использован в обсуждении).

По такой диаграмме уже можно сделать некоторые выводы; подробнее см. в её обсуждении в разделе «Примерная логика выполнения каждого этапа» на стр. 8.

## Вставка графиков в отчёт

Особенных сложностей эта операция представлять не должна, но тут следует учитывать несколько особенностей офисных продуктов.

*Первое.* Внедрение объектов в документ осуществляется двумя основными способами: а) вставкой в документ всего объекта целиком или б) вставкой в документ ссылки на объект, находящийся во внешнем файле. Последний способ выглядит теоретически более интересным: он позволяет редактированием внешнего документа, например, с графиком изменить его вид в основном отчёте. Однако на практике этот способ оказывается менее удобным, чем первый из-за одного нюанса: размеры внедрённых документов зависят при этом от обоих документов. Более того, зачастую они зависят от свойств выбранных принтеров: Microsoft Office, например, «запоминает» размеры объектов с привязкой к принтеру, так что изменение разрешения принтера или смена одного принтера на другой (даже с тем же разрешением при печати) приводит к сбросу и пересчёту размеров — а это, скорее всего, будет сопровождаться переразбивкой на страницы, переносом материала с одной страницы на другую и т. п.



Потенциально способ со вставкой ссылок на внешние документы является более эффективным, но требует (по крайней мере первое время) куда большей ручной работы и некоторой «привычки», позволяющей минимизировать эту работу. Кроме того, следует учитывать, что все связанные документы надо обязательно переносить совместно друг с другом и то, что после переноса может потребоваться исправление путей до связанных документов. Некоторые офисы предоставляют возможность в существующем документе с ссылками на внешние документы управлять «связями», выборочно обновлять те или иные документы или «разрывать» связь, когда в документ будет внедряться копия документа, на который ранее существовала ссылка. Например, в Libre Office для этого можно в меню «Правка» выбрать пункт «Связи...».

Начинать рекомендуется с более чуть простого способа: внедрения объектов целиком.

*Второе.* Один и тот же объект может быть внедрён в различных форматах, что будет связано с различными возможностями при работе с внедрённым объектом. Офисные продукты, как правило, поддерживают технологию ActiveX (OLE или эквивалентную), позволяющую осуществлять редактирование внедрённого объекта той программой, которая использовалась при его создании (или которая связана с объектами данного типа). Так прямо в отчёт, например, Microsoft Word может быть внедрён документ Microsoft Excel (таблица или график) так, что при работе с ним будут использованы все возможности Excel. Звучит это всё прекрасно, но... как, например, напечатать этот документ на машине, на которой нет Excel? Или на которой excel другой версии? Как быть с переносом таких документов из Microsoft Office в Libre Office, как последний сможет обработать объект неизвестного для него типа? Обычно при внедрении объектов сложных типов вместе с самим объектом внедряется его изображение в виде растровой картинки — это позволяет печатать, даже при отсутствии нужной программы. Однако печать — одна из самых простых задач; возможность редактирования в другом офисе является куда более сложной.

Более того — при этом возможно куча накладок, когда, например, документ, созданный в Microsoft Word с внедрённой диаграммой Microsoft Excel потом немного редактируют в Libre Office и ещё позже снова в Microsoft Office. В данном примере, скажем, Libre Office постарается отконвертировать внедрённый объект в формат Libre Calc, после чего этот объект уже не может быть открыт в Microsoft Excel, так как последний «не понимает» формата Libre Calc. Да и сама конвертация из одного формата в другой часто связана с искажениями объектов, изменением их размеров и т. п.

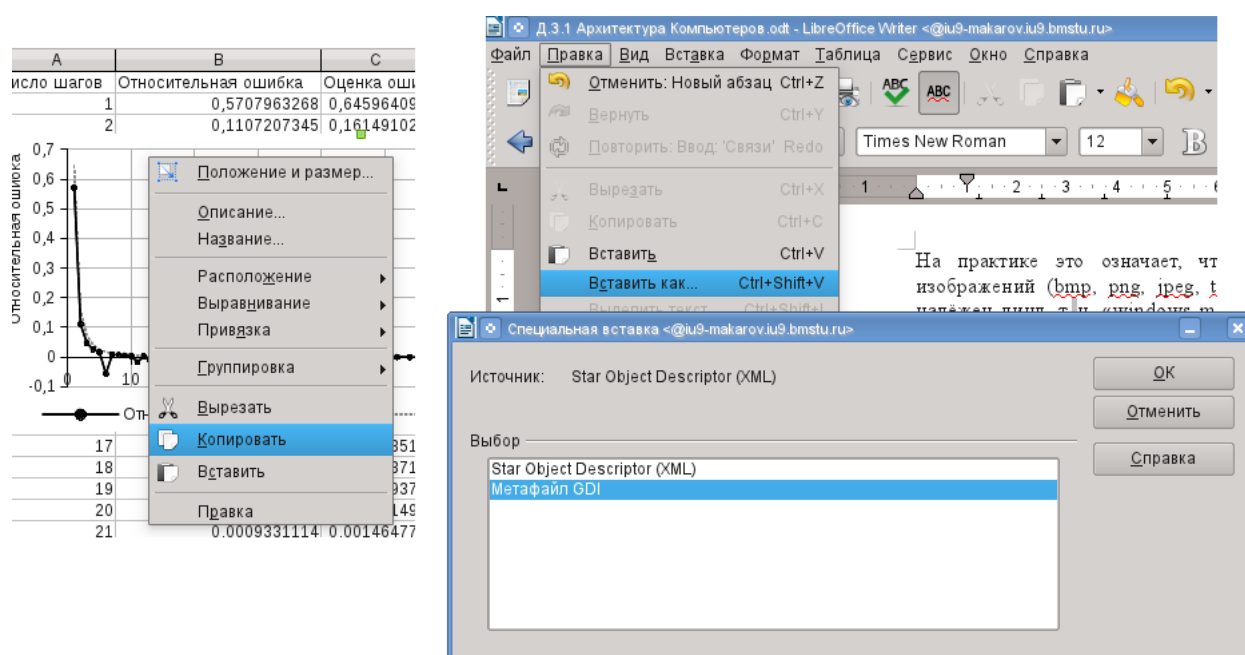


Рисунок 26: Копирование (слева, LibreOffice Calc) и вставка диаграммы в виде метафайла (справа, LibreOffice Writer).



С этой точки зрения полезно (особенно, если есть некоторая вероятность того, что документ придётся переносить между разными офисами) использовать какие-либо универсальные форматы внедрённых объектов, которые могут быть поняты обоими офисами. На практике это означает, что надо использовать либо объекты в виде растровых изображений (bmp, png, jpeg, tiff и т. п.) или в виде векторных (среди них фактически надёжен лишь т. н. «windows metafile», он же «wmf» или «enchanced metafile», он же «emf»; иногда их называют просто «metafile» или «метафайл»).

Собственно операция внедрения объекта целиком в нужном формате выполняется очень просто: сначала в одном редакторе копируем объект в буфер обмена (и Excel и Calc и большинство других более-менее мощных редакторов помещают в буфер обмена выбранный объект сразу в нескольких возможных форматах) и при вставке в нужном документе в нужном месте используем не простой вариант вставки, а чуть более сложный, либо через меню: «Правка | Вставить как...»<sup>3</sup>, либо (многие офисы поддерживают такую комбинацию клавиш) Ctrl-Shift-V. В этом случае появляется диалог, в котором перечислены доступные в буфере обмена форматы и вы должны выбрать нужный (см. рис. 26 выше).

К слову: рис. 24 на стр. 23 вставлен в документ именно как метафайл, его можно сравнить с рис. 19, стр. 20, который вставлен как растровое изображение. При вставке изображений важно проследить, чтобы размер встраиваемого изображения в документе был точно таким же, как и при его создании; это связано с некоторой потерей качества при масштабировании изображения; для растровых изображений масштабирование сильно ухудшает качество картинки, для векторных (таких, как метафайл), само качество не страдает, но могут сместиться друг относительно друга разные части изображений и ухудшится воспроизведение шрифтов (шрифт масштабируется плохо). Именно поэтому ещё при создании диаграммы надо следить за тем, чтобы она хорошо размещалась на листе итогового документа.

## Названия рисунков, таблиц, листингов

После вставки изображения (равно как и таблицы или листинга) необходимо предусмотреть его название (для рисунка название располагается *под* рисунком, для таблиц и листингов — *над* ними). Можно, конечно, формировать такую подпись «руками» под каждым рисунком, но это неудобно, так как *на любой рисунок, таблицу или листинг обязательно должна быть ссылка в тексте*. Если ссылки в тексте на какой-либо объект нет, то этот объект должен быть удалён из документа, он не нужен. А если нужен, то на него должна быть ссылка (как пример см. оформление рисунков и листингов в этих рекомендациях: подписи, ссылки и т. п.). Вставка ссылок руками чересчур трудоёмка, так как при необходимости вставить или удалить один из рисунков, всю нумерацию придётся исправлять во всех подписях и ссылках.

Для упрощения такой операции в офисных редакторах предусмотрена возможность автоматической вставки названий, их нумерация и возможность формирования ссылок. Названия для обычного, размещённого в документе текста (например, текста программы) в офисных программах не предусмотрены. Последнее решается не так и сложно — нужный текст просто вставляется либо во фрейм (не очень удобное решение), либо в таблицу из одной ячейки с невидимым обрамлением (это удобнее, т. к. ячейка таблицы может быть легко разбита между страницами, а фрейм всегда размещён на одной). Здесь, например, листинги (лист. 1 на стр. 7, лист. 4 на стр. 12 и лист. 5 на стр. 13) вставлены именно в таких невидимых таблицах.

Некоторая сложность тут появляется из-за того, что нужный объект (рисунок или текст) размещается в некотором «контейнере», которому назначается имя. Если «контейнером» является таблица, а содержимое — текст, то всё просто; иначе необходимо учитывать, как на листе совместно размещается объект и текст, причём как внутри контейнера, так и размещение текста и контейнера на странице.

---

3 Иногда этот пункт называется «Специальная вставка...», вместо «Вставить как...».

Существует два качественно разных способа: а) объект вставляется «как символ», т. е. является просто «очень большой буквой» и размещается в строке текста (даже если строка состоит только из этой «буквы»); современные офисные редакторы этот способ часто «маскируют» так, что не сразу и догадаешься, о чём идёт речь; и б) когда он вставляется «как объект» вокруг которого может «обтекать» текст. Для научных статей, отчётов, курсовых работ и т. п. приняты, может быть и не самые красивые с точки зрения дизайна, но зато простые и строгие правила: такие объекты вставляются без обтекания текстом, т. е. в той части листа, где расположен объект обычного текста быть не должно, даже если весь объект занимает только сантиметр от всей ширины листа (не стоит путать обтекание объекта текстом с возможностью вставки «составных» объектов из нескольких диаграмм, рисунков и т. п. рядом по ширине).

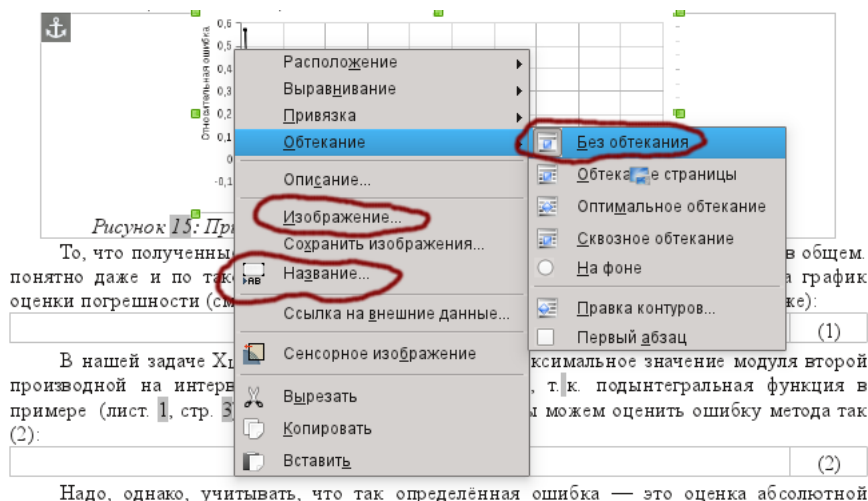


Рисунок 27: Режим обтекания изображения.

Перед придумыванием названия рисунка, листинга или таблицы стоит убедиться, что у него правильно задан режим обтекания (для таблиц это не требуется), см. рис. 27.

Аналогично, позже стоит проверить режим обтекания всего фрейма, для чего правой кнопкой мыши вызвать контекстное меню уже не для изображения, а для всего фрейма (щелчок правой кнопкой на рамке фрейма, а не на рисунке; перед этим убедиться, что ни фрейм, ни вложенный в него рисунок в данный момент не выделен). В случае каких-либо «странностей», например, текст оказывается наложен поверх изображения или фрейма и т. п., в первую очередь надо смотреть режим обтекания.

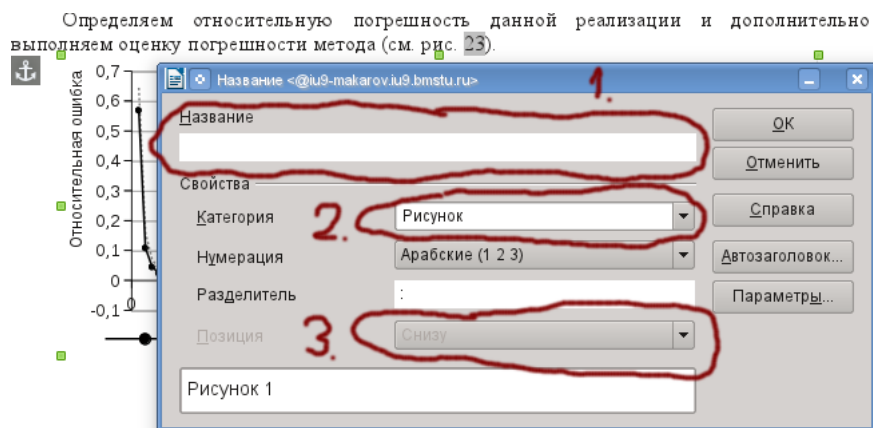


Рисунок 28: Задание названия для рисунка.

В контекстном меню доступны не все возможные режимы обтекания, а лишь самые часто используемые. Полное управление обтеканием возможно через диалог свойств объекта (для изображения пункт меню «Изображение...», как видно на рис. 27 выше, для фреймов «Врезка...», иногда «Свойства»); в этом диалоге есть специальная вкладка «Обтекание».

Для того, чтобы задать название объекта, в его контекстном меню (опять же, см. рис. 27 на стр. 26) надо выбрать пункт «Название...», после чего будет показан диалог, показанный на рис. 28:

В поле название надо ввести текст названия, не указывая начала («Рисунок». «Таблица» и т. п.), в поле «Категория», выбрать нужную категорию и, при необходимости, в поле «Позиция» указать, где название должно находиться: сверху или снизу от объекта. «Категории» можно придумывать самостоятельно — это поле может быть отредактировано. Например, стандартной категории «Листинг» не предусмотрено, но она была введена и использована в данном документе. При желании можно нажать кнопку «Параметры...» для задания дополнительных свойств (например, форматирования текста, используемого для текста названия); для стандартных категорий возможности по изменению их характеристик могут быть ограничены (здесь, скажем, считается, что для рисунков подпись всегда снизу и это не может быть изменено).

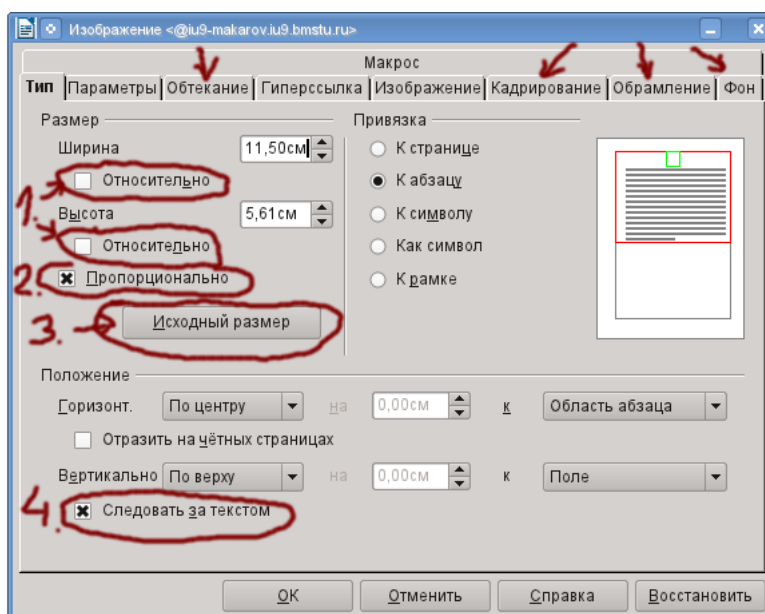


Рисунок 29: Проверка свойств изображения.

После задания названия офисный редактор обычно создаёт фрейм, содержащий внутри себя и само изображение и название. При этом фрейм может иметь ширину как находящаяся в нём картинка и быть заметно уже страницы, а текст названия при этом займёт много строчек. В этом случае (да и вообще, для единообразия оформления) ширину фрейма лучше увеличить до ширины области текста на странице, что легко сделать мышкой. Но при этом часто масштабируется сама картинка, находящаяся внутри фрейма. Для того, что бы вернуть размеры картинки к прежнему виду, надо вызвать диалог с её свойствами (см. рис. 29), через пункт «Изображение...» контекстного меню и в этом диалоге на вкладке «Тип» проверить, чтобы а) флажки «относительно были сняты»; б) стоял флажок «Пропорционально», после чего надо в) нажать кнопку «Исходный размер»; и г) поставить (если снят) флажок «Следовать за текстом». После этого диалог можно закрывать и, при необходимости, точно отцентрировать изображение (рисунки обычно центрируются по ширине) внутри фрейма.

На рис. 29 выше дополнительно стрелками отмечены несколько вкладок диалога, которые, возможно, надо будет проверить и подкорректировать.

## Перекрёстные ссылки

На любые материалы, вставляемые в текст отчёта, обязательно должны быть ссылки в тексте; материал, на который нет ссылки в тексте не нужен в отчёте. При использовании названий таких материалов формирование ссылок сложностей не представляет: для этого

надо воспользоваться стандартным инструментом, доступным в меню «Вставка | Перекрёстная ссылка...», см. рис. 30 ниже:

Далее будет показан диалог, в котором надо будет выбрать цель, на которую надо установить ссылку, и указать тип этой ссылки, см. рис. 31 ниже.

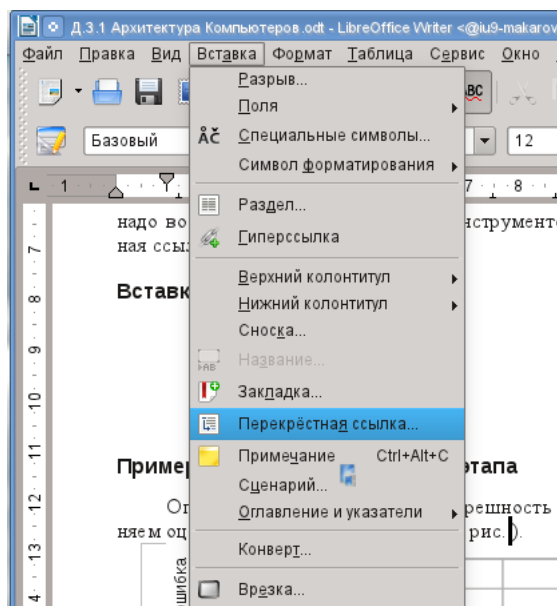


Рисунок 30: Вставка перекрёстной ссылки.

В этом диалоге надо открыть вкладку «Перекрёстные ссылки», далее в списке слева выбрать категорию, в списке справа найти название нужного материала и затем в списке снизу выбрать тип ссылки. Как минимум, надо указывать номер (выделен на рисунке) или «Категорию и номер» (в первом случае будет просто вставлен номер рисунка, таблицы, листинга и т. п., например, 31; во втором будет вставлено что-то вроде «31»). Обычно в документах используют сокращённые названия категорий, т. е. «рис.» вместо «Рисунок», заглавная буква тоже при этом не используется, поэтому в LibreOffice Writer удобнее вставлять только номер, а сокращённое название категории вписывать руками, а Microsoft Office можно нормально настроить для использования сокращённых названий категорий, что чуть удобнее.

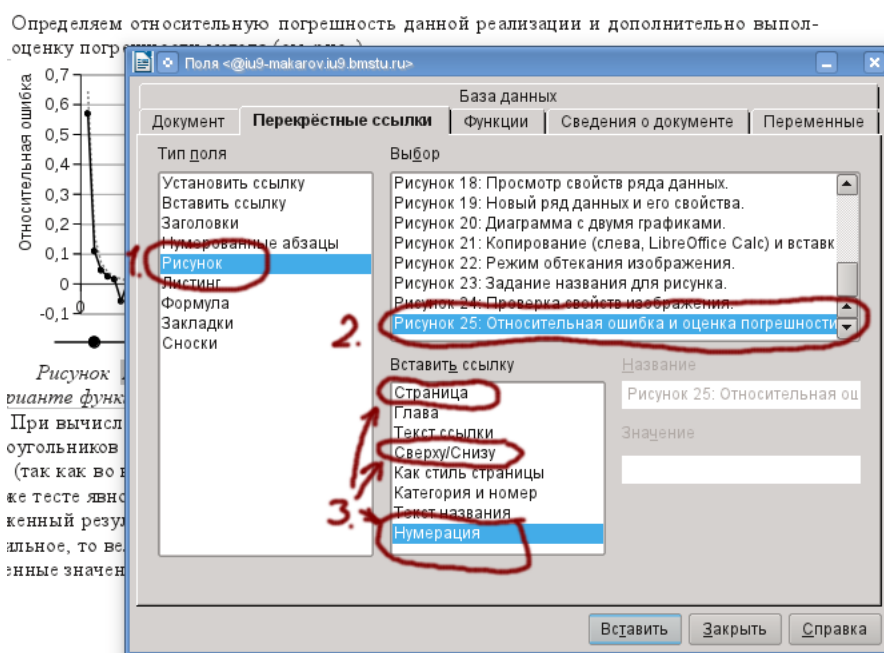


Рисунок 31: Выбор цели и задание типа ссылки.

Если объект, на который осуществляется ссылка, находится не на этой странице, то имеет смысл дополнительно указать, где его надо искать «выше», «ниже» или на какой странице; всё это можно сделать, вставляя ссылки соответствующего типа, например, «см. рис. 31 на стр. 28 выше» (здесь используется сразу три ссылки на один и тот же рисунок; если объект находится на расстоянии в сколько-то страниц, то рекомендуется указывать номер страницы, а не «выше» или «ниже», если близко, то достаточно направления).

## Вставка формул

Формулы вставляются либо прямо в тексте, если они небольшие и имеют поясняющий характер или задают конкретные условия, либо размещаются на отдельной строке; обязательно по центру строки (таково требование к оформлению математических формул). Справа от формулы у края области текста на странице в скобках указывается номер этой формулы. В качестве примера, см. стр. 20, формулы (13), (14) и (15), а также небольшие формулы, внедрённые прямо в строку между (13) и (14). Номера формул и ссылки на них заключаются в круглые скобки. Если на формулу нет никаких ссылок, а она используется лишь в контексте нескольких смежных (например, промежуточные шаги преобразований), то номер формулы указывать не обязательно.

Автоматическая нумерация формул может потребовать небольших ухищрений; офисные продукты созданы «с прицелом» не на научные статьи, а на автоматизацию работы офисных сотрудников с несколько иными типовыми задачами. Для того, чтобы можно было вставлять номера формул потребуется небольшая предварительная операция: надо вставить номера первой формулы определить специальную автоматически нумеруемую категорию ссылок. Для этого надо снова обратиться к меню «Вставка | Перекрёстная ссылка...», как показано на рис. 30 выше и в появившемся диалоге выбрать вкладку «Переменные», а не «Перекрёстные ссылки», см. рис. 32.

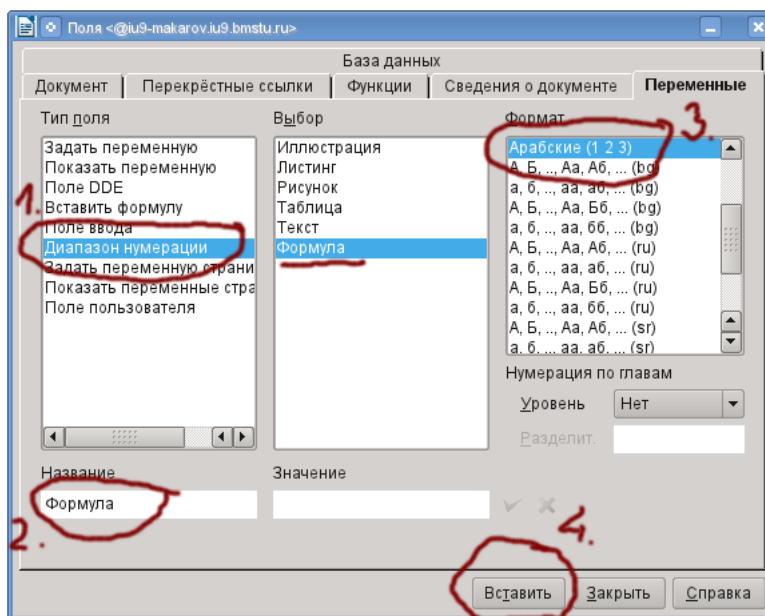


Рисунок 32: Создание нумеруемой категории и вставка автоматически увеличивающегося номера.

Первоначально категории «Формула» с средним списке нет, его надо ввести в поле «Название», после чего можно нажать кнопку «Вставить» для вставки номера в текст (если значений не задавать, то нумерация начинается с 1). В дальнейшем с помощью этого диалога надо просто вставлять переменную типа «Формула», взятую в круглые скобки справа от нужных формул, а при необходимости установить на неё ссылку из текста поступать точно так же, как и с прочими внедрёнными материалами: на вкладке «Перекрёстные ссылки» этого

документа (рис. 31 на стр. 28) в левом списке категорий будет присутствовать созданная нами категория «Формула».

Сами формулы, если они достаточно простые, лучше набирать в как обычный текст, используя по мере необходимости символы других алфавитов и/или шрифтов (здесь, например, удалось ограничиться такими средствами). В противном случае следует подобрать какой-либо внешний редактор формул (либо специальный инструмент с таким же названием «Редактор формул», входящий в состав офиса; либо вообще какая-либо иная программа, например, MathCad, Mathematica и т. п., внедряя созданные там формулы как объекты (если перенос в другую систему актуален, то удобнее вставлять как метафайлы).

## **Заголовки и оглавление**

Современные офисные средства предоставляют массу удобных инструментов; в том числе автоматическое формирование указателей, оглавлений, списков терминов и т. п. Большая их часть опирается на развитую работу с т. н. «стилями» - т. е. набором характерных черт форматирования текста. Стили делят на «стили символа» и «стили абзаца»; к сожалению, в офисах это деление стилей на две группы реально предполагается, но часто (или «местами») скрыто. Так в диалог настройки стиля обычно включают сразу и форматирование абзаца (отступы, межстрочный интервал, поля, фон и рамка, положение на странице, возможность разрыва на границе страниц и т. п.) и форматирование символов (шрифт, размер, наклонность и жирность шрифта, цвет шрифта, межсимвольный интервал, надстрочный или подстрочные знаки и т. п.), т. е. с точки зрения диалога отличить один тип стилей от другого нельзя, хотя в реальности разница имеет место.

Стили образуют иерархию: обычно стиль определяется не «на ровном месте», а как некоторый другой стиль, к которому добавлены новые атрибуты. Изменение стиля, на основе которого определены «наследники» меняет также все непереопределённые наследниками свойства. Грамотное использование стилей предполагает, что ещё до начала написания отчёта оценивается необходимый набор стилей, стили определяются и далее, по мере набора текста, назначаются сразу стили, а не отдельные элементы форматирования. Ну, например, имеет смысл использовать определить два базовых стиля (базовый стиль символов, скажем Times New Roman, 12 пунктов; и базовый стиль абзаца — красная строка 1 см, во всю ширину текста, двойной или полуторный интервал, перенос слов и т. п.); далее на их основе определить несколько дополнительных стилей (например, стиль символа для написания текста программ, скажем, Monospaced, размер как у базового; стиль абзаца для листингов — без красной строки, выравнивание влево, шрифт такой же, как стиля символов для программ, переносов нет) и т. д. и т. п.

Среди прочих важное место занимают стили заголовков; они важны не только, как элемент оформления текста, но как элемент структурирования. Дело в том, что в офисах существуют специальные мастера, помогающие сформировать автоматически оглавление по разметке документа — причём ориентируются они именно на стили заголовков или унаследованных от них стилей. Нумерация заголовков настраивается как свойства стиля. Во многих документах, например, записках к курсовым и дипломным работам требуется нумерация заголовков (в данном Д.З. нумерация не требуется).

В данных рекомендациях, к примеру, используется три уровня заголовков, стили которых называются «Заголовок 1», «Заголовок 2» и «Заголовок 3», при этом использованы стандартные стили заголовков со слегка изменёнными характеристиками (например, для стиля «Заголовок 3» уменьшен размер шрифта).

Если заголовки разделов размечены с использованием правильных стилей, то в конце документа достаточно установить курсор в пустую строку и выбрать пункт меню «Вставка | Оглавление и указатели | Оглавление и указатели...», см. рис. 33.

После чего запускается мастер формирования оглавления (см. рис. 34 ниже) результат работы которого после небольшой коррекции можно увидеть в этом документе.



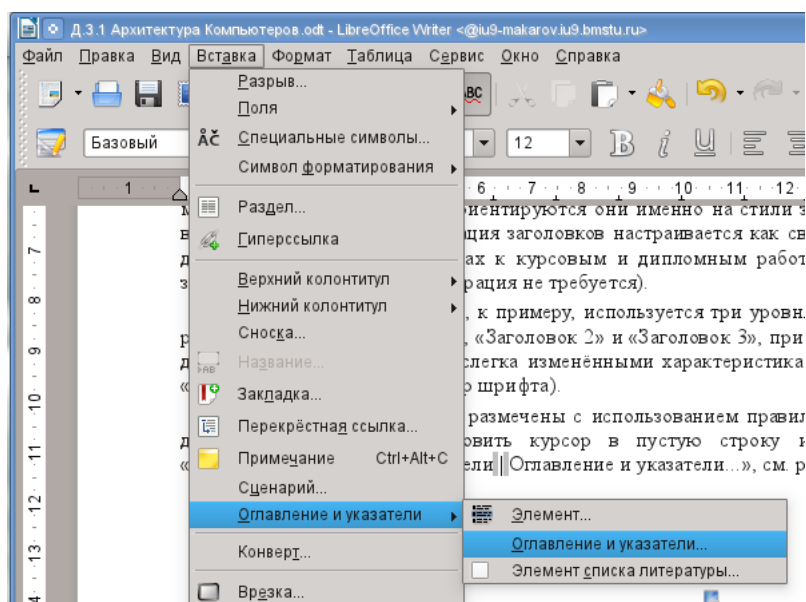


Рисунок 33: Вставка оглавления в документ.

В этом мастере задаётся заголовок оглавления (в нашем случае «Содержание»), указывается, что вставляем именно оглавление, а не указатель, флажок «Защищено от изменений вручную» во время работы с документом лучше держать отмеченным, а вот в финишном варианте, возможно, его придётся снять и руками подправить форматирование, переносы и т. п. отдельных пунктов. Здесь же можно ограничить максимальную уровень заголовков, которые будут включаться в оглавление. В частном случае для заголовков могут использоваться нестандартные стили, тогда можно воспользоваться флажком «Дополнительные стили» и кнопкой рядом с ним для настройки этих стилей. Обычно случае достаточно настроить оглавление, оставаясь на данной вкладке диалога; иногда может потребоваться существенное изменение представления пунктов оглавления разного уровня, добавление или изменение элементов и т. п.; всё это можно сделать с помощью остальных вкладок мастера.

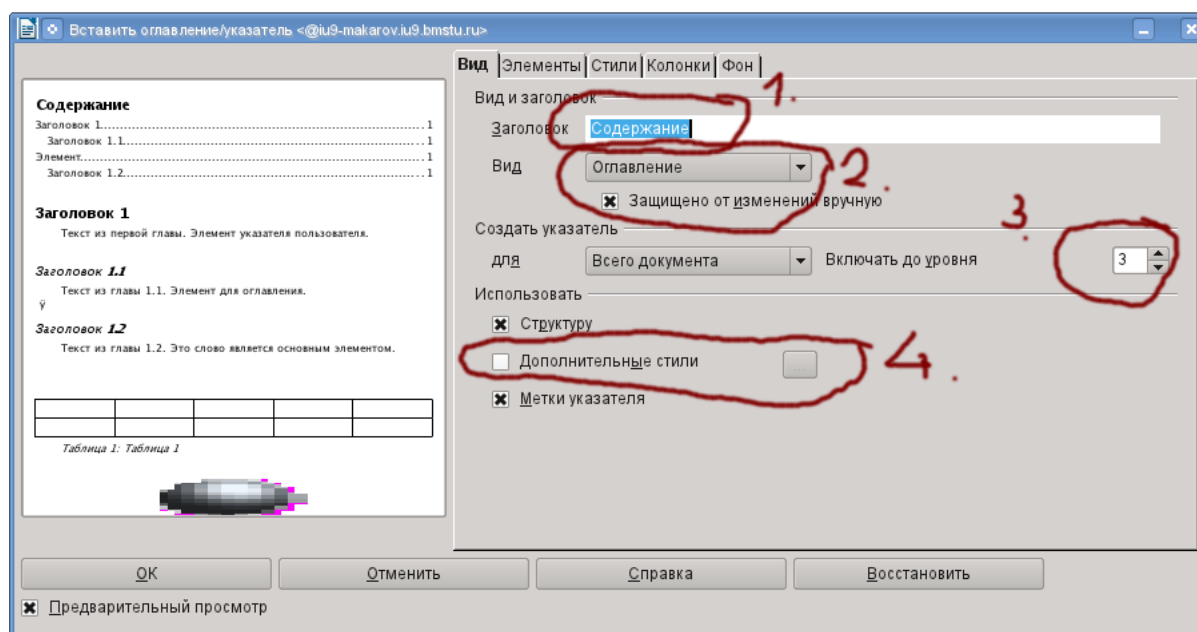


Рисунок 34: Мастер формирования оглавления.

Если посмотреть на созданное мастером оглавление (см. рис. 35), то можно заметить, что а) Заголовок оглавления следует непосредственно за текстом; обычно удобнее оглавление начинать с новой страницы и б) выравнивание левой и правой границ самого первого заголовка выглядит некрасиво. Для подобных изменений надо исправлять не само оглавление, а

используемые для него стили. Так, например, для стиля заголовка оглавления надо добавить признак «начинать с новой страницы», а для стилей оглавления изменить выравнивание (каждому стилю «Заголовок *i*» соответствует стиль «Оглавление *i*», используемый в оглавлении).

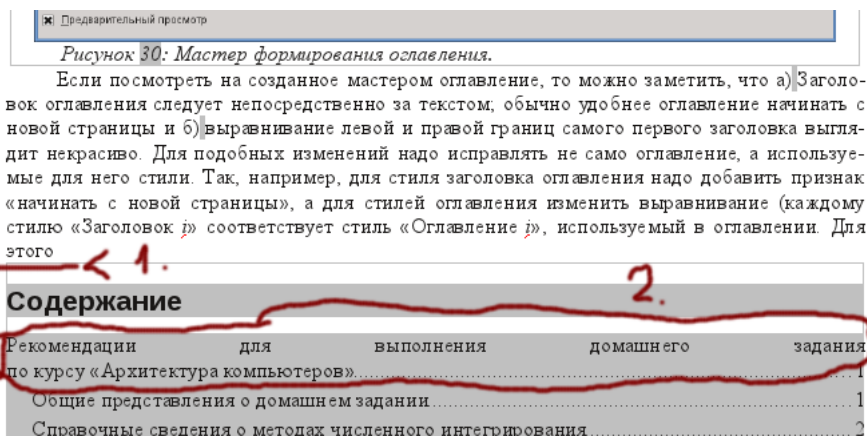


Рисунок 35: Сформированное мастером оглавление.

Для этого удобно в меню выбрать пункт «Формат | Стили», см. рис. 36 ниже. В появившемся списке стилей надо поочерёдно выбрать нужные нам стили и, вызвав правой кнопкой мыши для каждого из них контекстное меню, выбрать пункт «Изменить» для коррекции свойств стиля.

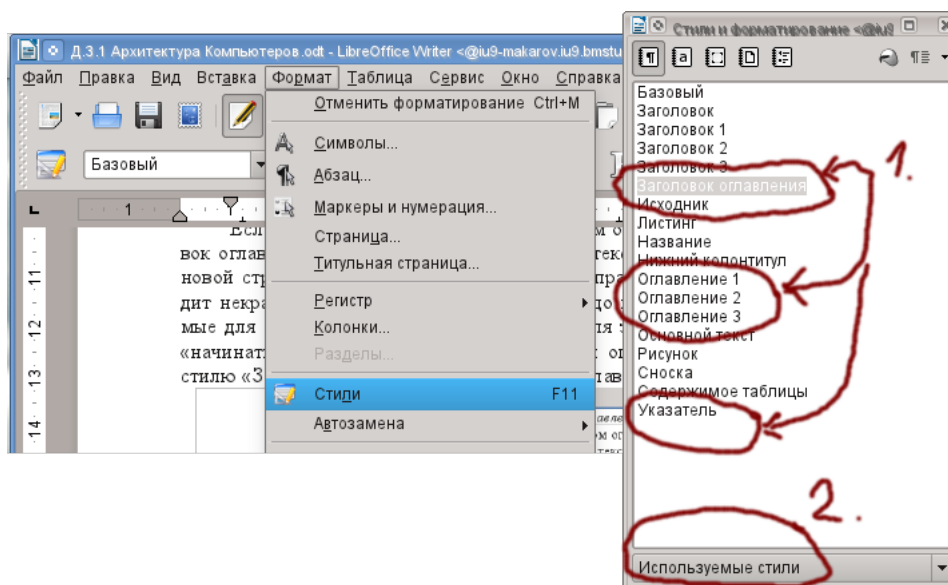


Рисунок 36: Просмотр и изменение используемых стилей.

Несколько примечаний:

- стили «Оглавление *i*» унаследованы от стиля «Указатель», так что выравнивание по левой границе можно указать как для каждого конкретного стиля «Оглавление ...», так и для стиля «Указатель», что повлияет на всех его наследников. По умолчанию, кстати, выравнивание по обоим краям для стиля «Указатель» не задано, режим выравнивания наследуется от стиля «Базовый». Именно выравнивание стиля «Базовый» и было изменено по сравнению со стандартным значением: для отчётов требуется выравнивание по левой и правой границам;
- если нужные стили в списке не показаны, то можно выбрать с помощью выпадающего списка в нижней части диалога указать, что надо перечислять не «Используемые стили» (в этом случае показаны только те стили, которые реально использованы в документе), а «Все стили».



На рис. 37 показаны две вкладки диалога задания свойств стилей: положение на странице (переносы и наличие разрыва страницы) и тип выравнивания абзаца. Обратите внимание на параметры слева «Не разрывать абзац», «Не отрывать от следующего» и возможность ограничения числа «висячих строк».

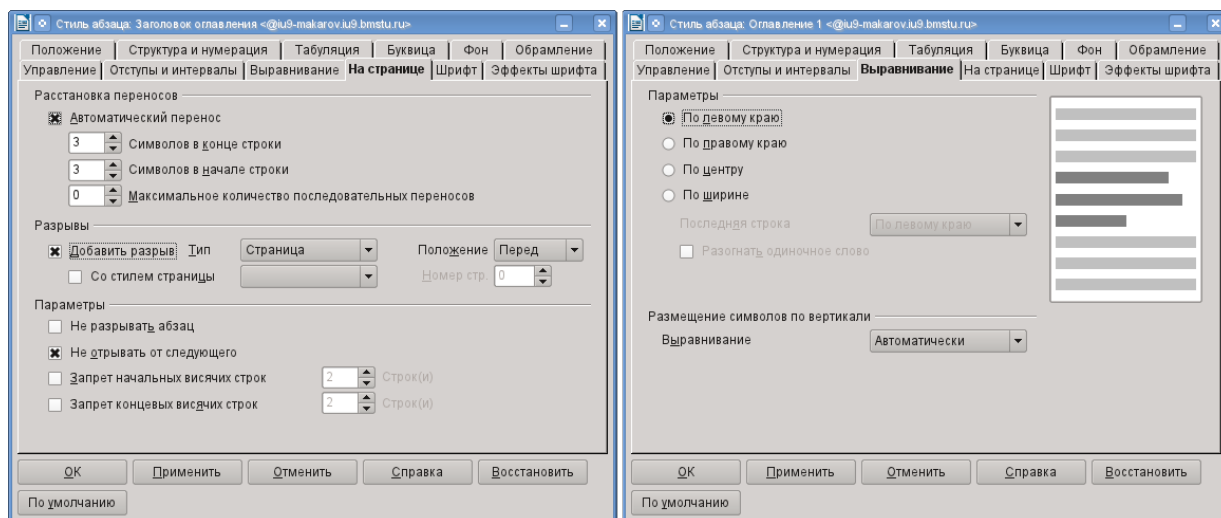


Рисунок 37: Редактирование свойств стилей (слева: положение на странице заголовка оглавления, справа: выравнивание элемента оглавления первого уровня).

Первые два параметра необходимо указывать для стилей заголовков (заголовок должен размещаться на странице целиком и следующий за ним абзац должен начинаться на этой же странице). Два последних настоятельно рекомендуется указывать для базового стиля текста: наличие отдельных т. н. «висячих строк» тоже является ошибкой (т. е. ситуаций, когда в конце страницы размещена только одна строка нового абзаца, а весь абзац на следующей странице или когда абзац почти закончился на этой странице, а на следующую перенесена одна строка). Обычно рекомендуется на странице размещать не менее двух-трёх строк абзаца.

«Красивое» размещение текста в общем случае требует, чтобы текст сразу набирался с учётом его последующего размещения на странице — если появляются висячие строки абзацев, заголовки отрываются от следующего за ними текста, в конце страницы оказывается много пустого места из-за перенесённого на другую страницу текста, надо корректировать предыдущий текст, либо сокращая, либо дописывая так, чтобы текст ровно заполнял страницы. За счёт коррекции форматирования (добавления небольших пропусков, коррекции межстрочного расстояния, коррекции отступов перед и после абзацев и т. п. допустимо обеспечивать лишь точное выравнивание в пределах одной строки. Если требуется серьёзная коррекция более чем на одну строку, то лучше изменять текст.

## Содержание

Рекомендации для выполнения домашнего задания по курсу «Архитектура компьютеров».....	1
Общие представления о домашнем задании.....	1
Справочные сведения о методах численного интегрирования.....	2
Метод трапеций.....	2
Метод средних прямоугольников.....	3
Программу рекомендуется оформить примерно таким образом:.....	5
Примерная логика выполнения каждого этапа.....	6
Попытка анализа.....	8
Дополнительные рекомендации.....	10
Получение результатов на каждом этапе.....	10
Получение графиков.....	11
Вставка графиков в отчёт.....	22
Названия рисунков, таблиц, листингов.....	25
Перекрёстные ссылки.....	28
Вставка формул.....	32
Заголовки и оглавление.....	33