инженерия программного обеспечения

*Сафонов Владимир Олегович*

# МОЛОДЫМ ПРОГРАММИСТАМ: КАК ПИСАТЬ НАУЧНЫЕ РАБОТЫ ПО ИТ

Посвящаю статью памяти моего учителя – Святослава Сергеевича Лаврова, 85 лет со дня рождения которого исполняется в 2008 году. Некоторые из рекомендаций, вошедших в статью, мне и моим коллегам в нашем молодом возрасте дал Святослав Сергеевич, о чем я всю жизнь вспоминаю с благодарностью и теплотой.

## ВВЕДЕНИЕ

Данная статья – третья из серии статей, предназначенных в первую очередь для молодых программистов, студентов и аспирантов в области ИТ, об основах их профессиональной деятельности. Первая статья данной серии [1] – о рекомендациях по переводу терминов из области ИТ, вторая [2] – о подготовке презентаций и докладов по программированию.

Цель статьи – научить студентов и аспирантов программистских специальностей правильно писать научные работы – курсовые, дипломы, статьи, тезисы докладов, диссертации – и убедить их в том, насколько им необходимо следить за работами других авторов и тем самым за развитием ИТ.

Статья может быть также полезна моим уважаемым коллегам – вузовским преподавателям.

## 1. ПОЧЕМУ НЕОБХОДИМО ЧИТАТЬ И УМЕТЬ ПИСАТЬ НАУЧНЫЕ РАБОТЫ

Не секрет, что для многих молодых программистов в нынешний период бурного развития коммерческих фирм и проектов по разработке программного обеспечения основной целью деятельности стали устройство в престижную фирму, желательно – в отделение широко известной в мире корпорации в нашем городе, получение высокой зарплаты и корпоративный карьерный рост. Это, разумеется, весьма важно (всем необходимо на что-то жить) и вполне естественно – еще 15–20 лет назад российские программисты таких возможностей практически не имели.

Однако печально видеть, когда молодежь – студенты, выпускники и даже аспиранты таких сильных и известных университетских школ, как наш мат-мех, ошибочно ограничивает самих себя только коммерческой работой, начисто забывая о повышении своего научного уровня и, как говорили учителя предыдущих поколений, расширении своего кругозора. Подобный ошибочный подход метко охарактеризовал еще Козьма Прутков: «Специалист подобен флюсу – полнота его односторонняя».

При всем моем уважении к коллегам из коммерческих фирм и к их нелегкому труду следует отметить, что тот молодой программист, кто замыкается в относи-

тельно узком кругу текущих корпоративных задач и проектов, рискует через годдва начисто забыть всю науку, которой его обучали в университете, перестать интересоваться новыми идеями, технологиями и инструментами, кроме очередной новой версии своего «родного» фирменного продукта CoolCenter версии 3.62 фирмы CoolCompany, в развитии которого он участвует (приведенные названия – вымышленные; они приведены лишь для иллюстрации проблемы).

Тем самым молодой программист превращается в узкого специалиста по одной или нескольким конкретным используемым в проекте технологиям, неуверенно себя чувствующего и комплексующего, как только речь заходит о новых технологиях в области ИТ, развиваемых другими фирмами или университетами, либо даже другим отделением своей же собственной фирмы.

Такой «специалист» не может ответить на элементарный профессиональный вопрос, не относящийся непосредственно к тому, за что он получает деньги. Иногда он настолько зашорен, закомплексован и ограничен, что бессознательно сторонится информации о новинках в области ИТ, повторяя ритуально-корпоративные заклинания типа: «Все равно наш CoolCenter лучший в мире! Его версия 3.62 несравненно лучше, чем версия 2.87, а я – самый крутой в мире программист!», – закрывая этим себе самому глаза на бурное развитие ИТ вокруг себя.

По прошествии времени, поработав годы в коммерческой фирме, он вдруг начинает ощущать голод ко всему новому, жадно поглощая, как живительный кислород, каждую статью и каждый научный семинар, который окажется в его поле зрения...

Я сознательно несколько сгустил краски, чтобы молодежь, которая будет читать эту статью, вовремя поняла, что необходимо внимательнее относиться к своим и чужим научным работам и инновациям, к своему научному и профессиональному росту, – в более широком смысле слова, чем этого требует конкретная фирма и ее конкретный проект.

Приведу реальный пример: один из моих учеников, работающий несколько лет в корпорации Microsoft в Редмонде в группе Microsoft Office, рассказывал мне, что их команда уже несколько лет использует опробованный инструмент – Visual C++ в среде Visual Studio, однако... не имеет представления о том, что такое .NET – новая перспективная платформа, разрабатываемая другими отделами той же корпорации.

Другой пример: лет пять назад в Кембридже мне пришлось объяснять одному сильному разработчику .NET CLR из Microsoft, что такое сопрограммы (coroutines).

Разумеется, все это касается отнюдь не только специалистов Microsoft, но и программистов из многих других коммерческих фирм.

Думаю, комментарии здесь излишни. Именно такой профессиональной узости следует избегать нашей программистской молодежи. Так будьте же настоящими экспертами в своей области. О том, как этого достичь,– все мои статьи данной серии.

Умение не только писать научные работы, но и следить за работами других авторов, постоянно держать себя в курсе развития ИТ, – неотъемлемая часть современной программистской культуры, как и умение правильно переводить профессиональные тексты с английского, правильно готовить презентации и делать доклады [1, 2].

## 2. ТИПИЧНЫЕ ОШИБКИ И РЕКОМЕНДАЦИИ ПО ИХ ИСПРАВЛЕНИЮ

### 2.1. НЕЗНАНИЕ ДРУГИХ РАБОТ, ОТСУТСТВИЕ ССЫЛОК НА НИХ

Не раз приходилось читать статьи молодых программистов или слушать их доклады, которые создавали впечатление, что автор буквально «свалился с Луны». Подчас молодой специалист, придумав и реализовав в своей программе какой-либо метод, сразу же энергично спешит рас-

сказать об этом всем, не позаботившись о том, чтобы сначала поинтересоваться, что же для решения аналогичных задач сделано другими специалистами, не открыл ли он Америку и не изобрел ли велосипед.

В результате, например, получается диссертация, о которой докладывал на нашей кафедре несколько лет назад молодой программист, посвященная системе автоматического синтеза программ, которая на поверку оказалась... слабым подобием классических систем ПРИЗ и NUT, разработанных еще в 1970-х и 1980-х гг. под руководством Э.Х. Тыугу (Эстония) [3]. К сожалению, новоиспеченная диссертация вообще не ссылается на эти выдающиеся работы, а автор диссертации о них даже не слышал.

Другой пример: недавно я знакомился с кандидатской диссертацией, посвященной автоматному подходу к программированию, в которой не обнаружил ссылок на известные работы 1980-х гг. И.В. Вельбицкого (Украина) по Р-технологии [4]. А ведь именно он более 25 лет назад одним из первых в СССР предложил использовать конечные автоматы как основу для архитектуры программ.

*Не открывайте Америку, не изобретайте велосипед.*

В обоих рассмотренных случаях я вовремя успел подсказать авторам диссертаций, на какие работы они должны сослаться и с какими признанными результатами других авторов они должны сравнить свои методы и системы, чтобы обосновать элементы новизны своего подхода. Однако сколько наверняка есть других работ, авторам которых опытные специалисты вовремя этого не подсказали!

Сейчас, когда есть Интернет, казалось бы, найти ссылки на другие работы гораздо легче: не нужно месяцами сидеть в библиотеках, достаточно сделать Web-серфинг в течение нескольких минут или секунд. Однако следует учитывать, что в Интернете по понятным причинам размещены в основном *новые* работы (начиная с 1990-х гг.). Классические же работы 1970-х и 1980-х гг. часто остаются вне поля зрения молодых программистов. А ведь именно их идеи положены в основу многих современных коммерческих систем. Чтобы найти ссылки на эти работы в Интернете, необходимо либо найти отечественный или зарубежный университетский сайт, на котором размещены материалы учебных курсов, ссылающиеся на эти классические работы (что далеко не всегда удается сделать, так как авторы курсов в молодости, возможно, также страдали «возрастным недугом» отсутствия интереса к классическим работам), либо все же обратиться в библиотеку и найти в ней необходимые работы – книги и журнальные статьи.

Ссылки на многие классические работы можно найти в Web-публикациях материалов моих курсов на академических сайтах фирм Microsoft и Sun. Список ссылок на Web-публикации моих курсов опубликован на сайте моей лаборатории Java-технологии [3].

Рекомендация молодым авторам:

*Необходимо ссылаться в тексте статьи, диплома или диссертации на наиболее важные труды других авторов, относящиеся к предмету Вашей работы. Необходимо ввести специальный раздел – аналитический обзор других работ, на основе которого в последующих разделах своей статьи сделать выводы о том, в чем Ваш подход опирается на другие работы и что Вы предлагаете нового, по сравнению с ними.*

### 2.2. РАЗГОВОРНЫЙ СТИЛЬ

Это одна из самых типичных ошибок любого молодого автора научных работ. Для многих работ начинающих авторов характерны выражения типа, например:

«*эту информацию надо хранить в таблице*». Здесь две ошибки, связанные с использованием «разговорного стиля». Вместо слова *эта* (например, *эта информация*) в научных работах принято использовать слово *данная* (либо *указанная*). Кроме того, разговорно-модальные выражения типа *нужно, надо, можно* в научной работе неуместны, и их рекомендуется заменять словами *следует, требуется*. Таким образом, возможный правильный вариант приведенной фразы следующий:

***Данную** информацию **следует** хранить в таблице*.

Еще более характерно для современных статей молодых авторов использование жаргонных выражений – например, *движок*. Они в научных работах недопустимы. Правильный, более научный по стилю вариант термина *движок*, – ***инструмент, утилита, драйвер*** и т. д., в зависимости от контекста.

*Не используйте разговорный стиль.*

### 2.3. ОТСУТСТВИЕ ОБОСНОВАНИЙ

Часто молодые программисты, воодушевленные своими идеями, с жаром описывают их, забывая о том, что прежде всего им необходимо *обосновать*, в чем именно новизна предлагаемого метода, чем он лучше, надежнее, эффективнее, чем другие, уже известные. В результате научная статья превращается в подобие беллетристики или необоснованной саморекламы.

Рекомендации авторам:

*В любой научной работе предлагаемый в ней новый метод, алгоритм, подход и т. д. должен быть обоснован.*

*Цель любого исследования по программированию – не просто разработать и реализовать новый подход, но и обосновать его.*

*Обоснование должно включать:*

*– во-первых, сравнение с другими аналогичными работами и вывод о том, в чем описываемый метод или алгоритм превосходит уже известные подходы и в чем опирается на них;*

*– во-вторых (если это возможно и уместно), доказательство корректности предложенного алгоритма;*

*– в-третьих, количественные оценки – оценку сложности алгоритма, экспериментальные данные, подтверждающие его эффективность, и др.*

*Без такого обоснования статья теряет смысл и научную ценность.*

Великолепным примером истинно научного описания алгоритмов является монография Д. Кнута [6].

### 2.4. РЕКЛАМНЫЙ СТИЛЬ

Необходимо помнить, что цель научной работы – обоснованное изложение нового метода, алгоритма, технологии, а не их реклама. Однако молодые авторы увлекаются тем, что в эмоциональном стиле буквально рекламируют свой подход, не сравнивая его с другими, не оценивая его применимость и эффективность и т. д. Такую статью читать не только неприятно, но и бесполезно. Реклама – двигатель торговли и бизнеса, а отнюдь не науки. Рекламный стиль в научных работах недопустим. К сожалению, он встречается не только в работах молодых авторов, но и в документации фирм по их новым продуктам, что, в свою очередь, может повлиять на молодежь, которая, вместо Д. Кнута, изберет для себя образцом для подражания в научных статьях эти рекламные «перлы».

Например, до сих пор помню рекламные утверждения авторов документации по объектно-ориентированной версии Паскаля – Турбо-Паскаль 5.5 (1989 г.): они утверждали, что впервые в мире предложили инструмент объектно-ориентированного программирования (!). С работами

1960-х гг. по языку СИМУЛА-67 и его реализации [7] им, видимо, некогда было познакомиться.

### 2.5. «ФОРМАЛИЗАЦИЯ РАДИ ФОРМАЛИЗАЦИИ» (С.С. Лавров)

Такое замечание однажды сделал Святослав Сергеевич Лавров к диссертации одного моего бывшего коллеги, который не вполне уместно ввел в свей работе искусственные формальные конструкции для описания столь простых понятий, как формат (выравнивание) и размер значения при представлении в памяти типов данных языка Паскаль.

В научных работах следует избегать искусственных формальных построений и усложнений, если суть вопроса может быть объяснена простой фразой русского языка или простым рисунком. К сожалению, у некоторых молодых программистов сложилось неправильное впечатление о том, что научная работа должна быть обязательно «заумной», «формальной», изобиловать формулами, высокопарными рассуждениями и т. д.

*Рекламный стиль в научных работах недопустим.*

На самом деле, *важная цель любой научной работы, как и любого университетского курса, – просто и доходчиво объяснить сложные понятия. В результате чтения Вашей работы читатель должен понять суть Вашего подхода и связанных с ним концепций проблемной области, а не наоборот, окончательно запутаться в них, причем по Вашей вине, поскольку Вы злоупотребили формализацией или «наукообразным» стилем. Научитесь объяснять суть излагаемых вопросов простыми доходчивыми фразами, на примерах, с использованием сравнений, аналогий, рисунков.* ***Цель научной работы – не запутывать простые понятия, а объяснять сложные.***

### 2.6. ЗЛОУПОТРЕБЛЕНИЕ АББРЕВИАТУРАМИ

В научных и технических статьях, особенно по программированию и электронике, часто используется значительное, иногда чрезмерное, число аббревиатур. Примеры наиболее общепринятых из них:

ИТ – информационные технологии, CLR – Common Language Runtime.

Более того, в зарубежных статьях и документах появились уже «аббревиатуры второго уровня» – аббревиатуры, отдельные буквы которых обозначают также аббревиатуры – например, JAX, где последняя буква X обозначает также аббревиатуру XML. На мой взгляд, *злоупотребление аббревиатурами значительно затрудняет понимание текста, особенно если аббревиатура используется не в традиционном смысле*.

Мне в 1990-х гг. пришлось встретиться с почти абсурдным примером использования сокращений. В заголовке (subject) сообщения, полученного по электронной почте из американской фирмы, значилось:

*FYI: FGN from PDE*

Необходимые пояснения:

– FYI – For Your Information (устоявшееся сокращение, используемое в американских электронных письмах);

– FGN – Friday Good News (психологический прием американских менеджеров, которые по пятницам рассылают сотрудникам «еженедельные хорошие новости» для поднятия их настроения и повышения производительности);

– PDE – Product Development Environment (имеется в виду группа сотрудников фирмы, отвечающая за разработку инструментального окружения программного проекта).

Разумеется, данный пример – *не образец для подражания*. Привожу его для того, чтобы молодежь *никогда* в своих статьях не уподоблялась такому стилю.

Между прочим, даже сами пресловутые технические аббревиатуры тоже имеют свои названия, причем, разумеется, тоже сокращенные: *TLA – Three-Letter Acronyms*. Одна из фирм по моей просьбе прислала мне специальный фирменный глоссарий таких сокращений.

Так не проще ли их избегать, чтобы не пришлось Ваши научные статьи читать с использованием глоссария, расшифровывающего Ваши таинственные «тарабарские» аббревиатуры?!

Кстати, если Вам приходится использовать нетрадиционные сокращения в работе, то Вы обязательно должны в конце статьи или книги привести глоссарий, в котором каждое из них расшифровано.

*Не запутывайте простые понятия, объясняйте сложные.*

Еще пример уже из российской реальности. В 1985 году мой аспирант, защищавший диссертацию по реализации языка Снобол для МВК «Эльбрус», в первоначальном варианте своей диссертации, до получения моих замечаний, использовал следующее сокращение:

*Сборка Мусора – СМ.*

Почему оно недопустимо? Как хорошо известно программистам моего и более старших поколений, сокращение СМ (точнее, СМ ЭВМ) использовалось для обозначения популярной серии мини-ЭВМ, выпускавшейся в СССР (машины этой серии были «клонами» мини-ЭВМ фирмы PDP). Если сокращение СМ использовать для обозначения совсем другого понятия, это может просто запутать читателя.

С моей точки зрения, аббревиатур вообще лучше избегать, либо, в крайнем случае, использовать только привычные, устоявшиеся аббревиатуры, например, ИТ и CLR (причем последнюю из аббревиатур – только в работах, касающихся платформы .NET).

### 2.7. «ГЛУХОЙ» ТЕКСТ

Этот удачный, на мой взгляд, хотя и разговорный термин взят из давней рецензии одного известного специалиста на первый вариант одной из моих ранних статей в журнале «Программирование».

Имеется в виду следующая ошибка. Иногда, увлекаясь изложением своего метода, включающего нетривиальные способы представления данных и алгоритмы, молодой автор описывает его, используя *только один сплошной текст, без единого примера кода, без единого рисунка или схемы*. И так на 10–15 (!) страницах своей научной работы. Разумеется, подобный текст вообще практически невозможно воспринимать. Любой научный руководитель или рецензент в такой ситуации просто потребует от автора добавления в текст примеров и иллюстраций.

Особенно это касается научных работ по ИТ. Не мыслю себе, как, например, можно объяснить какой-либо метод распределения памяти или сборки мусора без использования рисунков, поясняющих особенности метода.

Рекомендация авторам:

*Не допускайте «глухого» текста в своих научных работах. Обязательно иллюстрируйте описание методов и алгоритмов примерами кода и рисунками. Логику работы программы и ее использования целесообразно также иллюстрировать образами экрана (screenshots).*

### 2.8. ИЗЛОЖЕНИЕ «ОТ ЦАРЯ ГОРОХА»

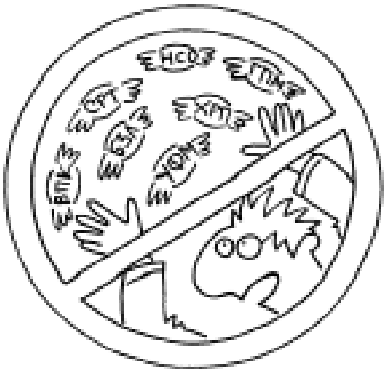
Разумеется, я сторонник кратких исторических обзоров в научных статьях. Они помогают лучше осознать ход развития ИТ в целом и способствуют более правильному взгляду на описываемую проблему. Однако и при таком подходе необходимо соблюдать чувство меры.

*Введения и исторические обзоры не должны излагать историю вопроса «от царя Гороха». Иначе молодой автор рискует вообще никогда так и не приблизиться к конкретному предмету своей статьи.*

Например, один из моих учеников прислал мне предварительный вариант своих тезисов доклада о разработке компилятора с использованием технологии Phoenix [7] объемом всего в одну страницу, которые начинались примерно так (я немного утрирую):

«*С давних времен в истории человечества большое внимание уделялось вычислениям. С появлением компьютеров...*» *и т. д.*

Разумеется, подобный стиль в научных работах лишен всякого смысла. Каждая работа пишется на конкретную тему. Поэтому в ее вводной части следует *кратко изложить историю рассматриваемой проблемы, а не историю ИТ, России, человечества или Вселенной.*



*Не злоупотребляйте аббревиатурами.*

### 2.9. ОШИБКИ ОРФОГРАФИИ И ПУНКТУАЦИИ

Увы, приходится писать и об этом. Большинство статей молодых программистов, присылаемых научным руководителям или рецензентам, содержат ошибки русской орфографии и пунктуации. Позволю себе опустить вполне уместные здесь рассуждения о качестве преподавания русского языка в наших средних школах и дам конкретные рекомендации в контексте примеров по ИТ.

Вот наиболее часто встречающиеся в статьях ошибки (за разъяснениями и правилами отсылаю молодых авторов к учебникам русского языка):

• Правописание *-тся* и *-ться*. Приведу несколько примеров типичных правильно написанных фраз которые, надеюсь, помогут молодым программистам избежать подобных ошибок:

*После инсталляции драйвера* ***приходится*** *перезагружать операционную систему* (мягкого знака нет).

*Ссылка на список управления доступом* ***хранится*** *в заголовке файла* (мягкого знака нет).

*Каждая программная система, используемая в сети, должна* ***защищаться*** *от сетевых атак* (мягкий знак обязателен).

• Правописание *также / так же*. Пример:

*Данный алгоритм* ***также*** *имеет сложность O(n) и работает почти* ***так же****, как и предыдущий.*

В первом случае ***также*** пишется слитно, во втором (***так же***) раздельно.

• Отсутствие запятых в сложносочиненных предложениях, в причастных и деепричастных оборотах. Например:

*На все работы других авторов, упоминаемые в тексте Вашей статьи, необходимо* ***ссылаться****.*

В данном предложении использован причастный оборот, который необходимо выделять *двумя* запятыми. К сожалению, *большинство* молодых авторов забывает поставить запятую *после* причастного оборота в подобных случаях. Пример служит также иллюстрацией правописания *-тся и -ться*.

Рекомендации молодым авторам:

• *Если чувствуете в этом необходимость, повторите правила русского языка по учебнику или самоучителю.*

• *В сомнительных случаях проверяйте себя по орфографическому словарю.*

• *Используйте русскую версию Microsoft Word, в которой включите контроль орфографии (spell check) и пунктуации. При этом неправильно написанные слова Word будет либо подчеркивать красной чертой, либо исправлять сам; предложения с неправильной пунктуацией Word будет подчеркивать зеленой чертой.*

Пожалуйста, помните, что *неграмотно писать студенту или аспиранту университета, мягко говоря, не вполне удобно. Ваши научные руководители – профессора и доценты – не должны заниматься исправлением Ваших ошибок в русском языке. Поверьте, у них есть много других дел.*

Но все это вполне поправимо – стоит лишь немного улучшить свой русский язык, а не только английский [1].

## 3. РЕКОМЕНДАЦИИ ПО СТРУКТУРЕ, СОДЕРЖАНИЮ И СТИЛЮ НАУЧНЫХ РАБОТ

Теперь суммирую свои рекомендации и привожу примерную струк-туру научной работы в целом и рекомендации по ее стилю и содержанию.

Порядок приведенных ниже пунктов 3.1–3.7 соответствует рекомендуемому порядку частей (разделов) Вашей научной работы.

### 3.1. НАЗВАНИЕ РАБОТЫ (ЗАГОЛОВОК)

Название должно быть по возможности кратким, не слишком общим, лишенным сокращений (кроме общепринятых и самых необходимых), рекламно-эмоциональных и общих слов.

Например, возможное название статьи, посвященной новому методу сборки мусора для .NET, может быть следующим (приведенный пример является вымышленным):

Метод сборки мусора для платформы .NET с использованием многоядерной архитектуры Intel Core Duo.

Неправильный вариант названия статьи на ту же тему:

*Новые эффективные методы сборки мусора для современной платформы Microsoft.NET на базе многоядерных архитектур*.

*Не допускайте «глухого» текста.*

Ошибки во втором варианте названия:

– название слишком общее;

– в названии присутствуют эмоционально-рекламные выражения (новый, эффективный, современный), не имеющие никакого смысла, так как о «не новых», «не эффективных» и «не современных» методах не пишут научные работы;

– не указано конкретное название процессора;

– наоборот, указана *конкретная реализация* стандарта .NET фирмы Microsoft, в то время как существует еще несколько реализаций .NET (SSCLI, Mono и др.), к которым данный метод также может оказаться применимым.

*Не следует забывать кроме названия в начале статьи указывать свои фамилию, имя, отчество, место работы или учебы (например, Санкт-Петербургский университет) и адрес электронной почты.*

Последнее, казалось бы, очевидно, однако почему-то почти каждый молодой автор, присылающий свою статью, в ее первом варианте это сделать забывает.

### 3.2. АННОТАЦИЯ (ABSTRACT)

*Аннотация* – это краткое описание содержания статьи. Рекомендации к ней: объем – 1–2 абзаца, не более. Например:

*В статье описывается метод параллельной сборки мусора для платформы .NET, использующий возможности многоядерной архитектуры процессора Intel Core Duo. В отличие от известных методов, предлагаемый метод позволяет параллельно выполнять подсчет ссылок на объекты и освобождение неиспользуемой памяти. Метод применен в системе MyDotNet – экспериментальной реализации стандарта .NET для архитектуры Intel Core Duo, разработанной в Энском университете на кафедре информатики.*

В аннотации недопустимы какие-либо рассуждения и эмоциональные высказывания, главное требование к ней – *краткость* и *ясность*. Необходимо, чтобы на основе лишь одной аннотации, не читая саму статью, любой специалист мог получить общее представление о том, чему посвящена статья, и решить, стоит ли тратить время на ее изучение.

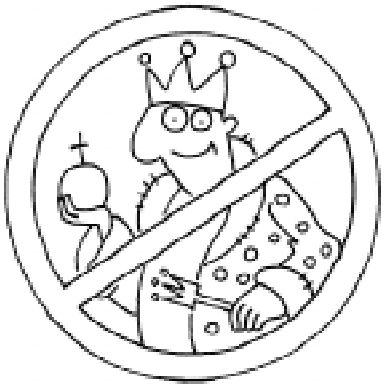
Часто, кроме аннотации, в начале статьи, согласно требованиям журнала, необходимо привести также список *ключевых слов (keywords)*. Для рассматриваемого примера статьи список ключевых слов может иметь вид:

*Ключевые слова: системное программирование, управление памятью, сборка мусора, параллельное программирование, .NET, многоядерные процессоры, Intel Core Duo.*

### 3.3. ВВЕДЕНИЕ

Во введении должна быть описана *постановка задачи*, должен быть дан *аналитический обзор других работ* по данной теме и должны быть кратко сформулированы и обоснованы *актуальность, новизна и научная ценность предлагаемого в статье подхода, по сравнению с другими*. Также во введении рекомендуется дать *определения*, по крайней мере, большинству основных терминов, используемых в статье. Определения могут быть словесными или с использованием формул, в зависимости от специфики работы.

*Не допускается использование в статье нетрадиционных терминов без (или до) их определений, которые как раз и должны быть даны именно во введении. Допускаются определения и в других частях работы перед первым использованием термина.*



*Не излагайте «от царя Гороха».*

### 3.4. ОСНОВНАЯ ЧАСТЬ

Поясню на всякий случай, что заголовок данного пункта моей статьи *не должен быть названием раздела (пункта) Вашей научной работы*. Содержательно ее основную часть должны составлять один или несколько *разделов*, излагающих суть нового подхода, которому посвящена статья.

Каждый *раздел* (например, в моей статье это разделы 1, 2 и т. д.) может состоять из *подразделов (пунктов)*, например, 1.1, 1.2 и т. д.

Не рекомендуется использование более глубокой нумерации пунктов – 1.1.1, 1.1.2 и т. д., – это может запутать читателя.

*Не рекомендуется также использование разделов или пунктов без номеров, поскольку на непронумерованные части работы неудобно ссылаться.* Например, для ссылки на рекомендации относительно названий статей достаточно лишь указать номер пункта:

*Рекомендации относительно названий статей даны в пункте 3.1.*

В статье *необходимы ссылки на все другие упоминаемые в работе методы, алгоритмы, системы и т.д., принадлежащие другим авторам.* Например:

*В работе [1] описывается метод сборки мусора для платформы .NET на основе концепции поколений (generational garbage collection).*

*В конце статьи необходимо привести список литературы* (в русскоязычных статьях он обычно носит название *Литература,* в англоязычных – *References*). Порядок работ в списке литературы должен, как правило, соответствовать *текстуальному порядку встречаемости ссылок в тексте работы*. Например, текстуально первая ссылка должна иметь номер [1], вторая – номер [2] и т. д.

Если Вы ссылаетесь сразу на несколько работ, допустимая форма подобной ссылки следующая:

*В работах [1, 2] описаны... В работах [3–5] рассмотрены...*

Таким образом, при ссылке на две или более работ их номера разделяются запятыми, при ссылке на диапазон работ в списке литературы начальный и конечный номера работ разделяются дефисом.

Некоторые журналы и издательства требуют от авторов другого принципа нумерации ссылок – например, расположения их в списке литературы по алфавиту. В таком случае (только для статьи в данный журнал), разумеется, необходимо выполнить его требования.

Существуют и другие принципы организации ссылок, но я бы не стал их рекомендовать.

Один из них, например, заключается в том, что в качестве ссылки используется не номер, а сокращение от фамилии автора и две последние цифры года публикации. Однако это может привести буквально к анекдотическим сокращениям, например:

*[Непей 77]* – возможная ссылка на работу известного российского логика Н.Н. Непейводы 1977 г. Заранее приношу уважаемым коллегам извинения за данный пример: мне пришлось его придумать для иллюстрации молодым авторам тех проблем, которые могут быть вызваны подобной формой ссылок;

*[Стой 80]* – ссылка на работу по типам данных Стоя (Stoy) 1980 г. Данная ссылка, увы, не вымышлена, а действительно встретилась в одном известном обзоре по абстрактным типам данных 1980-х гг.

Никогда не использую подобный метод организации ссылок. Не сочтите за излишнюю щепетильность, но мне в нем видится недостаточное уважение к авторам, на которых таким образом ссылаются.

*Не забывайте о том, чтобы иллюстрировать свое изложение примерами и рисунками (см. п. 2.7).*

*Стиль* научной работы должен быть, с одной стороны, достаточно строгим, без жаргона, разговорных выражений и анекдотов (однако примеры из практики приводить необходимо).

С другой стороны, стиль должен быть достаточно живым, хотя и в разумных пределах. Работу не должно быть скучно читать. Например, одну из книг по новому языку программирования (кстати, написанную очень талантливым и уважаемым мной автором) я использовал... для отдыха в электричке, пока ехал из Петродворца до Санкт-Петербурга: чтение книги с любого места в течение двух-трех минут гарантированно вызывало у меня... крепкий здоровый сон.

*Не забывайте правил русского языка.*

Научные работы в подобном стиле писать, конечно, не следует. Читать Вашу работу должно быть интересно. Поэтому, пожалуйста, *когда пишете научную работу, помните не только о своих заслугах и результатах, но и о своих читателях.*

Есть хороший рецепт для улучшения стиля научных работ: *читайте классиков – не только классиков в области ИТ, но и классиков русской прозы и русской поэзии.* Это поможет улучшить Ваш научный стиль и язык Вашего изложения.

В области ИТ рекомендую прежде всего чтение работ таких авторов, как Н. Вирт, Э. Дейкстра, Д. Кнут, А. Ахо, Ч. Хоар, А.П. Ершов, С.С. Лавров.

*Не злоупотребляйте эпиграфами и посвящениями.* Эпиграф, на мой взгляд, более уместнен в беллетристике, поэзии, исторических и философских работах, а отнюдь не в работах по программированию.

Если необходимо сослаться на афоризм классика в области ИТ, следует ссылаться на ту его работу, в которой он приведен, а не выносить его в качестве эпиграфа.

Злоупотребление эпиграфами может резко оттолкнуть читателя, который хочет найти в книге или статье научное содержание, а не излишние эмоции и амбиции автора, выраженные в эпиграфах.

Что касается посвящений, то мне бывает грустно читать в переводах американских книг посвящения типа:

*Большое спасибо моей Мэри, которая терпела мою работу по написанию этой книги целый год...*

Право же, в данной ситуации следует пожалеть и бедную Мэри, и самого автора, а подобных «горе-посвящений» в своих книгах не допускать.

Уместными считаю посвящения книг своим учителям. Впрочем, что касается книг, то альтернативным вариантом посвящения может быть благодарность своему учителю, которая уместна в специальном разделе *Благодарности (Acknowledgements)* в начале книги. Злоупотреблять посвящениями также не стоит.

*Каждое утверждение, каждый метод, каждый алгоритм, каждый вывод в Вашей работе должны быть обоснованными, то есть должны быть либо теоретически доказаны (для теоретических работ), либо подвергнуты количественному и сравнительному анализу, по результатам которого и должен быть сделан обоснованный вывод о применимости и эффективности метода.*

*Для программ количественный анализ – это пропуск специальных тестов для измерения их производительности (benchmarks) и анализ их результатов; использование метрик для измерения сложности и надежности программ и трудоемкости их разработки и т. д.*

*Голословные, необоснованные утверждения в научных работах недопустимы.*

### 3.5. ЗАКЛЮЧЕНИЕ

*Заключение обязательно должно присутствовать в каждой научной работе.*

Заключение должно содержать:

*– краткую формулировку (summary) результатов, описанных в работе,*

*– описание уже имеющегося или будущего практического применения результатов работы,*

*– краткое описание перспектив дальнейших исследований по данной теме.*

### 3.6. СПИСОК ЛИТЕРАТУРЫ

*Список литературы, как и заключение, обязательно должен присутствовать в каждой научной работе.*

В пункте 3.4 описана рекомендуемая форма ссылок на литературу.

Если ссылок на другие работы в первом варианте статьи нет, это говорит о неопытности автора и о его недостаточном знании предмета. Ситуация, когда ссылки в статье вообще не нужны, практически невозможна – каждое исследование в чем-то опирается на другие.

Помните, что до Вашей статьи по данному вопросу публиковали статьи и другие авторы, причем для многих областей ИТ (например, по компиляторам) – в течение более 50 лет.

### 3.7. ПРИЛОЖЕНИЯ

В качестве приложений к статье, книге, дипломной работе, диссертации по ИТ вполне целесообразно привести *примеры кода, результатов его выполнения, графические схемы (например, UML-диаграммы) архитектуры программных систем, таблицы с результатами измерения и сравнительного анализа программ* и т. д.

Если такого рода иллюстративный материал (особенно длинные примеры кода) размещен непосредственно в тексте работы, это может затруднить ее понимание. Рекомендую проиллюстрировать в тексте статьи описываемую концепцию небольшим фрагментом кода в 5–10 строк, а более содержательный пример кода на эту же тему поместить в приложение.

### 3.8. РЕКОМЕНДУЕМЫЙ ОБЪЕМ РАБОТЫ

Очень многие молодые авторы спрашивают, каков минимальный и максимальный объем научной статьи, дипломной записки, диссертации.

Могу рекомендовать следующее:

– объем студенческого реферата – 20–30 страниц;

– объем дипломной записки – 100–120 страниц;

– объем исследовательской научной статьи (о предлагаемом новом подходе, методе, алгоритме и т. д.) – 10–15 страниц;

– объем обзорной статьи – 20–30 страниц (как и реферата);

– объем кандидатской диссертации – 100–150 страниц.

Разумеется, приведенные цифры ни в коем случае не следует воспринимать как догму.

Все зависит от темы и от ситуации.

Однако, например, дипломные записки объемом в 10–20 страниц я просто не принимаю и требую от авторов более подробного описания других работ или своей системы. Как правило, в подобной ситуации автор просто не успел (либо не захотел) включить какую-либо обязательную часть в дипломную записку.

Желаю молодым программистам успеха в написании научных работ, а научным руководителям – интересных и талантливых статей ваших учеников.

Ваши отклики на статью присылайте, пожалуйста, по электронной почте: *v_o_safonov@mail.ru.*

**Литература**

1. *Сафонов В.О.* КПП: Коллекция Правильных Переводов // Компьютерные инструменты в образовании, 2007, № 4.

2. *Сафонов В.О.* Как подготовить презентацию и сделать доклад по программированию // Компьютерные инструменты в образовании, 2007, № 5.

3. Web-сайт лаборатории Java-технологии НИИ математики и механики Санкт-Петербургского университета. http://polyhimnie.math.spbu.ru

4. *Тыугу Э.Х.* Концептуальное программирование. М.: Мир, 1984.

5. *Вельбицкий И.В.* Технология программирования. Киев: Наука, 1984.

6. *Кнут Д.* Искусство программирования. Том 1. Основные алгоритмы. М.: Мир, 1977.

7. *У. Дал, К. Нюгорд.* Универсальный язык программирования СИМУЛА-67. М.: Мир, 1968.

8. Web-сайт Microsoft Phoenix. http://research.microsoft.com/phoenix



***Сафонов Владимир Олегович,***
***доктор технических наук, профессор кафедры информатики СПбГУ, руководитель лаборатории Java-технологии.***