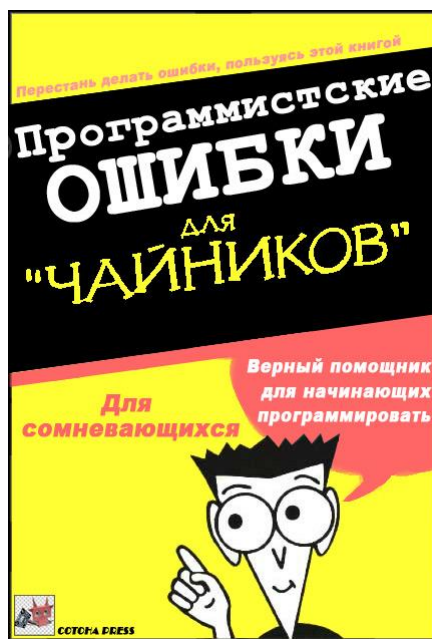


ТИПИЧНЫЕ ОШИБКИ НАЧИНАЮЩИХ ПРОГРАММИСТОВ

Конспект лекции "Типичные ошибки начинающих программистов", прочитанной в [ИНЖЕке](#) 29 Сентября 2008г. Думаю, будет полезна и всем остальным студентам... а может и не только.



ВСТУПЛЕНИЕ

Привет всем. Вы все тут собрались, чтобы прослушать лекцию на тему «типичные ошибки начинающих программистов». Некоторые из вас, наверняка, собираются вынести из следующих полутора часов некоторую полезную информацию (я это вижу по тому, что они приготовились записывать), а остальные пришли для галочки 😊. В любом случае начинаем.

О ЛЕКТОРЕ

Для начала обо мне. Я (как вам, наверное, уже должны были сказать) работаю в компании NIX Solutions Ltd. на должности директора по обучению и развитию персонала. Кроме того я успел поработать php-разработчиком, менеджером отдела продаж, начальником отдела продаж, менеджером проектов, начальником отдела php и начальником отдела .net. Это чтобы вы понимали, что эта моя лекция про ошибки начинающих программистов, которую меня попросили вам прочитать, не просто теория очередного менеджера о том, что должны делать идеальные программисты. Всё что я расскажу – суровая правда жизни. Я сам совершал многие из этих ошибок, и я постоянно вижу, как их совершают новички у нас на фирме.

МОТИВАЦИЯ

Эта лекция призвана вам помочь не совершить некоторые ошибки, которые совершать вовсе не обязательно. Все ведь помнят поговорку про то, что, по возможности, надо учиться на чужих ошибках. Поэтому, кроме перечисления самих ошибок, я попытаюсь дать некоторые рецепты, как их избежать.

Ошибок, которые совершают начинающие программисты, очень-очень много. Чтобы не перечислять их все (на что не хватит времени), я решил попытаться их сгруппировать и рассмотреть наиболее типичные группы.

ОШИБКИ

Я решил разделить ошибки на детские, тактические и стратегические. Такое разделение больше применительно для студентов, пишущих лабы, курсовики и дипломы, чем для работников IT-контор, участвующих в реальных проектах. Но, к сожалению, и среди последних встречаются те, к кому это можно приложить. Может это потому, что работники IT-контор часто являются студентами? ☺

ДЕТСКИЕ

Это самые заметные ошибки, т.к. они видны прямо в коде программы и/или в результатах её работы. Они просто бесят сорабтников и менеджеров проектов.

- 1. Невнимание к деталям.** Едва ли не основным признаком профессионала является внимание к деталям. В профессиональном приложении вы никогда не увидите ничего из ниже перечисленного:
 - **Недоформатирование текстов/Недовыравнивание элементов форм.** Куски лейбочек, который вылезает за панели; кнопки, которые перекрывают лейбы; тексты, которые уползают за край формы.
 - **Отсутствие единообразия в интерфейсе, форматировании текстов, расположении форм.** Это тот случай, когда лучше «безобразно, но единообразно». Это же, кстати, отличает айтишников от всех остальных при использовании MS Word: айтишники используют стили, и потом могут легко менять вид документа, изменяя стили, а остальные каждый элемент форматируют вручную и при необходимости изменить вид заголовка второго уровня переформатируют все заголовки 2го уровня в доке.
 - **Не учитывание возможности изменения размеров формы.** Самый обычный вариант тут - просто забыть о том, что формы могут меняться, забыть установить значения минимальной и максимальной ширины и высоты. Самый некрасивый и неюзабельный - задать абсолютную привязку ко всем сторонам формы, так что контролы наедут друг на друга при её увеличении. Также часто забывают запретить "распахивать" окно, хотя запрещают "ресайз".
 - **Использование названий контролов, которые за вас придумала IDE.** Button1, Form2, открытьИзображениеToolStripMenuItem_Click. Этого делать нельзя, т.к. радикально ухудшает читабельность кода, а значит, увеличивает стоимость его поддержки. Вообще «неговорящие» названия контролов плохо. Говорящие по-русски - спорно, т.к. часто вы будете писать приложения для зарубежных заказчиков, а это означает, что русские названия методов не пройдут процедуру приёмки исходного кода.
 - **Сортировка числовых значений по правилам строк.** Это часто происходит, когда значения хранятся в виде цифр (15.45), а отображаются в строковом виде (15грн.45коп.). И сортировки выполняются средствами стандартных компонентов. Надо это учитывать и реализовывать сортировки как надо.
- 2. Смешение парадигм процедурного и объектно-ориентированного программирования.** К сожалению, на собеседованиях я очень редко встречаю претендентов, которые бы могли внятно объяснить, чем ООП лучше. Более того – мало кто может сказать, лучше, чем что. Естественно, что не понимая, зачем ООП нужно, оно применяется неправильно.
 - **Применение ООП для отмазки.** Чем применять ООП «для галочки», лучше его не применять вообще, т.к. с одной стороны выглядят эти попытки жалко, а с другой запутывают ваших товарищей, которые ожидают «правильного» использования ООП, раз уж оно вообще использовано.
 - **Использование «глобальных переменных».** Особенно в виде «Form1.condition». Это снижает читабельность кода и является симптомом того, что вы неправильно спроектировали приложение.
 - **Использование статических методов, для решения бизнес-задачи.** Если у вашего класса есть только поведение (методы), но нет свойств (полей), значит, вы неправильно провели анализ предметной области (не всегда, но чаще всего).
- 3. Неправильные комментарии.** Комментарии - это отличный помощник, если их использовать по назначению. Многие игнорируют этот факт.
 - **Не писать вообще.** Cowboy-style. Хорошо, что преподаватели стараются с этим бороться... и порождают следующую проблему.
 - **Писать очевидные комментарии.** Когда что-то делается из-под палки, это всегда делается плохо. Так и тут - особенно умиляют комментарии вида `int a; // объявляем переменную`. Выглядит как издёвка (особенно, если через две строчки следует алгоритм без единого комментария с кучей циклов, ифов, break'ов и с несколькими return'ами), а издеваться над сотрудниками (и преподавателями) мало того, что нехорошо, так ещё и чревато.

- **Не описывать метод/класс.** С этим старается бороться IDE. Не надо ей мешать - она знает, что делает. Автодоки сохраняют время всей команде и вам лично, когда приходит время писать документацию на проект.
4. **Неследование code conventions.** Стандарты кодирования позволяют вам разобраться в чужом коде (или кому-то в вашем) намного быстрее и легче. Обычно программисты подсознательно следуют какому-то одному стилю, но встречаются и такие у которых 7 пятниц на неделе. В любом случае, я советую явно выбрать подходящий стиль и стараться ему следовать. По умолчанию можно использовать стандарты кодирования pear для php, sun для java и Microsoft для семейства .net.
- **Хаотичное форматирование кода.** Самый клинический случай, это когда у одного и того же человека разные куски кода отличаются по форматированию. Некоторые даже говорят, что это признак маниакально-депрессивного психоза. Ну, или ему просто курсовик помогали писать соседи по общежитию ☺
 - **Каждый член команды использует удобную ему конвенцию.** Это менее плохо, но тоже не хорошо. Для работы в команде просто необходимо принять общий стандарт кодирования, иначе это будет не работа в команде, а просто работа над одним проектом.

Детские ошибки можно перечислять и перечислять, но я ограничился только теми ошибками, которые я встретил в ваших работах больше 3х раз.

ТАКТИЧЕСКИЕ

Тактические ошибки – это ошибки, которые не напрямую связаны с кодом, а больше связаны с краткосрочными целями, которые преследует программист, при его написании.

1. **Не иметь правильной цели для проекта.** Первой и главной целью при написании курсовых, лаб и дипломов необходимо ставить «поднять свой уровень», а не «отмазаться от преподавателя». Подумайте, ведь в большинстве случаев, когда вы придёте на собеседование, эти программы будут единственным, чем вы сможете похвастаться – может стоит сделать их хорошо?
2. **И так сойдёт.** Отсутствие правильно цели приводит к тому, что программы делаются «тяп-ляп», как у зайца в старом мультике (все помнят?). В двух словах – на пользу это ему не пошло. Можно выделить несколько подпунктов:
 - **Размещение бизнес-логики в обработчиках событий.** Это наверное самая распространённая ошибка, которую совершают абсолютно все. Можем провести эксперимент – пусть честные люди поднимут руки, если они помещали решение задачи в какой-нибудь «onClick». Я так и думал... честных людей мало ☺. Чтобы побороть эту ошибку есть теоретически простой, но сложный в реализации метод... но полезный: сначала нужно написать консольное приложение, а потом докрутить к нему GUI.
 - **Тестирования приложения только на допустимых значениях.** Удивительно, как много приложений падает, как только ввести в поле для ввода цифр буквы. Когда вы пишете приложение, необходимо рассчитывать не на дружелюбного пользователя, а на агрессивную обезьяну, которая может ввести что угодно и нажать куда угодно. Приложение должно на всё реагировать корректно.
 - **Хардкод.** Внесение конфигурационных параметров, строковых литералов, магических цифр в код - это плохо. Для этого давным-давно придуманы специальные средства – конфигурационные файлы, файлы ресурсов, формы настроек. По возможности необходимо избегать перекомпиляции приложения, для его настройки, т.к. в реальном мире, за перекомпиляцией следует фаза развёртывания, а это очень дорого.
 - **Копи-паст.** Дублирование кода увеличивает стоимость его поддержки – ведь в любом коде будут производиться изменения, это данность. От этого никуда деться нельзя, а значит надо будет вместо одного изменения производить много (в зависимости от количества скопированных кусков) и ещё и синхронизировать их. За правило надо взять принцип, что если какой-то кусок кода понадобился больше двух раз, то его надо выделить в метод, класс или модуль.
 - **Плохой русский/украинский/английский в сопроводительной документации.** Документация - это неотъемлемая часть проекта. Если документацию неприятно читать, то это отношение распространится на весь проект в целом. Что я имею в виду? Например, откровенные глупости в части про БЖД, которые кочуют из записки в записку (я уже говорил, что копи-паст это плохо? на документацию это тоже распространяется). Или излишне вольное обращение с украинским языком, например «разрешение монитора», это «роздільна здатність», а не «дозвіл», как практически у всех вас в записках.
3. **Не соответствие результата и цели.** Если реализовывать что-то "тяп-ляп", то и результат получается посредственный. Любая программа решает некую бизнес задачу - вот реализация этой бизнес-задачи и есть цель вашего проекта, а значит и результат должен быть этой же реализацией. Например, если вам дали курсовой на тему «однопользовательская игра с графическим интерфейсом», то вам надо сконцентрироваться на игре, на её динамике, на удобстве интерфейса, на функциях, характерных для игр – тогда в результате у вас будет (сюрприз!) интересная однопользовательская игра. Но в большинстве случаев студенты концентрируются на том, чтобы просто сдать курсач. В итоге получается скучная игра, которой неудобно пользоваться, обёрнутая в стандартный

для windows-приложений интерфейс – стандартный проект-отмазка. Это не правильно. В первую очередь надо решить поставленную задачу – это позволит вам повысить свой уровень в программировании и анализе, и как приятный побочный эффект позволит вам получить высокую оценку за неё.

- **Недостаточная проработка теоретической части проекта.** Очень часто встречаются проекты, в которых не понятна задача, которую они решают. Не секрет, что большинство студенческих проектов - это некие интерфейсы к базам данных. Например, учёт книг в библиотеке, учёт шахт на Украине, учёт студентов в деканате. Это *разные* цели. А результат у всех *один и тот же* – они, по сути, они как братья близнецы - формы редактирования, поиска и добавления новых записей. ЭТО НЕ ИНТЕРЕСНО. И это не правильно. Если вы хотите добавить интереса в ваш проект, то опишите, какие насущные задачи решает проект, какие реальные процессы он автоматизирует. Не закливайтесь на добавлении и редактировании записей – ведь в большинстве случаев руками никто ничего добавлять не будет, т.к. базы данных уже есть и их надо просто импортировать. Редактирование не такой частый процесс - разве что кто-то выйдет замуж и сменит фамилию (если говорить про учёт студентов). Сконцентрируйтесь на правах доступа, на вариантах поиска, на том, что реально нужно. Сходите в деканат и спросите, чего нужно его работникам. Это зачтётся.
4. **Обман.** Да самый обычный обман пользователей (в частном случае преподавателей).
- Если в приложении заявлена некая функциональность, то её или надо реализовать в соответствии с требованиями или в сопровождающей документации указать на ограниченность реализации. Последний пример, который приходит на ум это «экспорт в формате MS Excel». Один из ваших товарищей реализовал это в виде генерации файла значений, разделённых пробелом, которому назначалось расширение XLS. Эксель, конечно, его открывал и автоматически распарсивал, но по сути это был обман, т.к. другие офисные системы, которые читают формат XLS, его уже не понимали. Обман производит дурное впечатление, хотя одна всего лишь фраза в документации о том, что эта функция реализована в ограниченном объёме могла бы всё исправить.
 - Не использовать/неправильно использовать ООП в курсовом по предмету "объектно-ориентированное программирование" – это тоже обман.
 - Покупать лабы/курсовые/дипломы – это тоже обман. Хорошо, если вы не собираетесь становиться программистом - тогда вы обманываете только преподавателя. Но если у вас есть такие планы, то вы обманываете ещё и себя.

СТРАТЕГИЧЕСКИЕ

Стратегические ошибки - это ошибки стиля жизни. Они связаны с долгосрочными целями или – чаще – с их отсутствием ☺

1. **Не знать, кем и почему вы хотите стать.** Это основная ошибка студентов. Они думают, что до окончания ВУЗа, этот вопрос ещё не актуален. А на самом деле потом, а может быть уже и сейчас, будет поздно. Первый вопрос, который я задаю на собеседованиях: «Почему ты решил стать программистом / тестировщиком / аналитиком?» И с грустью не слышу ничего вразумительного в ответ. Это важно, прежде всего, для вас - если вы НЕ понимаете, почему вы хотите стать программистом, то может быть вы вовсе и не хотите им становиться? Может вы модельер, а тут только тратите своё время? Подумайте...
2. **Не пытаться улучшить свой скил в программировании.** Второй вопрос на собеседовании, который я задаю всем: «Что ты делаешь, чтобы стать тем, кем ты хочешь стать?» Опять же очень не часто, мне могут ответить что-то кроме «учусь на соответствующей специальности». Этого мало.
 - **Не читать книг по теме.** Мне это странно, но не многие читают книги, а надо - там много полезного. Кто сходу может назвать автора какой-нибудь книги по .NET? А ещё одного? А третьего? На первый вопрос ответило 10 человек из 60, на второй 3, а на третий ни одного...
 - **Не писать программ, помимо лаб, курсовых и дипломов.** Это мне странно больше всего. Если вы хотите стать программистом, то почему не пишете программ? На собеседованиях некоторые на вопрос "почему программирование?" задумываются и говорят что-то вроде: "я люблю творить". Так почему не творишь? Ну да, я понимаю, что конъюнктура рынка сейчас такая, что IT-специальности популярны и многие идут туда ради приличного заработка, а не тварьбы. Но тогда возникает вопрос – почему вы не выполняете курсовые и дипломы качественно и профессионально? Ведь чем вы лучше себя покажете на собеседовании, тем лучше вам предложат стартовые условия.
 - **Не следить за новостями в мире выбранной технологии.** Я про новости с основных сайтов по технологиям, про блоги ведущих разработчиков, про новые книги и т.д. В ВУЗе вам просто физически не могут дать мейнстрим. Вам надо понять, что вас обучают не *программировать на сшарпе*, а *программировать на примере сшарпа*. Это значит, что вам никто не мешает использовать .NET Framework 3.5, LINQ и Entity Framework, хотя и не заставляют. Это ваш выбор – надо только не лениться его сделать.
 - **Не проходить различные программы сертификации.** Это момент конечно спорный, т.к. они денег стоят, но они окупаются. Кроме того, вам же всё равно надо учить .net, java или php, так почему бы не поучить по специальной литературе, по подготовке к сертификациям?

3. **Не знать английского.** Это вообще не обсуждается: хочешь работать в IT – надо знать английский. Английский сейчас де-факто стандарт в IT. Помимо того, что основная масса продуктов разрабатывается для англоязычных стран, есть ещё очень важная причина: документации к продуктам/API на неанглийском не существует. Её так мало, что ей реально можно пренебречь. А так что есть, отстаёт от английской версии на полгода минимум.

А ТЕПЕРЬ САМОЕ ВАЖНОЕ

Самое важное, что вы тут должны были увидеть – это то, что детские ошибки часто являются результатом тактических, а тактические – следствие стратегических. По большому счёту, можно всё решить одним махом – исправить стратегическую ошибку №2 ☺ Если это сделать, т.е. захотеть стать хорошим программистом, то тактические и детские ошибки... нет, не пропадут, но по крайней мере из системных перейдут в категорию случайных. А случайные ошибки легко устраняются опытом. Системные же ошибки препятствуют накоплению опыта.

ЗАКЛЮЧЕНИЕ

Я не питаю надежды, что раскрыл вам глаза на что-то новое. Всё что я рассказал – это здравый смысл и не более того, но теперь вы уже не сможете на собеседовании сказать «а я не знал, что так надо» ☺.

В общем и целом, я хотел бы вас поблагодарить за то, что присутствовали на этой сумбурной лекции, и пожелать в первую очередь целеустремлённости, а во вторую – удачи.

ПРИМЕЧАНИЯ

1. По ходу лекции больше всех записывал завкафедрой. И много кивал
2. На вопрос про "логику в обработчиках onClick" руку не поднял не один. Мучаюсь теперь вопросом – а может надо было спросить "кто знает, что такое обработчики событий onClick?"
3. Из авторов вспомнили только Троелсена и Сеппу, то есть вообще-то одного, т.к. Сеппа по ADO книгу написал, а не по .NET. Шилдт и Рихтер видимо из моды вышли...