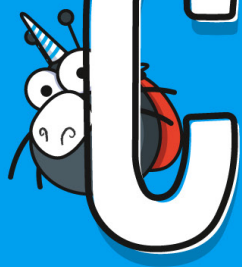
Андрей Карпов

ВРЕДНЫЕ СОВЕТЫ ДЛЯ C++ ПРОГРАММИСТОВ



Андрей Карпов

Вредные советы для C++ программистов

2023

Аннотация

В книге читателю предлагается обширный список «советов», которым на самом деле не стоит следовать. Каждый совет сопровождается подробным разбором и рассмотрением неочевидных моментов. Эти пояснения будут полезны новичкам, изучающим программирование. Впрочем, книга развлечёт и профессионалов рассмотрением некоторых нюансов программирования на C++.

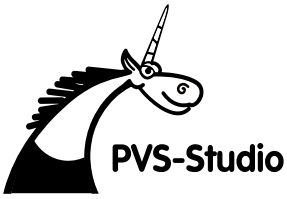
Введение

Изначально я задумал написать книгу о приёмах, помогающих в написании качественного кода. В ней хотелось рассмотреть такие темы, как оформление текста программ, на что стоит обращать внимание при обзоре кода, что из новых возможностей C++ стоит однозначно использовать, а что сомнительно. Но случайно меня увлекла другая тема, и я решил не сопротивляться потоку.

История началась с шуточной публикации на сайте «Хабр», где я сформулировал несколько вредных советов и предложил читателям продолжить этот список. Тема понравилась разработчикам, и они поделились массой своих идей о том, как можно создать ужасный код. Я взял часть советов, добавил свои и опубликовал в виде статьи. Там же я дал пояснение по некоторым советам, вся вредоносность которых может быть непонятна новичкам. Так и зародилась тема, которая в итоге превратилась в эту книгу.

Мне понравилось писать о том, как не надо делать, а не о том, как делать правильно. Хотя бы просто, для разнообразия. Я заскучал от написания правильных обучающих материалов. Захотелось чуть больше юмора и хулиганства.

Дело в том, что я многие годы занимаюсь темой программных ошибок и пишу о том, как их избегать. Это связано с моим участием в разработке и продвижении статического анализатора кода PVS-Studio.



Статический анализатор выдаёт предупреждения о фрагментах кода, которые с большой вероятностью содержат ошибки или не соответствуют используемым стандартам кодирования. PVS-Studio — один из таких инструментов. Его маскот — единорог — ещё не раз промелькнёт в этой книге. Сайт проекта: pvs-studio.ru

На примере открытых проектов я показываю, какие ошибки и потенциальные уязвимости могут быть выявлены с помощью этой методологии. Но, что ещё важнее, я стараюсь рассказывать, как написать код так, чтобы подобным ошибкам просто не нашлось места. В общем, вновь и вновь пишу, как лучше оформить код, какой комментарий написать, говорю о пользе обзоров кода и так далее.

Скучно. Пришло время творческих экспериментов. Я решил развить идею вредных советов и написать эту книгу. Впрочем, кроме идеи, от первоначального списка вредных советов мало что осталось. Я убрал или заменил всё слишком очевидное и неинтересное. Зато добавил много теоретического материала, в котором полезное найдут не только новички, но и опытные разработчики.

Нетрудно догадаться, что опыт, как стоит и не стоит писать код, я приобрёл, разрабатывая анализатор PVS-Studio и изучая обнаруженные с его помощью ошибки. Поэтому время от времени его упоминание будет всплывать в различных главах. Я настолько пропитался статическим анализом, что многие темы вызывают в моих воспоминаниях примеры реальных ошибок, которыми хочется поделиться для убедительности. А раз они убедительны, то почему бы о них и не написать.

Всё, теперь желаю приятного чтения.
Время приготовить чай/кофе, и вперёд.

Вредный совет N1. Только C++

Настоящие программисты пишут код на C++!

Нет ничего плохого в написании кода на C++. На этом языке написано множество прекрасных программ. Взять хотя бы список приложений с домашней страницы Бьёрна Страуструпа.

Вы можете познакомиться с этим списком, отсканировав приведённый ниже QR-код и перейдя по ссылке. Я буду использовать в книге QR-коды, когда нет смысла заимствовать какой-то материал из интернета или это невозможно. Ещё в тексте вы сможете встретить отсылки к терминам и приложениям, приведённым в конце книги. Заглядывайте туда, чтобы лучше разобраться, о чём идёт речь.

Here is a list of systems, *applications, and libraries that are completely or mostly written in C++. Naturally, this is not intended to be a complete list. In fact, I couldn't list a 1000th of all major C++ programs if I tried, and this list holds maybe 1000th of the ones I have heard of. It is a list of systems, applications, and libraries that a reader might have some familiarity with, that might give a novice an idea what is being done with C++, or that I simply thought «cool».*

Плохо, когда начинают использовать C++ только потому, что это «круто» или это единственный язык, с которым хорошо знакома команда.

Разнообразие языков программирования отражает многообразие задач, стоящих перед разработчиками приложений. Разные языки помогают элегантно решать различные классы задач.

Язык C++ претендует на звание универсального языка программирования. Однако универсальность не означает быстроту и простоту реализации конкретных приложений. Могут существовать языки, на которых проект будет реализован с меньшим вложением сил и времени.

Нет ничего страшного, если команда разработает небольшую вспомогательную утилиту на C++, хотя эффективнее для этого было бы использовать другой язык. Затраты на изучение нового языка могут превышать пользу от его применения.

Другое дело, когда перед командой стоит задача создания нового крупного проекта. В этот момент стоит остановиться и подумать: эффективно ли использовать для неё хорошо знакомый язык C++? Не лучше ли выбрать для этой задачи другой?

Если ответ — «да, использовать другой язык явно более эффективно», то, возможно, команде рационально потратить время на изучение этого языка. В перспективе это может на порядки сократить затраты на разработку и сопровождение. Или, возможно, стоит поручить этот проект другой команде, которая уже применяет более релевантный в данном случае язык.

Вредный совет N2. Табуляция в строковых литералах

Если в строковом литерале нужен символ табуляции, смело жмите клавишу Tab. Оставьте \t для яйцеголовых. Не парьтесь.

Речь идёт о строковых литералах, в которых требуется табуляцией отделять одни слова от других:

```
const char str[] = "AAA\tBBB\tCCC";
```

Казалось бы, по-другому и сделать нельзя. Тем не менее, нет пределов человеческой безалаберности. Кто-то, не задумываясь, нажимает клавишу Tab вместо того, чтобы использовать '\t'. Такое встречается в самых настоящих коммерческих приложениях.

Такой код компилируется и даже может работать. Однако явное использование символа табуляции плохо сразу по нескольким причинам:

1. На самом деле в литерале могут оказаться не табы, а пробелы. Это зависит от настроек редактора. Но выглядеть это будет так, как будто вставлена табуляция.
2. Человеку, который будет сопровождать код, не будет сразу очевидно, используется в качестве разделителей табуляция или пробелы.
3. Табуляция в процессе рефакторинга или использования утилит автоформатирования кода может превратиться в пробелы, что повлияет на результат работы программы.

Более того, однажды в реальном приложении я вообще видел приблизительно такой код:

```
 const char table[] = "\
bla-bla-bla bla-bla-bla bla-bla-bla-bla-bla \n\
      bla-bla-bla       bla-bla-bla\n\
          %s                %d\n\
          %s                %d\n\
          %s                %d\n\
";
  printf(table, "X", 1, "Foo", 2, "Time", 3);
```

Строка побита на части с помощью \. Явные символы табуляции использовались вперемешку с пробелами. К сожалению, не знаю, как здесь это показать. Поверьте, это смотрелось экстравагантно. Ещё и выравнивание от начала экрана. Бинго! Чего только не насмотришься, разрабатывая анализатор кода!

Явные табуляции, добавленные кнопкой Tab, очень легко «потерять». Если в процессе рефакторинга или автоформатирования они будут заменены на пробелы, то значения перестанут печататься в столбик:

```
bla-bla-bla bla-bla-bla bla-bla-bla-bla-bla
     bla-bla-bla       bla-bla-bla
         X                 1
         Foo                 2
         Time                  3
```

По-нормальному этот код следовало оформить как-то так:

```
const char table[] =
  "bla-bla-bla bla-bla-bla bla-bla-bla-bla-bla\n"
  "     bla-bla-bla       bla-bla-bla\n"
  "         %s\t            %d\n"
  "         %s\t            %d\n"
  "         %s\t            %d\n";


printf(table, "X", 1, "Foo", 2, "Time", 3);
```

Символы табуляции нужны, чтобы второй столбец выглядел выровненным несмотря на размер названий в первом столбце («X», «Foo», «Time»). Подразумевается, что эти названия всегда короткие, т. е. меньше 8 символов. Исправленный код распечатает:

```
bla-bla-bla bla-bla-bla bla-bla-bla-bla-bla

     bla-bla-bla       bla-bla-bla

         X                 1

         Foo               2

         Time              3
```

Кто-то может удивиться, что я рассказываю про такие простые и очевидные вещи. Более того, даже исправленный код попахивает и не тянет на хороший пример для книги. Однако, раз я сам видел явные символы табуляции в строках, про это стоит рассказать. Даже такое исправление лучше, чем если кто-то будет продолжать нажимать где попало Tab, не понимая, почему это плохо.