Полезная и расширенная программистская этика собрана по ссылке\*:

<https://tdoc.ru/c/programming/cpp/cpp-programming-rules.html>

\*является вспомогательной, а не основной литературой.

**Свод правил вежливости программиста, повышающих удобство чтения кода, составленного на языке *Visual C#***

**(для базового уровня освоения языка – консольный режим работы):**

**Общие:**

1. «В одной строке записывать только один оператор / одну операцию»:

Таблица 1 – Примеры корректного и некорректного составления кода с точки зрения размещения операторов в строке

|  |  |
| --- | --- |
| **Правильно** https://platinum1796.ir/wp-content/uploads/2022/04/1158_84611846a0ad287c2c71fc1ba5940659-1536x1223.png | **Неправильно** https://yt3.googleusercontent.com/ytc/AOPolaSeP1ubD-pvhOXDS8YbXfTMf-DPZbEqxDVXlRNQWQ=s900-c-k-c0x00ffffff-no-rj |
| static void Main(string[] args)  {  short inputVal, changeVal, sum = 0;  Console.Write("Введите ...");  short.TryParse(Console.ReadLine(),  out inputVal);  changeVal = inputVal;  sum += (short)(changeVal % 10);  changeVal /= 10;  sum += (short)(changeVal % 10);  changeVal /= 10;  sum += (short)(changeVal % 10);  changeVal /= 10;  sum += changeVal;  Console.WriteLine($"Сумма: {sum}");  Console.Write("Нажмите л. клавишу...");  Console.ReadKey(true);  } | static void Main(string[] args)  {  short inputVal; short sum = 0; short changeVal; Console.Write("Введите ..."); short.TryParse(Console.ReadLine(), out inputVal); changeVal = inputVal; sum += (short)(changeVal % 10); changeVal /= 10; sum += (short)(changeVal % 10); changeVal /= 10; sum += (short)(changeVal % 10); changeVal /= 10; sum += changeVal; Console.WriteLine($"Сумма: {sum}"); Console.Write("Нажмите л. клавишу..."); Console.ReadKey(true);  } |

Примеры из Таблицы 1 могут быть недостаточно показательными в связи с отсутствием явной демонстрации «вытягивания» многих операторов в одну строку. Следует рассмотреть аналогичный пример по иллюстрациям кода, записанного в интегрированной среде разработки (*IDE*) *Microsoft Visual Studio*.

На Рисунке 1 показано, как текст программы, собранный в одну строку, выходит за пределы рамки области кодирования (рабочей области) *IDE*.

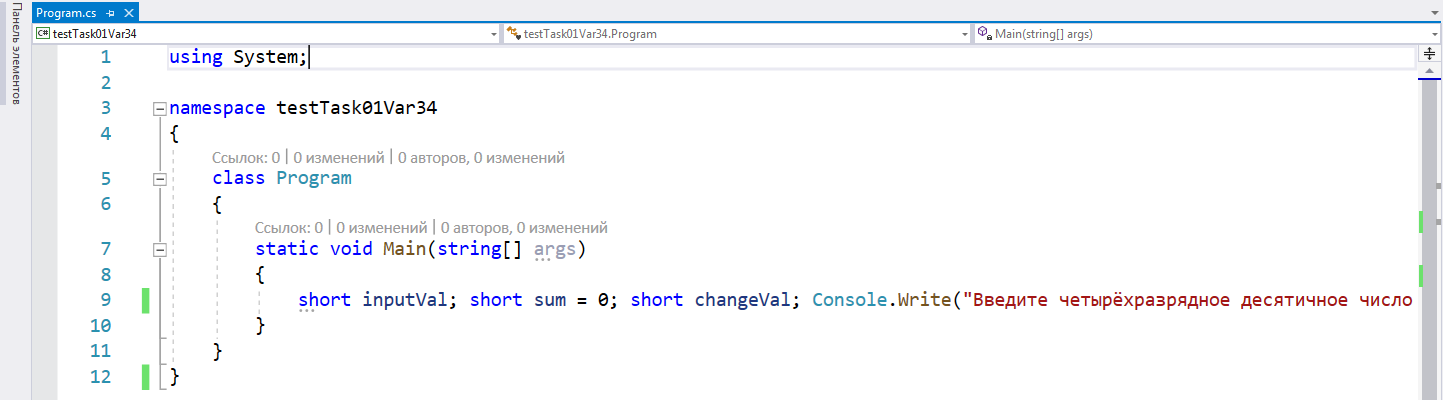


Рисунок 1 – Выход кода за рамки просматриваемой области кодирования (рабочей области) *IDE*

Можно воспользоваться инструментами среды *Microsoft Visual Studio* для просмотра и прочтения такого кода, но это потребует дополнительного ознакомления с инструкцией пользователя и доступными настройками *IDE* *Microsoft Visual Studio*.

**Внимание!** Применение последующих настроек может быть неудобно другим разработчикам программного обеспечения, использующим некоторый общий персональный компьютер (подразумевается коллективное использование персонального компьютера). В связи с этим по завершении работы одним пользователем он сам должен проследить за тем, чтобы вернуть исходные настройки *IDE*, для последующей нормальной работы второго пользователя. Если он первый пользователь не может гарантировать подобной степени самоорганизации или он не готов тратить время на перенастройки *IDE* при каждом использовании, ему следует руководствоваться правилами вежливости программиста, чтобы исключить необходимость применения последующей череды, приводимых в пример настроек.

На Рисунке 2 показан переход к перечню настроек *IDE*.

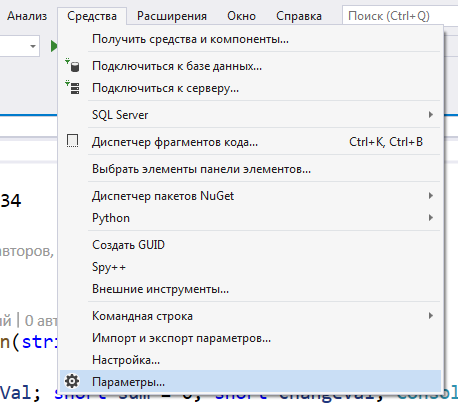


Рисунок 2 – Переход к диалогу «Параметры…» *IDE Microsoft Visual Studio*

На Рисунке 3 показано исходное состояния диалога «Параметры…».

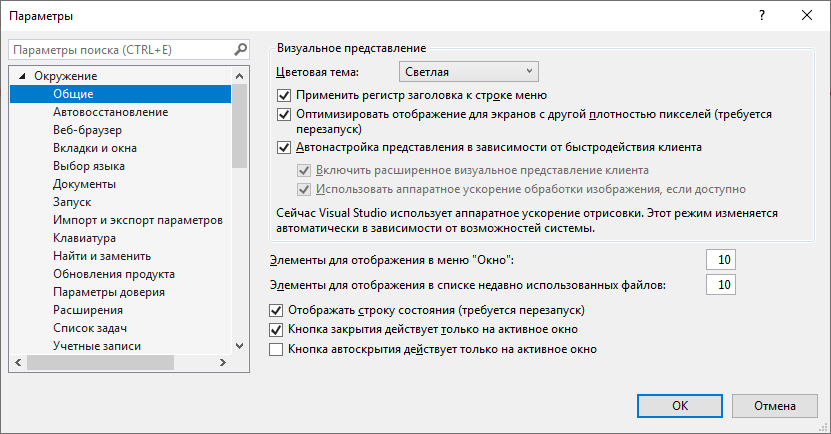


Рисунок 3 – Исходное состояние диалога «Параметры…» *IDE Microsoft Visual Studio*

На Рисунке 4 показан переход к настройкам текстового редактора (соответствующей вкладке «Текстовый редактор»).

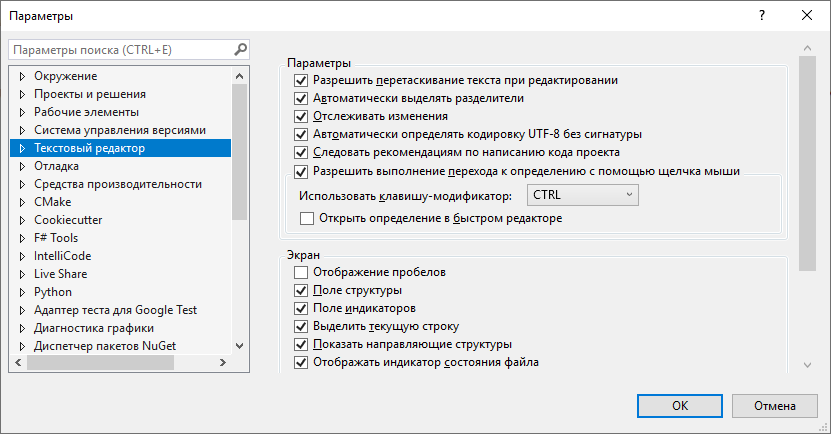


Рисунок 4 – Переход к настройкам текстового редактора в диалоге «Параметры…» *IDE Microsoft Visual Studio*

На Рисунке 5 показано, что внутри вкладки с настройками текстового редактора имеют место и настройки, применяемые к синтаксисам различных языков программирования. Здесь рассматриваются правила вежливости программиста *C#*, потому и последующий переход выполняется на вкладку *C#*, соответственно.

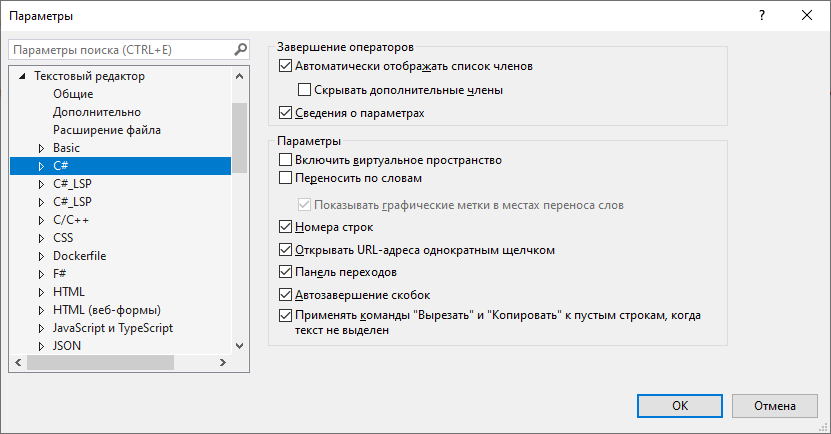


Рисунок 5 – Переход к настройкам текстового редактора в синтаксисе C# диалоге «Параметры…» *IDE Microsoft Visual Studio*

По умолчанию перенос по словам в перечне настроек *C#* не предусмотрен (Рисунок 6). Для повышения наглядности отображения кода, записанного в одну строку, в рассматриваемом примере опцию «Перенос по словам» необходимо активировать.

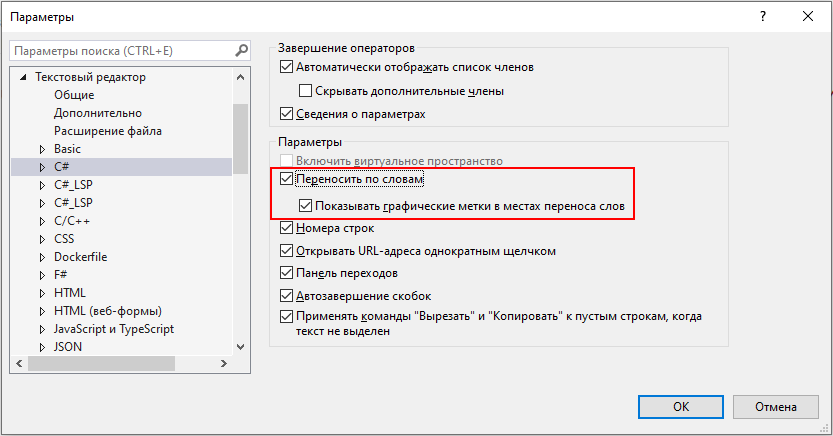


Рисунок 6 – Активация опций «Перенос по словам» и вложенной «Показывать графические метки в местах переноса слов»

После активации опции переноса по словам код в рабочей области начинает выглядеть так, как показано на Рисунке 7.

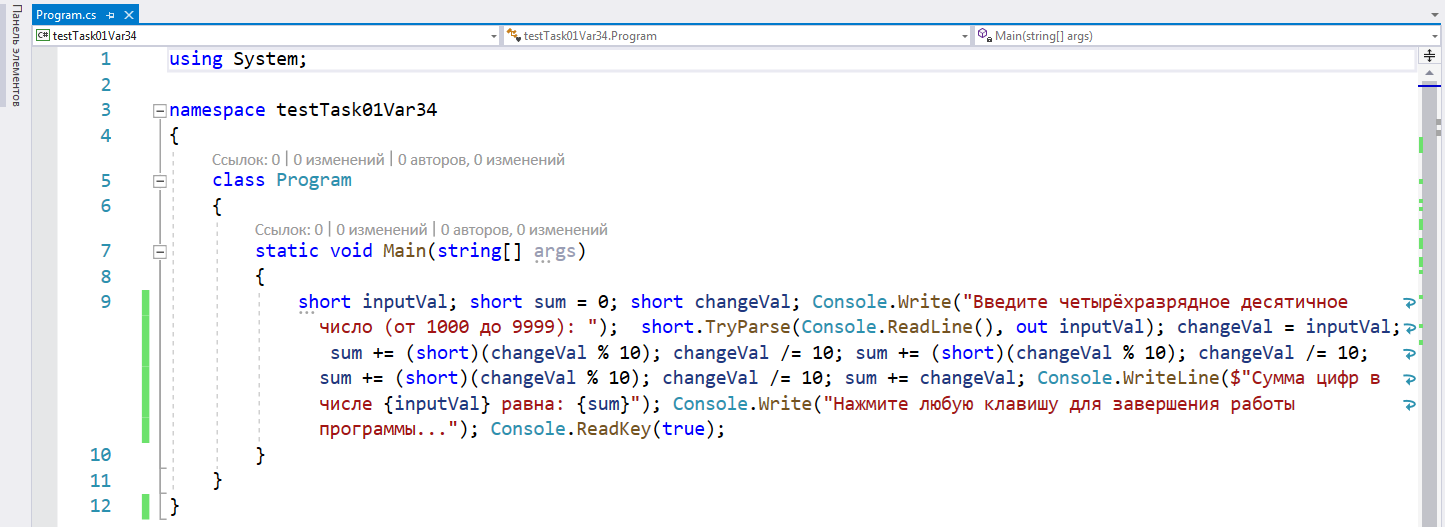


Рисунок 7 – Код программы, размещённой в одной строке, с переносом по словам по достижении границ области кодирования (рабочей области) *IDE Microsoft Visual Studio*

**Пояснение:**

а) Рассматриваемое правило вежливости программиста необходимо соблюдать, главным образом, для обеспечения удобства отладки программы. Если весь код будет записан только в одной строке, а ошибку предполагается «отлавливать», например, в 15-м операторе – пошагово придётся проходить все предыдущие 14-ть операторов, в то время, как если код записан построчно, то точку остановки (*breakpoint*) на полях можно выставить строго напротив 15-го оператора, а предыдущие 14-ть операторов могут быть пройдены не в пошаговом, а в автоматическом режиме. Отладка начнётся именно с 15-го оператора. Это удобно.

б) При соблюдении рассматриваемого правила вежливости программиста обеспечивается удобство чтения и восприятия информации, чтения и восприятия, непосредственно, кода. Строки с операторами, как правило, короткие, с операциями – более длинные. Операторы, как правило, с меньшей вероятностью приводят к ошибкам, нежели операции. По длине строки можно однозначно распознать, что там: оператор или операция; и начать последующую работу с проблемной областью. Это оперативно. Снижается риск систематической расфокусировки внимания программиста.