

Форматирование исходного кода



Правильное форматирование кода действительно важно, если хотите, чтобы код был легко-читаемым и был понятен не только вам. К примеру, студенты одной группы, которые читают одинаковую литературу по программированию, которым программирование преподает один и тот же человек, всё равно решают задачи разными способами. При этом у каждого свой особый стиль форматирования кода.

Если вы когда-либо пытались читать чужой код,

то замечали, что необходимо некоторое время, чтобы перестроиться и начать понимать, как работает этот код и что он реализует. Конечно, писать код можно по-разному, в том числе не совсем читабельно, и что самое интересное, программа будет замечательно работать. Вот посмотрите на этот код:

```
1 #include<iostream>
2 using namespace std;
3 int main(){ setlocale(LC_ALL, "rus");
4 int boxWithFruit=15;//количество ящиков на складе
5 int amountBoxForSale=0;// количество отгружаемых ящиков
6 cout<<"На складе сейчас "<<boxWithFruit<<" ящиков с фруктами.\n\n";
7 for(int i=1;;i++)//i - количество машин к погрузке
8 {cout<<"Сколько ящиков отгрузить в "<<i<<"-ю машину? ";
9 cin>>amountBoxForSale;
10 if(amountBoxForSale>boxWithFruit)
11 {cout<<"\nНа складе нет такого количества товара!";
12 cout<<"Осталось только "<<boxWithFruit<<" ящиков\n\n";
13 i--;//уменьшить счетчик на 1
14 }else{boxWithFruit-=amountBoxForSale;//перезаписываем значение
15 cout<<"Осталось "<<boxWithFruit<<" ящиков.\n";
16 }if(boxWithFruit==0)//если ящиков больше нет - выйти из цикла
17 {cout<<"Фрукты закончились! До свидания!\n";
18 break;}}return 0;}
```

Он полностью рабочий и никаких проблем при его выполнении не происходит. Но искренне жаль того программиста, которому «в наследство» достанется программа, написанная таким образом, и которому надо будет в эту программу что-то добавить или усовершенствовать её. Это еще хорошо, что в данной программе переменным даны осмысленные имена, а не **int a**, **int b**.

Хотя программа очень простая, она очень плохо читается.

Другое дело если она будет написана с использованием правил стандартов форматирования кода и соглашений о кодировании – с пробелами, отступами, переносами строк, комментариями:

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      setlocale(LC_ALL, "rus");
7
8      int boxWithFruit = 15; // количество ящиков на складе
9      int amountBoxForSale = 0; // количество отгружаемых ящиков
10
11     cout << "На складе сейчас" << boxWithFruit << " ящиков с фруктами.\n\n";
12     for (int i = 1;; i++) // i - количество машин к погрузке
13     {
14         cout << "Сколько ящиков отгрузить в " << i << "-ю машину? ";
15         cin >> amountBoxForSale;
16
17         if (amountBoxForSale > boxWithFruit)
18         {
19             cout << "\nНа складе нет такого количества товара!";
20             cout << "Осталось только " << boxWithFruit << " ящиков\n\n";
21             i--; // уменьшить счетчик на 1
22         }
23         else
24         {
25             boxWithFruit -= amountBoxForSale; // перезаписываем значение
26             cout << "Осталось " << boxWithFruit << " ящиков.\n";
27         }
28
29         if (boxWithFruit == 0) // если ящиков больше нет - выйти из цикла
30         {
31             cout << "Фрукты закончились! До свидания!\n";
32             break;
33         }
34     }
35     return 0;
36 }

```

Вы видите насколько проще для чтения и понимания стал этот код.

Практически всегда, над созданием одной программы работают несколько программистов. При этом сопровождать программу в дальнейшем может уже другая группа программистов. Становится очевидной необходимость создания единого стандарта кодирования. Если программисты будут придерживаться этих стандартов при написании кода – у всех он будет выглядеть одинаково, что в разы облегчит работу над коллективными проектами. Зачастую контора сама вырабатывает свой собственный стандарт кодирования (coding convention), но есть распространенные стандарты. К ним относятся, например:

[Google C++ Style Guide](#)

[Coding Standards IBM](#)

[Linux kernel coding style](#)

Полная версия [соглашения о кодировании на языке C++ на английском тут](#).

Если [посмотреть статистику](#) использования стандартов форматирования кода, становится очевидным, что большинству программистов есть еще над чем работать и к чему стремиться. Ведь больше половины из опрошенных, либо используют свой собственный стиль кодирования, либо вообще не знают, зачем стандарты форматирования и соглашения о кодировании созданы.

Поговорим о содержании стандартов и соглашений.

Основными задачами всех стандартов и соглашений о кодировании является способствование:

- написанию легко-читаемого кода, понятного для всех;
- написанию безопасного кода (ведь эти стандарты были созданы программистами-практиками, которые знают, какие ошибки может повлечь за собой неправильное оформление кода);
- единообразного кода (у всех структура кода выглядит схоже).

Просмотрев несколько стандартов и соглашений, можно увидеть, что в основном внимание уделяется следующим пунктам:

имена переменных, констант, функций, классов

Главное при объявлении переменной (функции, класса и т.д.) – дать осмысленное имя, как можно ближе к контексту использования. Допустим, в вашей программе определено больше 20-ти переменных, и всем дано имя в виде буквы алфавита. Согласитесь, даже вам, создателю этой программы, будет тяжело работать с этими переменными. Не говоря уже о том, что нужно так же помнить какие данные хранит каждая из них. Гораздо легче работать с переменными, имеющими обоснованные имена:

age – возраст;
number – номер;
amount – количество;
name – имя.

Желательно имена писать не английским транслитом, а английскими словами.

не vozrast – а age;
не kolichество – а amount.

Если вы не знаете перевода, воспользуйтесь одним из множества онлайн переводчиков. Заодно пополните ваш словарный запас.

Константам рекомендуется присваивать имена, либо состоящие из букв верхнего регистра (HOURS_IN_DAY, SIZE) либо каждое новое слово с большой буквы, как Google C++ Style Guide (kHoursInDay). Говоря о константах, их советуют использовать везде, где только можно. Не объявляйте переменные хранящие количество дней в неделе и месяцы в году – объявляйте константные переменные в таких случаях. Относительно функций – если функция не изменяет аргумент, передаваемый по ссылке или по указателю, то аргумент должен быть константой.

Тогда как для имен переменных используются существительные, для имен функций необходимо использовать глаголы или глагол + существительное. Так правильней потому, что функция выполняет определенное действие:

printData(); – печать данных
enterName(); – ввод имени
showStr(); – показать строку

В имени класса первая буква должна быть заглавной:

```
class Employee  
class Point
```

Если имя состоит из нескольких слов, написать его можно разными способами.

– каждое новое слово с большой буквы (верблюжий регистр): `boxWithApple`, `amountBoxesForSale`

– использовать нижний прочерк между словами: `box_with_apple`, `amount_boxes_for_sale`

Не бойтесь давать сложные имена. В среде **Microsoft Visual Studio** обращаясь к объявленной переменной (начиная вводить ее имя), вам будет предложено выбрать имя из выпадающего списка. Давать слишком длинные имена тоже не стоит. Вполне реально отразить суть переменной в нескольких словах.

Сразу хочу рассказать о «венгерской нотации» – соглашении об именовании переменных и других идентификаторов в коде программ. Суть «венгерской нотации» в том, что имя переменной (функции, массива и т.д.), начинается с префикса, состоящего из одной или нескольких букв. Приведу несколько примеров объявления имен идентификаторов с префиксами:

`int iAmount` или `nAmount`,

`bool bChoice`,

`int aNumbers` (а говорит о том, что `aNumbers` – это массив),

`string sName` (строка),

`int* pArr` (от слова `pointer` – указатель)

Когда мы встретим такое имя в коде, то его префикс будет говорить нам о том, что это за идентификатор и какие данные он хранит.

фигурные скобки

В некоторых соглашениях и стандартах рекомендовано использовать фигурные скобки в блоках `if`, `else`, `while`, `do`, `for` даже если они содержат всего одну строку либо не содержат ничего. Например:

```
1 for(int example = 0; example < 10; example ++)  
2 {  
3     cout << example << end;  
4 }
```

Каждую фигурную скобку желательно располагать в отдельной строке. Так очень легко проследить, где блок начинается и где заканчивается.

пробелы в строке и отступы между строками

Рекомендуется не использовать пробелы в конце строки перед оператором точка с запятой.

При использовании оператора присвоения значения пробелы необходимы с обеих сторон от этого оператора:

```
1 variable = 0;
```

Это же касается и операторов используемых в арифметических выражениях:

```
1 variable = a + b - c;  
2 variable=a+b-c; // слитно выглядит так
```

Применяя унарные операторы, пробелы не нужны:

```
1 variable = -4;  
3 variable++;
```

Многие используют табуляцию в строках, чтобы визуально выделить некоторые блоки кода. Так например выглядит фрагмент кода с [вложенными циклами](#) с использованием табуляции:

```
1 int main()  
2 {  
3     for (int i = 0; i < 12; i++)  
4     {  
5         for (int j = 0; j < 12; j++)  
6         {  
7             cout << '@';  
8         }  
9         cout << endl;  
10    }  
11    return 0;  
12 }
```

и так без нее:

```
1 int main()  
2 {  
3 for (int i = 0; i < 12; i++)  
4 {  
5 for (int j = 0; j < 12; j++)  
6 {  
7 cout << '@';  
8 }  
9 cout << endl;  
10 }  
11 return 0;  
12 }
```

По поводу отступов между строками кода, основной принцип сводится к тому, чтобы минимизировать их количество. Логика в том, что чем больше код, который умещается на одном экране, тем легче проследить и понять работу программы. Поэтому не используйте больше одной-двух пустых строк между блоками кода или функциями.

комментирование кода

К сожалению, многие не любят комментировать код, хотя комментарии играют важную роль в поддержании читаемости кода на высоком уровне. Как написано в Google C++ Style Guide «Комментарии важны, но лучше, когда код сам говорит за себя. Давать осмысленные имена переменным гораздо лучше, чем давать непонятные названия и затем раскрывать их суть в комментариях»

Оставлять комментарии в коде можно либо используя двойной слэш // (комментирование одной строки), либо /* комментарий */ . Вторым способом можно оформить многострочный комментарий. Но предпочтение авторы некоторых стандартов предлагают отдавать все же комментированию с использованием двойного слэша // .

Каждая функция должна иметь комментарии непосредственно перед ней, которые описывают то, что делает эта функция и как её использовать. Например, перед функцией можно написать:

```
// Открывает файл, или // печатает данные
```

Так же желательно, чтобы каждое определение класса имело сопроводительный комментарий, описывающий, что это такое и как его следует использовать.

В стандартах форматирования кода можно почерпнуть еще много чего полезного для формирования собственного хорошего стиля программирования. Всего в данной статье не перечислишь. Поэтому постарайтесь найти время и почитать некоторые первоисточники, ссылки на которые были приведены выше в статье.

Хочу обрадовать тех, кому все же придется столкнуться с кодом, написанным подобно первому листингу в этой статье. В среде разработки **Microsoft Visual Studio** есть “спасательная комбинация клавиш” Ctrl+K затем Ctrl+F, нажав которую осуществится форматирование выделенного исходного кода. То есть если выделить неотформатированный код и нажать эту комбинацию клавиш – автоматически добавятся и необходимые пробелы, и отступы, и скобки перенесутся в отдельные строки. В общем станет код выглядеть, как для людей.

Применяя на практике форматирование кода, описанное в стандартах и соглашениях о кодировании, вы добьетесь и достойной легкой читаемости кода, и его безопасности. Этим вы облегчите работу и себе и коллегам по проекту. Возможно, даже кто-то скажет вам спасибо.