1 Форматирование кода

Любая синтаксически правильно написанная программа на многих языках программирования(в том числе и на C++) будет работать, даже если не уделять внимания на форматирование.

Пример плохо форматированной программы :

```
#include<iostream>

int main(){ std::cout<<"Hello world";return 0;}
```

Не смотря на то, что эта программа небольшая, и написана синтаксически правильно, её сложно читать. Если форматировать таким же стилем и более большую программу, то её сложнее будет читать, отлаживать и поддерживать.

В тоже время эту простую программу нетрудно отформатировать и привести к читабельному виду :

```
#include<iostream>

int main() {
    std::cout << "Hello world";
    return 0;
    }
```

Когда один проект пишет много программистов с различными стилями, наступает т.н. "code hell когда весь код программы невозможно отнести к одному единому стилю; тогда между программистами возникают споры по поводу правильного форматирования кода. Тогда, на общем совещании программистами решается какой единый стиль форматирования использовать в проекте. Создается документ с описанием стиля команды, и, если в

команду приходит новый человек, то он обязательно должен ознакомиться с этим стилем и писать так, как пишут все. Иначе - его могут просто уволить. Также, т.к. все современные проекты содержаться в системах контроля версиями(cvs; svn,git,hg, etc.), в некоторых компаниях стоит фильтр на плохо форматированный код - невозможно будет добавить плохо отформатированный код в систему.

У многих компаний этот стиль стандартизирован и его можно посмотреть в открытом доступе :

- Стиль написания библиотек Qt - Qt - QtCodingStyle - Open wiki - Qt by Nokia
- Стиль написания в Google - Google C++ Style Guide

Вот еще пример стилей форматирования C++ :

- C++ Programming Style Guidelines
- C++ Coding Standard
- C and C++ Style Guides

Чаще всего понимание стиля форматирования вырабатывается с годами, поэтому нет ничего плохо в том, чтобы за основу брать какой-нибудь хорошо известный стиль форматирования.

1.1 Операции

Всегда разделяйте операции(иногда, не только арифметические) пробелами(но не переусердствуйте).

Сравните :

```
int a=5+7/9*(6+8)-6;
```

А читабельнее этот код выглядил бы так :

```
int a = 5 + 7 / 9 * (6 + 8) - 6;
```

Вот еще пример хорошого форматирования ;

```
a += c + d;
a = (a + b) / (c * d);
```

1.2 Глобальные переменные

Любая глобальная переменная должна начинаться с большой буквы :

```
const int N;
int Array[N][N];
```

Так вы не спутаете глобальные переменные с локальными.

1.3 Константы

Именуйте ваши константы большими буквами, разделяя, при необходимости, составные слова подчеркиваниями : Пример хороших констант :

```
const int N = 100;
const double PI = 3,14;
const double MAX_ITERATIONS = 100;
```

Иногда, константы именуют по первым большим буквам слов :

```
const double Pi = 3,14;
const double MaxIterations = 100;
```

Такое написание обладает следующим минусом : константы написанные таким образом легко спутать с глобалльными переменными.

1.4 Переменные итерации

Общепринятые имена для переменных итерации - i, j, k. Очень в редких случаях не хватает этих трёх имен. Если у вас в программе не хватает этих трех имен, хорошенько подумайте над дизайном программы - возможно вы что-то делаете не так.

1.5 Цикл for

Переменные итерации, если это цикл for, должны "жить"в нём и распространяться на его. Объяснить это очень просто - раз это переменные итерации, значит они должны итерировать только в цикле, нет необходимости их использовать вне цикла. Очень редко бывают случаи, когда нужно использовать переменные итерации вне области видимости цикла.

Сравните, плохой код :

```
int i = 0;
for(; i < N; i++)
    std::cout << i;
```

Или, также плохой код :

```
int i = 0;
for(i = 0; i < N; i++)
    std::cout << i;
```

И хороший код :

```
for(int i = 0; i < N; i++)
    std::cout << i;
```

Все современные компиляторы(g++ 4.4 и старше, MSVC 8 и старше) распространяют действие переменной только на внутреннюю область види-

мости цикла. Таким образом, создаем и используем переменные итерации внутри цикла.

1.6 Числовые константы-литералы

Если в вашей программе используются какие-то числовые константы, например размер массива, то создайте переменную-константу для этого числа.

Плохой код :

```
int ar[5] = {0, 1, 2, 3, 4};

for (int i = 0; i < 5; i++) {
    std::cout << ar[i];
}
```

Хороший код :

```
const int N = 5;
int ar[N] = {0, 1, 2, 3, 4};

for (int i = 0; i < N; i++) {
    std::cout << ar[i];
    }
```

1.7 Пустые строки

Не пишите код программы подряд на каждой строке по операции, разделяйте по мере возможности код на группы областей пустыми строками :

```
int i = 0, j = 0, k = 0;
```

```
std::cout << i << j << k;
```

В этом небольшом куске кода мы разделили области создания переменных и их вывода на консоль. Разделив большой алгоритм на логические части, можно будет проще контролировать каждую часть - код становится приятнее и читабельнее. Но не злоупотребляйте расстановкой пустых строк.

1.8 Форматирование области видимости

Области видимости или тела функций/классов/циклов/операторoа, разделяемые фигурными скобками должны быть оттабулированы, так будет видно где заканчивается действие той или иной области видимости.

Пример хорошо форматированного кода :

```
#include<iostream>

struct Point {
    int x;
    int y;
    };

int main() {
    for (int i = 0; i < 5; i++) {
        std::cout << "Hello world" << std::endl;
        }

    return 0;
    }
```

1.9 Расстановка фигурных скобок

Расстановка фигурных скобок во многих командах программистов бывает различной. Расстановка скобок зависит от того, как оформляются функции, сколько пробелов ставится для отступа тела функций и операторов и расположения фигурных скобок.

Подробнее о расстановке фигурных скобок и отступах можно прочитать здесь :

- Indent style - Wikipedia

Microsoft Visual Studio / Allman Style

Вот пример расстановки фигурных скобок по умолчанию в Microsoft Visual Studio :

```
int main(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 10; i++)
    {
        std::cout << i;
    }

    return 0;
}
```

До Visual Studio этот стиль был использован Эриком Оллманом в оформлении стандарта ANSI C, после чего стал называться стилем Оллмана. Такой стиль очень похож на Horstmann style и GNU style.

GNU style

GNU-стиль популярен из-за своего дополнительного переноса между типом возвращаемого параметра и названием функции :

```
static char *
concat (char *s1, char *s2)
```

```
{
  while (x == y)
    {
      something ();
      somethingelse ();
    }
  finalthing ();
}
```

По-видимому из-за влияния ФЯ Lisp Ричард Столлман популяризировал этот стиль.

Whitesmiths style

Можно было бы использовать и такой распространенный стиль :

```
int main(int argc, _TCHAR* argv[])
    {
    for (int i = 0; i < 10; i++)
        {
        std::cout << i;
        }

    return 0;
    }
```

KR стиль

Такой стиль был использован в Керниганом и Ритчи в их известной книге "Язык программирования C":

```
int main(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 10; i++) {
        std::cout << i;
```

```
        }

        return 0;
    }
```

Этот стиль расстановки скобок(особенно для вложенных конструкций) также называют египетскими скобками (egyptian brackets).

Banner C-style

Мы будем использовать в основном этот стиль форматрования, на котором закрывающя фигурная скобка также имеет отступ как и тело цикла/функции/if-а и др :

```
int main(int argc, _TCHAR* argv[]) {
    for (int i = 0; i < 10; i++) {
        std::cout << i;
        }

    return 0;
    }
```

Заметьте, что //после закрывающей круглой скобки стоит пробел// и только потом открывающая фигурная скобка. Этот стиль был выбран из-за своей удобности и простоты в офрмлении кода.

1.10 Табуляция vs Пробелы

Для создания области видимо обычно нажимают <Tab>. Выбор того, как выглядит таб - это один символ табуляции или набор из 4 или 8 пробелов - сугубо индивидуальный. Этот выбор зависит от используемого редактора, т.к. не все редакторы поддерживают замену использования вместо табуляции 4х/8ми пробелов. Однако большинство современных редакторов(Eclipse, Code:Blocks, gedit) поддерживают замену символа табуляции на пробелы.

Оглавление