C++ Code Style

Style

سازگاری مهمترین جنبه style است. دومین جنبه مهم، پیروی از سبکی است که برنامه نویس C++ معمولی به خواندن آن عادت دارد.

C++ اجازه می دهد تا نام های identifier با طول دلخواه را انتخاب کنید، بنابراین دلیلی برای کوتاه کردن نام آن ها وجود ندارد. از نام های توصیفی استفاده کنید و در style ثابت باشید.

- CamelCase
- snake_case

این دو استایل نمونه های رایج هستند. snake_case این مزیت را دارد که در صورت تمایل می تواند با spell checkers نیز کار کند.

ایجاد یک دستورالعمل Style

هر دستورالعمل Style که ایجاد می کنید، مطمئن شوید که یک فایل با فرمت clang. که سبک مورد انتظار شما را مشخص میکند، پیاده سازی کنید. در حالی که این نمی تواند به نامگذاری کمک کند، به ویژه برای یک پروژه منبع باز حفظ یک Style ثابت مهم است.

هر IDE و بسیاری از ویرایشگرها از فرمت clang پشتیبانی می کنند یا به راحتی با یک افزونه قابل نصب هستند.

- VSCode: Microsoft C/C++ extension for VS Code

  CLion: https://www.jetbrains.com/help/clion/clangformat-as-alternative-formatter.html

  VisualStudio https://marketplace.visualstudio.com/items?itemName=LLVMExtensions.ClangFormat#review-details
- Resharper C++: https://www.jetbrains.com/help/resharper/2017.2/Using_Clang_Format.html
- Vim
  - https://github.com/rhysd/vim-clang-format
  - https://github.com/chiel92/vim-autoformat
- XCode: https://github.com/travisjeffery/ClangFormat-Xcode

قراردادهای رایج نامگذاری در C ++

- تایپ با حروف بزرگ شروع می شوند: MyClass
- توابع و متغیرها با حروف کوچک شروع می شوند: myMethod
- ثابت ها همه حروف بزرگ هستند: const double PI= 3.14159265358979323

کتابخانه استاندارد C++ (و دیگر کتابخانه‌های معروف C++ مانند Boost) از این دستورالعمل‌ها استفاده می‌کنند:

- نام‌های ماکرو از حروف بزرگ با underscores استفاده می‌کنند: INT_MAX
- نام پارامترهای الگو از camel case استفاده می کند: InputIterator
- همه نام‌های دیگر از snake case استفاده می‌کنند: unordered_map

متمایز کردن پارامترهای تابع

مهمترین چیز ثبات در codebase شما است. این یک امکان برای کمک به ثبات است.

پارامترهای تابع را با پیشوند t_ نامگذاری کنید. t_ را می توان به عنوان "the" در نظر گرفت، اما معنایش قراردادی است. نکته این است که پارامترهای تابع را از سایر متغیرها در scope متمایز کنیم و در عین حال یک استراتژی نامگذاری ثابت به ما ارائه می دهد.

```
struct Size
{
  int width;
  int height;

  Size(int t_width, int t_height) : width(t_width), height(t_height) {}
};

// This version might make sense for thread safety or something,
// but more to the point, sometimes we need to hide data, sometimes we don't.
class PrivateSize
{
  public:
    int width() const { return m_width; }
    int height() const { return m_height; }
```

```
    PrivateSize(int t_width, int t_height) : m_width(t_width), m_height(t_height) {}

  private:
    int m_width;
    int m_height;
};
```

نام چیزی را با _ شروع نکنید

اگر این کار را انجام دهید، در معرض ریسک برخورد با نام های رزرو شده برای استفاده از کامپایلر و اجرای کتابخانه استاندارد هستید:
http://stackoverflow.com/questions/228783/what-are-the-rules-about-using-an-underscore-in-a-c-identifier

Well-Formed Example

```
class MyClass
{
public:
  MyClass(int t_data)
    : m_data(t_data)
  {
  }

  int getData() const
  {
    return m_data;
  }

private:
  int m_data;
};
```