ETロボコン

コーディング規約（C++）

|  |  |  |  |
| --- | --- | --- | --- |
| 版 | 作成者 | 作成日 | 備考 |
| 第１版 | 寺元 聖満 | 2011/10/01 | 新規作成 |

目次

[目次 2](#_Toc305338028)

[**1** **はじめに** 3](#_Toc305338029)

[**2** **コーディング上の規則** 3](#_Toc305338030)

[**2.1** **ファイルサイズ** 3](#_Toc305338031)

[**2.2** **禁止事項** 3](#_Toc305338032)

[**2.3** **その他** 3](#_Toc305338033)

[**3** **命名規則** 4](#_Toc305338034)

[**3.1** **ファイル名、クラス名** 4](#_Toc305338035)

[**3.2** **メソッド名** 4](#_Toc305338036)

[**3.3** **対称性** 4](#_Toc305338037)

[**3.4** **ループカウンタ** 4](#_Toc305338038)

[**3.5** **定数** 5](#_Toc305338039)

[**3.6** **変数** 5](#_Toc305338040)

[**4** **コメント** 5](#_Toc305338041)

[**4.1** **ファイルヘッダ** 5](#_Toc305338042)

[**4.2** **クラスヘッダ** 5](#_Toc305338043)

[**4.3** **メソッドヘッダ** 6](#_Toc305338044)

[**4.4** **変数定義など** 6](#_Toc305338045)

[**4.5** **その他コメント** 6](#_Toc305338046)

[**5** **実行文の記述** 6](#_Toc305338047)

[**5.1** **ネスト** 6](#_Toc305338048)

[**5.2** **インデンテーション** 6](#_Toc305338049)

[**5.2.1** **関数構造** 6](#_Toc305338050)

[**5.2.2** **構造体** 7](#_Toc305338051)

[**5.2.3** **enum** 7](#_Toc305338052)

[**5.2.4** **if文** 7](#_Toc305338053)

[**5.2.5** **switch文** 8](#_Toc305338054)

[**5.2.6** **for文** 8](#_Toc305338055)

[**5.2.7** **while文** 8](#_Toc305338056)

[**5.2.8** **do while文** 8](#_Toc305338057)

[**6** **データ型** 9](#_Toc305338058)

[**7** **パフォーマンス** 9](#_Toc305338059)

[**8** **参考資料** 9](#_Toc305338060)

1. はじめに

本規約はC++言語を用いてコーディングを行なうプログラムに適用する。

潜在不具合の防止と保守の容易性ため、C++言語ソースファイルの記述様式にについて定める。

1. コーディング上の規則

コーディングにおける規則を記述する。

* 1. ファイルサイズ
* 総行数の指定はないが、極力長くならないようする
* １行は最大80桁とし、それを超える場合は行を分割する
  1. 禁止事項
* goto文は使用しない
* ASCⅡ/Shift-JISを使用する
* 日本語は全角文字を使用し、半角カナはできるだけ使用しない
* 大きなサイズの自動変数は宣言しない
* main 関数を除き、原則としてクラス外部の変数や関数は使わない
* ローカル変数の再利用はしない
  1. その他
* 他人がみても分かりやすいことを優先する
* コメントは極力つける
* 変数名は必ず初期化する
* 必要最低限の変数宣言をする
* 妥当なアクセスメソッドを設けること
* constにできるものはconstにする
* スーパークラスの変数名と同じ変数名は使用しない
* 引数タイプによるメソッドのオーバーロードは極力避ける
* static メンバ変数を避ける
  + セミグローバルであり、文脈依存なコードを招き、副作用を覆いかくしてしまうため
* ローカル変数の宣言は必要な箇所の直前で行う

1. 命名規則

ファイル名、変数名の命名規則を記述する。

* 1. ファイル名、クラス名
* 原則として省略は行わない
* 「先頭文字」「区切り」を大文字とし、それ以外は小文字とする

例：class DriverManager

* 1. メソッド名
* 「区切り」を大文字とし、それ以外は小文字とする

例：DriverManager::createDriverTable()

* 属性の設定メソッドは「set」、取得メソッドは「get」をメソッド名の先頭に付与する
* コンバートメソッドは「to」「form」など変換先、変換元が分かるようにする
  1. 対称性

クラス名、メソッド名をつける際は、以下のように対称性にする

|  |  |
| --- | --- |
| add | remove |
| insert | delete |
| get | release |
| start | stop |
| begin | end |
| send | receive |
| first | last |
| put | get |
| up | down |
| open | close |
| old | new |
| increment | decrement |

* 1. ループカウンタ

スコープが狭いループなど原則としてカウンタ、イテレータは「ii」「jj」「kk」をこの順に使用する。

* 1. 定数

大文字を「\_」アンダーバーでつなげる

例：NO\_REQ

* 1. 変数

変数名を見て機能が分かるものにする。その他以下のルールに従う。

|  |  |  |
| --- | --- | --- |
| 種別 | 説明 | 例 |
| ローカル変数 | ― | S16 xxx |
| ポインタ変数 | 先頭に「p」を付与する | S16\* pXxx |
| グローバル変数 | 先頭に「g」を付与する | S16 gXxx |
| メンバ変数 | 先頭に「m」を付与する | S16 mXxx |

1. コメント

コメントについて記述する。

* ドキュメント自動生成ツールDoxygenの規約に準ずる
* JaveDocタイプのコメントルールとする
* インデントは4文字
* 改行やスペースの入れ方に統一感をもたせる
  1. ファイルヘッダ

/\*\*

\* @file *ファイル名*

\* @brief *ファイルの概要*

\* @author *作成者*

\* @date *作成日*

\*/

* 1. クラスヘッダ

/\*\*

\* @brief *クラスの概要*

\*/

* 1. メソッドヘッダ

/\*\*

\* @brief *機能の概要*

*\**

\* @param[in | out] *パラメータ1の説明*

\* @param[in | out] *パラメータ2の説明*

\* @return *戻り値の説明*

*\** @attention *仕様上の注意点*

\*/

* 1. 変数定義など

///< *概要記述*

* 1. その他コメント

// *内容*

* Doxygenに出力されない

1. 実行文の記述
   1. ネスト

最大ネストは５とする

* 1. インデンテーション

関数構造、構造体、enum、switch文、while文、if文、do while文の記述方法をそれぞれ記す。

* + 1. 関数構造

関数宣言と「{」「」」は同一カラムとする。関数内ステートメントは1レベル字下げる。

void main()

{

関数内ステートメント

}

* + 1. 構造体

typedef structと「」」は同一カラムとする。構造体要素は1レベル字下げる。

Typedef struct {

構造体要素

} NAME;

* + 1. enum

enumと「{」「」」は同一カラムとする。enum要素は1レベル字下げる。

enum

{

enum要素

}

* + 1. if文

ifと「{」「」」は同一カラムとする。文群は1レベル字下げる。

if (条件式)

{

文群1

}

else if (条件式)

{

文群2

}

else

{

文群3

}

* + 1. switch文

switchとcase、「{」「」」は同一カラムとする。文群は1レベル字下げる。

switch (条件式)

{

case 定数式1:

文群1

case 定数式2:

文群2

default:

文群3

}

* + 1. for文

forと「{」「」」は同一カラムとする。文群は1レベル字下げる。

for (式1; 式2; 式3)

{

文群

}

* + 1. while文

whileと「{」「」」は同一カラムとする。文群は1レベル字下げる。

while (条件式)

{

文群

}

* + 1. do while文

doとwhile、「{」「」」は同一カラムとする。文群は1レベル字下げる。

do {

文群

} while (条件式)

1. データ型

nxtOSEKで使用可能なデータ型について記述する。

ecrobot\_types.h, mytypes.h

|  |  |  |
| --- | --- | --- |
| 定義内容 | データ型 | サイズ（byte） |
| unsigned char | U8 | 1 |
| signed char | S8 | 1 |
| unsigned short | U16 | 2 |
| signed short | S16 | 2 |
| unsigned long | U32 | 4 |
| signed long | S32 | 4 |
| signed int | SINT | 4 |
| unsigned int | UINT | 4 |
| float | F32 | 4 |
| double | F64 | 8 |

1. パフォーマンス

パフォーマンス向上のため、推奨事項を挙げる。

* newは時間がかかるため、ループ内で繰り返し呼ばれることは避ける
* 関数の引数がオブジェクトの場合、値渡しとせず、constな参照渡しとする

1. 参考資料

C++ コーディング標準（翔ソフトウェア）

<http://www.shos.info/develop/xp/cplusstd.html>

Google C++スタイルガイド

<http://www.textdrop.net/google-styleguide-ja/cppguide.xml>

Doxygen

<http://www.doxygen.jp/docblocks.html>

以上