Форматиране на сорс-код

Опции на текстовия редактор – Опцията за табулация трябва да бъде настроена на 4 разстояния (space).

Класове и типове

Класове, структури, енумератори и др.

Именувайте класове, използвайки смесица от големи и малки букви, започвайки с главна буква и без подчертавки – например SolidCode, GenericPlayer.

Имената на интерфейсите трябва да започват с I и главна буква – например IresourceMgrClient.

Имената на структурите трябва да започват с главно S (пр. SplayerCoords), а енумераторите с E (enum EGameStates).

Забележка: Избягвайте typedef към анонимни структури, енумератори и др, като:

```
typedef struct
{
	std::string strMapName;
	int nPosX;
	int nPosY;
	int nPosZ;
} SPlayerCoords;
```

По-трудно е да се разбере какви са и къде са дефинирани в кода. Следният вариант е за предпочитане:

```
typedef struct SPlayerCoords
{
	std::string strMapName;
	int nPosX;
	int nPosY;
	int nPosZ;
};
```

Променливи

Използват се два вида представки – за тип и обхват.

Представки за тип
Обикновено променливите имат представка, която подсказва типа им. Представката за тип се записват с малки букви, точно преди променливата, която се започва с главна буква. Всяка значима дума в името на променливата също започва с главна буква – пр. NsomeVar. Когато не се използва представка, името на променливата започва с малка буква – var.

Променливи от тип Клас може да нямат префикс. В този случай името на променливата започва с малка буква – напр. Object obj.

Обхват

Представките за обхват се разделят с подчертавка – m_nVar. Следните представки са задължителни:

```
публични член-променливи            - n/a    ( var, използвана вътре в
класа: this->var )
не-публични член-променливи         - m_    ( m_var )
статични член-променливи            - s_    ( s_var )
глобални променливи                 - g_    ( g_var )
```

Шаблони за писане на код

Всички променливи, трябва да бъдат инициализирани, когато се дефинират.

Пример: int i = 0; Вместо int i; i = 0;

Не използвайте магически числа в кода. Вместо това използвайте константи или стойности от изброим тип – enum.

10-те правила на НАСА за разработката на критично-безопасен код

1: Ограничете всеки код до прости конструкции за контрол на потока. Не използвайте GOTO изрази, setjmp или longjmp конструкции, пряка или непряка рекурсия.

2: Всички цикли трябва да имат точно определена горна граница. Трябва да бъде тривиално и възможно за проверяващ инструмент да докаже статистически, че предварително зададена горна граница на броя итерации на цикъла не може да бъде надвишена. Ако границата на цикъла не може да бъде доказана статистически, правилото се счита за нарушено.

3: Не използвайте динамично заделяне на памет след инициализация.

4: Функция не трябва да бъде по-дълга от това, което може да бъде отпечатано на един лист (в стандартен формат А4 с един ред за израз и един за декларация). Обикновено това означава не повече от 60 реда кода за функция.

5: Гъстотата на използване на assert в кода трябва да има среден минимум от два броя assert за функция. Твърденията (assertions) не трябва да имат странични ефекти за програмата и трябва да бъдат дефинирани като Булеви тестове.

6: Обекти от данни трябва да бъдат декларирани на най-малкото ниво на обхват.

7: Всяка извикваща функция трябва да проверява не-void връщащи стойности на функции. Валидността на параметрите трябва да се проверява вътре във всяка функция.

8: Предпроцесора трябва да бъде ограничен до включването на заглавни файлове и прости дефиниции на макроси. Копиране на токени, списъци с аргументи на променливи (елипси) и рекурсивни извиквания на макроси не са разрешени.

9: Използването на указатели трябва да бъде ограничено. По-конкретно, повече от едно ниво на дереферциране, не е позволено. Операции, свързани с дереференциране на указатели, не трябва да бъдат скрити в дефиниции на макроси или вътре в typedef декларации.

10: Всеки код трябва да бъде компилиран от първия ден на разработка, с включени всички предупреждения на компилатора, при най-педантичната му настройка. Всеки код трябва да се компилира с тези настройки без каквито и да е предупреждения. Всеки код трябва да бъде проверяван ежедневно поне веднъж, за предпочитане повече от веднъж, с модерен статичен анализатор на кода, като трябва да премине анализа с нула предупреждения.