**C++ Coding Standard for QTeam**

**Names**

**Class Names**

• Use upper case letters as word separators, lower case for the rest of a word

• First character in a name is upper case

• No underbars ('\_')

Example

class MyClass

**Method Names**

• Use the same rule as for class names.

• Suffixes and prefixes are useful

• If method name is generated by Qt automatically, keep it. Don’t change it.

Example

Class MyClass

{

Public:

int CopyItem();

void DeleteItem();

bool IsDigit();

int GetVar();

int SetVar();

}

**Class attribute names**

• Attribute names should be prepended with the character 'a'.

• After the 'a' use the same rules as for class names.

Example

Class MyClass

{

Public:

int CopyItem();

void DeleteItem();

bool IsDigit();

int GetVar();

int SetVar();

private:

int aItemCnt;

string aItemName;

}

**C++ File Extensions**

Use .h extension for header files and .cpp for source files

**Variable Names on the stack**

• use all lower case letters

• use '\_' as the word separator.

Example

int Myclass::SetVar(){

int index\_of\_item = 0;

……

}

**Pointer Variables**

• pointers should be prepended by a 'p' in most cases

• place the \* close to variable name not pointer type

Example

MyClass \*pNewClass = new MyClass;

**Global varibles**

• Global variables should be prepended with a 'g'.

Example

int gCount;

**Global constants**

• Global constants should be all caps with '\_' separators.

Example

const int POP\_SIZE = 100;

**Formatting**

**Braces**

• Traditional Unix policy of placing the initial brace on the same line as the keyword and the trailing brace inline on its own line with the keyword:

• Always uses braces form

Example

while (true) {

…

}

**Parens**

• Do not put parens next to keywords. Put a space between.

• Do put parens next to function names.

• Do not use parens in return statements when it's not necessary.

Example

if (true) {

…

}

**Commenting**

• Use comments on starting and ending a Block

• Use comments before a function

Example

//delete an item

int MyClass::DeleteItem()

{

// Block1 (meaningful comment about Block1)

... some code

{

// Block2 (meaningful comment about Block2)

... some code

} // End Block2

} // End Block1

**Indentation/Tabs/Space**

• Indent using 4 spaces for each level.

• Tabs should be fixed at 4 spaces. Don't set tabs to a different spacing, uses spaces instead.

• Indent as much as needed, but no more. There are no arbitrary rules as to the maximum indenting level. If the indenting level is more than 4 or 5 levels you may think about factoring out code.

**Use header file guards**

Example

#ifndef filename\_h

#define filename\_h

**Errror checking**

• Use Exceptions Instead of Return Values to Indicate Error

• Check every system call for an error return.

• Handle errors using assertions from the (assert.h) library. The assert function is defined as follows: void assert(int expression);

• Include the system error text for every system error message.

Example

ifstream fIn(“input", ios::in);

if(!fIn){

cerr<<"unable to open input file"<< endl;

exit(-1);

}