985 实验室 C/C++程序变量和函数命名规范及注意事项

Version 1.0 by Y.Y.

1. 函数命名规则（驼峰命名法）：

普通函数：函数名由若干个单词组成，第一个单词全部小写，第二个单词开始首字母大写：

bool getMeanValue(…);

int csvToShp(…);

double** computeUrbanConversionMatrix(…);

注意：

a. 如果是 *inline* 类型的函数，在函数名前加下划线_：

inline int _getCuberInterpolationValue(…);

b. 如果是 static 类型的函数，函数名第一个单词首字母大写：

static int OpenFiles(…);

2. 变量命名规则（匈牙利命名法）：

原则 1：禁止使用简单英文单词命名变量，如 min,max,left,right，会造成和某些库保留变量名冲突

原则 2：变量作用域(w/m/_/无) + 指针/数组(p/pp/ppp/无) + 变量类型(c/u/n/f/d/s/v 等) + 变量名称

原则 3：迭代变量允许但不推荐 i,j,k,m,n。但尽量使用有意义的迭代变量名称，如：_nFilesIdx

表 1 变量作用域命名规则

变量作用域	前缀	例子	意义
全局变量	w	wnValue	全局 int 型变量
静态变量	S (大写)	SnValue	静态 int 型变量
类变量	m	mnValue	类 int 型变量
普通变量	不需要	nValue	int 型变量
临时变量	_	_nValue	临时 int 型变量

表 2 指针/数组命名规则

指针/数组	前缀	例子	意义
一维	p	pdValues	普通一维 double 型数组
二维	pp	ppdValues	普通二维 double 型数组
三维	ppp	pppdValues	普通三维 double 型数组
非指针	无	dValue	普通 double 型对象

表 3 变量类型命名规则

变量类型	前缀	例子	意义
char	c	cValue	
unsigned char, byte	u	uValue	
short, unsigned short	n	nValue	
int, unsigned int			
long, long long	l	lValue	
float	f	fVafue	
double	d	dVadue	
bool	b	bVabue	
char*, string	s / str / sz	strValue	
list, vector	v	vValues	链表/容器
file	in/out/fi	inFile / outFile / fiOutput	输入输出文件
map, hash	map	mapKeyValues	映射表/哈希表
HANDLE	h	hDesktop	桌面句柄
GDAL 相关指针	po	poDataset	GDAL 相关指针
控件	ctrl	ctrlTextView	TextView 控件
源对象	src	srcString	源字符串
目的对象	dst	dstString	目的字符串

例：

```
double** mppdDistanceMatrix;    //类内二维双精度型指针变量
QList<int>* wpvnValuesList;     //全局 int 型链表的指针
QList<int*> mvpnValueList;      //类内链表对象，存储 int 型指针
QProgressBar* mpctrlProgBar;    //类内 ProgressBar 指针对象
GDALDataset* mpoDataset;        //GDAL 栅格数据集指针对象
```

等

3. 其他非常重要的注意事项

a. 为防止对同一个文件被编译器多次编译，每个头文件（*.h）实现结构必须为：

***#ifndef** 随机唯一标识字符串，千万不要和其他的头文件重复*

***#define** 随机唯一标识字符串，千万不要和其他的头文件重复*

(类的实现)

#endif

e.g.

```
#ifndef MY_CLASS_HFILE_20150925 //该类唯一标识字符串
#define MY_CLASS_HFILE_20150925 //该类唯一标识字符串

#include <iostream>
using namespace std;

class MyClass
{
    public:
        ...
    protected:
        ...
    private:
        ...

};

#endif
```

b. **为防止头文件互包含造成编译错误，每个头文件（*.h）中 include 的其他头文件数量越少越好。或者只进行声明，在对应的实现文件（*.cpp）中将所需的头文件包含进来；**

e.g.

```
#ifndef MY_CLASS_HFILE_20150925 //该类唯一标识字符串
#define MY_CLASS_HFILE_20150925 //该类唯一标识字符串

class QString;  //在h文件只声明，不包含头文件
class QLabel;   //在h文件只声明，不包含头文件

class MyClass
{
    public:
        ...
    protected:
        ...
    private:
        QString msFilename;
        QLabel* mpctrlLabel;
        int mnParameter;

};

#endif
```

c. **为防止内存泄漏造成程序不稳定，在类和函数中声明的指针一定要先置 NULL 或 nullptr；每次使用前先判断是否为 NULL，若需要更新先 delete 再 new；并且在类的析构函数中或函数尾部对该指针 delete 并置 NULL 或 nullptr 处理。**

e.g.

```
#ifndef MY_CLASS_HFILE_20150925 //该类唯一标识字符串
#define MY_CLASS_HFILE_20150925 //该类唯一标识字符串

class MyClass
{
    public:
        MyClass()
        {
            mpVals = NULL;  //构造函数置NULL
        }

        ~MyClass()
        {
            if(mpVals != NULL)
                delete []mpVals;
            mpVals = NULL;  //析构函数delete并置NULL
        }

    protected:
        void myTestFunction(int nsize = 100)
        {
            //更新前判断是否为NULL，如果不为NULL，则delete
            if (mpVals != NULL)
            {
                delete []mpVals;
                mpVals = NULL;
            }

            //更新
            mpVals = new double[nsize];

            //置零
            memset(mpVals, 0, nsize*sizeof(double));
        }

    private:
        double* mpVals;

};

#endif
```

d. 如何动态创建和删除二维指针

e.g.

```
#ifndef TEST_FUNCTION_HFILE_20150925     //该类唯一标识字符串
#define TEST_FUNCTION_HFILE_20150925 //该类唯一标识字符串

//创建和删除一个nRows * nCols的二维数组
void makeAndDelete2DimMatrix(int nRows = 20, int nCols = 30)
{
    double** _ppdMatrix = NULL;
    int i=0;

    //建立nRows长度的指针数组
    _ppdMatrix = new double*[nRows];
    //指针数组中的每个指针开辟nCols大小的空间
    for (i=0; i<nRows; i++)
    {
        _ppdMatrix[i] = new double[nCols];

        //置零
        memset(_ppdMatrix[i], 0, nCols*sizeof(double));
    }

    //处理过程略

    //删除二维指针，释放空间
    //第一步删除每行动态数组
        for (i=0; i<nRows; i++)
    {
        delete []_ppdMatrix[i];
        _ppdMatrix[i] = NULL;
    }
    //第二部删除指针数组
    delete []_ppdMatrix;
    //置NULL
    _ppdMatrix = NULL;

}

#endif
```