规范与要求

- 不要把结构体定义、函数实现都放在一个.c中实现，按照需要建立.c和.h 文件，源代码文件名需以模块名开头；

【规则】文件名 = 组件/模块名 + 组件/模块设计结构层次名
【规则】遵循统一的布局顺序书写.h/.c文件

规范与要求

.h文件布局

```
文件头注释
#ifndef  文件名_H（全大写）
#define 文件名_H

/*-------------------- 头文件 --------------------*/
#include <标准库头文件>
#include <外部接口头文件>
#include  “内部头文件”

/*-------------------- 宏定义 --------------------*/

/*---------------------- 枚举定义 ----------------------*/

/*----------------------- 结构体 -----------------------*/

/*---------------------- 函数声明 ----------------------*/


#endif
```

规范与要求

.c文件布局

文件头注释

```
#include <标准库头文件>
#include <外部接口头文件>
#include  “内部头文件”

/*---------------------- 外部变量声明 ----------------------*/

/*---------------------- 外部函数声明 ----------------------*/


/*---------------------- 全局变量声明 ----------------------*/

/*---------------------- 静态函数声明 ----------------------*/


/*---------------------- 函数实现 ----------------------*/
```

规范与要求

- 函数里尽量不要出现%i0、&g0...等寄存器名，要解析出具体的变量名

【规则】暂不能确定的变量名 = 前缀([sg/g+“_”]+[指针(p)]+类型(i/l/b/e/d/s/c) + 偏移地址

【规则】自定义数据类型用“typedef” 关键字定义，数据类型名必须以模块标识开头，并注释上各字段偏移地址及整个结构体大小。

```
int      g_i0x3FF30;
double   g_d0x3FF30;
int     *g_pi0x400A8;
double  *g_pd0x400A8;
```

```
typedef struct
{
    int cycle_nr;                   //0x0
    int chuck;                      //0x4
    double residuals_without_fit;   //0x8
    double fit_residuals;           //0x10
    double spec_sigma_fit;          //0x18
    char s0x20[0x100];              //0x20
} EMAZ_fit_residuals;  //size = 0x120
```

规范与要求

类型前缀

前缀	数据类型	备注
b	bool	4字节
c	char	1字节
d	double	8字节
e	enum	4字节
f	float	4字节
i	int	4字节
l	long	8字节
n	short	2字节
s	结构/标准C字符串	必须保证字符串是以“NULL”字符结尾的
u	表示无符号的	加在类型前，如ul表示unsigned long
p	指针	加在类型前，表示地址，如pl表示long *
g s	全局变量 静态变量	加在最前面，表示作用域。例如： g_pul：表示全局的无符号长整型指针变量 s_i：表示本地静态整型变量 sg_i：表示静态全局整型变量 如果是除了静态变量之外的本地变量，则不需要添加作用域前缀

规范与要求

【规则】‘{’ 和 ‘}’ 应各独占一行并且位于同一列，同时与引用它们的语句左对齐。 ‘{’ 和 ‘}’ 之内的代码块使用缩进规则对齐。

正例

```
for (...)
{                       // 独占一行并与引用语句左对齐
    [代码块]            // 使用缩进规则
}                       // 独占一行并与引用语句左对齐

if (...)
{                       // 独占一行并与引用语句左对齐
    [代码块]            // 使用缩进规则
}                       // 独占一行并与引用语句左对齐

SMEE_INT32 foo(void)
{                       // 独占一行并与引用语句左对齐
    [代码块]            // 使用缩进规则
}                       // 独占一行并与引用语句左对齐
```

反例

```
for (...) {             // 未独占一行
    [代码块]
}

if (...) {              // 未独占一行
    [代码块]
}                       // 未与引用语句左对齐

SMEE_INT32 foo(void)
{                       // 未与引用语句左对齐
    [代码块]
}                       // 未与引用语句左对齐

SMEE_INT32 foo(void)
{
    [代码块]            // 未使用缩进规则
}
```

规范与要求

【规则】结构型的数组、多维的数组如果在定义时要赋初值，按照数组的矩阵结构分行书写，必须添加‘{’和‘}’。

正例

```
SMEE_INT32 array[4][3] =
{                                   // 独占一行并与引用语句左对齐
    { 1, 1, 1 },                        // 使用缩进规则
    { 2, 4, 8 },
    { 3, 9, 27 },
    { 4, 16, 64 }
};                                  // 与‘{’对齐
```

反例

```
SMEE_INT32 array[4][3] = { 1, 1, 1, 2, 4, 8, 3, 9, 27, 4, 16, 64 };  // 不够清晰
```

规范与要求

加空格的情况

- 比较操作符，赋值操作符“=”、“+=”，算术操作符“+”、“%”，逻辑操作符“&&”、“&”，位域操作符“<<”、“^”等双目操作符的前后加空格
- 在语句行中间“，”、“；”之后（而不是之前）加空格
- if、for、switch、while等关键字与左括号“(”之间加空格
- **只加一个空格**

正例

```
if ((0 == iErrorCode) && g_sInput.bAutoSaveMC)
{
    iErrorCode = save_mc();
    if ((0 != iErrorCode) && (XXXX_ABORT_ERR != iErrorCode))
    {
        iErrorLink[0] = iErrorCode;
        EH4A_show_exception(iErrorCode, iErrorLink,
            __FILE__, __LINE__, "save mc failed.");

        g_sWarningText = "save mc failed!");
    }
}
```

规范与要求

不加空格的情况

- 多重括号间
- “!”、“~”、“++”、“--”、“&”（地址运算符）等单目操作符前后
- “[ ]、“->”、“.”前后
- ‘,’、‘;’向前紧跟，‘(’向后紧跟，‘)’向前紧跟，紧跟处不留空格。
- 函数名之后不加空格。
- 有时为了代码紧凑，在双目操作符前后不加空格。

正例

```
if ((0 == iErrorCode) && g_sInput.bAutoSaveMC)
{
    iErrorCode = save_mc();
    if ((0 != iErrorCode) && (XXXX_ABORT_ERR != iErrorCode))
    {
        iErrorLink[0] = iErrorCode;
        EH4A_show_exception(iErrorCode, iErrorLink,
            __FILE__, __LINE__, "save mc failed.");

        g_sWarningText = "save mc failed!");
    }
}
```

规范与要求

加空行的情况

函数间、函数体局部变量声明完毕后。

〖建议〗不同逻辑程序块之间要使用空行分隔。

〖建议〗return语句（含出口trace）之前加空行。

〖建议〗只加一个空行

不加空行的情况

- 函数头注释与函数定义之间不加空行。
- 逻揖上密切相关的语句之间不加空行。

【规则】函数声明时，类型与名称不允许分行书写。

【规则】较长的语句要分成多行书写，长表达式要在低优先级操作符处拆分成新行，操作符放在新行之首（以便突出操作符）。拆分出的新行要进行适当的缩进，使排版整齐。

〖**建议**〗程序中一行的内容（含代码及注释）以小于80字符为宜，不要写得过长。

规范与要求

- 不得使用goto语句；
- 尽量保证函数有且只用一个return语句；
- 在复杂的代码流转处标示出地址（标签），方便查找；
- 对代码进行必要的注释，并记录解析过程中遇到的重要问题，有效注释量不少于10%；
- 交付的C源代码，需编译通过，依赖的外部头文件（除甲方提供的外）需自行构造。

ps. 不属于任务安排内的函数均认为是外部函数！

Thank you !