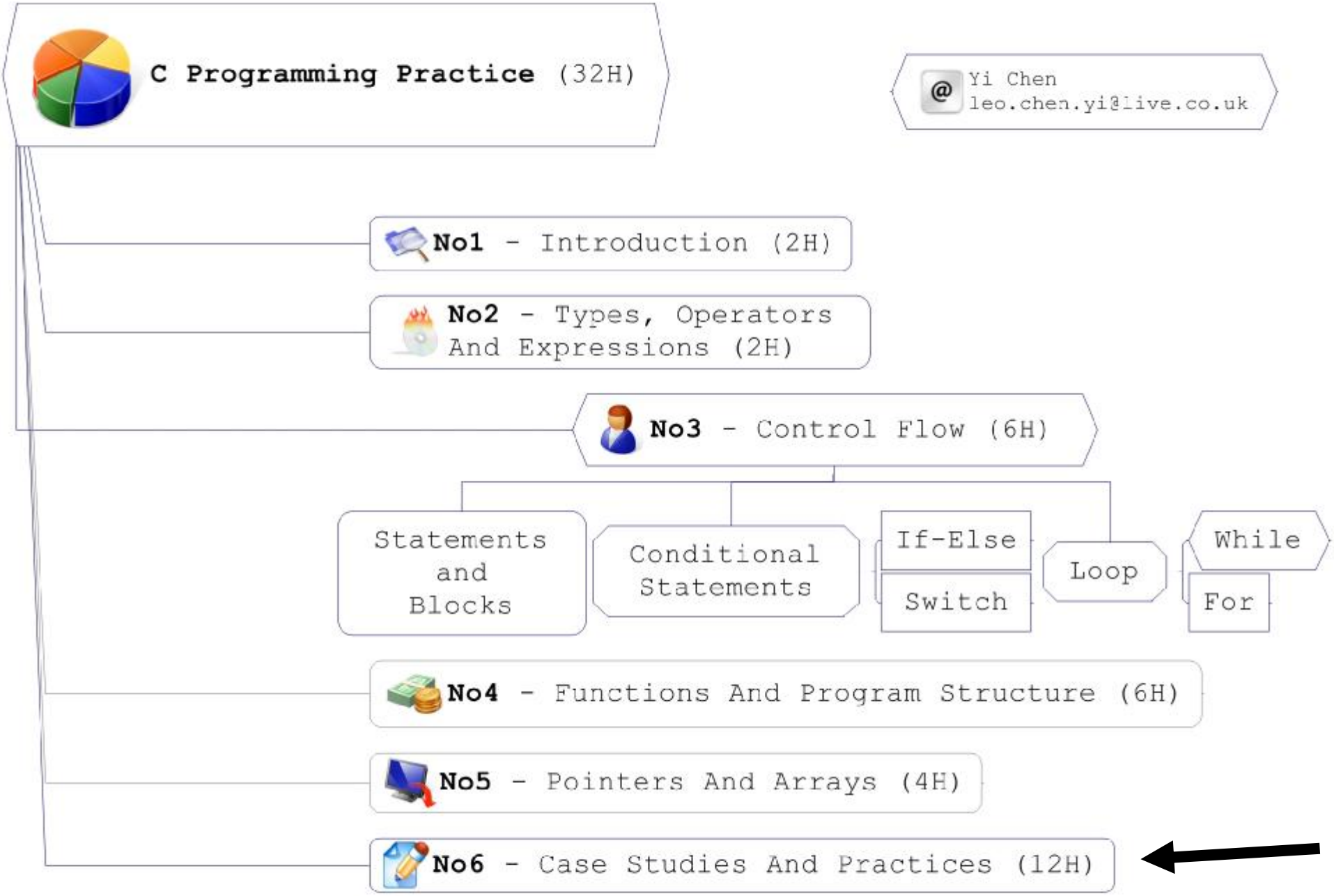
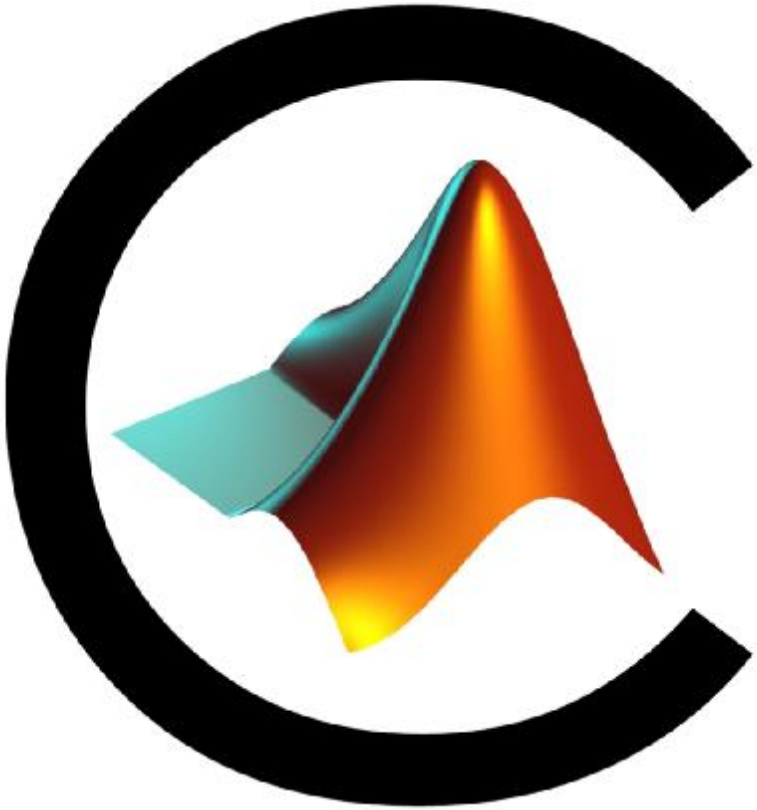

C Programming Practice (32H)
Yi Chen
leo.chen.yi@live.co.uk
No1 - Introduction (2H)
No2 - Types, Operators And Expressions (2H)
No3 - Control Flow (6H)
Statements and Blocks
Conditional Statements
If-Else
Switch
Loop
While
For
No4 - Functions And Program Structure (6H)
No5 - Pointers And Arrays (4H)
No6 - Case Studies And Practices (12H)

C Programming Practice
No[6]

Chen , Yi
leo.chen.yi@live.co.uk
30-Jul-2010



Case Studies and Practices
-- Coding Tips

BETA 0.0.0.1

Contents

Contents

- **Files and Variables Naming**
- Function
- Dynamic Memory Allocation
- How to define a structure
- Initialisation
- An array with it's length known
- An array with it's length un-know
- The C Preprocessor
- Remember the value set last time
- How to deal with "Memory Error "?

Files and Variables
Naming

```
        int i, j, k;
            /* not good */

or          /* good */
        int  ind  = 0;
        int  jdx, kdx = 0;
```

1. variable name

Names are there to clarify - so choose good names.

A good name can often take **as long to determine** as writing the code which it describes.

This point isn't obvious, but is important. When choosing a name, consider an uninitiated **reader** will **understand** exactly what is meant by it.

Files and Variables Naming

1. variable name

Avoid naming **type** or **structure** definitions with words like "**struct**" or "**data**".

Name them after what they do - it'll make it far easier for the reader to understand the purpose of the type or structure.

e.g. use "**XYZ_options_t**" instead of "**XYZ_data_options_t**" for a structure which takes input options.

e.g. use "**XYZ_results_t**" instead of "**XYZ_data_struct_t**" for a struct which returns results. You can probably name it even more meaningfully than these generic examples!

Files and Variables Naming

1. variable name

Use **full words** wherever possible. **Don't abbreviate** just for the sake of it. In particular don't remove all the vowels from a name just to save on typing - remember other people will have to read the code.

The length of an identifier should be proportional to its scope.

i is fine for the control variable of a short for loop.

A global function needs a **fully** descriptive name.

Files and Variables Naming

2. file name

SGA__str2cell.m

SGA_mutation.m

SGALAB_demo_SO_std.m

INPUT_AGE_database_math_sgn.txt

OUTPUT_best_coding_space.txt

SECF__AGE_generation.m

Files and Variables Naming

3. function name

(1) Entry function

```
 *       1)  mexFunction() - SMAT_matlabcall Matlab
Main Entrance
```

(2) Static internal functions:

```
*       1)  SMAT__pkt_handling_0_default()
*       2)  SMAT__pkt_handling_1_BEGINDLGPKT()
```

(3) Static functions:

```
 *       1)  SMAT_matlabcall_mathlink_open()
 *       2)  SMAT_matlabcall_mathlink_close()
 *       3)  SMAT_matlabcall_mathlink_close_must()
```

Contents

Function

```
extern double FTK_ask_max_element
(
double data[] , /*I*/
int data_length /*I*/
)
{
    double max =  data[ 0 ] ;
/* give the first data to max*/
    for (int idx= 1; idx < data_length; idx++)
    {
        if (max < data[ idx ]  )
        {
            max = data[ idx ]  ;
        }
    }
    return max;
}
```

Function

I – Input

O – Output

I/O – Input and Output

OF – Output and Need to be freed later later

I/OF – Input, Output and Need to freed later

```
/* HEAD SMAT_MATLABCALL CCC SMATLINK */
/*==========================================================

  Simple MATlab and MAThematica LINK laboratory Toolbox
for Matlab 7.x

==========================================================
File description:

    SMAT_matlabcall.c , This is a MEX-file for MATLAB.

    MATLAB call MATHEMATICA via MathLink Connection

Platform:
…
Additional information:

(1) Entry function
 *  1)  mexFunction() - SMAT_matlabcall Matlab Main
Entrance
```

```
(4) Problems Reporting List, TODOLIST
 *       1) Check input mode
 *       2) SMAT_matlabcall('TraditionalForm[D[f[x, y], {x,
2}, {y, 3}]]');

(5) Reference

============================================================
Date           Name         Description of Change
04-Aug-2008    Yi Chen      Initial
01-Sep-2008    Yi Chen      Add #include <windows.h>
$HISTORY$
============================================================*/
```

Contents

Dynamic Memory Allocation

Length Known

```
/*the length of array will always be the exact one
during all calculating period*/

#define MAX_STRING_LEN 255

char    file_name [MAX_STRING_LEN+1] = '\0';

double  data [256] = {0.0};
```

```
#define N_ELEMENTS(array) (sizeof(array)/sizeof(array[0]))
```

Dynamic Memory Allocation

Length Unknown

Case 1

```
int  n_bodies  = 0 ;

/* = int data , will be generated by another function
during calculating ,dynamically*/

tag_t   *bodies = NULL ;
/*step here, n_bodies have int value now, but it may has
different int value during calculating*/
n_bodies  = 100;

bodies = (tag_t *)malloc( n_bodies * sizeof(tag_t));
...
free(bodies);

bodies = NULL ;
```

Dynamic Memory Allocation

Length Unknown

Case 2

```
double (*bounding_boxes)[6] = …;
int    n_sheets  = …

/* = int data , will be generated by another function
during calculating ,dynamically*/

bounding_boxes = (double (*)[6])malloc( n_sheets *
sizeof(double[6]));

…

free(bounding_boxes);
```

Contents

How to define a structure

Define structure data type: *plot_xydata_s*

Generally,

in a ***header file .h***

```
typedef struct xydata_s
{
        double          *xdata ;
        double          *ydata ;

        char             xdata_unit[MAX_STRING_LEN+1] ;
        char             ydata_unit[MAX_STRING_LEN+1] ;

        int              data_length ;

        double           x_axis_length ;
        double           y_axis_length ;
        double           x_margin ;

        joint_driver_t driver_type;
} plot_xydata_t, *plot_xydata_p_t;
```

Define structure *name* and a *pointer*

How to define a structure

in a **c source *file .c***

```
static plot_xydata_t  xydata_preview     = { 0 } ;
static plot_xydata_t *xydata_preview_ptr = &xydata_preview ;
```

```
/*
A "static" variable does not necessarily need
to be initialised;

An "Auto" type of variable should be
initialised when defining.

And , always initialisation when defining a
variable is good habit.
*/
```

How to define a structure

Briefly:

```
/* Not Correct */
typedef structure
{
int    creation_status;
double origin[3];
}  XXX_data_t, * XXX_data_p_t;

/* Correct */
typedef structure XXX_data_s
{
int    creation_status;
double origin[3];
}  XXX_data_t, *XXX_data_p_t;
```

Contents

Initialisation — char array

```
[1] char *string = ”String Line”;

[2] char string [] = “String Line”;

[3] char buffer[80];

    sprintf( buffer,
             "An approximation of Pi is %f \n", M_PI );

/* the size of buffer[] need to be allocated properly
That is, it need greater than the string length we
need later */
```

Initialisation

char array

[4] snprintf

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     char str[10] = {'\0'};

     snprintf(str, sizeof(str), "012345678901234567890");

     printf("str = %s \n", str);

     return 0;
}
```

Initialisation

char array

```
[5] char string1[10];
      strcpy (string1,"String") ;

/* = strcpy(string1[0],"string"); */

[6] Static char name[2][8] = {"Leo","Alan" };
/* results: name[0] = "Leo" , name[1] = "Alan "*/

[7] char name[3] ;
      Name[0] = ' L ';
      Name[1] = ' e ';
      Name[2] = ' o '; /*result : name[0]="leo"*/
```

Initialisation

int/double array

[1] double data [2]= { 1.2 , 2.3 };

/*= double data [ ]= { 1.2 , 2.3 }; */

[2] double data [2] ;

```
data[0] = 1.2;
data[1] = 2.3;
```

[3] int a[5] ={0 ,0 , 0, 0, 0};

/*int a[5] = { 0 } ;*/

[4] if the length of array is un-known, see ***Dynamic Memory Allocation*** in ***Chapter: Pointer***

Initialisation

structure

```
struct student
{
 char name[12];
 char sex;
 int score;
} student1[2], *student1;

[1] Student1[2]= {
            { “leo”,  ”M”, ”88” },
            { “fredo”,”M”, ”90” }  };

[2] student1 = {0};
 /*only for every elements can be set to 0.
if don’t, you have to make a function of
init_fun( ) to initialise this structure */
```

Contents

An array with it's length known

Length Known

```
/*the length of array will always be the exact one
during all calculating period*/
```

e.g.1

```
char      file_name[MAX_STRING_LEN + 1] = '\0';
```

e.g.2

```
double    data[256] = {0.0} ;
```

Contents

An array with it's length un-know

e.g. 1

```
int  n_bodies  = 0 ;
/* will be generated by another function during
calculating ,dynamically*/

tag_t  *bodies = NULL ;
/*step here, n_bodies have int value now, but it may
has different int value during calculating*/

bodies = (tag_t *) malloc( n_bodies * sizeof(tag_t));
...

free(bodies);
bodies = NULL ;
```

An array with it's length un-know

e.g. 2

```
double (*bounding_boxes)[6] = …;

int     n_sheets  = …

/* = int data , will be generated by another function
during calculating ,dynamically*/

bounding_boxes = (double (*)[6]) malloc( n_sheets *
sizeof(double[6]) );
…
free(bounding_boxes);
```

Contents

File Inclusion

#include is the preferred way to tie the declarations together for a **large program**.

It guarantees that all the source files will be **supplied** with the **same definitions** and variable declarations, and thus eliminates a particularly nasty kind of bug.

Naturally, when an included file is changed, **all files** that **depend** on it **must be recompiled**.

The C Preprocessor

```
#define MAX_PROMPT_LINES            3
#define MAX_LINES                   25
#define MAX_LINE_LEN                80
#define SMAT_STATUS_OK              MLEOK /* MLEOK = 0 */
#define SMAT_STATUS_FAIL            -1

#define SMAT_STATUS_DATA_ACTIVE      2
#define SMAT_STATUS_DATA_INACTIVE    3

#define N_ELEMENTS(array) (sizeof(array)/sizeof(array[0]))
```

Contents

Remember the value
set last time

```
static logical first_time = TRUE;

   if( first_time )
   {
       /* Initialize dialog global data */
        …

       first_time = FALSE;
   }
```

Contents

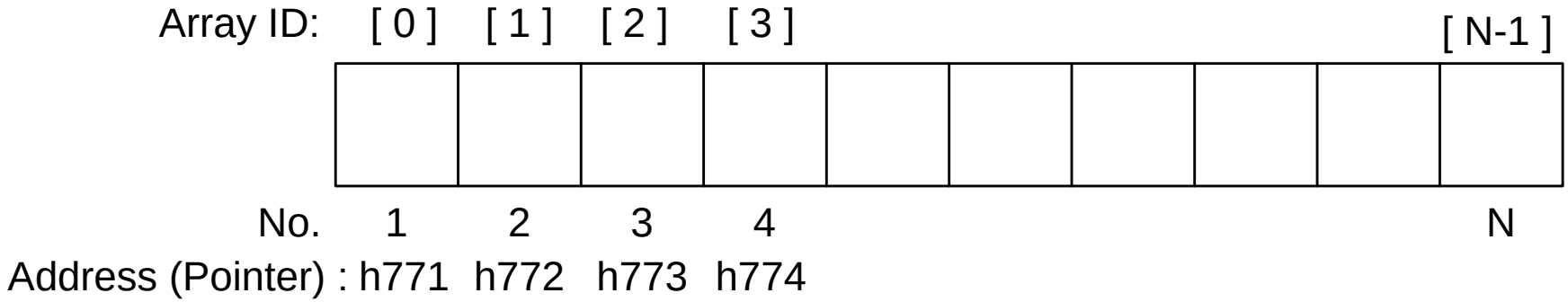
Files and Variables Naming
Function
Dynamic Memory Allocation
How to define a structure
Initialisation
An array with it's length known
An array with it's length un-know
The C Preprocessor
Remember the value set last time

à How to deal with "Memory Error "?

How to deal with "Memory Error "?

memory errors, such as "***Storage Memory Error***", "***Access Memory error***", the real reason of the memory errors is to use array **outside it's range.**

If one want to access the value '**lower limit**' or '**upper limit**' out of it's confine, the memory error will happen. See the figure below:



How to deal with "Memory Error "?

Array in C language ranges from **[0]~[n-1]**

1. Check lower limit

```
extern int normalise_data(
double   data[],          /* I */
int      xydata_length, /* I */
double   data_norm[ ]   /* O */ )
{
    int idx = 0;

    double  Scale_Operator  = 0.0;
    double  Offset_Operator = 0.0 ;

/* don't let array out of lower limit */

    if ( xydata_length <= 0 )
    return 1;
```

How to deal with "Memory Error"?

2. Check upper limit

2.1 ' < ' or ' <= ' in for ( ) ?

```
for (int idx = 0; idx < xydata_length ; idx++)

/*
for (int idx = 0; idx <= xydata_length-1 ; idx++)
 don't let array out of upper limit
*/
```

2.2 [- 1] or [+ 1] ?

```
 char    part_fspec[MAX_FSPEC_SIZE+1])   /* O */

for (int idx = 0; idx < length ; idx++);

for (int idx = 0; idx < = xydata_length - 1 ; idx++);
```

How to deal with "Memory Error"?

3. Check size limit

```
extern int translate_xydata(
    double              xdata[],           /* I/O */
    double              ydata[],           /* I/O */
    int                 xydata_length,     /* I */
    drawing_xyaxis_p_t drawing_xyaxis_ptr /* I */
                                                  )
{
   double *x_data_temp =
(double*)alloc(xydata_length*sizeof(double));

   double *y_data_temp =
(double *)alloc(xydata_length*sizeof(double));

…

free(x_data_temp);
free(y_data_temp);    }
```

How to deal with "Memory Error "?

C array:

```
data[0], data[1], data[N-1]

e.g.

data[idx] = inx*data(ind)
```

MATLAB array:

```
data(1), data(2),…,data(N)

e.g.
data = ones(4,4)

data =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> data(1,1) = 1
```

FAQ

FAQ 1 Useful Tools

FAQ 2 File/Function/Variable Naming

FAQ 1 Useful Tools

EmEditor Professional

Araxis Merge

Source.Insight

FAQ 2
File/Function/Variable Naming

Who can:

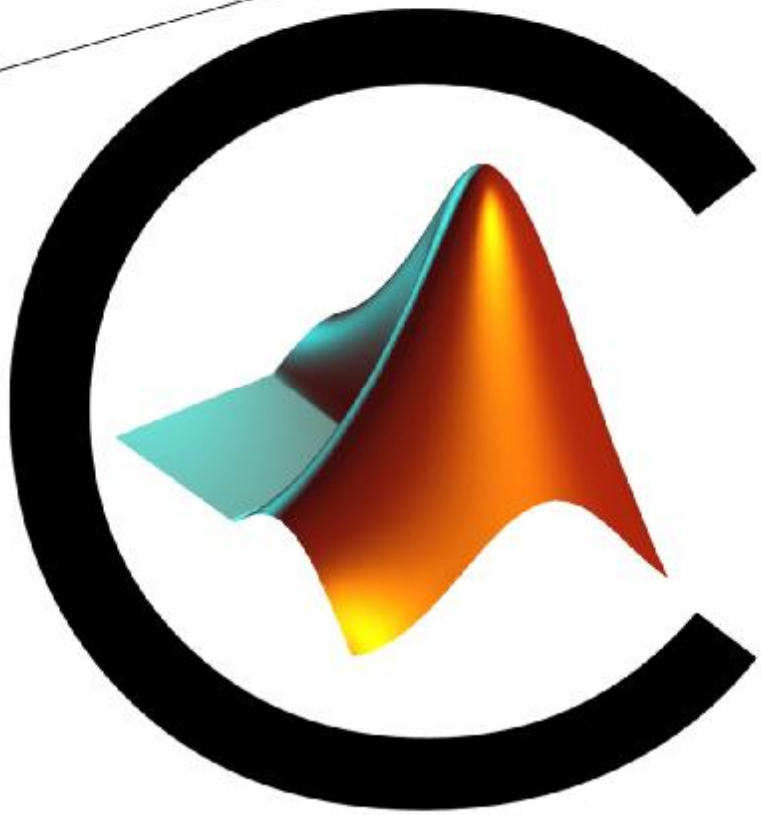
1) name a file: Project Manager

2) name a function: software project leader

3) name a variable: software developer

C Programming Practice No[6]

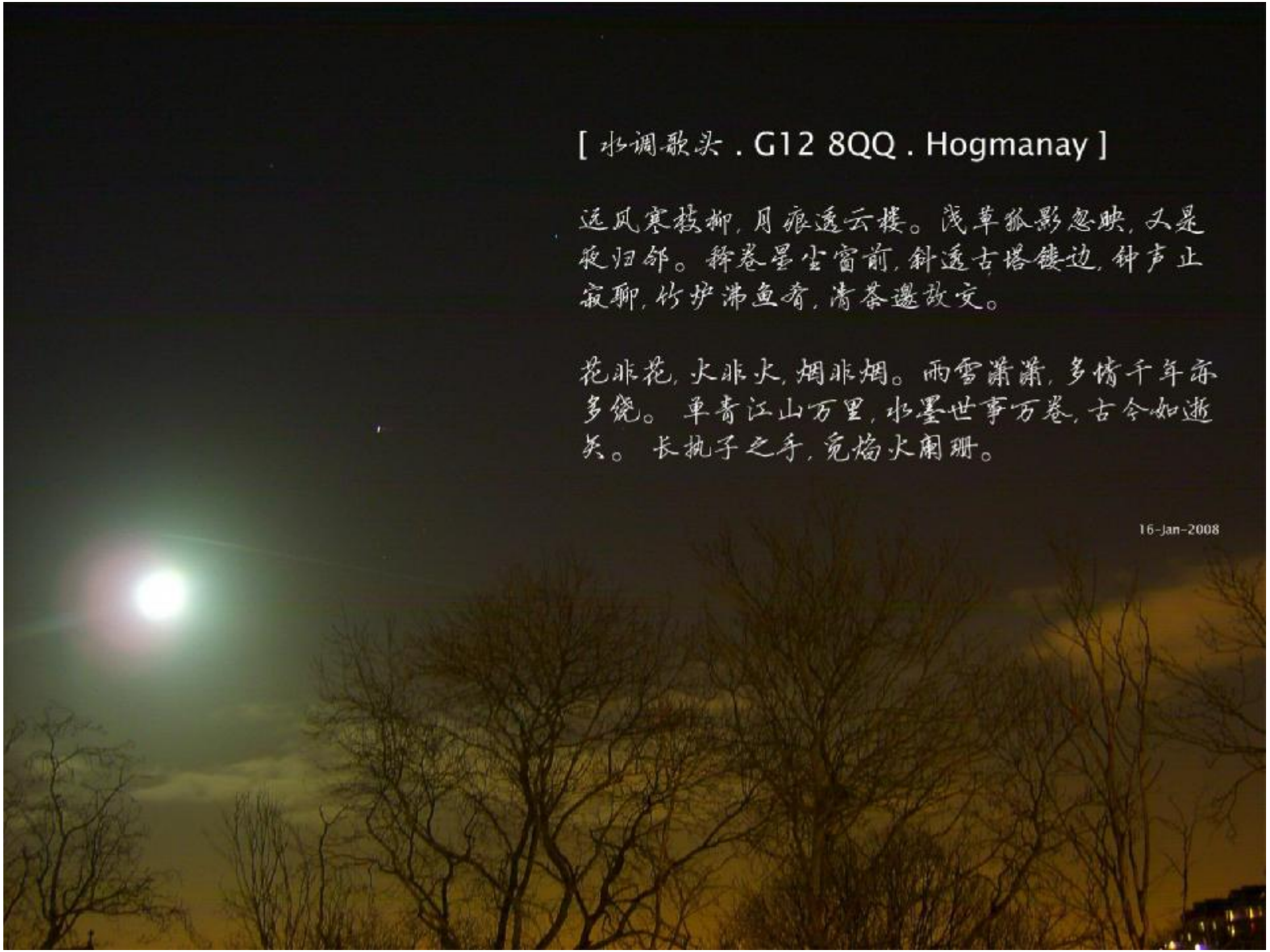
Chen , Yi
leo.chen.yi@live.co.uk
30-Jul-2010



Coding Tips

BETA 0.0.0.1

END


[水调歌头 . G12 8QQ . Hogmanay]
远风寒柳，月痕透云楼。浅草狐影忽映，人是夜归邻。释卷墨尘窗前，斜透古塔镂边，钟声止寂聊，竹炉沸鱼肴，清茶邀故交。
花非花，火非火，烟非烟。雨雪潇潇，多情千年亦多绕。 单青江山万里，水墨世事万卷，古今如逝矣。 长执子之手，免焰火阑珊。
16-Jan-2008