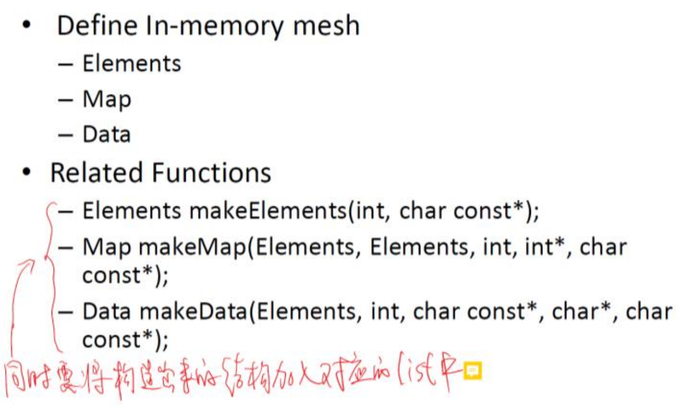
Mesh07 Memory Storage

19335015 陈恩婷

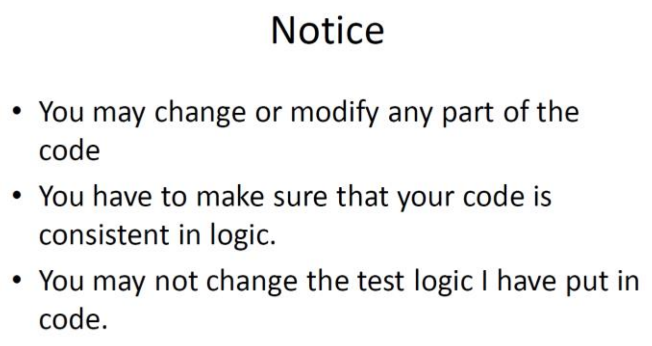
1. 项目目的

根据课件和程序中给出的函数原型和提示，完成 Elements、Map 和 Data 的创建函数，并实现 mesh 在二进制文件中的存储与读取，以及 Mesh 的初始化（清除掉 Mesh 中原有的数据）。相关课件截图如下：



- Define In-memory mesh
  - Elements
  - Map
  - Data
- Related Functions
  - Elements makeElements(int, char const*);
  - Map makeMap(Elements, Elements, int, int*, char const*);
  - Data makeData(Elements, int, char const*, char*, char const*);

同时要将构造出来的结构加入对应的list中



Notice

- You may change or modify any part of the code
- You have to make sure that your code is consistent in logic.
- You may not change the test logic I have put in code.

工程中原有的代码可以根据实际情况进行改动。

2. 为实现目的存在的各种技术问题

理解清楚 Elements，Maps 和 Data 结构体中各数据成员的具体含义

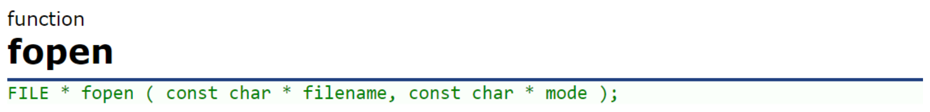
如何读写二进制文件，将数据深拷贝存入文件，并在读入时做相应的恢复

如何正确实现内存的申请与释放

3. 用什么算法、数据结构、语言机制解决这些问题

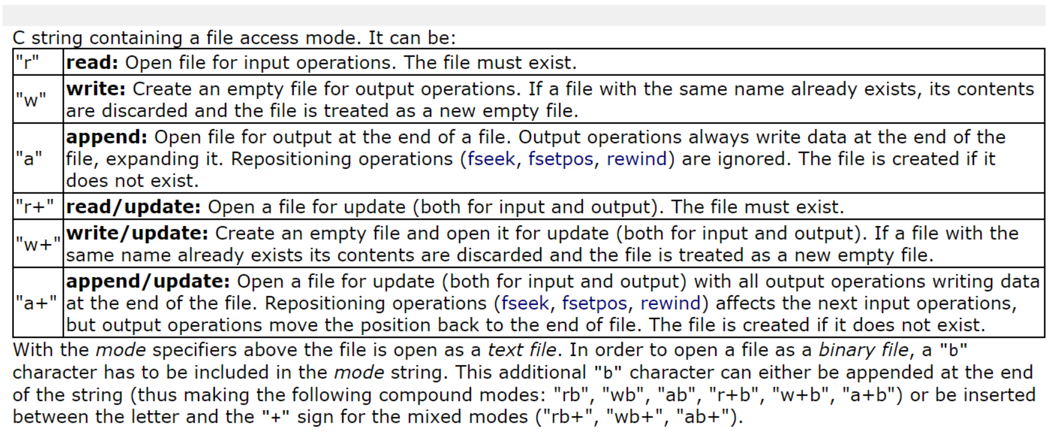
数据结构：Elements，Maps 和 Data 结构体

语言机制：MeshMemory 类中构造各结构体的成员函数



function
fopen

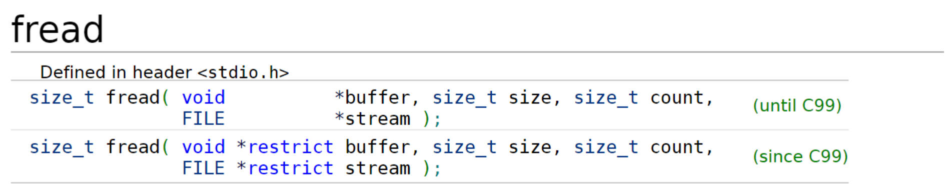
```
FILE * fopen ( const char * filename, const char * mode );
```



C string containing a file access mode. It can be:

"r"	**read:** Open file for input operations. The file must exist.
"w"	**write:** Create an empty file for output operations. If a file with the same name already exists, its contents are discarded and the file is treated as a new empty file.
"a"	**append:** Open file for output at the end of a file. Output operations always write data at the end of the file, expanding it. Repositioning operations (fseek, fsetpos, rewind) are ignored. The file is created if it does not exist.
"r+"	**read/update:** Open a file for update (both for input and output). The file must exist.
"w+"	**write/update:** Create an empty file and open it for update (both for input and output). If a file with the same name already exists its contents are discarded and the file is treated as a new empty file.
"a+"	**append/update:** Open a file for update (both for input and output) with all output operations writing data at the end of the file. Repositioning operations (fseek, fsetpos, rewind) affects the next input operations, but output operations move the position back to the end of file. The file is created if it does not exist.

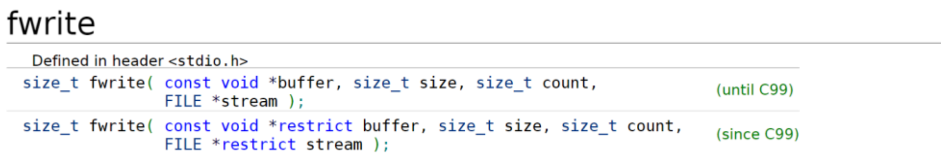
With the *mode* specifiers above the file is open as a *text file*. In order to open a file as a *binary file*, a "b" character has to be included in the *mode* string. This additional "b" character can either be appended at the end of the string (thus making the following compound modes: "rb", "wb", "ab", "r+b", "w+b", "a+b") or be inserted between the letter and the "+" sign for the mixed modes ("rb+", "wb+", "ab+").



fread

Defined in header <stdio.h>

```
size_t fread( void          *buffer, size_t size, size_t count,          (until C99)
              FILE          *stream );
size_t fread( void *restrict buffer, size_t size, size_t count,          (since C99)
              FILE *restrict stream );
```



fwrite

Defined in header <stdio.h>

```
size_t fwrite( const void *buffer, size_t size, size_t count,            (until C99)
               FILE *stream );
size_t fwrite( const void *restrict buffer, size_t size, size_t count,   (since C99)
               FILE *restrict stream );
```

4. 对应的程序框架和实现代码

构造结构体

比较简单，mesh 中的数据以及存储在相应的数组中了，只需要为结构体申请内存，将

各个数据成员赋好值，将数组指针也存入数据成员即可。如 makeMap 函数：

```
Map MeshMemory::makeMap( Elements map_from, Elements map_to, int
map_dim, int* map_map, char const* map_name ) {
    int arr_cnt = map_from->size;
    Map mapi = (Map)malloc( sizeof( map ) );
    mapi->index = map_list_index;
    mapi->from = map_from;
    mapi->to = map_to;
    mapi->dim = map_dim;
    mapi->map = map_map;
    strcpy( (char*)( mapi->name ), map_name );
    map_list[ map_list_index++ ] = (Map)mapi;
    return (Map)mapi;
}
```

将 mesh 写入二进制文件

框架如下：

```
bool MeshMemory::savetofile( const char* fileName ) {
    ( "writing in grid \n" );
    FILE* fp;
    if ( ( fp = fopen( fileName, "wb" ) ) == NULL ) {
        printf( "can't open file\n" );
        return 0;
    }

    writeHeader( fp );
    writeElements( fp );
    writeMaps( fp );
    writeData( fp );

    fclose( fp );
    return true;
}
```

首先，用 fopen 打开二进制文件，将 header 部分用 fprintf 写入二进制文件：

```
bool MeshMemory::writeHeader( FILE* fp ) {
    fprintf( fp, "#emd %d %d %d %d %d %d\n", element_list_size,
map_list_size, dat_list_size,
             element_list_index, map_list_index, dat_list_index );
    return 1;
}
```

再依次将 element_list，map_list 和 dat_list 写入文件，注意要分别写入结构体本身

与数据成员中的数据（深拷贝存到文件），如 writeData 函数：

```
bool MeshMemory::writeData( FILE* fp ) {
    for ( int i = 0; i < dat_list_index; i++ ) {
        fwrite(dat_list[i], sizeof(dat), 1, fp);
        Data d = dat_list[ i ];
        fwrite( d->data, sizeof( double ), d->dim * d->set->size, fp );
    }
    return 1;
}
```

从二进制文件中读取 mesh 的代码：

```
bool MeshMemory::readfromfile(const char* fileName) {
    FILE* fp;
    if ((fp = fopen(fileName, "rb")) == NULL) {
        printf("can't open file\n");
        return 0;
    }

    readHeader(fp);
    readElements(fp);
    readMaps(fp);
    readData(fp);

    fclose(fp);
    return true;
}
```

框架类似写入文件的函数，不同的是需要对 mesh 进行初始化：

```
void MeshMemory::init() {
    // to do sth}
    memset( element_list, 0, sizeof( Map ) * map_list_size );
    memset( map_list, 0, sizeof( Elements ) * element_list_size );
    memset( dat_list, 0, sizeof( Data ) * dat_list_size );
    element_list_index = 0;
    map_list_index = 0;
    dat_list_index = 0;
}
```

Init 函数对各个结构体进行 0 初始化，重新调整好各“index“的值。

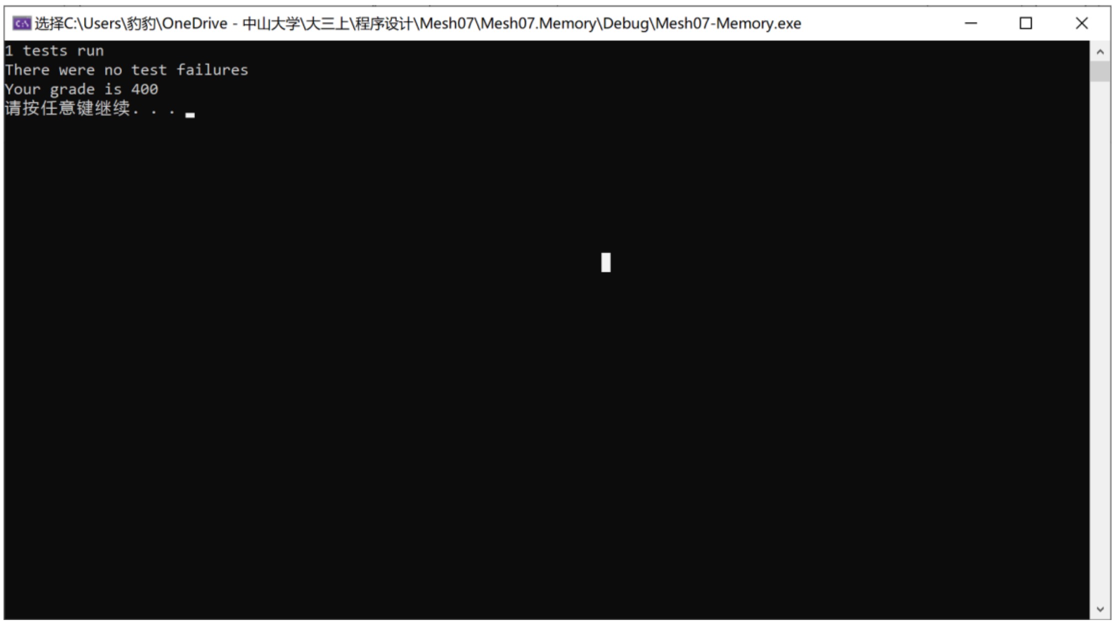
```
bool MeshMemory::readHeader(FILE* fp) {
    fscanf(fp, "#emd %d %d %d %d %d %d\n", &element_list_size,
&map_list_size, &dat_list_size,
        &element_list_index, &map_list_index, &dat_list_index);
    return 1;
}
```

接着再读入各结构体，对于每个结构体，先申请其内存，读入数据成员，再为相应的

数组数据成员分配内存并读入数据（深拷贝）。如 readData：

```
bool MeshMemory::readData( FILE* fp ) {
    for ( int i = 0; i < dat_list_index; i++ ) {
        Data tmp_dat = (Data)malloc( sizeof( dat ) );
        fread( tmp_dat, sizeof( dat ), 1, fp );
        tmp_dat->data = (char*)malloc( sizeof( double ) * tmp_dat->dim *
tmp_dat->set->size );
        fread( tmp_dat->data, sizeof( double ), tmp_dat->dim *
tmp_dat->set->size, fp );
        dat_list[ i ] = tmp_dat;
    }
    double* pd = (double*)( dat_list[ 3 ]->data );
    return 1;
}
```

5. 实验结果和结论


选择C:\Users\豹豹\OneDrive - 中山大学\大三上\程序设计\Mesh07\Mesh07.Memory\Debug\Mesh07-Memory.exe

```
1 tests run
There were no test failures
Your grade is 400
请按任意键继续. . .
```

如图所示，成功完成了 mesh 的读写和存储。