

Mesh08 Mesh Applications (Programs)

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，实现 Program.cpp 中的函数。相关课件截

图如下：

- Mesh application (program) to solve the above problem.
- Class Program
 - Kernel functions
 - help functions
 - Main program (User algorithm)

工程中原有的代码可以根据实际情况进行改动，尤其是 `CHECK_DOUBLES_EQUAL` 部分。

2. 为实现目的存在的各种技术问题

参考课件中的公式和 Matlab 程序，理解清楚需要实现的函数的计算逻辑

学会将类的函数成员作为参数传递给另一个函数成员

学会初始化类中的引用数据成员

3. 用什么算法、数据结构、语言机制解决这些问题

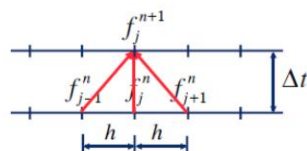
算法：近似公式

Finite Difference Approximations

Thus, given f at one time (or time level), f at the next time level is given by:

$$f_j^{n+1} = f_j^n - \frac{U\Delta t}{2h}(f_{j+1}^n - f_{j-1}^n) + \frac{D\Delta t}{h^2}(f_{j+1}^n - 2f_j^n + f_{j-1}^n)$$

The value of every point at level $n+1$ is given explicitly in terms of the values at the level n



供参考的 Matlab 代码

```
% one-dimensional advection-diffusion by the FTCS scheme
n=21; nstep=100; length=2.0; h=length/(n-1); dt=0.05; D=0.05;
f=zeros(n,1); y=zeros(n,1); ex=zeros(n,1); time=0.0;
for i=1:n, f(i)=0.5*sin(2*pi*h*(i-1)); end; % initial conditions
for m=1:nstep, m, time
    for i=1:n, ex(i)=exp(-4*pi*pi*D*time)*...
        0.5*sin(2*pi*h*(i-1)-time)); end; % exact solution 精确解
    hold off; plot(f,'linewidth',2); axis([1 n -2.0, 2.0]); % plot solution
    hold on; plot(ex,'r','linewidth',2); pause; % plot exact solution
    y=f; % store the solution
    for i=2:n-1, % advection by centered differences 离散化中心差分
        f(i)=y(i)-0.5*(dt/h)*(y(i+1)-y(i-1))+...
            D*(dt/h^2)*(y(i+1)-2*y(i)+y(i-1)));
    end;
    f(n)=y(n)-0.5*(dt/h)*(y(2)-y(n-1))+...
        D*(dt/h^2)*(y(2)-2*y(n)+y(n-1))); % do endpoints for % periodic boundaries 边界
    f(1)=f(n);
    time=time+dt;
end;
```

语言机制：1. 将函数作为参数传递给另一个参数

函数原型如下：

```
int loop(double (Program::*fx)(double* a), int element, int data);
```

实际调用的代码如下：

```
loop( &Program::advection_diffusion, e, d );
```

2. 初始化类的引用成员

声明如下：Mesh & s;

引用成员的初始化必须使用初始化列表，如下面解释的

c++类内可以定义引用成员变量，但要遵循以下三个规则：

- 1、不能用默认构造函数初始化，必须提供构造函数来初始化引用成员变量。否则会造成引用未初始化错误。
- 2、构造函数的形参也必须是引用类型
- 3、不能在构造函数里初始化，必须在初始化列表中进行初始化。

构造函数分为初始化和计算两个阶段，前者对应成员初始化链表，后者对应构造函数函数体。引用必须在初始化阶段，也即在成员初始化链表中完成，否则编译时会报错（引用未初始化）。

来源：[c++之类内定义引用成员](#)

4. 对应的程序框架和实现代码

Mesh 部分的代码和之前的实验基本一致，这里就不赘述了。

Program 类的声明与框架

```
class Program
{
    double dt, h, U, D;
    Mesh & s;
public:
    Program(Mesh & s);
    Program(Mesh & s, double dt, double h, double U, double D);
    ~Program();
    int init();
    int average(int d, int a, int b); //没有用到
    double advection_diffusion(double *f); //kernel function

    int loop(double (Program::*fx)(double* a), int element, int data);
    // 把fx在数组上操作一遍
    int program(int nstep, int e, int d);
};
```

以上是 Program 类的声明，主要需要自己实现的函数包括 program, loop, init,

advection_diffusion 等。主要代码如下：

```
int Program::init() { //需要修改
    int n = s.element_list[ 0 ]->size;
    double* dp = (double*)malloc( sizeof( double ) * ( n ) );
    double* dp3 = (double*)s.dat_list[ 3 ]->data;
    for ( int i = 0; i < n; i++ ) {
        dp[ i ] = 0.5 * sin( 2 * 3.1415926 * h * i );
    }
    s.makeData( s.element_list[ 0 ], 1, "double", (char*)dp,
    "data2_on_nodes" );
    return 0;
}
```

Init函数主要负责创建并用正弦函数初始化用于计算的数组。

```
double Program::advection_diffusion( double* f ) { //需要修改
    double fx;
    fx = *f - U * dt * ( *( f + 1 ) - *( f - 1 ) ) / ( 2 * h ) + D * dt *
    ( *( f + 1 ) + *( f - 1 ) - 2 * ( *f ) ) / ( h * h );
    // cout << fx << " ";
    return fx;
}
```

advection_diffusion就是程序中的核函数，负责数组局部的计算。

```

int Program::loop( double ( Program::*fx )( double* a ), int e, int d ) {
    int n = s.element_list[ e ]->size;
    double* f1 = (double*)s.dat_list[ d ]->data;
    double* pp = (double*)malloc( sizeof( double ) * n );
    memset( pp, 0, sizeof( double ) * n );
    for ( int i = 0; i < n; i++ ) {
        pp[ i ] = ( (double*)( s.dat_list[ d ]->data ) )[ i ];
    }
    for ( int i = 1; i < n - 1; i++ ) {
        f1[ i ] = advection_diffusion( pp + i );
    }
    f1[ n - 1 ] = pp[ n - 1 ] - U * dt * ( pp[ 1 ] - pp[ n - 2 ] ) / ( 2 *
h ) + D * dt * ( pp[ 1 ] - 2 * pp[ n - 1 ] + pp[ n - 2 ] ) / ( h * h );
    f1[ 0 ] = f1[ n - 1 ];
    return 0;
}

```

Loop函数主要负责拷贝原数组中的数据，并调用advection_diffusion计算新数据。

```

int Program::program( int nstep, int e, int d ) { //需要修改
    double tm = 0;
    int n = s.element_list[ e ]->size;
    for ( int j = 0; j < nstep; j++, tm += dt ) {
        loop( &Program::advection_diffusion, e, d );
    }
    return 0;
}

```

Program 函数主要负责最外层的循环，每次循环都调用 loop 函数完成数据的更新。

5. 实验结果和结论

我根据 Program.cpp 中给出的参数，修改了课件中的 matlab 代码并运行，结果如下：

```

1 % one-dimensional advection-diffusion by the FTCS scheme
2 n=16; nstep=200; length=2.0; h=0.1; dt=0.05; D=0.05;
3 f=zeros(n,1); y=zeros(n,1); ex=zeros(n,1); time=0.0;
4 for i=1:n, f(i)=0.5*sin(2*pi*h*(i-1)); end; % initial conditions
5 f
6 for m=1:nstep, m; time;
7 %for i=1:n, ex(i)=exp(-4*pi*pi*D*time)*...
8 %0.5*sin(2*pi*h*(i-1)-time)); end; % exact solution
9 %hold off; plot(f,'linewidth',2); axis([1 n -2.0, 2.0]); % plot s
10 %hold on; plot(ex,'r','linewidth',2);pause; % plot exact solution
11 y=f; % store the solution
12 for i=2:n-1,
13 f(i)=y(i)-0.5*(dt/h)*(y(i+1)-y(i-1))+...
14 D*(dt/h^2)*(y(i+1)-2*y(i)+y(i-1)); % advection by centered differer
15 end;
16 f(n)=y(n)-0.5*(dt/h)*(y(2)-y(n-1))+...
17 D*(dt/h^2)*(y(2)-2*y(n)+y(n-1)); % do endpoints for
18 f(1)=f(n); % periodic boundaries
19 time=time+dt;
20 end;
21 f
22

```

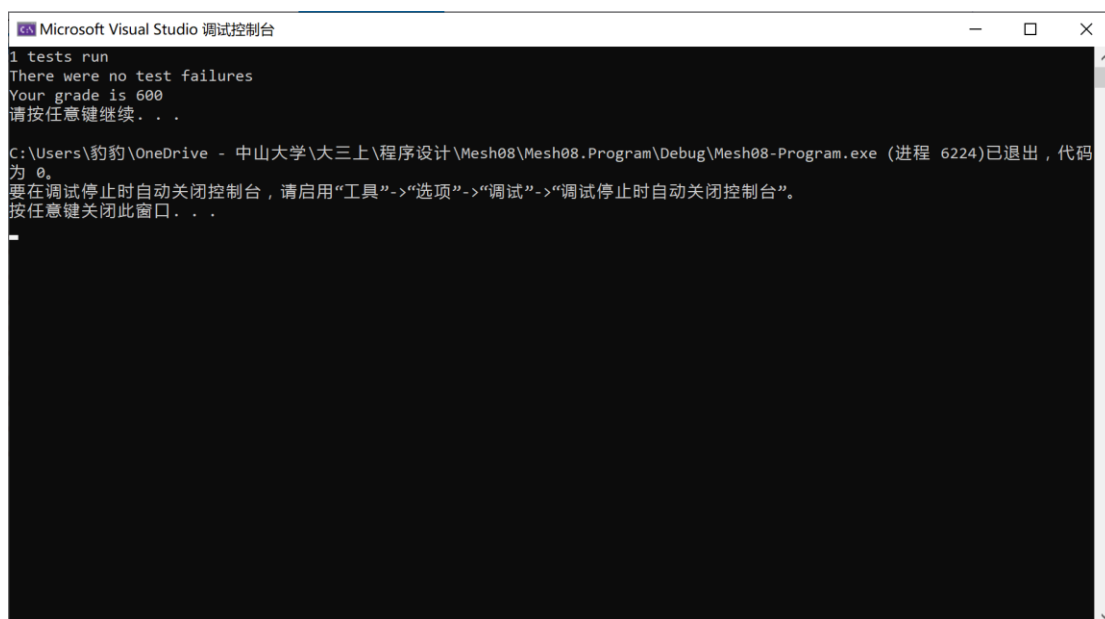
```

f = 16x1
0
0.293892626146237
0.475528258147577
0.475528258147577
0.293892626146237
0.000000000000000
-0.293892626146237
-0.475528258147577
-0.475528258147577
-0.293892626146237
:
:
:
f = 16x1
0.100331361031368
0.098935780149593
0.098171952784043
0.098171952784043
0.098935780149593
0.100331361031368
0.102117385265855
0.103985032120950
0.105611368812533
0.106715188302078
:
:
:

```

用右侧的 f 的结果修改 C++ 代码中的 `CHECK_DOUBLES_EQUAL` 部分，再运行程序，发现结

果和 Matlab 中的结果一样：



```
Microsoft Visual Studio 调试控制台
1 tests run
There were no test failures
Your grade is 600
请按任意键继续. . .

C:\Users\豹豹\OneDrive - 中山大学\大三上\程序设计\Mesh08\Mesh08.Program\Debug\Mesh08-Program.exe (进程 6224)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭窗口. . .
```