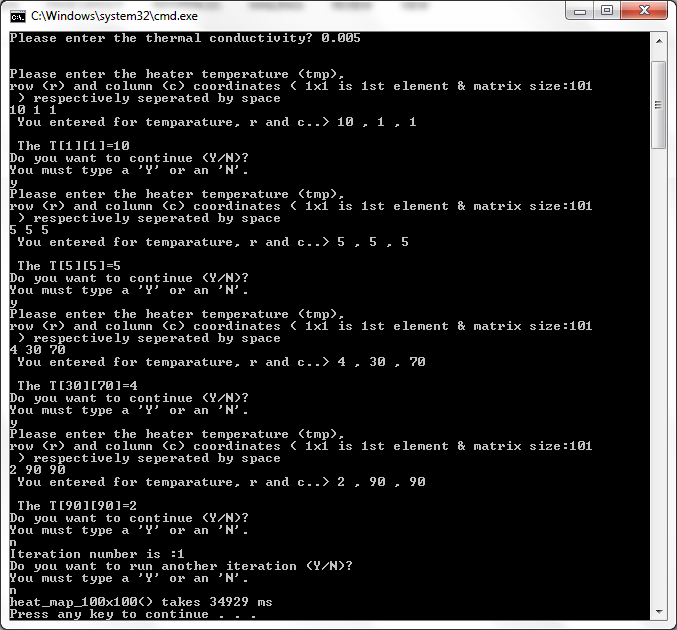
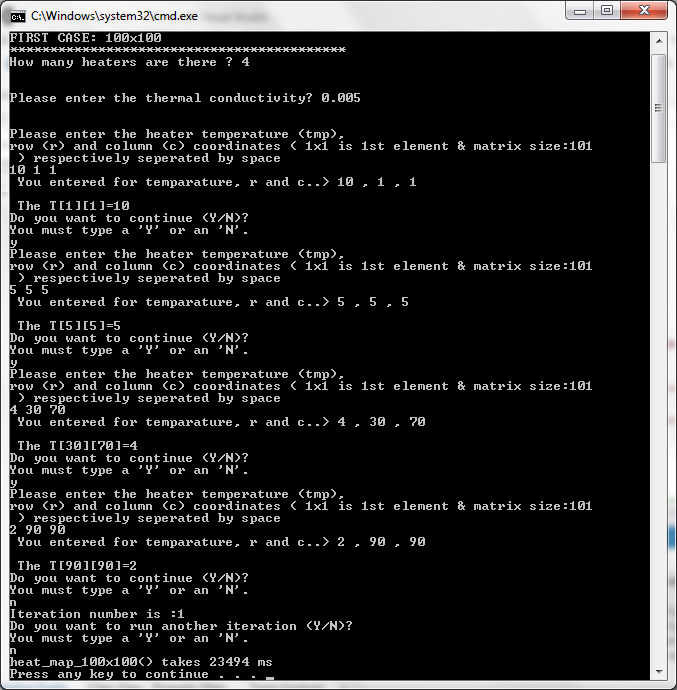
Case 1: 100x100 – Non Parallel



Case 1: 100x100 – Parallel

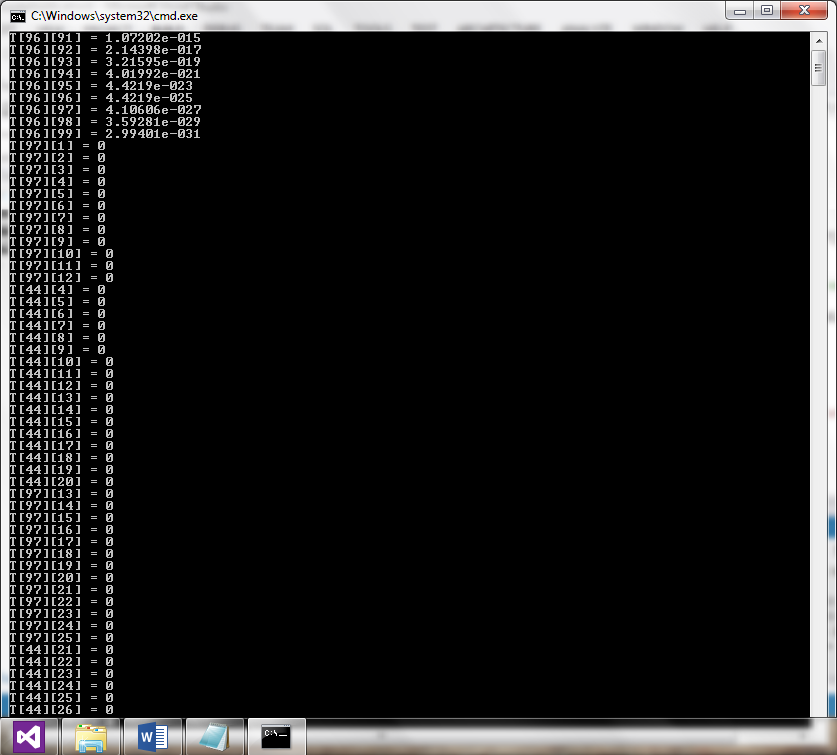


Below is example of the heat map for 100x100 case:

K=0.005, # of heaters=4, T[1][1]=10, T[5][5]=5, T[30][70]=4, T[90][90]=2 are entered by user. And one iteration is selected. I set the number of thread as 4 under assumption of the laptop has 4 cores. Each thread will handle 25 rows in its own core in concurrently. The parallel section is the heat computation section, the rest is master thread of course is serial.

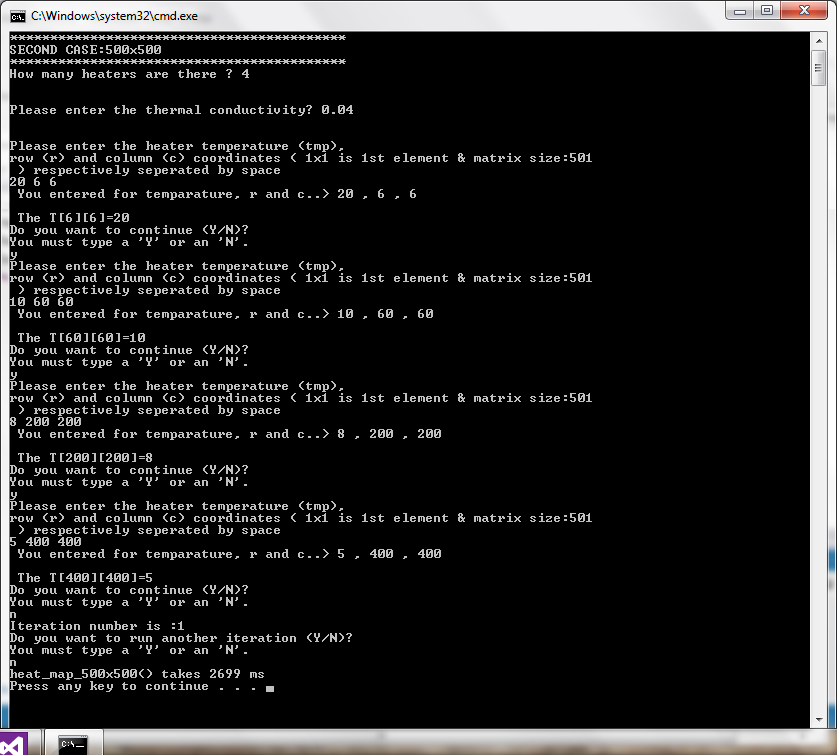
Non-paralel : 34929 ms vs. Parallel-version: 23494 ms

There is no way that we can fit 100x100 matrix elements or more in the command window screen.

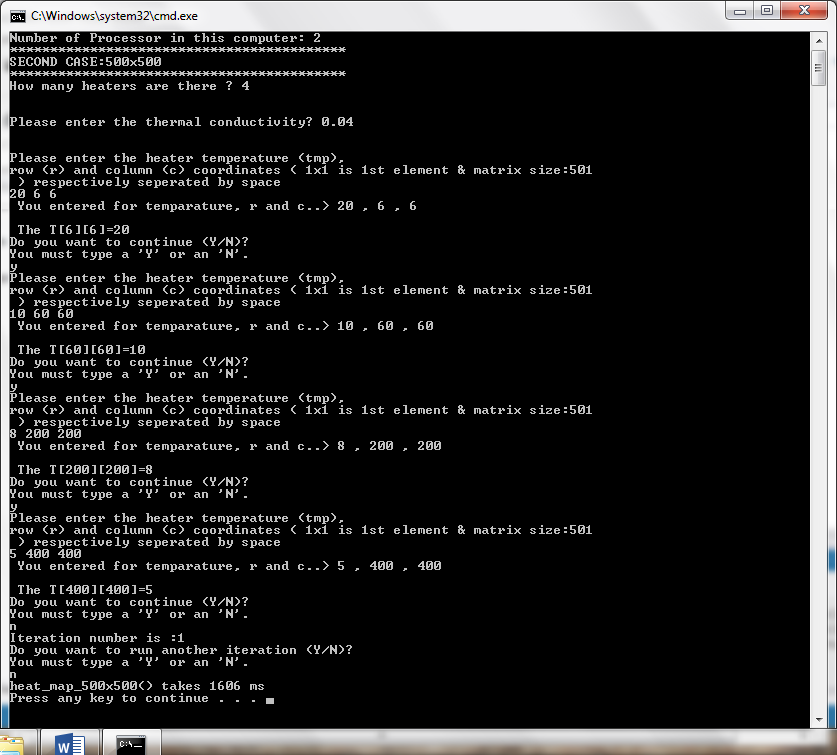


------------------------------------------------------------------------------------------------------------------------------------------

Case 2: 500x500 – Non Parallel



Case 2: 500x500 –Parallel



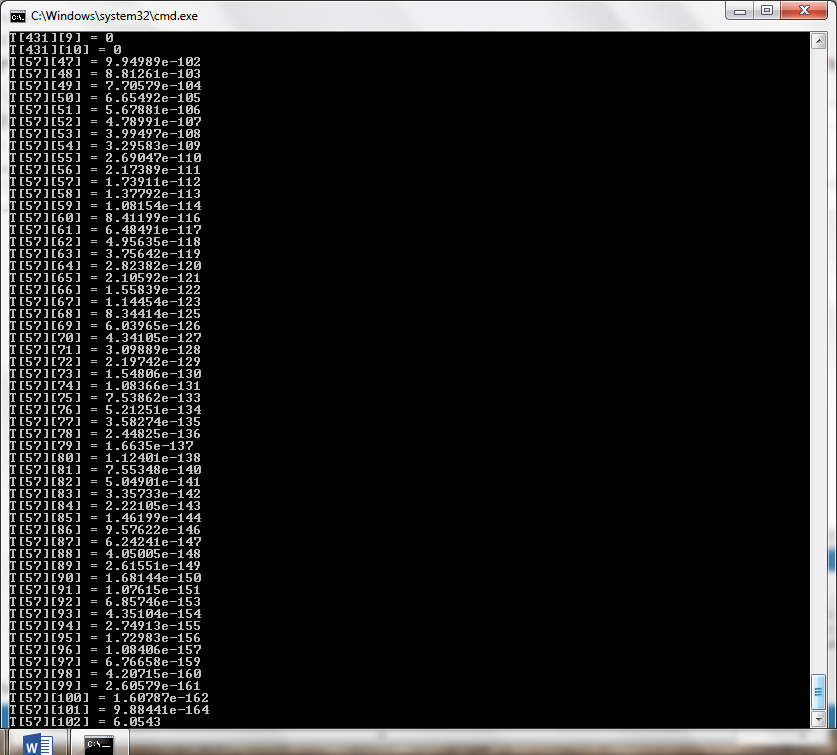
Below is example of the heat map for 500x500 case:

K=0.04, # of heaters=4, T[6][6]=20, T[60][60]=10, T[200][200]=8, T[400][400]=5 are entered by user. And one iteration is selected.

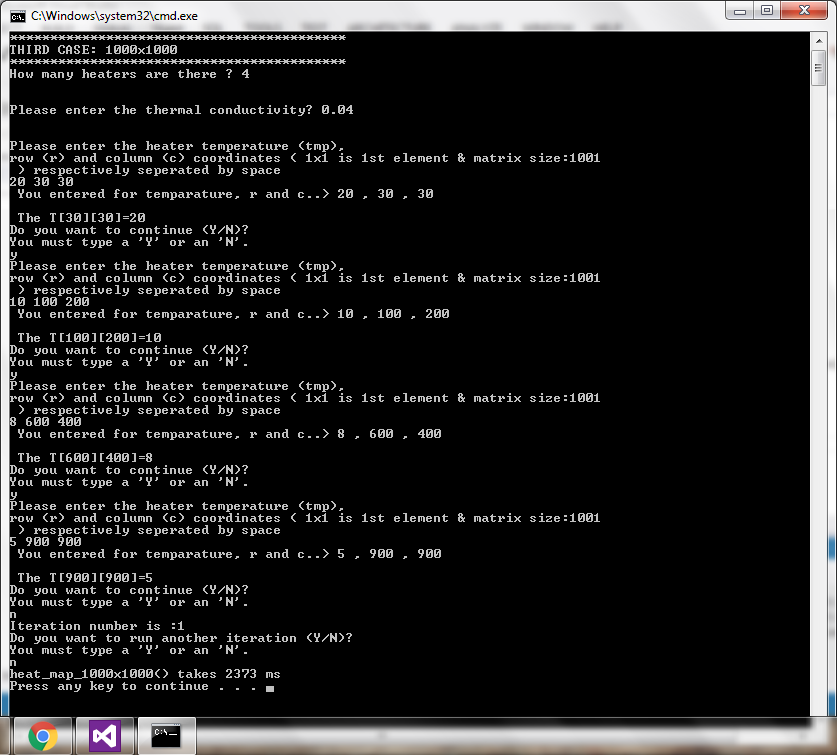
I set the number of thread as 4 under assumption of the laptop has 4 cores. Each thread will handle 25 rows in its own core in concurrently. The parallel section is the heat computation section, the rest is master thread of course is serial.

Non-paralel : 2699 ms vs. Parallel-version: 1606 ms

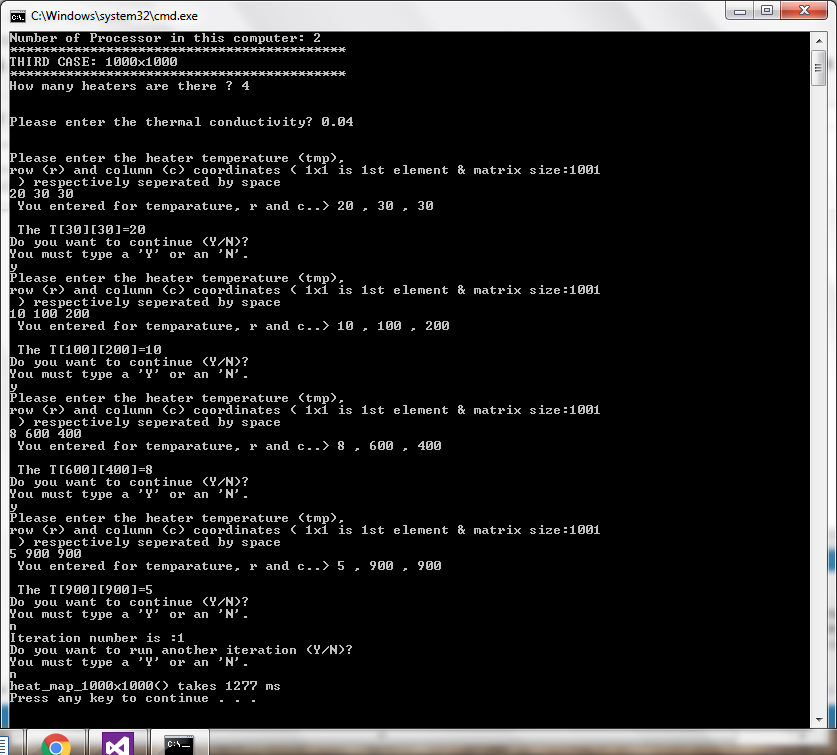
There is no way that we can fit 500x500 (250,000 elements) matrix elements or more in the command window screen.



Case 3: 1000x1000 – Non-Parallel



Case 3: 1000x1000 –Parallel



In these case 3, the following parameters are selected:

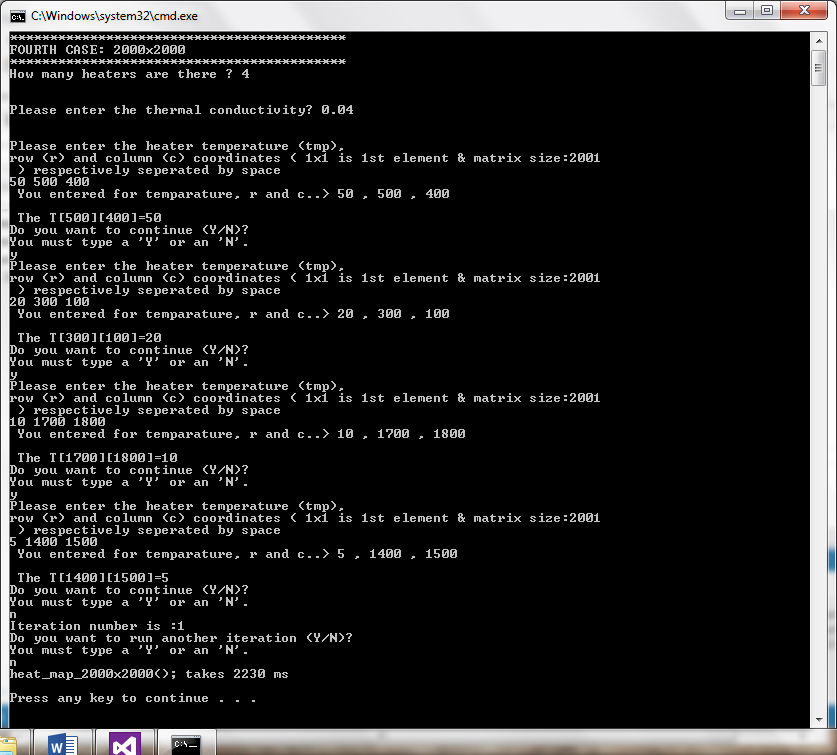
K=0.04, # of heaters=4, T[30][30]=20, T[100][200]=10, T[600][400]=8, T[900][900]=5 are entered by user. And one iteration is selected.

I set the number of thread as 4 under assumption of the laptop has 4 cores. Each thread will handle 25 rows in its own core in concurrently. The parallel section is the heat computation section, the rest is master thread of course is serial.

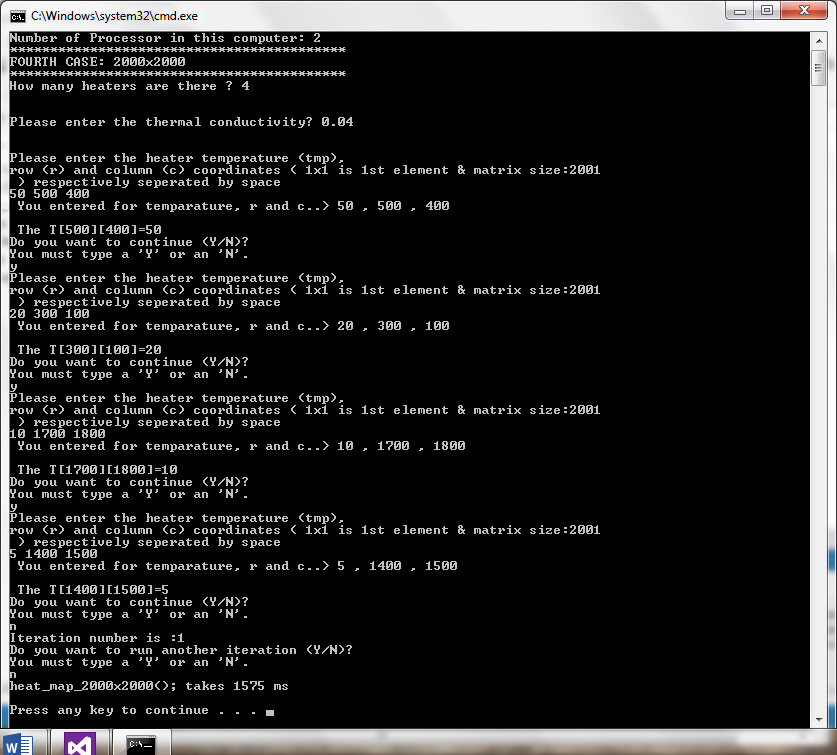
Non-paralel : 2373 ms vs. Parallel-version: 1277 ms

There is no way that we can fit 1000x1000 (250,000 elements) matrix elements or more in the command window screen.

Case 4: 2000x2000 –Non- Parallel



Case 4: 2000x2000 –Parallel



In these case 4, the following parameters are selected:

K=0.04, # of heaters=4, T[500][400]=50, T[300][100]=20, T[1700][1800]=10, T[1400][1500]=5 are entered by user. And one iteration is selected.

I set the number of thread as 4 under assumption of the laptop has 4 cores. Each thread will handle 25 rows in its own core in concurrently. The parallel section is the heat computation section, the rest is master thread of course is serial.

Non-paralel : 2230 ms vs. Parallel-version: 1575 ms

There is no way that we can fit 1000x1000 (250,000 elements) matrix elements or more in the command window screen.

#include<iostream>

#include<time.h>

#include<algorithm>

#include<ctime>

#include<typeinfo>

#include<cstdlib>

#include<limits>

#include<omp.h>

using namespace std;

void heat\_map\_100x100()

{

clock\_t start\_t, end\_t, total\_t;

const int size=101;

char ans,ans2;

int htr,r,c,count,i,j;

count=0;

double H,tmp,tcon;

double (\*T)[size]=new double[size][size];

for (int k = 0; k < size; ++k)

for (int l= 0; l < size; ++l)

{T[k][l] = 0;

}

omp\_set\_nested(0);

cout<<"How many heaters are there ? ";

cin>>htr;

cout<<endl<<endl;

cout<<"Please enter the thermal conductivity? ";

cin>>tcon;

cout<<endl<<endl;

do

{

cout<<"Please enter the heater temperature (tmp),"<<endl;

cout<<"row (r) and column (c) coordinates ( 1x1 is 1st element & matrix size:"<<size<<endl;

cout<<" ) respectively seperated by space "<<endl;

cin>>tmp>>r>>c;

if ((r>size)||(c>size)||(tmp<0))

{

cout<< "You entered r or c value are out of border, please enter proper value !!!"<<endl<<endl;

ans='y';

}

else {

cout<<" You entered for temparature, r and c..> "<<tmp<<" , "<<r<<" , "<<c<<endl<<endl;

T[r][c]=tmp;

cout<<" The T["<<r<<"]["<<c<<"]="<<T[r][c]<<endl;

//cout<<" The A["<<10<<"]["<<10<<"]="<<tmp<<endl;

cout<< "Do you want to continue (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans;

}

} while (ans != 'N' && ans != 'n');

do {

start\_t = clock();

#pragma omp parallel for private(j)

for (i = 1; i < size-1; ++i) {

for (j= 1; j < size-1; ++j)

{

T[i][j]= T[i][j]+tcon\*(T[i-1][j]+T[i+1][j]+T[i][j-1]+T[i][j+1]-4\*T[i][j]);

//#pragma omp critical

// cout<<"From Thread#"<<omp\_get\_thread\_num()<<endl;

//#pragma omp critical

cout<<"T["<<i<<"]["<<j<<"] = "<<T[i][j]<<endl;

}

}

count=count+1;

cout<< "Iteration number is :"<<count<<endl;\_

cout<< "Do you want to run another iteration (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans2;

} while (ans2 != 'N' && ans2 != 'n');

end\_t = clock();

total\_t = (double)(end\_t - start\_t);

cout<<"heat\_map\_100x100() takes "<<total\_t<<" ms"<<endl;

}

void heat\_map\_500x500()

{

clock\_t start\_t, end\_t, total\_t;

const int size=501;

char ans,ans2;

int htr,r,c,count,i,j;

double H,tmp,tcon;

double (\*T)[size]=new double[size][size];

for (int k = 0; k < size; ++k)

for (int l= 0; l < size; ++l)

{T[k][l] = 0;

}

omp\_set\_nested(0);

cout<<"How many heaters are there ? ";

cin>>htr;

cout<<endl<<endl;

cout<<"Please enter the thermal conductivity? ";

cin>>tcon;

cout<<endl<<endl;

do

{

cout<<"Please enter the heater temperature (tmp),"<<endl;

cout<<"row (r) and column (c) coordinates ( 1x1 is 1st element & matrix size:"<<size<<endl;

cout<<" ) respectively seperated by space "<<endl;

cin>>tmp>>r>>c;

if ((r>size)||(c>size)||(tmp<0))

{

cout<< "You entered r or c value are out of border, please enter proper value !!!"<<endl<<endl;

ans='y';

}

else {

cout<<" You entered for temparature, r and c..> "<<tmp<<" , "<<r<<" , "<<c<<endl<<endl;

T[r][c]=tmp;

cout<<" The T["<<r<<"]["<<c<<"]="<<T[r][c]<<endl;

//cout<<" The A["<<10<<"]["<<10<<"]="<<tmp<<endl;

cout<< "Do you want to continue (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans;

}

} while (ans != 'N' && ans != 'n');

count=0;

do {

start\_t = clock();

#pragma omp parallel for private(j)

for (i = 1; i < size-1; ++i) {

for (j= 1; j < size-1; ++j)

{

T[i][j]= T[i][j]+tcon\*(T[i-1][j]+T[i+1][j]+T[i][j-1]+T[i][j+1]-4\*T[i][j]);

//#pragma omp critical

//cout<<"From Thread#"<<omp\_get\_thread\_num()<<endl;

//#pragma omp critical

cout<<"T["<<i<<"]["<<j<<"] = "<<T[i][j]<<endl;

}

}

count=count+1;

cout<< "Iteration number is :"<<count<<endl;\_

cout<< "Do you want to run another iteration (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans2;

} while (ans2 != 'N' && ans2 != 'n');

end\_t = clock();

total\_t = (double)(end\_t - start\_t);

cout<<"heat\_map\_500x500() takes "<<total\_t<<" ms"<<endl;

}

void heat\_map\_1000x1000()

{

clock\_t start\_t, end\_t, total\_t;

const int size=1001;

char ans,ans2;

int htr,r,c,count,i,j;

double H,tmp,tcon;

double (\*T)[size]=new double[size][size];

for (int k = 0; k < size; ++k)

for (int l= 0; l < size; ++l)

{T[k][l] = 0;

}

omp\_set\_nested(0);

cout<<"How many heaters are there ? ";

cin>>htr;

cout<<endl<<endl;

cout<<"Please enter the thermal conductivity? ";

cin>>tcon;

cout<<endl<<endl;

do

{

cout<<"Please enter the heater temperature (tmp),"<<endl;

cout<<"row (r) and column (c) coordinates ( 1x1 is 1st element & matrix size:"<<size<<endl;

cout<<" ) respectively seperated by space "<<endl;

cin>>tmp>>r>>c;

if ((r>size)||(c>size)||(tmp<0))

{

cout<< "You entered r or c value are out of border, please enter proper value !!!"<<endl<<endl;

ans='y';

}

else {

cout<<" You entered for temparature, r and c..> "<<tmp<<" , "<<r<<" , "<<c<<endl<<endl;

T[r][c]=tmp;

cout<<" The T["<<r<<"]["<<c<<"]="<<T[r][c]<<endl;

//cout<<" The A["<<10<<"]["<<10<<"]="<<tmp<<endl;

cout<< "Do you want to continue (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans;

}

} while (ans != 'N' && ans != 'n');

count=0;

do {

start\_t = clock();

#pragma omp parallel for private(j)

for (i = 1; i < size-1; ++i) {

for (j= 1; j < size-1; ++j)

{

T[i][j]= T[i][j]+tcon\*(T[i-1][j]+T[i+1][j]+T[i][j-1]+T[i][j+1]-4\*T[i][j]);

//#pragma omp critical

//cout<<"From Thread#"<<omp\_get\_thread\_num()<<endl;

//#pragma omp critical

cout<<"Z["<<i<<"]["<<j<<"] = "<<T[i][j]<<endl;

}

}

count=count+1;

cout<< "Iteration number is :"<<count<<endl;\_

cout<< "Do you want to run another iteration (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans2;

} while (ans2 != 'N' && ans2 != 'n');

end\_t = clock();

total\_t = (double)(end\_t - start\_t);

cout<<"heat\_map\_1000x1000() takes "<<total\_t<<" ms"<<endl;

}

void heat\_map\_2000x2000()

{

clock\_t start\_t, end\_t, total\_t;

const int size=2001;

char ans,ans2;

int htr,r,c,count,i,j;

double H,tmp,tcon;

double (\*T)[size]=new double[size][size];

for (int k = 0; k < size; ++k)

for (int l= 0; l < size; ++l)

{T[k][l] = 0;

}

omp\_set\_nested(0);

cout<<"How many heaters are there ? ";

cin>>htr;

cout<<endl<<endl;

cout<<"Please enter the thermal conductivity? ";

cin>>tcon;

cout<<endl<<endl;

do

{

cout<<"Please enter the heater temperature (tmp),"<<endl;

cout<<"row (r) and column (c) coordinates ( 1x1 is 1st element & matrix size:"<<size<<endl;

cout<<" ) respectively seperated by space "<<endl;

cin>>tmp>>r>>c;

if ((r>size)||(c>size)||(tmp<0))

{

cout<< "You entered r or c value are out of border, please enter proper value !!!"<<endl<<endl;

ans='y';

}

else {

cout<<" You entered for temparature, r and c..> "<<tmp<<" , "<<r<<" , "<<c<<endl<<endl;

T[r][c]=tmp;

cout<<" The T["<<r<<"]["<<c<<"]="<<T[r][c]<<endl;

//cout<<" The A["<<10<<"]["<<10<<"]="<<tmp<<endl;

cout<< "Do you want to continue (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans;

}

} while (ans != 'N' && ans != 'n');

count=0;

do {

start\_t = clock();

#pragma omp parallel for private(j)

for (i = 1; i < size-1; ++i) {

for (j= 1; j < size-1; ++j)

{

T[i][j]= T[i][j]+tcon\*(T[i-1][j]+T[i+1][j]+T[i][j-1]+T[i][j+1]-4\*T[i][j]);

//#pragma omp critical

//cout<<"From Thread#"<<omp\_get\_thread\_num()<<endl;

//#pragma omp critical

cout<<"Z["<<i<<"]["<<j<<"] = "<<T[i][j]<<endl;

}

}

count=count+1;

cout<< "Iteration number is :"<<count<<endl;\_

cout<< "Do you want to run another iteration (Y/N)?\n";\_

cout<< "You must type a 'Y' or an 'N'.\n";\_

cin >> ans2;

} while (ans2 != 'N' && ans2 != 'n');

end\_t = clock();

total\_t = (double)(end\_t - start\_t);

cout<<"heat\_map\_2000x2000(); takes "<<total\_t<<" ms"<<endl<<endl;

}

int main ()

{

omp\_set\_num\_threads (4);

//set OMP\_NUM\_THREADS=4;

int x=omp\_get\_num\_procs ();

//double h;

cout<<"Number of Processor in this computer: "<<x<<endl;

//heat\_map\_100x100();

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"FIRST CASE: 100x100 "<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

heat\_map\_100x100();

// second case

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"SECOND CASE:500x500 "<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

heat\_map\_500x500();

// third case

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"THIRD CASE: 1000x1000 "<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

heat\_map\_1000x1000();

// fourth case

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"FOURTH CASE: 2000x2000 "<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

heat\_map\_2000x2000();

return 0;

}