

```

/*-----
* CSI702 :Bob Sorensen
*=====
* Due: TBD
* Assignment: Project
* Name      async_openmp_project_sorensen
* Date      Spring 2010
* -----
* Number of files in project:  2
* -----
* Jacobi Iteration with async openmp operation
*-----*/

/* includes and defines */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <stddef.h>
#include <time.h>

#define X_SIZE 512
#define Y_SIZE 512
#define MAX_ITERATIONS 100
#define TOLERANCE .000001

/* main function */
int main(int argc, char** argv)
{

    /* Variable declarations */

    FILE *fp_in_a,*fp_in_b ;
    FILE *fp_out;
    float a[X_SIZE * Y_SIZE];
    float b[Y_SIZE];
    float x_new[Y_SIZE];
    float x_old[Y_SIZE];
    int i,j, counter;
    double alpha, x_norm, x_norm_old, x_norm_delta;

    /* Variabel initializations */
    counter = 0;
    x_norm_old=500;
    x_norm_delta=500;
    time_t t0, t1; /* Wall clock variables */
    clock_t c0, c1; /* CPU time variables */

    /* Read in data file that contains matrix A */
    fp_in_a=fopen("matrix_a.txt","r");
    if (fp_in_a==NULL) return 0;
    /* read data */
    for(i=0;i<X_SIZE * Y_SIZE;i++) {
        fscanf(fp_in_a,"%f", &a[i]);
    }
}

```

```

}
fclose(fp_in_a);

/* Read in data file that contains array b */
fp_in_b=fopen("array_b.txt","r");
if (fp_in_b==NULL) return 0;
/* read data */
for(i=0;i<Y_SIZE;i++) {
    fscanf(fp_in_b,"%f",&b[i]);
}
fclose(fp_in_b);

/* start timing process */
c0 = clock();
t0 = time(NULL);

/* Jacobi Iteration */
/* Initialize x_old array */
/* That is make first guess all zeroes */
for(i=0;i<Y_SIZE;i++) {
    x_old[i]=0;
}

/* Loop until a convergene/max iter is reached */
while ((x_norm_delta > TOLERANCE) && (counter < MAX_INTERATIONS))
{
#pragma omp parallel for private(alpha,i, j,) shared(x_new)
    for(i=0;i<X_SIZE;i++) {
        alpha = 0;
        for(j=0;j<Y_SIZE;j++) {
            if (j != i) {
                alpha = alpha + (a[(i*X_SIZE)+ j] *
x_new[j]);
            }
        }
        while(0) {};
        x_new[i]= (b[i]-alpha) / (a[(i*X_SIZE) + i]);
    }

    /* Check convergence */
    x_norm=0;
    for(i=0;i<Y_SIZE;i++) {
        x_norm= x_norm + (x_new[i]* x_new[i]);
    }
    x_norm= sqrt(x_norm);
    /* Move new into old */
    for(i=0;i<Y_SIZE;i++) {
        x_old[i]=x_new[i];;
    }
    counter++;
    /* configure epsilon for stopping condition */
    x_norm_delta= fabs(x_norm_old-x_norm);
    x_norm_old=x_norm;
} /* end while loop */

```

```

/* Complete Timing Process and Print out results */
c1 = clock();
t1 = time(NULL);
printf(" \n===== \n");
printf("Timing for Asynch Openmp Jacobi Iteration :\n");
printf (" Elapsed Wall Clock Time: %ld\n", (long) (t1 - t0));
printf(" CPU time:%.5f\n", (float) (c1 - c0)/CLOCKS_PER_SEC);
printf(" Number of Loops Used %d\n", counter);
printf(" Final Convergence Test, 2_norm of X= %f", x_norm);
printf(" \n===== \n");

/* Write out array for display purposes */
fp_out=fopen("x_array_async_openmp.txt","w");
if (fp_out==NULL) return 0;
for(i=0;i<Y_SIZE;i++) {
    fprintf(fp_out, " %.5f \n",x_old[i]);
}
fclose(fp_out);

return 0;
}

```