Explanation of C Programs

This report provides a detailed explanation of the given C programs. Each section explains the objective, working logic, and important code snippets to help understand how the program functions internally.

DAXPY Program (daxpy.c)

Objective: To perform the DAXPY operation used in linear algebra.

Explanation: This program applies the mathematical operation Y = aX + Y, where 'a' is a scalar and X and Y are vectors. It uses a loop to process each element individually. Such operations are commonly used in scientific and parallel computing applications.

Sample Code Snippet:

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

#define N (1<<16)

int main() {
    double *X = (double*)malloc(N * sizeof(double));
    double *Y = (double*)malloc(N * sizeof(double));
    double a = 2.5;

    // Initialize vectors
    for (int i = 0; i < N; i++) {
        X[i] = 1.0;
        Y[i] = 2.0;
```

1D Matrix Program (matrix_1d.c)

Objective: To demonstrate matrix handling using a one-dimensional array.

Explanation: The program stores matrix elements in a linear array and accesses them using index calculations. This approach is memory-efficient and helps in understanding how matrices are actually stored in memory.

Sample Code Snippet:

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

#define N 1000

int main() {
    static double A[N][N], B[N][N], C[N][N];

    // Initialize matrices
    #pragma omp parallel for
```

```
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            A[i][j] = B[i][j] = 1.0;

    double start = omp_get_wtime();
```

2D Matrix Program (matrix_2d.c)

Objective: To perform matrix operations using a two-dimensional array.

Explanation: The program uses nested loops to traverse rows and columns. This representation is more readable and closely matches the mathematical representation of a matrix.

Sample Code Snippet:

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

#define N 1000

int main() {
    static double A[N][N], B[N][N], C[N][N];

    // Initialize matrices
    #pragma omp parallel for collapse(2)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            A[i][j] = B[i][j] = 1.0;

    double start = omp_get_wtime();
```

PI Calculation Program (pi.c)

Objective: To compute the value of PI using an approximation method.

Explanation: The program uses an iterative mathematical formula to approximate PI. Increasing the number of iterations improves accuracy, demonstrating the concept of numerical approximation.

Sample Code Snippet:

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

static long num_steps = 100000000;

int main() {
    double step = 1.0 / (double) num_steps;
    double sum = 0.0;

    double start = omp_get_wtime();

    // Parallel computation using reduction
    #pragma omp parallel for reduction(+:sum)
    for (long i = 0; i < num_steps; i++) {
        double x = (i + 0.5) * step;
        }
```