PYTHON CODE

```
import numpy as np
import time

matrix1 = np.random.rand(1000, 1000)
matrix2 = np.random.rand(1000, 1000)

start_time = time.time()

result_matrix = np.dot(matrix1, matrix2)

end_time = time.time()

runtime = end_time - start_time

print("1:")
print(matrix1)
print("\n2:")
print(matrix2)
print("\nproduct:")
print(result_matrix)
print(f"\nRuntime: {runtime} seconds”)
```

C++ CODE

```
#include <iostream>
#include <ctime>
#include <cstdlib>

int main() {
    const int size = 1000;

    std::srand(static_cast<unsigned>(std::time(nullptr)));

    double matrix1[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix1[i][j] = static_cast<double>(std::rand()) / RAND_MAX;
        }
    }

    double matrix2[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix2[i][j] = static_cast<double>(std::rand()) / RAND_MAX;
        }
    }

    double result[size][size] = {0};

    clock_t start_time = clock();
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            for (int k = 0; k < size; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }

    clock_t end_time = clock();
    double runtime = static_cast<double>(end_time - start_time) / CLOCKS_PER_SEC;

    std::cout << "Runtime: " << runtime << " seconds" << std::endl;

    return 0;
}
```

R LANGUAGE

```
size <- 10

set.seed(Sys.time())
matrix1 <- matrix(runif(size^2), nrow = size)
matrix2 <- matrix(runif(size^2), nrow = size)

result_matrix <- matrix(0, nrow = size, ncol = size)

start_time <- Sys.time()
for (i in 1:size) {
  for (j in 1:size) {
    for (k in 1:size) {
      result_matrix[i, j] <- result_matrix[i, j] + matrix1[i, k] * matrix2[k, j]
    }
  }
}

end_time <- Sys.time()

runtime <- end_time - start_time

cat("Runtime: ", runtime, " seconds\n”)
```

C LANGUAGE

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

int main() {
    const int size = 10;

    srand((unsigned)time(NULL));

    double matrix1[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix1[i][j] = (double)rand() / RAND_MAX;
        }
    }

    double matrix2[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix2[i][j] = (double)rand() / RAND_MAX;
        }
    }
    double result[size][size];

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[i][j] = 0;
        }
    }

    clock_t start_time = clock();

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            for (int k = 0; k < size; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
    clock_t end_time = clock();
    double runtime = (double)(end_time - start_time) / CLOCKS_PER_SEC;
    printf("Runtime: %lf seconds\n", runtime);

    return 0;
}
```

MATLAB

```
par (mfrow-c(1,3))
＜-C〇
for (i in 1:9000){
s [1]-mean (sample (dataSwall. Thickness, 10, replace=TRUE))
hist (5)
abline (v=z, lty=1)
SS ＜-CO
• for (1 in 1:9000){
ss [i]-mean (sample (dataswall. Thickness, 50, replace=TRUE))
• 3
hist (55)
abline(v=z, 1ty=1)
SW ＜-CO
- for (i in 1:9000){
sw[i]-mean (sample (dataSWall. Thickness, 500, replace-TRUE))
+ }
hist (sw)
abline (v=z, lty=1)
```