

```
1  #include <iostream>
2  #include <vector>
3  #include <cstdlib>
4  #include <ctime>
5  #include <chrono>
6
7  using namespace std;
8  using namespace chrono;
9
10 const int M = 500, N = 500, X = 500;
11
12 vector<vector<int>> gerarMatriz(int linha, int coluna) {
13     vector<vector<int>> mat(linha, vector<int>(coluna));
14     for (int i = 0; i < linha; i++) {
15         for (int j = 0; j < coluna; j++) {
16             mat[i][j] = rand() % 10;
17         }
18     }
19     return mat;
20 }
21
22 vector<vector<int>> multiplicaLinha(const vector<vector<int>> &A, const
vector<vector<int>> &B) {
23     vector<vector<int>> C(M, vector<int>(X,0));
24     for (int i = 0; i < M; i++) {
25         for (int j = 0; j < N; j++) {
26             for (int k = 0; k < X; k++) {
27                 C[i][k] += A[i][j] * B[j][k];
28             }
29         }
30     }
31     return C;
32 }
33
34 int main() {
35     srand(time(nullptr));
36
37     vector<vector<int>> A = gerarMatriz(M, N);
38     vector<vector<int>> B = gerarMatriz(N, X);
```

```
38     vector<vector<int>> B = gerarMatriz(N, X);
39
40     auto start = high_resolution_clock::now();
41     vector<vector<int>> C = multiplicaLinha(A, B);
42     auto end = high_resolution_clock::now();
43     cout << "Tempo (linha-maior): " << duration_cast<milliseconds>(end - start).count()
44     << " ms" << endl;
45     return 0;
46 }
```

código fonte da multiplicação de matrizes por linha

```

Packager
> make -s
  ./main
Tempo (linha-maior): 43 ms

Packager
  make -s

  ./main
Tempo (linha-maior): 39 ms

Packager
  make -s

  ./main
Tempo (linha-maior): 58 ms

Packager
  make -s

  ./main
Tempo (linha-maior): 48 ms

Packager
  make -s

  ./main
Tempo (linha-maior): 50 ms

```

Testes do tempo de execução (Tempo médio de 49,3 ms)

```

1  #include <iostream>
2  #include <vector>
3  #include <cstdlib>
4  #include <ctime>
5  #include <chrono>
6
7  using namespace std;
8  using namespace chrono;
9
10 const int M = 500, N = 500, X = 500;
11
12 vector<vector<int>> gerarMatrix(int linha, int coluna) {
13     vector<vector<int>> mat(linha, vector<int>(coluna));
14     for (int i = 0; i < linha; ++i) {
15         for (int j = 0; j < coluna; ++j) {
16             mat[i][j] = rand() % 10;
17         }
18     }
19     return mat;
20 }
21
22 vector<vector<int>> multiplicaColuna(const vector<vector<int>> &A, const vector<vector<int>> &B) {
23     vector<vector<int>> C(M, vector<int>(X, 0));
24     for (int j = 0; j < X; ++j) {
25         for (int k = 0; k < N; ++k) {
26             for (int i = 0; i < M; ++i) {
27                 C[i][j] += A[i][k] * B[k][j];
28             }
29         }
30     }
31     return C;
32 }
33
34 int main() {
35     srand(time(nullptr));
36
37     vector<vector<int>> A = gerarMatrix(M, N);
38     vector<vector<int>> B = gerarMatrix(N, X);
39
40     auto start = high_resolution_clock::now();
41     vector<vector<int>> C = multiplicaColuna(A, B);
42     auto end = high_resolution_clock::now();
43     cout << "Tempo (coluna-maior): " << duration_cast<milliseconds>(end - start).count() << " ms" << endl;
44
45     return 0;
46 }

```

Código fonte da multiplicação de coluna.

```
Tempo (coluna-maior): 430 ms

Packager
make -s

./main

Tempo (coluna-maior): 248 ms

Packager
make -s

./main

Tempo (coluna-maior): 255 ms

Packager
make -s

./main

Tempo (coluna-maior): 240 ms
```

Teste de tempo de execução do código (Tempo médio de 300 ms).

```

1  #include <iostream>
2  #include <vector>
3  #include <cstdlib>
4  #include <ctime>
5  #include <chrono>
6
7  using namespace std;
8  using namespace chrono;
9
10 const int M = 500, N = 500, X = 500;
11 const int BLOCK_SIZE = 32; // Ajustável para otimização do cache
12
13 vector<vector<int>> gerarMatrix(int linha, int coluna) {
14     vector<vector<int>> mat(linha, vector<int>(coluna));
15     for (int i = 0; i < linha; ++i) {
16         for (int j = 0; j < coluna; ++j) {
17             mat[i][j] = rand() % 10;
18         }
19     }
20     return mat;
21 }
22
23 vector<vector<int>> multiplicaBlocagem(const vector<vector<int>> &A, const vector<vector<int>> &B) {
24     vector<vector<int>> C(M, vector<int>(X, 0));
25     for (int ii = 0; ii < M; ii += BLOCK_SIZE) {
26         for (int jj = 0; jj < X; jj += BLOCK_SIZE) {
27             for (int kk = 0; kk < N; kk += BLOCK_SIZE) {
28                 for (int i = ii; i < min(ii + BLOCK_SIZE, M); ++i) {
29                     for (int k = kk; k < min(kk + BLOCK_SIZE, N); ++k) {
30                         for (int j = jj; j < min(jj + BLOCK_SIZE, X); ++j) {
31                             C[i][j] += A[i][k] * B[k][j];
32                         }
33                     }
34                 }
35             }
36         }
37     }
38     return C;
39 }
40
41 int main() {
42     srand(time(nullptr));
43
44     vector<vector<int>> A = gerarMatrix(M, N);
45     vector<vector<int>> B = gerarMatrix(N, X);
46
47     auto start = high_resolution_clock::now();
48     vector<vector<int>> C = multiplicaBlocagem(A, B);
49     auto end = high_resolution_clock::now();
50     cout << "Tempo (blocagem): " << duration_cast<milliseconds>(end - start).count() << " ms" << endl;
51
52     return 0;
53 }

```

Código fonte de blocagem

```

  ▾ ./main
Tempo (blocagem): 64 ms

  ▾ Packager
  ▾ make -s

  ▾ ./main
Tempo (blocagem): 37 ms

  ▾ Packager
  ▾ make -s

  ▾ ./main
Tempo (blocagem): 38 ms

  ▾ Packager
  ▾ make -s

  ▾ ./main
Tempo (blocagem): 44 ms

  ▾ Packager
  ▾ make -s

  ▾ ./main
Tempo (blocagem): 46 ms

```

Teste de tempo do programa (Tempo Médio 47,1 ms)

OBS: GPT frequentemente utilizado mais para eu saber como programar em C++