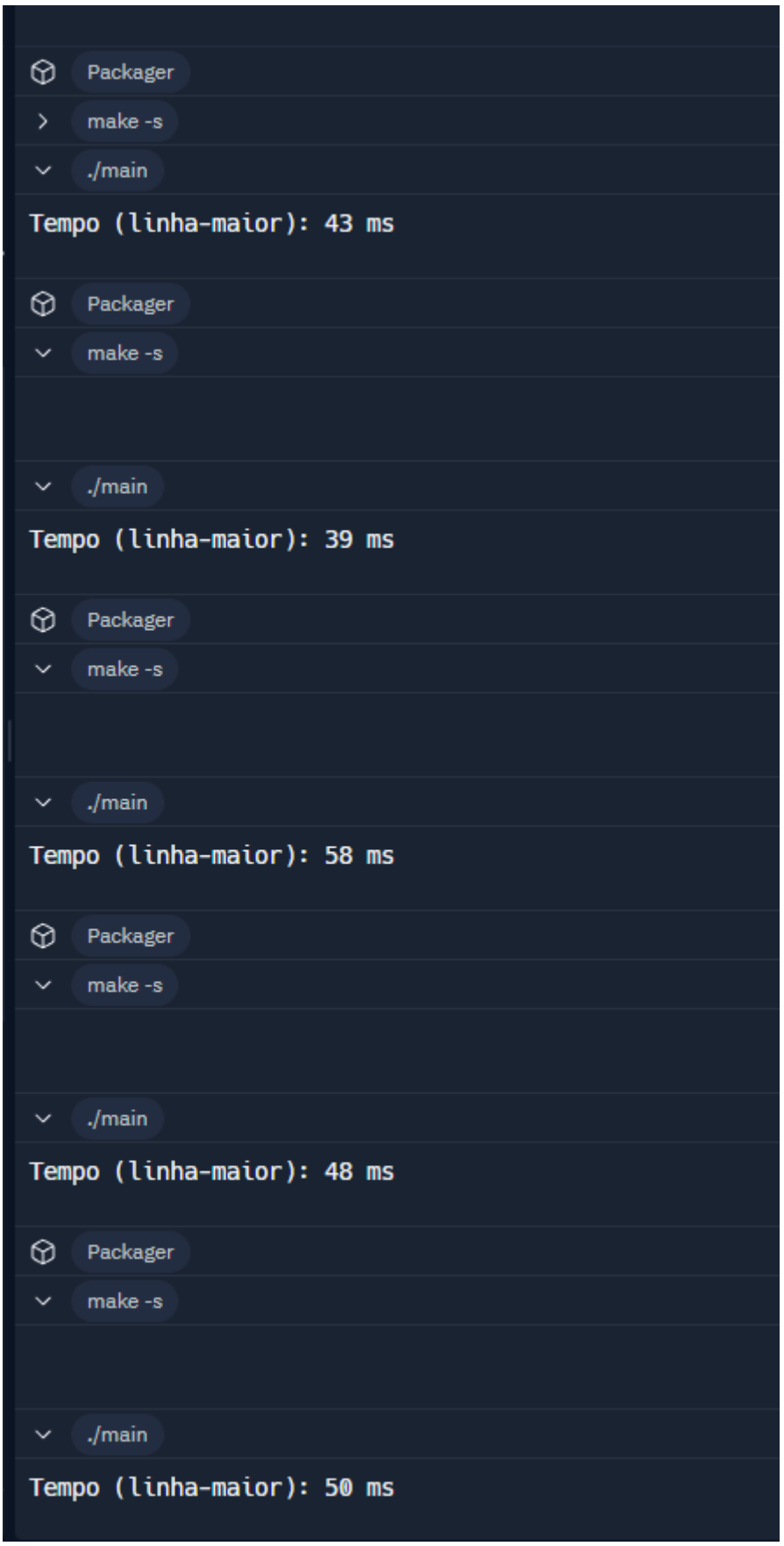
Nome: Alexandre Luppi Severo e Silva RA: 10419724

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;
using namespace chrono;

const int M = 500, N = 500, X = 500;

vector<vector<int>> gerarMatriz(int linha, int coluna) {
  vector<vector<int>> mat(linha, vector<int>(coluna));
  for (int i = 0; i < linha; i++) {
    for (int j = 0; j < coluna; j++) {
      mat[i][j] = rand() % 10;
    }
  }
  return mat;
}

vector<vector<int>> multiplicaLinha(const vector<vector<int>> &A, const
vector<vector<int>> &B) {
  vector<vector<int>> C(M, vector<int>(X,0));
  for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
      for (int k = 0; k < X; k++) {
        C[i][k] += A[i][j] * B[j][k];
      }
    }
  }
  return C;
}

int main() {
  srand(time(nullptr));

  vector<vector<int>> A = gerarMatriz(M, N);
  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()
    << " ms" << endl;
44
45    return 0;
46  }
```

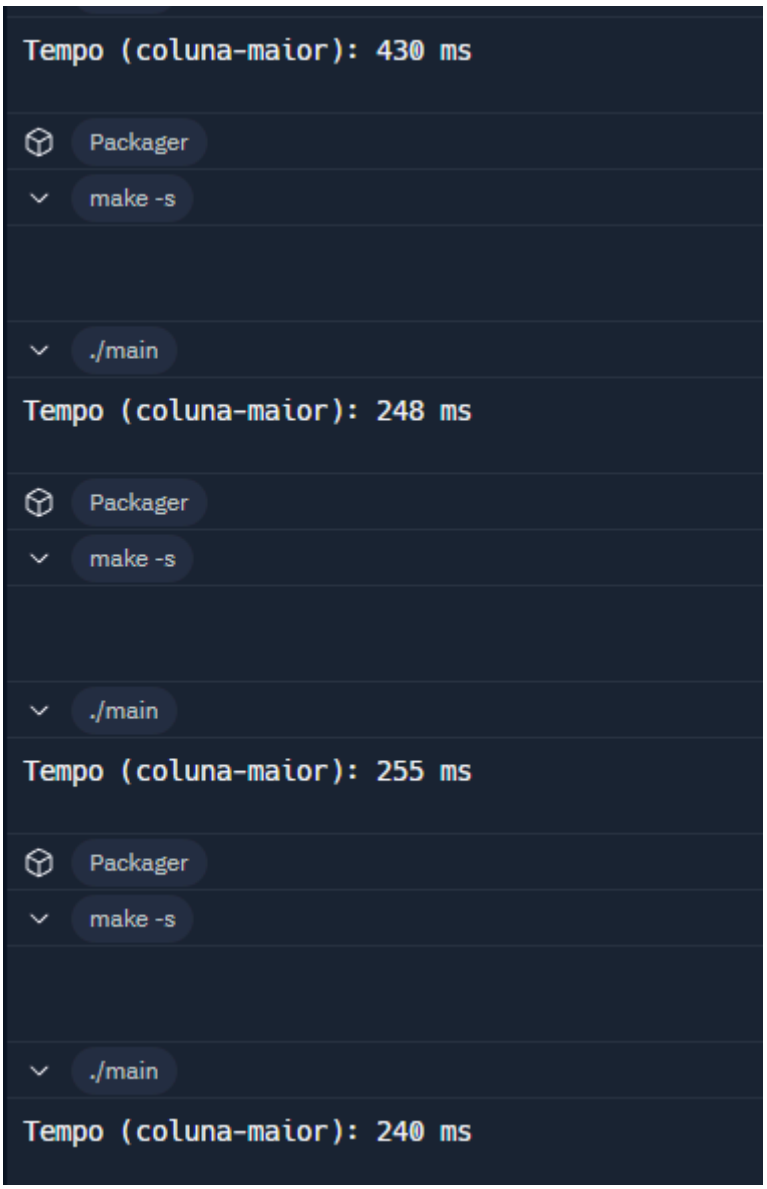
código fonte da multiplicação de matrizes por linha



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

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;
using namespace chrono;

const int M = 500, N = 500, X = 500;

vector<vector<int>> gerarMatrix(int linha, int coluna) {
    vector<vector<int>> mat(linha, vector<int>(coluna));
    for (int i = 0; i < linha; ++i) {
        for (int j = 0; j < coluna; ++j) {
            mat[i][j] = rand() % 10;
        }
    }
    return mat;
}

vector<vector<int>> multiplicaColuna(const vector<vector<int>> &A, const vector<vector<int>> &B) {
    vector<vector<int>> C(M, vector<int>(X, 0));
    for (int j = 0; j < X; ++j) {
        for (int k = 0; k < N; ++k) {
            for (int i = 0; i < M; ++i) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}

int main() {
    srand(time(nullptr));

    vector<vector<int>> A = gerarMatrix(M, N);
    vector<vector<int>> B = gerarMatrix(N, X);

    auto start = high_resolution_clock::now();
    vector<vector<int>> C = multiplicaColuna(A, B);
    auto end = high_resolution_clock::now();
    cout << "Tempo (coluna-maior): " << duration_cast<milliseconds>(end - start).count() << " ms" << endl;

    return 0;
}
```

Código fonte da multiplicação de coluna.

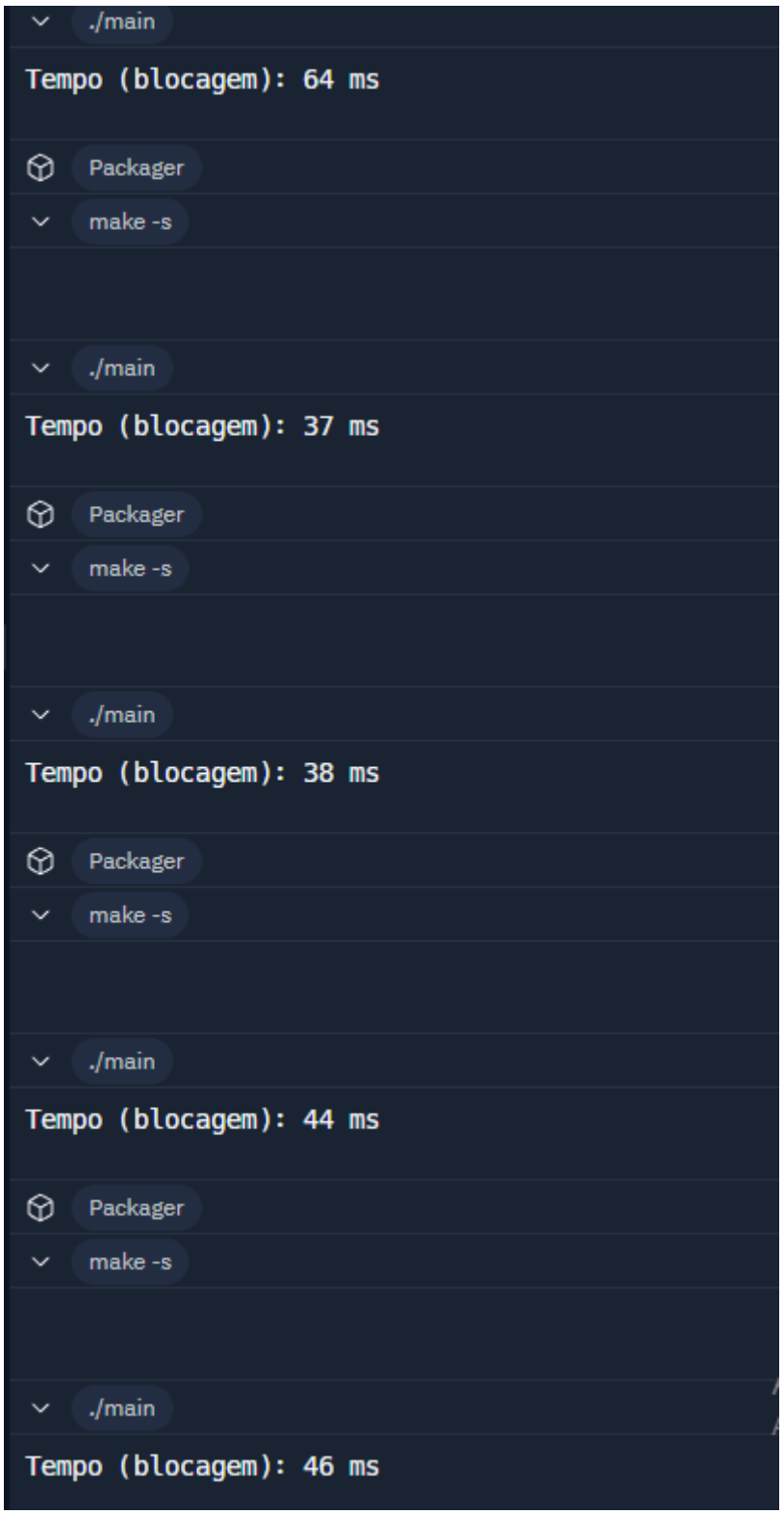


Teste de tempo de execução do código (Tempo médio de 300 ms).

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;
using namespace chrono;

const int M = 500, N = 500, X = 500;
const int BLOCK_SIZE = 32; // Ajustável para otimização do cache

vector<vector<int>> gerarMatrix(int linha, int coluna) {
    vector<vector<int>> mat(linha, vector<int>(coluna));
    for (int i = 0; i < linha; ++i) {
        for (int j = 0; j < coluna; ++j) {
            mat[i][j] = rand() % 10;
        }
    }
    return mat;
}

vector<vector<int>> multiplicaBlocagem(const vector<vector<int>> &A, const vector<vector<int>> &B) {
    vector<vector<int>> C(M, vector<int>(X, 0));
    for (int ii = 0; ii < M; ii += BLOCK_SIZE) {
        for (int jj = 0; jj < X; jj += BLOCK_SIZE) {
            for (int kk = 0; kk < N; kk += BLOCK_SIZE) {
                for (int i = ii; i < min(ii + BLOCK_SIZE, M); ++i) {
                    for (int k = kk; k < min(kk + BLOCK_SIZE, N); ++k) {
                        for (int j = jj; j < min(jj + BLOCK_SIZE, X); ++j) {
                            C[i][j] += A[i][k] * B[k][j];
                        }
                    }
                }
            }
        }
    }
    return C;
}
```

```cpp
int main() {
    srand(time(nullptr));

    vector<vector<int>> A = gerarMatrix(M, N);
    vector<vector<int>> B = gerarMatrix(N, X);

    auto start = high_resolution_clock::now();
    vector<vector<int>> C = multiplicaBlocagem(A, B);
    auto end = high_resolution_clock::now();
    cout << "Tempo (blocagem): " << duration_cast<milliseconds>(end - start).count() << " ms" << endl;

    return 0;
}
```

Código fonte de blocagem



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

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