**Лабораторная работа №1**

**Умножение вектора на матрицу**

**ПОСТАНОВКА ЗАДАЧИ**

Для больших матриц провести операцию умножения матрицы на вектор.

**КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**



Всем процессам передаётся вектор B

Каждому процессу передаётся своя часть A

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include "mpi.h"

#include <iostream>

#include <fstream>

#include <sys/timeb.h>

double realtime() {

struct \_timeb tp;

\_ftime(&tp);

return((double)(tp.time) \* 1000 + (double)(tp.millitm));

}

int main(int argc, char\* argv[])

{

int procsRank, procsCount;

int i, j, n = 1000, local\_n;

double\* A, \* b, \* C, \* a, \* c;

double t1, t2, t3, t4;

t1 = realtime();

MPI\_Init(&argc, &argv);

t2 = realtime();

MPI\_Comm\_size(MPI\_COMM\_WORLD, &procsCount);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &procsRank);, записывает в procs\_rank

local\_n = n / procsCount;

A = (double\*)malloc((local\_n \* n) \* sizeof(double));

b = (double\*)malloc(n \* sizeof(double));

C = (double\*)malloc(local\_n \* sizeof(double));

c = (double\*)malloc(n \* sizeof(double));

a = (double\*)malloc((n \* n) \* sizeof(double));

std::ifstream fin;

//чтение из файла

if (procsRank == 0)

{

fin.open("input.txt");

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

{

fin >> a[i \* n + j];

}

for (i = 0; i < n; i++) {

fin >> b[i];

}

fin.close();

}

double time\_start = MPI\_Wtime();

MPI\_Bcast(b, n, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

MPI\_Scatter(a, n \* local\_n, MPI\_DOUBLE, A, n \* local\_n, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

t3 = MPI\_Wtime(); //время старта вычислений на одном процессе

for (i = 0; i < local\_n; i++)

{

C[i] = 0;

for (j = 0; j < n; j++)

C[i] += b[j]\*A[i \* n + j] ;

}

t4 = MPI\_Wtime();

MPI\_Gather(C, local\_n, MPI\_DOUBLE, c, local\_n, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD); //собирает данные со всех процессов, ответ лежит в векторе C

double time\_end = MPI\_Wtime();

//вывод результатов

if (procsRank == 0)

{

std::ofstream fout("out.txt");

for (auto i = 0; i < n; i++) {

for (auto j = 0; j < n; j++)

fout << C[i \* n + j] << " ";

fout << "\n";

}

std::cout << "init: " << t2 - t1 << "\n";

std::cout << t4 - t3 <<"\n";

std::cout << time\_end - time\_start<<"\n";

}

MPI\_Finalize(); //завершение работы MPI

return 0;

}