**Лабораторная работа №4**

**Производные типы данных в MPI**

**Цель:** изучить основные принципы использования производных типов данных в технологии MPI на примере использования в рамках языка С++.

**Лабораторные задания** (№ варианта = 2)

**Задание.** Реализуйте на основе технологии MPI многопроцессную программу в соответствии с вариантом задания**.** Найдите возможные способы проверки корректности работы программы. Оцените их возможные достоинства и недостатки. Проверьте корректность работы программы при помощи одного из них. **Результаты занесите в отчет.**

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| **2** | Реализуйте тип «полином».  Напишите программу, которая осуществляет умножение A полиномов заданной степени. |

#include <stdio.h>

#include "mpi.h"

#include <iostream>

#include <time.h>

#include <locale>

#include <vector>

using namespace std;

int generate\_int(int min\_v, int max\_v) {

return min\_v + std::rand() % (max\_v - min\_v);

}

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

{

int ProcNum, ProcRank;

int RecvRank = 0;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

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

MPI\_Datatype polynom\_type;

MPI\_Status Status;

int size\_p = 5;

int tag1 = 1;

int tag2 = 2;

int tag3 = 3;

int tag4 = 4;

int tag5 = 5;

int tag6 = 6;

int tag7 = 7;

int tag8 = 8;

int tag11 = 11;

int tag21 = 21;

int tag31 = 31;

int tag41 = 41;

srand(time(0));

MPI\_Type\_contiguous(8\*size\_p-7, MPI\_INT, &polynom\_type);

MPI\_Type\_commit(&polynom\_type);

if (ProcRank == 0) {

int\* pol1 = new int[8\*size\_p-7];

int\* pol2 = new int[8 \* size\_p - 7];

int\* pol3 = new int[8 \* size\_p - 7];

int\* pol4 = new int[8 \* size\_p - 7];

int\* pol5 = new int[8 \* size\_p - 7];

int\* pol6 = new int[8 \* size\_p - 7];

int\* pol7 = new int[8 \* size\_p - 7];

int\* pol8 = new int[8 \* size\_p - 7];

int\* pol12 = new int[8 \* size\_p - 7];

for (int i = 0; i < 8 \* size\_p - 7; i++) {

pol1[i] = 0;

pol2[i] = 0;

pol3[i] = 0;

pol4[i] = 0;

pol5[i] = 0;

pol6[i] = 0;

pol7[i] = 0;

pol8[i] = 0;

pol12[i] = 0;

}

for (int i = 0; i < size\_p; i++) {

pol1[i] = generate\_int(1, 3);

pol2[i] = generate\_int(1, 3);

pol3[i] = generate\_int(1, 3);

pol4[i] = generate\_int(1, 3);

pol5[i] = generate\_int(1, 3);

pol6[i] = generate\_int(1, 3);

pol7[i] = generate\_int(1, 3);

pol8[i] = generate\_int(1, 3);

}

MPI\_Send(&pol1[0], 1, polynom\_type, 1, tag1, MPI\_COMM\_WORLD);

MPI\_Send(&pol2[0], 1, polynom\_type, 1, tag2, MPI\_COMM\_WORLD);

MPI\_Send(&pol3[0], 1, polynom\_type, 2, tag3, MPI\_COMM\_WORLD);

MPI\_Send(&pol4[0], 1, polynom\_type, 2, tag4, MPI\_COMM\_WORLD);

MPI\_Send(&pol5[0], 1, polynom\_type, 3, tag5, MPI\_COMM\_WORLD);

MPI\_Send(&pol6[0], 1, polynom\_type, 3, tag6, MPI\_COMM\_WORLD);

MPI\_Send(&pol7[0], 1, polynom\_type, 4, tag7, MPI\_COMM\_WORLD);

MPI\_Send(&pol8[0], 1, polynom\_type, 4, tag8, MPI\_COMM\_WORLD);

MPI\_Recv(&pol12[0], 1, polynom\_type, 1, tag11, MPI\_COMM\_WORLD, &Status);

printf("TAG 0, 11: ");

for (int j = 0; j < 8 \* size\_p - 7; j++) {

printf("%d ", pol12[j]);

}

printf("\n");

}

else if (ProcRank == 1) {

printf("PROC: 1\_\n");

int\* pol1 = new int[8 \* size\_p - 7];

int\* pol2 = new int[8 \* size\_p - 7];

int\* pol3 = new int[8 \* size\_p - 7];

int\* pol4 = new int[8 \* size\_p - 7];

int\* pol12 = new int[8 \* size\_p - 7];

int\* pol13 = new int[8 \* size\_p - 7];

int\* polres = new int[8 \* size\_p - 7];

MPI\_Recv(&pol1[0], 1, polynom\_type, 0, tag1, MPI\_COMM\_WORLD, &Status);

MPI\_Recv(&pol2[0], 1, polynom\_type, 0, tag2, MPI\_COMM\_WORLD, &Status);

for (int j = 0; j < 8 \* size\_p - 7; j++) {

pol12[j] = 0;

pol13[j] = 0;

polres[j] = 0;

}

for (int i = 0; i < size\_p; i++) {

for (int j = 0; j < size\_p; j++) {

pol12[i+j] += pol1[i] \* pol2[j];

}

}

printf("TAG 1,2: ");

for (int j = 0; j < 2 \* size\_p - 1; j++) {

printf("%d ", pol12[j]);

}

printf("\n");

MPI\_Recv(&pol3[0], 1, polynom\_type, 2, tag21, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < 2\*size\_p-1; i++) {

for (int j = 0; j < 2\*size\_p-1; j++) {

pol13[i+j] += pol3[i] \* pol12[j];

}

}

printf("TAG 1,21: ");

for (int j = 0; j < 4 \* size\_p - 3; j++) {

printf("%d ", pol13[j]);

}

printf("\n");

MPI\_Recv(&pol4[0], 1, polynom\_type, 3, tag31, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < 4\*size\_p-3; i++) {

for (int j = 0; j < 4\*size\_p-3; j++) {

polres[i+j] += pol4[i] \* pol12[j];

}

}

printf("TAG 1,31: ");

for (int j = 0; j < 8 \* size\_p - 7; j++) {

printf("%d ", polres[j]);

}

printf("\n");

MPI\_Send(&polres[0], 1, polynom\_type, 0, tag11, MPI\_COMM\_WORLD);

delete[] pol1;

delete[] pol2;

delete[] pol3;

delete[] pol4;

delete[] pol12;

}

else if (ProcRank == 2) {

printf("PROC: 2\_\n");

int\* pol1 = new int[8 \* size\_p - 7];

int\* pol2 = new int[8 \* size\_p - 7];

int\* pol12 = new int[8 \* size\_p - 7];

for (int i = 0; i < 2\*size\_p-1; i++) {

pol12[i] = 0;

}

MPI\_Recv(&pol1[0], 1, polynom\_type, 0, tag3, MPI\_COMM\_WORLD, &Status);

MPI\_Recv(&pol2[0], 1, polynom\_type, 0, tag4, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < size\_p; i++) {

for (int j = 0; j < size\_p; j++) {

pol12[i+j] += pol1[i] \* pol2[j];

}

}

printf("TAG 3,4: ");

for (int j = 0; j < 2 \* size\_p - 1; j++) {

printf("%d ", pol12[j]);

}

printf("\n");

MPI\_Send(&pol12[0], 1, polynom\_type, 1, tag21, MPI\_COMM\_WORLD);

delete[] pol1;

delete[] pol2;

delete[] pol12;

}

else if (ProcRank == 3) {

printf("PROC: 3\_\n");

int\* pol1 = new int[8 \* size\_p - 7];

int\* pol2 = new int[8 \* size\_p - 7];

int\* pol3 = new int[8 \* size\_p - 7];

int\* pol12 = new int[8 \* size\_p - 7];

int\* polres = new int[8 \* size\_p - 7];

for (int i = 0; i < 8 \* size\_p - 7; i++) {

pol12[i] = 0;

polres[i] = 0;

}

MPI\_Recv(&pol1[0], 1, polynom\_type, 0, tag5, MPI\_COMM\_WORLD, &Status);

MPI\_Recv(&pol2[0], 1, polynom\_type, 0, tag6, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < size\_p; i++) {

for (int j = 0; j < size\_p; j++) {

pol12[i+j] += pol1[i] \* pol2[j];

}

}

printf("TAG 5,6: ");

for (int j = 0; j < 2 \* size\_p - 1; j++) {

printf("%d ", pol12[j]);

}

printf("\n");

MPI\_Recv(&pol3[0], 1, polynom\_type, 4, tag41, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < 2\*size\_p-1; i++) {

for (int j = 0; j < 2\*size\_p-1; j++) {

polres[i+j] += pol3[i] \* pol12[j];

}

}

printf("TAG 3, 41: ");

for (int j = 0; j < 4 \* size\_p - 3; j++) {

printf("%d ", polres[j]);

}

printf("\n");

MPI\_Send(&polres[0], 1, polynom\_type, 1, tag31, MPI\_COMM\_WORLD);

delete[] pol1;

delete[] pol2;

delete[] pol3;

delete[] pol12;

delete[] polres;

}

else if (ProcRank == 4) {

printf("PROC: 4\_\n");

int\* pol1 = new int[8\*size\_p-7];

int\* pol2 = new int[8 \* size\_p - 7];

int\* pol12 = new int[8 \* size\_p - 7];

for (int i = 0; i < 8 \* size\_p - 7; i++) {

pol12[i] = 0;

}

MPI\_Recv(&pol1[0], 1, polynom\_type, 0, tag7, MPI\_COMM\_WORLD, &Status);

MPI\_Recv(&pol2[0], 1, polynom\_type, 0, tag8, MPI\_COMM\_WORLD, &Status);

for (int i = 0; i < size\_p; i++) {

for (int j = 0; j < size\_p; j++) {

pol12[i+j] += pol1[i] \* pol2[j];

}

}

printf("TAG 7,8: ");

for (int j = 0; j < 2 \* size\_p - 1; j++) {

printf("%d ", pol12[j]);

}

printf("\n");

MPI\_Send(&pol12[0], 1, polynom\_type, 3, tag41, MPI\_COMM\_WORLD);

delete[] pol1;

delete[] pol2;

delete[] pol12;

}

MPI\_Type\_free(&polynom\_type);

MPI\_Finalize();

return 0;

}