Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

# ПРОИЗВОДНЫЕ ТИПЫ В MPI

Отчёт о лабораторной работе № 3 по дисциплине «Параллельное программирование»

Студент гр. 431-3

\_\_\_\_\_\_\_ Д.П. Андреев

«\_\_» \_\_\_\_\_\_\_\_\_ 2024

Проверил

Доцент каф. АСУ, к.т.н

\_\_\_\_\_\_\_ С.М. Алфёров

«\_\_» \_\_\_\_\_\_\_\_\_ 2024

Томск 2024

**1 Цель лабораторной работы**

Цель: освоить применение функций MPI для конструирования производных типов данных и передачи на их основе выбранных частей двумерного массива.

## 2 Задание

Задание на лабораторную работу (вариант 9): сконструировать подходящий производный тип, послать матрицу по рангу, взять левый верхний треугольник матрицы.

### 3 Использованные MPI функции

В этом коде используем индексный производный тип, т.к. это будет лучше всего для нелинейного размера нашей матрицы. Используем новые функции MPI:

1. MPI\_Type\_indexed(N, block\_lengths.data(), displacements.data(), MPI\_INT, &triangle\_type);

Описание: Функция для создания производного типа данных с использованием индексного конструктора.

**Аргументы**:

N: Количество блоков, которые будут определены в производном типе данных. В данном случае, это количество строк верхнего треугольника матрицы.

block\_lengths.data(): Указатель на массив, определяющий количество элементов в каждом блоке. В контексте левого верхнего треугольника, количество элементов уменьшается от N до 1 по строкам.

displacements.data(): Указатель на массив, определяющий смещения (в байтах) для каждого блока относительно начала буфера. Это позволяет указать, где каждый блок начинается в памяти. Для левого верхнего треугольника, это смещения начала каждой строки в исходной матрице.

MPI\_INT: Тип базовых данных в каждом блоке. В данном случае это int, так как мы работаем с матрицей целых чисел.

&triangle\_type: Указатель на переменную типа MPI\_Datatype, которая будет заполнена информацией о созданном производном типе данных.

2. MPI\_Type\_commit(&triangle\_type);

**Описание**: Функция для регистрации производного типа данных, чтобы его можно было использовать в функциях передачи данных.

**Аргументы**:

&triangle\_type: Указатель на переменную типа MPI\_Datatype, которую мы зарегистрируем. После вызова этой функции производный тип данных становится доступным для использования в функциях обмена данными MPI (например, MPI\_Send и MPI\_Recv).

**4 Листинг программы**

Main.cpp:

#include <mpi.h>

#include <iostream>

#include <vector>

constexpr int N = 5; // Размер матрицы N x N

// Печать матрицы

void print\_matrix(const int\* matrix, int rows, int cols, bool trim = false) {

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

if ((j < rows-i) || !trim) {

std::cout << matrix[i \* cols + j] << " ";

} else {

std::cout << "0 ";

}

}

std::cout << std::endl;

}

}

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

MPI\_Init(&argc, &argv);

int rank;

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

int matrix[N \* N]; // Исходная матрица

int recv\_matrix[N \* N] = {0}; // Матрица для приема данных

int T=(N\*(N+1))/2;

int select\_vector[T]= {0};

if (rank == 0) {

// Заполняем исходную матрицу числами от 1 до N\*N

for (int i = 0; i < N \* N; ++i) {

matrix[i] = i + 1;

}

std::cout << "Процесс 0: Оригинальная матрица:" << std::endl;

print\_matrix(matrix, N, N);

}

// Определяем массивы смещений и количества элементов для индексного типа

std::vector<int> block\_lengths(N);

std::vector<int> displacements(N);

// Для левого верхнего треугольника

for (int i = 0; i < N; ++i) {

block\_lengths[i] = N - i; // Количество элементов в строке уменьшается

displacements[i] = i \* N; // Смещение начала строки

}

// Создание производного типа данных (индексного)

MPI\_Datatype triangle\_type;

MPI\_Type\_indexed(N, block\_lengths.data(), displacements.data(), MPI\_INT, &triangle\_type);

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

if (rank == 0) {

// Отправка данных во второй процесс (процесс 1)

MPI\_Send(matrix, 1, triangle\_type, 1, 0, MPI\_COMM\_WORLD);

} else if (rank == 1) {

// Прием данных в процесс 1

MPI\_Recv(recv\_matrix, 1, triangle\_type, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Send(recv\_matrix, N\*N, MPI\_INT, 2, 0, MPI\_COMM\_WORLD);

}

else if (rank ==2){

// Прием данных в процесс 2

MPI\_Recv(recv\_matrix, N\*N, MPI\_INT, 1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

std::cout << "Процесс 2: Левый треугольник матрицы:" << std::endl;

print\_matrix(recv\_matrix, N, N, true);

int itr=0;

int tr=N;

for (int i = 0; i < 5; ++i)

{

for (int j = 0; j <tr; ++j)

{

select\_vector[itr] = recv\_matrix[i \* 5 + j];

itr++;

}

tr--;

}

std::cout << std::endl;

}

if (rank == 2) {

// Отправка данных с второго в нулевой процесс

MPI\_Send(select\_vector, T, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

} else if (rank == 0) {

// Прием данных в процесе 0

MPI\_Recv(recv\_matrix, 1, triangle\_type, 2, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

std::cout << "Процесс 0: Другая матрица:" << std::endl;

print\_matrix(recv\_matrix, N, N, true);

}

// Освобождение производного типа данных

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

MPI\_Finalize();

return 0;

}

### 5 Примеры работы программы

Разберём работу программы. Пример работы изображён на рисунке 5.1.

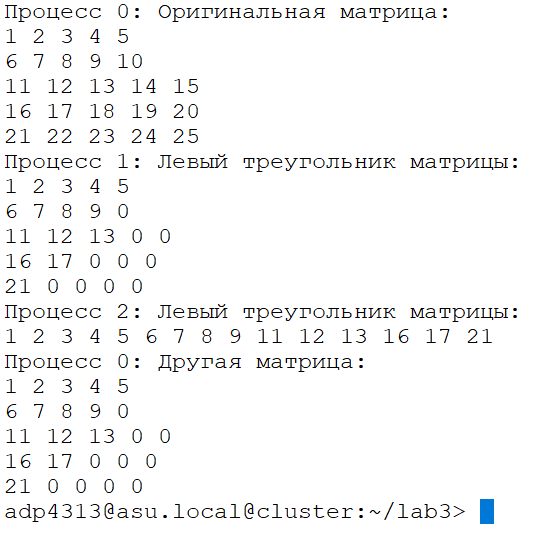


Рисунок 5.1 — Пример работы программы

На рисунке изображён процесс взятия левого верхнего треугольника с матрицы.

### 6 Выводы

Таким образом, я изучил основы работы с конструированием производных типом в MPI и научился их передавать в ранги. С помощью этого способа я решил задачу со взятием левого верхнего угла матрицы.