МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Кемеровский государственный университет»**

**Институт фундаментальных наук**

**ДОМАШНЯЯ РАБОТА №11**

**ПО ДИСЦИПЛИНЕ**

**“Технологии параллельных вычислений”**

студента 3 курса

**Сулима Роман Иванович**

Направление 02.03.02 – Фундаментальная информатика и информационные технологии

Преподаватель:

к-т физ.-мат.наук, доцент

С.В. Стуколов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2021

СОДЕРЖАНИЕ

[1. Постановка задачи 2](#_Toc88855962)

[2. Описание используемых функций 2](#_Toc88855963)

[3. Реализация 2](#_Toc88855964)

[Заключение 5](#_Toc88855965)

# 1. Постановка задачи

Посмотрите в прилагаемом материале еще раз примеры использования совмещенных коллективных операций. Выполните упражнения, приведенные в конце параграфа - 1, 2 и 3-ю задачи. Оформляете в одном отчете - номер задачи, текст условия, текст программы с поясняющими комментариями в программе, скрин компиляции и скрин запуска на несколько процессов. Удачи!

# 2. Описание используемых функций

MPI\_Reduce – Функция выполняющая выбранную мат. Операцию над вектором. Особенность данной функции в том, что результат сохраняется в одном процессе.

MPI\_Reduce\_scatter – Функция совмещаю в себе возможность функции MPI\_Reduce и MPI\_Scatter.

# 3. Реализация

Задание 1.

Создайте и выполните программу, используя распределенную   
вычислительную операцию (MPI\_Reduce), реализующую следующий   
алгоритм: на каждом процессоре с помощью датчика случайных чисел   
генерируется целочисленное значение от 0 до 100, из этих значений   
выбирается максимальное и сохраняется в памяти нулевого процессора.

Программный код:

#include <stdio.h>

#include <stdlib.h>

#include "mpi.h"

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

    int rank;

    int size;

    int s = 0;

    MPI\_Init(&argc, &argv);

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

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

    int b= rand() % 10 + rank;

    MPI\_Reduce(&b ,&s,1,MPI\_INT,MPI\_MAX,0,MPI\_COMM\_WORLD);

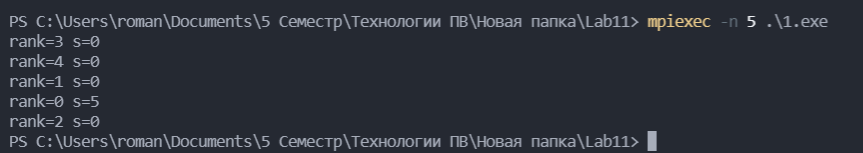
    printf("rank=%d s=%d \n",rank, s);

    MPI\_Finalize();

    return 0;

}

Выполнение:



Задание 2.

Используя функцию MPI\_Reduce\_scatter, напишите следующую   
программу: на 0-м процессе считывается значение переменной n и   
рассылается каждому; каждый процесс объявляет массив a[2\*n\*size] и   
присваивает его элементам значения; каждый процесс объявляет массив   
b[2\*n]; применяя функцию MPI\_Reduce\_scatter, результат суммирования   
распределяется по процессам равными блоками 2\*n элементов в каждом с   
сохранением в массив b.

Программный код:

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[])  {

    int rank;

    int size;

    int n,i,s=0;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    if(rank==0) scanf("%d",&n);

    MPI\_Bcast(&n,1,MPI\_INT,0,MPI\_COMM\_WORLD);

    int \*a=new int[2\*n\*size];

    for(i=0;i<2\*n\*size;i++)

        a[i]=rank+i;

    int \*b=new int[2\*n];

    int \*rc=new int[size];

    for(i=0;i<size;i++)

        rc[i]=2\*n;

    MPI\_Reduce\_scatter(a,b,rc,MPI\_INT,MPI\_SUM,MPI\_COMM\_WORLD);

    for(i=0; i<100000000\*rank; i++)

        s+=1;

    printf("rank= %d  b: ",rank);

    for(i=0;i<2\*n;i++)

    printf(" %d ", b[i]);

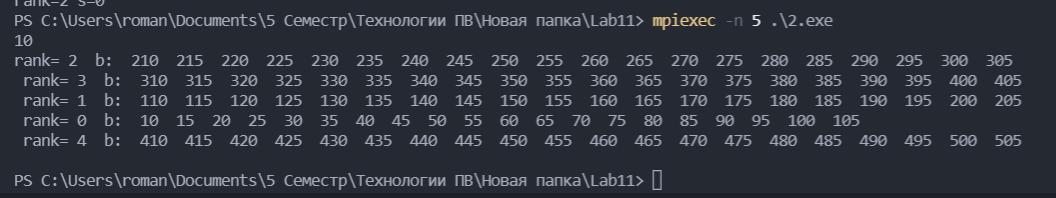
    printf("\n ");

    MPI\_Finalize();

    return 0;

}

Выполнение:



Задание 3.

Модифицируйте предыдущую программу таким образом, чтобы   
результат был распределен следующим образом: на 0-м процессе – b[0],   
на 1-м процессе – b[1], b[2], на 2-м процессе – три следующих элемента   
результирующего вектора и т.д.

Программный код:

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[])  {

    int rank;

    int size;

    int n,i,s=0;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    if(rank==0) scanf("%d",&n);

    MPI\_Bcast(&n,1,MPI\_INT,0,MPI\_COMM\_WORLD);

    int \*a=new int[2\*n\*size];

    for(i=0;i<2\*n\*size;i++)

        a[i]=rank+i;

    int \*b=new int[rank + 1];

    int \*rc=new int[size];

    for(i=0;i<size;i++)

        rc[i]=i + 1;

    MPI\_Reduce\_scatter(a,b,rc,MPI\_INT,MPI\_SUM,MPI\_COMM\_WORLD);

    for(i=0; i<100000000\*rank; i++)

        s+=1;

    printf("rank= %d  b: ",rank);

    for(i=0;i<rank + 1;i++)

    printf(" %d ", b[i]);

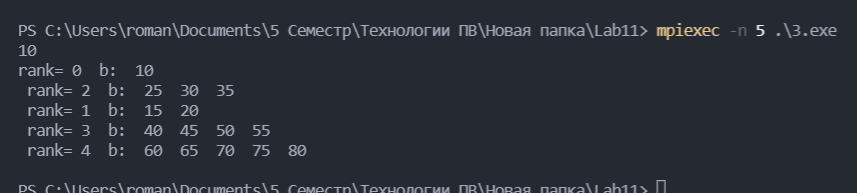
    printf("\n ");

    MPI\_Finalize();

    return 0;

}

Выполнение:



# Заключение

Выполнив лабораторную работу, научился использовать функции для выполнения операций над векторами и сохраняя результат как на один процесс, так и на все.