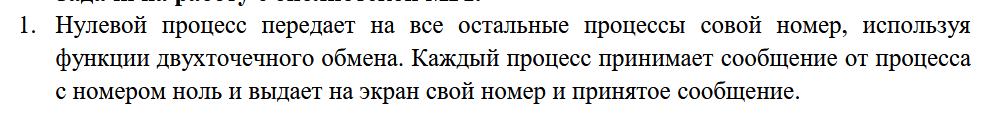
Тестовое задание

Задачи на работу с lib MPI

Боязитов Вадим, 932125

18.04.2025

## Задание 1



### Алгоритм

1. Подготовка

Создать переменные для получения информации о текущем исполняемом процессе.

Создать переменные для приема данных от других процессов.

Инициализация MPI.

2. Условие

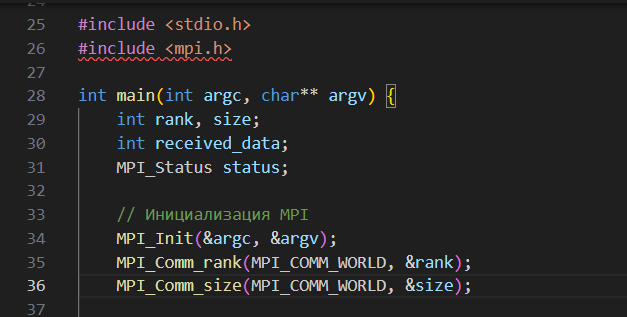
Если процесс 0, то отправить в цикле остальным сообщение.

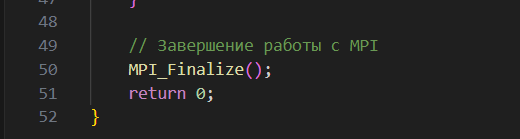
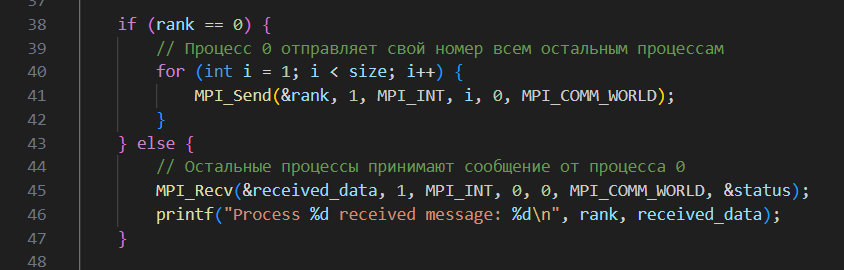
Иначе ждать сообщение от процесса 0.

3. Завершить работу

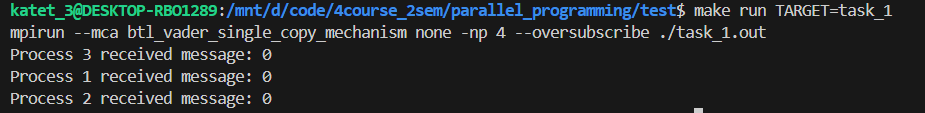
### Программная реализация

Окружение Linix. Язык C.





### Тестирование

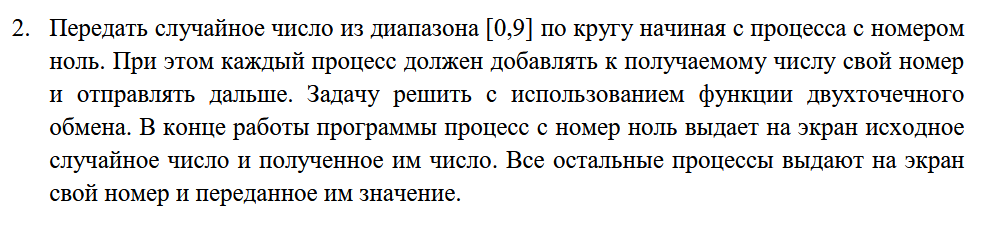


### 

### Вывод

Посредством функций MPI, получилось с коммутировать работу разных процессов.

## Задание 2



### Алгоритм

1. Подготовка

Создать переменные для получения информации о текущем исполняемом процессе.

Создать переменные для приема данных от других процессов.

Создать переменную для случайного числа.

Инициализация MPI

2. Генерация случайного числа

Если процесс 0, то сгенерировать число и запомнить в переменной.

3. Передача числа по кругу

Если Процесс 0:

Отправляет случайное число следующему процессу.

Ожидает число от последнего процесса.

Иначе:

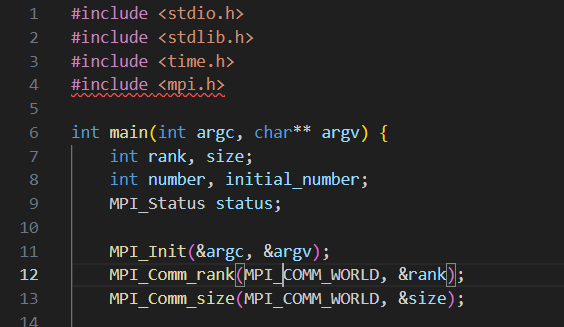
Процесс ждет сообщение от предыдущего процесса.

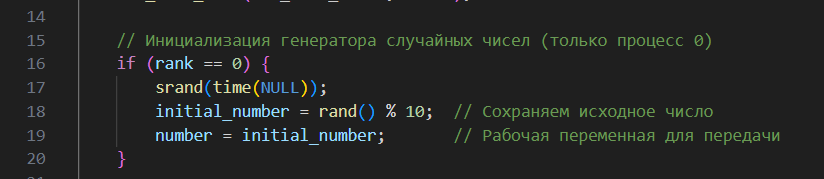
Текущий процесс отправляет сумму полученого числа со своим номером, следующему процессу.

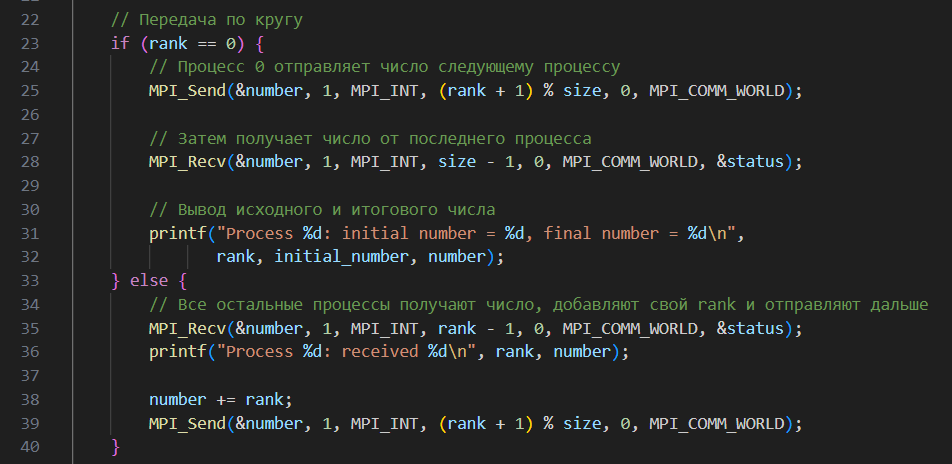
4. Завершить работу

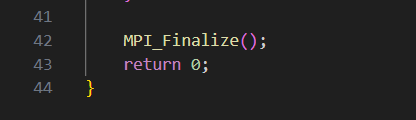
### Программная реализация

Окружение Linix. Язык C.

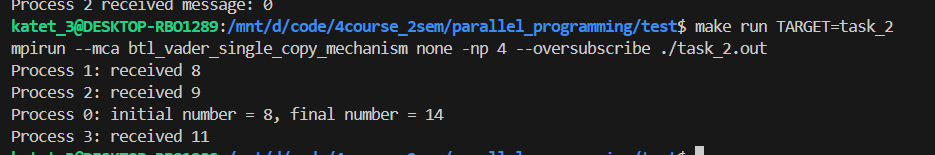








### Тестирование

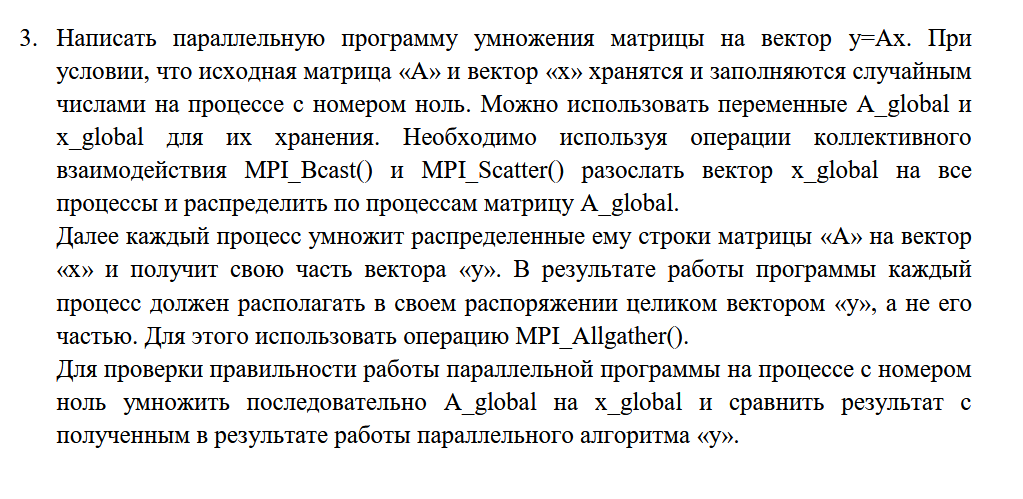


### 

### Вывод

Посредством функций MPI, получилось с коммутировать работу разных процессов.

## Задание 3



### Алгоритм

1. Подготовка

Инициализация MPI

Создать переменные для получения информации о текущем исполняемом процессе.

Создать переменные для умножения вектора и матрицы.

2. Генерация случайной матрицы и вектора

Если процесс 0, то сгенерировать матрицу и вектор в переменной.

3. Отправить вектор другим процессам.

4. Распределить строки матрицы между процессами

5. Каждый процесс умножает свои строки на вектор

6. Собрать полученные части вектора

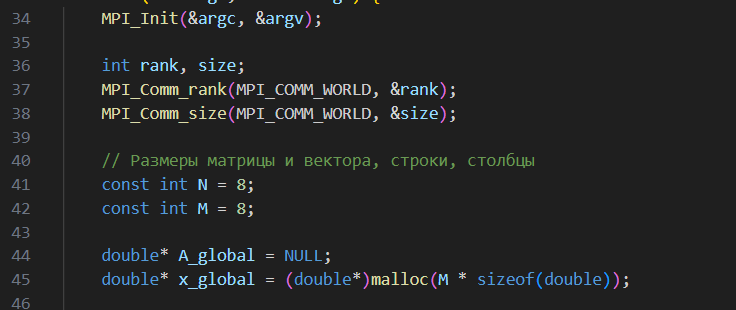
7. Вывод результата с 0 процесса

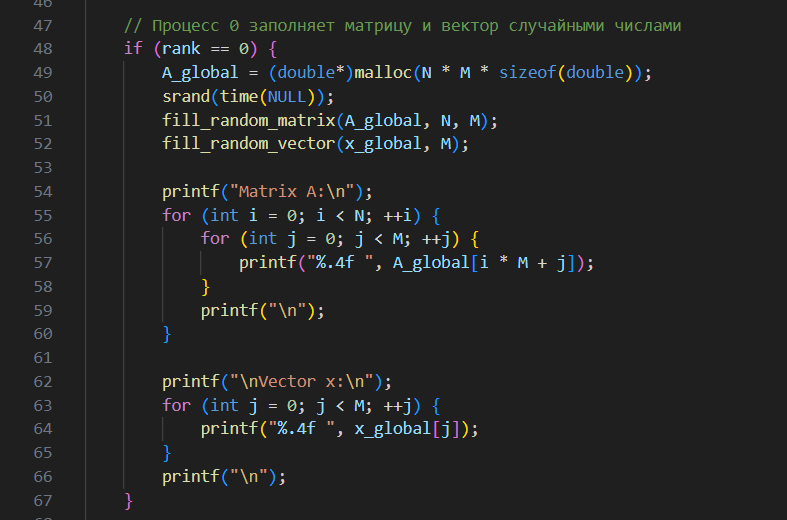
8. Освобождение памяти

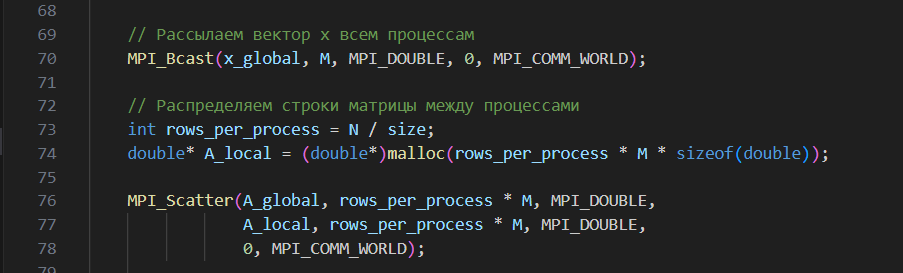
9. Завершить работу

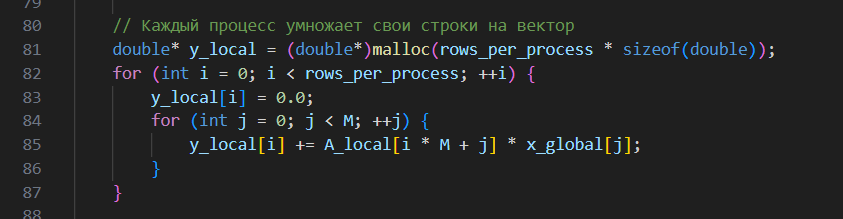
### Программная реализация

Окружение Linix. Язык C.



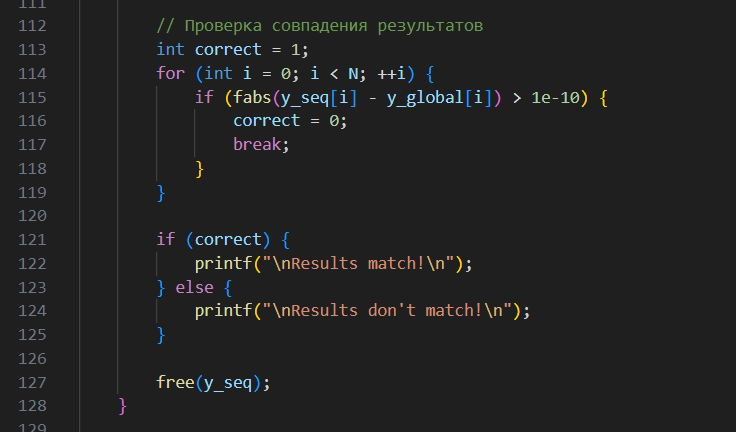


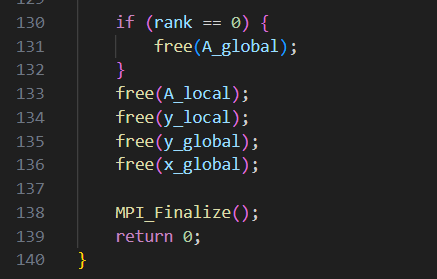




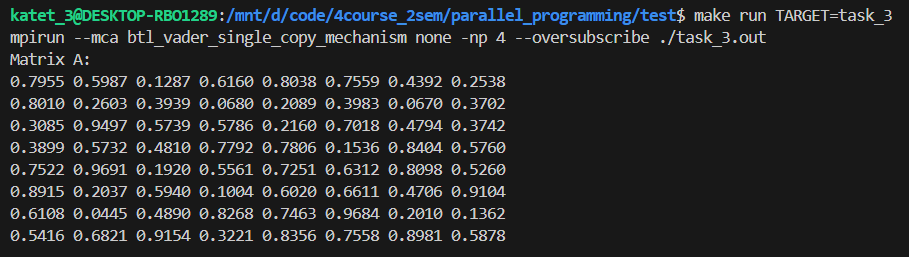
### 

### 





### Тестирование



### 

### Вывод

Посредством функций MPI, получилось с коммутировать работу разных процессов.