МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №00
по дисциплине «Параллельные алгоритмы»
Тема: Запуск параллельной программы на различном числе одновременно работающих процессов и упорядочение вывода ее результатов

Студент гр. 9381 _____ Колованов Р.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2023

Цель работы.

Запуск параллельной программы на различном числе одновременно работающих процессов, упорядочение вывода результатов.

Формулировка задания.

Запустить программу на 1, 2, …, N процессах несколько раз. Проанализировать порядок вывода сообщений на экран. Вывести правило, определяющее порядок вывода сообщений. Модифицировать программу таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

Выполнение работы.

Для начала исходная программа, написанная с использованием технологии MPI, была переписана с использованием технологии OpenMP. Логика работы исходной программы была сохранена:

- В самом начале всегда выводится сообщение с номером процесса 0;
- После этого выводится N – 1 сообщений со случайным порядком следования номеров.

Далее программа, написанная с использованием технологии OpenMP, была модифицирована таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего потока.

Краткое описание алгоритма.

В глобальной переменной *THREADS* задается количество потоков для создания. В глобальном массиве *FLAG* хранится информация о том, какой из потоков уже вывел сообщение со своим номером. Индекс массива соответствует номеру потока.

Функция *printHello* выводит сообщение с номером потока в стандартный поток вывода сообщений, после чего устанавливает соответствующей ячейке массива *FLAG* значение *true*.

В не модифицированной программе функция *printHello* сначала вызывается в основном потоке, после чего данная функция передается на выполнение другим потокам, кроме основного.

В модифицированную программу добавляется синхронизация между потоками при помощи массива *FLAG*. Потоки в бесконечном цикле ждут, пока предыдущий по номеру поток не выведет сообщение на экран, после чего сами его выводят.

Формальное описание алгоритма.

Формальное описание алгоритма на сетях Петри для не модифицированной программы представлена на рис. 1.

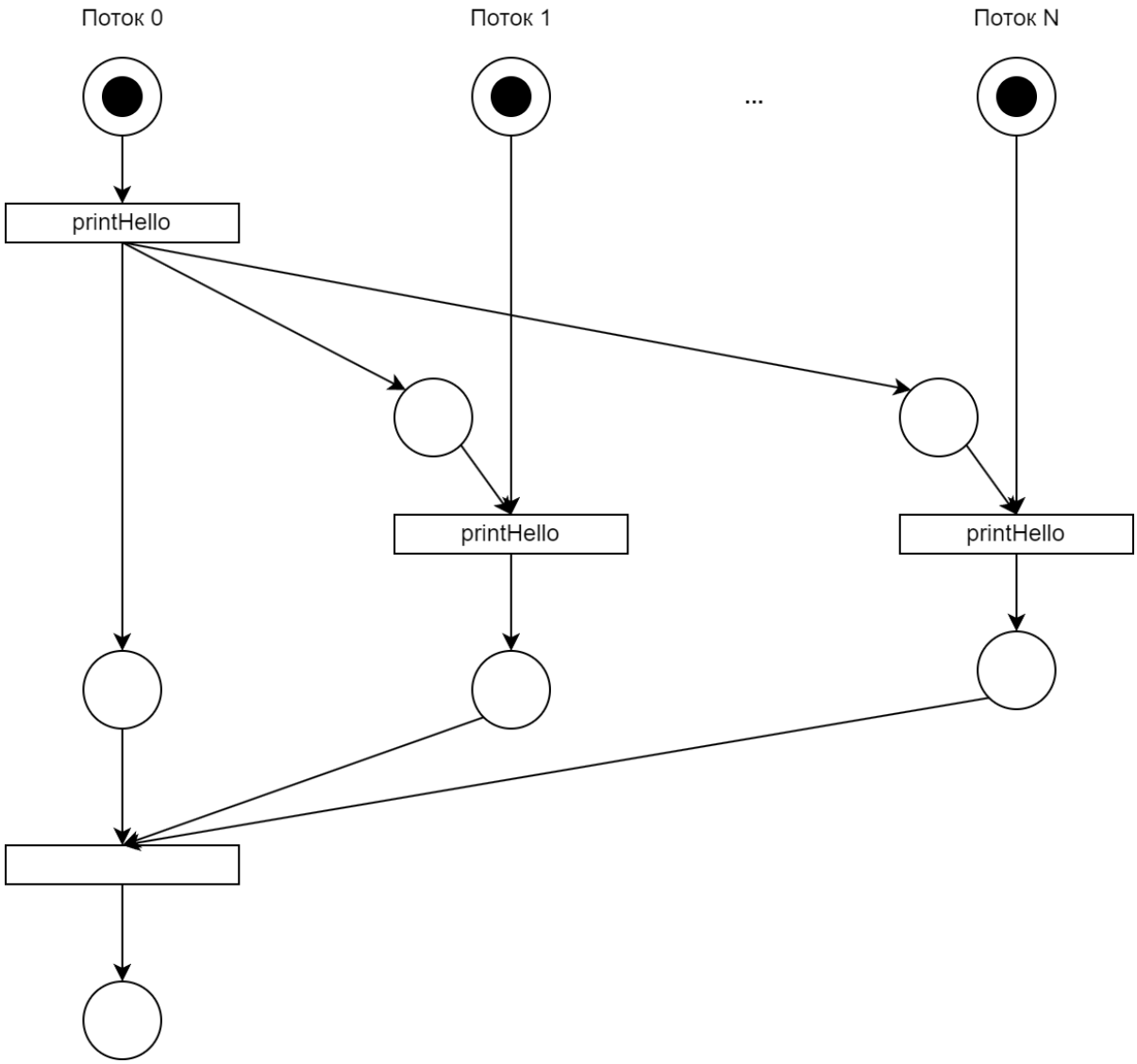

Рисунок 1 – Сеть Петри для не модифицированной программы.

Формальное описание алгоритма на сетях Петри для модифицированной программы представлена на рис. 2.

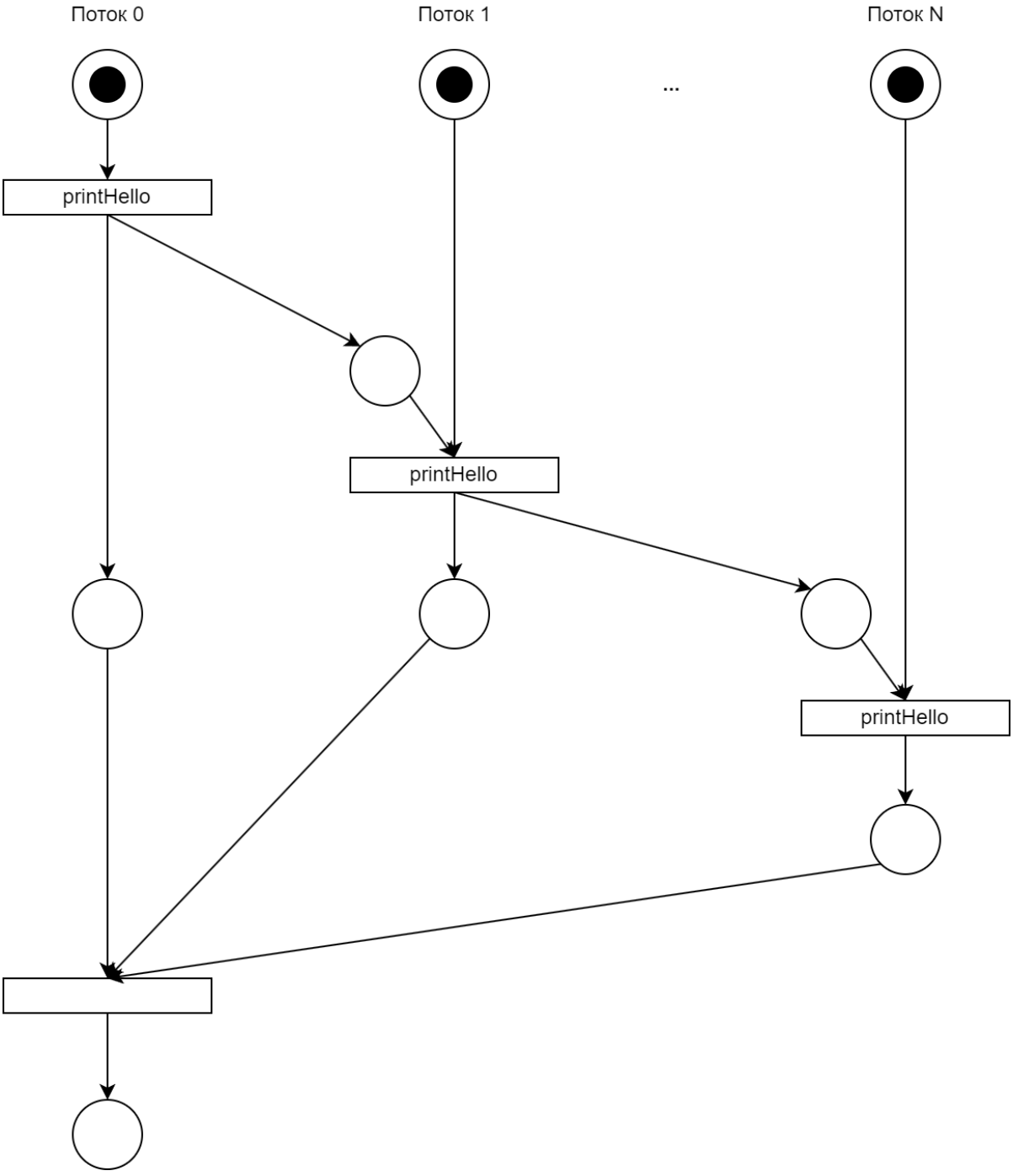

Рисунок 2 – Сеть Петри для модифицированной программы.

Листинг программы.

Исходная программа, использующая технологию MPI, представлена в листинге 1.

Листинг 1. Исходная программа на основе MPI.

```
#include <iostream>
#include <mpi.h>


int main(int argc, char** argv) {
	int processNumber, processRank, recievedRank;
	MPI_Status status;

	MPI_Init(&argc, &argv);
	MPI_Comm_size(MPI_COMM_WORLD, &processNumber);
	MPI_Comm_rank(MPI_COMM_WORLD, &processRank);

	if (processRank == 0) {
		std::cout << "Hello from process " << processRank << "\n";

		for (int i = 1; i < processNumber; i++) {
			MPI_Recv(&recievedRank,	1,	MPI_INT,	i,	MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
			std::cout << "Hello from process " << recievedRank << "\n";
		}

	} else {
		MPI_Send(&processRank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
	}

	MPI_Finalize();

	return 0;
}
```

Не модифицированная программа, использующая технологию OpenMP, представлена в листинге 2.

Листинг 2. Программа на основе OpenMP до модификации.

```
#include <iostream>
#include <string>
#include <omp.h>

namespace
{
  constexpr int THREADS = 1;
}

inline void printHello()
{
  std::cout << "Hello from process " + std::to_string(omp_get_thread_num()) +
"\n";
}

int main()
```

```
{
  printHello();

  #pragma omp parallel num_threads(THREADS)
  {
      if (omp_get_thread_num() != 0)
      {
        printHello();
      }
  }

  return 0;
}
```

Не модифицированная программа, использующая технологию OpenMP, представлена в листинге 3.

Листинг 3. Программа на основе OpenMP после модификации.

```
#include <iostream>
#include <string>
#include <omp.h>

namespace
{
  constexpr int THREADS = 1;
  bool FLAG[THREADS];
}

inline void printHello()
{
  const int threadId = omp_get_thread_num();
  std::cout << "Hello from process " + std::to_string(threadId) + "\n";
  FLAG[threadId] = true;
}

int main()
{
  for (int i = 0; i < THREADS; ++i)
  {
      FLAG[i] = false;
  }

  printHello();

  #pragma omp parallel num_threads(THREADS)
  {
      if (omp_get_thread_num() != 0)
      {
        while (FLAG[omp_get_thread_num() - 1] == false)
        {
            continue;
        }
        printHello();
      }
  }

  return 0;
}
```

Результаты работы программы.

Результаты работы программы до модификации представлены в таблице 1.

Таблица 1. Результаты работы не модифицированной программы.

№ п/п	Количество процессоров	Результаты работы программы
1.	1	Hello from process 0
2.	2	Hello from process 0 Hello from process 1
3.	4	Hello from process 0 Hello from process 2 Hello from process 3 Hello from process 1
4.		Hello from process 0 Hello from process 1 Hello from process 3 Hello from process 2
5.	8	Hello from process 0 Hello from process 1 Hello from process 4 Hello from process 6 Hello from process 3 Hello from process 5 Hello from process 7 Hello from process 2
6.		Hello from process 0 Hello from process 1 Hello from process 7 Hello from process 3 Hello from process 4 Hello from process 5 Hello from process 6 Hello from process 2

Результаты работы программы после модификации представлены в таблице 2.

Таблица 2. Результаты работы модифицированной программы.

№ п/п	Количество процессоров	Результаты работы программы
1.	1	Hello from process 0
2.	2	Hello from process 0 Hello from process 1

Продолжение таблицы 2.

3.	4	Hello from process 0 Hello from process 1 Hello from process 2 Hello from process 3
4.	8	Hello from process 0 Hello from process 1 Hello from process 2 Hello from process 3 Hello from process 4 Hello from process 5 Hello from process 6 Hello from process 7

Выводы.

Были получены навыки компиляции, разработки и запуска параллельной программы, использующей библиотеку OpenMP, на различном числе одновременно работающих потоков. Была выполнена модификация программы таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

Детерминированность результатов работы параллельного алгоритма уменьшает эффективность алгоритма, поскольку в этом случае требуется внедрение механизмов синхронизации, которые замедляют скорость работы.