

Документ подписан простой электронной подписью  
Информация о владельце:  
ФИО: Таскаев Сергей Валерьевич  
Должность: Ректор  
Дата подписания: 15.06.2026 12:27:13  
Уникальный программный ключ:  
04c19e0d80bf98f4b6b775486b9a3788b8312474

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Челябинский государственный университет» (ФГБОУ ВО «ЧелГУ»)

Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю) «Современные компьютерные технологии» по направлению подготовки 01.04.02 «Прикладная математика и информатика» направленности «Математическое моделирование и искусственный интеллект» ФГБОУ ВО «ЧелГУ»

стр. 1

Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю)  
**«Современные компьютерные технологии»**

Направление подготовки (специальность)  
**01.04.02 «Прикладная математика и информатика»**

Направленность (профиль)  
**«Математическое моделирование и искусственный интеллект»**

Присваиваемая квалификация  
**Магистр**

Форма обучения  
**Очная**

Год набора  
**2026**

Челябинск, 2026 г.



## Содержание

1. Паспорт фонда оценочных средств .....	3
2. Перечень формируемых компетенций.....	4
3. Содержание оценочных средств по дисциплине .....	5
3.1. Виды оценочных средств .....	5
3.2. Содержание оценочных средств.....	5
4. Порядок проведения и критерии оценивания промежуточной аттестации .....	21
4.1. Порядок проведения промежуточной аттестации .....	21
4.2. Критерии оценивания промежуточной аттестации по видам оценочных средств.....	21
4.3. Результаты промежуточной аттестации и уровни сформированности компетенций .....	22



## 1. Паспорт фонда оценочных средств

Направление подготовки: 01.04.02 Прикладная математика и информатика.

Направленность: Математическое моделирование и искусственный интеллект.

Дисциплина: Современные компьютерные технологии.

Семестры: 1.

Форма промежуточной аттестации: дифференцированный зачёт в 1 семестре.

Для оценивания результатов обучения используется балльно-рейтинговая система.



## 2. Перечень формируемых компетенций

Изучение дисциплины «Современные компьютерные технологии» направлено на формирование компетенций, приведённых в Таблице 1.

Таблица 1. Результаты обучения по дисциплине.

Код и наименование компетенции согласно ФГОС	Индикаторы достижения компетенций согласно ОПОП ВО	Перечень планируемых результатов обучения по дисциплине
1	2	3
<b>ОПК-4:</b> Способен комбинировать и адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности.	ОПК-4.1. Обладает знаниями о существующих информационнокоммуникационных технологиях и основных требованиях информационной безопасности. ОПК-4.2. Демонстрирует умения комбинировать и адаптировать существующие информационнокоммуникационные технологии, а также умение учитывать основные требования информационной безопасности при решении прикладных задач. ОПК-4.3. Имеет практический опыт комбинирования и адаптации существующих информационнокоммуникационных технологий и учета основных требований информационной безопасности при решении прикладных задач.	Знать требования информационной безопасности в области своей профессиональной деятельности. Уметь использовать информационно-коммуникационные технологии для решения своих профессиональных задач. Владеть практическим опытом использования информационно-коммуникационные технологии для решения своих профессиональных задач.
<b>ПК-2:</b> Способен исследовать и разрабатывать архитектуры систем искусственного интеллекта для различных предметных областей на основе комплексов методов и инструментальных средств систем искусственного интеллекта	ПК-2.1. Исследует и разрабатывает архитектуры систем искусственного интеллекта для различных предметных областей.	Знать методы исследования и разработки архитектуры систем искусственного интеллекта для различных предметных областей. Уметь модернизировать программное и аппаратное обеспечение технологий и систем искусственного интеллекта для решения профессиональных задач. Владеть разработки программного и аппаратного обеспечения технологий и систем искусственного интеллекта.



### 3. Содержание оценочных средств по дисциплине

#### 3.1. Виды оценочных средств

Таблица 2. Виды оценочных средств.

Код, наименование компетенции согласно ФГОС	Перечень планируемых результатов обучения по дисциплине	Контролируемые темы/разделы (номер и название раздела из РПД п.2.2)	Семестр	Номер задания	Наименование оценочного средства
<b>ОПК-4:</b> Способен комбинировать и адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности.	Знать требования информационной безопасности в области своей профессиональной деятельности.	Современные компьютерные технологии.	1	1-6	Тест по технологии MPI
	Уметь использовать информационно-коммуникационные технологии для решения своих профессиональных задач.	Параллельные вычислительные системы Технологии параллельного программирования		1-12	Практические задания по технологии MPI
	Владеть практическим опытом использования информационно-коммуникационные технологии для решения своих профессиональных задач.	Технология OpenMP Технология MPI Возможности вычислительной платформы Julia для обработки больших наборов данных		1-27	Реферат
<b>ПК-2:</b> Способен исследовать и разрабатывать архитектуры систем искусственного интеллекта для различных предметных областей на основе комплексов методов и инструментальных средств систем искусственного интеллекта	Знать методы исследования и разработки архитектуры систем искусственного интеллекта для различных предметных областей.			1-21	Задания по OpenMP
	Уметь модернизировать программное и аппаратное обеспечение технологий и систем искусственного интеллекта для решения профессиональных задач.			1-32	Вопросы к зачету
	Владеть разработки программного и аппаратного обеспечения технологий и систем искусственного интеллекта.				

Типовые задания, критерии и показатели оценивания в рамках текущего контроля представлены в рабочей программе дисциплины (модуля). Полные комплекты оценочных средств и контрольно-измерительных материалов хранятся на кафедре.

#### 3.2. Содержание оценочных средств

Промежуточная аттестация проводится в виде дифференцированного зачёта в 1 семестре.



Оценочные средства представлены набором заданий для доклада, реферата, разноплановых задач, теста, выполняемых студентами в заданные сроки, набором заданий и вопросов к дифференцированному зачёту.

### Темы рефератов:

1. Операционные системы (общего назначения, специального назначения);
2. Языки программирования;
3. Технология архитектуры клиент-сервер;
4. Технологии масштабирования (мощности компьютера, СУБД, сети);
5. Технологии нейровычислений (искусственный интеллект);
6. Телекоммуникационные технологии;
7. Базовые технологии INTERNET;
8. Технологии INTRANET;
9. Современные корпоративные СУБД (Oracle, Informix, Sybase);
10. Технология информационных хранилищ;
11. Экспертные системы и системы поддержки принятия решений;
12. Технологии обработки текстов;
13. Геоинформационные технологии;
14. Мультимедиа технологии и технологии виртуальной реальности;
15. Технологии криптозащиты информации;
16. Технологии человеко-машинного интерфейса;
17. Технологии цифро-аналоговых преобразований;
18. Технологии проектирования информационных систем, контроля и экспертизы;
19. Информационные технологии (ИТ) реализации информационных ресурсов;
20. ИТ в системах массового обслуживания населения;
21. ИТ в обработке экономической информации;
22. ИТ в сфере организации управления;
23. ИТ в сфере интеллектуального потенциала;
24. ИТ в производстве;
25. ИТ в сфере государственной безопасности, в социальной и политической сфере;
26. ИТ сопровождения и сервиса;
27. ИТ обучения.

Примечание. Вышеперечисленные темы задают направления развития информационных технологий. При выборе конкретной темы допустимо и уместно выбрать для обзора одну или несколько конкретных технологий, попадающих в выбранное направление.

### Тестовые задания по технологии MPI

1. Для данной MPI программы запускаемой на 3 процессах ( $n_{proc} = 3$ ) с входными данными (в каждой строке набор чисел для одного процесса):

2 7 6 2 5 2

5 4 3 8 7 1

8 3 9 3 2 6

выбрать вариант с правильным выводом на (общую) консоль результата работы программы:

a) 2 6 5 3 7 2

b) 5 3 3 2 2 1



с) 8 7 9 8 7 6

d) 2 3 3 2 2 1 Правильный!

////////////////////////////////////

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[]) {
    int rank, nprocs;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    MPI_Status status;

    int count = 3 + nprocs;
    int rec_numbers[count], send_numbers[count];

    for (int i = 0; i < count; i++) {
        //Здесь загружаются данные для каждого процесса
    }

    MPI_Reduce(send_numbers, rec_numbers, count, MPI_INT, MPI_MIN, 0,
    MPI_COMM_WORLD);

    if (rank == 0) {
        for (int i = 0; i < count; i++) {
            printf("%d ", rec_numbers[i]);
        }
        printf("\n");
    }
    MPI_Finalize();
    return 0;
}
```

////////////////////////////////////

2. Для данной MPI программы, запускаемой на 3 процессах (nprocs = 3) с входными данными (в каждой строке набор чисел для одного процесса):

6 7 6 2 3 5  
8 1 3 8 7 1  
5 4 9 1 5 6

выбрать вариант с правильным выводом на (общую) консоль результата работы программы:

a) 2 6 5 3 7 2

b) 8 7 9 8 7 6 Правильный!

с) 7 9 8 7 6 6



d) 2 3 3 2 2 1

////////////////////////////////////

```
#include <stdio.h>
```

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

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

```
    int rank, nprocs;
```

```
    MPI_Init(&argc, &argv);
```

```
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
```

```
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
    MPI_Status status;
```

```
    int count = 3 + nprocs;
```

```
    int rec_numbers[count], send_numbers[count];
```

```
    for (int i = 0; i < count; i++) {
```

```
        //Здесь загружаются данные для каждого процесса
```

```
    }
```

```
    MPI_Reduce(send_numbers, rec_numbers, count, MPI_INT, MPI_MAX, 0,  
MPI_COMM_WORLD);
```

```
    if (rank == 0) {
```

```
        for (int i = 0; i < count; i++) {
```

```
            printf("%d ", rec_numbers[i]);
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
    MPI_Finalize();
```

```
    return 0;
```

```
}
```

////////////////////////////////////

3. Для данной MPI программы, запускаемой на 3 процессах (проц = 3) с входными данными (в каждой строке набор чисел для одного процесса):

1 2 3 2 1 4

1 2 2 3 2 1

2 3 2 3 4 2

выбрать вариант с правильным выводом на (общую) консоль результата работы программы:

a) 2 12 12 18 8 8

2 12 12 18 8 8

2 12 12 18 8 8

b) 4 7 7 8 7 7

4 7 7 8 7 7



4 7 7 8 7 7

с) 2 12 12 18 8 8 Правильный!  
2 12 12 18 8 8

d) 2 12 12 18 8 8

```
////////////////////////////////////  
#include <stdio.h>  
#include <mpi.h>  
int main(int argc, char* argv[]) {  
    int nprocs, rank;  
    MPI_Status status;  
    MPI_Init(&argc, &argv);  
  
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);  
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);  
    int n = nprocs + 3;  
    int nums_send[n], nums_rec[n];  
    for(int i = 0; i < n; i++) {  
        //Здесь загружаются начальные данные  
    }  
    MPI_Allreduce(nums_send, nums_rec, n, MPI_INT, MPI_PROD,  
MPI_COMM_WORLD);  
    if (rank < 2){  
        for(int i = 0; i < n; i++) {  
            printf("%d ", nums_rec[i]);  
        }  
        printf("\n\n");  
    }  
    MPI_Finalize();  
    return 0;  
}  
////////////////////////////////////
```

4. Для данной MPI программы, запускаемой на 3 процессах (nprocs = 3) с входными данными (в каждой строке набор чисел для одного процесса):

2 2 3 2 1 3  
2 2 2 3 2 1  
2 1 3 3 2 2

выбрать вариант с правильным выводом на (общую) консоль результата работы программы:

a) 3 2 3 3 2 2 Правильный!

b) 2 2 3 3 2 3  
2 2 3 3 2 3  
2 2 3 3 2 3



c) 3 2 3 3 2 2  
3 2 3 3 2 2  
3 2 3 3 2 2

d) 2 2 3 3 2 3

```
////////////////////////////////////  
#include <stdio.h>  
#include <mpi.h>  
int main(int argc, char* argv[]) {  
    int nprocs, rank;  
    MPI_Status status;  
    MPI_Init(&argc, &argv);  
  
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);  
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);  
    int n = nprocs + 3;  
    int nums_send[n], nums_rec[n];  
    for(int i = 0; i < n; i++) {  
        //Здесь загружаются начальные данные  
    }  
    MPI_Allreduce(nums_send, nums_rec, n, MPI_INT, MPI_MAX,  
MPI_COMM_WORLD);  
    if (rank == 0){  
        for(int i = n - 1; i > -1; --i) {  
            printf("%d ", nums_rec[i]);  
        }  
        printf("\n\n");  
    }  
    MPI_Finalize();  
    return 0;  
}
```

5. Для данной MPI программы, запускаемой на 3 процессах (проц = 3) выбрать возможный вариант с правильным выводом на (общую) консоль результата работы программы:

a) 2 2 2 Правильно!  
4 4 4  
6 6 6  
  
1 1 1  
2 2 2  
3 3 3  
  
3 3 3



6 6 6  
9 9 9

b) 3 6 9  
3 6 9  
3 6 9

2 4 6  
2 4 6  
2 4 6

1 2 3  
1 2 3  
1 2 3

c) 3 3 3  
9 9 9  
6 6 6

2 4 6  
2 4 6  
2 4 6

1 1 1  
2 2 2  
3 3 3

////////////////////////////////////

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

```
int main(int argc, char *argv[]) {  
    int rank, nprocs;  
  
    MPI_Init(&argc, &argv);  
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);  
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);  
  
    int inbuf [nprocs][3];  
  
    for (int i = 0; i < nprocs; i++){  
        for(int j = 0; j < 3; j++){  
            inbuf[i][j] = (i + 1)*(rank + 1);  
        }  
    }  
    int outbuf [nprocs][3];  
    MPI_Alltoall(inbuf, 3, MPI_INT, outbuf, 3, MPI_INT, MPI_COMM_WORLD);  
    for (int i = 0; i < nprocs; i++){
```



```
        for (int j = 0; j < 3; j++){  
            printf("%d ", outbuf[i][j]);  
        }  
        printf("\n");  
    }  
    printf("\n");  
    MPI_Finalize();  
    return 0;  
}
```

////////////////////////////////////

6. Для данной MPI программы, запускаемой на 3 процессах (проц = 3) выбрать возможный вариант с правильным выводом на (общую) консоль результата работы программы:

a) 2 2 2  
4 4 4  
6 6 6

3 3 3  
6 6 6  
9 9 9

1 1 1  
2 2 2  
3 3 3

b) 3 6 9 //Правильно!  
3 6 9  
3 6 9

2 4 6  
2 4 6  
2 4 6

1 2 3  
1 2 3  
1 2 3

c) 3 3 3  
9 9 9  
6 6 6

2 4 6  
2 4 6  
2 4 6

1 1 1



```
2 2 2
3 3 3
////////////////////////////////////
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int rank, nprocs;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    int inbuf [nprocs][3];

    for (int i = 0; i < nprocs; i++){
        for(int j = 0; j < 3; j++){
            inbuf[i][j] = (i + 1)*(rank + 1);
        }
    }
    int outbuf [nprocs][3];
    MPI_Alltoall(inbuf, 3, MPI_INT, outbuf, 3, MPI_INT, MPI_COMM_WORLD);
    for (int i = 0; i < nprocs; i++){
        for (int j = 0; j < 3; ++j){
            printf("%d ", outbuf[j][i]);
        }
        printf("\n");
    }
    printf("\n");
    MPI_Finalize();
    return 0;
}
////////////////////////////////////
```

### **Практические задания по использованию технологии MPI (начальный уровень)**

Если количество процессов в задании не определено, то можно считать, что это количество не превосходит 12. Под *главным процессом* всюду в формулировках заданий понимается процесс ранга 0 для коммуникатора MPI\_COMM\_WORLD. Для всех процессов ненулевого ранга в заданиях используется общее наименование *подчиненных процессов*.

Если в задании не указан тип обрабатываемых чисел, то числа считаются вещественными. Если в задании не определяется максимальный размер набора чисел, то его следует считать равным 10.

### **ПРОЦЕССЫ И РАНГИ**

**Задание 1.** В процессах четного ранга (включая главный) ввести целое число, в процессах нечетного ранга ввести вещественное число. В каждом процессе вывести удвоенное значение введенного числа.



**Задание 2.** В каждом процессе дано целое число  $N (> 0)$  и набор из  $N$  чисел. В процессах четного ранга (включая главный) вывести сумму чисел из данного набора, в процессах нечетного ранга вывести среднее арифметическое чисел из данного набора.

#### ОБМЕН СООБЩЕНИЯМИ МЕЖДУ ОТДЕЛЬНЫМИ ПРОЦЕССАМИ

**Задание 3.** В главном процессе дан набор вещественных чисел; количество чисел равно количеству подчиненных процессов. С помощью функции `MPI_Bsend` переслать по одному числу в каждый из подчиненных процессов, перебирая процессы в обратном порядке (первое число в последний процесс, второе — в предпоследний процесс, и т.д.), и вывести в подчиненных процессах полученные числа.

**Задание 4.** Количество процессов — четное число. В каждом процессе дано целое число  $N (0 < N < 5)$  и набор из  $N$  чисел. С помощью функции `MPI_Sendrecv` выполнить обмен исходными наборами между парами процессов 0 и 1, 2 и 3, и т.д. В каждом процессе вывести полученный набор чисел.

**Задание 5.** В каждом подчиненном процессе даны два целых числа  $T, N$  и набор из  $N$  чисел. Число  $T$  равно 0 или 1. Набор содержит целые числа, если  $T = 0$ , и вещественные числа, если  $T = 1$ . Переслать исходные наборы в главный процесс и вывести полученные числа в порядке возрастания рангов переславших их процессов. Для передачи информации о типе пересланного числа указывать число  $T$  в качестве параметра `msgtag` функции `MPI_Send`, для получения этой информации использовать функцию `MPI_Probe` с параметром `MPI_ANY_TAG`.

#### КОЛЛЕКТИВНАЯ ПЕРЕСЫЛКА ДАННЫХ

**Задание 6.** В главном процессе дан набор из  $K$  чисел, где  $K$  — количество процессов. Не меняя порядок расположения чисел в исходном наборе и используя функцию `MPI_Scatterv`, переслать по одному числу в каждый процесс; при этом первое число надо переслать в процесс  $K-1$ , второе число — в процесс  $K-2$ , ..., последнее число — в процесс 0. Вывести в каждом процессе полученное число.

**Задание 7.** В каждом процессе даны четыре целых числа. Используя функцию `MPI_Allgather`, переслать эти числа во все процессы и вывести их в каждом процессе в порядке возрастания рангов переславших их процессов (включая числа, полученные из этого же процесса).

**Задание 8.** В каждом процессе дан набор из  $3K$  целых чисел, где  $K$  — количество процессов. Используя функцию `MPI_Alltoall`, переслать в каждый процесс три очередных числа из каждого набора (в процесс 0 — первые три числа, в процесс 1 — следующие три числа, и т.д.). В каждом процессе вывести числа в порядке возрастания рангов переславших их процессов (включая числа, полученные из этого же процесса).

#### КОЛЛЕКТИВНЫЕ ОПЕРАЦИИ РЕДУКЦИИ

**Задание 9.** В каждом процессе дан набор из  $K+5$  чисел, где  $K$  — количество процессов. Используя функцию `MPI_Reduce` для операции `MPI_MIN`, найти минимальное значение среди элементов данных наборов с одним и тем же порядковым номером и вывести полученные минимумы в главном процессе.

**Задание 10.** В каждом процессе дан набор из  $K+5$  чисел, где  $K$  — количество процессов. Используя функцию `MPI_Allreduce` для операции `MPI_PROD`, перемножить элементы данных наборов с одним и тем же порядковым номером и вывести полученные произведения во всех процессах.



## ПРОИЗВОДНЫЕ ТИПЫ И УПАКОВКА ДАННЫХ

**Задание 11.** В главном процессе дана  $K-1$  тройка целых чисел, где  $K$  — количество процессов. Используя производный тип, содержащий три целых числа, и одну коллективную операцию пересылки данных, переслать по одной тройке чисел в каждый из подчиненных процессов и вывести их в подчиненных процессах в том же порядке.

**Задание 12.** В каждом процессе даны три числа: первое и третье являются целыми, а второе — вещественным. Используя производный тип-структуру, содержащую три числа соответствующего типа, переслать данные из каждого процесса во все процессы и вывести в каждом процессе полученные числа в порядке возрастания рангов переславших их процессов (включая числа, полученные из этого же процесса).

### Задания по OpenMP (начальный уровень)

#### *Задание 1. Многопоточная программа «Hello World!»*

Напишите OpenMP-программу, в которой создается 4 нити и каждая нить выводит на экран строку «Hello World!».

**Входные данные:** нет.

**Выходные данные:** 4-е строки «Hello World!».

#### *Задание 2. Программа «I am!»*

1. Напишите программу, в которой создается  $k$  нитей, и каждая нить выводит на экран свой номер и общее количество нитей в параллельной области в формате:

I am <Номер нити> thread from <Количество нитей> threads!

**Входные данные:**  $k$  – количество нитей в параллельной области.

**Выходные данные:**  $k$  строк вида «I am <Номер нити> thread from <Количество нитей> threads!».

2. Модифицируйте программу таким образом, чтобы строку I am <Номер нити> thread from <Количество нитей> threads! выводили только нити с четным номером.

#### *Задание 3. Общие и частные переменные в OpenMP: программа «Скрытая ошибка»*

Изучите конструкции для управления работой с данными shared и private. Напишите программу, в которой создается  $k$  нитей, и каждая нить выводит на экран свой номер через переменную rank следующим образом:

```
rank = omp_get_thread_num(); printf("I am %d thread.\n", rank);
```

Экспериментами определите, общей или частной должна быть переменная rank.

**Входные данные:** целое число  $k$  – количество нитей в параллельной области.

**Выходные данные:**  $k$  строк вида «I am <Номер нити>.».

#### *Задание 4. Общие и частные переменные в OpenMP: параметр reduction*

1. Напишите программу, в которой две нити параллельно вычисляют сумму чисел от 1 до  $N$ . Распределите работу по нитям с помощью оператора if языка C. Для сложения результатов вычисления нитей воспользуйтесь OpenMP-параметром reduction.

**Входные данные:** целое число  $N$  – количество чисел.

**Выходные данные:** каждая нить выводит свою частичную сумму в формате «[Номер\_нити]: Sum = <частичная\_сумма>», один раз выводится общая сумма в формате «Sum = <сумма>».



2\*. Модифицируйте программу таким образом, чтобы она работала для  $k$  нитей.

**Входные данные:** целое число  $k$  – количество нитей, целое число  $N$  – количество чисел.

**Выходные данные:** каждая нить выводит свою частичную сумму в формате «[Номер\_нити]: Sum = <частичная\_сумма>», один раз выводится общая сумма в формате «Sum = <сумма>».

**Задание 5. Распараллеливание циклов в OpenMP: программа «Сумма чисел»**

Изучите OpenMP-директиву параллельного выполнения цикла for. Напишите программу, в которой  $k$  нитей параллельно вычисляют сумму чисел от 1 до  $N$ . Распределите работу по нитям с помощью OpenMP- директивы for.

**Входные данные:** целое число  $k$  – количество нитей, целое число  $N$  – количество чисел.

**Выходные данные:** каждая нить выводит свою частичную сумму в формате «[Номер\_нити]: Sum = <частичная\_сумма>», один раз выводится общая сумма в формате «Sum = <сумма>»

**Задание 6. Распараллеливание циклов в OpenMP: параметр schedule**

Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания 5 таким образом, чтобы дополнительно выводилось на экран сообщение о том, какая нить, какую итерацию цикла выполняет:

[<Номер нити>]: calculation of the iteration number <Номер итерации>.

Задайте  $k = 4$ ,  $N = 10$ . Заполните следующую таблицу распределения итераций цикла по нитям в зависимости от параметра schedule:

Номер итерации	Значение параметра schedule						
	static	static, 1	static, 2	dynamic	dynamic, 2	guided	guided, 2
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

**Задание 7. Распараллеливание циклов в OpenMP: программа «Число  $\pi$ »**

Напишите OpenMP-программу, которая вычисляет число  $\pi$  с точностью до  $N$  знаков после запятой. Используйте следующую формулу:

$$\pi = \left( \frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \dots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, i = \overline{0, N-1}$$



Распределите работу по нитям с помощью OpenMP-директивы for.

**Входные данные:** одно целое число N (точность вычисления).

**Выходные данные:** одно вещественное число  $\rho_i$ .

**Задание 8. Распараллеливание циклов в OpenMP: программа «Матрица»**

Напишите OpenMP-программу, которая вычисляет произведение двух квадратных матриц  $A \times B = C$  размера  $n \times n$ . Используйте следующую формулу:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{pmatrix}$$
$$c_{im} = \sum_{j=1}^n a_{ij} \cdot b_{jm}; i = 1, 2, \dots, n; m = 1, 2, \dots, n$$
$$C = \begin{pmatrix} \sum_{j=1}^n a_{1j} \cdot b_{j1} & \sum_{j=1}^n a_{1j} \cdot b_{j2} & \sum_{j=1}^n a_{1j} \cdot b_{j3} & \dots & \sum_{j=1}^n a_{1j} \cdot b_{jn} \\ \sum_{j=1}^n a_{2j} \cdot b_{j1} & \sum_{j=1}^n a_{2j} \cdot b_{j2} & \sum_{j=1}^n a_{2j} \cdot b_{j3} & \dots & \sum_{j=1}^n a_{2j} \cdot b_{jn} \\ \dots & \dots & \dots & \dots & \dots \\ \sum_{j=1}^n a_{nj} \cdot b_{j1} & \sum_{j=1}^n a_{nj} \cdot b_{j2} & \sum_{j=1}^n a_{nj} \cdot b_{j3} & \dots & \sum_{j=1}^n a_{nj} \cdot b_{jn} \end{pmatrix}$$

**Входные данные:** целое число n,  $1 \leq n \leq 10$ , n<sup>2</sup> вещественных элементов матрицы A и n<sup>2</sup> вещественных элементов матрицы B.

**Выходные данные:** n<sup>2</sup> вещественных элементов матрицы C.

**Задание 9. Параллельные секции в OpenMP: программа «I'm here»**

Изучите OpenMP-директивы создания параллельных секций sections и section. Напишите программу, содержащую 3 параллельные секции, внутри каждой из которых должно выводиться сообщение:

[<Номер нити>]: same in section <Номер секции>

Вне секций внутри параллельной области должно выводиться следующее сообщение:

[<Номер нити>]: parallel region

Запустите приложение на 2-х, 3-х, 4-х нитях. Проследите, как нити распределяются по параллельным секциям.

**Входные данные:** k – количество нитей в параллельной области.

**Выходные данные:** k-строк вида «[<Номер нити>]: same in section <Номер секции>», k-строк вида «[<Номер нити>]: parallel region».

**Задание 10. Гонка потоков в OpenMP: программа «Сумма чисел» с atomic**



Перепишите программу, в которой параллельно вычисляется сумма чисел от 1 до  $N$  (см. задание 5), без использования параметра `reduction`. Вместо параметра `reduction` используйте директиву `atomic`.

***Задание 11. Гонка потоков в OpenMP: программа «Число  $\pi$ » с `critical`***

Перепишите параллельную программу вычисления числа  $\pi$  (см. задание 7) без использования параметра `reduction`. Вместо параметра `reduction` используйте директиву `critical`.

ДЛЯ КАЖДОЙ ИЗ ТРЕХ СЛЕДУЮЩИХ ЗАДАЧ ОПРЕДЕЛИТЬ УСКОРЕНИЕ И ПОСТРОИТЬ ТАБЛИЦУ ЗАВИСИМОСТИ УСКОРЕНИЯ ОТ ЧИСЛА ПОТОКОВ ( $N=1..8$ ). ФОРМУ ТАБЛИЦЫ РАЗРАБОТАТЬ САМОСТОЯТЕЛЬНО. ЕСЛИ ЗАДАЧУ МОЖНО РАСПАРАЛЛЕЛИТЬ БОЛЕЕ ЧЕМ ОДНИМ СПОСОБОМ, ПОПРОБУЙТЕ НАЙТИ НАИБОЛЕЕ ПРОИЗВОДИТЕЛЬНЫЙ СПОСОБ.

***Задание 12. Напишите параллельную программу, реализующую скалярное произведение 2 векторов***

***Задание 13. Напишите параллельную программу, реализующую поиск максимального значения вектора.***

***Задание 14. Напишите параллельную программу, реализующую транспонирование матрицы  $n \times n$ .***

**Задания по OpenMP (промежуточный уровень)**

Задания №15 - №19 начинать с разработки «последовательных» версий алгоритмов. После создания «параллельных» версий решений вычислить ускорения и время выполнения, варьируя количество потоков от 1 до 8. Результаты занести в таблицу (для каждого задания). ОБЯЗАТЕЛЬНО – подобрать начальные параметры задач таким образом, чтобы последовательный вариант выполнялся не менее 5 с. Результаты измерений усреднять не менее чем по 5 замерам.

***Задание 15. Рекурсивное вычисление чисел Фибоначчи***

Разработать параллельный алгоритм, написать и отладить программу вычисления чисел Фибоначчи (рекурсией).

**Указание:** использовать директиву `task` – для запуска в нитях рекурсивных вызовов функции

***Задание 16. Параллельная «быстрая» сортировка***

Разработать параллельный алгоритм «быстрой» сортировки (метод Хоара, или Quicksort), написать и отладить программу.

***Задание 17. Параллельный алгоритм Флойда-Уоршелла***

Разработать параллельную версию алгоритма Флойда – Уоршелла, поиска кратчайших путей, между вершинами ориентированного (или неориентированного) взвешенного графа.

***Задание 18. Параллельный алгоритм Дейкстры***



Разработать параллельную версию алгоритма Дейкстры, поиска кратчайших путей от одной из вершин взвешенного графа (с положительными весами всех ребер) до всех остальных.

***Задание 19. Решение СЛАУ методом сопряженных градиентов, локально-оптимальная схема в OpenMP***

Разработать параллельный алгоритм, написать и отладить параллельную программу решения СЛАУ методом сопряженных градиентов локально-оптимальная схема в OpenMP.

Использовать программу для последовательного алгоритма в соответствующем указании.

***Задание 20. Параллельный алгоритм решения СЛАУ методом Гаусса.***

Разработать параллельный алгоритм, написать и отладить параллельную программу решения СЛАУ методом Гаусса. В данном алгоритме предполагается, что матрица коэффициентов распределена по нитям одним из способов, описанных в соответствующем указании.

Реализовать оба способа.

***Задание 21. Параллельный алгоритм решения задачи Пуассона в 3d методом Зейделя***

Разработать параллельный алгоритм, написать и отладить параллельную программу решения задачи Пуассона методом Зейделя. Использовать программу для последовательного алгоритма в соответствующем указании.

Вопросы и задания к дифференцированному зачёту:

1. Классификация компьютерных технологий по уровню взаимодействия элементов;
2. Тенденции развития компьютерных технологий;
3. Облачные и параллельные технологии;
4. Определение параллелизма, его разновидности и уровни;
5. Определение параллельной вычислительной системы (ПВС);
6. История возникновения, развитие ПВС;
7. Архитектура ПВС;
8. Классификация по Флинну, иерархия, тенденции развития;
9. Параллельная форма алгоритма;
10. Графы зависимостей;
11. Гипотеза об ограниченном количестве базовых информационных структур;
12. Методики построения параллельных алгоритмов;
13. Закон Амдала;
14. Критерии выбора параллельных технологий;
15. Классификация параллельных технологий (по конечной реализации);
16. Программирование в общей памяти, модель Fork and Join;
17. Характеристика стандарта OpenMP, роль компилятора;
18. Общая структура OpenMP – программы;
19. Элементы технологии: директивы; подпрограммы; переменные окружения;
20. Базовые приемы распараллеливания с помощью OpenMP;
21. Модель передачи сообщений в системах с распределенной памятью;
22. Стандарт MPI. Цикл разработки программы в MPI;
23. Элементы MPI: сообщения; коммуникаторы; функции;
24. Общая структура MPI-программы;
25. Виды межпроцессорного взаимодействия, обслуживающие их функции;



26. Коллективные операции. Основные приемы и особенности распараллеливания с помощью MPI;
27. Основы гибридного программирования в стандартах OpenMP и MPI;
28. Возможности вычислительной платформы Julia;
29. Средства разработки под платформу Julia ;
30. Характеристика и возможности языка программирования Julia;
31. Структура программы на языке Julia, организация работы в оболочке REPL;
32. Пакеты для обработки и визуализации наборов данных и их базовая функциональность.



## 4. Порядок проведения и критерии оценивания промежуточной аттестации

### 4.1. Порядок проведения промежуточной аттестации

Промежуточная аттестация проводится в форме дифференцированного зачета в 1 семестре.

При проведении промежуточной аттестации, осуществляется подсчет суммарного количества баллов, полученного студентом в процессе текущего контроля. Оценка «зачтено» выставляется если количество набранных баллов  $>80$ , «не зачтено» ставиться если количество баллов  $\leq 80$ .

### 4.2. Критерии оценивания промежуточной аттестации по видам оценочных средств

Таблица оценки средств контроля:

Наименование средства	Максимальное количество баллов
Реферат	15 баллов
Доклад	15 баллов
Задания OpenMP 1ч.	10 баллов
Задания OpenMP 2ч.	35 баллов
Задания MPI	15 баллов
Тест (MPI)	10 баллов
Итого:	100 баллов

Замечание. Прохождение студентом всех средств контроля обязательно. В случае отсутствия хотя бы одного пройденного средства может быть выставлена оценка «не зачтено» вне зависимости от набранного суммарного количества баллов.

Критерии оценивания промежуточной аттестации по видам оценочных средств:

Критерии оценивания доклада:

Минимальная оценка 5 балла

до +5 баллов может быть добавлено за раскрытие темы

до +5 баллов может быть добавлено за уверенность исполнения

Критерии оценивания реферата:

Минимальная оценка 5 балла

до +10 баллов может быть добавлено за раскрытие темы

Критерии оценивания решения задач:

Задачи по OpenMP ч.1:

0.72 балла за задачу (всего 14 задач – 10 баллов)

Задачи по OpenMP ч.2:

3.64 балла за задачу с номерами от 15 до 18 (4 задачи – 9 баллов)

8 баллов за задачу с номерами от 19 до 21 (3 задачи – 24 балла)

Задачи по MPI:

1.25 балла за задачу (всего 12 задач - 15 баллов)

Критерии оценивания теста (MPI):



Тест считается пройденным, если правильно отвечено на 5 из 6 вопросов – в этом случае засчитывается 10 баллов, противном случае – 0 баллов.

#### **4.3. Результаты промежуточной аттестации и уровни сформированности компетенций**

Возможные оценки на дифференцированном зачёте:

«Зачтено / Отлично» («5») – студент глубоко и полно владеет содержанием учебного материала и понятийным аппаратом; умеет связывать теорию с практикой, иллюстрировать примерами, фактами, данными научных исследований; обозначает межпредметные связи. Делает выводы логично, четко. Ясно и кратко излагает ответы на поставленные вопросы; умеет обосновывать свои суждения и профессионально-личностную позицию по излагаемому вопросу. Ответ носит самостоятельный характер.

«Зачтено / Хорошо» («4») – ответ студента соответствует указанным выше критериям, но содержание ответа имеет отдельные неточности (несущественные ошибки) в изложении теоретического и практического материала, отличается меньшей обстоятельностью, глубиной и полнотой; допущенные ошибки исправляются студентом после дополнительных вопросов экзаменатора.

«Зачтено / Удовлетворительно» («3») – студент обнаруживает знание и понимание основных положений учебного материала, но излагает его неполно, непоследовательно, допускает неточности и существенные ошибки в определении понятий, формулировке положений, не привлекает для аргументации ответа основные положения исследовательских, концептуальных и нормативных документов, не умеет обосновать свои суждения; наблюдается нарушение логики изложения. Ответ отличается низким уровнем самостоятельности, не содержит собственной профессионально-личностной позиции.

«Не зачтено / Неудовлетворительно» («2») – студент имеет разрозненные, бессистемные знания: не умеет выделять главное и второстепенное; допускает ошибки в определении понятий, формулировке теоретических положений, искажающие их смысл; не ориентируется в нормативно-концептуальных, программно-методических, исследовательских материалах, беспорядочно и неуверенно излагает материал; не умеет соединять теоретические положения с педагогической практикой; не умеет применять знания для обоснования и объяснения фактов.

Особенности проведения процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья обозначены в рабочей программе дисциплины (модуля).

Уровни сформированности компетенций определяются следующим образом:

1. Высокий уровень сформированности компетенций соответствует оценке отлично:

- предполагает формирование компетенций на высоком уровне, готовность к самостоятельной профессиональной деятельности: формируются навыки практическим опытом разработки мобильных приложений с учётом требований информационной безопасности;
- студент способен разрабатывать мобильные приложения с учётом требований информационной безопасности.

2. Средний уровень соответствует оценке хорошо:

- предполагает формирование компетенций на среднем уровне: формируется комплексное знание классификации мобильных устройств и программных платформ для создания мобильных приложений;



- студент способен разрабатывать мобильные приложения с учётом требований информационной безопасности на уровне не ниже оценки «хорошо».
3. Базовый уровень соответствует оценке удовлетворительно:
- предполагает формирование компетенций на начальном уровне: знание и понимание основных понятий мобильных устройств и программных платформ для создания мобильных приложений;
  - студент способен отвечать на вопросы, выполнять типовые практические задания по образцу.
4. Низкий уровень соответствует оценке неудовлетворительно.

**01.04.02 Прикладная математика и информатика, Направленность (профиль)  
Математическое моделирование и искусственный интеллект, ФОС  
Современные компьютерные технологии, 2026 г.н., очная форма обучения.**

**Фонд оценочных средств дисциплины (модуля) одобрен и рекомендован:**

Проректор по учебной работе утверждено 27.02.26

А.А. Саламатов

Ученым советом математического факультета

Протокол заседания № 8 от 19.02.2026

Председатель Ученого совета  
математического факультета

согласовано

А. С. Скорынин

**Заседанием кафедры вычислительной механики и информационных технологий**

Протокол заседания № 7 от 18.02.2026

Заведующий кафедрой

согласовано

М.В. Плеханова

Автор (составитель)

Е.М. Ижбердеева

**Структура фондов оценочных средств соответствует приказу ректора ФГБОУ ВО «ЧелГУ»  
от 27 сентября 2022 №573-1**