

Министерство науки и высшего образования Российской Федерации
Нижекамский химико-технологический институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
«Казанский национальный исследовательский технологический университет»
(НХТИ ФГБОУ ВО «КНИТУ»)

УТВЕРЖДАЮ



Заместитель директора по УР

Н.И. Никифорова

« 12 » 04 2021 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

Б1.В.14 «Технологии распределенных систем»
(наименование дисциплины (модуля))

09.03.02 «Информационные системы и технологии»
(код и наименование направления подготовки/ специальности)

Системы информационной безопасности
(наименование профиля)

бакалавр
квалификация

очная
форма обучения

Нижнекамск, 2021


Составитель ФОС:
доцент кафедры ИСТ


(подпись)

О.В. Матухина


ФОС рассмотрен и одобрен на заседании кафедры ИСТ,
протокол от 15.03.2021 г. № 7

Зав. кафедрой


(подпись)

О.В.Матухина
(Ф.И.О.)

Эксперт:

Руководитель ООП, доцент ИСТ НХТИ ФГБОУ ВО «КНИТУ»  Л.Р. Вотякова
Ф.И.О., должность, организация, подпись

Перечень компетенций и индикаторов достижения компетенций с указанием этапов формирования в процессе освоения дисциплины

Компетенция:

ПК-2. Способен обеспечить информационную безопасность на уровне баз данных

Индикаторы достижения компетенции:

ПК-2.1 Знает угрозы безопасности баз данных, способы предотвращения

ПК-2.2. Умеет выявлять угрозы безопасности на уровне баз данных

ПК-2.3. Владеет навыками применения способов предотвращения угроз безопасности на уровне баз данных

Компетенция:

ПК-3. Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы

Индикаторы достижения компетенции:

ПК-3.1. Знает инструменты и методы проектирования архитектуры ИС, устройство, функционирование вычислительных систем и современных ИС, автоматизирующих задачи организационного управления и бизнес-процессы

ПК-3.2. Умеет проектировать архитектуру ИС, анализировать входную информацию, разрабатывать структуру баз данных, автоматизирующих задачи организационного управления и бизнес-процессы

ПК-3.3. Владеет навыками проектирования архитектуры ИС, структуры баз данных, работы современных ИС, автоматизирующих задачи организационного управления и бизнес-процессы

Компетенция:

ПК-4. Способен администрировать сетевую подсистему инфокоммуникационной системы организации

Индикаторы достижения компетенции:

ПК-4.1. Знает общие принципы функционирования аппаратных, программных и программно-аппаратных средств администрируемой сети

ПК-4.2. Умеет использовать современные средства администрирования баз данных

ПК-4.3. Владеет навыками администрирования сетевой системы и программного обеспечения инфокоммуникационной системы.

Индикаторы достижения компетенции	Этапы формирования в процессе освоения дисциплины				Наименование оце- ночного средства
	Лекции	Практические занятия	Лабораторные занятия	Курсовой проект (работа)	
ПК-2.1	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-2.2	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-2.3	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-3.1	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-3.2	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-3.3	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-4.1	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-4.2	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы
ПК-4.3	Разделы дис- циплины 1-2.	Не предусмотрен учебным планом	Разделы дисциплины 1-2.	Не предусмотрен учебным планом	Расчетно-графическая работа, лабораторные работы

Перечень оценочных средств по дисциплине

Оценочные средства	Кол-во	Min, баллов (базовый уровень)	Max, баллов (повышенный уровень)
Лабораторные работы	2	36	60
Расчетно-графические работы	1	24	40

Шкала оценивания

Цифровое выражение	Выражение в баллах:	Словесное выражение	Критерии оценки индикаторов достижения при форме контроля:
			Зачет с оценкой
5	87 - 100	Отлично	Оценка «отлично» выставляется студенту, если теоретическое содержание курса освоено полностью, без пробелов; исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует в ответе дополнительный материал все предусмотренные программой задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному; анализирует полученные результаты; проявляет самостоятельность при выполнении заданий
4	74 - 86	Хорошо (Оценка «хорошо» выставляется студенту, если теоретическое содержание курса освоено полностью, необходимые практические компетенции в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос.
3	60 - 73	Удовлетворительно	Оценка «удовлетворительно» выставляется студенту, если теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, большинство предусмотренных программой заданий выполнены, но в них имеются ошибки, при ответе на поставленный вопрос студент допускает неточности, недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении программного материала.
2	Ниже 60	Неудовлетворительно	Оценка «неудовлетворительно» выставляется студенту, если он не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, необходимые практические компетенции не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, качество их выполнения оценено числом баллов, близким к минимальному

Краткая характеристика оценочных средства

<i>№ п/п</i>	<i>Наименование оценочного средства</i>	<i>Краткая характеристика оценочного средства</i>	<i>Представление оценочного сред- ства в фонде</i>
<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>
1.	Лабораторная работа	Это вид учебной работы, целью которой является изучение (исследование, измерение) характеристик лабораторного объекта. Цель лабораторных занятий: освоение изучаемой учебной дисциплины; приобретение навыков практического применения знаний учебной дисциплины (дисциплин) с использованием технических средств и (или) оборудования	Темы лабораторных работ, контрольные вопросы по теме лабораторной работы
2.	Расчетно-графическая работа	Средство проверки умений применять полученные знания по заранее определенной методике для решения задач или заданий по модулю или дисциплине в целом.	Комплект заданий для выполнения расчетно-графической работы

Министерство науки и высшего образования Российской Федерации
Нижекамский химико-технологический институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
«Казанский национальный исследовательский технологический университет»

Факультет информационных технологий
Кафедра информационных систем и технологий
Направление подготовки 09.03.02 «Информационные системы и технологии»

Учебным планом по направлению подготовки 09.03.02 «Информационные системы и технологии» для обучающихся предусмотрено проведение лабораторных работ по дисциплине «Технологии распределенных систем».

Лабораторная работа 1

Параллельное программирование для систем с общей памятью с использованием технологии OpenMP.

Цель работы: –изучить технологию OpenMP и основы разработки параллельных программ для многоядерных процессоров, написать и отладить на языке C параллельную программу для решения поставленной задачи на системе с общей памятью.

1.1. Основные сведения об OpenMP

Технология OpenMP [10, 11] в настоящее время является одним из наиболее популярных средств параллельного программирования для компьютеров с общей памятью, базирующейся на традиционных последовательных языках программирования и использовании специальных комментариев. За основу берётся последовательная программа, а для создания её параллельной версии пользователю предоставляется набор директив, функций и переменных окружения. Технология OpenMP нацелена на то, чтобы пользователь имел один и тот же вариант программы как для параллельного, так и для последовательного режима выполнения. Параллелизм в OpenMP реализуется с помощью многопоточности. При запуске программы создается единственный «главный» (master) поток, который затем создает набор «подчиненных» (slave) потоков, и вычисления распределяются между всеми потоками. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (ядрами), причём количество процессоров/ядер не обязательно должно быть больше или равно количеству потоков. Другими словами, потоки параллельной программы могут обычным образом конкурировать между собой за время процессора/ядра.

Важным достоинством технологии OpenMP является возможность реализации так называемого инкрементального программирования, когда программист постепенно находит участки в программе, содержащие ресурс параллелизма, с помощью предоставляемых механизмов делает их параллельными, а затем переходит к анализу следующих участков. Таким образом, распараллеленные участки постепенно охватывают всё большую часть программы. Этот подход значительно облегчает процесс адаптации последовательных программ к параллельным компьютерам, а также отладку и оптимизацию программ.

Модель исполнения параллельной программы, подготовленной с помощью технологии OpenMP, можно сформулировать следующим образом:

- Программа содержит набор последовательных и параллельных областей (или секций или регионов).
- В начальный момент времени создается главный поток, выполняющий впоследствии все последовательные области программы.
- При входе в параллельную область главным потоком выполняется операция fork, порождающая совокупность подчиненных потоков. Каждый поток имеет свой уникальный числовой идентификатор (главному потоку соответствует 0). При распараллеливании циклов все параллельные потоки исполняют один и тот же код, но с разными данными. В общем случае потоки могут исполнять различные фрагменты кода.

- При выходе из параллельной области всеми потоками выполняется операция join. Завершается выполнение всех потоков, кроме главного.

На рис. 1. проиллюстрировано создание и исполнение двух параллельных областей. В первой области все потоки приходят к моменту выхода из нее практически одновременно, во второй – в разные моменты времени (как правило, это более соответствует реальной картине).

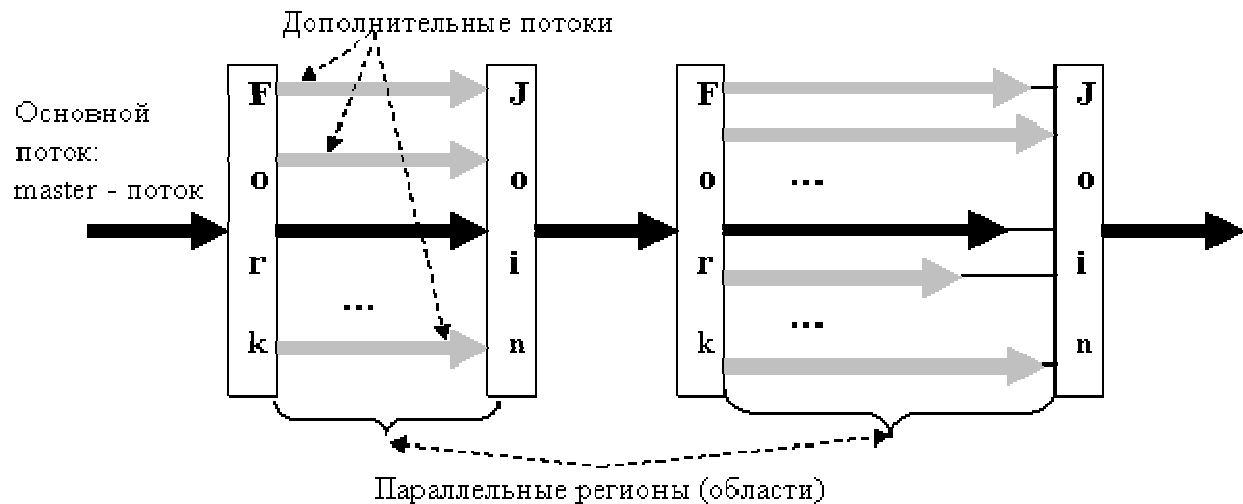


Рис. 1. Создание двух параллельных регионов

Обычно допускается возможность образования вложенных параллельных областей, когда поток, созданный как дополнительный, становится master-потоком для новой параллельной области, как показано на рис. 2.

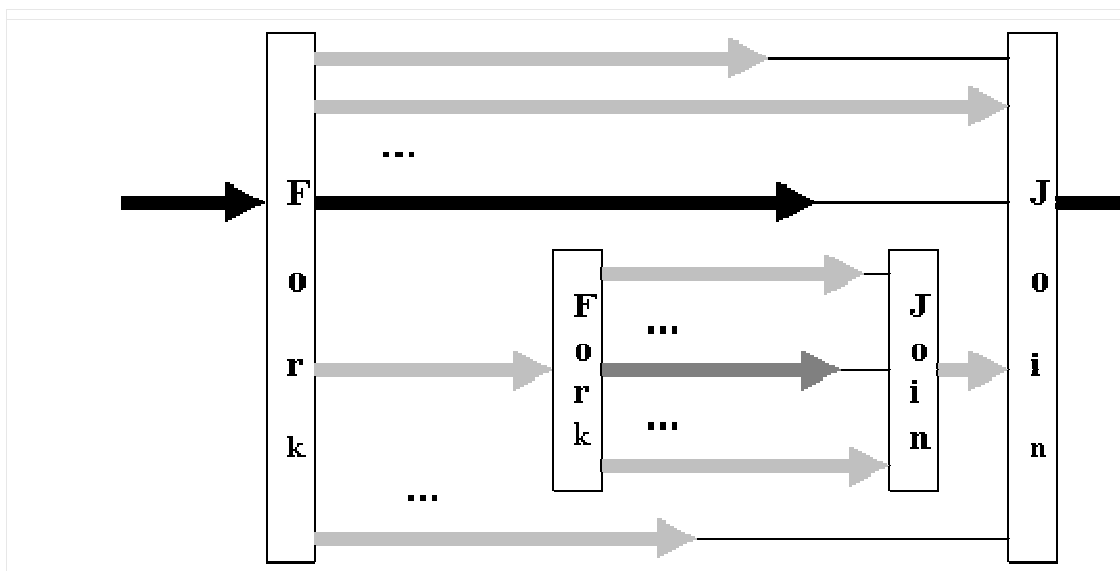


Рис. 2. Создание вложенных параллельных регионов

Под OpenMP понимается совокупность следующих компонент:

- Директивы компилятора – используются для создания потоков, распределения работы между пото-

ками и их синхронизации. Директивы включаются программистом в исходный текст программы.

- Подпрограммы библиотеки времени выполнения – используются для установки и определения атрибутов потоков. Вызовы этих подпрограмм включаются программистом в исходный текст.

- Переменные окружения – используются для управления поведением параллельной программы. Переменные окружения задаются для среды выполнения параллельной программы соответствующими средствами операционной системы.

Использование директив компилятора и подпрограмм библиотеки времени выполнения подчиняется правилам, которые различаются для разных языков программирования. Совокупность таких правил для одного языка программирования называется привязкой к языку. Технология OpenMP создана и поддерживается для языков C, C++ и Fortran. В этой лабораторной работе предполагается использование языка C/C++ в среде разработки [Microsoft Visual Studio](#) (начиная с версии 2005). Для того, чтобы ее компилятор нужным образом реагировал на директивы OpenMP и строил параллельную программу, в командную строку его запуска должна быть включена опция /openmp (например, путем включения флажка «OpenMPsupport» в свойствах проекта на закладке Configurationproperties\C/C++\Language).

Для того, чтобы в программе на языке C/C++ стали доступны возможности технологии OpenMP, в нее нужно включить заголовочный файл omp.h:

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

Если эту программу транслировать компилятором, не поддерживающим технологию OpenMP, то она будет построена в обычном последовательном (однопоточном) варианте, поскольку все директивы, задающие распараллеливание, оформляются в виде так называемых прагм (рекомендаций), и просто игнорируются такими компиляторами.

В программах на языке C/C++ все прагмы, имена функций и переменных окружения OpenMP начинаются со строки «omp». Формат директивы:

```
#pragma omp директива [опция_1[, опция_2, ...]]
```

Каждая директива вместе со всеми ее опциями обязательно должна занимать ровно одну строку текста. Действие некоторых директив распространяется только на один оператор (или блок операторов, заключенных в фигурные скобки), непосредственно следующий за директивой в тексте программы. Для таких директив будет указан

<структурный блок кода>.

Перечень **директив OpenMP** включает в себя:

1. Директива задания параллельно выполняемой секции:

```
#pragma omp parallel [опция_1[, опция_2, ...]]  
<структурный блок кода>
```

С помощью опций этой директивы можно указать:

- требуемое количество потоков n (*num_threads(n)*);
- условие, при котором параллельная область действительно создается (*if(условие)*);
- список общих переменных для всех потоков данной секции (*shared(список переменных)*);

- список переменных, которые будут локальными в каждом потоке секции (*private(список переменных)*), причем их начальные значения не будут определены;
- список переменных, которые будут локальными в каждом потоке секции (*firstprivate(список переменных)*), причем в качестве их начальных значений будут установлены значения одноименных переменных из главного потока;
- способ назначения класса памяти *default(shared|none)* всем переменным потоков, которым класс не назначен явно с помощью опции *shared* (слово *none* означает, что класс памяти всех локальных переменных должен быть задан явно); в реализациях для языка Fortran могут назначаться классы *private* и *firstprivate*;
- список переменных, объявленных директивой *threadprivate* (см. ниже), которые при входе в параллельную секцию инициализируются значениями соответствующих переменных в потоке-мастере;
- оператор сведения и список общих переменных *reduction(оператор : список переменных)*; для каждой указанной в списке переменной создаются локальные копии в каждом потоке; локальные копии инициализируются соответственно типу оператора (для аддитивных операций ноль или его аналоги, для мультипликативных операций единица или её аналоги); над всеми локальными копиями каждой переменной после завершения параллельной секции будет выполнен заданный оператор сведения, результаты будут занесены в одноименные общие переменные; в качестве оператора можно указывать: +, −, *, &, |, ^, &&, ||.

2. Директива определения цикла, итерации которого нужно распределить между параллельно выполняемыми потоками:

```
#pragma omp for [опция_1[, опция_2, ...]]
<структурный блок кода>
```

С использованием опций директивы *for* можно указать:

- список переменных, которые будут локальными в каждом потоке секции (*private(список переменных)*), причем их начальные значения не будут определены;
 - список переменных, которые будут локальными в каждом потоке секции (*firstprivate(список переменных)*), причем в качестве их начальных значений будут установлены значения одноименных переменных из главного потока;
 - список переменных главного потока (*lastprivate(список переменных)*), которым будут присвоены значения, полученные при выполнении последней итерации цикла;
 - оператор сведения и список общих переменных *reduction(оператор : список переменных)*; для каждой указанной в списке переменной создаются локальные копии в каждом потоке; локальные копии инициализируются соответственно типу оператора (для аддитивных операций ноль или его аналоги, для мультипликативных операций единица или её аналоги); над всеми локальными копиями каждой переменной после завершения параллельной секции будет выполнен заданный оператор сведения, результаты будут занесены в одноименные общие переменные; в качестве оператора можно указывать: +, −, *, &, |, ^, &&, ||;
 - способ распределения итераций цикла между потоками параллельной секции (*schedule(type[, chunk])*); параметр *chunk* этой опции определяет количество итераций на один поток (по умолчанию 1), параметр *type* указывает тип распределения и может иметь значения:
 - § *static* – статический, т. е. при компиляции,
 - § *dynamic* – динамический, т. е. при выполнении,
 - § *guided* – динамический с уменьшением количества итераций на поток от начального, определяемого автоматически, до значения *chunk*,
 - § *auto* – способ распределения выбирается компилятором или исполняющей системой,
 - § *runtime* – способ распределения задается специальной переменной окружения операционной системы;
 - возможность (опция *ordered*) появления в теле цикла директивы *ordered*, которая требует исполнения охваченного ею блока операторов в точности в той же последовательности, какая реализуется в последовательной версии данного цикла;
 - отмену (*nowait*) неявной барьерной синхронизации потоков, достигших конца выполнения своей части итераций цикла (при отсутствии этой опции участок программы после цикла будет выполняться только тогда, когда все потоки выполнят все свои итерации);
 - глубину *n* вложенных друг в друга циклов (*collapse(n)*), пространство итераций которых подлежит распределению между параллельными потоками (доступно только в реализации 2.5 технологии OpenMP);
- Директивы *parallel* и *for* можно объединять в одну директиву, в которой можно указывать опции как директивы *parallel*, так и директивы *for*:
- ```
#pragma omp parallel for [опция_1[, опция_2, ...]]
```

3. Директива, указывающая на необходимость исполнения охватываемого ею участка кода (части тела цикла) в той последовательности, в которой он выполняется в чисто последовательном режиме

*#pragma omp ordered*

*<структурный блок кода>*

4. Директива задания области нециклического параллелизма (участки структурного блока кода, которые могут выполняться параллельно, выделяются с помощью описываемой далее директивы *section*):

*#pragma omp sections [опция\_1[, опция\_2, ...]]*

*<структурный блок кода>*

В качестве опций этой директивы можно указывать *private*, *firstprivate*, *lastprivate*, *reduction* и *nowait*, имеющие в точности такой же синтаксис и тот же смысл, что и у директивы *for*.

5. Директива определения участка нециклического кода для одного потока:

*#pragma omp section*

С помощью этой директивы внутри участка кода, охваченного директивой *sections*, выделяются отдельные фрагменты для исполнения параллельными потоками. Перед самым первым фрагментом участка нециклического кода директиву *section* можно не указывать.

6. Директива объявления списка локальных переменных потоков

*#pragma omp threadprivate(список переменных)*

Эта директива позволяет сделать локальные копии для статических переменных языка C/C++ (и COMMON-блоков языка Фортран), которые по умолчанию являются общими.

7. Директива создания отдельной независимой задачи (начиная с версии 2.5 OpenMP):

*#pragma omp task [опция\_1[, опция\_2, ...]]*

*<структурный блок кода>*

Текущий поток создает в качестве задачи ассоциированный с директивой блок операторов. Эта задача может выполняться немедленно после создания или быть отложенной на неопределённое время и выполняться по частям. Размер таких частей, а также порядок выполнения частей разных отложенных задач определяется реализацией OpenMP.

### **Лабораторная работа 3**

Распараллеливание вычислений при численном решении аналитических задач.

**Цель работы:** –изучить методы и алгоритмы разработки параллельных программ для решения задач численного дифференцирования, интегрирования, моделирования.

В лабораторной работе нужно разработать параллельную программу для решения задач:

1. Численное дифференцирование.
2. Численное интегрирование.
3. Численное моделирование.

Методы и алгоритмы для решения выбирается студентом самостоятельно.

#### **Критерии оценки**

| Вид контроля          | Минимальное количество баллов | Максимальное количество баллов |
|-----------------------|-------------------------------|--------------------------------|
| Лабораторная работа 1 | 18                            | 30                             |
| Лабораторная работа 2 | 18                            | 30                             |
| <b>Итого</b>          | <b>36</b>                     | <b>60</b>                      |

Министерство науки и высшего образования Российской Федерации  
Нижекамский химико-технологический институт (филиал)  
федерального государственного бюджетного образовательного учреждения  
высшего образования  
«Казанский национальный исследовательский технологический университет»

Факультет информационных технологий  
Кафедра информационных систем и технологий  
Направление подготовки 09.03.02 «Информационные системы и технологии»

Комплект заданий для выполнения расчетно-графических работ  
по дисциплине «Технологии распределенных систем»

Реализовать на языке высоко уровня:

1. Гибридную архитектуру NUMA.
2. PVP- архитектуру.

***Критерии оценки***

| Вид контроля                | Минимальное количество баллов | Максимальное количество баллов |
|-----------------------------|-------------------------------|--------------------------------|
| Расчетно-графическая работа | 24                            | 40                             |
| <b>Итого</b>                | <b>24</b>                     | <b>40</b>                      |