МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой
ПиИТ
/Махортов С.Д./
05.03.2024

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.ДВ.03.02 Программирование с использованием STL

1. Код и наименование направления подготовки/специальности:
09.03.04 Программная инженерия

2. Профиль подготовки/специализация:
Информационные системы и сетевые технологии

3. Квалификация выпускника: бакалавр

4. Форма обучения: очная

5. Кафедра, отвечающая за реализацию дисциплины:
Программирования и информационных технологий (ПиИТ)

6. Составители программы: Махортов Сергей Дмитриевич, д.ф.-м.н., доцент

7. Рекомендована НМС ФКН, протокол № 5 от 05.03.2024.

_____
отметки о продлении вносятся вручную)

_____

8. Учебный год: 2026 / 2027 **Семестр(ы):** 6

9.**Цели и задачи учебной дисциплины:**

- изложить теоретические и практические основы применения стандартной библиотеки шаблонов (STL) в языке программирования C++ для разработки эффективных и масштабируемых программных решений, охватывая ключевые аспекты контейнеров, алгоритмов и итераторов;
- рассмотреть и проанализировать типовые задачи, связанные с управлением, сортировкой и фильтрацией данных, а также обработкой больших последовательностей, используя возможности STL для достижения оптимальных результатов;
- развить у студентов навыки профессионального проектирования алгоритмов и структур данных с применением STL, а также научить их оценивать временную и пространственную сложность алгоритмов с целью дальнейшей оптимизации кода и его адаптации к конкретным условиям предметной области;
- сформировать у студентов компетенции по интеграции STL в программные проекты, обеспечивая высокую производительность и минимальное использование ресурсов;
- выработать способности и мотивацию к решению сложных и нестандартных задач, связанных с программной инженерией, а также способствовать приобретению практических навыков разработки, тестирования и оптимизации программного кода, применяя современные подходы и методологии разработки.

10. Место учебной дисциплины в структуре ООП: Часть, формируемая участниками образовательных отношений (вариативная) блока Б1**.** Требуется предварительное знание основ дискретной математики и теории алгоритмов, программирования. Предшествует дисциплинам: производственная практика, научно-исследовательская работа.

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями выпускников):

Код	Название компетенции	Код(ы)	Индикатор(ы)	Планируемые результаты обучения
ПК-9	Способен описывать алгоритмы компонентов системы, включая методы и схемы.	ПК-9.1 ПК-9.2	Описывает применяемые математические методы, допущения и ограничения, связанные с выбранным математическим материалом. Описывает алгоритмы и (или) функционирование программы с обоснованием выбора схем алгорит-мов решения задач, возможных взаимодействий программы с	Знать: ключевые компоненты и принципы функционирования стандартных библиотек C++, используемых для работы с данными, включая структуры хранения, алгоритмы обработки и механизмы управления последовательностями. Уметь: разрабатывать и оптимизировать программные решения с применением встроенных шаблонов и алгоритмов, эффективно решая задачи сортировки, поиска и обработки данных, а также оценивать их производительность и ресурсоемкость. Владеть: методами анализа сложности программных решений, навыками проектирования гибких и масштабируемых структур данных, а также практическими инструментами для реализации высокоэффективных алгоритмов.

			другими программами.	

12. Объем дисциплины в зачетных единицах/час.(*в соответствии с учебным планом)* — 3 / 108.

Форма промежуточной аттестации *(зачет/экзамен)* зачет с оценкой.

13. Трудоемкость по видам учебной работы

Вид учебной работы		Трудоемкость			
		Всего	По семестрам		
			№ 6	№ семестра	…
Аудиторные занятия		64	64		
в том числе:	лекции	32	32		
	практические	16	16		
	лабораторные	16	16		
Самостоятельная работа		44	44		
в том числе: курсовая работа (проект)		-	-		
Форма промежуточной аттестации *(зачет с оценкой – 0 час.)*		-	-		
Итого:		108	108		

13.1. Содержание дисциплины

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины
1. Лекции		
1.1	Введение в стандартную библиотеку шаблонов (STL)	Обзор архитектуры и компонентов STL: контейнеры, итераторы, алгоритмы. Рекомендации по выбору контейнеров для различных типов задач. Основы управления памятью и ресурсов.
1.2	Контейнеры последовательного доступа и их применение	Подробное изучение vector, deque, list: их структура и методы, управление памятью, примеры использования в задачах различной сложности. Оценка производительности в зависимости от типа контейнера.
1.3	Ассоциативные контейнеры и их применение	Детальный анализ контейнеров set, map, multiset, multimap: принципы работы, особенности реализации, методы вставки, удаления и поиска элементов. Применение ассоциативных контейнеров для задач обработки данных.
1.4	Алгоритмы STL: реализация и оптимизация	Введение в алгоритмы STL: сортировка (sort, stable_sort), поиск (find, binary_search), модификация (transform, remove_if). Оптимизация алгоритмов и оценка их временной сложности.
1.5	Итераторы STL	Типы итераторов: прямые, обратные, итераторы вставки и их использование. Примеры создания и применения пользовательских итераторов для работы с комплексными структурами данных.
2. Практические занятия		
2.1	Контейнеры STL и работа с итераторами	Изучение и реализация на практике различных типов контейнеров и итераторов. Решение задач по эффективному доступу и управлению данными с использованием итераторов STL.
2.2	Продвинутая работа с ассоциативными контейнерами	Оптимизация поиска данных и работы с деревьями (red-black trees) через ассоциативные контейнеры STL. Сравнение производительности различных подходов.
2.3	Алгоритмы STL: теория и практика применения	Применение алгоритмов STL для решения задач сортировки и фильтрации данных. Сравнение стандартных алгоритмов с пользовательскими реализациями.
2.4	Применение алгоритмов STLс использованием STL	Практическое задание по сортировке данных и поиску элементов в массивах с использованием алгоритмов STL. Оценка производительности стандартных алгоритмов и их оптимизация для конкретных задач.

2.5	Итераторы и функциональные объекты	Задачи по созданию и применению пользовательских итераторов для обхода структур данных. Разработка функциональных объектов и их использование в алгоритмах STL.
3. Лабораторные работы		
3.1	Введение в использование STL	Разработка простых программ, демонстрирующих работу с базовыми компонентами STL: создание и заполнение контейнеров, использование базовых итераторов.
3.2	Контейнеры последовательного доступа	Реализация задач с использованием vector, deque и list для хранения и обработки данных различного типа. Сравнение производительности и профилирование использования памяти для разных сценариев.
3.3	Ассоциативные контейнеры в задачах обработки данных	Разработка приложений, использующих контейнеры map и set для построения словарей и реализации операций поиска и сортировки. Оценка эффективности операций и сравнение с последовательными контейнерами.
3.4	Реализация и оптимизация алгоритмов STL	Применение стандартных алгоритмов STL для обработки данных: сортировка, поиск и фильтрация элементов. Оптимизация алгоритмов для повышения производительности и анализ временной сложности.
3.5	Итераторы и их использование в проектных задачах	Создание и тестирование пользовательских итераторов для работы с комплексными структурами данных. Примеры использования итераторов для обхода и модификации элементов контейнеров.

13.2. Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1	Введение в стандартную библиотеку шаблонов (STL)	4	2	2	6	14
2	Контейнеры последовательного доступа и их применение	8	4	4	8	24
3	Ассоциативные контейнеры и их применение	8	4	4	10	26
4	Алгоритмы STL: реализация и оптимизация	8	4	4	10	26
5	Итераторы STL	4	2	2	10	28
	Итого:	32	16	16	44	108

14. Методические указания для обучающихся по освоению дисциплины

(рекомендации обучающимся по освоению дисциплины: работа с конспектами лекций, презентационным материалом, выполнение практических заданий, тестов, заданий текущей аттестации и т.д.)

Работа с конспектами лекций и презентационным материалом; выполнение практических заданий и тестов; выполнение лабораторных заданий; подготовка к заданиям текущей аттестации.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимых для освоения дисциплины

(список литературы оформляется в соответствии с требованиями ГОСТ и используется общая сквозная нумерация для всех видов источников)

а) основная литература:

№ п/п	Источник
1	Ашарина, И.В. Объектно-ориентированное программирование в C++: лекции и упражнения: учебное пособие – М.: Горячая линия - Телеком, 2017. – 336 с.
2	Лаптев В.В. C++. Объектно-ориентированное программирование: [учебное пособие] – СПб: Питер, 2008. – 457 с.
3	Москвин П.В. Азбука STL. – М.: Горячая линия - Телеком, 2003. – 262 с.
4	Аммерааль, Л. STL для программистов на C+. – М.: ДМК, 1999. – 239 с.
5	Мейерс, С. Эффективное использование STL / Пер. с англ. – СПб: Питер, 2002. – 224 с.
6	Подбельский В.В. Стандартный Си++: [учебное пособие для студ. вузов., обуч. по направлению подготовки 230400 "Прикладная математика"]. – М.: Финансы и статистика, 2008. – 687 с.
7	Шилдт Г. C++: базовый курс / Пер. с англ. 3-е изд. – М.: Вильямс, 2015. – 620 с.
8	Страуструп Б. Язык программирования C++ = The C++ Programming Language: Специальное издание / Пер. с англ. – М.: Бином-Пресс, 2008. – 1098 с.

9	Паппас К.Х. Отладка в С+: Рук. для разработчиков / Пер. с англ. – М.: Бином, 2001. – 509 с.
10	STL. Стандартная библиотека шаблонов С++. Пер. с англ. – СПб.: БХВ-Петербург, 2004. – 656 с.

б) дополнительная литература:

№ п/п	Источник

в) информационные электронно-образовательные ресурсы (официальные ресурсы интернет)*:

№ п/п	Ресурс

* Вначале указываются ЭБС, с которыми имеются договора у ВГУ, затем открытые электронно-образовательные ресурсы

16. Перечень учебно-методического обеспечения для самостоятельной работы

(учебно-методические рекомендации, пособия, задачники, методические указания по выполнению практических (контрольных) работ и др.)

№ п/п	Источник

17. Образовательные технологии, используемые при реализации учебной дисциплины, включая дистанционные образовательные технологии (ДОТ), электронное обучение (ЭО), смешанное обучение)

Для реализации учебного процесса применяются: современные инструментальные системы Visual Studio Code, JetBrains CLion, PyCharm Community, Microsoft Visual Studio Community; ресурс «Электронный университет» (https://edu.vsu.ru/).

18. Материально-техническое обеспечение дисциплины

1. Компьютерный класс №5 (ауд. 295). ПК-Intel-Core2 14 шт., рабочее место преподавателя: проектор, видеокоммутатор, специализированная мебель: доска маркерная 1 шт., столы 16 шт., стулья 33 шт. В классе находится точка доступа беспроводной сети для доступа в Интернет и к учебно-методическим материалам, расположенным на внутренних серверах факультета.
2. Компьютерный класс №7 (ауд. 316п). ПК на базе IntelCore2Duo 2,8ГГц, ОЗУ 2ГБ, диск 160Gb – 30 шт. Специализированная мебель: доска маркерная 1 шт., доска интерактивная 1 шт., столы 32 шт., стулья 64 шт.; рабочее место преподавателя: проектор, видеокоммутатор. В классе находится точка доступа беспроводной сети для доступа в Интернет и к учебно-методическим материалам, расположенным на внутренних серверах факультета.

19. Оценочные средства для проведения текущего контроля и промежуточной аттестаций

19.1. Текущий контроль

Текущая аттестация проводится в соответствии с Положением о текущей аттестации обучающихся по программам высшего образования Воронежского государственного университета. Текущая аттестация проводится в форме письменных работ (контрольные, выполнение практико-ориентированных заданий). Критерии оценивания приведены ниже.

19.1.1 Задания для текущего контроля успеваемости

Приведённые ниже задания рекомендуется использовать при проведении диагностических работ для оценки остаточных знаний по дисциплине

Тестовые задания – 1 балл

1. Какие контейнеры относятся к последовательным контейнерам в STL?
   1) `map`, `set`, `multimap`
   2) `vector`, `deque`, `list`
   3) `stack`, `queue`

2. Какую сложность имеет операция вставки элемента в контейнер `vector` в худшем случае?
   1) O(1)
   2) O(n)
   3) O(log n)

3. Какой ассоциативный контейнер STL лучше всего подходит для реализации словаря?
   1) `vector`
   2) `map`
   3) `deque`

4. Что делает функция `sort` в STL?
   1) Сортирует элементы по возрастанию
   2) Удаляет дубликаты элементов
   3) Находит максимальное значение в контейнере

5. Какая сложность алгоритма `binary_search` в STL?
   1) O(n)
   2) O(log n)
   3) O(n^2)

6. Какой из контейнеров в STL поддерживает быстрый доступ к элементам по индексу?
   1) `vector`
   2) `set`
   3) `list`

7. Какой итератор используется для изменения элементов контейнера в STL?
   1) Константный итератор
   2) Прямой итератор
   3) Обратный итератор

8. Какое свойство имеет ассоциативный контейнер `set` в STL?
   1) Содержит только уникальные элементы
   2) Хранит элементы в произвольном порядке
   3) Поддерживает дублирование элементов

9. Какая функция в STL используется для удаления всех элементов, удовлетворяющих определенному условию?
   1) `remove`
   2) `erase`
   3) `remove_if`

10. Какой контейнер лучше всего использовать для реализации стека в STL?
   1) `vector`

2) `stack`
3) `queue`

11. Какой контейнер STL автоматически упорядочивает элементы при добавлении?
1) `list`
2) `set`
3) `vector`

12. Какое преимущество имеет `unordered_map` перед `map`?
1) Поддерживает порядок вставки
2) Быстрая вставка и доступ к элементам
3) Хранит элементы в отсортированном виде

13. Какой алгоритм в STL можно использовать для выполнения поиска минимального элемента в контейнере?
1) `find`
2) `min_element`
3) `accumulate`

14. Какую функцию можно использовать для применения лямбда-выражения к каждому элементу контейнера?
1) `transform`
2) `for_each`
3) `find_if`

15. Какой контейнер STL подходит для реализации очереди с приоритетом?
1) `vector`
2) `queue`
3) `priority_queue`

16. В каком случае использование `deque` предпочтительнее `vector`?
1) При частом доступе к элементам по индексу
2) При частых вставках и удалениях с обоих концов
3) При необходимости хранения уникальных значений

17. Какая функция STL создает итератор для обратного прохода по контейнеру?
1) `begin()`
2) `rbegin()`
3) `rend()`

18. Какой из контейнеров в STL не поддерживает последовательный доступ к элементам?
1) `vector`
2) `list`
3) `unordered_map`

19. Какая функция используется для получения количества элементов в контейнере?
1) `size()`
2) `length()`
3) `count()`

20. Какой контейнер в STL лучше всего использовать для реализации двусторонней очереди?
    1) `list`
    2) `deque`
    3) `vector`

Номер вопроса	Ответ
1	2
2	2
3	2
4	1
5	2
6	1
7	2
8	1
9	3
10	2
11	2
12	2
13	2
14	2
15	3
16	2
17	2
18	3
19	1
20	2

Задания с кратким ответом – 2 балла

1. Какая вычислительная сложность считается приемлемой для алгоритмов работы с контейнерами STL при обработке больших данных?
2. Какова длина пустого контейнера vector?
3. Какова вычислительная сложность вставки элемента в конец vector в среднем случае?
4. Сколько уникальных элементов может содержать контейнер set с добавлением дубликатов?
5. Какова вычислительная сложность поиска элемента в map?
6. Какова вычислительная сложность вставки элемента в ассоциативный контейнер unordered_map?
7. Какова вычислительная сложность удаления элемента из vector по индексу?
8. Какова вычислительная сложность удаления всех элементов, соответствующих условию, из контейнера list?
9. Какова вычислительная сложность алгоритма сортировки sort в STL?
10. Какой элемент подвергается преобразованию в алгоритме transform?
11. Какой из алгоритмов STL имеет сублинейное среднее время работы при поиске элемента?
12. Какой алгоритм в STL может работать в режиме "online" (обрабатывая данные по мере их поступления)?
13. Какие алгоритмы STL можно отнести к категории вероятностных?
14. Какой из алгоритмов сортировки в STL имеет вероятностное обоснование эффективности?
15. Какова вычислительная сложность построения дерева на основе контейнера set?
16. Какова вычислительная сложность выполнения вставки в priority_queue?
17. Какова вычислительная сложность удаления минимального элемента из priority_queue?
18. Какова вычислительная сложность алгоритма объединения двух отсортированных контейнеров с использованием merge?
19. Какова вычислительная сложность алгоритма nth_element в STL?
20. Какова вычислительная сложность алгоритма поиска минимального элемента (min_element) в контейнере?

Номер вопроса	Ответ
1	Линейная (O(n))
2	0
3	Амортизированная O(1)
4	Количество уникальных элементов (все дубликаты отбрасываются)
5	Логарифмическая (O(log n))
6	Постоянная (O(1)) в среднем случае
7	Линейная (O(n))
8	Линейная (O(n))
9	Логарифмическая (O(log n))
10	Каждый элемент контейнера
11	lower_bound (при работе с отсортированным контейнером)
12	accumulate
13	Random_shuffle, shuffle
14	th_element
15	Логарифмическая (O(log n))
16	Логарифмическая (O(log n))
17	Логарифмическая (O(log n))
18	Линейная (O(n))
19	Линейная (O(n))
20	Линейная (O(n))

Задания с развернутым ответом – 3 балла

1. Алгоритм сортировки `sort` в STL. Описание и оценки сложности.
2. Алгоритм бинарного поиска `binary_search` в STL. Описание и оценки сложности.
3. Алгоритм объединения отсортированных контейнеров `merge`. Описание и примеры использования.
4. Алгоритм удаления элементов, соответствующих условию, `remove_if`. Описание и оценки сложности.
5. Алгоритм частичного сортирования `partial_sort`. Описание и оценки сложности.
6. Алгоритм поиска минимального и максимального элементов `min_element` и `max_element`. Описание и оценки сложности.
7. Алгоритм поиска первого соответствия с использованием предиката `find_if`. Описание и примеры использования.
8. Определение контейнера `map` и его применение для реализации словарей.
9. Алгоритм вставки и удаления элементов из ассоциативного контейнера `set`. Описание и оценки сложности.
10. Приложение контейнеров `unordered_map` и `unordered_set` для быстрого поиска.
11. Ассоциативные контейнеры и их применение для хранения уникальных элементов.
12. Итераторы в STL. Основные определения и примеры использования.
13. Алгоритм объединения контейнеров `merge` и его применение для слияния данных.
14. Алгоритм поразрядной сортировки данных с использованием STL. Описание и оценки сложности.
15. Реализация алгоритма приоритетной очереди с использованием `priority_queue`. Описание и примеры.

Критерии оценивания	*Шкала оценок (в баллах)*
Обучающийся отвечает на все вопросы правильно.	3
Обучающийся отвечает на все вопросы. Допускаются незначительные неточности.	2
Обучающийся отвечает не на все вопросы. Ответ не содержит грубых ошибок.	1
Обучающийся отвечает не на все вопросы. Присутствуют грубые ошибки.	0

19.2. Промежуточная аттестация

Промежуточная аттестация проводится в соответствии с Положением о промежуточной аттестации обучающихся по программам высшего образования.

Контрольно-измерительные материалы промежуточной аттестации включают в себя теоретические вопросы, позволяющие оценить уровень полученных знаний и практические задания, позволяющие оценить степень сформированности умений и навыков.

Для оценивания результатов обучения на зачете с оценкой используются следующие показатели: владение теоретическими основами дисциплины, способность иллюстрировать ответ примерами, применять теоретические знания для решения практических задач.

Для оценивания результатов обучения на зачете применяется 4-балльная шкала: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Используются следующие показатели и их соотношения:

- уверенное владение теоретическими основами дисциплины, способность применять теоретические знания для решения практических задач, результаты выполнения всех заданий лабораторного практикума – «отлично»;
- хорошее владение теоретическими основами дисциплины, способность применять теоретические знания для решения практических задач, результаты выполнения большинства заданий лабораторного практикума – «хорошо»;
- неполное владение теоретическими основами дисциплины, затруднения в применении теоретических знаний для решения практических задач, результаты выполнения не менее 30% заданий лабораторного практикума – «удовлетворительно»;
- слабое владение теоретическими основами дисциплины, неспособность применять теоретические знания для решения практических задач, результаты выполнения менее 30% заданий лабораторного практикума – «неудовлетворительно».

Промежуточная аттестация по дисциплине осуществляется с помощью следующих оценочных средств: собеседование по зачетным билетам (КИМ). Перечень вопросов к зачету:

Введение

1.1. Основные принципы использования STL и его компоненты

1.2. Асимптотическая сложность алгоритмов и выбор подходящих контейнеров

1.3. Сравнение производительности контейнеров STL

Контейнеры STL

2.1. Последовательные контейнеры: vector, deque, list

2.2. Ассоциативные контейнеры: map, set, multimap, multiset

2.3. Неупорядоченные контейнеры: unordered_map, unordered_set

2.4. Итераторы и их типы: прямые, обратные, константные итераторы
Алгоритмы STL
3.1. Алгоритмы сортировки: sort, stable_sort, partial_sort
3.2. Поиск в контейнерах: find, binary_search, find_if
3.3. Алгоритмы модификации данных: remove, remove_if, transform
3.4. Объединение контейнеров: merge, set_union, set_intersection
3.5. Применение алгоритмов для фильтрации и обработки данных

Ассоциативные и неупорядоченные контейнеры
4.1. Внутреннее устройство и применение map и set
4.2. Вставка и удаление элементов в unordered_map и unordered_set
4.3. Применение хэш-таблиц для оптимизации поиска и доступа к данным
4.4. Различия и выбор между map и unordered_map в зависимости от задачи

Работа с итераторами
5.1. Использование итераторов для обхода и модификации контейнеров
5.2. Создание пользовательских итераторов для сложных структур данных
5.3. Разница между итераторами контейнеров последовательного и ассоциативного доступа
5.4. Применение итераторов вставки: inserter, front_inserter, back_inserter