ПРИЛОЖЕНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

«Основы алгоритмизации и объектно-ориентированное программирование»

Направление подготовки
02.03.03 «Математическое обеспечение
и администрирование информационных систем»

Профиль:
Программное обеспечение компьютерных технологий и систем искусственного интеллекта

Квалификация (степень) выпускника – бакалавр

Форма обучения – очная

Рязань

1 ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов (практических заданий, описаний форм и процедур проверки), предназначенных для оценки качества освоения обучающимися данной дисциплины как части ОПОП.

Цель – оценить соответствие знаний, умений и владений, приобретенных обучающимся в процессе изучения дисциплины, целям и требованиям ОПОП в ходе проведения промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности компетенций.

Контроль знаний обучающихся проводится в форме промежуточной аттестации.

Промежуточная аттестация проводится в форме зачета, экзамена и защиты курсовой работы. Форма проведения зачета и экзамена - тестирование, письменный опрос по теоретическим вопросам и выполнение практического задания.

2 ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Сформированность каждой компетенции (или ее части) в рамках освоения данной дисциплины оценивается по трехуровневой шкале:

1) пороговый уровень является обязательным для всех обучающихся по завершении освоения дисциплины;

2) продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;

3) эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

Уровень освоения компетенций, формируемых дисциплиной:

Описание критериев и шкалы оценивания тестирования:

Шкала оценивания	Критерий
3 балла (эталонный уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 85 до 100%
2 балла (продвинутый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 70 до 84%
1 балл (пороговый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 50 до 69%
0 баллов	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 0 до 49%

Описание критериев и шкалы оценивания теоретического вопроса:

Шкала оценивания	Критерий
3 балла (эталонный уровень)	выставляется студенту, который дал полный ответ на вопрос, показал глубокие систематизированные знания, смог привести примеры, ответил на дополнительные вопросы преподавателя
2 балла (продвинутый уровень)	выставляется студенту, который дал полный ответ на вопрос, но на некоторые дополнительные вопросы преподавателя ответил только с помощью наводящих вопросов
1 балл (пороговый уровень)	выставляется студенту, который дал неполный ответ на вопрос в билете и смог ответить на дополнительные вопросы только с помощью преподавателя
0 баллов	выставляется студенту, который не смог ответить на вопрос

Описание критериев и шкалы оценивания практического задания:

Шкала оценивания	Критерий
3 балла (эталонный уровень)	Задача решена верно
2 балла (продвинутый уровень)	Задача решена верно, но имеются неточности в логике решения
1 балл (пороговый уровень)	Задача решена верно, с дополнительными наводящими вопросами преподавателя
0 баллов	Задача не решена

Описание критериев и шкалы оценивания курсовой работы

Шкала оценивания	Критерий
Оценка «отлично» (эталонный уровень)	Курсовая работа (КР) выполнена в полном объеме, нет замечаний по разработке алгоритмов и программ, работа выполнена самостоятельно, пояснительная записка к КР оформлена аккуратно, соблюдались сроки сдачи и защиты КР, при защите КР студент ответил на все предложенные вопросы
Оценка «хорошо» (продвинутый уровень)	Курсовая работа выполнена в полном объеме, присутствуют незначительные замечания по разработке алгоритмов и программ, проект выполнен самостоятельно, пояснительная записка к КР оформлена аккуратно, соблюдались сроки сдачи и защиты КР, при защите КР студент ответил не на все предложенные вопросы (правильных ответов не менее 75%)
Оценка «удовлетворительно» (пороговый уровень)	Курсовая работа выполнена в полном объеме, присутствуют ошибки при разработке алгоритмов и программ, КР выполнена самостоятельно, по оформлению пояснительной записки к КР имеются замечания, частично соблюдались сроки сдачи и защиты КР, при защите КР студент ответил не на все предложенные вопросы (правильных ответов не менее 50%)
Оценка «неудовлетворительно»	Курсовая работа выполнен не в полном объеме, присутствуют грубые ошибки при разработке алгоритмов и программ, КР выполнена не самостоятельно, по оформлению пояснительной записки к КР имеются замечания, не соблюдались сроки сдачи и защиты КР, при защите КР студент ответил не на все предложенные вопросы (правильных ответов менее 50%)

На промежуточную аттестацию выносится тест, два теоретических вопроса и 1 задача. Максимально студент может набрать 12 баллов. Итоговый суммарный балл студента, полученный при прохождении промежуточной аттестации, переводится в традиционную форму по системе «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно», «зачтено», «не зачтено».

Оценка «отлично» выставляется студенту, который набрал в сумме 12 баллов (выполнил все задания на эталонном уровне). Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «хорошо» выставляется студенту, который набрал в сумме от 8 до 11 баллов при условии выполнения всех заданий на уровне не ниже продвинутого. Обязательным

условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «удовлетворительно» выставляется студенту, который набрал в сумме от 4 до 7 баллов при условии выполнения всех заданий на уровне не ниже порогового. Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «неудовлетворительно» выставляется студенту, который набрал в сумме менее 4 баллов или не выполнил все предусмотренные в течение семестра практические задания.

Оценка «зачтено» выставляется студенту, который набрал в сумме не менее 4 баллов при условии выполнения всех заданий на уровне не ниже порогового. Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «не зачтено» выставляется студенту, который набрал в сумме менее 4 баллов или не выполнил все предусмотренные в течение семестра практические задания.

Код компетенции	**Результаты освоения ОПОП Содержание компетенций**
ПК-1	Способен проектировать программное обеспечение с использованием современных инструментальных средств

ПК-1.1: Проектирует и разрабатывает программное обеспечение

ПК-1.2: Применяет современные инструментальные средства при разработке программного обеспечения

Код компетенции	**Результаты освоения ОПОП Содержание компетенций**
ПК-9	Способен применять языки программирования C/C++ для решения задач в области ИИ (PL-4)

ПК-9.1: Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений

ПК-9.2: Разрабатывает и отлаживает системы ИИ на C++ под конкретные аппаратные платформы с ограничениями по вычислительной мощности, в том числе для встроенных систем

3 ПАСПОРТ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ДИСЦИПЛИНЕ

	Контролируемые разделы (темы) дисциплины	**Код контролируемой компетенции (или её части)**	**Вид, метод, форма оценочного мероприятия**
1.	Общие принципы разработки программного обеспечения	**ПК-1.1, ПК-1.2**	Зачет, курсовая работа
2.	Основные элементы языка программирования C/C++	**ПК-1.1, ПК-1.2**	Зачет, курсовая работа
3.	Операции языка. Выражения. Математические функции	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет, курсовая работа
4.	Операторы языка C++	**ПК-1.1, ПК-1.2 ПК-9.1, ПК-9.2**	Зачет, курсовая работа
5.	Указатели, адресная арифметика	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет, курсовая работа
6.	Массивы, строки и указатели	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет, курсовая работа

7.	Функции и классы памяти	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет, курсовая работа
8.	Структуры и объединения	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет, курсовая работа
9.	Потоковый ввод-вывод. Файлы	**ПК-1.1, ПК-1.2**	Зачет, курсовая работа
10.	Указатели и динамические структуры данных	**ПК-1.1, ПК-1.2**	Зачет, курсовая работа
11.	Простейший графический интерфейс в Visual C++.	**ПК-1.2**	Зачет, курсовая работа
12.	Базовые принципы ООП. Объекты и классы в языке C++	**ПК-1.1, ПК-1.2, ПК-9.2**	Зачет
13.	Перегрузка операторов	**ПК-1.1**	Экзамен
14.	Шаблоны	**ПК-1.1, ПК-1.2, ПК-9.2**	Экзамен
15.	Наследование. Производные классы.	**ПК-1.1, ПК-1.2**	Экзамен
16.	Виртуальные функции	**ПК-1.1, ПК-1.2**	Экзамен
17.	Обработка исключений.	**ПК-1.1, ПК-1.2**	Экзамен
18.	Потоковые классы	**ПК-1.1, ПК-1.2**	Экзамен
19.	Стандартная библиотека (STL).	**ПК-1.1, ПК-1.2**	Экзамен
20.	Многопоточность в C++	**ПК-1.1, ПК-1.2, ПК-9.1**	Экзамен

4 ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ

4.1. Промежуточная аттестация в форме экзамена и зачета

Код компетенции	Результаты освоения ОПОП Содержание компетенций
ПК-1	Способен проектировать программное обеспечение с использованием современных инструментальных средств

ПК-1.1. Проектирует и разрабатывает программное обеспечение

Типовые тестовые вопросы:

1. В языке C функция, с которой начинается выполнение программы, называется _____.
Ответ: `main`

2. В C++ для выделения динамической памяти под один объект используется оператор _____, а для освобождения — _____.
Ответ: `new, delete`

3. В C++ функция может быть перегружена, если у неё отличаются _____ или _____ параметров.
Ответ: типы, количество

4. В C++ виртуальный метод объявляется с ключевым словом _____, а чисто виртуальный — с добавлением _____.
Ответ: `virtual, = 0`

5. Чтобы функция могла изменять значение аргумента в вызывающей программе, в C используется передача по _____, а в C++ — по _____ или _____.
Ответ: указателю, ссылке, указателю

6. Функции, которые вызывают сами себя, называются _____.
Ответ: рекурсивными

7. Указатель – это переменная, которая содержит в качестве своего значения _____ другой переменной.
Ответ: адрес

8. Функция-член класса, которая вызывается автоматически при создании объекта, называется _____.
Ответ: конструктор

9. Деструктор объявляется с символом _____ перед именем.
Ответ: 4. ~ (тильда)

10. Ключевое слово _____ позволяет функции или классу получать доступ к приватным членам другого класса.
Ответ: `friend` (дружественная)

11. Шаблон класса объявляется с помощью ключевого слова _____.
Ответ: `template`

12. Для обработки исключений блок кода, который может вызвать исключение, помещается в блок _____.
Ответ: `try`

13. Спецификатор доступа _____ делает члены класса доступными в производных классах, но не извне.
Ответ: `protected`

14. В C++ класс по умолчанию имеет уровень доступа _____ для своих членов.
Ответ: `private`

15. Оператор _____ перегружается для создания «функциональных объектов» (функторов).
Ответ: `operator()`

16. `std::vector` — это контейнер с _____ доступом к элементам и динамическим размером.
Ответ: произвольным

17. `std::list` реализован как _____ связанный список.
Ответ: двусвязный, двунаправленный

18. `std::map` хранит пары «ключ-значение», отсортированные по _____.
Ответ: ключу (или «возрастанию ключей»)

19. Чтобы получить итератор на первый элемент контейнера, вызывают метод _____.
Ответ: `begin()`

20. Чтобы получить количество элементов в контейнере, вызывают метод _____.
Ответ: `size()`

21. Выберите верное определение алгоритма:

1. Алгоритм – набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата.
2. Алгоритм – набор упорядоченных действий исполнителя для достижения некоторого результата.
+3. Алгоритм – набор инструкций, описывающих порядок действий компьютера для достижения некоторого результата.
4. Алгоритм – набор инструкций для написания программного кода и составления схемы алгоритма при разработке программы.

22. Выберите все свойства алгоритма:
+1. Массовость
2. Идейность
+3. Понятность
+4. Дискретность
5. Лаконичность
+6. Конечность
+7. Определенность
8. Уникальность
9. Работоспособность
+10. Эффективность.

23. Что такое отладка и зачем она нужна?
1. Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Она необходима для того, чтобы пользователь мог увидеть ошибки в программе и сообщить о них разработчику.
+2. Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Она необходима для того, чтобы разработчик мог найти ошибки в программе и устранить их.
3. Отладка – этап разработки компьютерной программы, на котором отлаживают совместную работу программного обеспечения.

24. Что определяет тип данных?
+1. Определяет возможный диапазон значения переменных, а также операции и функции с ними
2. Определяет внешний вид значений при записи в коде программы
3. Определяет внешний вид значений, возможный диапазон и операции с ними
4. Определяет внутреннее представление данных

25. Переменные – это:
+1. величины, которые могут менять свое значение в процессе выполнения программы
2. величины, которые не могут менять своего значения в процессе выполнения программы
3. обозначают строки программы, на которые передается управление во время выполнение программы

26. Какие три базовые управляющие структуры лежат в основе структурного программирования?
1. Последовательность, ветвление, рекурсия
+2. Ветвление, цикл, последовательность
3. Цикл, функция, класс
4. Условие, исключение, последовательность

27. Какое условие используется для выхода из цикла `while`?
1. Условие должно стать истинным
+2. Условие должно стать ложным
3. Условие не влияет на выход из цикла
4. Цикл `while` не имеет условия выхода

28. Что произойдет, если условие цикла `while` изначально ложно?
1. Цикл выполнится один раз
+2. Тело цикла не выполнится ни разу
3. Программа завершится с ошибкой
4 Цикл будет выполняться бесконечно

29. Какой из следующих терминов описывает "один интерфейс — множество реализаций"?
1. Инкапсуляция
2. Абстракция
+3. Полиморфизм
4. Наследование

30. Что такое множественное наследование?
+1. Наследование от более чем одного базового класса
2. Наследование с использованием шаблонов
3. Наследование только интерфейсов
4. Рекурсивное наследование

31. Что обеспечивает полиморфизм во время выполнения в C++?
1. Перегрузка функций
2. Шаблоны функций (function templates)
3. Указатели на функции
+4. Виртуальные функции и механизм динамического связывания

32. Что такое "шаблон класса"?
1. Готовый класс для использования в любой программе.
+2. "Рецепт" для создания семейства классов, где в качестве параметров можно передавать типы данных.
3. Абстрактный базовый класс.
4. Класс, все методы которого являются виртуальными.

33. Какая из этих пар ключевых слов используется для управления исключениями в C++?
```
1. throw / catch
2. try / handle
3. error / except
+4. try / catch
```

34. Какой из перечисленных контейнеров НЕ поддерживает произвольный доступ к элементам через оператор `[]`?
```
1. std::vector
2. std::deque
+3. std::list
4. std::array
```

35. Какой контейнер лучше всего использовать, если нужна структура `FIFO`?

1. std::stack
+2. std::queue
3. std::priority_queue
4. std::deque

36. Какой контейнер гарантирует, что элементы хранятся в отсортированном порядке?
1. std::unordered_set
2. std::vector, если его отсортировать
+3. std::set
4. std::deque

37. Для чего используется `std::pair`?
1. Для хранения двух однотипных значений
+2. Для хранения двух значений (возможно, разных типов)
3. Как базовый класс для `std::tuple`
4. Только в составе `std::map`

38. Чем отличается std::multimap от std::map?
1. `multimap` не сортирует элементы
+2. `multimap` позволяет дубликаты ключей
3. `multimap` хранит только уникальные значения
4. `multimap` работает быстрее при вставке

39. Какой заголовочный файл нужен для `std::sort`?
1. <utility>
+2. <algorithm>
3. <functional>
4. <iterator>

40. Какой из перечисленных контейнеров имеет фиксированный размер после инициализации?
1. std::vector
2. std::list
+3. std::array
4. std::deque

Типовые теоретические вопросы

1. Какие основные типы данных в C/C++?
Ответ:
- Целочисленные: `char, short, int, long, long long`
- Вещественные: `float, double, long double`
- Логический: `bool` (только в C++)
- Пустой тип: `void`
- Пользовательские: структуры, объединения, перечисления

2. Что такое структуры и как они отличаются в C и C++?
Ответ: Структуры - составные типы данных, группирующие переменные разных типов.
В C - только данные, в C++ - могут содержать функции-члены, конструкторы, модификаторы доступа.

3. Что такое шаблоны функций в C++?

Ответ: Шаблоны позволяют создавать обобщенные функции, работающие с разными типами данных.

4. Что такое динамическое выделение памяти и какие функции для этого используются?
Ответ: Выделение памяти во время выполнения программы.
В C: `malloc(), calloc(), realloc(), free()`
В C++: `new, delete, new[], delete[]`

5. Что такое `inline`-функции в C++ и когда их используют?
Ответ: `inline`-функции - функции, код которых подставляется в месте вызова вместо обычного вызова. Используются для небольших часто вызываемых функций чтобы избежать накладных расходов на вызов.

6. Как передать массив в функцию в C?
Фактически передаётся указатель на первый элемент. Поэтому рекомендуется также передавать длину массива:

7. Что такое ссылка в C++?
Псевдоним для существующей переменной. Объявляется как `тип &имя`.

8. Что такое инкапсуляция в C++?
Ответ: Механизм объединения данных и методов работы с ними в единый объект с контролем доступа через спецификаторы `public, private, protected`.

9. Что такое конструктор?
Ответ: Специальная функция-член, которая автоматически вызывается при создании объекта для его инициализации.

10. Что такое абстрактный класс?
Ответ: Класс, содержащий хотя бы одну чисто виртуальную функцию. Нельзя создать объект абстрактного класса.

11. Что такое шаблоны классов?
Ответ: Механизм для создания обобщенных классов, работающих с разными типами данных.

12. Как обрабатываются исключения в C++?
Ответ: С помощью `try-catch` блоков. Исключения генерируются через `throw`.

13. Что такое STL и из каких основных компонентов она состоит?
Ответ: STL - стандартная библиотека шаблонов C++, предоставляющая набор готовых компонентов для работы с данными. Основные компоненты: контейнеры, адаптеры контейнеров, алгоритмы, итераторы, функциональные объекты (функторы).

14. Какие основные типы контейнеров существуют в STL?
Ответ:
- Последовательные: vector, `list, deque, array`
- Ассоциативные: `set, map, multiset, multimap`
- Неупорядоченные ассоциативные: `unordered_set, unordered_map`
- Адаптеры контейнеров: `stack, queue, priority_queue`

15. Что такое алгоритмы STL и как они используют итераторы?

Ответ: Алгоритмы STL - шаблонные функции для обработки данных. Они работают через итераторы, что делает их независимыми от конкретных контейнеров:

16. Как работает контейнер `map` и что такое `pair`?
Ответ: `map` - ассоциативный контейнер "ключ-значение". `pair` - структура для хранения двух элементов:

17. Что такое адаптеры контейнеров?
Ответ: Адаптеры используют другие контейнеры для реализации специфичного интерфейса: `stack` - LIFO (последний вошел - первый вышел), `queue` - FIFO (первый вошел - первый вышел), `priority_queue` - очередь с приоритетами

18. Как работает vector и что такое `capacity` vs `size`?
Ответ: `size` - текущее количество элементов, `capacity` - объем выделенной памяти. При добавлении элементов `capacity` может увеличиваться с запасом для эффективности

19. Чем последовательные контейнеры отличаются от ассоциативных?
Ответ: Последовательные (`vector, deque, list`) хранят элементы в определённом порядке (линейно), доступ — по позиции. Ассоциативные (`set, map, unordered_set, unordered_map`) хранят элементы по ключу, обеспечивают быстрый поиск.

20. Чем `std::array` отличается от C-массива и `std::vector`?
Ответ: `std::array` — обёртка над C-массивом: фиксированный размер (известен на этапе компиляции), стековое хранение, поддержка `.size()`, итераторов, `std::begin/end`. В отличие от `vector`, не может менять размер и не выделяет память в куче.

ПК-1.2. Применяет современные инструментальные средства при разработке программного обеспечения

Типовые тестовые вопросы:

1. _____ - это интегрированная среда разработки, в которую входят инструменты MSVC.
Ответ: Visual Studio

2. Компилятор MSVC поддерживает современные стандарты языка _____.
Ответ: C++

3. Для поиска и исправления ошибок в выполняемой программе используется _____.
Ответ: отладчик (debugger).

4. Система сборки проектов C++ в Visual Studio называется _____.
Ответ: MSBuild

5. Технология _____ помогает создавать приложения с графическим интерфейсом методом "перетаскивания".
Ответ: WinForms

6. Файл решения в Visual Studio имеет расширение _____.
Ответ: `.sln`

7. Проект C++ в Visual Studio имеет расширение _____.
Ответ: `.vcxproj`.

8. Что такое MSVC в контексте разработки программного обеспечения?
1. Среда для работы с базами данных
2. Операционная система от Microsoft
+3. Компилятор и набор инструментов для языков C и C++ от Microsoft
4. Графический редактор

9. Какой инструмент НЕ является системой сборки, поддерживаемой в Visual Studio?
1. MSBuild
2. CMake
+3. Apache Kafka

10. Какой инструмент позволяет "шагать" по коду строка за строкой во время отладки?
1. Профилировщик (Profiler)
2. Конструктор форм (Form Designer)
+3. Пошаговое выполнение (Step Over, Step Into)
4. Статический анализатор кода

11. Какой инструмент используется для измерения производительности программы, например, для поиска "узких мест" в коде?
1. Отладчик (Debugger)
+2. Профилировщик (Profiler)
3. Компилятор (Compiler)
4. Редактор кода (Code Editor)

12. Какой анализатор помогает находить потенциальные ошибки, утечки памяти и проблемы с безопасностью, не запуская программу?
1. Динамический анализатор
+2. Статический анализатор кода
3. Отладчик
4. Профилировщик

13. Какой инструмент НЕ является частью инструментария MSVC для анализа производительности?
1. Профилировщик использования ЦП
2. Профилировщик использования памяти
+3. Диспетчер задач Windows
4. Профилировщик ввода-вывода

14. Какой бесплатный выпуск Visual Studio подходит для индивидуальных разработчиков и небольших команд?
1. Visual Studio Enterprise
2. Visual Studio Professional
+3. Visual Studio Community
4. Visual Studio Code

Типовые теоретические вопросы

1. Что такое форма?
Форма — это главный контейнер, в котором размещаются компоненты самой среды. С помощью этих компонентов и реализуется конкретный алгоритм определенной задачи.

2. Какие типы компонентов вы знаете?
Различают два типа компонентов: визуальные и невизуальные.

3. Визуальный компонент – это …
элемент пользовательского интерфейса, например, поле редактирования (TextBox), поле отображения текста (Label), кнопка (Button).

4. Визуальные компоненты отображаются …
как на форме (во время разработки программы), так и в окне программы во время ее работы.

5. Невизуальные компоненты отображаются …
только на форме во время разработки программы.

6. Компоненты, которые программист может использовать при разработке программ, в среде Visual Studio находятся …
на вкладке Toolbox.

7. Каждый компонент характеризуется наборами данных, определяющими его функциональность: …
свойства, события и методы.

8. Свойства – это…
переменные, которые влияют на состояние объекта. Например, ширина, высота, положение кнопки на форме или надпись на ней.

9. Методы – это …
функции, то есть это то, что объект умеет делать (вычислять).

10. События – это …
функции, которые вызываются при наступлении определенного события. Например, пользователь нажал на кнопку, вызывается процедура обработки этого нажатия.

Код компетенции	**Результаты освоения ОПОП** **Содержание компетенций**
ПК-9	Способен применять языки программирования C/C++ для решения задач в области ИИ (PL-4)

ПК-9.1: Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений

Типовые тестовые вопросы:

1. Для создания потока используется класс _____ из стандартной библиотеки.
Ответ: `std::thread`

2. Функция _____ используется для ожидания завершения работы потока.
Ответ: `join()`

3. Для защиты общих данных от состояний гонки используется _____.
Ответ: мьютекс (или mutex)

4. После завершения работы с объектом std::thread необходимо вызвать либо join(), либо _____() — иначе при уничтожении будет вызван std::terminate().
Ответ: **`detach`**

5. Чтобы получить уникальный идентификатор текущего потока, вызывают:
auto id = std::this_thread::_____();
Ответ: **`get_id`**

6. Чтобы на 500 мс приостановить текущий поток, вызывают:
std::this_thread::_____(500ms);
Ответ: **`sleep_for`**

7. Класс _____ обеспечивает автоматическое освобождение мьютекса при выходе из области видимости.
Ответ: `lock_guard`

8. _____ возникает, когда два или более потока бесконечно ожидают ресурсы, захваченные друг другом.
Ответ: Взаимная блокировка (Deadlock)

9. Условные переменные (`condition_variable`) используются для _____ между потоками.
Ответ: взаимодействия (или синхронизации)

10. Для атомарных операций с простыми типами используется шаблонный класс _____.
Ответ*:* **`std::`**`atomic`

11. Для одновременного захвата нескольких мьютексов без риска взаимной блокировки используется функция _____.
Ответ*:* `std::lock`

12. Функция `std::_____` позволяет выполнить функцию асинхронно и получить будущий результат.
Ответ*:* `async`

13. Что такое состояние гонки (race condition)?
1. Когда программа работает медленнее из-за большого количества потоков
+2. Когда несколько потоков обращаются к общим данным без синхронизации, и результат зависит от порядка выполнения
3. Когда потоки соревнуются за процессорное время
4. Когда мьютекс не может быть захвачен

14. Какой класс НЕ является примитивом синхронизации в стандартной библиотеке C++?
```
1. std::mutex
2. std::atomic
+3. std::thread
4. std::condition_variable
```

15. Какой из этих методов НЕ принадлежит `std::thread`?
```
1. join()
2. detach()
```

+3. lock()
4. get_id()

16. Для чего используется `std::condition_variable`?
1. Для создания потоков-обработчиков
+2. Для ожидания выполнения определенного условия
3. Для атомарных операций
4. Для измерения производительности

17. Какой класс следует использовать для отложенного захвата мьютекса?
1. std::lock_guard
2. std::thread
+3. std::unique_lock
4. std::atomic

18. Что гарантирует `std::atomic<int>`?
1. Что операции с **`int`** будут выполняться быстрее
+2. Что операции с **`int`** будут потокобезопасными без использования мьютексов
3. Что **`int`** будет занимать меньше памяти
4. Что **`int`** будет автоматически инициализирован нулем

19. Как правильно защитить общие данные от состояния гонки?
1. Использовать больше потоков
+2. Использовать мьютексы или атомарные операции
3. Увеличить приоритет потоков
4. Уменьшить количество кэшей процессора

20. Что означает "потокобезопасность"?
1. Что программа использует только один поток
+2. Что код корректно работает при одновременном выполнении из нескольких потоков
3. Что потоки никогда не блокируются
4. Что программа не использует системные вызовы

21. Какой метод `std::thread` используется для ожидания завершения потока?
1. wait()
2. detach()
+3. join()
4. sleep()

22. Что такое `std::async`?
1. Мьютекс для асинхронных операций
2. Условная переменная для **`callback`**-ов
3 Поток с приоритетом
+4. Средство для запуска асинхронных задач

23. Как правильно передать переменную `int x` по ссылке в лямбду, запускаемую в `std::thread`?
+1. std::thread t([&x]{ x = 42; });
2. std::thread t([x]{ x = 42; });
3. std::thread t([=]{ x = 42; });
4. std::thread t([&]{}, x);

24. Какое условие необходимо для возникновения `deadlock`?

+1. Наличие хотя бы двух потоков
+2. Циклическое ожидание ресурсов
3. Использование **`std::atomic`**
4. Отсутствие **`std::cout`** в критической секции

25. Какой из вариантов потокобезопасен для одновременной записи из нескольких потоков?
`1. std::vector<int>` без синхронизации
`2. std::array<int, 10>`
`3. std::string` при вызове **`c_str()`**
`+4. std::vector<int> с std::mutex` вокруг каждого **`push_back`**

Типовые теоретические вопросы

1. Что такое поток (`thread`) в C++?
Ответ: Поток — это наименьшая единица выполнения, которая может выполняться параллельно с другими потоками в рамках одного процесса.

2. Как создать поток в современном C++?
Ответ: С помощью класса `std::thread`, передав в конструктор функцию или лямбда-выражение: `std::thread` t(функция);

3. Что такое состояние гонки (`race condition`)?
Ответ: Это ошибка, когда несколько потоков одновременно обращаются к общим данным, и хотя бы один поток изменяет эти данные, что приводит к непредсказуемому результату.

4. Что такое взаимная блокировка (`deadlock`)?
Ответ: Ситуация, когда два или более потока бесконечно ожидают ресурсы, захваченные друг другом.

5. Для чего используется мьютекс (`mutex`)?
Ответ: Для защиты общих данных от одновременного доступа нескольких потоков, обеспечивая взаимное исключение.

6. Что такое условная переменная (condition variable)?
Ответ: Механизм синхронизации, позволяющий потокам ожидать выполнения определенного условия.

7. Что такое атомарная операция?
Ответ: Операция, которая выполняется как единое целое без возможности прерывания другими потоками.

8. Для чего используется `std::atomic`?
Ответ: Для обеспечения атомарного доступа к данным без использования мьютексов.

9. В чем преимущество `std::atomic` перед мьютексом?
Ответ: Атомарные операции обычно более эффективны, так как не требуют блокировок и используют аппаратную поддержку процессора.

10. В чем разница между `join()` и `detach()`?
Ответ: `join()` ожидает завершения потока, `detach()` разрешает потоку работать независимо от объекта `std::thread`.

11. Что такое потокобезопасность?
Ответ: Свойство кода корректно работать при одновременном выполнении из нескольких потоков.

12. Что такое модель памяти в C++?
Ответ: Набор правил, определяющих, как операции в разных потоках взаимодействуют через память.

ПК-9.2: Разрабатывает и отлаживает системы ИИ на C++ под конкретные аппаратные платформы с ограничениями по вычислительной мощности, в том числе для встроенных систем

Типовые тестовые вопросы:

1. В системах с ограниченной памятью рекомендуется использовать _____ выделение памяти вместо динамического.
Ответ: статическое

2. Арифметика _____ используется как альтернатива плавающей точке на системах без FPU.
Ответ: фиксированной точки

3. Алгоритмы стандартной библиотеки STL часто избегают из-за непредсказуемого использования _____ памяти.
Ответ: динамической

4. Ключевое слово _____ в C++ гарантирует, что функция будет встроена в точку вызова, если это возможно.
Ответ: `inline`

5. Битовые _____ позволяют эффективно упаковывать несколько логических значений в одну переменную.
Ответ: поля

6. При работе с ограниченными ресурсами часто используют _____ программирование, отказываясь от некоторых возможностей C++ в пользу C.
Ответ: процедурное

7. Какая стратегия управления памятью наиболее предпочтительна в real-time embedded системах?
1. Динамическое выделение через `new/delete`
+2. Статическое выделение при компиляции
3. Использование умных указателей
4. Сборка мусора

8. Что из перечисленного является основной причиной отключения исключений в embedded C++?
1. Сложность синтаксиса
+2. Накладные расходы на память и время выполнения

3. Отсутствие поддержки в компиляторах
4. Несовместимость с языком C

9. Какой тип арифметики чаще используют вместо floating-point на системах без FPU?
1. `BigInt` арифметика
+2. `Fixed-point` арифметика
3. `Decimal` арифметика
4. `Complex` number арифметика

10. Какая из этих структур данных наиболее предпочтительна для embedded систем?
1. `std::list` с динамическим выделением
2. `std::vector` с автоматическим ростом
+3. `std::array` фиксированного размера
4. `std::map` с балансировкой дерева

11. Какой подход к обработке ошибок наиболее распространен в embedded C?
1. Исключения
+2. Коды возврата
3. `Assert` макросы
4. `Longjmp/setjmp`

12. Для чего используется битовая упаковка?
1. Для ускорения вычислений
+2. Для экономии памяти
3. Для упрощения синтаксиса
4. Для автоматической оптимизации

Типовые теоретические вопросы:

1. Какие основные ограничения встроенных систем влияют на программирование на C++?
Ответ: Основные ограничения:
- Ограниченная память (ОЗУ от нескольких КБ до МБ)
- Низкая тактовая частота процессора
- Отсутствие или ограниченная система управления памятью (MMU)
- Энергопотребление и тепловыделение
- Отсутствие операционной системы или использование RTOS
- Жесткие требования к реальному времени

2. Почему исключения C++ часто отключают во встроенных системах?
Ответ: Исключения требуют:
- Дополнительной памяти для хранения информации о типах
- Накладных расходов на раскрутку стека
- Непредсказуемого времени выполнения

3. Какие особенности использования динамической памяти во встроенных системах?
Ответ:
- `new/delete` могут фрагментировать память
- Непредсказуемое время выделения памяти
- Риск исчерпания памяти во время выполнения
- Часто используют статическое выделение или пулы памяти

4. Как ограничения памяти влияют на использование STL?
Ответ: STL контейнеры могут:
- Создавать неожиданные вызовы `new/delete`
- Занимать много памяти из-за шаблонного инстанцирования (создания конкретной версии шаблона с заданными параметрами типа или значениями.)
- Иметь избыточную функциональность
- Часто используют упрощенные реализации или собственные контейнеры

5. Какие ограничения накладывают 8-битные и 16-битные микроконтроллеры?
Ответ:
- Ограниченный набор инструкций
- Маленький стек (часто 256 байт)
- Отсутствие аппаратной поддержки операций с плавающей точкой
- Ограниченная адресуемая память
- Часто используют подмножество C++ без исключений, RTTI, сложных шаблонов

6. Какие особенности использования floating-point операций?
Ответ:
- На системах без FPU используются `software`-эмуляция (медленно)
- `Fixed-point` арифметика как альтернатива
- Осторожность с накоплением ошибок округления
- Использование `float` вместо `double` на 32-битных системах

7. Почему в микроконтроллерах с малым объёмом RAM (например, 2 КБ) не рекомендуется использовать исключения (`try/catch`)?
Ответ:
Потому что обработка исключений требует значительных накладных расходов:

8. Почему STL (особенно `std::vector`, `std::string`) часто избегают в bare-metal прошивках?
Ответ:
- Выделяют память в куче через `new/malloc`,
- Динамическое выделение памяти небезопасно в real-time (фрагментация, неопределённое время),
- Исключения в `std::bad_alloc`,
- `std::string` — дорого по памяти и скорости.
Поэтому используют: статические буферы, `std::array`, или вообще `C-style (char buf[N]).`

9. Почему printf не рекомендуется использовать в production-прошивках?
- Тяжёлый по памяти (несколько КБ Flash),
- Использует кучу через `malloc` (в некоторых реализациях),

10. Почему в embedded-разработке предпочтение часто отдают C, а не C++?
Ответ:
- Си проще, предсказуемее и ближе к «железу»;
- Отсутствие скрытых накладных расходов

4.2. Промежуточная аттестация в форме курсовой работы

Код компетенции	Результаты освоения ОПОП Содержание компетенций
ПК-1	Способен проектировать программное обеспечение с использованием современных инструментальных средств

ПК-1.1 Проектирует и разрабатывает программное обеспечение

ПК-1.2 Применяет современные инструментальные средства при разработке программного обеспечения

Типовое задание для курсовой работы по дисциплине:

Целью курсового проектирования являются разработка и отладка приложения с использованием динамических структур данных, написанного на алгоритмическом языке программирования C++ в среде визуального программирования Visual C++.

Тема курсовой работы: ««Разработка приложения с использованием динамических структур данных».

Курсовая работа (КР) представляет собой самостоятельную работу по заданной теме. Работа предполагает:

- домашнюю внеаудиторную подготовку;
- выполнение практической части дома или в классе персональных компьютеров на кафедре;
- консультации по КР;
- предъявление промежуточных результатов для проверки и контроля хода курсового проектирования;
- написание и оформление пояснительной записки;
- сдачу и защиту КР в сроки согласно учебному графику.

Исходные данные

- Тип компьютера IBM PC совместимый.
- Операционная система Windows XP (не ниже).
- Язык программирования C/C++.
- Среда программирования: Microsoft Visual Studio.
- Текст задания согласно варианту.

Общие требования

Все варианты заданий связаны с разработкой таблицы данных с использованием линейных однонаправленных списков.

Разрабатываемая программа должна обязательно выполнять следующие запросы:

- заполнение пустой таблицы (создание списка);
- сохранение таблицы (списка) в файле;
- чтение таблицы (списка) из файла;
- вывод таблицы на экран;
- добавление элементов в таблицу (в список);
- удаление элементов из таблицы (из списка);
- а также все запросы, которые указаны в индивидуальном задании.

Вызовы запросов должны осуществляться через систему меню с использованием средств визуального программирования. Необходимо предусмотреть контроль ошибок пользователя

при вводе данных. Результаты некоторых запросов (по согласованию с преподавателем на этапе уточнения технического задания) должны выводиться в виде графиков или диаграмм. Приложение обязательно должно иметь заставку.

Все элементарные действия (заполнение списка, запись элемента в список и т.д.) должны быть оформлены в виде подпрограмм, а все (или некоторые) объявления и подпрограммы должны быть оформлены в виде модуля (модулей).

Перечень индивидуальных заданий:

Вариант № 1. Абоненты библиотеки

Информация об абонентах библиотеки следующая:
- номер читательского билета;
- ФИО;
- год рождения;
- пол;
- подразделение (кафедра, номер группы);
- должность;
- отметка о перерегистрации;
- имеются книги на срок;
- дата возврата книг.

Написать программу, которая выполняет следующие запросы:
- вывод информации об абоненте по номеру читательского билета;
- упорядочение таблицы по ФИО;
- вывод списка абонентов-должников определенного факультета;
- вывод списка абонентов, которые не прошли перерегистрацию;
- вывод процентного соотношения сотрудников и студентов среди абонентов.

Вариант № 2. Анкета студента

Анкета студента имеет следующие пункты:
- ФИО;
- пол;
- факультет;
- номер группы;
- адрес постоянного проживания;
- сведения о получении стипендии;
- вид спорта.

Написать программу, которая выполняет следующие запросы:
- вывод списка студентов определенной группы по алфавиту;
- вывод списка студентов на факультете, которые занимаются спортом;
- вывод списка иногородних студентов на факультете;
- вывод списка студентов на факультете, которые не получают стипендию;
- вывод процентного соотношения мужчин и женщин на факультете.

Вариант № 3. Анкета абитуриента

Анкета абитуриента имеет следующие пункты:
- ФИО;
- адрес постоянного проживания;
- льгота (инвалидность, сирота, целевой набор);
- баллы по ЕГ (математика, физика, русский язык);

– номер направления. По каждой специальности определен проходной балл.

Написать программу, которая выполняет следующие запросы:

– упорядочение таблицы по ФИО абитуриента;

– определение процента иногородних абитуриентов;

– вывод списка абитуриентов, которые поступили на определенное направление;

– вывод номеров направлений в порядке убывания «популярности»;

– вывод процентного соотношения льготников и абитуриентов, которые поступают в вуз на общих основаниях.

Полный список индивидуальных заданий находится в методических указаниях к КР. Дублирование тем для индивидуального исследования в пределах одной учебной группы не допускается.

Защита курсовой работы назначается по итогам проверки предоставленной пояснительной записки, оформленной в соответствии с требованиями и разработанного приложения. Защита осуществляется в форме ответов на вопросы преподавателя.

Типовые вопросы на защите курсовой работы:

1. Как разрабатывался графический интерфейс для вашего приложения?
2. Какие визуальные и не визуальные компоненты вы использовали?
3. Есть ли в вашем приложении модальные окна? Для каких целей вы их используете?
4. Как создавалась заставка для вашего приложения?
5. Как создается однонаправленный динамический список?
6. Как в списке происходит поиск данных?
7. Как отображаются в вашем приложении табличные данные?
8. Как осуществляется чтение и запись данных
9. Опишите назначение компонента MainMenu. Как вы используете эту компоненту в проекте?
10. Объясните алгоритм(ы) различных функций.
11. Как эти алгоритмы реализуются на языке C++.

Типовые теоретические вопросы для зачета по дисциплине (семестр 1)

1. Метод проектирования программных средств. Основные этапы решения задач на ЭВМ.
2. Понятия алгоритма, его свойства. Способы представления алгоритмов. Базовые алгоритмические структуры: следование, разветвление, повторение. Способы их изображения.
3. Структура языка: алфавит, синтаксис языка. Лексическая структура языка. Ключевые слова. Идентификаторы. Константы. Комментарии.
4. Структура программы. Стиль записи программы на C/C++.
5. Понятие и классификация типов данных. Скалярные (простые) типы данных: целые, символьные, вещественные, логические. Определение и инициализация переменных.
6. Операции в C/C++: присваивания, арифметические, логические, операции сдвига, сравнения, побитовые, условная, `sizeof`, запятая. Приоритеты операций и их направленность выполнения.
7. Выражение. Приоритет операций в выражении.
8. Явное преобразование типов. Неявное преобразование типов: расширение типа в выражении; преобразование при присваивании.
9. Понятие потока. Ввод и вывод в стелях Си и C++.
10. Разветвляющиеся алгоритмы. Оператор выбора `switch` и команда ветвления `if else`. Запись команды выбора с помощью команд ветвления.
11. Операторы циклов. Использование шаблонов при создании циклов. Операторы

прерывания циклов.
12. Классификация структурированных типов. Одномерные и многомерные массивы. Объявление и инициализация.
13. Указатели. Объявление и инициализация. Указатели и массивы. Операции с указателями.
14. Определение, описание и вызов функций. Тип функции. Оператор `return`.
15. Понятие фактических и формальных параметров. Прототипы функции.
16. Способы передачи параметров в функцию. Ссылка. Передача массивов в функцию.
17. Понятие блока. Область действия и область видимости глобальных и локальных переменных.
18. Классы памяти.
19. Техника многомодульного программирования
20. Указатели типа `void`. Преобразование типа указателя.
21. Указатели на указатели. Указатели и многомерные массивы. Массивы указателей.
22. Функции, возвращающие указатели. Указатели на функции.
23. Интерпретация сложных описаний.
24. Аргументы функции main().

Типовые теоретические вопросы для зачета по дисциплине (семестр 2)

1. Строки: определение, инициализация, функции для работы со строками.
2. Структуры. Определение и инициализация. Создание таблиц: массивы структур и структуры содержащие массивы. Вложенные структуры. Указатели как поля структур. Указатели на структуры и передача структур в функцию. Битовые поля структур.
3. Объединения. Определение и инициализация.
4. Файлы. Основные определения. Основные этапы работы с файлом.
5. Символьный, форматированный и строковый ввод и вывод. Блочный ввод-вывод. Произвольный доступ.
6. Динамическое распределение памяти в стилях С и С++.
7. Одномерные динамические массивы в стилях Си и С++.
8. Динамические структуры данных. Список и стек.
9. Двумерные динамические массивы в стилях Си и С++.
10. Типичные ошибки при работе с динамической памятью.
11. ООП. Основные свойства ООП.
12. Описание класса. Выделение памяти под объекты.
13. Конструкторы. Конструкторы без параметров и с параметрами. Конструктор копирования. Деструкторы.
14. Функции с переменным числом аргументов
15. Рекурсия.
16. Встроенные функции.
17. Макрофункции.
18. Параметры со значениями по умолчанию

Типовые теоретические вопросы для экзамена по дисциплине (семестр 3)

1. ООП. Основные свойства. Классы и объекты (определение и использование класса, вызов методов класса). Конструкторы. Перегруженные конструкторы. Конструктор копирования по умолчанию. Деструкторы.
2. Определение методов класса вне класса. Объекты в качестве аргументов функций. Объекты, возвращаемые функцией. Классы, объекты и память. Статические и константные элементы классов
3. Перегрузка операций. Перегрузка унарных операций. Перегрузка бинарных операций. Операции арифметического присваивания.
4. Наследование. Понятие базового и производного классов. Доступ к базовому классу. Спецификаторы доступа. Неизменность базового класса. Конструкторы производного

класса. Перегрузка функций для производного класса. Операция разрешения.

5. Иерархия классов. Общее и частное наследование. Уровни наследования. Множественное наследование. Неопределенность при множественном наследовании.
6. Указатели и ссылки на производные типы. Виртуальные функции. Виртуальные деструкторы. Наследование виртуальных функций.
7. Чисто виртуальные функции и абстрактные классы. Сравнение раннего связывания с поздним. Виртуальные базовые классы.
8. Дружественные функции. Дружественные функции как мосты между классами. Дружественные классы.
9. Шаблонные функции. Явно заданная перегрузка обобщенной функции. Перегрузка шаблона функции. Использование стандартных параметров в шаблонных функциях. Ограничения при использовании обобщенных функций.
10. Шаблоны классов. Контекстозависимое имя класса. Создание обобщенного класса безопасного массива. Использование в обобщенных классах аргументов, не являющихся типами. Использование в шаблонных классах аргументов по умолчанию. Явно задаваемые специализации классов.
11. Основы обработки исключительных ситуаций.
25. Класс `string`.
12. Файловый ввод-вывод в стиле C++. Работа с текстовыми файлами. Неформатированный ввод-вывод данных в двоичном режиме. Произвольный доступ. Проверка статуса ввода-вывода
13. Стандартная библиотека шаблонов STL. Состав, основные понятия и определения.
14. Алгоритмы STL.
15. Функциональные объекты и лямбда-функция. Предопределенные функциональные объекты.
16. Последовательные контейнеры.
17. Итераторы. Категории итераторов и их возможности. Специализированные итераторы.
18. Ассоциативные контейнеры.
19. Основы многопоточности: запуск и управление потоками
20. Модель памяти C++ и синхронизация
21. Координация потоков: ожидание и инициализация
22. Отладка многопоточных программ.
23. Тестирование и оценка производительности