

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Панарин Андрей Александрович

Должность: Ректор

Дата подписания: 18.06.2026 12:03:06

Уникальный программный ключ:

a5da3d9896e9d535580e3f9a7da4892154ef8502



Образовательная автономная некоммерческая организация высшего образования

«Московский психолого-социальный университет»

Лицензия № 1478 от 28 мая 2015 г., серия 90Л01 № 0008476 (бессрочная)

Свидетельство государственной аккредитации № 2783 от 07 марта 2018 года, серия 90А01 №0002920 (бессрочно)

Кафедра «Экономики и цифровых технологий»

**Фонд
оценочных средств
по дисциплине: «Программирование и основы алгоритмизации»**

**Уровень высшего образования
БАКАЛАВРИАТ**

**Направление подготовки
38.03.05 Бизнес-информатика**

**Направленность (профиль)
Data Science**

**Квалификация (степень) выпускника
бакалавр**

**Форма обучения
очная**

Москва

2026 год набора

Фонд оценочных средств предназначен для контроля знаний обучающихся по направлению подготовки 38.03.05 Бизнес-информатика, утвержденным приказом Министерство науки и высшего образования Российской Федерации от 29 июля 2020г. № 838» по дисциплине «Программирование и основы алгоритмизации».

Фонд оценочных средств рассмотрен и утвержден на заседании кафедры «Экономики и цифровых технологий»

Протокол № 7 от 22 января 2026 г.

И.о. заведующий кафедрой
«Экономики и цифровых технологий»

И.Ф. Иорданиди

Согласовано:

Декан экономического факультета

М.К. Чистякова

1. Перечень планируемых результатов обучения, соотнесенных с планируемыми результатами освоения основной профессиональной образовательной программы ОПК-2.1; ОПК-3.1; ОПК-3.2

Код и описание компетенции	Код и наименование индикатора достижения	Планируемые результаты обучения по дисциплине
ОПК-2. Способен проводить исследование и анализ рынка информационных систем (ИС) и информационно - коммуникационных технологий (ИКТ), выбирать рациональные решения для управления бизнесом	ОПК-2.1. Исследует и анализирует рынок информационных систем и информационно-коммуникационных технологий	<u>Контактная работа:</u> Лекции Практические занятия Самостоятельная работа
ОПК-3. Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации;	ОПК 3.1. Планирует и организует процесс создания ИТ продуктов и услуг	<u>Контактная работа:</u> Лекции Практические занятия Самостоятельная работа
	ОПК-3.2. Руководит, контролирует и разрабатывает алгоритмы и программы практической реализации продуктов и услуг в сфере ИКТ.	<u>Контактная работа:</u> Лекции Практические занятия Самостоятельная работа

ТИПОВЫЕ ИНСТРУКЦИИ ПО ВЫПОЛНЕНИЮ ТЕСТОВЫХ ЗАДАНИЙ

Тип задания	Инструкция
Задание закрытого типа с выбором одного или нескольких ответов	Прочитайте текст и выберите правильный ответ (Если несколько ответов, то прочитайте текст и выберите правильные ответы)
Задание закрытого типа на установление соответствия	Прочитайте текст и установите соответствие
Задания закрытого типа на установление правильной последовательности	Прочитайте текст и установите последовательность
Задания комбинированного типа с выбором одного верного ответа с обоснованием	Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа
Задания комбинированного типа с выбором нескольких ответов с обоснованием	Прочитайте текст, выберите правильные ответы и запишите аргументы, обосновывающие выбор ответов
Задания с развернутым ответом	Прочитайте текст и запишите развернутый обоснованный ответ

СИСТЕМА ОЦЕНИВАНИЯ ЗАДАНИЙ

Тип задания	Указания по оцениванию	Результат оценивания
Задание закрытого типа на установление соответствия	Задание закрытого типа на установление соответствия считается верным если правильно установлены все соответствия	Верно/неверно
Задания закрытого типа на установление правильной последовательности	Задание закрытого типа на установление правильной последовательности считается верным если правильно указываются все последовательности	Верно/неверно
Задания комбинированного типа с выбором одного верного ответа с обоснованием	Задание комбинированного типа с выбором одного верного ответа из предложенных с обоснованием считается верным если правильно указан ответ и приведены корректные аргументы, используемые при выборе ответа.	Верно/неверно
Задания комбинированного типа с выбором нескольких ответов с обоснованием	Задание комбинированного типа с выбором нескольких ответов из предложенных с обоснованием считается верным если правильно указаны ответы и приведены корректные аргументы, используемые при выборе ответа.	Верно/неверно
Задания открытого типа с развернутым ответом	Задания открытого типа с развернутым ответом считается верным, если ответ совпадает с эталоном по содержанию и полноте.	Верно/неверно

3. Контрольные задания или иные материалы, необходимые для процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения основной профессиональной образовательной программы

3.1. Задания для проведения текущего контроля обучающихся

Содержание вопроса	Компетенции	Уровень освоения
<i>Прочитайте текст и выберите правильный ответ</i> 1. Что такое алгоритм? а) Программа на языке программирования б) Последовательность действий для решения задачи с) Схема блок-диаграммы д) Результат работы программы	ОПК-2.1. Исследует и анализирует рынок информационных систем и информационно-коммуникационных технологий	Базовый 1-3 минуты

2. Какой из способов записи алгоритмов является наиболее формальным? a) Словесный b) Графический (блок-схема) c) Псевдокод d) На языке программирования 3. Что обозначает блок "Начало" в блок-схеме? a) Конец выполнения алгоритма b) Начало выполнения алгоритма c) Ввод данных d) Вывод данных 4. Что обозначает блок "Конец" в блок-схеме? a) Начало выполнения алгоритма b) Конец выполнения алгоритма c) Ввод данных d) Вывод данных 5. Какой блок используется для проверки условия в блок-схеме? a) Прямоугольник b) Ромб c) Параллелограмм d) Овал 6. Какой блок используется для вывода данных в блок-схеме? a) Прямоугольник b) Ромб c) Параллелограмм d) Овал 7. Какой блок используется для ввода данных в блок-схеме? a) Прямоугольник b) Ромб c) Параллелограмм d) Овал 8. Что такое цикл в программировании? a) Последовательность команд, выполняемых один раз b) Последовательность команд, выполняемых многократно c) Условный оператор d) Функция 9. Какие типы циклов вы знаете?		
--	--	--

a) Цикл с предусловием b) Цикл с постусловием c) Цикл с параметром d) Все вышеперечисленные 10. Что такое условный оператор? a) Оператор, выполняющий арифметические действия b) Оператор, выполняющий логические действия c) Оператор, выполняющий ввод данных d) Оператор, выполняющий вывод данных		
<i>Прочитайте текст и выберите правильный ответ</i> 11. Какой оператор используется для сравнения в условном операторе? a) = b) == c) != d) b) и c) 12. Что такое массив? a) Переменная, хранящая одно значение b) Набор переменных одного типа, хранящих несколько значений c) Функция d) Структура 13. Как обращаются к элементам массива? a) По имени b) По индексу c) По значению d) По типу 14. Что такое функция? a) Переменная b) Блок кода, выполняющий определенную задачу c) Массив d) Цикл 15. Что такое рекурсия? a) Повторение кода b) Функция, вызывающая сама себя c) Цикл d) Массив 16. Что такое структура данных? a) Способ организации данных в памяти компьютера b) Программа c) Алгоритм	ОПК 3.1. Планирует и организует процесс создания ИТ продуктов и услуг	Повышен ный 3-5 минут

d) Функция 17. Какие виды структур данных вы знаете? a) Массивы b) Связные списки c) Деревья d) Все вышеперечисленные 18. Что такое стек? a) Структура данных "первый вошел – последний вышел" (LIFO) b) Структура данных "первый вошел – первый вышел" (FIFO) c) Массив d) Связный список 19. Что такое очередь? a) Структура данных "первый вошел – последний вышел" (LIFO) b) Структура данных "первый вошел – первый вышел" (FIFO) c) Массив d) Связный список 20. Что такое сортировка? a) Поиск элемента в массиве b) Упорядочение элементов массива c) Удаление элемента из массива d) Добавление элемента в массив		
Прочитайте текст и запишите развернутый обоснованный ответ Кейс 1: Нахождение максимального элемента в массиве - Условие: Найти максимальный элемент в одномерном массиве целых чисел. - Алгоритм: - Инициализировать переменную <code>max_element</code> первым элементом массива. - Пройтись по всем элементам массива, сравнивая каждый с <code>max_element</code>. - Если элемент больше <code>max_element</code>, обновить <code>max_element</code>. - После обхода всего массива, <code>max_element</code> будет содержать максимальный элемент. - Код:	ОПК-2.1. Исследует и анализирует рынок информационных систем и информационно-коммуникационных технологий ОПК 3.1. Планирует и организует процесс создания ИТ продуктов и услуг ОПК-3.2. Руководит, контролирует и разрабатывает алгоритмы и программы практической реализации продуктов	Высокий 5-10 минут

<pre>def find_max(arr): """Находит максимальный элемент в массиве.""" if not arr: return None # Обработка пустого массива max_element = arr[0] for element in arr: if element > max_element: max_element = element return max_element my_array = [1, 5, 2, 8, 3] max_value = find_max(my_array) print(f"Максимальный элемент: {max_value}")</pre> Кейс 2: Подсчет факториала числа • Условие: Вычислить факториал заданного неотрицательного целого числа. • Алгоритм: Использовать итеративный или рекурсивный подход. Итеративный подход более эффективен для больших чисел. • Код (итеративный): <pre>def factorial_iterative(n): """Вычисляет факториал итеративно.""" if n < 0: return None # Обработка отрицательных чисел elif n == 0: return 1 else: result = 1 for i in range(1, n + 1): result *= i return result print(factorial_iterative(5)) # Вывод: 120</pre> Кейс 3: Поиск элемента в отсортированном массиве (бинарный поиск) • Условие: Найти индекс заданного элемента в отсортированном массиве. • Алгоритм: Использовать бинарный поиск. • Код: <pre>def binary_search(arr, target): """Реализует бинарный поиск.""" low = 0 high = len(arr) - 1</pre>	и услуг в сфере ИКТ.	
---	----------------------	--

```

while low <= high:
    mid = (low + high) // 2
    if arr[mid] == target:
        return mid
    elif arr[mid] < target:
        low = mid + 1
    else:
        high = mid - 1
return -1 # Элемент не найден

```

```

sorted_array = [2, 5, 7, 8, 11, 12]
target_element = 11
index = binary_search(sorted_array, target_element)
print(f"Индекс элемента {target_element}: {index}")

```

Кейс 4: Обращение строки

- Условие: Обратить строку.
- Алгоритм: Использовать срезы или итеративный подход.
- Код (срезы):

```

def reverse_string(s):
    """Обращает строку используя срезы."""
    return s[::-1]

```

```

string = "hello"
reversed_string = reverse_string(string)
print(f"Обращенная строка: {reversed_string}")

```

Кейс 5: Проверка на палиндром

- Условие: Проверить, является ли строка палиндромом (читается одинаково слева направо и справа налево).
- Алгоритм: Сравнить строку с её обращенной версией.
- Код:

```

def is_palindrome(s):
    """Проверяет, является ли строка палиндромом."""
    processed_s = ''.join(c for c in s.lower() if c.isalnum())
    #обработка пробелов и регистра
    return processed_s == processed_s[::-1]

```

```

string = "A man, a plan, a canal: Panama"
print(f"Строка '{string}' является палиндромом: {is_palindrome(string)}")

```

3.2.2. Задания для проведения промежуточной аттестации обучающихся

Содержание вопроса	Компетенции	Уровень освоения
<i>Прочитайте текст и выберите правильный ответ</i> 21. Какие методы сортировки вы знаете? а) Сортировка пузырьком б) Сортировка выбором в) Сортировка вставками г) Все вышеперечисленные 22. Что такое поиск? а) Упорядочение элементов массива б) Поиск элемента в массиве в) Удаление элемента из массива г) Добавление элемента в массив 23. Какие методы поиска вы знаете? а) Линейный поиск б) Бинарный поиск в) Хеширование г) Все вышеперечисленные 24. Что такое отладка программы? а) Написание программы б) Исправление ошибок в программе в) Компиляция программы г) Запуск программы 25. Что такое компиляция? а) Перевод программы с языка высокого уровня на машинный код б) Запуск программы в) Написание программы г) Отладка программы 26. Что такое интерпретация? а) Перевод программы с языка высокого уровня на машинный код б) Построчное выполнение программы в) Запуск программы г) Отладка программы 27. Что такое переменная? а) Перевод программы с языка высокого уровня на машинный код б) Построчное выполнение программы в) Запуск программы	ОПК-2.1. Исследует и анализирует рынок информационных систем и информационно-коммуникационных технологий	Базовый 1-3 минуты

d) Отладка программы 27. Что такое переменная? a) Константа b) Место в памяти компьютера для хранения данных c) Функция d) Алгоритм 28. Что такое оператор присваивания? a) = b) == c) != d) > 29. Что такое тип данных? a) Название переменной b) Характеристика данных, хранящихся в переменной c) Значение переменной d) Адрес переменной в памяти 30. Какие типы данных вы знаете? a) Целочисленные b) Вещественные c) Символьные d) Все вышеперечисленные		
<i>Прочитайте текст и выберите правильный ответ</i> 31. Что такое комментарии в коде? a) Части кода, которые игнорируются компилятором/интерпретатором b) Описание кода, предназначенное для программиста c) Части кода, которые всегда выполняются d) a) и b) 32. Что такое модульность в программировании? a) Разбиение программы на отдельные, независимые блоки b) Написание программы в одном файле c) Использование только одного цикла в программе d) Использование только одной функции в программе 33. Что такое абстракция данных? a) Подробное описание данных b) Соккрытие деталей реализации данных c) Использование массивов d) Использование структур	ОПК 3.1. Планирует и организует процесс создания ИТ продуктов и услуг	Повышенный 3-5 минут

34. Что такое инкапсуляция?

- a) Объединение данных и методов работы с ними в единую сущность
- b) Разделение данных и методов работы с ними
- c) Использование массивов
- d) Использование структур

35. Что такое наследование (в объектно-ориентированном программировании)?

- a) Создание нового класса на основе существующего
- b) Создание нового массива
- c) Создание новой функции
- d) Создание новой структуры

36. Что такое полиморфизм (в объектно-ориентированном программировании)?

- a) Использование одного имени для разных методов
- b) Использование разных имен для разных методов
- c) Наследование
- d) Инкапсуляция

37. Что такое объектно-ориентированное программирование?

- a) Процедурный подход к программированию
- b) Парадигма программирования, основанная на использовании объектов
- c) Функциональный подход к программированию
- d) Логическое программирование

38. Какие основные принципы объектно-ориентированного программирования вы знаете?

- a) Инкапсуляция
- b) Наследование
- c) Полиморфизм
- d) Все вышеперечисленные

39. Что такое отладчик (debugger)?

- a) Программа для написания кода
- b) Программа для запуска программ
- c) Программа для поиска и исправления ошибок в коде
- d) Программа для компиляции кода

40. Что такое тестирование программы? - Процесс проверки работоспособности программы - Процесс написания программы - Процесс компиляции программы - Процесс отладки программы		
Прочитайте текст и запишите развернутый обоснованный ответ Кейс 6: Сортировка массива пузырьком - Условие: Отсортировать массив целых чисел методом пузырька. - Алгоритм: Многократное прохождение по массиву, сравнение соседних элементов и их обмен, если они находятся в неправильном порядке. - Код: <pre>def bubble_sort(arr): """Сортирует массив методом пузырька.""" n = len(arr) for i in range(n): for j in range(0, n-i-1): if arr[j] > arr[j+1]: arr[j], arr[j+1] = arr[j+1], arr[j] return arr my_array = [64, 34, 25, 12, 22, 11, 90] sorted_array = bubble_sort(my_array) print(f"Отсортированный массив: {sorted_array}")</pre> Кейс 7: Поиск чисел Фибоначчи - Условие: Вывести n первых чисел Фибоначчи. - Алгоритм: Использовать итеративный или рекурсивный подход. Итеративный подход предпочтительнее для больших n из-за эффективности. - Код (итеративный): <pre>def fibonacci(n): """Выводит n первых чисел Фибоначчи.""" a, b = 0, 1 if n <= 0: return [] elif n == 1: return [0] else: result = [0, 1]</pre>	ОПК-2.1. Исследует и анализирует рынок информационных систем и информационно-коммуникационных технологий ОПК 3.1. Планирует и организует процесс создания ИТ продуктов и услуг ОПК-3.2. Руководит, контролирует и разрабатывает алгоритмы и программы практической реализации продуктов и услуг в сфере ИКТ.	Высокий 5-10 минут

<pre> for i in range(2, n): c = a + b result.append(c) a, b = b, c return result print(fibonacci(10)) # Вывод: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34] Кейс 8: Реализация стека с помощью списка • Условие: Реализовать структуру данных "стек" (LIFO - Last In, First Out) с помощью списка. Должны быть методы push (добавить элемент) и pop (извлечь элемент). • Алгоритм: Использовать список Python, где добавление в конец списка соответствует push, а удаление последнего элемента - pop. • Код: class Stack: def __init__(self): self.items = [] def push(self, item): self.items.append(item) def pop(self): if not self.is_empty(): return self.items.pop() else: return None def is_empty(self): return len(self.items) == 0 def peek(self): #добавим метод для просмотра верхушки стека if not self.is_empty(): return self.items[-1] else: return None my_stack = Stack() my_stack.push(1) my_stack.push(2) my_stack.push(3) print(my_stack.pop()) # Вывод: 3 print(my_stack.peek()) # Вывод: 2 </pre>		
---	--	--

print(my_stack.is_empty()) # Вывод: False		
---	--	--

3.3. Вопросы к зачету (промежуточная аттестация), формирование компетенций (ОПК-3.2. , ОПК-3.1, ОПК-2.1.)

1. Программа, алгоритм, формы записи алгоритмов.
2. Языки программирования.
3. Обработка программы, транслятор, компилятор.
4. Основные логические операции.
5. Законы булевой алгебры.
6. Алфавит, структура программы.
7. Стандартные типы данных.
8. Типы данных, определяемые программистом.
9. Создание элементарных консольных приложений.
10. Операторы присваивания, ввода-вывода, составной оператор
11. Оператор условного перехода, оператор множественного выбора,
12. Операторы циклов.
13. Выполнение задачи с использованием ПО Turbo Delphi Explorer для разработки программ на современных языках программирования.
14. Массивы.
15. Одномерные, двумерные и многомерные.
16. Определение максимума и минимума.
17. Методы сортировки.
18. Обработка двумерных массивов.
19. Массивы символов и строки.
20. Функции работы со строками.
21. Выполнение задачи с использованием ПО Turbo Delphi Explorer для документирования
22. Процедуры и функции.
23. Передача параметров.
24. Области видимости.
25. Глобальные и локальные переменные.
26. Основы объектно-ориентированного программирования.
27. Свойства и методы.
28. Компоненты.
29. Форма, свойства формы.
30. Создание элементарных оконных приложений.
31. Выполнение задачи с использованием ПО Turbo Delphi Explorer для разработки программ на современных языках программирования и документирования созданных программ.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения основной профессиональной образовательной программы

№ п/п	Форма контроля/ коды	Процедура оценивания	Шкала и критерии оценки, балл
-------	----------------------	----------------------	-------------------------------

	оцениваемых компетенций		
1.	Зачет ОПК-3.2. , ОПК-3.1, ОПК-2.1.	При подготовке к зачету необходимо ориентироваться на конспекты лекций, рабочую программу дисциплины, нормативную, основную и дополнительную учебную литературу. Основное в подготовке к сдаче зачета - это повторение всего материала дисциплины. При подготовке к сдаче зачета обучающийся весь объем работы должен распределять равномерно по дням, отведенным для подготовки к зачету, контролировать каждый день выполнение намеченной работы. В период подготовки к зачету обучающийся вновь обращается к уже изученному (пройденному) учебному материалу. Подготовка обучающегося к зачету включает в себя три этапа: самостоятельная работа в течение семестра; непосредственная подготовка в дни, предшествующие зачету по темам курса; подготовка к ответу на задания, содержащиеся в вопросах (тестах) зачета. Зачет проводится по вопросам (тестам), охватывающим весь пройденный материал дисциплины, включая вопросы, отведенные для самостоятельного изучения.	1)«зачтено» - правильность ответов на вопросы билета (верное, четкое, достаточно глубокое изложение идей, понятий, фактов, нормативно-правового материала и т.п.) и правильное разрешение задачи; полнота и лаконичность ответа; степень использования и понимания научных и нормативных источников; умение связывать теорию с практикой; логика и аргументированность изложения материала; грамотное комментирование, приведение примеров, аналогий; культура речи; 2)«не зачтено» предполагает, что обучающимся либо не дан ответ на вопрос и (или) не решена предложенная задача, либо обучающийся не знает основных понятий, не может определить предмет дисциплины.
2.	Тестирование ОПК-3.2. , ОПК-3.1, ОПК-2.1.	Полнота знаний теоретического контролируемого материала. Количество правильных ответов	«отлично» - процент правильных ответов = > 90%; «хорошо» - процент правильных ответов = > 70%; «удовлетворительно» - процент правильных ответов = > 50%; «неудовлетворительно» - процент правильных ответов < 50%.

С целью определения уровня овладения компетенциями, закрепленными за дисциплиной, в заданные преподавателем сроки проводится текущий и промежуточный контроль знаний, умений и навыков каждого обучающегося. Все виды текущего контроля осуществляются на практических занятиях. Исключение составляет устный опрос, который может проводиться в начале или конце лекции в течение 15-20 мин. с целью закрепления знаний терминологии по дисциплине. При оценке компетенций принимается во внимание формирование профессионального мировоззрения, определенного уровня включённости в занятия, рефлексивные навыки, владение изучаемым материалом.

Процедура оценивания компетенций обучающихся основана на следующих стандартах:

1. Периодичность проведения оценки.
2. Многоступенчатость: оценка (как преподавателем, так и обучающимися группы) и самооценка обучающегося, обсуждение результатов и комплекс мер по устранению недостатков.
3. Единство используемой технологии для всех обучающихся, выполнение условий сопоставимости результатов оценивания.
4. Соблюдение последовательности проведения оценки.

Текущая аттестация обучающихся. Текущая аттестация обучающихся по дисциплине проводится в соответствии с локальными нормативными актами ОАНО ВО МПСУ и является обязательной.

Текущая аттестация проводится в форме опроса и контрольных мероприятий по оцениванию фактических результатов обучения обучающихся и осуществляется преподавателем дисциплины.

Объектами оценивания выступают:

- 1) учебная дисциплина (активность на занятиях, своевременность выполнения различных видов заданий, посещаемость всех видов занятий по аттестуемой дисциплине);
- 2) степень усвоения теоретических знаний в качестве «ключей анализа»;
- 3) уровень овладения практическими умениями и навыками по всем видам учебной работы;
- 4) результаты самостоятельной работы (изучение книг из списка основной и дополнительной литературы).

Активность обучающегося на занятиях оценивается на основе выполненных обучающимся работ и заданий, предусмотренных данной рабочей программой дисциплины.

Кроме того, оценивание обучающегося проводится на текущем контроле по дисциплине. Оценивание обучающегося на контрольной неделе проводится преподавателем независимо от наличия или отсутствия обучающегося (по уважительной или неуважительной причине) на занятии. Оценка носит комплексный характер и учитывает достижения обучающегося по основным компонентам учебного процесса за текущий период.

Оценивание обучающегося носит комплексный характер и учитывает достижения обучающегося по основным компонентам учебного процесса за текущий период с выставлением оценок в ведомости.

Промежуточная аттестация обучающихся. Промежуточная аттестация проводится в соответствии с локальными нормативными актами ОАНО ВО «МПСУ» и является обязательной.

Промежуточная аттестация проводится в соответствии с учебным планом в виде **зачета** в период зачётно-экзаменационной сессии в соответствии с графиком проведения.

Обучающиеся допускаются к зачету в случае выполнения ими учебного плана по дисциплине: выполнения всех заданий и мероприятий, предусмотренных программой дисциплины.

Оценка знаний обучающегося на зачёте определяется его учебными достижениями и результатами текущего контроля знаний и выполнением им заданий.

Основой для определения оценки служит уровень усвоения обучающимися материала, предусмотренного данной рабочей программой дисциплины.

5. Методические указания для обучающихся по освоению дисциплины

Вид деятельности	Методические указания по организации деятельности студента
Лекция	Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения; помечать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, пометить и попытаться найти ответ в рекомендуемой литературе. Если

	самостоятельно не удастся разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, на практическом занятии.
Практические занятия	Проработка рабочей программы, уделяя особое внимание целям и задачам, структуре и содержанию дисциплины. Конспектирование источников. Работа с конспектом лекций, подготовка ответов к контрольным вопросам, просмотр рекомендуемой литературы, работа с текстом. Прослушивание аудио- и видеозаписей по заданной теме, решение расчетно-графических заданий, решение задач по алгоритму и др.
Индивидуальные задания	Знакомство с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующихся для запоминания и являющихся основополагающими в этой теме. Составление аннотаций к прочитанным литературным источникам и др.
Самостоятельная работа	Самостоятельная работа проводится с целью: систематизации и закрепления полученных теоретических знаний и практических умений обучающихся; углубления и расширения теоретических знаний студентов; формирования умений использовать нормативную, правовую, справочную документацию, учебную и специальную литературу; развития познавательных способностей и активности обучающихся: творческой инициативы, самостоятельности, ответственности, организованности; формирование самостоятельности мышления, способностей к саморазвитию, совершенствованию и самоорганизации; формирования профессиональных компетенций; развитию исследовательских умений обучающихся. Формы и виды самостоятельной работы: чтение основной и дополнительной литературы – самостоятельное изучение материала по рекомендуемым литературным источникам; работа с библиотечным каталогом, самостоятельный подбор необходимой литературы; работа со словарем, справочником; поиск необходимой информации в сети Интернет; конспектирование источников; реферирование источников; составление аннотаций к прочитанным литературным источникам; составление рецензий и отзывов на прочитанный материал; составление обзора публикаций по теме; составление и разработка терминологического словаря; составление хронологической таблицы; составление библиографии (библиографической картотеки); подготовка к различным формам текущей и промежуточной аттестации (к тестированию, зачету, экзамену); выполнение домашних контрольных работ; самостоятельное выполнение практических заданий репродуктивного типа (ответы на вопросы, тесты; выполнение творческих заданий). Технология организации самостоятельной работы обучающихся включает использование информационных и материально-технических ресурсов образовательного учреждения: библиотеку с читальным залом, укомплектованную в соответствии с существующими нормами; учебно-методическую базу учебных кабинетов, лабораторий и зала кодификации; компьютерные классы с возможностью работы в сети Интернет; аудитории (классы) для консультационной деятельности; учебную и учебно-методическую литературу, разработанную с учетом увеличения доли самостоятельной работы студентов, и иные методические материалы. Перед выполнением обучающимися внеаудиторной самостоятельной работы преподаватель проводит консультирование по выполнению задания, который включает цель задания, его содержания, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. Во время выполнения обучающимися внеаудиторной самостоятельной работы и при необходимости преподаватель может проводить индивидуальные и групповые консультации. Самостоятельная работа может осуществляться индивидуально или группами обучающихся в зависимости от цели, объема, конкретной тематики

	самостоятельной работы, уровня сложности, уровня умений обучающихся. Контроль самостоятельной работы предусматривает: • соотнесение содержания контроля с целями обучения; объективность контроля; • валидность контроля (соответствие предъявляемых заданий тому, что предполагается проверить); • дифференциацию контрольно-измерительных материалов. Формы контроля самостоятельной работы: • просмотр и проверка выполнения самостоятельной работы преподавателем; • организация самопроверки, • взаимопроверки выполненного задания в группе; обсуждение результатов выполненной работы на занятии; • проведение письменного опроса; • проведение устного опроса; • организация и проведение индивидуального собеседования; организация и проведение собеседования с группой; • защита отчетов о проделанной работе.
Опрос	Опрос - это средство контроля, организованное как специальная беседа преподавателя с обучающимся на темы, связанные с изучаемой дисциплиной, и рассчитанное на выявление объема знаний по определенному разделу, теме, проблеме и т.п. Проблематика, выносимая на опрос определена в заданиях для самостоятельной работы обучающегося, а также может определяться преподавателем, ведущим семинарские занятия. Во время проведения опроса обучающийся должен уметь обсудить с преподавателем соответствующую проблематику на уровне диалога.
Коллоквиум	Коллоквиум (от латинского colloquium – разговор, беседа) – одна из форм учебных занятий, беседа преподавателя с учащимися на определенную тему из учебной программы. Цель проведения коллоквиума состоит в выяснении уровня знаний, полученных учащимися в результате прослушивания лекций, посещения семинаров, а также в результате самостоятельного изучения материала. В рамках поставленной цели решаются следующие задачи: • выяснение качества и степени понимания учащимися лекционного материала; • развитие и закрепление навыков выражения учащимися своих мыслей; • расширение вариантов самостоятельной целенаправленной подготовки учащихся; • развитие навыков обобщения различных литературных источников; • предоставление возможности учащимся сопоставлять разные точки зрения по рассматриваемому вопросу. В результате проведения коллоквиума преподаватель должен иметь представление: • качества лекционного материала; • сильных и слабых сторонах своей методики чтения лекций; • сильных и слабых сторонах своей методики проведения семинарских занятий; • об уровне самостоятельной работы учащихся; • об умении обучающихся вести дискуссию и доказывать свою точку зрения; • степени эрудированности учащихся; • степени индивидуального освоения материала конкретными обучающимися.

	В результате проведения коллоквиума обучающийся должен иметь представление: • об уровне своих знаний по рассматриваемым вопросам в соответствии с требованиями преподавателя и относительно других студентов группы; • недостатках самостоятельной проработки материала; • своем умении излагать материал; • своем умении вести дискуссию и доказывать свою точку зрения. В зависимости от степени подготовки группы можно использовать разные подходы к проведению коллоквиума. В случае, если большинство группы с трудом воспринимает содержание лекций и на практических занятиях демонстрирует недостаточную способность активно оперировать со смысловыми единицами и терминологией курса, то коллоквиум можно разделить на две части. Сначала преподаватель излагает базовые понятия, содержащиеся в программе. Это должно занять не более четверти занятия. Остальные три четверти необходимо посвятить дискуссии, в ходе которой обучающиеся должны убедиться и, главное, убедить друг друга в обоснованности и доказательности полученного видения вопроса и его соответствия реальной практике. Если же преподаватель имеет дело с более подготовленной, самостоятельно думающей и активно усваивающей смысловые единицы и терминологию курса аудиторией, то коллоквиум необходимо провести так, чтобы сами обучающиеся сформулировали изложенные в программе понятия, высказали несовпадающие точки зрения и привели практические примеры. За преподавателем остается роль модератора (ведущего дискуссии), который в конце «лишь» суммирует совместно полученные результаты.
Тестирование	Контроль в виде тестов может использоваться после изучения каждой темы курса. Итоговое тестирование можно проводить в форме: • компьютерного тестирования, т.е. компьютер произвольно выбирает вопросы из базы данных по степени сложности; • письменных ответов, т.е. преподаватель задает вопрос и дает несколько вариантов ответа, а обучающийся на отдельном листе записывает номера вопросов и номера соответствующих ответов. Для достижения большей достоверности результатов тестирования следует строить текст так, чтобы у обучающихся было не более 40 – 50 секунд для ответа на один вопрос. Итоговый тест должен включать не менее 60 вопросов по всему курсу. Значит, итоговое тестирование займет целое занятие.
Подготовка к зачёту	При подготовке к зачёту необходимо ориентироваться на конспекты лекций, рекомендуемую литературу и др. Основное в подготовке к сдаче зачёта – это повторение всего материала дисциплины, по которому необходимо сдавать зачёт. При подготовке обучающийся весь объем работы должен распределять равномерно по дням, отведенным для подготовки, контролировать каждый день выполнение намеченной работы. Подготовка включает в себя три этапа: • самостоятельная работа в течение семестра; • непосредственная подготовка в дни, предшествующие зачёту по темам курса; • подготовка к ответу на задания, содержащиеся в билетах (тестах) зачёта. Для успешной сдачи зачёта обучающиеся должны принимать во внимание, что: • все основные вопросы, указанные в рабочей программе, нужно знать, понимать их смысл и уметь его разъяснить; • указанные в рабочей программе формируемые профессиональные компетенции в результате освоения дисциплины должны быть продемонстрированы студентом; • семинарские занятия способствуют получению более высокого уровня

	знаний и, как следствие, успешной сдаче зачёта; • готовиться к зачёту необходимо начинать с первой лекции и первого семинара.
--	--



Образовательная автономная некоммерческая организация высшего образования
«Московский психолого-социальный университет»
Лицензия № 1478 от 28 мая 2015 г., серия 90Л01 № 0008476 (бессрочная)
Свидетельство государственной аккредитации № 2783 от 07 марта 2018 года, серия 90А01 №0002920 (бессрочно)

КЛЮЧИ ПРАВИЛЬНЫХ ОТВЕТОВ К ФОНДУ ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине: «Программирование и основы алгоритмизации»

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки
38.03.05 Бизнес-информатика

Направленность (профиль)
Data Science

Квалификация (степень) выпускника
бакалавр

Форма обучения
очная

Москва
2025 год набора

ОТВЕТЫ К ТЕСТУ:

1. b
2. d
3. b
4. b
5. b
6. c
7. c
8. b
9. d
10. b
11. d
12. b
13. b
14. b
15. b
16. a
17. d
18. a
19. b
20. b
21. d
22. b
23. d
24. b
25. a
26. b
27. b
28. a
29. b
30. d
31. d
32. a
33. b
34. a
35. a
36. a
37. b
38. d
39. c
40. a

Каждый кейс включает в себя условие, алгоритм решения и пример кода (на Python).

Кейс 1: Нахождение максимального элемента в массиве

- Условие: Найти максимальный элемент в одномерном массиве целых чисел.
- Алгоритм:
 1. Инициализировать переменную `max_element` первым элементом массива.
 2. Пройтись по всем элементам массива, сравнивая каждый с `max_element`.
 3. Если элемент больше `max_element`, обновить `max_element`.
 4. После обхода всего массива, `max_element` будет содержать максимальный элемент.
- Код:

```
def find_max(arr):  
    """Находит максимальный элемент в массиве."""  
    if not arr:  
        return None # Обработка пустого массива  
    max_element = arr[0]  
    for element in arr:  
        if element > max_element:  
            max_element = element  
    return max_element  
  
my_array = [1, 5, 2, 8, 3]  
max_value = find_max(my_array)  
print(f"Максимальный элемент: {max_value}")
```

Кейс 2: Подсчет факториала числа

- Условие: Вычислить факториал заданного неотрицательного целого числа.
- Алгоритм: Использовать итеративный или рекурсивный подход. Итеративный подход более эффективен для больших чисел.
- Код (итеративный):

```
def factorial_iterative(n):  
    """Вычисляет факториал итеративно."""  
    if n < 0:  
        return None # Обработка отрицательных чисел  
    elif n == 0:  
        return 1  
    else:  
        result = 1  
        for i in range(1, n + 1):  
            result *= i  
        return result  
  
print(factorial_iterative(5)) # Вывод: 120
```

Кейс 3: Поиск элемента в отсортированном массиве (бинарный поиск)

- Условие: Найти индекс заданного элемента в отсортированном массиве.
- Алгоритм: Использовать бинарный поиск.
- Код:

```
def binary_search(arr, target):
    """Реализует бинарный поиск."""
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1 # Элемент не найден
```

```
sorted_array = [2, 5, 7, 8, 11, 12]
target_element = 11
index = binary_search(sorted_array, target_element)
print(f"Индекс элемента {target_element}: {index}")
```

Кейс 4: Обращение строки

- Условие: Обратить строку.
- Алгоритм: Использовать срезы или итеративный подход.
- Код (срезы):

```
def reverse_string(s):
    """Обращает строку используя срезы."""
    return s[::-1]

string = "hello"
reversed_string = reverse_string(string)
print(f"Обращенная строка: {reversed_string}")
```

Кейс 5: Проверка на палиндром

- Условие: Проверить, является ли строка палиндромом (читается одинаково слева направо и справа налево).
- Алгоритм: Сравнить строку с её обращенной версией.
- Код:

```
def is_palindrome(s):
    """Проверяет, является ли строка палиндромом."""
    processed_s = ''.join(c for c in s.lower() if c.isalnum()) #обработка пробелов и регистра
    return processed_s == processed_s[::-1]
```

```
string = "A man, a plan, a canal: Panama"
print(f"Строка '{string}' является палиндромом: {is_palindrome(string)}")
```

Кейс 6: Сортировка массива пузырьком

- Условие: Отсортировать массив целых чисел методом пузырька.
- Алгоритм: Многократное прохождение по массиву, сравнение соседних элементов и их обмен, если они находятся в неправильном порядке.
- Код:

```
def bubble_sort(arr):
    """Сортирует массив методом пузырька."""
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

my_array = [64, 34, 25, 12, 22, 11, 90]
sorted_array = bubble_sort(my_array)
print(f"Отсортированный массив: {sorted_array}")
```

Кейс 7: Поиск чисел Фибоначчи

- Условие: Вывести n первых чисел Фибоначчи.
- Алгоритм: Использовать итеративный или рекурсивный подход. Итеративный подход предпочтительнее для больших n из-за эффективности.
- Код (итеративный):

```
def fibonacci(n):
    """Выводит n первых чисел Фибоначчи."""
    a, b = 0, 1
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    else:
        result = [0, 1]
        for i in range(2, n):
            c = a + b
            result.append(c)
            a, b = b, c
        return result

print(fibonacci(10)) # Вывод: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Кейс 8: Реализация стека с помощью списка

- Условие: Реализовать структуру данных "стек" (LIFO - Last In, First Out) с помощью списка. Должны быть методы push (добавить элемент) и pop (извлечь элемент).
- Алгоритм: Использовать список Python, где добавление в конец списка соответствует push, а удаление последнего элемента - pop.
- Код:

```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def is_empty(self):
        return len(self.items) == 0

    def peek(self): #добавим метод для просмотра верхушки стека
        if not self.is_empty():
            return self.items[-1]
        else:
            return None

my_stack = Stack()
my_stack.push(1)
my_stack.push(2)
my_stack.push(3)
print(my_stack.pop()) # Вывод: 3
print(my_stack.peek()) # Вывод: 2
print(my_stack.is_empty()) # Вывод: False
```