**Правительство Ленинградской области**

**Комитет общего и профессионального образования Ленинградской области**

**Автономное образовательное учреждение высшего образования Ленинградской области**

**«ГОСУДАРСТВЕННЫЙ ИНСТИТУТ ЭКОНОМИКИ, ФИНАНСОВ, ПРАВА И ТЕХНОЛОГИЙ»**

**(АОУ ВО ЛО «ГИЭФПТ»)**

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ**

«ОП.04. Основы алгоритмизации и программирования»

**Уровень профессионального образования**

Среднее профессиональное образование

Образовательная программа

подготовки специалистов среднего звена

Специальность 09.02.07 Информационные системы и программирование

Квалификация выпускника

Программист

Форма обучения: очная

Гатчина

2024

Фонд оценочных средств для проведения процедур внутренней и внешней оценки качества образовательной деятельности по дисциплине «ОП.04 Основы алгоритмизации и программирования» разработан на основе Федерального государственного образовательного стандарта по направлению подготовки 09.02.07 Информационные системы и программирование.

Организация-разработчик: АОУ ВО ЛО «Государственный институт экономики, финансов, права и технологий»

Разработчик(и): преподаватель, В. С. Лебедев

**1 Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Шкала оцени­вания** | **Планируемые результаты обучения** | **Критерии оценивания результатов обучения** | | | |
| **Оценка «неудовлетво­рительно» / «незачет»** | **Оценка «удовлетвори­тельно» / «зачтено»** | **Оценка «хорошо» / «зачтено»** | **Оценка «отлично» / «зачтено»** |
| **ОК 1 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам** | | | | | |
| **Описание показателей и критериев оценивания компетенций ОК-1** | Знает:   * актуальный профессиональный и социальный контекст, в котором приходится работать и жить * структура плана для решения задач, алгоритмы выполнения работ в профессиональной и смежных областях * основные источники информации и ресурсы для решения задач и/или проблем в профессиональном и/или социальном контексте * методы работы в профессиональной и смежных сферах * порядок оценки результатов решения задач профессиональной деятельности | Не знает. Допускает грубые ошибки | Демонстрирует частичные знания без грубых ошибок | Знает достаточно в базовом объеме | Демонстрирует высокий уровень знаний |
| Умеет:   * распознавать задачу и/или проблему в профессиональном и/или социальном контексте, анализировать и выделять её составные части * определять этапы решения задачи, составлять план действия, реализовывать составленный план, определять необходимые ресурсы * выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы * владеть актуальными методами работы в профессиональной и смежных сферах * оценивать результат и последствия своих действий (самостоятельно или с помощью наставника) | Не умеет. Демонстрирует частичные умения, допуская грубые ошибки | Демонстрирует частичные умения без грубых ошибок | Умеет применять знания на практике в базовом объеме | Демонстрирует высокий уровень умений |
| **ОК-9 Пользоваться профессиональной документацией на государственном и иностранном языках** | | | | | |
| **Описание показателей и критериев оценивания компетенций ОК-9** | Знает   * правила построения простых и сложных предложений на профессиональные темы * основные общеупотребительные глаголы (бытовая и профессиональная лексика) * лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности * особенности произношения терминов профессиональной направленности * правила чтения текстов профессиональной направленности | Не знает. Допускает грубые ошибки | Демонстрирует частичные знания без грубых ошибок | Знает достаточно в базовом объеме | Демонстрирует высокий уровень знаний |
| Умеет:   * понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы * участвовать в диалогах на знакомые общие и профессиональные темы – строить простые высказывания о себе и о своей профессиональной деятельности * кратко обосновывать и объяснять свои действия (текущие и планируемые) * писать простые связные сообщения на знакомые или интересующие профессиональные темы | Не умеет. Демонстрирует частичные умения, допуская грубые ошибки | Демонстрирует частичные умения без грубых ошибок | Умеет применять знания на практике в базовом объеме | Демонстрирует высокий уровень умений |

**2. Оценочные средства для проведения процедур внутренней и внешней оценки качества образовательной деятельности**

Тестовые задания для оценивания компетенции:

**«ОК 1 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам»**

|  |  |  |
| --- | --- | --- |
| **N п/п** | **Формулировка и содержание задания** | **Правильный ответ** |
|  | Вставьте пропущенное ключевое слово в код python, чтобы код корректно проверял, является ли число положительным, отрицательным или нулём.  num = float(input("Введите число: "))  if num > 0:  print("Положительное")  \_\_\_\_ num < 0:  print("Отрицательное")  else:  print("Ноль")  **Варианты ответа:**   1. elif 2. else if 3. when | elif |
|  | Вставьте пропущенное выражение в код python, чтобы цикл выводил числа от 1 до 5 (включительно).  for i in \_\_\_\_:  print(i)  **Варианты ответа:**   1. range(1, 5) 2. range(1, 6) 3. range(5) | range(1, 6) |
|  | Вставьте пропущенное ключевое слово в код python, чтобы определить функцию, возвращающую квадрат числа.  \_\_\_\_ square(n):  return n \*\* 2  **Варианты ответа:**   1. def 2. func 3. function | def |
|  | Вставьте метод в код python, который добавит элемент 'apple' в конец списка fruits.  fruits = ['banana', 'orange']  fruits.\_\_\_\_('apple')  print(fruits) # Вывод: ['banana', 'orange', 'apple']  **Варианты ответа:**   1. add 2. insert 3. append | append |
|  | Вставьте ключ в код python, чтобы получить значение 'Berlin' из словаря capitals.  capitals = {'France': 'Paris', 'Germany': 'Berlin', 'Italy': 'Rome'}  city = capitals[\_\_\_\_]  print(city) # Вывод: Berlin  Варианты ответа:   1. 'Berlin' 2. 'Germany' 3. 'DE' | 'Germany' |
|  | Вставьте пропущенное ключевое слово в код python, чтобы код корректно проверял, является ли число положительным, отрицательным или нулём.  num = float(input("Введите число: "))  if num > 0:  print("Положительное")  \_\_\_\_ num < 0:  print("Отрицательное")  else:  print("Ноль")  **Варианты ответа:**   1. elif 2. when 3. else if 4. ifelse | elif, else if |
|  | Вставьте пропущенное слово в код python, чтобы создать класс MyClass:  \_\_\_\_ MyClass:  def \_\_init\_\_(self, name):  self.name = name | class |
|  | Допишите код python, чтобы преобразовать строку text в верхний регистр:  text = "привет, мир!"  res = text.\_\_\_\_()  print(res) | upper |
|  | Вставьте пропущенное слово в код python, чтобы получить длину строки my\_string:  my\_string = "Пример строки"  length = \_\_\_\_ (my\_string)  print(length) | len |
|  | Допишите код python, чтобы создать список из слов строки sentence, используя метод строк:  sentence = "Это пример предложения"  words = sentence.\_\_\_\_(" ")  print(words) | split |
|  | Какой из перечисленных алгоритмов сортировки обычно демонстрирует лучшую эффективность на больших наборах данных?  **Варианты ответа:**   1. Пузырьковая сортировка; 2. Сортировка вставками; 3. Сортировка слиянием; 4. Сортировка выбором. | Сортировка слиянием |
|  | Какая временная сложность характерна для быстрой сортировки в среднем случае?  **Варианты ответа:**   1. O(n²); 2. O(n log n); 3. O(n); 4. O(log n). | O(n log n); |
|  | Вставьте пропущенные слова в код для создания основного окна приложения:  import tkinter as tk  # Создание основного окна  root = tk.\_\_\_\_\_()  root.title("Моё приложение")  root.\_\_\_\_\_("300x200")  root.mainloop() | Tk, geometry |
|  | Дополните код для добавления кнопки в приложение:  import tkinter as tk  def click\_handler():  print("Кнопка нажата!")  root = tk.Tk()  # Добавление кнопки  button = tk.\_\_\_\_\_(root, text="Нажми меня", command=\_\_\_\_\_)  button.pack()  root.mainloop() | Button, click\_handler |
|  | Сопоставьте регулярные выражения и значения, которые они найдут в строке 7355608  Регулярные выражения:  7\d+5  7\d+?5  Значения:  735  5560  7355608  755  7355  608 | 7\d+5 - 7355  7\d+?5 - 735 |
|  | Сопоставьте код с его выводом в консоль.  Код:  1)  a = 10  b = 5  print("{a} - {b} = {a - b}")  2)  a = 10  b = 5  print(f'{a} - {b} = {a - b}')  3)  a = 5  b = 10  print(r"{b} - {a} = {b - a}")  Вывод:  {b} - {a} = {b - a}  10 - 5 = 5  {a} - {b} = {a - b} | 1 - {a} - {b} = {a - b}  2 - 10 - 5 = 5  3 - {b} - {a} = {b - a} |
|  | Сопоставьте квантификаторы регулярных выражений их аналогам:  Квантификаторы:  +  \*  ?  Аналоги:  {1,}  {0,1}  {0,} | + - {1,}  \* - {0,}  ? - {0,1} |
|  | Вставьте недостающие части кода для создания метки (Label):  import tkinter as tk  root = tk.Tk()  # Создание метки  label = tk.\_\_\_\_\_(root, text="Привет, мир!")  label.\_\_\_\_\_()  root.mainloop() | Label, pack |
|  | Дополните код для установки размера окна и его расположения на экране:  import tkinter as tk  root = tk.Tk()  root.title("Настройка окна")  # Установка размера и расположения окна  root.\_\_\_\_\_("500x400+100+100")  root.mainloop() | geometry |
|  | Вставьте пропущенные элементы в код для создания поля ввода (Entry):  import tkinter as tk  root = tk.Tk()  # Создание поля ввода  entry = tk.\_\_\_\_\_(root)  entry.\_\_\_\_\_()  root.mainloop() | Entry, pack |

Тестовые задания для оценивания компетенции:

**«ОК-9 Пользоваться профессиональной документацией на государственном и иностранном языках»**

|  |  |  |
| --- | --- | --- |
| **N п/п** | **Формулировка и содержание задания** | **Правильный ответ** |
|  | Что такое функции высшего порядка? Выбери верный вариант ответа.  **Варианты ответа:**   1. это функции, которые могут принимать в качестве аргументов другие функции 2. это безымянные функции 3. это функции без побочных эффектов 4. это рекурсивные функции | это функции, которые могут принимать в качестве аргументов другие функции |
|  | Что такое структура данных?   1. Алгоритм сортировки информации 2. Метод шифрования данных 3. Язык программирования 4. Способ организации и хранения информации в памяти компьютера | Способ организации и хранения информации в памяти компьютера |
|  | Какой метод python используется для добавления элемента в конец списка?  **Варианты ответа:**   1. append() 2. pop() 3. insert() 4. count() | append() |
|  | **Что происходит при релокации динамического массива?**  **Варианты ответа:**   1. **Удаляются все элементы** 2. **Создается новый массив с увеличенной емкостью, куда копируются старые элементы** 3. **Меняется тип данных элементов** 4. **Массив превращается в связный список** | Создается новый массив с увеличенной емкостью, куда копируются старые элементы |
|  | Какой принцип лежит в основе структурного подхода?  **Варианты ответа:**   1. Декомпозиция сложных систем на небольшие подпрограммы 2. Объединение всех задач в один модуль 3. Использование исключительно глобальных данных 4. Отказ от подпрограмм | Декомпозиция сложных систем на небольшие подпрограммы |
|  | Как называется шаблон для создания объектов, который определяет их свойства и методы? | класс |
|  | Как называется экземпляр класса, обладающий собственным состоянием и поведением? | объект |
|  | Как называется механизм, позволяющий создавать классы на основе других классов? | наследование |
|  | Как называется блок кода внутри класса, определяющий поведение объекта? | метод |
|  | Какое ключевое слово используется в Python для определения класса? | class |
|  | **Сопоставьте понятия с их описаниями:**   1. Модуль. 2. Пакет. 3. Импорт (import). 4. Инициализационный файл (\_\_init\_\_.py). 5. Пространство имён.   **Описания:**   1. Механизм, который позволяет включить в программу код из внешнего файла или библиотеки. 2. Контейнер, который может содержать несколько модулей и подпакетов, организованный в виде директории с определённым набором файлов. 3. Отдельный файл с кодом на Python, который может быть импортирован и использован в других программах. 4. Файл, который указывает интерпретатору Python, что директория должна рассматриваться как пакет, и может содержать код для инициализации пакета. 5. Система уникальных имён, которая помогает избежать конфликтов между идентификаторами (например, именами функций или классов) в разных модулях. | 1 — В. 2 — Б. 3 — А. 4 — Г. 5 — Д. |
|  | **Сопоставьте понятия с их описаниями:**   1. Класс. 2. Объект. 3. Наследование. 4. Полиморфизм. 5. Инкапсуляция.   **Описания:**   1. Механизм, который позволяет создавать новые классы на основе существующих, перенимая их свойства и методы. 2. Принцип, позволяющий объектам разных классов использовать одни и те же интерфейсы или методы, но с различным поведением. 3. Шаблон для создания объектов, который определяет их свойства и методы. 4. Процесс сокрытия деталей реализации и предоставления внешнего интерфейса для взаимодействия с объектом. 5. Экземпляр класса, обладающий собственным состоянием и поведением. | 1 — В. 2 — Д. 3 — А. 4 — Б. 5 — Г. |
|  | **Сопоставьте ключевое слово с его ролью в контексте циклов:**   1. break 2. continue 3. pass   **Роли:**   1. Пропускает текущую итерацию цикла и переходит к следующей. 2. Завершает цикл досрочно. 3. Ничего не делает, может использоваться как «заполнитель» в синтаксически необходимых блоках кода. | 1 — Б 2 — А 3 — В |
|  | **Сопоставьте операцию с её описанием:**   1. in. 2. min(). 3. sum().   **Описания:**   1. Возвращает сумму всех элементов коллекции (если они числовые). 2. Проверяет, есть ли элемент в коллекции. 3. Находит минимальный элемент в коллекции. | 1 — Б. 2 — В. 3 — А. |
|  | **Сопоставьте тип коллекции с его характеристикой:**   1. Список (list). 2. Кортеж (tuple). 3. Множество (set). 4. Словарь (dict).   **Характеристики:**   1. Неупорядоченная коллекция уникальных элементов, не допускающая дубликатов. 2. Упорядоченная изменяемая коллекция элементов, которые могут быть разных типов. 3. Упорядоченная неизменяемая коллекция элементов, которая может содержать дубликаты. 4. Коллекция пар «ключ-значение», где ключи уникальны. | Ответ:  1 — Б.  2 — В.  3 — А.  4 — Г. |
|  | Что такое декоратор в Python?  **Варианты ответа:**   1. Функция, которая изменяет другую функцию. 2. Встроенный тип данных. 3. Тип обработки ошибок. 4. Способ создания классов. | Функция, которая изменяет другую функцию. |
|  | Могут ли декораторы принимать аргументы?  **Варианты ответа:**   1. Да. 2. Нет. 3. Только некоторые декораторы. 4. Зависит от версии Python. | Да |
|  | Какой символ используется для применения декоратора в Python?  **Варианты ответа:**   1. @ 2. # 3. & 4. $ | @ |
|  | Какова цель декоратора functools.wraps?  **Варианты ответа:**   1. Сохранить метаданные исходной функции. 2. Изменить возвращаемый тип. 3. Регистрировать вызовы функций. 4. Ограничить время выполнения. | Сохранить метаданные исходной функции |
|  | Что из приведённого ниже является примером встроенного декоратора в Python?   1. @staticmethod. 2. @classmethod. 3. @property. 4. Всё вышеперечисленное. | Всё вышеперечисленное. |