Министерство сельского хозяйства Российской Федерации
Иркутский государственный аграрный университет
имени А.А. Ежевского

Колледж автомобильного транспорта и агротехнологий

УТВЕРЖДАЮ:
Директор


_____ Н.Н. Бельков
«31» марта 2023 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ОП.04 Основы алгоритмизации и программирования

Специальность: 09.02.07 Информационные системы и программирование
(программа подготовки специалистов среднего звена)

Форма обучения: очная
2 курс; 3, 4 семестр

Молодежный 2023

1. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Фонд оценочных средств для промежуточной аттестации по учебной дисциплине ОП.04 Основы алгоритмизации и программирования, включает:

- перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы;
- описание шкал оценивания;
- типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения (промежуточной аттестации) по дисциплине, характеризующих этапы формирования компетенций и (или) для итогового контроля сформированности компетенций.

2.ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ С УКАЗАНИЕМ ЭТАПОВ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Рабочая программа профессионального модуля определяет перечень планируемых результатов обучения модулю, соотнесенных с планируемыми результатами освоения образовательной программы.

Код	**Наименование компетенции (планируемые результаты освоения ОП)**	**Планируемые результаты обучения по дисциплине, характеризующие этапы формирования компетенции**
	Общие компетенции	**В области знания и понимания (А)**
Вид деятельности: Осуществление интеграции программных модулей		**Уметь**: распознавать задачу и/или проблему в профессиональном и/или социальном контексте; анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы; **Знать:** актуальный профессиональный и социальный контекст, в котором приходится работать и жить; основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте; алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	

		профессиональной и смежных сферах; структуру плана для решения задач; порядок оценки результатов решения задач профессиональной деятельности
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности	**Уметь:** определять задачи для поиска информации; определять необходимые источники информации; планировать процесс поиска; структурировать получаемую информацию; выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска **Знать:** номенклатура информационных источников, применяемых в профессиональной деятельности; приемы структурирования информации; формат оформления результатов поиска информации
ОК 09	Использовать информационные технологии в профессиональной деятельности	**Уметь:** применять средства информационных технологий для решения профессиональных задач; использовать современное программное обеспечение **Знать:** современные средства и устройства информатизации; порядок их применения и программное обеспечение в профессиональной деятельности
	Профессиональные компетенции	**В области интеллектуальных навыков (В)**

ПК 2.1.	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.	**Уметь:** Анализировать проектную и техническую документацию. Использовать специализированные графические средства построения и анализа архитектуры программных продуктов. Организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов. Определять источники и приемники данных. Проводить сравнительный анализ. Выполнять отладку, используя методы и инструменты условной компиляции (классы Debug и Trace). Оценивать размер минимального набора тестов. Разрабатывать тестовые пакеты и тестовые сценарии. Выявлять ошибки в системных компонентах на основе спецификаций. **Знать:** Модели процесса разработки программного обеспечения. Основные принципы процесса разработки программного обеспечения. Основные подходы к интегрированию программных модулей. Виды и варианты интеграционных решений. Современные технологии и инструменты интеграции. Основные протоколы доступа к данным. Методы и способы идентификации сбоев и ошибок при интеграции приложений. Методы отладочных классов. Стандарты качества программной документации.

		Основы организации инспектирования и верификации. Встроенные и основные специализированные инструменты анализа качества программных продуктов. Графические средства проектирования архитектуры программных продуктов. Методы организации работы в команде разработчиков. **Практический опыт:** Разрабатывать и оформлять требования к программным модулям по предложенной документации. Разрабатывать тестовые наборы (пакеты) для программного модуля. Разрабатывать тестовые сценарии программного средства. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.
ПК 5.1.	Собирать исходные данные для разработки проектной документации на информационную систему.	**Уметь:** Осуществлять постановку задачи по обработке информации. Выполнять анализ предметной области. Использовать алгоритмы обработки информации для различных приложений. Работать с инструментальными средствами обработки информации. Осуществлять выбор модели построения информационной системы. Осуществлять выбор модели и средства построения информационной системы и программных средств. **Знать:** Основные виды и процедуры обработки информации, модели и методы решения задач обработки информации.

		Основные платформы для создания, исполнения и управления информационной системой. Основные модели построения информационных систем, их структуру, особенности и области применения. Платформы для создания, исполнения и управления информационной системой. Основные процессы управления проектом разработки. Методы и средства проектирования, разработки и тестирования информационных систем. **Практический опыт:** Анализировать предметную область. Использовать инструментальные средства обработки информации. Обеспечивать сбор данных для анализа использования и функционирования информационной системы. Определять состав оборудования и программных средств разработки информационной системы. Выполнять работы предпроектной стадии.

3.ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

При проведении промежуточной аттестации в колледже используются традиционные формы аттестации:

Элемент модуля	**Форма промежуточной аттестации**	**Шкала оценивания**
ПМ.04 Основы алгоритмизации и программирования	Экзамен	"отлично", "хорошо", "удовлетворительно", "неудовлетворительно"

4. ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ (ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ И (ИЛИ) ДЛЯ ИТОГОВОГО КОНТРОЛЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИИ

4.1. Примерный перечень вопросов к экзамену

ОП.04 Основы алгоритмизации и программирования

(семестр 4)

1. Какой алгоритм называется линейным? Нарисуйте общий вид линейного алгоритма.

Линейный алгоритм — это алгоритм, образуемый командами, которые выполняются однократно и именно в той последовательности, в которой записаны.

Общий вид линейного алгоритма можно представить в виде блок-схемы, которая представляет собой геометрические фигуры (блоки), соединённые стрелками. Стрелки показывают связь между этапами и последовательность их выполнения. Каждый блок сопровождается надписью.

Пример линейного алгоритма — алгоритм «Открой дверь»:

1. **Начало.**
2. **Достань ключ из кармана.**
3. **Вставь ключ в замочную скважину.**
4. **Поверни ключ два раза.**
5. **Вытащи ключ.**
6. **Конец.**

2. С какой целью используется оператор exit (0) в программах, написанных на языке Python?

Оператор exit(0) в программах на языке Python используется для успешного завершения программы.

Принято считать, что статус 0 означает успешное выполнение, а любой ненулевой статус указывает на ошибку или ненормальное завершение.

3. Поясните назначение метода format и приведите примеры его применения.

Назначение метода format() в Python — форматирование строк. Он позволяет вставлять значения в строки, заменяя заданные метки формата в строке на конкретные значения. Это полезно, когда нужно выводить строки, которые зависят от некоторых данных, например, при выводе результатов вычислений.

Примеры применения метода format(): вывод числа с определённым числом знаков после запятой.

```
pi = 3.141592653589793
formatted_pi = "The value of pi is {:.2f}".format(pi)
print(formatted_pi)  # "The value of pi is 3.14"
```

4. Что называется разветвляющимся алгоритмом?

Разветвляющийся алгоритм — это алгоритм, в котором действие выполняется по одной из возможных ветвей решения задачи, в зависимости от выполнения условий.

Каждое из возможных направлений дальнейших действий называется ветвью.

Разветвляющиеся алгоритмы могут быть полными и неполными.

Полное ветвление — это тип алгоритма, в котором выполняется проверка условия и в зависимости от результата проверки выполняется то или иное действие. Например: «Если (условие выполняется) телефон разрядился, то (истина) заряжу его, иначе (ложь) буду им пользоваться дальше».

Неполное ветвление — это тип алгоритма, в котором проверка условия не выполняется. Если условие не выполняется, то выполняется действие, расположенное после «если». Например: «Если пойдёт дождь, то возьму зонт».

5. Как записывается простой условный оператор в блок-схемах, в программах? Как работает простой условный оператор?

Простой условный оператор в Python записывается с использованием оператора if. Затем следует само условие, после идёт двоеточие и с новой строки с отступом начинается код.

Пример:

```
x = 10
if x > 5:
   print('x больше 5')
```

Скопировать

В этом примере, если переменная x больше пяти, Python выведет в консоль сообщение «x больше 5». В остальных случаях код внутри блока if не будет выполняться и пользователь не увидит сообщение.

Конструкция if-else позволяет выполнять два блока кода: для истинного и ложного условий. Это позволяет управлять поведением программы в зависимости от входных данных и состояний. Синтаксис:

```
if условие:
    Блок_кода_1
else:
   Блок_кода_2
```

Простой условный оператор в Python работает так: программа оценивает тестовое выражение и выполняет оператор, только если тестовое выражение истинно. Если тестовое выражение имеет значение False, оператор не выполняется.

6. Как записывается сокращенный условный оператор в блок-схемах, в программах? Как работает сокращенный условный оператор?

Сокращённый условный оператор в Python записывается с использованием условных выражений (тернарных операторов). Они позволяют сократить код условного оператора до одной строки.

Синтаксис:

```
<выражение1> if <условие> else <выражение2>
```

Пример:

```
age = 18
status = "голосующий" if age >= 18 else "неголосующий"
print(f"Вы являетесь {status}.")
```

Работа сокращённого условного оператора: если условие истинно, выполняется блок кода, находящийся внутри if. Если условие ложно, выполняется блок кода, находящийся внутри else.

Также в Python позволительна сокращённая запись сложного логического выражения, например:

if 0 < a < b: print(b - a)

7. Как записывается составной условный оператор в блок-схемах, в программах? Как работает составной условный оператор?

Составной условный оператор в Python записывается с использованием оператора elif. Он позволяет проверять несколько выражений на истинность и выполнять блок кода, как только одно из условий оценивается как истинное.

Конструкция оператора elif:

if выражение1: оператор(ы)

elif выражение2: оператор(ы)

elif выражение3: оператор(ы)

else: оператор(ы)

Оператор else в этом случае содержит блок кода, который выполняется, если условное выражение в операторе if принимает значение 0 или ложь (FALSE). Оператор else является необязательным, и за ним может быть только один оператор else.

Работа составного условного оператора заключается в том, что он выбирает не более одного блока команд, который будет выполнен. Если ни одно условие не выполняется и не задана секция else, условный оператор не перейдёт ни к какому блоку команд и передаст управление команде, следующей сразу после последнего блока команд, описанного в условном операторе.

8. Как записываются многозначные ветвления в блок схемах, в программах?

В языке Python многозначные ветвления записываются с помощью каскадного ветвления. Для этого между блоками if и else добавляется ещё один блок — elif с некоторым условием и инструкциями.

Пример записи:

if условие 1:

Блок инструкций 1

elif условие 2:

Блок инструкций 2

else:

Блок инструкций 3

Дополнительных условий и связанных с ними блоков elif может быть сколько угодно, но важно отметить, что в такой сложной конструкции будет выполнен всегда только один блок кода. Другими словами, как только некоторое условие оказалось истинным, соответствующий блок кода выполняется, и дальнейшие условия не проверяются.

Также внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы, что позволяет сделать выбор не из двух, а из нескольких вариантов.

9. Как работает условный оператор if при проверке нескольких условий?

Условный оператор if в Python позволяет выполнить блок кода, если заданное условие истинно. Если условие равно False, блок кода будет пропущен.

Для проверки нескольких условий одновременно в операторе if в Python можно использовать логические операторы:

- **and — возвращает True, если оба условия истинны, и False — если ложны;**
- **or — возвращает True, если хотя бы одно из условий истинно, и False в противном случае;**
- **not — возвращает True, если условие ложно, и False — если истинно.**

Также для последовательной проверки нескольких условий можно использовать оператор if-elif-else:

1. **if — проверяет первое условие;**
2. **elif (сокращение от else if) — проверяет следующие условия, если предыдущее условие ложно;**
3. **else — выполняет блок кода, если все предыдущие условия ложны.**

10. Дайте определение циклического алгоритма.

Циклический алгоритм — это алгоритм, в котором некоторая часть операций повторяется многократно.

Он описывает действия, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

11. Расскажите о работе оператора цикла for по возрастающим значениям параметра, нарисовав общий вид алгоритма и синтаксис этого оператора.

Оператор цикла for в Python поэлементно перебирает последовательность и выполняет код, который записан в теле цикла. Главное условие успешной работы цикла — объект должен быть итерируемым.

Общий вид алгоритма: указывают переменную, которая будет хранить значения элементов последовательности, затем указывают саму последовательность, которую нужно перебрать. Далее в теле цикла выполняют действия с каждым элементом последовательности.

Синтаксис цикла for:

for [элемент] in [последовательность]: [тело цикла]

Пример использования: чтобы просуммировать значения чисел от 1 до n, можно воспользоваться следующей программой:

```
sum = 0

for i in range(1, n + 1):

   sum += i
```

В этом примере переменная i принимает значения 1, 2, …, n, и значение переменной sum последовательно увеличивается на указанные значения.

12. Расскажите о работе сложного циклического процесса, нарисовав общий вид алгоритма и синтаксис этого оператора.

Оператор цикла for в Python поэлементно перебирает последовательность и выполняет код, который записан в теле цикла. Главное условие успешной работы цикла — объект должен быть итерируемым.

Общий вид алгоритма: указывают переменную, которая будет хранить значения элементов последовательности, затем указывают саму последовательность, которую нужно перебрать. Далее в теле цикла выполняют действия с каждым элементом последовательности.

Синтаксис цикла for:

for [элемент] in [последовательность]: [тело цикла]

Пример использования: чтобы просуммировать значения чисел от 1 до n, можно воспользоваться следующей программой:

```
sum = 0

for i in range(1, n + 1):

   sum += i
```

В этом примере переменная i принимает значения 1, 2, …, n, и значение переменной sum последовательно увеличивается на указанные значения.

13. Какой цикл называется внешним, а какой – внутренним?

Любой цикл, содержащий внутри себя один или несколько других циклов, называется вложенным. Цикл, охватывающий другие циклы, называется внешним, а остальные циклы - внутренними. Типы циклических алгоритмов внешнего и внутреннего циклов при этом могут быть любыми. Внутренний цикл, от подготовки до завершения, должен находиться внутри внешнего, точнее, быть его телом.

14. Нарисуйте общий вид алгоритма оператора цикла while. Напишите синтаксис оператора цикла while.

Общий вид алгоритма оператора цикла while в Python: сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передаётся на следующую инструкцию после тела цикла while. Если условие истинно, то выполняется инструкция, после чего условие проверяется снова и снова выполняется инструкция. Так продолжается до тех пор, пока условие будет истинно. Как только условие станет ложно, работа цикла завершится и управление передастся следующей инструкции после цикла.

Синтаксис цикла while в Python:

while выражение: оператор(ы)

Здесь оператор(ы) могут быть одиночным оператором или блоком операторов. Условие может быть любым выражением, а true — любым ненулевым значением.

В синтаксисе Python в конце строки с условием ставится двоеточие, а всё тело выделяется отступом (табуляцией или четырьмя пробелами).

15. Расскажите о работе оператора цикла while. Приведите примеры.

Цикл while в Python позволяет выполнять блок кода до тех пор, пока условие остаётся истинным. Это делает его особенно полезным для задач, где количество итераций заранее неизвестно.

Синтаксис цикла while:

while условие: блок_кода

Условие — это логическое выражение, которое проверяется перед каждой итерацией цикла. Если условие истинно (True), блок кода выполняется. Если условие ложно (False), выполнение цикла прекращается.

Пример базового цикла while, который выводит числа от 1 до 5:

```
i = 1
```

```
while i <= 5:
    print(i)
    i += 1
```

Ещё один пример использования цикла while для определения количества цифр натурального числа n:

```
n = int(input())
length = 0
while n > 0:
    n //= 10
    length += 1
```

Ещё один пример обработки пользовательского ввода до тех пор, пока не будет введено корректное значение:

```
user_input = ""
while user_input.lower() != "exit":
    user_input = input("Введите команду (или 'exit' для выхода): ")
    print(f"Вы ввели: {user_input}")
```

Критерии экзамена

Отметка «5 (отлично)» ставится в случае:

знания, понимания, глубины усвоения обучающимся всего объема программного материала;

творчески применять полученные знания в незнакомой ситуации;

отсутствия ошибок и недочётов при воспроизведении изученного материала, при устных ответах, устранения отдельных неточностей с помощью дополнительных вопросов педагога;

соблюдения культуры письменной и устной речи, правил оформления письменных работ.

Отметка «4 (хорошо)» ставится в случае:

знания всего изученного материала;

умения выделять главные положения в изученном материале, на основании фактов и примеров обобщать, делать выводы, устанавливать межпредметные и внутрипредметные связи, применять полученные знания на практике;

наличие незначительных (негрубых) ошибок при воспроизведении изученного материала;

соблюдения основных правил культуры письменной и устной речи, правил оформления письменных работ.

Отметка «3 (удовлетворительно)» ставится в случае:

- знания и усвоения материала на уровне минимальных требований программы, затруднения при самостоятельном воспроизведении, необходимости незначительной помощи учителя;

умения работать на уровне воспроизведения, затруднения при ответах на видоизменённые вопросы;

наличия 1-2 грубых ошибок, нескольких негрубых при воспроизведении изученного материла;

незначительного несоблюдения основных правил культуры письменной и устной речи, правил оформления письменных работ.

Отметка «2 (неудовлетворительно)» ставится в случае:

знания и усвоения учебного материала на уровне ниже минимальных требований программы;

отсутствия умения работать на уровне воспроизведения, затруднения при ответах на стандартные вопросы;

наличия нескольких грубых ошибок, большого числа негрубых при воспроизведении изученного материала;

- значительного несоблюдения основных правил культуры письменной и устной речи, правил оформления письменных работ.

Отметка «1 (неудовлетворительно)» ставится в случае:

- отказ обучающегося от ответа, выполнения работы, теста, отсутствие выполненного (в том числе, домашнего) задания.

При выставлении отметок необходимо учитывать классификацию ошибок и их количество:

грубые ошибки;

однотипные ошибки;

негрубые ошибки;

недочеты.

К грубым ошибкам следует относить:

незнание определения основных понятий, правил,

неумение выделять главное в ответе;

неумение делать выводы и обобщения;

неумение пользоваться первоисточниками, учебником и справочником.

К **однотипным ошибкам** относятся ошибки на одно и то же правило.

К **негрубым ошибкам** следует относить:

неточность формулировок, определений, понятий, правил, вызванная неполнотой охвата основных признаков определяемого понятия или замена 1-2 из этих признаков второстепенными;

нерациональные методы работы с учебной и справочной литературой

Разработчик: преподаватель Шмелёва Е.И.

ФОС одобрен на заседании предметно-цикловой комиссии социально-экономических и естественнонаучных № 8 от «29» марта 2023 г.

Председатель ПЦК Е.А.Хуснудинова

(подпись)

СОГЛАСОВАНО:

Внешний эксперт:

Директор ИЭУПИ Иркутского ГАУ

доцент, к.т.н М.Н. Барсукова

(должность, звание, квалификационная категория) *(подпись)* (Ф.И.О.)