Учреждение профессионального образования
«Колледж Казанского инновационного университета»
Альметьевский филиал

УТВЕРЖДЕНА
в составе Основной
образовательной программы –
программы подготовки специалистов среднего звена
протокол № 6 от «28» августа 2024 г.

Фонд оценочных средств

ОП.04 «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

программы подготовки специалистов среднего звена

по специальности

09.02.07 Информационные системы и программирование
(на базе основного общего образования)

Срок получения СПО по ППССЗ – 3 г.10 мес.

Форма обучения - очная

Присваиваемая квалификация
Программист

Альметьевск 2024

Программа составлена в соответствии с требованиями ФГОС СОО и учебным планом программы подготовки специалистов среднего звена по специальности: 09.02.07 Информационные системы и программирование для обучающихся на базе основного общего образования

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Фонд оценочных средств для осуществления текущего контроля освоения дисциплины «Основы алгоритмизации и программирования» и проведения промежуточной аттестации обеспечивает соответствие персональных достижений, обучающихся поэтапным требованиям образовательной программы.

Успешное изучение дисциплины «Основы алгоритмизации и программирования» предполагает не только освоение лекционного материала и закрепление его на лабораторных и практических занятиях, но и самостоятельную работу с языками программирования; формирование у обучающихся общекультурных и профессиональных компетенций, необходимых и достаточных для осуществления профессиональной деятельности.

Фонд включает следующие виды оценочных средств: практические задачи по программированию и дополнительные вопросы. За решение задачи и ответов на вопросы обучающийся набирает баллы, соответствующие сложности задачи и вопросов. Набрав необходимый минимум баллов, обучающийся получает зачёт. Все задачи для получения экзамена доступны обучающемуся, что стимулирует студентов готовиться к экзамену дома, самостоятельно. Задачи на лабораторных и практических занятиях подобны задачам, приведённым в ФОС. Освоив тему лабораторного или практического занятия, обучающиеся могут закрепить её, решая самостоятельно задачи и одновременно готовясь к экзамену.

2. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Изучение дисциплины «Основы алгоритмизации и программирования» обеспечивает формирование у выпускников по специальности 09.02.07 Информационные системы и программирование следующих общих компетенций (ОК) и профессиональных компетенций (ПК):

Результаты освоения учебной дисциплины направлены на формирование профессиональных и общих компетенций ОК 01.; ОК 02.; ОК 04.; ОК 05.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 2.4.; ПК 2.5.:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности;

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде;

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК.1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

3.

Код ПК, ОК	Умения	Знания
ОК 1 ОК 2 ОК 4 ОК 5 ОК 9 ПК 1.1 ПК 1.2 ПК 1.3 ПК 1.4 ПК 1.5 ПК 2.4, ПК 2.5	Разрабатывать алгоритмы для конкретных задач. Использовать программы для графического отображения алгоритмов. Определять сложность работы алгоритмов. Работать в среде программирования. Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования. Оформлять код программы в соответствии со стандартом кодирования. Выполнять проверку, отладку кода программы. .	Понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции. Эволюцию языков программирования, их классификацию, понятие системы программирования. Основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти. Подпрограммы, составление библиотек подпрограмм Объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

В результате освоения учебной дисциплины обучающийся должен приобрести практический опыт:

- Разрабатывать алгоритм решения поставленной задачи и реализовывать его средствами автоматизированного проектирования.
- Разрабатывать код программного продукта на основе готовой спецификации на уровне модуля.
- Разрабатывать мобильные приложения.
- Использовать инструментальные средства на этапе отладки программного продукта.
- Проводить тестирование программного модуля по определенному сценарию.
- Использовать инструментальные средства на этапе тестирования программного продукта.
- Разрабатывать тестовые наборы (пакеты) для программного модуля.
- Разрабатывать тестовые сценарии программного средства.
- Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.

4. ОЦЕНОЧНЫЕ СРЕДСТВА (ФОРМЫ) ТЕКУЩЕГО КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

4.1. План проведения оценочных мероприятий

По тематике разделов и тем предусмотрены решение практической задачи накомпьютере, в зависимости от сложности по выбору, устный ответ на поставленные вопросы.

№ п/п	Наименование разделов и тем	Формы текущего контроля успеваемости
Раздел 1. Введение в программирование		
1.	Тема 1.1. Языки программирования. ОК 1,ОК 2,ОК 4, ОК 5, ОК 9, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
2.	Тема 1.2. Типы данных ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
Раздел 2. Операторы языка программирования		
3.	Тема 2.1. Операторы языка программирования ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
Раздел 3. Модульное программирование. Организация динамических структур данных		
4.	Тема 3.1. Процедуры и функции. ОК ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
5.	Тема 3.2. Структуризация в программировании ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
6.	Тема 3.3. Модульное программирование ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
Раздел 4. Основные конструкции языков программирования		
7.	Тема 4.1. Указатели. ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
Раздел 5. Объектно-ориентированное программирование		

8.	Тема 5.1. Основные принципы объектно-ориентированного программирования (ООП) ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
9.	Тема 5.2. Интегрированная среда разработчика. ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
10.	Тема 5.3. Визуальное событийно-управляемое программирование. ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
11.	Тема 5.4 Разработка оконного приложения ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
12.	Тема 5.5 Этапы разработки приложений ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы
13.	Тема 5.6 Иерархия классов. ОК 01, 02, 04, 09, ПК 1.1- ПК 1.5, ПК 2.4, 2.5	Устный опрос Лабораторные работы

4. ЗАДАНИЯ ДЛЯ ОЦЕНКИ ОСВОЕНИЯ ДИСЦИПЛИНЫ ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

4.2. Оценочные средства, применяемые на лабораторных и практических занятиях

Критерии оценивания:

При получении 12 баллов и выше выставляется положительную оценку.

1. Задача на 5 баллов.

Вычислить объем пирамиды, основанием которой является треугольник, для значений A, B, C и H данных в контрольном примере. Для вычисления площади основания использовать формулу Герона:

$$S = \sqrt{P(P-A)(P-B)(P-C)},$$

где: $P = \frac{A+B+C}{2}$, $V = S \cdot \frac{H}{2}$,

Исходные данные взять из контрольного примера. Контрольный пример: A=3, B=4, C=5, H=6. Результат V=12.

2. Задача на 7 баллов

Найти значение функции, вычисляемое по формуле: $y=\cos^2 x$ при $0<x<2$, иначе $y=1-\sin(x^2)$.

3. Задача на 10 баллов

С помощью цикла «пока» или цикла «до» написать программу возведения числа A в целую степень N.

4. Задание на 15 баллов:

В массиве из 10 чисел есть хотя бы один нулевой элемент. Вычислить произведение элементов массива до первого нуля.

5. Задача на 5 баллов.

По заданным величинам радиусов оснований *R* и *r* и высоты *h* найти объем площадь поверхности усеченного конуса по формулам:

$V = \frac{\pi h}{3}(R^2 + r^2 + Rr)$, $l = \sqrt{h^2+(R-r)^2}$

Исходные данные взять из контрольного примера. Контрольный пример: *R*=20, *r*=10, *h*=30.Результат: *S*=4548.866, *V*=21980.

6. Задача на 7 баллов

Перераспределить значения переменных X и Y так, чтобы в X оказалось большее из этихзначений, а в Y – меньшее.

7. Задача на 10 баллов

С помощью цикла «пока» или цикла «до» написать программу вычисления факториалазаданного целого числа.

Факториал числа N вычисляется по следующей формуле:

N!=1*2*3*...*N.

8. Задание на 15 баллов:

В массиве из 10 чисел подсчитать сумму элементов, стоящих на четных местах.

Ключ к заданиям

Вариант 1.Часть А

№ задания	Верный ответ	Критерии
1	12	1б – полное правильное соответствие 0 б – остальные случаи

2	import math
	
	def calculate_y(x):
	if 0 < x < 2:
	y = math.cos(2 * x)
	else:
	y = 1 - math.sin(x ** 2)
	return y
	
	# Пример использования функции
	x = float(input("Введите значение x: "))
	y = calculate_y(x)
	print(f"Значение y при x = {x}: {y}")
		1б – полное правильное соответствие
		0 б – остальные случаи

3	def power(a, n):
	result = 1
	while n > 0:
	result *= a
	n -= 1
	return result
	
	# Пример использования функции
		1б – полное правильное соответствие
		0 б – остальные случаи

	a = int(input("Введите число A: ")) n = int(input("Введите степень N: ")) result = power(a, n) print(f"Число {a} в степени {n} равно {result}")	

4	def product_until_zero(numbers):
	product = 1
	for number in numbers:
	if number == 0:
	break
	product *= number
	return product
	
	# Пример использования функции
	numbers = [2, 3, 4, 0, 5, 6, 7, 8, 9, 10]
	result = product_until_zero(numbers)
	print(f"Произведение элементов до первого нуля: {result}")
		1б – полное правильное соответствие
		0 б – остальные случаи

5	import math
	
	def calculate_volume_and_surface_area(R, r, h):
	# Вычисление образующей
	l = math.sqrt((R - r)**2 + h**2)
	
	# Вычисление объёма
	V = (1/3) * math.pi * h * (R**2 + r**2 + R*r)
	
	# Вычисление площади поверхности
	S = math.pi * (R + r) * l + math.pi * R**2 + math.pi * r**2
	
	return V, S
	
	# Пример использования функции
	R = 20
	r = 10
	h = 30
	V, S = calculate_volume_and_surface_area(R, r, h)
	print(f"Объём усечённого конуса:
		1б – полное правильное соответствие
		0 б – остальные случаи

	{V}") print(f"Площадь поверхности усечённого конуса: {S}")	

6	def swap_values(x, y):
	if x < y:
	x, y = y, x
	return x, y
	
	# Пример использования функции
	x = int(input("Введите значение X: "))
	y = int(input("Введите значение Y: "))
	x, y = swap_values(x, y)
	print(f"X = {x}, Y = {y}")
		1б – полное правильное соответствие
		0 б – остальные случаи

7	def factorial(n):
	result = 1
	while n > 1:
	result *= n
	n -= 1
	return result
	
	# Пример использования функции
	n = int(input("Введите число: "))
	result = factorial(n)
	print(f"Факториал числа {n} равен {result}")
		1б – полное правильное соответствие
		0 б – остальные случаи

8	def sum_of_even_positions(numbers):
	sum_even = 0
	for i in range(len(numbers)):
	if (i + 1) % 2 == 0:
	sum_even += numbers[i]
	return sum_even
	
	# Пример использования функции
	numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
	result = sum_of_even_positions(numbers)
	print(f"Сумма элементов на четных местах: {result}")
		1б – полное правильное соответствие
		0 б – остальные случаи

3.4. Проверочные и контрольные работы по дисциплине

Проверочная работа 1. Простые типы данных, операции, стандартные функции, условный оператор.

Вариант 1

1.С клавиатуры вводятся 2 прописные буквы латинского алфавита, программа должна вывести букву, расположенную раньше по алфавиту.

2. Дано трехзначное число. Вывести число. полученное при перестановке цифр десятков и единиц исходного числа (например, 123 перейдет в 132).

3.Проверить истинность высказывания: "Цифры данного трехзначного числа расположены в порядке убывания"

4. Дано значение температуры TC в градусах Цельсия. Определить значение этой же температуры в градусах Фаренгейта TE. Температура по Цельсию TC и температура по Фаренгейту TE связаны следующим соотношением:

TC = (TE - 32)*5/9.

5. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% — если сумма больше 1000 руб., например, если стоимость покупки равна 1100 руб. то со скидкой это будет 1045 руб.

Вариант 2

1.Даны 2 числа и символ, обозначающий арифметическую операцию(+.-.*). В зависимости от значения символа, вывести результат операции на экран.

2.Дано натуральное трехзначное число. Напечатать true, если сумма цифр этого числа четная, и false иначе.

3.Даны числа x, y. Проверить истинность высказывания: "Точка с координатами (x, y) лежит во второй или четвертой координатной четверти

4.Написать программу вычисления стоимости поездки на автомобиле на дачу (туда и обратно). Исходными данными являются: расстояние до дачи (в километрах); количество бензина, которое потребляет автомобиль на 100 км пробега: цена одного литра бензина.

5. Написать программу, которая вычисляет оптимальный вес пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправиться или похудеть. Оптимальный вес вычисляется по

формуле: рост (в сантиметрах) —100.

Проверочная работа 2. Операторы цикла. Подпрограммы- функции. Подпрограммы- процедуры.

Вариант 1

1. Описать процедуру RectPS(x1, y1, x2, y2, P, S). вычисляющую периметр P и площадь S
прямоугольника со сторонами, параллельными осям координат, по координатам (x1, y1), (x2, y2) его
противоположных вершин (x1, У1, x2, y2 — входные, P и S — выходные параметры вещественного
типа). С помощью этой процедуры найти периметры и площади трех прямоугольников с данными
противоположными вершинами.

2. Описать функцию Calc(A, B, Op) вещественного типа, выполняющую над ненулевыми
вещественными числами A и B одну из арифметических операций и возвращающую ее результат.
Вид операции определяется целым параметром Op: | — вычитание, 2 — умножение, 3 — деление,
остальные значения — сложение. С помощью Calc выполнить для данных A и B операции,
определяемые данными целыми N1, N2, N3

3. Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деления, найти
количество и сумму его цифр.

Вариант 2

1. Описать процедуру ShiftRight3(A, B, C), выполняющую правый циклический сдвиг: значение A
переходит в B, значение B — в C. значение C — в A (A, B, C — вещественные параметры,
являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить
правый циклический сдвиг для двух данных наборов из трех чисел: (A1, B1, C1) и(A2, B2, C2).

2. Описать функцию RootCount (A, B, C) целого типа. определяющую количество корней квадратного
уравнения A*x^2 + B*x + C =0 (A, B. C — вещественные параметры, A <> 0).

С ее помощью найти
количество корней для каждого из трех квадратных уравнений с данными коэффициентами.

3. Спортсмен-лыжник начал тренировки. пробежав в первый день 10 км. Каждый следующий день
он увеличивал длину пробега на P процентов от пробега предыдущего дня (P — вещественное,
0<P< 50). По данному P определить, после какого дня суммарный пробег лыжника за все дни
превысит 200 км, и вывести найденное количество дней K (целое) и суммарный пробег S
(вещественное число).

4.Описать процедуру ShiftLeft3(A, B, C), выполняющую левый циклический сдвиг: значение A переходит в C, значение B — в A, значение C — в B (A, B, C — целые параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить левый циклический сдвиг для трёх данных наборов из трёх чисел: (A1, B1, C1), (A2, B2, C2) и (A3, B3, C3).

5. Описать процедуру Swap(A, B), меняющую местами значения двух переменных A и B. С помощью этой процедуры поменять местами значения двух наборов переменных: (A1, B1) и (A2, B2).

Ключи к задачам:

Вариант 1.

1. def earlier_letter(a, b):
    if a < b:
        return a
    else:
        return b

# Пример использования функции
a = input("Введите первую букву: ")
b = input("Введите вторую букву: ")
result = earlier_letter(a, b)
    print(f"Буква, расположенная раньше по алфавиту: {result}")

2. def swap_digits(number):
    units = number % 10

```
    tens = (number // 10) % 10
    hundreds = number // 100
    return hundreds * 100 + units * 10 + tens

# Пример использования функции
number = int(input("Введите трёхзначное число: "))
result = swap_digits(number)
print(f"Число после перестановки цифр: {result}")
```

3.
```
def check_descending_order(number):
    units = number % 10
    tens = (number // 10) % 10
    hundreds = number // 100
    return hundreds > tens > units
# Пример использования функции
number = int(input("Введите трёхзначное число: "))
result = check_descending_order(number)
print(f"Цифры числа расположены в порядке убывания: {result}")
```

4.
```
def celsius_to_fahrenheit(celsius):
    fahrenheit = (celsius * 9 / 5) + 32
  return fahrenheit
# Пример использования функции
celsius = float(input("Введите температуру в градусах Цельсия: "))
result = celsius_to_fahrenheit(celsius)
print(f"Температура в градусах Фаренгейта: {result}")
```

5.
```
def calculate_discount(price):
    if price > 1000:
     discount = 0.05
  elif price > 500:
     discount = 0.03
  else:
```

```
    discount = 0
    return price - (price * discount)

# Пример использования функции
price = float(input("Введите стоимость покупки: "))
result = calculate_discount(price)
print(f"Стоимость покупки с учётом скидки: {result}")
```

Ключи к задачам:

Вариант 2.

1.

```
def ShiftRight3(A, B, C):
    temp = C
    C = B
    B = A
    A = temp
    return A, B, C

# Пример использования процедуры
A1, B1, C1 = 1.0, 2.0, 3.0
A2, B2, C2 = 4.0, 5.0, 6.0

print(f"Исходные значения: (A1, B1, C1) = {A1, B1, C1}, (A2, B2, C2) = {A2, B2, C2}")

A1, B1, C1 = ShiftRight3(A1, B1, C1)
A2, B2, C2 = ShiftRight3(A2, B2, C2)

print(f"Значения после сдвига: (A1, B1, C1) = {A1, B1, C1}, (A2, B2, C2) = {A2, B2, C2}")
```

2.

```
import math

def RootCount(A, B, C):
    discriminant = B**2 - 4*A*C
    if discriminant > 0:
        return 2
    elif discriminant == 0:
        return 1
```

```
    else:
        return 0

# Пример использования функции
equations = [(1, 2, 1), (1, 0, -1), (1, 1, 1)]
for A, B, C in equations:
    count = RootCount(A, B, C)
    print(f"Уравнение {A}x^2 + {B}x + {C} = 0 имеет {count} корней")
```

3.

```
def ski_training(P):
    day = 1
    total_distance = 10
    current_distance = 10

    while total_distance <= 200:
        day += 1
        current_distance *= 1 + P / 100
        total_distance += current_distance

    return day, total_distance

# Пример использования функции
P = float(input("Введите значение P (0 < P < 50): "))
K, S = ski_training(P)
print(f"Количество дней K: {K}")
print(f"Суммарный пробег S: {S:.2f} км")
```

5.

```
def ShiftLeft3(A, B, C):
    temp = A
    A = B
    B = C
    C = temp
    return A, B, C

# Пример использования процедуры
A1, B1, C1 = 1, 2, 3
A2, B2, C2 = 4, 5, 6
A3, B3, C3 = 7, 8, 9

print(f"Исходные значения: (A1, B1, C1) = {A1, B1, C1}, (A2, B2, C2) = {A2, B2, C2}, (A3, B3, C3) = {A3, B3, C3}")
```

```
A1, B1, C1 = ShiftLeft3(A1, B1, C1)
A2, B2, C2 = ShiftLeft3(A2, B2, C2)
A3, B3, C3 = ShiftLeft3(A3, B3, C3)

print(f"Значения после сдвига: (A1, B1, C1) = {A1, B1, C1}, (A2, B2, C2) = {A2, B2, C2}, (A3, B3, C3) = {A3, B3, C3}")
```

6.

```
def Swap(A, B):
    temp = A
    A = B
    B = temp
    return A, B

# Пример использования процедуры
A1, B1 = 1, 2
A2, B2 = 3, 4

print(f"Исходные значения: (A1, B1) = {A1, B1}, (A2, B2) = {A2, B2}")

A1, B1 = Swap(A1, B1)
A2, B2 = Swap(A2, B2)

print(f"Значения после обмена: (A1, B1) = {A1, B1}, (A2, B2) = {A2, B2}")
```

Контрольная работа 1.

Структурированные типы данных. Одномерные массивы. Двумерные массивы. Строки. Файлы. Записи.

Вариант 1

1. Дана строка, содержащая полное имя файла. то есть имя диска, список каталогов (путь), собственно имя
и расширение. Выделить и напечатать имя файла с расширением.

Например, для строки 'C: \1\2\ris1.bmp' должно быть напечатано ris1.bmp.

2. Дано число n. Записать в текстовый файл квадраты всех нечетных чисел от 1 до n.

3. Дан целочисленный массив размера M. не содержащий одинаковых чисел. Проверить, образуют ли
его элементы арифметическую прогрессию

4.Поле шахматной доски описывается как запись type sell = record vert(: (a, b, c, d, e, f, g, , h); Bo; 1..8 еп@;
Описать программу. проверяющую. имеют ли поля п1 и п2 одинаковый цвет.

5. Дана квадратная матрица n-го порядка. Проверить, является ли она симметричной.

Вариант 2

1. Дана строка S из нулей и единиц. Число единиц в строке не меньше 1. Составить из данного набора
нулей и единиц максимальное нечетное число.

2. Дан текстовый файл, в котором построчно записаны числа. Найти максимальное.

3. Дан массив ненулевых целых чисел размера N. Проверить. образуют ли его элементы геометрическую
прогрессию.

4.Момент времени описывается как запись type time = record h: 0..23; m,s: 0..59 end;

Описать и использовать в программе функцию (процедуру), решающую следующую задачу: проверить,

предшествует ли в рамках суток время {1 времени 12:

5.Дана матрица с m строками и n столбцами. Вывести транспонированную матрицу.

Контрольная работа 2. Создание и использование собственных классов

Вариант 1.

Создать класс TRatio для работы с рациональными числами. Описать методы класса, позволяющие

1. привести рациональное число к несократимому виду;
2. проверить на равенство два рациональных числа:
3. сложить два рациональных числа:

Написать программу, использующую данный класс.

Вариант 2.

Создать класс TTreug для работы с треугольниками. Описать методы класса, позволяющие

1. Проверить, что треугольник не вырожденный
2. Найти периметр треугольника
3. Проверить на равенство два треугольника

Написать программу, использующую данный класс.

Контрольная работа 3. Оконные приложения для Windows. Работа со стандартными компонентами.

Вариант 1

1.На форме есть меню, 2 многострочных редактора. кнопка. При выборе соответствующего пункта меню в
один многострочный редактор загружается текст из файла. По нажатию кнопки скопировать в другой
многострочный редактор все слова, начинающиеся с заданной буквы алфавита. Буква задается с помощью
полосы прокрутки.

2.Написать программу. вычисляющую значение тригонометрических функций (Sin, Cos, Tg, Ctg).

Предусмотреть возможность выбора угловой меры (радианы. градусы). Выбор функции и меры
осуществляется с помощью переключателей.

Вариант 2

1.На форме есть меню, 2 многострочных редактора. кнопка. При выборе соответствующего пункта меню в
один многострочный редактор загружается текст из файла. По нажатию кнопки скопировать в другой
многострочный редактор все слова, длина которых равна заданной. Длина задается с помощью полосы
прокрутки.

2. Управление видимостью и доступностью компонентов. При включении опции “Включить демонстрацию
примера” группа переключателей становится видимой. иначе пользователь ее не видит. При выборе первого
переключателя из группы текстовое поле Memo становится видимым и доступным для редактирования,
второго-видимым и недоступным для редактирования. третьего-невидимым.

Ключи к задачам:

Контрольная работа 1. Вариант 1.

1.

```
def get_file_name(full_path):
    parts = full_path.split('\\')
    return parts[-1]


# Пример использования функции
full_path = 'C:\\1\\2\\ris1.bmp'
file_name = get_file_name(full_path)
print(f"Имя файла с расширением: {file_name}")
```

2.

```
def write_squares_to_file(n):
 with open('squares.txt', 'w') as file:
for i in range(1, n + 1, 2):
file.write(f"{i**2}\n")
# Пример использования функции
n = int(input("Введите число n: "))
write_squares_to_file(n)
```

3.

```
def is_arithmetic_progression(arr):
    if len(arr) < 2:
  return False
 diff = arr[1] - arr[0]
for i in range(2, len(arr)):
 if arr[i] - arr[i-1] != diff:
return False
  return True

# Пример использования функции
arr = [1, 3, 5, 7, 9]
result = is_arithmetic_progression(arr)
print(f"Элементы массива образуют арифметическую прогрессию: {result}")
```

4.

```
def same_color(p1, p2):
 color1 = (ord(p1[0]) - ord('a')) % 2 + p1[1] % 2
 color2 = (ord(p2[0]) - ord('a')) % 2 + p2[1] % 2
return color1 == color2
# Пример использования функции
p1 = ('a', 1)
p2 = ('b', 2)
```

```
result = same_color(p1, p2)
print(f"Поля {p1} и {p2} имеют одинаковый цвет: {result}")
```

5.

```
 def is_symmetric(matrix):
   n = len(matrix)
for i in range(n):
for j in range(n):
if matrix[i][j] != matrix[j][i]:
  return False
 return True
# Пример использования функции
matrix = [[1, 2, 3], [2, 4, 5], [3, 5, 6]]
result = is_symmetric(matrix)
print(f"Матрица является симметричной: {result}")
```

Контрольная работа 1. Вариант 2.

1.

```
def max_odd_number(S):
 ones = S.count('1')
 if ones == 0:
 return None
max_odd = '1' + '0' * (len(S) - ones) + '1' * (ones - 1)
return max_odd
# Пример использования функции
S = "001001"
result = max_odd_number(S)
print(f"Максимальное нечётное число: {result}")
```

2.

```
 def find_max_in_file(file_name):
```

```
  max_num = float('-inf')
 with open(file_name, 'r') as file:
  for line in file:
num = int(line.strip())
if num > max_num:
   max_num = num
   return max_num
# Пример использования функции
file_name = "numbers.txt"
result = find_max_in_file(file_name)
print(f"Максимальное число в файле: {result}")
```

3.

```
 def is_geometric_progression(arr):
    if len(arr) < 2:
        return False
    ratio = arr[1] / arr[0]
    for i in range(2, len(arr)):
        if arr[i] / arr[i-1] != ratio:
            return False
    return True

# Пример использования функции
arr = [2, 4, 8, 16]
result = is_geometric_progression(arr)
print(f"Элементы массива образуют геометрическую прогрессию: {result}")
```

4.

```
def time_to_minutes(t):
    return t['h'] * 60 + t['m']
```

```
def compare_times(t1, t2):
    return time_to_minutes(t1) < time_to_minutes(t2)

# Пример использования функции
t1 = {'h': 12, 'm': 0}
t2 = {'h': 13, 'm': 30}
result = compare_times(t1, t2)
print(f"Время {t1} предшествует времени {t2}: {result}")
```

5.

```
def transpose(matrix):
    m = len(matrix)
    n = len(matrix[0])
    transposed = [[0 for _ in range(m)] for _ in range(n)]
    for i in range(m):
        for j in range(n):
            transposed[j][i] = matrix[i][j]
    return transposed

# Пример использования функции
matrix = [[1, 2, 3], [4, 5, 6]]
transposed_matrix = transpose(matrix)
print("Транспонированная матрица:")
for row in transposed_matrix:
    print(row)
```

Контрольная работа 2. Вариант 1.

```
import math

class TRatio:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
```

```
        self.denominator = denominator
        self.reduce()  # Приводим дробь к несократимому виду при инициализации

    def reduce(self):
        """Приводит рациональное число к несократимому виду."""
        gcd = math.gcd(self.numerator, self.denominator)
        self.numerator //= gcd
        self.denominator //= gcd

    def __eq__(self, other):
        """Проверяет на равенство два рациональных числа."""
        return self.numerator == other.numerator and self.denominator == other.denominator

    def __add__(self, other):
        """Складывает два рациональных числа."""
        new_numerator = self.numerator * other.denominator + other.numerator *
self.denominator
        new_denominator = self.denominator * other.denominator
        return TRatio(new_numerator, new_denominator)

    def __str__(self):
        return f"{self.numerator}/{self.denominator}"

# Пример использования класса
if __name__ == "__main__":
    ratio1 = TRatio(4, 6)  # Создаем рациональное число 4/6
    print(f"Рациональное число 1: {ratio1}")  # Выводим рациональное число
    ratio1.reduce()  # Приводим рациональное число к несократимому виду
    print(f"Рациональное число 1 (несократимый вид): {ratio1}")

    ratio2 = TRatio(2, 3)  # Создаем рациональное число 2/3
    print(f"Рациональное число 2: {ratio2}")

    if ratio1 == ratio2:  # Проверяем на равенство два рациональных числа
        print("Рациональные числа равны")
    else:
        print("Рациональные числа не равны")

    sum_ratio = ratio1 + ratio2  # Складываем два рациональных числа
    print(f"Сумма рациональных чисел: {sum_ratio}")
```

Контрольная работа 2. Вариант 2.

```
class TTreug:
```

```
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def is_valid(self):
        """Проверяет, что треугольник не вырожденный."""
        return self.a + self.b > self.c and self.a + self.c > self.b and self.b + self.c > self.a

    def perimeter(self):
        """Находит периметр треугольника."""
        return self.a + self.b + self.c

    def __eq__(self, other):
        """Проверяет на равенство два треугольника."""
        return (self.a == other.a and self.b == other.b and self.c == other.c) or \
               (self.a == other.a and self.b == other.c and self.c == other.b) or \
               (self.a == other.b and self.b == other.a and self.c == other.c) or \
               (self.a == other.b and self.b == other.c and self.c == other.a) or \
               (self.a == other.c and self.b == other.a and self.c == other.b) or \
               (self.a == other.c and self.b == other.b and self.c == other.a)

# Пример использования класса
if __name__ == "__main__":
    treug1 = TTreug(3, 4, 5)  # Создаем треугольник со сторонами 3, 4, 5
    print(f"Треугольник 1: {treug1.a}, {treug1.b}, {treug1.c}")  # Выводим стороны треугольника

    if treug1.is_valid():  # Проверяем, что треугольник не вырожденный
        print("Треугольник не вырожденный")
    else:
        print("Треугольник вырожденный")

    print(f"Периметр треугольника 1: {treug1.perimeter()}")  # Находим периметр треугольника

    treug2 = TTreug(4, 3, 5)  # Создаем треугольник со сторонами 4, 3, 5
    print(f"Треугольник 2: {treug2.a}, {treug2.b}, {treug2.c}")

    if treug1 == treug2:  # Проверяем на равенство два треугольника
        print("Треугольники равны")
    else:
        print("Треугольники не равны")
```

Контрольная работа 3. Вариант 1.

Задача 1. Решение

```
import tkinter as tk
from tkinter import filedialog

def load_text_from_file():
    file_path = filedialog.askopenfilename()
    with open(file_path, 'r') as file:
        text.delete('1.0', tk.END)
        text.insert(tk.END, file.read())

def copy_words_starting_with_letter():
    letter = chr(scrollbar.get() + ord('A'))
    words = text.get('1.0', tk.END).split()
    filtered_words = [word for word in words if word.startswith(letter)]
    text2.delete('1.0', tk.END)
    text2.insert(tk.END, ' '.join(filtered_words))

root = tk.Tk()

menu = tk.Menu(root)
root.config(menu=menu)
file_menu = tk.Menu(menu, tearoff=0)
menu.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Open", command=load_text_from_file)

text = tk.Text(root)
text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

scrollbar = tk.Scale(root, from_=0, to=25, orient=tk.HORIZONTAL)
scrollbar.pack(side=tk.BOTTOM, fill=tk.X)

text2 = tk.Text(root)
text2.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

button = tk.Button(root, text="Copy Words", command=copy_words_starting_with_letter)
button.pack()

root.mainloop()
```

Задача 2. Решение

```
import tkinter as tk
import math

def calculate():
    angle = float(entry.get())
    if radians_var.get():
        angle = math.radians(angle)
    sin_result.config(text=f"Sin: {math.sin(angle):.2f}")
    cos_result.config(text=f"Cos: {math.cos(angle):.2f}")
    tan_result.config(text=f"Tan: {math.tan(angle):.2f}")
    cot_result.config(text=f"Cot: {1 / math.tan(angle):.2f}" if math.tan(angle) != 0 else "Cot: undefined")

root = tk.Tk()

entry = tk.Entry(root)
entry.pack()

radians_var = tk.IntVar()
radians_check = tk.Checkbutton(root, text="Radians", variable=radians_var)
radians_check.pack()

button = tk.Button(root, text="Calculate", command=calculate)
button.pack()

sin_result = tk.Label(root)
sin_result.pack()

cos_result = tk.Label(root)
cos_result.pack()

tan_result = tk.Label(root)
tan_result.pack()

cot_result = tk.Label(root)
cot_result.pack()

root.mainloop()
```

Контрольная работа 3. Вариант 2.

Задача 1. Решение

```
import tkinter as tk
from tkinter import filedialog

def load_text_from_file():
    file_path = filedialog.askopenfilename()
    with open(file_path, 'r') as file:
        text.delete('1.0', tk.END)
        text.insert(tk.END, file.read())

def copy_words_of_length():
    length = int(scrollbar.get())
    words = text.get('1.0', tk.END).split()
    filtered_words = [word for word in words if len(word) == length]
    text2.delete('1.0', tk.END)
    text2.insert(tk.END, ' '.join(filtered_words))

root = tk.Tk()

menu = tk.Menu(root)
root.config(menu=menu)
file_menu = tk.Menu(menu, tearoff=0)
menu.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Open", command=load_text_from_file)

text = tk.Text(root)
text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

scrollbar = tk.Scale(root, from_=1, to=20, orient=tk.HORIZONTAL)
scrollbar.pack(side=tk.BOTTOM, fill=tk.X)

text2 = tk.Text(root)
text2.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

button = tk.Button(root, text="Copy Words", command=copy_words_of_length)
button.pack()

root.mainloop()
```

Задача 2. Решение

```
import tkinter as tk

def toggle_demo():
    if demo_var.get():
        radio_frame.grid()
    else:
        radio_frame.grid_remove()

def update_memo_visibility():
    if radio_var.get() == 1:
        memo.config(state=tk.NORMAL, bg='white')
    elif radio_var.get() == 2:
        memo.config(state=tk.DISABLED, bg='lightgray')
    elif radio_var.get() == 3:
        memo.grid_remove()

root = tk.Tk()

demo_var = tk.IntVar()
demo_check = tk.Checkbutton(root, text="Включить демонстрацию примера", variable=demo_var, command=toggle_demo)
demo_check.pack()

radio_frame = tk.Frame(root)
radio_var = tk.IntVar()

radio1 = tk.Radiobutton(radio_frame, text="Видимый и доступный", variable=radio_var, value=1, command=update_memo_visibility)
radio2 = tk.Radiobutton(radio_frame, text="Видимый и недоступный", variable=radio_var, value=2, command=update_memo_visibility)
radio3 = tk.Radiobutton(radio_frame, text="Невидимый", variable=radio_var, value=3, command=update_memo_visibility)

radio1.grid(row=0, column=0)
radio2.grid(row=1, column=0)
radio3.grid(row=2, column=0)

memo = tk.Text(root, height=5, width=30)
memo.grid(row=1, column=1)

root.mainloop()
```

САМОСТОЯТЕЛЬНАЯ РАБОТА

Самостоятельная работа студентов состоит в изучении рекомендуемой литературы,проработке лекционного материала, выполнения предложенных заданий.

ТЕМАТИКА ЗАНЯТИЙ

Алгоритмические структуры

6. Разработка линейных алгоритмов.

7. Разработка алгоритмов с ветвлением.

8. Разработка циклических алгоритмов (циклы с пред- и постусловием, цикл спараметром).

9. Трассировка алгоритма.

10. Разработка алгоритмов с подпрограммами.Основные операторы языка

1. Алгебраические и логические выражения, правила их записи.

2. Присваивание. Совместимость по присваиванию.

3. Ввод и вывод данных в консольном режиме.

4. Условный оператор.

5. Оператор выбора.

6. Операторы цикла (циклы с пред- и постусловием, цикл с параметром).Структурированные типы языка программирования высокого уровня

1. Характеристики структурированных типов данных.

2. Массивы. Линейные и двумерные массивы.

3. Длинная арифметика.

4. Строки.

5. Множества.

6. Записи.

7. Типизированные файлы.

8. Организация файлов записей.

9. Нетипизированные файлы.

10. Текстовые файлы.

11. Прямой доступ к компонентам файлов.

12. Сортировка

файлов. Процедуры и

функции. Модули

1. Процедуры. Разработка и вызов.

2. Функции. Разработка и вызов.

3. Разработка программ на основе структурного подхода.

4. Внешние подпрограммы.

5. Рекурсивные подпрограммы.

6. Модули. Структура и разработка.

7. Стандартные модули.

Организация динамических структур данных (абстрактных типов данных): стек, очередь,двоичное дерево поиска

1. Динамически распределяемая память и ее использование при работе со стандартнымитипами данных.

2. Однонаправленные списки.

3. Двунаправленные списки.

4. Стеки.

5. Очереди.

6. Деки.

7. Двоичные деревья поиска.

Введение в объектно-ориентированное программирование

1. Основные понятия ООП.

2. Разработка программ на основе ООП.

3. Наследование и полиморфизм в ООП.

Реализация абстракций данных методами объектно-ориентированного программирования

1. Абстрактные типы и структуры данных.

2. Классы, объекты, поля, методы.

3. Конструкторы и деструкторы.

4. Свойства и методы объектов.

5. Раннее связывание и позднее связывание.

6. Математические объекты: рациональные и комплексные числа, вектора, матрицы.

Самостоятельная работа обучающихся должна быть направлена на формирование и углубление практических навыков работы с алгоритмическими структурами, структурами данных, процедурами и функциями, программирования на языках высокого уровня.

Для контроля самостоятельной работы обучающихся организуются дискуссии по рассматриваемым проблемам.

4. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПОДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

Вопросы для подготовки к экзамену по дисциплине

1. Алгоритм. Типы алгоритмов. Блок-схемы. Примеры (алгоритм решения квадратного уравнения, алгоритм Евклида нахождения НОД двух натуральных чисел и т.д.).

2. Структура программы. Объявление констант, переменных, типов. Примеры. Операторы ввода-вывода, Примеры (обмен значениями 2-х переменных).

3. Целые тины (арифметические операции, функции, выражения, приоритет операций, операции div, mod. сдвига, форматированный вывод). Представление целых чисел в двоичной системе счисления. Прямой, дополнительный код.

4 Вещественные типы арифметические операции, функции, выражения, приоритет операций) Представление чисел с плавающей точкой. Форматированный вывод. Совместимость типов.

5 Порядковые типы, операции нал порядковыми типами, Логический тип (логические операции, операции pred, succ, ord, сравнение). Таблица истинности. Примеры.

6 Порядковые типы, операции нал порядковыми типами. Вычислительная структура символов. Объекты и операции. Таблица кодировок. Примеры.

7. Разветвляющийся алгоритм. Блок-схема. Условный оператор. Составной оператор. Примеры максимума, значения функции: нахождение корней квадратного уравнения)

8 Порядковые типы, операции над порядковыми типами. Перечислимый и диапазонный типы. Многоальтернативный выбор. Оператор Сазе. Кодирование над двоичным алфавитом.

9. Подпрограммы, Процедуры и функции в ТР. Объявление, вызов. Формальные и фактические параметры, параметры-значения и параметры-переменные. Примеры.

11 Циклические алгоритмы, Блок-схема. Счетный оператор цикла. Составной

оператор. Примеры (подсчет произведения, суммы). Вложенный шик. Досрочный выход из цикла.

12 Циклические алгоритмы. Блок-схема, Оператор цикла с предусловием. Примеры (количество цифр в натуральном числе, сумма элементов последовательности, оканчивающейся 0). Признак конца строки. Оператор цикла с постусловием. Досрочный выход из цикла.

13. Массив как набор переменных с индексами. Задание собственного типа для массива в разделе type. Способы задания массивов типизированные константы, ввод с клавиатуры). Поиск максимального элемента массиве, Одномерные и многомерные массивы, Вывод двумерного массива в виде таблицы.

14. Сороки (тип данных ниц в TP). Операции над строками, сравнение строк. Примеры (поиск подстроки в строке и т.д.)

15.Файлы в TP. Работа © текстовыми файлами в TP (операции чтения, записи и т.д.) Примеры определения количества строк в текстовом файле, поиск строки максимальной длины в файле, запись таблицы умножения в текстовый файл и т.д.)

16. Понятие записи. Тип данных record в TP. Оператор присоединения Примеры. Работа с типизированными файлами в TP (операции; чтения, записи и т.д. запись вещественных чисел в типизированный файл с проверкой существования)

17. Модуль (unit) Структура. Пример (модуль содержащий тригонометрические функции).

18. Статические переменные. Динамическая память. Адресация памяти. Объявление указателей. Типизированные и не типизированные указатели в TP. Операции. Указатель на массив.

19. Динамические структуры данных. Понятие списка. Организация списка с добавлением элементов в начало, просмотр списка. Включение, исключение элементов. Стек— понятие и основные операции.

20. Модуль Graph в TP (инициализация графического режима. графические примитивы). Построение графика функции.

21. Массивы. Алгоритмы сортировки обменами(пузырьком). вставкой, выбором,

22 Массивы. Поиск максимального элемента в массиве. Алгоритм линейного поиска в массиве. Алгоритм двоичного поиска в упорядоченном массиве

5. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРУ ОЦЕНИВАНИЯ

Критерии оценки устных ответов студентов

Развёрнутый ответ обучающегося должен представлять логически последовательное сообщение на заданный вопрос, показывать его умение применять определения, термины в конкретных случаях.

При оценке ответа студента надо руководствоваться следующими критериями, учитывать:

1) полноту и правильность ответа;
2) степень осознанности, понимания изученного;
3) логичность оформление ответа;

Оценка «5» ставится, если студент:

1) полно излагает изученный материал, даёт правильное определенное терминов ипонятий;
2) обнаруживает понимание материала, может обосновать свои суждения, применитьзнания на практике, самостоятельная работа студента;
3) излагает материал последовательно и правильно;

Оценка «4» ставится, если студент даёт ответ, удовлетворяющий тем же требованиям, что и для отметки «5», но допускает 1-2 ошибки, которые сам же исправляет, и 1-2 недочёта в последовательности и оформлении излагаемого.

Оценка «3» ставится, если студент обнаруживает знание и понимание основных положений данной темы, но:

1) излагает материал неполно и допускает неточности в определении понятий;
2) не использует самостоятельно наработанный материал;
3) излагает материал непоследовательно и допускает существенные ошибки.

Оценка «2» ставится, если студент обнаруживает незнание большей части соответствующего раздела изучаемого материала, допускает ошибки в формулировке определений и понятий, искажающие их смысл, беспорядочно и неуверенно излагает материал.

Критерии оценки письменных ответов студентов

1. Оценка «5» выставляется за безошибочную работу.
2. Оценка «4» выставляется при наличии в работе незначительных ошибок профессионального характера.
3. Оценка «3» выставляется при наличии в работе ряда значимых ошибок.
4. Оценка «2» выставляется за работу, которая не соответствует предъявленным требованиям. Оценка индивидуальных образовательных достижений по результатам текущего контроля и промежуточной аттестации производится в соответствии с универсальной шкалой (таблица).

Процент результативности (правильных ответов)	Качественная оценка индивидуальных образовательных достижений	
	балл (отметка)	вербальный аналог
90 ÷ 100	5	отлично
89 ÷ 70	4	хорошо
69 ÷ 59	3	удовлетворительно
менее 59	2	неудовлетворительно