Министерство науки и высшего образования Российской Федерации
Лысьвенский филиал
федерального государственного бюджетного образовательного учреждения
высшего образования
«Пермский национальный исследовательский политехнический университет»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
для проведения промежуточной аттестации обучающихся по дисциплине «Основы алгоритмизации и программирования»
Приложение к рабочей программе дисциплины

Направление подготовки:	09.03.01 Информатика и вычислительная техника
Направленность (профиль) образовательной программы:	Компьютерные системы
Квалификация выпускника:	«Бакалавр»
Выпускающая кафедра:	Общенаучных дисциплин
Форма обучения:	Очная

Курс: 1 **Семестр**: 1,2

Трудоёмкость:
Кредитов по рабочему учебному плану: 9 ЗЕ
Часов по рабочему учебному плану: 324 ч.

Форма промежуточной аттестации:
Экзамен: 1,2 семестр

Лысьва 2019

Фонд оценочных средств для проведения промежуточной аттестации обучающихся для проведения промежуточной аттестации обучающихся по дисциплине является частью (приложением) к рабочей программе дисциплины. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине разработан в соответствии с общей частью фонда оценочных средств для проведения промежуточной аттестации основной образовательной программы, которая устанавливает систему оценивания результатов промежуточной аттестации и критерии выставления оценок**.** Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине устанавливает формы и процедуры текущего контроля успеваемости и промежуточной аттестации обучающихся по дисциплине.

1. Перечень контролируемых результатов обучения по дисциплине, объекты оценивания и виды контроля

Согласно РПД освоение учебного материала дисциплины запланировано в течение двух семестров (1-го и 2 семестра учебного плана). В каждом разделе предусмотрены аудиторные лекционные и лабораторные занятия, а также самостоятельная работа студентов. В рамках освоения учебного материала дисциплины формируется компоненты компетенций *знать, уметь, владеть*, указанные в РПД, которые выступают в качестве контролируемых результатов обучения по дисциплине (табл. 1.1).

Контроль уровня усвоенных знаний, освоенных умений и приобретенных владений осуществляется в рамках текущего, рубежного и промежуточного контроля при изучении теоретического материала, сдаче отчетов по лабораторным работам и экзамена. Виды контроля сведены в таблицу 1.1.

Таблица 1.1. Перечень контролируемых результатов обучения по дисциплине

Контролируемые результаты обучения по дисциплине (ЗУВы)	**Вид контроля**					
	Текущий		**Рубежный**		**Промежуточный**	
	С	**ТО**	**ОЛР**	**Т/КР**		**Экзамен**
Усвоенные знания						
З.1 основы математики, физики, вычислительной техники и программирования, необходимые для решения прикладных задач при проектировании и реализации алгоритмов		ТО	ОЛР	Т		ТВ
З.2 алгоритмические языки программирования семейства С, современные интегрированные среды разработки программного обеспечения, операционные системы семейства Windows и Linux		ТО	ОЛР	Т		ТВ
Освоенные умения						
У.1 решать задачи по проектированию и реализации алгоритмов с применением естественнонаучных и общеинженерных знаний, методов математического анализа и моделирования.			ОЛР	Т		ПЗ
У.2 составлять алгоритмы без привязки к отдельным языкам программирования, писать и отлаживать код на языке С++, тестировать работоспособность программы при помощи средств разработки, интегрировать программные модули.			ОЛР	Т		ПЗ
Приобретенные владения						
В.1 навыками теоретического и экспериментального			ОЛР			КЗ

исследования алгоритмов						
В.2 навыками отладки и тестирования работоспособности программы; языком программирования С++			ОЛР			КЗ

ТО – коллоквиум (теоретический опрос); ОЛР – отчет по лабораторной работе; Т/КР – рубежное тестирование (контрольная работа); ТВ – теоретический вопрос; ПЗ – практическое задание; КЗ – комплексное задание экзамена.

Итоговой оценкой достижения результатов обучения по дисциплине является промежуточная аттестация в виде экзамена, проводимая с учетом результатов текущего и рубежного контроля.

1. Виды контроля, типовые контрольные задания и шкалы оценивания результатов обучения

Текущий контроль успеваемости имеет целью обеспечение максимальной эффективности учебного процесса, управление процессом формирования заданных компетенций обучаемых, повышение мотивации к учебе и предусматривает оценивание хода освоения дисциплины. В соответствии с Положением о проведении текущего контроля успеваемости и промежуточной аттестации обучающихся по образовательным программам высшего образования – программам бакалавриата, специалитета и магистратуры в ПНИПУ предусмотрены следующие виды и периодичность текущего контроля успеваемости обучающихся:

- входной контроль, проверка исходного уровня подготовленности обучаемого и его соответствия предъявляемым требованиям для изучения данной дисциплины;

- текущий контроль усвоения материала (уровня освоения компонента «знать» заданных компетенций) на каждом групповом занятии и контроль посещаемости лекционных занятий;

- промежуточный и рубежный контроль освоения обучаемыми отдельных компонентов «знать», «уметь» заданных компетенций путем компьютерного или бланочного тестирования, контрольных опросов, контрольных работ (индивидуальных домашних заданий), защиты отчетов по лабораторным работам, рефератов, эссе и т.д.

Рубежный контроль по дисциплине проводится на следующей неделе после прохождения модуля дисциплины, а промежуточный – во время каждого контрольного мероприятия внутри модулей дисциплины;

- межсессионная аттестация, единовременное подведение итогов текущей успеваемости не менее одного раза в семестр по всем дисциплинам для каждого направления подготовки (специальности), курса, группы;

- контроль остаточных знаний.

2.1. Текущий контроль усвоения материала

Текущий контроль усвоения материала в форме выборочного теоретического опроса студентов проводится по каждой теме. Результаты по 4-балльной шкале оценивания заносятся в книжку преподавателя и учитываются в виде интегральной оценки при проведении промежуточной аттестации.

2.2. Рубежный контроль

Рубежный контроль для комплексного оценивания усвоенных знаний,

освоенных умений и приобретенных владений (табл. 1.1) проводится в форме защиты лабораторных работ и тестирования (после изучения каждого раздела учебной дисциплины).

2.2.1. Защита лабораторных работ

Всего запланировано 13 лабораторных работ. Типовые темы лабораторных работ приведены в РПД.

Защита лабораторной работы проводится индивидуально каждым студентом или группой студентов. Типовые шкала и критерии оценки приведены в общей части ФОС образовательной программы.

2.2.2. Тестирование

Согласно РПД запланировано тестирование.

Типовые тестовые задания (Приложение1)

Типовые шкала и критерии оценки результатов рубежной контрольной работы приведены в общей части ФОС образовательной программы.

2.3. Промежуточная аттестация (итоговый контроль)

Допуск к промежуточной аттестации осуществляется по результатам текущего и рубежного контроля. Условиями допуска являются успешная сдача всех лабораторных работ и положительная интегральная оценка по результатам текущего и рубежного контроля.

Промежуточная аттестация, согласно РПД, проводится в виде экзамена по дисциплине устно по билетам. Билет содержит теоретические вопросы (ТВ) для проверки усвоенных знаний, практические задания (ПЗ) для проверки освоенных умений и комплексные задания (КЗ) для контроля уровня приобретенных владений всех заявленных компетенций.

Билет формируется таким образом, чтобы в него попали вопросы и практические задания, контролирующие уровень сформированности ***всех*** заявленных компетенций. Форма билета представлена в общей части ФОС образовательной программы.

2.3.1. Типовые вопросы и задания для экзамена по дисциплине

Типовые вопросы для контроля усвоенных знаний:

1. Классификация языков программирования.
2. Понятие алгоритма. Свойства и виды алгоритмов. Способы описания алгоритмов. Основные конструкции алгоритмического языка: линейный алгоритм, ветвление, цикл
3. Этапы решения задач с помощью ЭВМ: постановка задачи, создание модели, алгоритм, кодирование, анализ результатов
4. Виды, классификации и назначение языков программирования. Стандарты языков программирования. Среда программирования. Компиляторы и интерпретаторы
5. Жизненный цикл программы. Программный продукт и его характеристики
6. Переменные и константы. Объявление объектов данных. Внутреннее представление данных в памяти компьютера
7. Типы данных. Простые типы данных. Производные и структурированные типы данных

8. История появления термина «алгоритм». Эволюция языков программирования

9. Понятие системы программирования. Основные функции системы программирования

10. Типы данных, определяемые программистом. Перечисляемый и интервальный типы

11. Форматированный вывод данных

12. Приоритеты выполнения действий в выражениях

13. Составление и отладка программ с использованием арифметических выражений.

14. Общие сведения о подпрограммах. Процедуры и функции. Определение и вызов подпрограмм. Описание, выполнение процедур и функций. Виды параметров в подпрограммах. Области видимости переменных. Механизм передачи параметров. Составление библиотек подпрограмм

15. Основы структурного программирования. Методы структурного программирования

16. Модульное программирование. Понятие модуля. Структура модуля. Компиляция и компановка программы. Библиотеки подпрограмм. Схемы вызова библиотек. Статическое и динамическое связывание

17. Рекурсивные подпрограммы. Привести примеры

18. Достоинства и недостатки структурного программирования

19. Модуль: синтаксис, заголовок, разделы. Использование библиотек подпрограмм

20. Объявление массива. Инициализация. Действия над массивами. Заполнение массива данными. Ввод и вывод одномерных и двумерных массивов. Стандартные функции для массива целых и вещественных чисел. Обработка массива. Удаление и вставка элементов в массив

21. Символьный и строковый типы. Объявление типов. Поиск, удаление, замена и добавление символов в строке. Операции со строками

22. Стандартные функции и процедуры для работы со строками. Массив символов, строки и их обработка

23. Множественный тип данных. Элемент множества. Способы задания множества. Операции над множествами: объединение, разность, пересечение. Логические операции над множествами: проверка принадлежности элемента множеству, проверка включения элемента во множество, сравнение множеств

24. Определение типа Запись. Правила работы с записями. Запись с вариантной частью

25. Типы файлов. Организация доступа к файлам. Файлы последовательного доступа: открытие и закрытие, запись в файл и чтение из файла. Файлы произвольного доступа. Порядок работы с файлами произвольного доступа: открытие и закрытие, запись и считывание из файла произвольного доступа. Создание структуры записи. Использование файла произвольного доступа

26. Стандартные процедуры и функции для работы с файлами разного типа. Использование стандартных процедур и функций для работы с файлами. Работа с текстовыми файлами

27. Работа с массивами. Способы ввода элементов в массив – ввод с клавиатуры, через формулы, оператор присваивания или случайным образом Randomize.

28. Примеры использования строковых типов данных. Отладка и тестирование программ с использованием строковых типов данных.

29. Выполнение анализа процедур при работе со множествами. Представление множеств линейными массивами.

30. Общая схема работы с файлами. Типизированные и нетипизированные файлы.

31. Событийно-управляемая модель программирования. Компонентно-ориентированный подход. Классы объектов. Компоненты и их свойства

32. Формы и компоненты. Принципы визуального программирования. Свойства компонентов. Создание простого приложения

33. Переменные. Типы данных в С++.

34. Структура программы. Команда присваивания в С++.

35. Ввод-вывод данных. Формат выводимых данных.

36. Ввод-вывод данных. Стандартные потоки ввода и вывода. Примеры.

37. Алгоритм линейной структуры в Си++..

38. Структура IF, классификация в С++. Примеры.

39. Структура switch(выбор) и ее программирование в С++. Примеры.

40. Алгоритмы циклической итерационной структуры. Оператор цикла While в С++. Примеры использования.

41. Алгоритмы циклической итерационной структуры. Оператор цикла do… while в С++. Примеры использования.

42. Алгоритмы циклической итерационной структуры. Оператор цикла For в С++. Примеры использования.

43. Операторы break и continue в С++. Примеры использования.

44. Одномерные массивы в С++. Задание массивам первоначальных значений.

45. Операции над массивами и их совместимость. Ввод-вывод массивов в С++.

46. Случайные числа в языке программирования С++.

47. Понятие подпрограммы в С++. Описание подпрограммы.

48. Формальные и фактические параметры в С++.

49. Понятие о локальных и глобальных переменных в С++.

50. Основные математические функции в С++. Примеры.

51. Определение алгоритма. Свойства алгоритма. Формы записи алгоритмов. Примеры.

52. Запись алгоритмов блок-схемами. Основные элементы блок-схем

53. Алгоритмы с ветвлением. Пример алгоритма.

54. Алгоритм цикла с предусловием. Пример алгоритма.

55. Алгоритм цикла с постусловием. Пример алгоритма.

56. Алгоритм цикла с управляющей переменной. Пример алгоритма.

Типовые вопросы и практические задания для контроля освоенных умений:

1. Дан файл, содержащий текст, включающий русские и английские слова. Подсчитать, каких букв в тексте больше — русских или латинских.

2. Дан файл, содержащий текст. Сколько слов в тексте? Сколько цифр в тексте?

3. Дан файл, содержащий текст, включающий русские и английские слова. Получить новый файл, заменив в исходном все заглавные буквы строчными и наоборот.

4. Дан файл, содержащий зашифрованный русский текст. Каждая буква заменяется на следующую за ней (буква **я** заменяется на **а**). Получить в новом файле расшифровку данного текста.

5. Даны два текстовых файла f_1 и f_2. Файл f_1 содержит произвольный текст. Слова в тексте разделены пробелами и знаками препинания. Файл f_2 содержит не более 30 слов, которые разделены запятыми. Эти слова образуют пары: каждое второе является синонимом первого. Заменить в файле f_1 те слова, которые можно, их синонимами. Результат поместить в новый файл.

6. В массив $A[N]$ занесены натуральные числа. Найти сумму тех элементов, которые кратны данному K.

7. В целочисленной последовательности есть нулевые элементы. Создать массив из номеров этих элементов.

8. Дана последовательность целых чисел $a_1, a_2, \ldots, a_n$. Выяснить, какое число встречается раньше — положительное или отрицательное.

9. Дана последовательность действительных чисел $a_1, a_2, \ldots, a_n$. Выяснить, будет ли она возрастающей.

10. Дана последовательность натуральных чисел $a_1, a_2, \ldots, a_n$. Создать массив из четных чисел этой последовательности. Если таких чисел нет, то вывести сообщение об этом факте.

11. В одномерном массиве все отрицательные элементы переместить в начало массива, а остальные — в конец с сохранением порядка следования. Дополнительный массив заводить не разрешается.

12. В одномерном массиве с четным количеством элементов ($2N$) находятся координаты N точек плоскости. Они располагаются в следующем порядке: $x_1, y_1, x_2, y_2, x_3, y_3$, и т.д.

а) Определить минимальный радиус окружности с центром в начале координат, которая содержит все точки.

б) Определить кольцо с центром в начале координат, которое содержит все точки.

в) Определить номера точек, которые могут являться вершинами квадрата.

г) Определить номера точек, которые могут являться вершинами равнобедренного треугольника.

д) Найти номера самых удаленных друг от друга точек и наименее удаленных друг от друга точек.

е) Определить три точки, которые являются вершинами треугольника, для которого разность точек вне его и внутри является минимальной.

13. Заполнить файл последовательного доступа *f* целыми числами, полученными с помощью генератора случайных чисел. Получить в файле *g* те компоненты файла *f*, которые являются четными.

14. Записать в файл последовательного доступа *N* действительных чисел. Вычислить произведение компонент файла и вывести на печать.

15. Заполнить файл последовательного доступа *f* целыми числами, полученными с помощью генератора случайных чисел. Получить в файле *g* все компоненты файла *f*, которые делятся на *m* и не делятся на *n*.

16. Записать в файл последовательного доступа *N* целых чисел, полученных с помощью генератора случайных чисел. Подсчитать количество пар противоположных чисел среди компонент этого файла.

Типовые комплексные задания для контроля приобретенных владений:

1. Решить задачу.

Выполнить операции (здесь *A*, *B*, *C*, *D* — квадратные матрицы порядка *n*). Действия с матрицами оформить в виде класса

1. $5A-4BCD$;
2. $A+4B-5C^T$;
3. $A^T+4B-5C^T$;
4. $5A-4B^TCD^T$;
5. $3D-4A^TBCD$;
6. $BCD-A^T$;
7. $A(B+C)$;
8. $B^TC^TD^T$;
9. $A(B-C)$;
10. $(B+C)AD$;
11. $(B-C)AD$;
12. $AB+AC$;
13. $AB-AC$;
14. A^TB+AC;
15. A^TB-AC;
16. $AB+A^TC$;
17. $AB-A^TC$;
18. $(AB)^TC$;
19. $A^T+B^T+C^T$;
20. $A^T-B^T+C^T-D^T$;
21. $A^TB^T+C^T-D^T$;
22. $A^T-B^TC^T-D^T$;
23. $A^T-B^TCD^T$;
24. $(AB-C)D$;
25. $D(A-BC)$.

2. Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами 1, 2, 3, …, n^2, записывая их в нее «по спирали».

Например, для $n = 5$ получаем следующую матрицу:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

3. Дана действительная квадратная матрица порядка $2n$. Получить новую матрицу, переставляя ее блоки размера $n \times n$ по часовой стрелке, начиная с блока в левом верхнем углу.

4. Решить задачу

а) Для заданного линейного массива вычислить $\sqrt{|a_1 * a_2 * \ldots * a_n|}$.

б) Для заданного линейного массива вычислить $a_1+2a_2+3a_3+...+na_n$

в) Для заданного линейного массива вычислить $a_1*a_2+a_2*a_3+a_3*a_4+...+ a_{n-1}*a_n$

г) Для заданного линейного массива вычислить

$|a_1 – a_2| + |a_2 – a_3| + |a_3 – a_4| + ... + |a_{n-1} – a_n|$

д) Для заданного линейного массива вычислить $a_1–2a_2+2a_3–2a_4 +...+2(–1)^n a_n$

е) Для заданного линейного массива вычислить

$a_1*2a_2–3a_3*4a_4+5a_5*6a_6+…$

ж) Для заданного линейного массива вычислить

$a_1–2a_2+4a_3–8a_4...+2^{n-1}(–1)^n a_n$

з) Составить программу, позволяющую в одномерном массиве вычислить сумму модулей отрицательных элементов массива.

и) Составить программу, позволяющую в одномерном массиве вычислить количество элементов массива, не принадлежащих интервалу (a, b).

к) Составить программу, позволяющую в одномерном массиве вычислить наименьший из элементов массива, принадлежащих отрезку [a, b].

л) Составить программу, позволяющую в одномерном массиве вычислить количество элементов массива, равных первому элементу.

м) В данном линейном массиве четные положительные элементы заменить на 1, а нечетные отрицательные — на -1. Остальные оставить без изменения.

5. Решить задачу

Вариант 1

1) Создать базовый класс с тремя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с тремя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Масса». Наследует базовый класс. Поля определяют центнеры, килограммы, граммы. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с тремя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода «ЦЦ ц КК кг ГГГ г»; б) сложение (сложить две массы – получить третью массу); в) вычитание (вычесть kкг, где k – любое неотрицательное целое число, в т.ч. больше 100).

3) Класс «Точка в трехмерном пространстве». Наследует базовый класс. Поля определяют координаты точки. Два конструктора: без параметров и с тремя параметрами – обращается к соответствующим конструкторам базового класса. Методы: а) сдвиг на вектор (к координатам точки прибавляются координаты вектора); б) расстояние между двумя точками; в) точка, симметричная данной относительно начала координат. Вывод используется из базового класса.

Вариант 2

1) Создать базовый класс с двумя символьными полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с двумя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Строка». Наследует базовый класс. Поле — двухсимвольная строка. Два конструктора: без параметров – использует конструктор базового класса (символы сцепляются); с двумя параметрами – использует конструктор базового класса (символы сцепляются). Методы: а) вывод (переопределить метод базового класса); б) сравнение; в) вычитание (уменьшить код каждого символа на k. Если получается отрицательный код – ответ – пустая строка).

3) Класс «Массив, элементами которого являются пары символов». Наследует базовый класс. Два конструктора: заполнение случайно и копированием (задействовать конструкторы базового класса). Методы: а) сортировка по первому символу, но пара переставляется целиком; б) поиск пары с максимальной и минимальной суммой символов (суммируются коды); в) изменение регистра (если символ – буква), - на противоположный); г) переопределить метод вывода (задействовать при этом метод вывода из базового класса).

Вариант 3

1) Создать базовый класс с двумя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с двумя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Обыкновенная дробь». Наследует базовый класс. Поля определяют числитель и знаменатель. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с двумя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода «Ч / З»; б) сложение; в) вычитание.

3) Класс «Прямоугольник». Наследует базовый класс. Поля определяют ширину и длину прямоугольника. Два конструктора: без параметров и с двумя параметрами – обращается к соответствующим конструкторам базового класса. Методы: а) периметр; б) площадь; в) выбор из двух прямоугольников прямоугольника с наибольшей площадью. Вывод используется из базового класса.

Вариант 4

1) Создать базовый класс с тремя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с тремя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Окружность на плоскости». Наследует базовый класс. Поля определяют координаты центра окружности и радиус. Два конструктора: без параметров, с тремя параметрами – обращаются к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода «(x; y), r»; б) площадь; в) длина окружности; г) окружность минимального целочисленного радиуса такая, что центр ее располагается в центре первой заданной окружности, и она включает вторую заданную окружность.

3) Класс «Точка в трехмерном пространстве». Наследует базовый класс. Поля определяют координаты точки. Два конструктора: без параметров и с тремя параметрами – обращается к соответствующим конструкторам базового класса. Методы: а) точка с целочисленными координатами, которая располагается примерно посередине между данной и началом координат; б) точки, симметричные данной относительно каждой из координатных осей; в) точки, являющиеся проекцией данной на каждую из координатных осей. Вывод используется из базового класса.

Вариант 5

1) Создать базовый класс с двумя вещественными полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с двумя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Треугольник на плоскости». Наследует базовый класс. Добавляются ещё 4 вещественных поля. Пары чисел обозначают координаты точки на плоскости. Конструкторы тоже должны быть ориентированы на работу с ещё 4 дополнительными полями. Переопределить метод вывода. Методы: а) проверка (логический метод), являются ли заданные точки вершинами треугольника (т.е. не лежат ли на одной прямой); б) периметр треугольника; в) площадь треугольника.

3) Класс «Массив треугольников». Наследует класс «Треугольник на плоскости». Метод: а) треугольник наибольшей площади; б) треугольник наименьшего периметра.

Вариант 6

1) Создать базовый класс с четырьмя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с четырьмя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Отрезок на плоскости». Наследует базовый класс. Первые два числа обозначают координаты одного, следующие два — другого конца отрезка. Переопределить вывод. Методы: а) длина отрезка; б) отрезок, симметричный данному относительно оси Ox; в)отрезок, симметричный данному относительно начала координат.

3) Класс «Прямоугольник на плоскости со сторонами, параллельными координатным осям». Наследует базовый класс. Первые два числа обозначают координаты левого верхнего, следующие два — координаты правого нижнего углов прямоугольника. Конструктор без параметров переопределить, чтобы координаты генерировались корректно. Переопределить вывод. Методы: а) периметр; б) площадь; в) выбор из двух прямоугольников прямоугольника с наибольшей площадью.

Вариант 7

1) Создать базовый класс с тремя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с тремя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Обыкновенная дробь с целой частью». Наследует базовый класс. Поля определяют целую часть, числитель и знаменатель. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с тремя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода «Ц Ч / З»; б) сокращение дроби; в) сложение; г) вычитание.

3) Класс «Треугольник на плоскости». Наследует базовый класс. Поля обозначают длины сторон треугольника. Методы: а) проверка (логический метод), являются ли заданные числа длинами сторон треугольника (выполняется ли неравенство треугольника); б) периметр треугольника; в) площадь треугольника.

Вариант 8

1) Создать базовый класс с тремя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с тремя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Время». Наследует базовый класс. Поля определяют часы, минуты, секунды. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с тремя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода ЧЧ:ММ:СС; б) сложение (сложить два времени – получить третье время); в) вычитание (вычесть kсекунд, где k – любое неотрицательное целое число, в т.ч. больше 60).

3) Класс «Вектор в трехмерном пространстве». Наследует базовый класс. Поля определяют координаты вектора. Два конструктора: без параметров и с тремя параметрами – обращается к соответствующим конструкторам базового класса.

Методы: а) сложение векторов; б) вычитание векторов; в) умножение вектора на целое число. Вывод используется из базового класса.

Вариант 9

1) Создать базовый класс с двумя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с двумя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Масса». Наследует базовый класс. Поля определяют тонны, килограммы. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с двумя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода «ТТТТ т ККК кг»; б) сложение (сложить две массы – получить третью массу); в) вычитание (вычесть ккг, где k – любое неотрицательное целое число, в т.ч. больше 1000).

3) Класс «Точка в двумерном пространстве». Наследует базовый класс. Поля определяют координаты точки. Два конструктора: без параметров и с двумя параметрами – обращается к соответствующим конструкторам базового класса. Методы: а) сдвиг на вектор (к координатам точки прибавляются координаты вектора); б) расстояние между двумя точками; в) точка, симметричная данной относительно начала координат. Вывод используется из базового класса.

Вариант 10

1) Создать базовый класс с тремя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с тремя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс «Квадрат на плоскости со сторонами, параллельными осям координат».Наследует базовый класс. Поля определяют: первые два числа — координаты левого верхнего угла, третье — длина стороны. Переопределить метод вывода. Методы: а) периметр; б) площадь; в) квадрат, симметричный данному относительно начала координат.

3) Класс «Время». Наследует базовый класс. Поля определяют часы, минуты, секунды. Два конструктора: без параметров – переопределяется конструктор базового класса (на данные накладываются ограничения, но они по-прежнему формируются случайно); с тремя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) вывод (переопределить метод базового класса) – формат вывода ЧЧ:ММ:СС; б) сравнение (результат — «раньше», «позже», «равны»); в) умножение— умножить нанатуральное число k(например, для 12:02:22 при умножении на 2 получим 00:04:44).Указание. Действие выполняем над временем в секундах и затем переводим обратно в часы, минуты, секунды, убрав целые сутки, если они прошли.

Вариант 11

1) Создать базовый класс с одним целым полем. Разработать два конструктора: первый – без параметров – заполняет поле случайным положительным числом; второй – с параметром – конструктор копирования. Разработать метод: вывод поля.

2) Класс «Натуральное число». Наследует базовый класс. Поле определяет натуральное число. Методы: а) количество цифр; б) сумма цифр; в) является ли палиндромом (true, false); г) «перевертыш» числа.

3) Класс «Массив натуральных чисел». Наследует класс «Натуральное число». Переопределить метод вывода. Отсортировать по количеству цифр и сумме цифр (т.е. в каждой группе с одинаковым количеством цифр сортируем по сумме цифр).

Вариант 12

1) Создать базовый класс с двумя целыми полями. Разработать два конструктора: первый – без параметров – заполняет поля случайными данными; второй – с двумя параметрами – конструктор копирования. Разработать метод: вывода полей.

2) Класс«Прямоугольный треугольник». Наследует базовый класс. Поля определяют длины катетов. Два конструктора: без параметров – переопределяется конструктор базового класса; с двумя параметрами – обращается к соответствующему конструктору базового класса. Методы: а) гипотенуза; б) периметр; в) площадь. Примечание. Катеты должны быть таковы, чтобы гипотенуза тоже была целым числом!

3) Класс «Длина». Наследует базовый класс. Поля обозначают длину в метрах и сантиметрах. Методы: а) вывод (переопределить метод базового класса) – формат вывода «ММММмССсм»; б) сложение (сложить две длины – получить третью длину); в) вычитание (вычесть kсм, где k – любое неотрицательное целое число, в т.ч. больше 99).

Перечень типовых ситуационных заданий и кейсов для проверки умений и владений представлен в приложении 1. *Полный перечень теоретических вопросов и практических заданий в форме утвержденного комплекта экзаменационных билетов хранится на выпускающей кафедре.*

2.3.2. Шкалы оценивания результатов обучения на экзамене

Оценка результатов обучения по дисциплине в форме уровня сформированности компонентов *знать, уметь, владеть* заявленных компетенций проводится по 4-х балльной шкале оценивания путем выборочного контроля во время экзамена.

Типовые шкала и критерии оценки результатов обучения при сдаче экзамена для компонентов *знать, уметь и владеть* приведены в общей части ФОС образовательной программы.

3. Критерии оценивания уровня сформированности компонентов и компетенций

3.1. Оценка уровня сформированности компонентов компетенций

При оценке уровня сформированности компетенций в рамках выборочного контроля при экзамене считается, что *полученная оценка за компонент*

проверяемой в билете компетенции обобщается на соответствующий компонент всех компетенций, формируемых в рамках данной учебной дисциплины.

Типовые критерии и шкалы оценивания уровня сформированности компонентов компетенций приведены в общей части ФОС образовательной программы.

3.2. Оценка уровня сформированности компетенций

Общая оценка уровня сформированности всех компетенций проводится путем агрегирования оценок, полученных студентом за каждый компонент формируемых компетенций, с учетом результатов текущего и рубежного контроля в виде интегральной оценки по 4-х балльной шкале. Все результаты контроля заносятся в оценочный лист и заполняются преподавателем по итогам промежуточной аттестации.

Форма оценочного листа и требования к его заполнению приведены в общей части ФОС образовательной программы.

При формировании итоговой оценки промежуточной аттестации в виде экзамена используются типовые критерии, приведенные в общей части ФОС образовательной программы.

Тестовые задания для самопроверки

В А Р И А Н Т I

1. Синтаксические ошибки в программе на Turbo Pascal выявляются в процессе

1) отладки; 2) тестирования; 3) компиляции; 4) набора текста программы; 5) сохранения текста программы на диске.

2. Наибольшее целое число без знака, кодируемое восьмью битами:

1) 127; 2) 512; 3) 256; 4) 255; 5) 999.

3. Кодируется восьмибитовое целое со знаком (тип ShortInt). 10000101_2 является кодом числа

1) –132; 2) 123; 3) 132; 4) –123; 5) –2.

4. Четыре из приведенных ниже арифметических выражений соответствуют одной и той же формуле, а одно — другой. Найти это «лишнее» выражение.

1) A * B / C * D / E; 2) B / E / C * (A * D); 3) (A * B) / (C * D) / E; 4) (A * D * B) / (C * E);
5) A * B * D/ C / E.

5. При $y = -2$ выражение Q and (Cos(X) < 0) or ((X – 2 * Y) < (X – Y * Y)) принимает значение TRUE при значениях переменных

1) $x = 0$,Q = FALSE; 2) $x = 5$, Q = FALSE; 3) $x = -1$, Q = TRUE; 4) $x = -2$, Q = TRUE;
5) $x = 5$, Q = TRUE.

6. Тип величины A, значение которой выводится процедурой WriteLn(A : 10 : 2):

1) Integer; 2) String; 3) Real; 4) Boolean; 5) любой из указанных в пп. 1-4.

7. Из приведенных описаний констант

```
Const Max = 100; {а}
      Min = 10; {б}
      Srednee = (Max – Min) / 2; {в}
      Nazv = ’Массив чисел’; {г}
      Predmet : string [11] = ’Информатика’; {д}
```

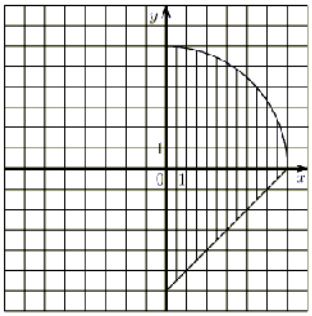
синтаксически верными являются:

1) а, б, г; 2) а, б, в; 3) а, б, г, д; 4) а, б, д; 5) все.

8. Выбрать фрагмент программы на Pascal, в котором переменной Y присваивается значение, равное 0, если $-5 < X < 5$; в противном случае переменной Y присваивается значение, равное 1:

1) If X > –5 Then If X < 5 Then Y:= 0; Y:= 1;
2) If X > –5 Then If X < 5 Then Y:= 0 Else Y:= 1;
3) If X < –5 Then If X > 5 Then Y:= 1 Else Y:= 0;
4) Y:= 1; If X > 5 Then If X < –5 Then Y:= 0;
5) If X > –5 Then Y:= 0 Else Y:=1; If X < 5 Then Y:= 1.

9. Выбрать выражение, принимающее значение TRUE, если точка с координатами (x, y) попадает внутрь заштрихованной области (единичный отрезок равен одной клетке):



1) (Sqr(x) + Sqr(y) <= 36) Or (x >= 0) And (y <= 0) And (y – x >= –6);
2) (Sqr(x) + Sqr(y) <= 36) And (x >= 0) And (y >= 0) Or (x >= 0) And (y <= 0) And (y–x >= –6);
3) (Sqr(x) + Sqr(y) >= 36) And (x >= 0) And (y >= 0) Or (x >= 0) And (y <= 0) And (y – x >= –6);
4) (Sqr(x) + Sqr(y) <= 36) And (x >= 0) And (y >= 0) And (x >= 0) And (y <= 0) And (y – x >= –6);
5) (Sqr(x) + Sqr(y) <= 36) Or (x >= 0) Or (y >= 0) Or (x >= 0) Or (y <= 0) Or (y – x >= –6).

10. По окончанию исполнения фрагмента программы

```
If (A > 0) And (B > 0) Then S := Sqrt(A * B) Else S := (A + B) / 2;
```

при $A = 12$ и $B = -3$ будет получено значение S, равное

1) –6.0; 2) 6.0; 3) –4.5; 4) 4.5; 5) 7.5.

11. По окончанию исполнения фрагмента программы

```
S := 0; For I := 1 To N Do S := S – i;
```

при $N = 4$ будет получено значение S, равное

1) 0; 2) 4; 3) –10; 4) 10; 5) –4.

12. Значения переменных *S* и *J* после выполнения фрагмента программы на Pascal

```
S := 0; J := 4; Repeat S := S + J; J := J – 1 Until J > 0;
```

будут равны

1) J = 0, S = 10; 2) J = 3, S = 4; 3) J = –1, S = 10; 4) J = 3, S = 7; 5) J = 1, S = 9.

13. Цикл в фрагменте программы (*A*, *B* — целые)

```
Repeat If A > B Then A := A – B Else B := B – A Until A = B;
```

при $A = 48$ и $B = 48$ будет исполнен

1) 0 раз; 2) 1 раз; 3) 4 раза; 4) 5 раз; 5) бесконечное число раз.

14. Сумма первых *N* натуральных чисел не будет определена алгоритмом

1) S := 0; For I := 1 To N Do S := S + I;
2) S := (1 + N) * N Div 2;
3) S := 0; I := 1; While I <= N Do Begin S := S + I; I := I + 1 End;
4) S := 0; For I := N DownTo 1 Do S := S + I;
5) среди ответов нет правильного.

15. Массивы в Pascal могут быть

1) только линейными; 2) только линейными и двухмерными; 3) любой размерности; 4) только двухмерными; 5) линейными, двухмерными и трехмерными.

16. В качестве индексов элементов массива не могут быть использованы величины типа

1) Integer; 2) Boolean; 3) Char; 4) Real; 5) Byte.

17. Элементы массива a[1], …, a[6] имеют соответственно значения x, x^2, $x^2 - x$, x^3, $x^3 - x$, $x^3 - x^2$. При $x = 2$ значение выражения a[a[a[3]]] + a[a[2] + a[1] – a[3]] равно

1) 2; 2) 4; 3) 8; 4) 16; 5) 5.

18. Элементы массива $A[1, 1]$, $A[1, 2]$, $A[2, 1]$, $A[2, 2]$ имеют соответственно значения b, $-b$, b^2, $-b^2$. Значение выражения $A[2, A[A[1, 1] - 1, -A[1, 2] - 1]] * A[-A[2, 2] - 2, -A[2, 1] + 5]$ при $b = 2$ равно

1) 16; 2) –16; 3) 8; 4) –8; 5) 4.

19. Фрагмент программы

```
For I := 1 To N Do For J := 1 To N Do If (I = J) Or (I + J = N + 1) Then A[I, J] := I Else A[I, J] := 10;
```

заполняет квадратную матрицу размером $N \times N$ по следующему принципу:

1) всю матрицу числом 10; 2) на главную диагональ помещается число, равное номеру строки, а в качестве остальных элементов — число 10; 3) на главную и побочную диагонали помещается число, равное номеру строки, а в качестве остальных элементов — число 10; 4) на главную и побочную диагонали помещается число 10, а все остальные элементы совпадают с номером строки; 5) на главную и побочную диагонали помещается число, равное номеру столбца, а в качестве остальных элементов — число 10.

20. Фрагмент программы

```
Write('Введите натуральное k: '); ReadLn(k);
S := 0; R := 0;
For I := 1 To N Do Begin S := S + A[k, I]; If A[I, k] < 0 Then R := R + 1 End;
```

определяет

1) сумму и количество отрицательных элементов массива; 2) сумму элементов k-й строки и число отрицательных элементов k-го столбца в массиве; 3) сумму элементов k-го столбца и число отрицательных элементов k-й строки в массиве; 4) число отрицательных элементов в k-й строке и k-м столбце в массиве; 5) сумму элементов в k-й строке и k-м столбце в массиве.

21. Для построения алгоритма поиска в таблице $A[1..N]$ максимального индекса элемента, равного x, нужно записать блоки

A	For k := 1	B	For i := N
C	To N Do	D	DownTo 1 Do
E	If (x = a[k])	F	If (x = a[i])
G	Then s := a[i];	H	Then P := i;
I	Then P := k;	J	P := 0;

в следующем порядке:

1) ACEI; 2) BCFG; 3) BDFH; 4) ADEI; 5) ADEI.

22. Фрагмент программы

```
S := 1; R := 1; M1 := A[1, 1]; M2 := A[1, 1];
For I := 1 To N Do
For J := 1 To M Do
Begin  If A[I, J] < M1 Then Begin M1 := A[I, J]; R := I End;
       If A[I, J] > M2 Then Begin M2 := A[I, J]; S := I End;
End;
For I := 1 To M Do Begin Vsp := A[S, I]; A[S, I] := A[R, I]; A[R, I] := Vsp End;
```

выполняет следующее действие:

1) меняет местами строки с заданными номерами S и R; 2) меняет местами столбцы с заданными номерами S и R; 3) меняет местами строки, в которых впервые появляются соответственно минимальный и максимальный элементы массива; 4) меняет местами столбцы, в которых впервые появляются соответственно минимальный и максимальный элементы массива; 5) копирует строку с заданным номером S в строку с номером R.

23. Значение строковой переменной *S* есть 'космодром'. В переменной *S* можно получить значение 'содом' с помощью фрагмента программы:

1) S := Copy(S, 3, 5);
2) S := S[3] + S[2] + S[6] + S[2] + Copy(S, 4, 2);
3) Delete(S, 1, 2); Delete(S, 2, 1); Delete(S, 4, 1);
4) Delete(S, 1, 2); Insert('o', S, 4); Delete(S, 4, 1);
5) S := S[3] + Copy(S, 5, 2) + 'дом'.

24. По окончанию исполнения фрагмента программы

```
Function F(N : LongInt) : Byte;
Begin If (N > 0) And (N < 10) Then F := N Else F := F(N Div 10) + N Mod 10 End;
Begin WriteLn(F(34215)) End.
```

будет выведено

1) 5; 2) 15; 3) 10; 4) сообщение об ошибке; 5) 0.

25. Присвоить переменной *C* значение выражения A^B (*A*, *B* — натуральные) не позволяет фрагмент программы

1) C := 1; For I := 1 To B Do C := C * A;
2) Function St(M, N : Integer) : LongInt;
   Var V : LongInt;
   Begin V := 1; While N > 0 Do Begin V := V * M; N := N – 1 End; St := V End;
   Begin A := 4; B := 4; C := St(A, B) End.
3) Function St(M, N : Integer) : LongInt;
      Begin If N = 1 Then St := 1 Else St := St(M, N – 1) * M End;
   Begin A := 4; B := 4; C := St(A, B) End.
4) Var A, B, C : LongInt;
      Procedure St(M, N : Integer) : LongInt;
      Var C : LongInt;
      Begin C := 1; While N > 0 Do Begin C := C * M; N := N – 1 End End;
   Begin A := 4; B := 4; St(A, B) End.
5) C := Round(Exp(B * Ln(A)));

26. Объявлен следующий тип данных

```
Type Fam = Record F : String[30]; Mes : 1..12; Voz : 1..7; Gr : 1..10 End;
```

Объём памяти, занимаемый величиной типа Fam, составляет в байтах

1) 59; 2) 4; 3) 60; 4) 34; 5) 100.

27. Результатом вычисления значения выражения *A* – *B* – *C* * *B*
при *A* = [5..10], *B* = [1, 5, 7, 10, 12..15], *C* = [6..12] будет множество

1) [6, 8, 9]; 2) [1, 5..15]; 3) []; 4) [7, 10, 12]; 5) [5].

28. Дано натуральное число *N*. Фрагмент алгоритма

```
M := 0;
While N <> 0 Do Begin If N Mod 10 > M Then M := N mod 10; N := N div 10 End;
```

1) находит минимальную цифру в записи числа; 2) находит цифру в самом старшем разряде числа; 3) находит максимальную цифру в записи числа; 4) находит цифру в самом младшем разряде числа; 5) находит любую цифру в записи числа, отличную от нуля.

29. Произвольный (прямой) доступ к компонентам файла возможен в

1) только типизированных файлах; 2) только нетипизированных файлах; 3) в текстовых файлах; 4) только в типизированных и нетипизированных файлах; 5) в типизированных, нетипизированных и текстовых файлах.

30. При работе с файлами не используется процедура

1) Read; 2) Write; 3) Assign; 4) Reset; 5) Delete.

31. Компонентом типизированного файла не может быть

1) запись; 2) файл; 3) массив; 4) строка; 5) среди ответов 1–4 нет правильного.

32. В приведенном ниже фрагменте программы (F — файловая переменная)

```
Randomize; Repeat B := –100 + Random * 200; WriteLn(F, B : 10 : 6) Until Abs(B) < 1e–7;
```

осуществляется запись данных в

1) типизированный файл; 2) нетипизированный файл; 3) текстовый файл; 4) типизированный или нетипизированный файл; 5) приведенный фрагмент программы является ошибочным.

33. По окончанию исполнения фрагмента программы

```
WriteLn(6 * Ord(1 = 1) And Ord('b' = 'b'));
```

будет выведено

1) 6 * TRUE And TRUE; 2) FALSE; 3) TRUE; 4) сообщение об ошибке; 5) 0.

34. В приведенном ниже фрагменте программы (F — файловая переменная; $0 \le K \le$ FileSize(F) – 1)

```
For I := FileSize(F) – 1 DownTo K Do
    Begin Seek(F, I); Read(F, Vsp); Write(F, Vsp) End;
    Seek(F, K); Write(F, R);
```

выполняется следующее действие

1) вместо элемента с номером *K* записывается элемент, значение которого *R*; 2) файл начиная с элемента с номером *K* заполняется значением, равным *R*; 3) в *K*-ю позицию в файле вставляется элемент, значение которого равно *R*; 4) все элементы файла, номер которых больше *K*, удаляются, а в *K*-ю позицию записывается элемент, значение которого равно *R*; 5) все элементы файла, номер которых меньше *K*, удаляются, а в *K*-ю позицию записывается элемент, значение которого равно *R*.

35. Основное отличие динамической переменной от статической заключается в следующем:

1) динамическая переменная может чаще изменять своё значение, чем статическая; 2) время доступа к динамическим переменным меньше, чем к статическим; 3) для динамических величин применяется принципиально другой способ представления их в памяти ЭВМ; 4) динамические переменные могут создаваться и уничтожаться во время работы программы; 5) динамические переменные практически не расходуют оперативную память ЭВМ.

36. Компоненты модуля: *а) заголовок*; *б) интерфейсный раздел*; *в) раздел реализации*; *г) раздел инициализации* располагаются в порядке

1) а, б, в, г; 2) а, г, в, б; 3) а, в, г, б; 4) а, б, г, в; 5) г, б, а, в.

37. Имеется описание

```
Type    BaseType = LongInt; Plist = ^Tlist; Data = Record X, Y : BaseType End;
        Tlist = Record D : Data; N, P : Plist End;
Var             X, Y : Plist;
```

Указать недопустимое присваивание

1) X^.D.X := Y^.D.Y; 2) X^.D := Y^.D; 3) X^.N := X^.P; 4) X^.P := Y; 5) X^.D := Y^.N.

38. Дан условный оператор

```
If (Sun = 'село') or (rain = 'не идет') then b=0 else b=1;
```

Переменная *b* будет равна 1,

1) только если и *Sun* не равно *'село'* и *rain* не равен '*не идет*'; 2) при выполнении хотя бы одного из условий *Sun = 'село'* или *rain = 'не идет';* 3) при выполнении хотя бы одного из

условий *Sun = 'село'* или *rain = 'идет';* 4) если какая-нибудь из двух переменных не равна *'идет'*; 5) при равенстве хотя бы одной переменной *'село'*.

39. При каких значениях переменных *A, B, C* логическое выражение *A* and not (*B* or not *C*) будет истинным?

1) *A* = True, *B* = True, *C* = True;
2) *A* = True, *B* = True, *C* = False;
3) *A* = True, *B* = False, *C* = False;
4) *A* = False, *B* = True, *C* = False;
5) *A* = False, *B* = False, *C* = False.

40. Дан фрагмент программы

```
Var I, J, K : Word;
Begin        ReadLn(I, J, K);
     Repeat I := (I + J) Mod K; J := 2 * J – I Div K Until I = J;
     WriteLn(I)
End.
```

При *I* = 2, *J* = 2, *K* = 4 результатом выполнения данной программы будет

1) 2; 2) 4; 3) ошибка исполнения; 4) зацикливание; 5) 32768.

В А Р И А Н Т I I

1. Наименьшее целое число со знаком, кодируемое 16 битами, равно

1) –255; 2) –32768; 3) –32767; 4) –65535; 5) –65536.

2. Алгоритмы, которые решают некоторую подзадачу главной задачи и, как правило, выполняются многократно, называются

1) вспомогательными; 2) линейными; 3) адаптивными; 4) циклическими; 5) основными.

3. Интерпретатор осуществляет

1) полный предварительный перевод программы на язык машинных команд;
2) перевод отдельного оператора программы на язык машинных команд и тут же его выполняет;
3) выполнение программы на языке машинных команд, полученной после компиляции;
4) перевод программы на язык Ассемблер;
5) перевод отдельного оператора программы на язык Ассемблер и тут же его выполняет.

4. Среди перечисленных типов данных в Pascal структурированным не является:

1) строковый; 2) множественный; 3) файловый; 4) перечисляемый; 5) комбинированный.

5. Имеются описания

```
Type Digit = ’0’..’9’;
Var D : Digit ; K : 0..9; N : Integer;
```

Недопустимо присваивание

1) D := ’7’ ; 2) K := 5 mod 2; 3) K := Ord(d); 4) K := Ord(d) – Ord(‘0’); 5) N := K.

6. Из утверждений

а) диапазон 1..260 является поддиапазоном типа Byte;
б) диапазон 0..65585 является поддиапазоном типа Word;
в) диапазон ’a’..’z’ является поддиапазоном типа Char;
г) для вещественных переменных обычно применяется тип Real;
д) значение 32000 входит в тип Integer

верными являются 1) б, г, д; 2) а, б, г; 3) г, д; 4) а, б, в; 5) в, г, д.

7. По окончанию исполнения фрагмента программы

```
WriteLn(N Div 1000 = N Mod 10);
```

при N = 1231 на экран будет выведено

1) –1; 2) 0; 3) FALSE; 4) TRUE; 5) сообщение об ошибке.

8. Фрагмент программы

```
K := 0; I := 1;
While (I <= Length(S)) And (S[I] <> '!') do
begin If S[I] In ['0'..'9'] Then K:= K + 1; I := I + 1 end;
```

1) подсчитывает количество цифр в строке;
2) подсчитывает количество символов, предшествующих символу '!';
3) подсчитывает сумму цифр, предшествующих символу '!';
4) подсчитывает количество и сумму цифр в строке;
5) подсчитывает количество цифр, предшествующих первому символу '!'.

9. Чему равны значения переменных P и Q после выполнения последовательности действий:

```
P := 4*5 div 3 mod 5;
Q := 34 mod P*5 – 29 mod 5*2;
```

1) P = 2, Q = –8; 2) P = 1, Q = –5; 3) P = 6, Q = –5; 4) P = 2, Q = –5; 5) P = 1, Q = –8.

10. Имеется условный оператор If d <> 10 Then Writeln ('Ура!') Else Writeln ('плохо…'); Какие из следующих операторов ему эквивалентны

а) If d = 10 Then Writeln ('Ура!') Else Writeln ('плохо…');
б) If not (d =10) Then Writeln ('Ура!') Else Writeln ('плохо…');
в) If not(d =10) Then Writeln ('плохо…') Else Writeln ('Ура!');
г) If not (d <> 10) Then Writeln ('плохо…') Else Writeln ('Ура!');

1) а, в 2) б, г 3) а, б 4) б, в, г 5) а, г.

11. В настоящей программе

```
Var X, Y, R : Real;
Begin
    WriteLn('Введите два числа: '); ReadLn(X, Y);
    Case X < Y Of
        True: R := X;
        False: R := Y
    End;
    WriteLn(R : 8 : 2)
End.
```

определяется

1) произведение двух чисел; 2) минимальное из двух чисел; 3) максимальное из двух чисел; 4) среднее арифметическое двух чисел; 5) логическое значение TRUE, если первое из двух введенных чисел меньше второго.

12. Чему будут равны значения переменных A и M после выполнения фрагмента программы на Pascal, если начальные значения переменных A = 3, B = 4, C = 9?

```
If A > B Then M:= A Else M:=B; If C > M Then M:=C; M:=M – A; A:= B + M;
```

1) A=7, M=0 2) A=13, M=6 3) A=10, M=9 4) A=4, M=0 5) A=10, M=6.

13. Чему будут равны значения переменных P и Q после выполнения фрагмента программы на Pascal, если начальное значение переменной k = 3?

```
P:= True; Q:= 1;
Case 2*k mod 2 Of
    3, 2, 7, 5 : Q:= k;
    4, 8 : Begin P:= False; Q:= 2 End;
    9, 6 : Begin P:= False; Q:= 3 End
Else Begin P := False; Q := k End
```

```
End;
```

1) P = True, Q = 36; 2) P = True, Q = 1; 3) P = False, Q = 2; 4) P = False, Q = 3; 5) P = False, Q = 36.

14. Среди стандартных подпрограмм для строковых величин функцией является

1) Insert (S, S1, k); 2) Delete (S, k, n); 3) Copy (S, k, n); 4) Str (n, S); 5) Val (S, n, Code).

15. Цикл с параметром применяется в тех случаях, когда

1) никакие другие циклы не подходят;
2) известно условие окончания цикла;
3) выполнение тела цикла осуществляется хотя бы один раз;
4) известно, сколько раз должно быть выполнено тело цикла;
5) осуществляется работа с массивом.

16. Тело цикла в программе

```
Program Test;
Var I, J : Word;
Begin     J := 0; I := 5;
          Repeat Inc (J); I := I + (J – 1) Until I mod 5 = 0;
          WriteLn (I, ' ', J);
End.
```

выполнится

1) 5 раз; 2) 2 раза; 3) 1 раз; 4) бесконечное число раз; 5) ни разу.

17. Циклу For I := N1 To N2 Do S; эквивалентен набор операторов

1) I := N1; While I <= N2 Do S; I := I + 1;
2) I := N1; Repeat S; I := I + 1 Until I <= N2;
3) I := N1; If N1 <= N2 Then Repeat S; I := I + 1 Until I > N2;
4) I := N1; While I > N2 Do begin S; I := I + 1 end;
5) I := N1; Repeat S Until I + 1 > N2;

18. Тип элементов массива может быть

1) любым простым; 2) любым, исключая файловые; 3) только порядковым; 4) любым скалярным, кроме вещественного; 5) целочисленным.

19. Среди описаний массивов

а) Var M : Array [Byte] of 1..10;
б) Const Count = 6; Var M : Array [1..2*Count] of Char;
в) Var M : Array [–9..9] of Real;
г) Type Mass = Array [1..10] of Integer; Var M : Array [1..10] Of Mass;

синтаксически верными являются

1) а, в, г 2) б, в 3) а, б, в 4) нет верных 5) все верные.

20. Определить, что будет выдано на печать программой

```
Program Print;
Type Vect = Array [1..2] of Real; Var A : Vect; I : Integer;
     Procedure R(Var K : Integer; Var X : Real);
     Begin K := 2; X := 0 End;
     Begin
       A[1] := 1; A [2] := 2;
       I := 1; R(I, A[1]);
       Writeln(A[1] : 3 : 1, ' ', A[2] : 3 : 1)
     End.
```

1) 0.0 2.0; 2) 2.0 0.0; 3) 1.0 2.0; 4) 1.0 0.0; 5) 2.0 2.0.

21. При исполнении фрагмента программы

```
Var C : Integer;
    Procedure R1(Var A : Integer; C : Boolean);
        Procedure R2;
        Var C : String;
        Begin WriteLn(C); A := 1 End;
    Begin C := True; R2 End;
Begin C := 100; R1(C, False) End.
```

будет напечатано значение переменной *C*

1) True; 2) 1; 3) 100; 4) неизвестно, поскольку значение переменной *C* не определено; 5) 'мама'.

22. Значения переменных *S* и *J* после выполнения фрагмента программы на Pascal

```
S := 0; J := 0; While J < 5 Do J := J + 1; S := S + 2 * J;
```

будут соответственно равны

1) J = 5, S = 30; 2) J = 4, S = 8; 3) J = 5, S = 10; 4) J = 4, S = 20; 5) J = 5, S = 20.

23. Цикл в фрагменте программы

```
P := 4; While P > 0.0001 Do P := P * 0.1;
```

будет исполнен

1) 0 раз; 2) 1 раз; 3) 4 раза; 4) 5 раз; 5) бесконечное число раз.

24. Выбрать фрагмент программы на Pascal, в котором переменной *P* присваивается значение выражения 2^5.

1) P := 1; J := 0; While J > 5 Do J := J + 1; P := P * 2;
2) P := 1; J := 0; While J < 5 Do J := J + 1; P := P * 2;
3) P := 1; J := 0; While J > 5 Do begin J := J + 1; P := P * 2 end;
4) P := 1; J := 0; While J < 5 Do begin J := J + 1; P := P * 2 end;
5) P := 0; J := 0; While J < 5 Do begin J := J + 1; P := P * 2 end;

25. Значения переменных *S* и *J* после выполнения фрагмента программы на Pascal

```
J := –2 ; S := 0;
While J < 2 Do
Begin J := J + 1; If J = 0 Then A := 1 Else A := 1 / J; S := S + A End;
```

будут соответственно равны

1) J = 1, S = 1.0; 2) J = 2, S = 1.0; 3) J = 2, S = 0.5; 4) J = 2, S = 1.5; 5) J = 1, S = 0.5.

26. Формальными называются параметры, которые

1) передаются процедуре (функции) при ее вызове; 2) описываются в заголовке процедуры (функции); 3) объявляются в процедуре (функции) после заголовка; 4) объявляются в разделе описаний основной программы; 5) нигде не описываются.

27. Выбрать функцию, которая правильно вычисляет факториал от числа *N*:

1) Function F (N : Integer): Integer; Begin F := N*F(N – 1) End;
2) Function F (N : Integer): Integer; Begin If N = 0 Then F := 1 Else F := F(N + 1) / (N + 1) End;
3) Function F (N : Integer): Integer; Begin If N = 0 Then F := 1 Else F := N * (N – 1) * F(N + 1) End;
4) Function F (N : Integer): Integer; Begin If N = 0 Then F := 1 Else F := N * F(N – 1) End;
5) Function F (N : Integer): Boolean; Begin If N = 0 Then F := 1 Else F := N * F(N – 1) End;

28. Программой

```
Program PP;
Var A, B, C, D : Integer;
    Procedure P (Var B : Integer; C : Integer);
    Var D : Integer;
    Begin A := 5; B := 6; D := 8; Write (A, B, C, D,' '); End;
```

```
Begin   A := 1; B := 2; C := 3; D := 4;
        P (A, B); Write (A, B, C, D);
End.
```

будет напечатано

1) 6628 1234; 2) 5628 6234; 3) 5638 1234; 4) 5238 6234; 5) 6628 6234.

29. Пересечением двух множеств называется множество элементов,

1) принадлежащих хотя бы одному из этих множеств;
2) принадлежащих одновременно и первому, и второму множествам;
3) принадлежащих первому множеству, но не принадлежащих второму;
4) принадлежащих второму множеству, но не принадлежащих первому;
5) базового типа, исключая те элементы, которые принадлежат заданным множествам.

30. Результат выполнения операции [1, 3..9, 15, 16] * [8..15, 20] равен

1) [1, 3..7, 16]; 2) [10..14, 20]; 3) [8, 9, 15]; 4) [1, 3..16, 20]; 5) [15].

31. Результат [1..3, 5, 7, 11] – [3..8, 10, 12, 15..20] равен

1) [1, 2, 11]; 2) [5, 7, 10..12, 15..20]; 3) [1, 2, 10..12, 15..20]; 4) [11]; 5) [1, 2].

32.
```
Type Krug = Record radius: Real; center: Record x, y : Real End End;
Var K : Krug;
```

Требуется переменной *K* присвоить значение, соответствующее кругу радиуса 2.5 с центром в точке (0, 1.8). В каких из следующих операторов присоединения правильно решается эта задача?

а) With K Do begin radius := 2.5; center.x := 0; center.y := 1.8 end;
б) With K, center Do begin radius := 2.5; x := 0; y := 1.8 end;
в) With K Do begin radius := 2.5; With center Do begin x := 0; y := 1.8 end end;
г) With center, K Do begin radius := 2.5; x := 0; y := 1.8 end;

1) а, в; 2) а, б, в; 3) а; 4) б, в, г; 5) во всех.

33.
```
Type Complex = Record re, im : Real end; Point = Record x, y : Real end;
Var z, w : Complex; p : Point; re : Real;
```

Определить, какие значения будут иметь переменные p и re после выполнения следующих операторов:

```
With z Do begin re := 0; im := 1 end;
w := z; re := 2;
With z Do begin re := 1;
With z, w Do im := –im;
With p Do begi x := re; y := w.im end;
```

1) p.x = 2, p.y = 1, re = 2; 2) p.x = 1, p.y = 2, re =1; 3) p.x = 2, p.y = –1, re = 2; 4) p.x = 1, p.y =1, re = 2; 5) p.x = 2, p.y = –1, re = 1.

34. Из описаний файловой переменной недопустимо

1) Var F : File of Real; 2) Var F1, F2 : File of Char; 3) Var F : Text; 4) Var F : File of File; 5) Var F : File of String;

35. Среди операторов

а) Assign (F1, ‘A:STR1.dat‘); б) Reset (F1, F2); в) ReWrite ; г) Assign (F2, C :\ TT\ TAB1.dat); д) ReWrite (F1);

не содержат синтаксических ошибок

1) а, б; 2) а, б, в; 3) а, в, д; 4) а, д; 5) г, д.

36.
```
Var F : File of Integer; X, Y : Integer;
```

Пусть файл F содержит два элемента: 3 и 7; определить, какое значение будет иметь переменная *Y* после выполнения операторов:

Reset (F); Y := 1; Repeat Read (F, X); Y := Y * X Until Eof (F);

1) 3 2) 7 3) 21 4) 10 5) нет правильного ответа.

37. Определить содержимое файла F после выполнения операторов

```
ReWrite (F);
If Eof (F) Then Write (F, 1) Else Write (F, 2);
If Eof (F) Then Write (F, 3) Else Write (F, 4);
```

1) 1, 3 2) 2, 4 3) 2, 3 4) 1, 4 5) нет правильного ответа.

38. Исполнен фрагмент программы (*A, B, C, D* — указатели, которые ссылаются на разные ячейки, и других указателей в программе нет)

A^ := C^; C^ := 7; D:= C; B^ := B^ + D^; C := Nil;

Указать все переменные, на которые утеряны ссылки:

1) A^; 2) A^ и B^; 3) B^; 4) D^; 5) C^ и D^.

39. Интерфейсная часть модуля содержит

1) операторы, необходимые для инициализации модуля; 2) объявления, включая процедуры и функции, представленные списком заголовков и доступные пользователям; 3) тела процедур и функций, перечисленных ранее; 4) служебное слово UNIT и имя модуля; 5) стандартные процедуры и функции.

40. Для того чтобы оператор

a[i]^[k].m := 2;

был синтаксически корректен, переменная *a* должна быть описана как

1) запись, содержащая поле типа ссылки на массив целых; 2) массив записей, содержащих поле типа ссылки на запись, содержащую массив целых; 3) массив ссылок на массив записей, имеющих поле типа целое; 4) массив записей, содержащих поле типа записи, имеющей поле типа ссылки на целое; 5) запись, содержащую поле типа массива ссылок на массив записей, содержащих поле типа целое.

В А Р И А Н Т I I I

1. Логическое выражение

(N Mod 10 Mod 2) Or (N Div 10 Mod 10 Mod 2 = 0) Or (N Mod 100 Div 10 Mod 2 = 0)

должно принимать значение TRUE тогда и только тогда, когда истинно высказывание:

1) в трёхзначном натуральном числе все цифры чётные; 2) в трёхзначном натуральном числе одна чётная цифра; 3) в трёхзначном натуральном числе две чётных цифры; 4) в трёхзначном натуральном числе хотя бы одна чётная цифра; 5) в трёхзначном натуральном числе нет чётных цифр.

2. Ошибку "Structure too large" (структура превышает максимально допустимый объём статической памяти) вызовет описание

1) Type Vector = Array[Byte] Of Integer; Var C : Array[1..10] Of Vector;
2) Ver T : File Of String;
3) Type A = Record S : String; A, B, C : Array[10..20] Of Real End;
   Var M : Array[1..5, 1..8] Of A;
4) Var K : Array [Byte, Byte] Of String[6];
5) Var S : Array[–10000..10000] Of Sring[2].

3. К процедурам для работы с динамическими переменными не относится

1) Mark; 2) New; 3) Release; 4) Seek; 5) Dispose.

4. Имеется описание

Type A = Array[0..100] Of Real; B = ^A; Var M : Array[1..5] Of B;

Для хранения массива *M* необходим объём памяти (байт)

606; 2) 4; 3) 20; 4) 12120; 5) 6.

5. Фрагмент программы

```
K := 0;
While Not Eof(F) Do
Begin     ReadLn(F, S); I := 1;
  While I <= Length(S) Do
  Begin  If S[I] In ['A'..'Я', 'a'..'п', 'р'..'я']
           Then   Begin  K := K + 1;
                         Delete(S, I, 1); I := I – 1
                  End;
           I := I + 1
  End
End;
```

1) удаляет из текстового файла F все русские буквы; 2) определяет в текстовом файле количество символов, являющихся русскими буквами; 3) определяет в текстовом файле количество символов, не являющихся русскими буквами; 4) определяет в текстовом файле количество символов; 5) удаляет из текстового файла F все символы, не являющиеся русскими буквами.

6. В фрагменте программы (F : File Of Integer; I, K, Vsp : Integer;)

```
Reset(F); K := FileSize(F) – 1;
For I := 0 To K Do
Begin Seek(F, I); Read(F, Vsp); Seek(F, FileSize(F)); Write(F, Vsp) End;
```

выполняется

1) сортировка файла; 2) изменение порядка следования элементов на обратный; 3) дописывание в конец исходного файла полной его копии с сохранением порядка следования элементов; 4) дописывание в конец исходного файла полной его копии с изменением порядка следования элементов на противоположный; 5) не выполняется никаких действий по изменению файла.

7. Имеется описание

Type Dn = (pn, vt, sr, cht, ptn, sb, vs); Mn = Set Of Dn; Var V : Mn;

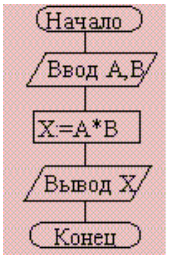
и фрагмент программы

V := [pn..ptn] * [sr, ptn..vs] – [sb];

После исполнения этого фрагмента переменная *V* имеет значение

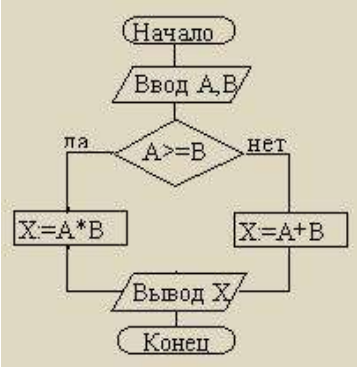
1) [pn..vs]; 2) [sr, ptn]; 3) [sb]; 4) []; 5) [pn..ptn].

8. Алгоритм какого типа изображен на блок-схеме?



1) циклический; 2) разветвляющийся; 3) вспомогательный; 4) линейный; 5) комбинация развилки и цикла.

9. После исполнения фрагмента программы, изображенного на блок-схеме,



при $A = 5$, $B = 4$ значение X будет равно

1) 20; 2) 9; 3) 5; 4) 4; 5) 1.

10. В приведенном фрагменте программы (N типа LongInt, $N > 0$)

```
P := 1;
While P <= N Do Begin
        Left := N Div (P * 10) * (P * 10); Right := N Mod P;
        K := ((N Mod (P * 10) Div P + 1) Mod 10) * P;
        N := Left + K + Right;
        P := P * 10 End;
```

натуральное число N изменяется по следующему правилу:

1) не изменяется; 2) в каждый разряд прибавляется 1; 3) из каждого разряда вычитается 1; 4) в каждый разряд прибавляется 1, если значение в разряде — не девять, иначе заменяется на нуль; 5) каждая девятка в десятичной записи числа заменяется на нуль.

11. Цикл с предусловием выполняется так:

1) выполняется тело цикла, изменяется параметр цикла, проверяется условие продолжения выполнения цикла; 2) изменяется параметр цикла, проверяется условие продолжения выполнения цикла, выполняется тело цикла; 3) проверяется условие продолжения выполнения цикла, выполняется тело цикла; 4) тело цикла выполняется N раз (N — натуральное); 5) определяется, сколько раз должен быть выполнен цикл, и далее цикл с предусловием сводится к циклу с параметром.

12. В текстовом файле каждая строка заканчивается

1) числами 10 и 13; 2) символами с кодами 10 и 13; 3) символом с кодом 13; 4) числом 0; 5) символом с кодом 10.

13. Процедуры ReadLn и WriteLn можно использовать при работе с

1) типизированными файлами; 2) нетипизированными файлами; 3) типизированными и нетипизированными файлами; 4) текстовыми файлами; 5) любыми файлами.

14. Значение выражения Ord(x > y) + Ord(Ord(z = 'F')) при $x = 7$, $y = 0$, z = 'F' равно

1) TRUE; 2) FALSE; 3) 0; 4) 1; 5) 2.

15. Идентификатор в Turbo Pascal не может начинаться с

1) латинской буквы; 2) заглавной латинской буквы; 3) цифры; 4) знака подчёркивания; 5) латинской буквы, а затем знака подчёркивания.

16. В приведенном фрагменте программы (First — ссылка на первый элемент списка; список объявлен следующим образом: Type SS = ^List; List = Record A : LongInt; Next : SS End;)

```
P := First; S := 0; While Not (P = Nil) Do Begin S := S + 1; P := P^.Next End;
```

определяется

1) первый элемент списка; 2) сумма элементов списка; 3) сумма первого и последнего элементов списка; 4) количество элементов списка; 5) количество звеньев списка, где указатель на следующее звено не Nil.

17. При исполнении фрагмента программы

```
Var C : Integer;
    Procedure R1(Var A : Integer; C : Boolean);
        Procedure R2;
        Var C : String;
        Begin A := 1 End;
    Begin C := True; R2 End;
Begin C := 100; R1(C, False); WriteLn(C) End.
```

будет напечатано значение переменной *C*

1) True; 2) 1; 3) 100; 4) неизвестно что, поскольку значение переменной *C* не определено; 5) False.

18. Цикл в фрагменте программы

```
P := 4; Repeat P := P * 0.1 Until P < 0.0001;
```

будет исполнен

1) 0 раз; 2) 1 раз; 3) 4 раза; 4) 5 раз; 5) бесконечное число раз.

19. Кодируется шестнадцатибитовое целое со знаком (тип Integer). 1111111111110000_2 является кодом числа

1) –15; 2) 15; 3) 16; 4) –16; 5) –30000.

20. Свойством алгоритма является

1) результативность; 2) цикличность; 3) возможность изменения последовательности выполнения команд; 4) возможность выполнения алгоритма в обратном порядке; 5) простота при записи на языках программирования.

21. Из перечисленных ниже в программе обязателен

1) раздел Var; 2) раздел Const; 3) раздел Type; 4) раздел Label; 5) раздел Begin … End.

22. Ввод данных — это

1) процесс передачи данных из оперативной памяти на внешний носитель; 2) процесс ввода с клавиатуры каких-либо значений; 3) передача данных от внешнего носителя в оперативную память для обработки; 4) присваивание конкретных значений переменным, которые используются в программе; 5) запись файла на диск.

23. Значение *R* после выполнения операции логического присваивания

```
R := Not (A Or B Or (X > 2) And (Y < 0))
```

при *A* = False, *B* = False, *X* = 3, *Y* = 2 будет равно

1) –1; 2) False; 3) True; 4) 0; 5) 1.

24. С помощью какой из приведенных серий команд переменной *B* присваивается значение выражения

$$\left(\frac{x+y}{x-y}-\frac{x-y}{x+y}\right)\cdot\left(\frac{x-y}{x+y}+\frac{x+y}{x-y}\right)$$

а) A := (x + y) / (x – y); B := (A – 1 / A) * (1 / A + A);
б) A := (x + y) / (x – y); B := Sqr (A) – Sqr (1 / A);
в) A := (x – y) / (x + y); B := Sqr (1 / A) – Sqr (A);

1) а; 2) б; 3) в; 4) всех трех; 5) ни один из ответов 1–4 не является верным.

25. Значения переменных *a* и *b* после выполнения следующих действий

a := 15 Div (16 Mod 7); b := 34 Mod a * 5 – 29 Mod 5 * 2;

будут равны

1) a = 1, b = 160; 2) a = 1, b = 4; 3) a = 7, b = 25; 4) a = 7, b = 22; 5) a = 7, b = 28.

26. Во фрагмент алгоритма

```
For K := 10 To 99 Do
Begin   P1 := K Div 10;
        P2 := K Mod 10;
        S := P1 + P2;
        If ____ Then WriteLn (K) End;
```

печатающего все двузначные числа, в записи которых есть цифра *N* или сумма цифр которых равна самим числам, нужно вписать логическое выражение

1) (P1 = N) Or (P2 = N) And (S = K); 2) (P1 = N) Or (P2 = N) Or (S = K); 3) (P1 = N) And (P2 = N) Or (S = K); 4) ((P1 = N) Or (P2 = N)) And (S = K); 5) (P1 = N) And (P2 = N) And (S = N).

27. Значения переменных *p* и *d* после выполнения фрагмента алгоритма

```
k := 47;
Case k Mod 9 Of
        5: Begin d := k; p := True End;
        0..2: Begin d := 2; p := False End;
        8: Begin d := 1; p := False End
        Else Begin d := 1; p := True End
        End;
```

равны

1) p = True, d = 1; 2) p = False, d = 2; 3) p = False, d = 3; 4) p = True, d = 47; 5) p = True, d = 2.

28. Тело цикла в программе

a := 1; b := 1; While a + b < 8 Do begin a := a + 1; b := b + 2 end;

выполнится

1) 1 раз; 2) 2 раза; 3) 3 раза; 4) ни разу; 5) бесконечное число раз.

29. Элементы массива p[1..5] равны соответственно 1, –1, 5, 2, 4. Значение выражения

p[1] * p[3] – p[2 * p[2] + p[p[5] – p[2]]]

равно

1) 8; 2) –8; 3) 12; 4) –12; 5) 6.

30. Задана строка *St*.Фрагмент алгоритма

S := 0; For I := 1 To Length (St) Do Begin Val (St[I], d, k); If K = 0 Then S :=S + d End;

1) определяет количество цифр в строке; 2) подсчитывает количество нулей в строке; 3) определяет сумму номеров позиций в строке, где стоят цифры; 4) подсчитывает сумму цифр в строке; 5) определяет сумму номеров позиций в строке, где стоят нули.

31. Какая из приведенных серий операторов определяет и печатает индекс последнего отрицательного элемента в линейном массиве из *n* элементов?

а) i := n; While (i >= 1) And (m[i] > 0) Do Dec (i); If i < 1 Then WriteLn (‘i = 0’) Else WriteLn (‘i = ’, i);

б) k := 0; For i := 1 To n Do If m[i] < 0 Then k := i; WriteLn (‘i = ’, k);

в) i := n; Repeat i := i – 1 Until (m[i] < 0); WriteLn (‘i = ’, i);

1) а, б; 2) б, в; 3) а, б, в; 4) б; 5) ни один из ответов 1–4 не верен.

32. Задан линейный массив M[1..n].

```
Function Control (M: Myarray): Boolean;
Var I : Integer;
Begin I := 1;
While (I <= n) And (M[I] > 0) Do Inc(I);
Control := (I <= n);
End;
```

Если в данном массиве все элементы положительные, приведенная функция возвращает значение

1) n; 2) True; 3) False; 4) I <= n; 5) ни один из ответов 1–4 не верен.

33. Задан двумерный массив X[1..n, 1..m]. Процедура

```
Procedure Sub (Var X: Myarray);
Var i, j: Integer;
Begin For i := 1 To n Do
For j := 1 To m Div 2 Do X[i, 2 * j] := X[i, 2 * j] + X[i, 1];
End;
```

1) к элементам столбцов в первой половине матрицы прибавляет элементы первого столбца соответствующей строки; 2) добавляет к матрице еще M столбцов с элементами, равными соответствующим элементам первого столбца; 3) к элементам четных столбцов прибавляет элементы первого столбца соответствующей строки; 4) к элементам четных строк прибавляет элементы первой строки соответствующего столбца; 5) меняет порядок столбцов таблицы.

34. Задан двумерный массив X[1..n, 1..m]. Функция

```
Function Check (X: Myarray): Boolean;
Var i, j : Integer; t : Boolean;
Begin   t := True; i := 1;
        While t And (i <= n) Do
                Begin  j := 1; While (j <= m) And (X[i, j] <> 0) Do Inc (j);
                        t := (j = m + 1); Inc (i)
                End; Check := Not t;
End;
```

возвращает значение

1) True, если все элементы массива ненулевые; 2) True, если в массиве есть элемент, равный нулю; 3) False, если в массиве есть элемент, равный нулю; 4) Not t; 5) ни один из ответов 1–4 не верен.

35. Среди перечисленных соответствий, которые необходимо соблюдать между формальными и фактическими параметрами:

а) соответствие по типу параметров; б) соответствие по количеству параметров; в) соответствие по типу используемых вспомогательных переменных; г) соответствие по порядку перечисления

лишним является

1) а; 2) б; 3) в; 4) г; 5) ни один из ответов 1–4 не верен.

36. Определите тип выражения (здесь A : Array[1..20] Of Real; B : Boolean; C : Integer)

```
C + Ord(Round(A[7]) + Ord(B)) – Trunc(A[1])
```

1) Real; 2) Integer; 3) Boolean; 4) LongInt; 5) среди ответов 1–4 нет верного.

37. Список объявлен следующим образом

```
Type Ukaz = ^Zveno; Zveno = Record X : String; N : Ukaz End;
Var First : Ukaz; {ссылка на начало списка}
```

В фрагменте программы

```
P := First;
While P^.N <> Nil Do
Begin B := P; M := P;
      While B <> Nil Do Begin If B^.X < M^.X Then M := B; B := B^.N End;
      S := P^.X; P^.X := M^.X; M^.X := S; P := P^.N
End;
```

выполняется

1) перемещение компонента к началу списка; 2) сортировка компонентов списка в порядке возрастания; 3) сортировка компонентов списка в порядке убывания; 4) перестановка соседних компонентов списка; 5) добавление в список нескольких новых компонент.

38. Фрагмент программы

```
S := A; A := B; B := S;
```

выполняет

1) обмен значений переменных *A, B*; 2) присваивание переменным *A, B* значения *S*; 3) замена значения переменной *A* значением переменной *B*; 4) во фрагменте не выполняется никаких действий; 5) замена значения переменной *B* значением переменной *A.*

39. Имеется следующее описание

```
Type U = ^Zveno; Zveno = Record X, Y : Boolean; Pred, Next : U End;
Var Logic : Boolean; A, B : Pointer; X, Y : U;
```

К ошибке компиляции «Несовместимость типов» приведет следующее присваивание:

1) A := X^.Next^.Next; 2) X := Y; 3) Logic := X^.X; 4) X^.Next := A; 5) X^ := Y^.Next.

40. Что изменяется при присваивании?

1) алгоритм; 2) имя переменной; 3) тип переменной; 4) значение переменной; 5) значение типизированной константы.