Министерство образования Иркутской области
Государственное бюджетное профессиональное образовательное учреждение Иркутской области
«Иркутский авиационный техникум»

УТВЕРЖДАЮ
Директор
ГБПОУИО «ИАТ»
_____ Якубовский А.Н.
«31» мая 2018 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ

ОП.08 Теория алгоритмов

специальности

09.02.03 Программирование в компьютерных системах

Иркутск, 2018

Рассмотрена
цикловой комиссией
ПКС протокол № 17 от
22.05.2018 г.

Председатель ЦК

 /М.А. Кудрявцева /

№	Разработчик ФИО
1	Еримеев Антон Михайлович

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Область применения фонда оценочных средств (ФОС)

ФОС по дисциплине является частью программы подготовки специалистов среднего звена по специальности 09.02.03 Программирование в компьютерных системах

1.2. Место дисциплины в структуре ППССЗ:

ОП.00 Общепрофессиональный цикл.

1.3. Цели и задачи дисциплины – требования к результатам освоения дисциплины

В результате освоения дисциплины обучающийся должен	№ дидактической единицы	Формируемая дидактическая единица
Знать	1.1	основные модели алгоритмов;
	1.2	методы построения алгоритмов;
	1.3	методы вычисления сложности работы алгоритмов
Уметь	2.1	разрабатывать алгоритмы для конкретных задач;
	2.2	определять сложность работы алгоритмов;
	2.3	Формальное определение алгоритма и вычисление функции

1.4. Формируемые компетенции:

ОК.1 Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.
ОК.2 Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
ОК.3 Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.
ОК.4 Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.
ОК.5 Использовать информационно-коммуникационные технологии в

профессиональной деятельности.
ОК.6 Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.
ОК.7 Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.
ОК.8 Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.
ОК.9 Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

ПК.1.1 Выполнять разработку спецификаций отдельных компонент.
ПК.1.2 Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

2. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДИСЦИПЛИНЫ, ИСПОЛЬЗУЕМЫЙ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ

2.1 Текущий контроль (ТК) № 1

Тема занятия: 1.1.4.Оценка эффективности алгоритма.
Метод и форма контроля: Лабораторная работа (Опрос)
Вид контроля: Компьютерное тестирование
Дидактическая единица: 1.1 основные модели алгоритмов;
Занятие(-я):
1.1.1.Введение в дисциплину. Понятие алгоритма. Типы алгоритмов, свойства.
1.1.2.Способы описания алгоритмов. Блок-схема

Задание №1
дать определение "детерминированность".
Образец ответа:
Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.

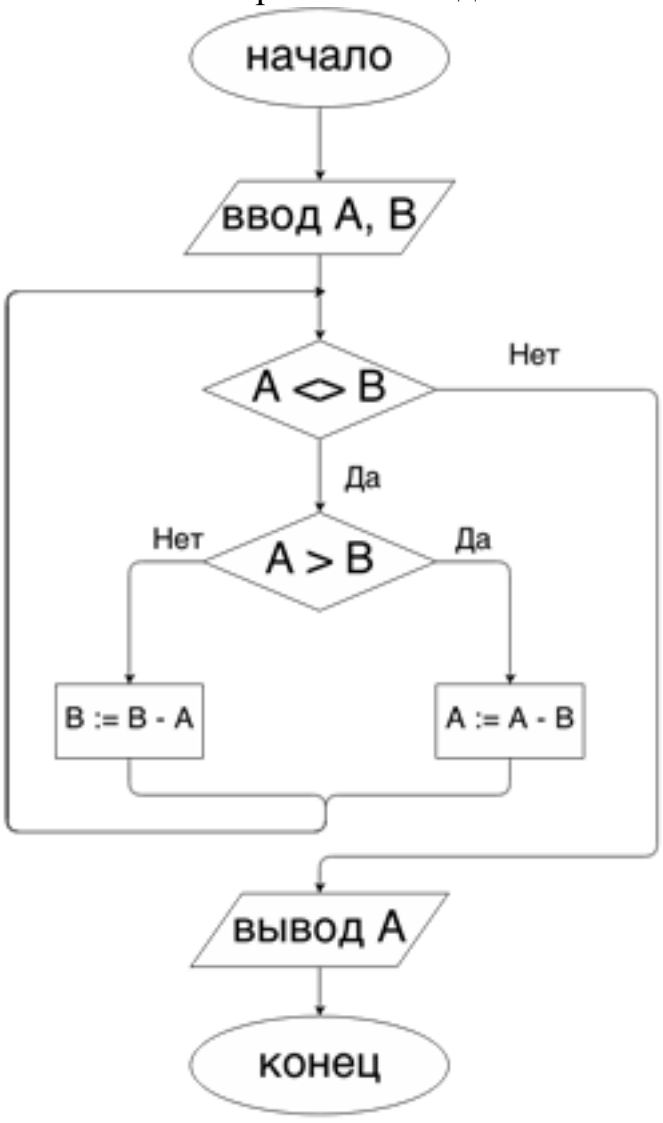
Оценка	***Показатели оценки***
5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Дидактическая единица: 2.3 Формальное определение алгоритма и вычисление функции
Занятие(-я):
1.1.3.Основы Pascal. Операторы, функции, процедуры.

Задание №1
Привести примеры алгоритмов Евклида.
Например:

Алгоритм Евклида «с вычитанием»
Пусть a и b — целые числа, тогда верны следующие утверждения:
Все общие делители пары a и b являются также общими делителями пары a — b, b;
И наоборот, все общие делители пары a — b и b являются также общими делителями пары a и b;
НОД(A, B) = НОД(A — B, B) , если A > B;
НОД(A, 0) = A.

Блок-схема алгоритма Евклида «с вычитанием»



Листинг:
```
var
a, b: integer;
begin
write('a = ');
readln(a);
write('b = ');
```

```
readln(b);
while a <> b do
if a > b then
a := a - b
else
b := b - a;
writeln('NOD = ', a);
end.
```

Оценка	*Показатели оценки*
5	Приведены три примера алгоритма Евклида
4	Приведены два примера алгоритма Евклида
3	Приведен один пример алгоритма Евклида

2.2 Текущий контроль (ТК) № 2

Тема занятия: 2.1.7.Решение задач на составление разветвляющихся алгоритмов
Метод и форма контроля: Письменный опрос (Опрос)
Вид контроля: Проверочная работа
Дидактическая единица: 1.2 методы построения алгоритмов;
Занятие(-я):
1.1.2.Способы описания алгоритмов. Блок-схема
1.1.3.Основы Pascal. Операторы, функции, процедуры.
2.1.1.Линейные алгоритмы
2.1.5.Разветвляющие алгоритмы

Задание №1

Выведите НОД n чисел - Pascal.

Листинг:

```
var n,i,g,k:integer;
m:array[1..1000] of integer; function nod(a,b:longint):longint; begin
if b mod a=0 then nod:=a else nod:=nod(b,a mod b); end;
begin write('n='); read(n); label m1; i:=1;
read(m[i]);
i:=i+1;
if i<n then goto m1; g:=m[1];
label m2; k:=2
g:=nod(g,m[i]); k:=k+1;
if k<n then goto m2; writeln('NOD=',g); readln;
end.
```

Оценка	*Показатели оценки*

5	Описаны входные данные (их типы, диапазон). Описаные выходные данные. Записано математическое соотношение, связывающее результат с исходными данными. Алгоритм решения задачи соответствует математическому понятию. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем».
4	Описаны входные данные (их типы, диапазон). Описаные выходные данные. Записано математическое соотношение, связывающее результат с исходными данными. Блок-схема соответствует выбранному алгоритму решения задачи.
3	Описаны входные данные (их типы, диапазон). Описаные выходные данные. Записано математическое соотношение, связывающее результат с исходными данными.

Дидактическая единица: 2.1 разрабатывать алгоритмы для конкретных задач;
Занятие(-я):
2.1.2.Решение задач на составление линейных алгоритмов
2.1.3.Решение задач на составление линейных алгоритмов
2.1.4.Решение задач на составление линейных алгоритмов
2.1.6.Решение задач на составление разветвляющихся алгоритмов
Задание №1
Разработать алгоритм работы программы в виде блок-схемы и программный код на любом языке программирования для следующей задачи:
Дано натуральное число. Определить четное оно или нечетное.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.

4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

2.3 Текущий контроль (ТК) № 3

Тема занятия: 2.2.8.Решение задач на составление циклических алгоритмов. Вложенные циклы. Определение сложности циклических алгоритмов
Метод и форма контроля: Практическая работа (Опрос)
Вид контроля: Проверочная работа
Дидактическая единица: 1.2 методы построения алгоритмов;
Занятие(-я):
2.1.8.Решение задач на составление разветвляющихся алгоритмов
2.1.9.Решение задач на составление разветвляющихся алгоритмов
2.2.1.Циклические алгоритмы. Цикл с параметром
2.2.2.Цикл с постусловием и с предусловием
2.2.7.Вложенные циклы. Вычисление сложности циклического алгоритма
Задание №1

1. Какие способы описания алгоритмов вы знаете?

1)_____
2)_____
3)_____
4)_____

2. Дайте определение.

Программа это -_____

Образец ответа:

Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.

3. Дайте определение.

Программа это -_____

Оценка	***Показатели оценки***
3	Дано определение термину. Одноиз трех заданий.
4	Дано полное формальнологическое определение термину. Приведены примеры. Два из трех заданий.
5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Все три задания.

Дидактическая единица: 2.1 разрабатывать алгоритмы для конкретных задач;

Занятие(-я):

2.1.7.Решение задач на составление разветвляющихся алгоритмов

2.1.8.Решение задач на составление разветвляющихся алгоритмов

2.1.9.Решение задач на составление разветвляющихся алгоритмов

2.2.3.Решение задач на составление циклических алгоритмов

2.2.4.Решение задач на составление циклических алгоритмов.

2.2.5.Решение задач на составление циклических алгоритмов.

2.2.6.Решение задач на составление циклических алгоритмов.

Задание №1

Разработать алгоритм работы программы в виде блок-схемы и программный код на любом языке программирования для следующей задачи:

Дано натуральное число. Определить произведение всех его цифр.

Оценка	*Показатели оценки*
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Блок-схема соответствует выбранному алгоритму решения задачи.
5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует математическому решению произведения цифр числа. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

2.4 Текущий контроль (ТК) № 4

Тема занятия: 2.3.5.Составление алгоритмов на обработку одномерного массива. Сортировка элементов массива по заданному условию

Метод и форма контроля: Письменный опрос (Опрос)

Вид контроля: Проверочная работа

Дидактическая единица: 1.2 методы построения алгоритмов;

Занятие(-я):

2.3.1.Основные понятия об одномерном массиве. Поиск и замена элементов массива по заданному условию. Удаление и добавление элементов массива по заданному условию

2.3.2.Сортировка элементов одномерного массива по заданному условию

Задание №1

Выполните 5 теоретических заданий.

Каждое правильно выполненное задание оценивается в один балл. Возможен только один правильный ответ в каждом задании. Максимальное количество баллов - 5.

1. В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются.
   For i:=0 to 9 do
   A[i]:=9+I;
   For i:=0 to 4 do
   Begin
   K:=A[i]; A[i]:=A[9-i]; A[9-i]:=A[i]-k;
   End;
   Постройте блок-схему по программе. Чему будут равны элементы этого массива после выполнения фрагмента программы?
   1) 9 10 11 12 13 9 8 7 6 5
   2) 18 17 16 15 14 9 7 5 3 1
   3) 18 17 16 15 14 1 3 5 7 9
   4) 18 17 16 15 14 0 0 0 0 0

2. Определите, какое число будет напечатано в результате выполнения следующего алгоритма
   Var a, b, t, m, r: integer;
   Function F(x: integer): integer;
   begin
   F:=4(x+2)*(x-4);*
   End;
   Begin
   A:=-20; b:=20; m:=a; r:=F(a);
   For t:=a to b d Begin
   If (F(t)<r) then begin m:=t; r:=F(t); end;
   End;
   Write (m);
   End.

3. Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:
   F(1) = 1,
   *F(n) = F(n-1)*n, при n>1.*
   Чему равно значение функции F(6)? В ответ запишите только натуральное число

4. Выберите верное утверждение:

Одномерный массив характеризуется множеством элементов, которые стоят в памяти, имеют одинаковый тип данных.

Одномерный массив характеризуется множеством элементов, которые стоят в памяти, имеют общее имя и одинаковый тип данных.

Одномерный массив характеризуется множеством элементов, которые стоят в памяти рядом, имеют общее имя и одинаковый тип данных.

5. Определите чему равен x после прохождения следующего алгоритма

```
a[5]={1,2,3,4,5};
for i:=1 to 5 do
a[i]=n-i+1;
a[1]:=a[2];
x:=a[1];
```

Оценка	***Показатели оценки***
3	Обучающийся выполнил 2-3 задания
4	Обучающийся выполнил 4 задания
5	Обучающийся выполнил 5 заданий

Дидактическая единица: 2.1 разрабатывать алгоритмы для конкретных задач;
Занятие(-я):
2.2.8.Решение задач на составление циклических алгоритмов. Вложенные циклы. Определение сложности циклических алгоритмов
2.2.9.Решение задач на составление циклических алгоритмов. Вложенные циклы. Определение сложности циклических алгоритмов
2.3.3.Составление алгоритмов на обработку одномерного массива. Поиск и замена элементов массива по заданному условию.
2.3.4.Составление алгоритмов на обработку одномерного массива. Удаление и добавление элементов массива по заданному условию

Задание №1

Разработать алгоритм работы программы в виде блок-схемы и программный код на любом языке программирования для следующей задачи:
Дан одномерный массив из 20 чисел со значениями в диапазоне [-10;40].
Определить количество значений кратных 5.

Оценка	***Показатели оценки***

3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Алгоритм решения задачи соответствует понятиям работы с одномерными массивами.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с одномерными массивами. Блок-схема соответствует выбранному алгоритму решения задачи.
5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с одномерными массивами. Алгоритм решения задачи соответствует математическому определению чисел кратных 5. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

2.5 Текущий контроль (ТК) № 5

Тема занятия: 2.4.7.Составление алгоритмов на обработку двумерного массива. Сортировка элементов массива по заданному условию

Метод и форма контроля: Контрольная работа (Опрос)

Вид контроля: Письменная контрольная работа

Дидактическая единица: 2.1 разрабатывать алгоритмы для конкретных задач;

Занятие(-я):

2.3.5.Составление алгоритмов на обработку одномерного массива. Сортировка элементов массива по заданному условию

2.4.3.Составление алгоритмов на обработку двумерного массива. Поиск и замена элементов массива по заданному условию

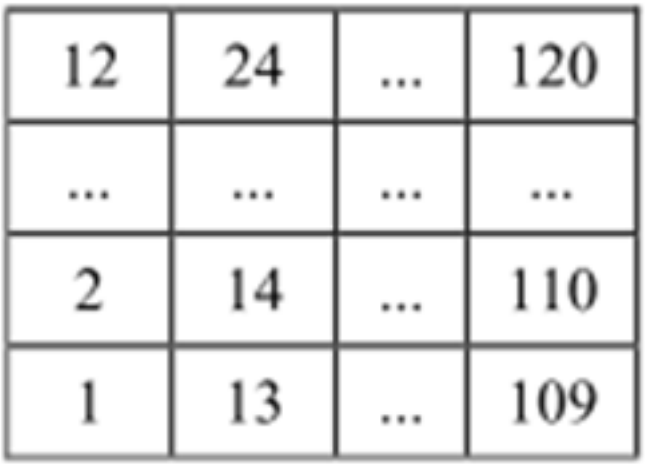
2.4.4.Составление алгоритмов на обработку двумерного массива. Удаление и

добавление элементов массива по заданному условию

2.4.6.Составление алгоритмов на обработку двумерного массива. Сортировка элементов массива по заданному условию

Задание №1

Разработать программный код, который заполнит массив так, как показано на рисунке.



12	24	...	120
...	...	...	...
2	14	...	110
1	13	...	109

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами.

5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами. Алгоритм решения задачи соответствует понятиям работы с памятью.

Задание №2

Разработать программный код для следующей задачи:

В зрительном зале 25 рядов. В каждом ряду 36 мест (кресел). Информация о проданных билетах хранится в двумерном массиве (номер строки – номер ряда, номер столбца – номер места). Если билет продан значение массива = 1, если нет, то значение= 0. Места распределите случайным образом и составьте программу, определяющую число проданных мест в 12-м ряду

После вывода ответа на вторую задачу поменяйте местами второй и предпоследний столбец местами.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами.

5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами. Алгоритм решения задачи соответствует понятиям работы с памятью.

2.6 Текущий контроль (ТК) № 6

Тема занятия: 2.5.4.Составление алгоритмов на обработку текстовых данных
Метод и форма контроля: Контрольная работа (Опрос)
Вид контроля: Письменная контрольная работа
Дидактическая единица: 1.3 методы вычисления сложности работы алгоритмов
Занятие(-я):
1.1.4.Оценка эффективности алгоритма.
2.2.7.Вложенные циклы. Вычисление сложности циклического алгоритма
Задание №1
Описать методы вычисления сложности работы алгоритмов.

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать один из двух методов: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Дидактическая единица: 2.2 определять сложность работы алгоритмов;
Занятие(-я):
2.2.8.Решение задач на составление циклических алгоритмов. Вложенные циклы. Определение сложности циклических алгоритмов

Задание №1

Дан программный код, определить сложность работы алгоритма:

```
min := array[1]
```

```
for i:= 2 to N do
  if array[i] < min then min := array[i];
write(min);
```

Оценка	***Показатели оценки***
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

3. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДИСЦИПЛИНЫ, ИСПОЛЬЗУЕМЫЙ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

№ семестра	Вид промежуточной аттестации
3	Экзамен

Экзамен может быть выставлен автоматически по результатам текущих контролей
Текущий контроль №1
Текущий контроль №2
Текущий контроль №3
Текущий контроль №4
Текущий контроль №5
Текущий контроль №6

Метод и форма контроля: Контрольная работа (Опрос)
Вид контроля: экзаменационный билет содержит одно теоретическое и одно практическое задание
Дидактическая единица для контроля:
1.1 основные модели алгоритмов;
Задание №1
Дать определение "**Дискретность**".

Оценка	*Показатели оценки*

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №2

Дать определение "**Формальность**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №3

Дать определение "**Результативность**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №4

Дать определение "**Конечность**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №5

Дать определение "**Массовость**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №6

Дать определение "**Язык программирования**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №7

Дать определение "**Алгоритм Евклида**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №8

Дать определение "***Алгоритм***".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №9

Дать определение "**Однозначность** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №10

Дать определение "**Алфавит языка** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №11

Дать определение "**Синтаксис** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №12

Дать определение "**Семантика** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Дидактическая единица для контроля:

1.2 методы построения алгоритмов;

Задание №1

Дать определение "**Тип данных** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №2

Дать определение "**Описание данных**".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №3

Дать определение "**Переменная** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №4

Дать определение "**Целый тип** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №5

Дать определение "**Вещественный тип** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №6

Дать определение "**Арифметические выражения** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №7

Дать определение "**Оператор присваивания** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Задание №8

Дать определение "**Логический тип** ".

Оценка	***Показатели оценки***

5	Дано полное формальнологическое определение термину. Пояснено и приведены примеры применения термина. Образец ответа: Детерминированность (от лат. determinans — определяющий) — определяемость. Детерминированность может подразумевать определяемость на общегносеологическом уровне или для конкретного алгоритма. Под жесткой детерминированностью процессов в мире понимается однозначная предопределенность, т. е. у каждого следствия есть строго определенная причина. В таком смысле является антонимом стохастичности. Но детерминированность не всегда тождественна предопределенности. Например, может быть детерминированность будущим (целевая детерминация), когда предполагаемые субъектом цели в его возможном будущем определяют его поведение в настоящем.
4	Дано полное формальнологическое определение термину. Приведены примеры.
3	Дано определение термину.

Дидактическая единица для контроля:

1.3 методы вычисления сложности работы алгоритмов

Задание №1

Описать метод вычисления сложности работы алгоритма:

```
var n: integer; a,b,c: byte;begin randomize; n := random(900
) + 100; writeln(n); a := n div 100; b := n div 10 mod 10; c
 := n mod 10; writeln('?????: ',a+b+c); writeln('??????????
?: ',a*b*c);end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №2

Описать метод вычисления сложности работы алгоритма:

```
var  x1,y1,x2,y2: real; k, b: real; begin write('A(x1;y1): '
); readln(x1, y1); write('B(x2;y2): '); readln(x2, y2);  k :
= (y1 - y2) / (x1 - x2); b := y2 - k * x2;  writeln('y = ',k
:0:2,'x + ',b:0:2);end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №3

Описать метод вычисления сложности работы алгоритма:

```
var n: integer; begin    randomize;     n := random(900)+100
;     if n mod 10 <> 0 then  n := n - n mod 10;  writeln(n);
 end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №4

Описать метод вычисления сложности работы алгоритма:

```
var a, b: integer;begin writeln('??????? ???? ?????? ? ???? ???????? ?????'); readln(a, b); if a mod 2 = 0 then  writeln (b,' - ???????? ?????') else  writeln(a,' - ???????? ?????') ;end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №5

Описать метод вычисления сложности работы алгоритма:

```
var  n: char;begin write('Think of a number of from 1 to 5 a
nd press Enter'); readln;  write('The number more then 3? (y
/n) '); readln(n); if n = 'y' then begin  write('The number
is 4? (y/n) '); readln(n);  if n = 'n' then   writeln('The n
umber is 5.'); end else begin  write('The number is 1? (y/n)
 '); readln(n);  if n = 'n' then begin   write('The number i
s 2? (y/n) '); readln(n);   if n = 'n' then    writeln('The
number is 3.');  end; end; writeln('Guessed!');end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №6

Описать метод вычисления сложности работы алгоритма:

```
var    n: integer; begin    write('n = '); readln(n);     if
 n = 0 then  write('Zero') else begin  if n > 0 then   write
('Positive ')  else    write('Negative ');  if abs(n) < 10 t
hen   write('single')  else    if (abs(n) >= 10) and (abs(n)
 < 100) then    write('two-digit')   else    write('three-
digit or more'); end; writeln;end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №7

Описать метод вычисления сложности работы алгоритма:

```
var    year: word; begin    write('Input year: '); readln(ye
ar);     if year mod 4 = 0 then  if (year mod 100 = 0) and (
year mod 400 <> 0) then   year := 365  else   year := 366 el
se  year := 365;  writeln('The number of days in the year: '
, year);end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Задание №8

Описать метод вычисления сложности работы алгоритма:

```
var    x, y, r: real;    r_xy: real; begin    write('x = ');
 readln(x);    write('y = '); readln(y);    write('R = '); r
eadln(r);     r_xy := sqrt(x*x + y*y);     if r_xy <= r then
  writeln('The point belongs to the circle.') else  writeln(
'The point does not belong to the circle.');end.
```

Оценка	***Показатели оценки***

5	Обучающийся должен: I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать каждый метод: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик) 2) Сложность алгоритма по памяти: • память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных; • каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки; • количество памяти достаточно для выполнения любого алгоритма; • процессор выполняет любую элементарную операцию за один временной шаг; • циклы и функции не считаются элементарными операциями.

4	Обучающийся должен I. Указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов. II. Описать методом: 1) Вычислительная сложность - *подсчет количества выполняемых операций*. Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. (привести примеры расчета, достаточно - счетчик)
3	Обучающийся должен указать в виде оценивающего ресурса процессорное время (*вычислительная сложность*) и память (*сложность алгоритма по памяти*). Оценка позволяет предсказать время выполнения и сравнивать эффективность алгоритмов.

Дидактическая единица для контроля:

2.1 разрабатывать алгоритмы для конкретных задач;

Задание №1 (из текущего контроля)

Разработать программный код для следующей задачи:

В зрительном зале 25 рядов. В каждом ряду 36 мест (кресел). Информация о проданных билетах хранится в двумерном массиве (номер строки – номер ряда, номер столбца – номер места). Если билет продан значение массива = 1, если нет, то значение= 0. Места распределите случайным образом и составьте программу, определяющую число проданных мест в 12-м ряду

После вывода ответа на вторую задачу поменяйте местами второй и предпоследний столбец местами.

Оценка	***Показатели оценки***

3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами.
5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами. Алгоритм решения задачи соответствует понятиям работы с памятью.

Задание №2

Разработать программный код для следующей задачи:

1. Создать одномерный массив из 20 чисел со значениями в диапазоне [-7;23]. В созданном массиве найти максимальное значение и вывести на экран.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами.

5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными. Алгоритм решения задачи соответствует понятиям работы с двумерными массивами. Алгоритм решения задачи соответствует понятиям работы с памятью.

Задание №3

Разработать программный код для следующей задачи:

1. Дан двумерный массив размером 5x6, значения вводятся с клавиатуры. В первом его столбце найти:

а) сумму нечетных элементов;

б) количество положительных элементов

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.

5	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №4

Разработать программный код для следующей задачи:

1. Создать одномерный массив из 30 чисел со значениями в диапазоне [-20;40). В созданном массиве найти минимальное значение и вывести на экран.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №5

Разработать программный код для следующей задачи:

1. Создать одномерный массив из 15 чисел со значениями в диапазоне [-1;20).

В созданном массиве найти индекс числа с минимальным значение и вывести на экран.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №6

Разработать программный код для следующей задачи:

1. В поезде 18 вагонов, в каждом из которых 36 мест.

Информация о проданных на поезд билетах хранится в двумерном массиве, номера строк которых соответствуют номерам вагонов, а номера столбцов — номерам мест.

Если билет на то или иное место продан, то соответствующий элемент массива имеет значение 1,

в противном случае — 0. Составить программу, определяющую число свободных

мест в любом из вагонов поезда.

Оценка	*Показатели оценки*
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №7

Разработать программный код для следующей задачи:

1. В двумерном массиве хранится информация о зарплате 20 человек за каждый месяц года

(первого человека — в первой строке, второго — во второй и т. д.). Минимальная заработная плата 12 тыс. рублей., максимум 80тыс.

Составить программу для расчета общей зарплаты, полученной за год любым человеком, информация о зарплате которого представлена в массиве.

Оценка	*Показатели оценки*

3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №8

Разработать программный код для следующей задачи:

1. Дан двумерный массив размером 10х15. Значения вводятся с клавиатуры. Определить:

а) сумму всех элементов массива;

б) сумму квадратов всех элементов массива;

в) среднее арифметическое всех элементов массива.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.

4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.
5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Задание №9

Разработать программный код для следующей задачи:

1. Дан квадратный двумерный массив, размер которого определяется пользователем, а значения случайны в диапазоне [-10;10) .

Определить:

а) сумму квадратов элементов четвертого столбца массива;

б) сумму квадратов элементов *k*-й строки массива.

Оценка	***Показатели оценки***
3	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными.
4	Необходимо выполнить следующие операции и приемы: Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Блок-схема соответствует выбранному алгоритму решения задачи.

5	Описание входных данных (их типов, диапазонов); Описание выходных данных; Запись математического соотношения, связывающего результат с исходными данными; Алгоритм решения задачи соответствует математическому понятию четных и нечетных чисел. Блок-схема построена в соответствии с ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»; Блок-схема соответствует выбранному алгоритму решения задачи.

Дидактическая единица для контроля:

2.2 определять сложность работы алгоритмов;

Задание №1 (из текущего контроля)

Дан программный код, определить сложность работы алгоритма:

```
min := array[1]
for i:= 2 to N do
  if array[i] < min then min := array[i];
write(min);
```

Оценка	***Показатели оценки***
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №2

Дан программный код, определить сложность работы алгоритма:

```
var     n: longint;    sum: integer;begin    readln(n);
  sum := 0;    while n > 0 do begin        if n mod 10 mod 2
```

```
 = 0 then            sum := sum + n mod 10;        n := n di
v 10;    end;        writeln(sum);end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №3

Дан программный код, определить сложность работы алгоритма:

```
const     N = 20;var     a: array[1..N] of integer;    i: by
te;begin    randomize;    for i:=1 to N do begin        a[i]
 := random(100);        write(a[i]:4);    end;    writeln;
  writeln('???????? ?????? ???????????: ');    for i:=1 to N
-1 do        if a[i] < a[i+1] then            write(a[i+1]:
4);    writeln;end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №4

Дан программный код, определить сложность работы алгоритма:

```
const    M = 10;var    C: array[1..M] of word;    maxC: word
;    i: byte; begin    randomize;    maxC := 0;    write('??
?????? ??????: ');    for i := 1 to M do begin        C[i] :
= random(1000);        write (C[i]:4);        if maxC < C[i]
 then             maxC := C[i];    end;    writeln;    write
ln('????????: ', maxC);    write('???????: ');    for i := 1
 to M do begin        write (C[i]/maxC:6:2);    end;    writ
eln;end.
```

Оценка	***Показатели оценки***
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №5

Дан программный код, определить сложность работы алгоритма:

```
const N = 20;var     arr: array[1..N] of integer;    i: byte
;begin    randomize;    for i:=1 to N do begin        arr[i]
 := random(100) - 75;        write(arr[i]:4);    end;    wri
teln;    for i:=1 to N do        if arr[i] > 0 then begin
```

```
        writeln(i, ' ', arr[i]);            break;        e
nd;end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №6

Дан программный код, определить сложность работы алгоритма:

```
const N = 10;var    arr: array[1..N,1..N] of integer;    max
: integer;    i,j: byte;    begin    randomize;    for i:=1
to N do begin        for j:=1 to N do begin            arr[i
,j] := random(1000);            write(' |',arr[i,j]:3,'| ');
        end;        writeln;    end;    for i:=1 to N do
     write(' ----- ');    writeln;    for j:=1 to N do begin
        max := arr[1,j];        for i:=2 to N do
if arr[i,j] > max then                max := arr[i,j];
  write('  ',max:3,'  ');    end;    writeln;end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.

4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №7

Дан программный код, определить сложность работы алгоритма:

```
var    str1,str2,str3: string;    l,i: byte;    begin    rea
dln(str1);    str2 := '';    str3 := '';    l := length(str1
);    for i:=1 to l do        if odd(i) then            str3
 := str3 + str1[i]        else            str2 := str2 + str
1[i];    writeln(str2);    writeln(str3);end.
```

Оценка	***Показатели оценки***
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №8

Дан программный код, определить сложность работы алгоритма:

```
const N = 10;var    arr: array[1..N] of integer;    i: byte;
begin    randomize;    for i:=1 to N do begin        arr[i]
```

```
:= random(100) - 50;        write(arr[i]:4);    end;    writ
eln;    for i:=1 to N do         arr[i] := -1 * arr[i];    f
or i:=1 to N do         write(arr[i]:4);    writeln;end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.
4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Задание №9

Дан программный код, определить сложность работы алгоритма:

```
const     N = 7; M = 5;var    arr: array[1..N,1..M] of byte;
    i,j,a,b,buff: byte;begin    randomize;    for i:=1 to N
do begin        for j:=1 to M do begin            arr[i,j] :
= random(20);            write(arr[i,j]:3);        end;
   writeln;    end;    write('????? ?????? ????????: ');
readln(a,b);    for j:=1 to M do begin        buff := arr[a,
j];        arr[a,j] := arr[b,j];        arr[b,j] := buff;
 end;    for i:=1 to N do begin        for j:=1 to M do
       write(arr[i,j]:3);        writeln;    end;end.
```

Оценка	*Показатели оценки*
3	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* с использованием таблицы определения сложности алгоритма.

4	Рассчитана сложность работы алгоритма одним из методов (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.
5	Рассчитана сложность работы алгоритма двумя методами (*вычислительная сложность* или *сложность алгоритма по памяти)* без использования таблицы определения сложности алгоритма.

Дидактическая единица для контроля:

2.3 Формальное определение алгоритма и вычисление функции

Задание №1

Привести примеры алгоритмов Евклида.

Оценка	***Показатели оценки***
5	Привести три примера алгоритма Евклида
4	Привести два примера алгоритма Евклида
3	Привести один пример алгоритма Евклида

Задание №2

Почему необходимо формальное определение алгоритма.

Оценка	***Показатели оценки***
5	Привести три примера.
4	Привести два примера.
3	Привести один пример.