Согласовано
Начальник отдела эксплуатации и
внедрения информационных систем
ОГАУЗ СОМИАЦ
_____ Я.А.Комиссаров
«31» 08 2020 г.

УТВЕРЖДАЮ
Заместитель директора по
учебной работе
_____ И. В. Иванешко
«31» 08 2020 г.

Контрольно-оценочные материалы для промежуточной аттестации по общепрофессиональной дисциплине
ОП.08. Теория алгоритмов
Для специальности 09.02.03 Программирование в компьютерных системах

Экзамен является промежуточной формой контроля, подводит итог освоения дисциплины ОП.08. Теория алгоритмов.

В результате освоения дисциплины студент должен освоить следующие профессиональные компетенции:

ПК 1.1. Выполнять разработку спецификаций отдельных компонент

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля

А также общие 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 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.

В ходе экзамена проверяется сформированность:

Знаний:

З1 основные модели алгоритмов;
З2 методы построения алгоритмов;
З3 методы вычисления сложности работы алгоритмов.

Умений:

У1 разрабатывать алгоритмы для конкретных задач;
У2 определять сложность работы алгоритмов.

Экзамен проводится в форме тестирования. Тест содержит 30 вопросов (суммарно тестовых позиций и теоретических вопросов с ответом) из 120, выбираемых случайным образом программой из блоков заданий по ПК.1.1 - 15 вопросов (10 вопросов с выборов ответа, 5 вопросов с кратким ответом), ПК.2.2 – 15 вопросов (10 вопросов с выборов ответа, 5 вопросов с кратким ответом). Время тестирования – 50 минут (по 1 минуте на каждый вопрос с выбором ответа, по 3 минуты на вопрос с ответом).

Критерии оценивания

«5» - соответствует работа, содержащая 100-90% правильных ответов;
«4» - соответствует работа, содержащая 75-89% правильных ответов;
«3» - соответствует работа, содержащая 60-74% правильных ответов;
«2» - соответствует работа, содержащая менее 60% правильных ответов.

Шкала оценивания образовательных результатов:

Оценка	Критерии
«отлично»	Студент набрал 5 баллов (по весу критерия)
«хорошо»	Студент набрал 4 балла (по весу критерия)
«удовлетворительно»	Студент набрал 3 балла (по весу критерия)
«неудовлетворительно»	Студент набрал 0-2 балла (по весу критерия)

Первый блок заданий – вопросы с выбором ответа:

Проверяемая компетенция - ПК 1.1.
Проверяемые общие компетенции - ОК1-ОК9

1. Какой из приведенных способов не будет являться способом записи алгоритма?
    a) Словесный;
    b) Графический;
    c) Текст программы;
    d) Логический

2 . Цикл какого типа должен выполниться хотя бы один раз?
    a) Цикл с предусловием;
    b) Цикл с постусловием;
    c) Цикл со счетчиком;
    d) Все циклические конструкции;

3. Алгоритмы бывают:
    a) Целые;
    b) Вещественные;
    c) Циклические;
    d) Логические;

4. Укажите алгоритмическую конструкцию или её разновидность, которая просматривается в указанном стихотворении:

«Кабы я была царица, -
Говорит одна девица, -
То на весь крещённый мир
Приготовила б я пир».

    a) Цикл с предусловием;
    b) Цикл с постусловием;
    c) Бесконечный цикл;
    d) Ветвление;

5. Цикл с предусловием:
    a) Должен выполниться хотя бы один раз;
    b) Не выполняется ни разу;
    c) Может не выполниться ни разу;
    d) Выполняется заданное количество раз;

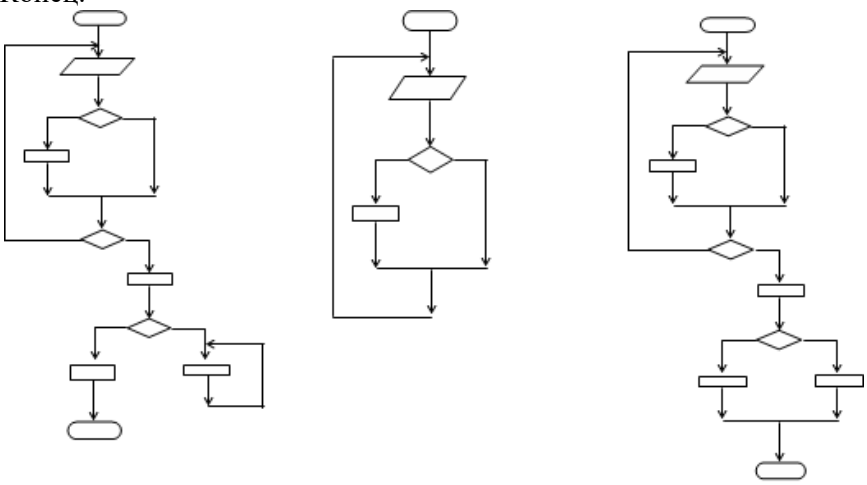
6. Укажите алгоритмическую конструкцию или её разновидность:
«12 поросят на палубе сидят.
Они песенки поют, им уроки задают.
Один из них устал и с палубы удрал,
И вот результат – 11 поросят
11 поросят на палубе сидят…»
И так далее…
    a) Цикл со счетчиком;
    b) Ветвление;
    c) Последовательность;

7. Какую алгоритмическую конструкцию можно соотнести со следующим литературным фрагментом?

Если ты попал в больницу и не хочешь там лечиться,
Жди, когда к тебе в палату самый главный врач придет.
Если ты его укусишь - кончится твое леченье –
В тот же вечер из больницы заберут тебя домой.

a) Последовательность;
b) Цикл со счетчиком;
c) Ветвление;
d) Цикл с предусловием;

8. Какая блок-схема соответствует следующему алгоритму? «поиск Мориарти»
Взять из памяти приметы очередного преступника. Если он худ, и лыс, и старше 50 лет, то: занести его в список кандидатов.
Если остались ещё преступники в памяти, то вернуться к шагу 1.
Показать Холмсу кандидата с наибольшим числом очков.
Если это Мориарти, то: поймать его и остановить поиск
иначе: вычеркнуть его из списка кандидатов и вернуться к шагу 4.
Конец.



a) Первая;
b) Вторая;
c) Третья;

9. Цикл со счетчиком:
a) Выполняется заданное количество раз;
b) Выполняется, пока истинное условие;
c) Не является универсальным;
d) Является универсальным;

10. Циклы бывают:
a) С подусловием;
b) С предусловием;
c) С заданным количеством повторений;
d) С постепенным условием;

12. Цикл с постусловием:
a) Должен выполниться хотя бы один раз;
b) Не выполнится ни разу;
c) Выполнится один раз;
d) Выполнится заданное количество раз;

13. Приведенная программа Тьюринга описывает следующий алгоритм:

	a_0	0	1	2	...	7
q_1	1 Н q_0	1 Н q_0	2 Н q_0	3 Н q_0	...	0 Л q_1

a) Увеличение 8-ричного числа на 1
b) Увеличение 10-чного числа на 1

c) Уменьшение 8-ричного числа на 1
d) Уменьшение 10-чного числа на 1

14. Сортировка - это:
a) Упорядочение данных
b) Выбор только положительных элементов
c) Нумерация элементов массива с 1

15. Данный псевдокод программы:

```
For I = 0 To 3 do
  For J = 0 To 4 - i do
    If M[J]> M[J + 1] Then
     Begin
     Tmp: = M[J];
      M[J]: = M[J + 1];
      M[J + 1]: = Tmp;
    End;
```

a) Сортирует массив по убыванию методом прямого выбора
b) Сортирует массив по возрастанию методом пузырька
c) Меняет порядок элементов массива на обратный
d) Сортирует массив по убыванию методом пузырька

16. К методу грубой силы можно отнести разработку следующих алгоритмов:
a) Сортировка методом прямого выбора
b) Быстрая сортировка
c) Последовательный поиск
d) Сортировка слиянием
e) Пузырьковая сортировка
f) Алгоритм Евклида

17. В чем достоинства метода грубой силы при разработке алгоритмов:
a) низкая эффективность
b) редко дает "красивые" алгоритмы
c) широкая применимость и простота
d) низкая стоимость разработки

18. К методу декомпозиции можно отнести следующие алгоритмы:
a) Сортировка методом прямого выбора
b) Быстрая сортировка
c) Последовательный поиск
d) Сортировка слиянием
e) Пузырьковая сортировка

19. В чем недостаток метода сортировки слиянием:
a) имеет низкое быстродействие
b) нерациональный алгоритм
c) часто требует дополнительные объемы памяти

20. Метод преобразования задачи может быть основан на:
a) Упрощении экземпляра задачи
b) Разбиении задачи на некоторые подзадачи
c) Решении "в лоб"
d) Преобразовании к другой задаче

21. К методу преобразования задачи можно отнести алгоритмы:
a) Вычисление определителя
b) Вычисление обратной матрицы

c) Задача о поиске фальшивой монеты
d) Умножение по-русски
e) Вычисление полинома методом Горнера

22. Общая эффективность алгоритма будет зависеть от:

a) Количества выполненных элементарных операций
b) Количества памяти для хранения промежуточных ресурсов
c) Времени, затраченного разработчиком на построение алгоритма

23. Алгоритм умножения матриц будет относиться:

a) К количественно-зависимым
b) К параметрически зависимым
c) К количественно-параметрически зависимым

24. Алгоритм поиска максимума в массиве будет относиться:

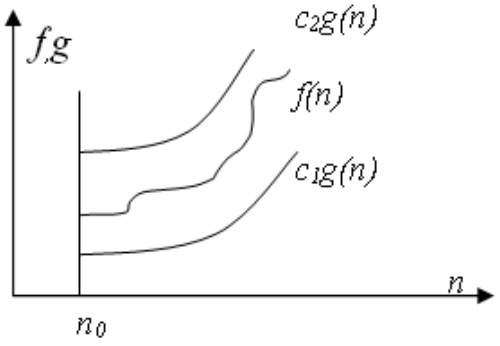
- К количественно-зависимым
- К параметрически зависимым
- К количественно-параметрически зависимым

25. Данная функция трудоемкости отражает:

$$F_a^{\vee}(N) = \min_{D \in D_N} \{F_a(D)\}$$

- лучший случай
- худший случай
- средний случай

26. На рисунке изображена следующая асимптотическая оценка функции трудоемкости



a) Оценка Θ (тетта)
b) Оценка Ω (Омега)
c) Оценка O (О большое)

27. К элементарным операциям при оценке алгоритма относятся:

a) Простое присваивание
b) Арифметические операции: (*, /, -, +);
c) Конструкция «Цикл»
d) Логическое ИЛИ
e) Конструкция «Ветвление»

28. Сложность алгоритма рекурсивной сортировки слиянием является:

a) квазилинейной n* log n
b) квадратичной n2
c) экспоненциальной 2n
d) линейной n

29. Переход от функции трудоемкости к времени выполнения программы для алгоритма с известной трудоемкостью зависит от:

a) времени выполнения элементарных операций в среде языка реализации алгоритма
b) количества элементарных операций в алгоритме

c) специфики реализации алгоритма (квалификация программиста, выбор структуры данных) на данном языке программирования
d) количества входящих параметров в функцию

30. В данной формуле перехода от функции трудоемкости к временным оценкам ta(N) = t ср * fa(N) t ср является

a) средним временем выполнения алгоритма
b) средним временем выполнения обобщенной элементарной операции
c) средним временем выполнения обобщенной элементарной операции для данного языка и процессора
d) средним временем выполнения алгоритма для данного языка и процессора

Проверяемая компетенция - ПК 1.2.
Проверяемые общие компетенции - ОК1-ОК9

1. Что выполняет данная программа?

```
const
SIZE=10;
var
a:array[1..SIZE] of integer;
max:integer;
j:integer;
buf:integer;
i,k:integer;

begin
for i:=l to SIZE do
readln(a[i]);
for i:=l to SIZE-1 do
 begin
 max:=i;
 for j:=i+l to SIZE do
    if a[j] > a [max]   then max:=j;
  buf:=a[i];
  a[i]:=a[max];
  a[max]:=buf;
  end;
for k:=l to SIZE do
writeln a[i];
end.
```

a) сортирует массив по возрастанию методом прямого выбора
b) сортирует массив по убыванию методом прямого выбора
c) сортирует массив по возрастанию методом пузырька
d) сортирует массив по убыванию методом пузырька
e) поиск наибольшего элемента массива

2. Псевдокод данного алгоритма представляет собой:

```
ЦИКЛ ДЛЯ J=1 ДО N-1 ШАГ 1
НЦ
  F=0
  ЦИКЛ ДЛЯ I=1 ДО N-J ШАГ 1
  НЦ
   ЕСЛИ A[I] > A[I+1] ТО
        ОБМЕН A[I],A[I+1]
        F=1
   КЦ
   ЕСЛИ F=0 ТО ВЫХОД ИЗ ЦИКЛА
  КЦ
```

a) реализацию сортировки по возрастанию методом прямого выбора
b) реализацию сортировки по убыванию методом прямого выбора
c) реализацию сортировки по возрастанию усовершенствованным методом пузырька
d) реализацию сортировки по убыванию усовершенствованным методом пузырька
e) реализацию сортировки по возрастанию простым методом пузырька
f) реализацию сортировки по убыванию простым методом пузырька

3. Данный псевдокод программы:

```
For I = 0 To 3 do
 For J = 0 To 4 - i do
  If M[J]> M[J + 1] Then
   Begin
   Tmp: = M[J];
    M[J]: = M[J + 1];
    M[J + 1]: = Tmp;
  End;
```

a) Сортирует массив по убыванию методом прямого выбора
b) Сортирует массив по возрастанию методом пузырька
c) Меняет порядок элементов массива на обратный
d) Сортирует массив по убыванию методом пузырька

4. Что выполнит данный псевдокод программы?

```
АЛГОРИТМ Merge(B[0..p - 1], C[0..q - 1], A[0..p + q - 1])
  // Слияние двух отсортированных массивов в один
  // Входные данные: Сортированные массивы B[0..p - 1]
  //                 и C[0..q - 1]
  // Выходные данные: Сортированный массив A[0..p + q - 1],
  //                  состоящий из элементов B и C
  i ← 0; j ← 0; k ← 0
  while i < p and j < q do
    if B[i] ≤ C[j]
      A[k] ← B[i]; i ← i + 1
    else
      A[k] ← C[j]; j ← j + 1
    k ← k + 1
  if i = p
    Копировать C[j..q - 1] в A[k..p + q - 1]
  else
    Копировать B[i..p - 1] в A[k..p + q - 1]
```

a) Он выполнит сортировку массива по возрастанию
b) Он выполнит сортировку массива по убыванию
c) Он выполнит слияние двух массивов в один, сортируя по убыванию
d) Он выполнит слияние двух массивов в один, сортируя по возрастанию

5. В чем недостаток метода быстрой сортировки?
a) имеет низкое быстродействие
b) рекурсивность алгоритма
c) часто требует дополнительные объемы памяти

6. Что выполнит псевдокод подпрограммы:

```
For I = 0 To 3 do
  For J = 0 To 4 - i do
    If M[J]> M[J + 1] Then
     Begin
     Tmp: = M[J];
      M[J]: = M[J + 1];
      M[J + 1]: = Tmp;
    End;
```

a) Сортирует массив по убыванию методом прямого выбора
b) Сортирует массив по возрастанию методом пузырька
c) Меняет порядок элементов массива на обратный
d) Сортирует массив по убыванию методом пузырька

7. Дан следующий код быстрой сортировки:

```
function Part(l, r: integer):integer;
var
  v, i, j, b: integer;
begin
  V:=a[r];
  I:=l-1;
  j:=r;
  repeat
    repeat
      dec(j)
    until (a[j]<=v) or (j=i+ 1);
    repeat
      inc(i)
    until (a[i]>=v) or (i=j- 1);
    b:=a[i];
    a[i]:=a[j];
    a[j]:=b;
  until i>=j;
  a[j]:=a[i];
  a[i]:= a[r];
  a[r]:=b;
  part:=i;
end;
```

В нем в качестве опорного элемента выбран:

a) Первый элемент массива
b) Последний элемент массива
c) Средний элемент массива
d) Любой элемент массива

8. Дан фрагмент программы. Какой алгоритм он реализует?

```
Begin
   t := 0;
  For i := 1 to c[0] do
  Begin
    t := t + a[i] + b[i];
    c[i] := t mod base;
    t := t div base;
  End;
  If t <> 0 then
  Begin
    inc(c[0]);
    c[c[0]] := t;
  End;
 End;
```

a) Складывает 2 длинных целых числа
b) Умножает 2 длинных целых числа
c) Умножает длинное целое число на обычное

9. При выполнении данного программного кода буден выполнено:

```
procedure Inser;
var
  i, l: integer;
  x: DataItem;
begin
  for i := 2 to count do
  begin
    x := item[i];
    j := i-1;
    while (x<item[j]) and (j>0) do
    begin
      item[j+1] := item[j];
      j := j-1;
    end;
    item[j+1] := x;

  end;
end;
```

a) выполнена сортировка массива по возрастанию
b) выполнена сортировка массива по убыванию
c) выполнен поиск в массиве в ширину

10. Данный фрагмент программы реализует:

```
while M[i] < M[i-1] do
    begin
       swap(M[i],M[i-1]);
       i := i-1
    end
```

a) выполняет перемещение элементов массива, меняя с опорным для сортировки по возрастанию
b) сортирует массив по возрастанию
c) Сортирует массив по убыванию

11. Выберите достоинства сортировки методом вставки:

a) эффективен на наборах данных, которые уже частично отсортированы
b) низкая сложность
c) эффективен на неотсортированных массивах
d) эффективен на небольших наборах данных

12. Что выполняет следующий фрагмент программы?

```
for i:=1 to n do
 begin
   for j:=i+1 to n do
     begin
       a[j,i]:=-a[j,i]/a[i,i];
       for k:=i+1 to n do
           a[j,k]:=a[j,k]+a[j,i]*a[i,k];
       b[j]:=b[j]+a[j,i]*b[i];
     end;
```

a) Решает систему линейных уравнений методом Гаусса
b) Приводит матрицу к треугольному виду
c) Преобразует матрицу коэффициентов к расширенной
d) Умножает 2 матрицы

13. Для приведенного алгоритма асимптотическая сложность будет выражаться следующей формулой:

```
SumM(A,N;Sum);
Sum:=0;
for i:=1 to N
for j:=1 to N
Sum:=Sum+A[i,j];
end for j
end for i
Return (Sum);
End;
```

a) N^2
b) N
c) N*log(n)

14. Для приведенного алгоритма трудоемкость при n=2 , будет равна

```
SumM(A,N;Sum);
Sum:=0;
for i:=1 to N
for j:=1 to N
Sum:=Sum+A[i,j];
end for j
end for i
Return (Sum);
End;
```

a) 38
b) 13
c) 4

15. Конструкция простого цикла потребует выполнения
a) 1 + 3*N+N*fтц элементарных операций
b) 3 элементарных операции
c) 1 элементарная операция
d) 3N элементарных операций
e) 1 + 3*N элементарных операций

16. Для алгоритма вычисления среднего арифметического элементов одномерного массива размера n функция трудоемкости будет иметь вид:
a) 4+6n
b) 2+6n
c) 2+3n
d) 1+3n+n

17. Сложность приведенного алгоритма является:

```
Sum:=0;
for i:=1 to N
for j:=1 to N
Sum:=Sum+A[i,j];
```

a) Линейной
b) Квадратичной
c) Логарифмической
d) кубической

18. Сложность приведенного алгоритма является:

```
Sum:=0;
for i:=1 to N
If A[i]>0
then
Sum:=Sum+A[i];
```

a) Линейной
b) Квадратичной
c) Логарифмической
d) кубической

19. Сложность приведенного алгоритма является:

```
Sum:=0;
for i:=1 to N
Sum:=Sum+A[i];
```

a) Линейной
b) Квадратичной
c) Логарифмической
d) кубической

20. Сложность приведенного алгоритма является:

```
Sum:=0;
for i:=1 to N
for j:=1 to N
for k:=1 to N
Sum:=Sum+A[i,j,k];
```

a) Линейной
b) Квадратичной
c) Кубической
d) экспоненциальной

21. Сложность приведенного алгоритма является:

```
Sum = 0
For i = 1 to n
If A[i]<0
```

```
Then  Sum = Sum + A[i]
```

a) Линейной
b) Квадратичной
c) Кубической
d) экспоненциальной

22. Дан алгоритм. Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

a) Все элементы отрицательные
b) Все элементы положительные
c) Нет разницы, какие элементы
d) Отрицательных и положительных поровну

23. Дан алгоритм. Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

a) Все элементы отрицательные
b) Все элементы положительные
c) Нет разницы, какие элементы
d) Отрицательных и положительных поровну

24. Дан алгоритм. Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

```
Sum = 0
For i = 1 to n
Sum = Sum + A[i]
```

a) Все элементы отрицательные
b) Все элементы положительные
c) Нет разницы, какие элементы
d) Отрицательных и положительных поровну

25. Дан алгоритм. Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

```
max = A[1]

For i = 2 to n

 If A[i]>max
```

Then max = A[i]

a) Максимальный элемент стоит первым
b) Элементы расположены по возрастанию
c) Нет разницы, как расположены элементы

26. Дан алгоритм. Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

```
max = A[1]

For i = 2 to n

 If A[i]>max

 Then  max = A[i]
```

a) Максимальный элемент стоит первым
b) Элементы расположены по возрастанию
c) Нет разницы, как расположены элементы

27. Дан алгоритм. Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

a) Все элементы четные
b) Все элементы нечетные
c) Нет разницы, какие элементы в массиве

28. Дан алгоритм. Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

a) Все элементы четные
b) Все элементы нечетные
c) Нет разницы, какие элементы в массиве

29. Дан алгоритм. Какие исходные данные дадут средний случай функции трудоемкости алгоритма?

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

a) Все элементы четные
b) Все элементы нечетные
c) Нет разницы, какие элементы в массиве
d) Поровну четных и нечетных элементов

30. Дан алгоритм. Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

```
kol = 0
For i = 1 to n
kol = kol + A[i]
```

a) Все элементы отрицательные
b) Все элементы положительные
c) Нет разницы, какие элементы
d) Отрицательных и положительных поровну

Второй блок заданий – вопросы с требуемым ответом

Проверяемая компетенция - ПК 1.1.

Проверяемые общие компетенции - ОК1-ОК9

1. Дайте определение алгоритма.

2. Объясните суть свойства «результативность».

3. Какие типы данных относятся к базовым**?**

.

4. Перечислите способы записи алгоритмов

5. Можно ли данную последовательность действий считать алгоритмом? Почему?
Достать ключ. Вставить его в замочную скважину. Повернуть ключ 2 раза против часовой стрелки. Вынуть ключ. Открыть дверь.

6. Перечислите основные свойства алгоритма

7. Перечислите основные виды алгоритмов

8. Охарактеризуйте разницу между циклом с предусловием и цикла с постусловием

9. Приведен алгоритм Тьюринга для алгоритма умножения на 2. Введите недостающую команду:

	a_0	0	1	2	3	4	5	6	7	8	9
q_1	a_0 Л q_2	0 П q_1	1 П q_1	2 П q_1	3 П q_1	4 П q_1	5 П q_1	6 П q_1	7 П q_1	8 П q_1	9 П q_1
q_2	a_0 Н q_0	0 Л q_2	2 Л q_2	4 Л q_2	6 Л q_2	8 Л q_2	0 Л q_3	2 Л q_3		6 Л q_3	8 Л q_3
q_3	1 Н q_0	1 Л q_2	3 Л q_2	5 Л q_2	7 Л q_2	9 Л q_2	1 Л q_3	3 Л q_3	5 Л q_3	7 Л q_3	9 Л q_3

10. Дан алгоритм Тьюринга для увеличения восьмеричного числа на 2. Введите недостающую команду

	A0	0	1	2	3	4	5	6	7
Q1		2 Н q0	3 Н q0	4 Н q0	5 Н q0	6 Н q0	7 Н q0	0 Л q2	
Q2	1 Н q0	1 Н q0	2 Н q0	3 Н q0	4 Н q0	5 Н q0	6 Н q0	7 Н q0	0 Л q2

11. Дан алгоритм Тьюринга для увеличения восьмеричного числа на 4. Введите недостающую команду:

	A0	0	1	2	3	4	5	6	7
Q1		4 Н q0	5 Н q0	6 Н q0	7 Н q0	0 Л q2	1 Л q2	2 Л q2	3 Л q2
Q2	1 Н q0	1 Н q0	2 Н q0	3 Н q0	4 Н q0	5 Н q0	6 Н q0	7 Н q0	

12. В чем заключается метод грубой силы при построении алгоритмов?
13. Дайте определение рекурсивной функции.
14. Что такое подпрограмма?
15. Напишите преимущества разделения программы на подпрограммы (модули).
16. Какие категории простых типов данных в языках программировании высокого уровня вы можете назвать?
17. Дайте определение трудоемкости алгоритма.
18. Дайте определение худшего случая алгоритма.
19. Дайте определение лучшего случая алгоритма.
20. Приведите элементарные операции в языке записи алгоритмов:
21. Является ли операция взятия индекса [] элементарной операцией при расчете функции трудоемкости?
22. Приведите расчет трудоемкости для конструкции «Цикл».
23. В чем заключается метод декомпозиции при построении алгоритмов?
24. В чем заключается метод преобразования при построении алгоритмов?
25. В чем заключается метод уменьшения размера задачи при построении алгоритмов?
26. Приведите классификацию алгоритмов по виду функции трудоемкости.
27. Кратко объясните алгоритм сортировки массива методом пузырька.
28. Какой алгоритм записан на данной программе Тьюринга?

	a_0	0	1	2	3	4	...	7	8	9
q_1	1 Н q_0	1 Н q_0	2 Н q_0	3 Н q_0	4 Н q_0	5 Н q_0	...	8 Н q_0	9 Н q_0	0 Л q_1

29. Какой алгоритм записан на данной программе Тьюринга?

	a_0	0	1	2	...	7
q_1	1 Н q_0	1 Н q_0	2 Н q_0	3 Н q_0	...	0 Л q_1

30. Какой алгоритм записан на данной программе Тьюринга?

	a_0	0	1	2	...	8	9
q_1		9 Л q_1	0 Н q_0	1 Н q_0	...	7 Н q_0	8 Н q_0

Проверяемая компетенция - ПК 1.2.
Проверяемые общие компетенции - ОК1-ОК9

1. Что выполнит данный фрагмент кода?

```
begin
for i:=l to SIZE do
readln(a[i]);
for i:=l to SIZE-1 do
 begin
 max:=i;
 for j:=i+l to SIZE do
    if a[j] > a [max]   then max:=j;
  buf:=a[i];
  a[i]:=a[max];
  a[max]:=buf;
  end;
```

2. Каким методом выполнит сортировку данный алгоритм?

```
ЦИКЛ ДЛЯ J=1 ДО N-1 ШАГ 1
НЦ
  F=0
  ЦИКЛ ДЛЯ I=1 ДО N-J ШАГ 1
  НЦ
    ЕСЛИ A[I] > A[I+1] ТО
         ОБМЕН A[I],A[I+1]
         F=1
   КЦ
    ЕСЛИ F=0 ТО ВЫХОД ИЗ ЦИКЛА
  КЦ
```

3. В каком порядке будут расположены элементы после сортировки данным алгоритмом?

```
For I = 0 To 3 do
     For J = 0 To 4 - i do
       If M[J]> M[J + 1] Then
        Begin
        Tmp: = M[J];
         M[J]: = M[J + 1];
         M[J + 1]: = Tmp;
       End;
```

4. В каком порядке будут расположены элементы после сортировки данным алгоритмом?

```
For I = 0 To 3 do
    For J = 0 To 4 - i do
      If M[J]< M[J + 1] Then
       Begin
       Tmp: = M[J];
        M[J]: = M[J + 1];
        M[J + 1]: = Tmp;
       End;
```

5. При выполнении программного кода в каком порядке будет выполнена сортировка?

```
procedure Inser;
var
  i, l: integer;
  x: DataItem;
begin
  for i := 2 to count do
  begin
    x := item[i];
    j := i-1;
    while (x<item[j]) and (j>0) do
    begin
      item[j+1] := item[j];
      j := j-1;
    end;
    item[j+1] := x;

  end;
end;
```

6. Для алгоритма

```
SumM(A,N;Sum);
Sum:=0;
for i:=1 to N
for j:=1 to N
Sum:=Sum+A[i,j];
```

7. Для данного алгоритма какова будет асимптотическая сложность?

```
Sum:=0;
for i:=1 to N
    If A[i]>0
    then
    Sum:=Sum+A[i];
```

8. Для данного алгоритма какова будет асимптотическая сложность?

```
Sum:=0;
for i:=1 to N
Sum:=Sum+A[i];
```

9. Для данного алгоритма какова будет асимптотическая сложность?

```
Sum:=0;
for i:=1 to N
for j:=1 to N
    for k:=1 to N
    Sum:=Sum+A[i,j,k];
```

10. Для данного алгоритма какова будет асимптотическая сложность?

```
Sum = 0
        For i = 1 to n
        If A[i]<0
        Then  Sum = Sum + A[i]
```

11. Дан алгоритм:

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

12. Дан алгоритм:

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

13. Дан алгоритм:

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

Какие исходные данные дадут средний случай функции трудоемкости алгоритма?

14. Дан алгоритм:

```
Sum = 0
For i = 1 to n
Sum = Sum + A[i]
```

Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

15. Дан алгоритм:

```
max = A[1]

For i = 2 to n

 If A[i]>max

 Then  max = A[i]
```

Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

16. Дан алгоритм:

```
max = A[1]
```

```
For i = 2 to n

 If A[i]>max

 Then  max = A[i]
```

Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

17. Дан алгоритм:

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

Какие исходные данные дадут лучший случай функции трудоемкости алгоритма?

18. Дан алгоритм:

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

Какие исходные данные дадут худший случай функции трудоемкости алгоритма?

19. Дан алгоритм:

```
Sum = 0

For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

Какие исходные данные дадут средний случай функции трудоемкости алгоритма?

20. Оценка сложности произведения двух функций $O(f*g)$ равна:
21. Оценка произведения сложности функции на постоянный множитель k (константа) $O(k*f)$ равна:
22. Оценка сложности частного двух функций $O(f/g)$ равна:
23. Оценка сложности суммы двух функций $O(f+g)$ равна:
24. Какую задачу решает данный алгоритм?:

```
Алг максимум (арг вещ a, в, с, арг рез max)
   Дано 3 вещественных не равных числа
   Надо Найти максимальное
Нач
   Ввод a, в, с
   Если a> b то max=a
```

Иначе max = b
Все
Если c>max то max=c

Вывод max

Кон

25. Какую задачу решает данный алгоритм?

```
If  n=1
Then
F = 1
Else
F = n*F(n-1);
```

26. Какую задачу решает следующая подпрограмма?

```
void QuickSort(double A[], int i1, int i2)
 {
     if (i1 < i2)
     {
         double pivot = A[i1];
         int is = i1;
         for (int i = i1 + 1; i < i2; i++)
         {
             if (A[i] <= pivot)
             {
                 is = is + 1;
                 swap(A[is], A[i]);
             }
         }
     }
     swap(A[i1], A[is]);
     QuickSort(A, i1,     is);
     QuickSort(A, is + 1, i2);
 }
```

27. Какую задачу решает следующая подпрограмма?

```
public static int linearSearch(int arr[], int elementToSearch) {

   for (int index = 0; index < arr.length; index++) {
      if (arr[index] == elementToSearch)
         return index;
   }
   return -1;
}
```

28. Какую задачу решает следующий алгоритм?

```
Sum = 0
For i = 1 to n
If A[i]<0
Then  Sum = Sum + A[i]
```

29. Какую задачу решает следующий алгоритм?

```
Sum = 0
```

```
For i = 1 to n

 If A[i] mod 2=0

 Then  sum=sum+ A[i]
```

30. Какую задачу решает следующий алгоритм?

```
kol = 0
For i = 1 to n
If A[i]>0
kol=kol+1
```

Составила Мохнач О.А.