Тестовые вопросы (алгоритмика, общие подходы к разработке программного обеспечения, ООП, регулярные выражения, реляционные базы данных, ОС).

1. Даны две функции вычисления чисел Фибоначчи (псевдокод):

|  |  |
| --- | --- |
|  |  |

Оцените асимптотическую сложность (О(n)) вычисления n-го числа Фибоначчи для обоих случаев (С – константа).

1. Для , для
2. Для обоих случаев
3. Для , для
4. Для , для
5. Оцените время работы O(n) , где n – количество элементов, для следующих операций над линейным односвязным списком (List) и массивом (Array):

* Поиск элемента (операция search)
* Вставка элемента в начало последовательности (операция insert)
* Доступ к элементу по индексу (операция at)

*Примечание: элементы в обеих структурах данных неупорядоченные, С - константа.*

1. Для List: search – O(n), insert – O(n), at - O(C). Для Array - search – O(C), insert – O(n), at - O(C).
2. Для List: search – O(n), insert – O(C), at - O(C). Для Array - search – O(n), insert – O(C), at - O(n).
3. Для List: search – O(n), insert – O(C), at - O(n). Для Array - search – O(n), insert – O(n), at - O(C).
4. Для List: search – O(C), insert – O(n), at - O(C). Для Array - search – O(C), insert – O(C), at - O(n).
5. Какая из структур данных даст худшее время поиска по значению ?
6. Упорядоченный массив
7. Хеш-таблица
8. Неупорядоченный линейный список
9. Бинарная куча
10. Дана структура данных H, представляющая собой хэш-таблицу и ее хэш-функция . В какую структуру выродится H при условии, что , т.е. хэш-функция возвращает одно и то же значение для любых данных?
11. Массив
12. Линейный список
13. Бинарное дерево
14. Упорядоченный массив
15. Что из перечисленного относится к механизмам разрешения коллизий при построении хэш-таблицы?
16. Двойное хеширование
17. Преобразование ячейки таблицы в массив ключей
18. Замена хеш-функции
19. Нормализация значений ключей
20. Какое регулярное выражение соответствует следующей строке (без кавычек) «user@mymail.com»?
21. ^use[A-E][+@[B-Z]{2,5}\.(com|ru|de)$](mailto:+@[B-Z]%7b2,5%7d\.(com|ru|de)$)
22. ^u[A-Za-z]+@my[a-z]{1,10}\.(com|ru)$
23. [^user@[^A-Za-z]\.com](mailto:%5euser@[%5eA-Za-z]\.com)$
24. ^[A-Za-z]{,3}@my[a-z]{1,10}\.(com|ru)$
25. Для каких целей можно использовать абстрактные классы в объектно-ориентированных языках программирования?
26. Определение интерфейса наследующего объекта без привязки к реализации
27. Создание типонезависмых алгоритмов
28. Вызов функции, переопределённой в унаследованном классе, с использованием указателя на базовый класс
29. Корректное освобождение ресурсов при разрушении объекта
30. Что понимается под динамическим полиморфизмом в современных объектно-ориентированных языках программирования?
31. Использование шаблонной реализации классов и методов для множества типов данных
32. Определение и подстановка адреса вызываемого метода на этапе компиляции в зависимости от типов аргументов
33. Определение и подстановка адреса вызываемого метода на этапе исполнения в зависимости от реального класса объекта, содержащего данный метод
34. Определение и подстановка адреса вызываемого метода на этапе исполнения в зависимости от реального класса объекта, вызывающего данный метод
35. Дана таблица Printers в базе данных, представляющая номенклатуру принтеров с указанием цены:

|  |  |  |
| --- | --- | --- |
| Model | Type | Price |
| HP Lj-1200 | Laser | 500 |
| Canon NP 300MN | Color | 300 |
| ……. |  |  |

Укажите SQL-запрос, отображающий модель и цену всех принтеров, имеющих максимальную цену:

1. SELECT model, price FROM Printer WHERE price=(SELECT MAX(price) FROM Printer)
2. SELECT model FROM Printer WHERE price=(SELECT MAX(price) FROM Printer)
3. SELECT model, MAX(price) FROM Printer
4. SELECT model, price, type FROM Printer WHERE price=(SELECT MAX(price) FROM Printer) AND type=’Laser’
5. Что такое «суррогатный ключ» в реляционных базах данных?
6. Служебное поле, добавляемое к уже имеющимся информационным полям таблицы, единственное предназначение которого, единственное предназначение которого – служить первичным ключом
7. Служебное поле, добавляемое к уже имеющимся информационным полям таблицы, единственное предназначение которого, единственное предназначение которого – служить составным ключом
8. Служебное поле, добавляемое к уже имеющимся информационным полям таблицы, единственное предназначение которого, единственное предназначение которого – служить внешним ключом
9. Некорректно выбранный первичный ключ, нарушающий свойство уникальности
10. В чем основное отличие языков с динамической типизацией от языков со статической типизацией?
11. В языках с динамической типизацией данные не имеют типа
12. В языках с динамической типизацией тип данных определяется на этапе выполнения
13. В языках с динамической типизацией тип данных определяется на этапе компиляции
14. В языках с динамической типизацией данные всегда принадлежат множеству типов
15. Какова сложность работы алгоритма поиска в бинарном сбалансированном дереве, например, АВЛ-дереве, в зависимости от количества элементов n в нем?
16. Какую структуру данных можно применить для реализации множества (коллекция неупорядоченных уникальных элементов) с гарантированным временем извлечения элемента порядка :
17. Хэш-таблица
18. Бинарное сбалансированное дерево
19. Очередь
20. Неупорядоченный список
21. В чем заключается отличие «жадных» (greedy) квантификаторов регулярных выражений от «ленивых» (lazy)?
22. «Жадному» квантификатору всегда соответствует несколько строк анализируемого текста.
23. При использовании «жадного» квантификатора гарантируется извлечение подстроки минимальной длины, соответствующей выражению
24. Встретив «жадный» квантификатор движок регулярных выражений будет выполнять сопоставление в обратную сторону
25. «Жадному» квантификатору соответствует строка максимальной длины из всех, соответствующих квантифицируемому выражению
26. Дано регулярное выражение: <.\*?> и текст: “<p>Данный <b>тест</b> проверяет знания <span>регулярных</span> выражений</p>”. Сколько сопоставлений даст поиск по тексту с использованием данного регулярного выражения?
27. 1 – весь текст
28. 3: “<p>”, “<b>”, “<span>”
29. 6: “<p>”, “</p>”, “<b>”, “</b>”, “<span>”, “</span>”
30. Ни одного
31. Что понимается под ядром операционной системы?
32. Набор файлов, представляющих основные функции ОС
33. Программный модуль, являющийся центральной частью операционной системы, в задачу которого входит обеспечение приложениям координированного доступа к основным ресурсам компьютера (процессорное время, память, устройства ввода-вывода и т.д.)
34. Программный модуль, предоставляющий другим приложениям абстракции файловой системы, графической подсистемы и подсистемы драйверов устройств.
35. Программный модуль, запускающий другие приложения ОС.
36. В современных операционных системах (например, семейства Linux или Windows) каким образом организовано взаимодействие прикладного ПО с ядром ОС?
37. Прикладное ПО модифицирует структуры данных ядра с использованием механизмов прямого доступа к памяти.
38. Прикладное ПО использует специальные функции – системные вызовы (system call), предоставляемые операционной системой.
39. Для взаимодействия с ядром ОС необходимо реализовывать прикладное ПО в виде модуля ядра.
40. Прикладное ПО использует механизм аппаратных прерываний.
41. В чем отличие интерпретируемых языков программирования от компилируемых?
42. Интерпретируемый язык программирования использует динамическую типизацию.
43. В интерпретируемых языках программирования отсутствуют механизмы обеспечения полиморфизма.
44. Интерпретируемые языки программирования обладают более высоким быстродействием.
45. Исходные файлы программ, разработанных на интерпретируемых языках программирования не нуждаются в компиляции перед исполнением
46. Что понимается под инкапсуляцией в объектно-ориентированном программировании?
47. Сокрытие деталей реализации и внутреннего поведения объекта.
48. Проектирование с использованием абстрактных классов.
49. Использование только private и protected модификаторов доступа для свойств объекта.
50. Использование шаблонов для реализации типонезависимого поведения алгоритмов

Вопросы по языку программирования (С++).

1. Объявлена функция: *void f(const MyClass\* m)*. Какие операции над объектом m могут быть выполнены внутри этой функции?
2. Доступ к полям объекта, вызов методов.
3. Доступ к полям объекта только на чтение, вызов методов.
4. Доступ к полям объекта только на чтение, вызов const-методов.
5. Доступ к полям объекта, в том числе помеченным как private, вызов методов.
6. Дана следующая программа на языке C++:

*#include <iostream>*

*class StatiClass {*

*public:*

*static int i;*

*};*

*int StatiClass::i;*

*int main(int argc, char\*\* argv) {*

*StatiClass\* c = (StatiClass\*)0;*

*std::cout << c->i;*

*}*

Что произойдет при ее исполнении

1. Программа «упадет»
2. Программа не скомпилируется
3. На экран будет выведено число 0
4. На экран будет выведено случайное число
5. Что произойдет в программе на С++ при попытке чтения\записи за границей массива?
6. Программа «упадет»
7. Неопределенное поведение, зависит от компилятора
8. Будет сгенерировано исключение *std::IndexOutOfRangeException*
9. Программа «зависнет»
10. В чем отличие ссылки от указателя?
11. По ссылке невозможно передавать примитивные типы – только объекты или структуры
12. Указатель должен быть обязательно проинициализирован в момент объявления, ссылка – нет.
13. Если ссылка является ссылкой на объект базового класса, то она не может указывать на объект производного класса
14. Ссылку невозможно изменить
15. Для каких целей можно использовать передачу параметра в функцию следующим образом: *void f(MyClass \*\* p)?*
16. Для изменения адреса, на который указывает указатель \*p
17. Таким образом можно получить доступ к private-членам объекта
18. Таким образом можно снять ограничение константности
19. Таким образом передается указатель на двумерный массив
20. Что означает следующее выражение: *typedef int T;*
21. T является шаблонным параметром типа int
22. Далее в программе T можно использовать как псевдоним типа int
23. Далее в программе int можно использовать как псевдоним типа T
24. Если тип T не является типом int – будет вызвана ошибка во время компиляции программы
25. Что означает следующая запись: *typedef int (\*T)(int ,int\*)*?
26. Здесь объявляется новый тип T, являющийся указателем на функцию, которая возвращает значение типа int и принимает значения типа int и указатель на int
27. Здесь объявляется новая функция с именем T, которая возвращает значение типа int и принимает значения типа int и указатель на int
28. Здесь происходит инициализация указателя T, являющегося указателем на функцию
29. Ничего не означает, пример синтаксически некорректен и не скомпилируется
30. Каким образом из функции языка C++ можно вернуть несколько значений?
31. Использовать тип данных union
32. Использовать встроенный тип данных кортеж (tuple)
33. Ввести в сигнатуру функции дополнительные параметры, передаваемые по неконстантной ссылке или указателю и записывать возвращаемые значения в них
34. Из функции языка C++ невозможно вернуть несколько значений
35. Дан следующий исходный код:

*#include <iostream>*

*union Un {*

*char a[2];*

*short b;*

*};*

*int main(int argc, char\*\* argv) {*

*Un u;*

*u.a[0] = 1;*

*u.a[1] = 1;*

*std::cout << u.b;*

*}*

Какой результат будет выведен на экран?

1. Число 1
2. Число 11
3. Число -1
4. Число 257
5. Что произойдет с локально созданным объектом при выходе программы за пределы блока кода, где данный объект был создан?
6. Ничего, объект останется в памяти.
7. Память занимаемая объектом будет очищена
8. Будет вызван деструктор объекта, затем очищена занимаемая им память
9. Будут последовательно вызваны деструкторы сначала самого объекта, затем его базового класса и далее по цепочке наследования, а затем очищена занимаемая объектом память