1. Алгоритм. Способы представления алгоритмов. Язык блок-схем.

А) Алгоритм - последовательность действий, которые необходимо выполнять для решения определённой задачи.

Б) Способы: словесная (записи на естественном языке), в виде блок-схем, в виде программы (тексты на языках программирования)

В) При блок-схемах алгоритм изображается в виде последовательности связанных между собой функциональных блоков и стрелок, соединяющих эти блоки. Каждому типу действий соответствует геометрическая фигура, представленная в виде блочного символа

2. Алгоритм. Способы представления алгоритмов. Язык структурного программирования,

А) Алгоритм - последовательность действий, которые необходимо выполнять для решения определённой задачи.

Б) Способы: словесная (записи на естественном языке), в виде блок-схем, в виде программы (тексты на языках программирования)

В) Язык включает 3 основные операции: действие, условие, цикл. Основная идея: отказ от оператора перехода в языках высокоуровневнего программирования, ибо могут возникать проблемы при работе программы (не выполняется очистка памяти). Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков.

3. Язык структурного программирования. Принципы структурного программирования. Методы разработки программ.

А) Язык включает 3 основные операции: действие, условие, цикл. Основная идея: отказ от оператора перехода в языках высокоуровневнего программирования, ибо могут возникать проблемы при работе программы (не выполняется очистка памяти). Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков.

Б) Принципы: 1) отказ от операторов безусловного перехода 2) любая программа строится на основе 3 базовых операций 3) базовые конструкции могу быть вложены друг в друга 4) повторяющиеся фрагменты желательно оформлять в виде подпрограмм 5) логически законченные блоки конструкций желательно объединять в блоки 6) все конструкции должны иметь один вход и выход 7) разработка идет по методу «сверху-вниз»

В) Снизу-вверх, сверху-вниз, смешанное.

4. Программа, Программирование, Компиляция. Трансляция. Интерпретация. (отличия)

А) Программа - алгоритмы, реализованные на языке программирования или языке машинных команд,

Б) Программирование – реализация алгоритма в виде набора команд или инструкций на каком-либо языке.

В) Компиляция - преобразование программы в набор машинных команд.

Г) Трансляция - перевод программы на другой язык

Д) Интерпретация - построчная трансляция и компиляция программы

5. Основные типы данных и их характеристики.

int - целочисленный, float - вещественный одинарной точности, doublе – вещественный двойной точности, char – символьный, string – строковый, bool - логический, pointer - типизированный или не типизированный указатель.

6. Статические и динамические массивы. Правила объявления и обращения к элементам

А) Массивы - последовательность данных одного типа. Статические - объем памяти не меняется. Динамические - меняются в ходе программы.

Б) Статический - тип имя [размер], динамический - тип \*имя = пем тип [размер]

7. Префиксные и суффиксные операции. Отличия.

Префиксные - операторы предшествуют операндам, суффиксные - операторы идут после операндов.

8. Арифметические, логические и побитовые операции. Обозначения. Принципы работы.

А) Арифметические: присваивание =, сложение +, вычитание -, умножение \*, целочисленное деление /, остаточное деление %, инкремент ++, декремент --. Логические: ! – не, true, false. Побитовые: инверсия - ~, ||- или, &- и, ^- исключающее или, « - сдвиг влево, » - сдвиг вправо.

9. Операции сравнения и условные операции.

А) == - равно, != - неравно, >/< - больше/меньше й, >=/<= - больше или равно/меньше или равно.

Б) if/else- если/иначе, switch case - множественный

10. Способы организации и условные операции.

Через объединение условий с помощью побитовых операций или использование switch case.

11. Циклы. Счётный цикл. Цикл с условием.

А) Цикл – управляющая конструкция в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.

Б) Цикл с использованием переменной-счетчика.

В) Цикл с условием-цикл работающий пока условие выполнимо.

12. Циклы. Циклы с предусловием и постусловием.

А) Цикл – управляющая конструкция в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.

Б) Цикл с постусловием – цикл, в котором условие проверяется после выполнения тела цикла. Цикл с предусловием – цикл, в котором выполняется, пока истинно некоторое условие, указанное перед его началом.

13. Проверка вводимых данных. Основные проблемы и способы их решения,

А) Проверка на ОДЗ – у четных корней, прямых и обратных тригонометрических функций. Проверка на границы - нельзя копировать больше чем вставить, иначе будет переполнение. Принудительное приведение к одной форме – нельзя использовать вместе знаковые и беззнаковые переменные.

14. Исключения (определение). Виды. Назначение. Способы использования.

А) Исключение - событие при выполнении программы, которое приводит к ее ненормальному или неправильному поведению.

Б) Аппаратные: деление на 0, выход за границы массива, обращение к невыделенной памяти, переполнение разрядной сетки. Программные: ОС и прикладные программы генерируют или возбуждают исключения.

В) В случае получения неверных входных данных.

15. Сигналы. Назначение. Сложности использования.

А) Сигналы - механизм передачи сообщений и команд между ОС и процессами программы.

Б) Сигнал представляет собой специальное сообщение, посылаемое процессу. В ответ на полученный сигнал процесс выполняет ряд действий в зависимости от типа сигнала.

В) Сигналы - последняя воля системы.

16. Алгоритм обработки. Поиск максимума и минимума.

А) Поиск минимума или максимума, сортировка массива, поиска в массивах вычисление

суммы элементов и т.д.

Б) Первый элементы становится опорным, сравниваем его с другим, если след. Элемент больше/меньше, то опорным становится этот элемент (проходим до конца массива)

17. Алгоритм обработки. Сортировка «пузырёк».

А) Поиск минимума или максимума, сортировка массива, поиска в массивах, вычисление суммы элементов и т.д.

Б) Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

18. Алгоритм обработки. Гномья сортировка.

А) Поиск минимума или максимума, сортировка массива, поиска в массивах, вычисление суммы элементов и т.д.

Б) Алгоритм похож на сортировку пузырьком - меняются по два элемента до тех пор, пока массив не будет отсортирован, но в отличии от пузырька при встрече неправильной пары алгоритм останавливается и начинается сначала

19. Алгоритм обработки. Сортировка расчёской.

А) Поиск минимума или максимума, сортировка массива, поиска в массивах, вычисление

суммы элементов и т.д.

Б) Алгоритм похож на пузырёк. Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального.

20. Алгоритм обработки. Быстрая сортировка.

А) Поиск минимума или максимума, сортировка массива, поиска в массивах, вычисление

суммы элементов и т.д.

Б) Выбирается опорное значение, выбираются два индекса и заносится в границы массива, массив переупорядочивается, так, что слева от опорного значения элементы меньше, справа больше, когда индексы будут равны друг другу – массив отсортирован.

21. Рекурсия (определение). Ограничения применения.

А) Рекурсия – вызова функции самой себя с изменением входных параметров.

Б) Главное условие – наличие базы – есть значение входного параметра, позволяющее выйти из рекурсии. Ограничивающий фактор – объем спец. памяти, где хранится контекст и адрес возврата.

22. Подпрограммы. Принцип работы подпрограмм. Отличие процедур и функций друг от друга.

А) Часть компьютерной программы, содержащая описание определённого набора действий. Способы реализации: процедуры и функции.

Б) Сохранение контекста, сохранение адреса возврата, пересдача управления первой команде подпрограммы, работа подпрограммы, после завершения подпрограммы восстановление сохранённой служебной информации и передача управления.

С) Функции отличаются от процедур фактом возвращения результата по своему имени.

23. Правила объявления функций и возвращения результата их работы.

Объявление: тип\_результата имя(параметры) {код}. Функция завершает работу после встречи return(возвращаемое значение) или дохода до конца, но в этом случае не будет сформировано возвращаемое значение

24. Виды параметров функций. Правила объявления. Назначения, возможности и ограничения каждого вида параметров.

А) Передача значения - данные обратно не передаются - значение аргумента копируется в параметр внутри функции. Объявление: тип\_результата имя(параметры) {код}. На значение аргумента никак не влияет.

Б) Передача адреса - функция имеет доступ к данным вызвавшей программы и может их менять. Объявление: тип\_результата &имя(параметры) {код}. Изменение параметра - изменение аргумента.

С) Передача указателя - передаётся адрес переменной, по которому нужно читать и писать данные. Объявление: тип\_результата \*имя(параметры) {код}. Изменение адреса - изменение адреса в параметре.

25. Параметры функции «по умолчанию». Правила объявления. Назначения, возможности и ограничения.

А) Параметр по умолчанию - это параметр функции, который имеет определённое (по умолчанию) значение. Если пользователь не передаёт в функцию значение для параметра по умолчанию, то используется значение по умолчанию. Если же пользователь передаёт значение, то это значение используется вместо значения по умолчанию.

Б) Задается как обычный параметр, к которому приравняли значение.

Все параметры по умолчанию в прототипе или в определении функции должны находиться справа. Если имеется более одного параметра по умолчанию, то самым левым параметром по умолчанию должен быть тот, который с наибольшей вероятностью (среди всех остальных параметров) будет явно переопределён пользователем.

26. Алгоритмы поиска. Основные виды. Принципы построения. Достоинства и недостатки.

А) Поиск – получение необходимого набора данных или информации из ранее сохраненной. Основные виды: последовательный, по ключам, бинарный поиск.

Б) Поиск по ключам - заменяет поиск содержимого ячейки поиском цифрового ключа (проблема удаления элементов)

В) Бинарный поиск - в упорядоченном массиве можно использовать метод дихотомии – половинного деления (требует поддержание деревьев в корректной форме при каждом изменении)

27. Файл. Определение для языка С++. Отличие от потоков. Виды потоков.

А) Файл - логическая концепция, применимая ко всему, с чем взаимодействует программа.

Б) Файл- средство взаимодействия с программой.

С) Поток - структура данных, обеспечивающих доступ к файлам

Д) По типу данных: текстовые и двоичные, по организации памяти: с последовательным и произвольным доступом, появляющиеся однократно и с возможностью неоднократного обращения.

28. Файл. Стандартные операции с файлами.

А) Файл- логическая концепция, применимая ко всему, с чем взаимодействует программа.

Б) Открытие файла, закрытие потока, чтение данных, запись данных, получение текущей

позиции в файле, перемещение в заданную позицию в файле, принудительная запись буфера.

29. Файл. Режимы доступа к файлам.

А) Файл - логическая концепция, применимая ко всему, с чем взаимодействует программа.

Б) Только чтение, только запись, чтение и запись, добавление информации после последней строки и байта, текстовый режим, двоичный режим, без буферизации.

30. Файл. Способы проверки правильности записи и чтения данных в файле. Контрольные суммы. Хэш-функции.

А) Файл - логическая концепция, применимая ко всему, с чем взаимодействует программа.

Б)

В) Контрольная сумма(хеш) - определенное значение, рассчитанное для данных с помощью известных алгоритмов. Предназначается для проверки целостности данных при передаче.

Г) Хеш-функция или функция свёртки — функция, осуществляющая преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины, выполняемую определенным алгоритмом.

31. Файл. Структура данных в файле. Типы. Достоинства и недостатки.

А) Файл - логическая концепция, применимая ко всему, с чем взаимодействует программа.

Б)

В) Символьные, числовые и логические данные (переменные) представляют собой простейшие и основные типы данных, хранимых и обрабатываемых в ЭВМ.

32. Динамическое выделение памяти. Правила выделения и освобождения. Операторы C++.

А) ДВП – способ работы с памятью, когда размер запрашиваемой памяти определяется изменяется в процессе работы.

Б) Выделение (для массива): тип \*имя [размер]; имя = new тип. Освобождение: delete имя; имя = NULL. Если нет присвоения значения NULL, то в C++ нет возможности узнать, свободна ли память или нет, программа уязвима, возможны ошибки. Изменение размера памяти требует: 1. выделение новой памяти, 2. копирование туда данных, 3. освобождение старой памяти, 4. установление указателей в NULL (nullptr). Перед использованием указатели нужно проверять на неравенство NULL.

В) New – выделение памяти; delete – освобождение памяти.

33. Динамическое выделение памяти. Списки. Их достоинства и недостатки.

А) ДВП - способ работы с памятью, когда размер запрашиваемой памяти определяется и изменяется в процессе работы.

Б) Связанный список - блок данных в памяти, имеющий связи с другими блоками, описывающие другие аналогичные наборы данных, или блоки иной структуры, хранящие дополнительные данные.

В) Недостатки(для двунаправленных списков): расход памяти на указатели, сложность в организации доступа к произвольному элементу. Достоинства: отсутствие необходимости выделения большого блока памяти, возможность увеличение размера элемента без выделения новой памяти.

34. Препроцессор. Назначение. Директивы препроцессора.

А) Препроцессор – программа, осуществляющая предварительную обработку какой-либо информации для другой программы. В программировании – программа, подготавливающая исходный код к компиляции.

Б) Препроцессор добавляет, изменяет, заменяет и/или удаляют программный код на основании алгоритмов своей реализации и специальных директив. Основные функции: объединение предварительно разделенной на несколько строк программы в одну строку, удаление комментариев, подключение содержимого другого файла, макроподстановки и макроопределение, проверка условий комплектации, вывод сообщений о предупреждениях и ошибках.

В) Директивы препроцессора - инструкция специального вида, предназначенная для сообщения процессору, каким образом нужно обработать код программы или его часть. Директивы начинается со знака #, далее следуют одно из ключевых слов: include, define, undef, if, ifdef, ifndef, else, elif, endif, error, warning, pragma.

35. Препроцессор. Правила создания макросов.

А) Препроцессор – программа, осуществляющая предварительную обработку какой-либо информации для другой программы. В программировании – программа, подготавливающая исходный код к компиляции.

Б) Макросы - это препроцессоры «функции», то есть лексемы, созданные с помощью директивы #define, которая принимают параметры подобно функциям. После директивы #define указывается имя макроса, за которым в скобках (без пробелов) параметры, отделенной запятыми и определение макроса, определённое пробелом. Параметры макросов лучше писать в скобках.

36. Перечисляемые типы. Назначение. Объявление и использование.

А) Перечисляемый тип - нечто среднее между структурой и макросом. Перечисление (или ещё перечисляемый тип) - это тип данных, где любое значение (или ещё «перечислитель») определяется как символьная константа. Объявить перечисление можно с помощью ключевого слова enum.

Б) Используются для выбора множества вариантов.

В) Enum имя типа (символьное значение 1, символьное значение Н). Можно приравнивать собственные значения, но по умолчанию идёт нумерация с нуля с прибавлением 1.

37. Перечисляемые типы. Комбинируемые и не комбинируемые типы. Объявление и использование.

А) Перечисляемый тип - нечто среднее между структурой и макросом. Перечисление (или ещё перечисляемый тип) - это тип данных, где любое значение (или ещё «перечислитель») определяется как символьная константа. Объявить перечисление можно с помощью ключевого слова enum.

Б) Комбинируемые - используются, когда задание режима или получения результата предполагает необходимость комбинации нескольких вариантов параметров. Не комбинируемые - используются для создания конкретного режима или получения единственного результата из множества.

В) Не комбинируемый: enum {значение 1, значение2, и т.д. }. Комбинируемый: enum {имя пер1\_зн1, имя пер1\_зн2, имя пер2\_зн1, и т.д.}.

38. Преобразование типов. Явные и не явные преобразования. Назначение.

А) Преобразование типов - процесс переноса числовых, символьных или иных данных из одного типа в другой. Бывает явным и не явным.

Б) Явное –оператор присваивания содержит разные типы данных, и компилятор знает правила их преобразования (делается программистом вручную), не явное – осуществляется для аргументов функций, не соответствующих типу параметров, при условии, что передаются значения, а не указатели ли адреса, компилятор может выдавать предупреждения (компутер осуществляет сам автоматически).

39. Преобразование типов. Примеры не явного преобразование.

А) Преобразование типов - процесс переноса числовых, символьных или иных данных из одного типа в другой. Бывает явным и не явным.

Б) Арифметическое преобразование (приводится к наибольшему типу): int + double -> double + double. Присваивание (меняется на тип переменой, к которой присваивают): int = double -> int = int. Передача функции аргумента (тип фактического к типу параметра): void f1(double x); int a; f1(a) -> f1(int) -> f1(double). Возврат из функции значения (тип возвращаемого к типу объявленного): double f2(void) {int b; return(b);} return(int) -> return(double)

40. Преобразование типов. Правила проведения явного преобразования.

А) Преобразование типов - процесс переноса числовых, символьных или иных данных из одного типа в другой. Бывает явным и не явным.

Б) Явное преобразование типов предполагает, что программер сам укажет, какие данные и к какому типу следует преобразовывать. A = int(b) или a = (int)b - устаревшие функции, ststic\_cast и dynamic\_cast (только для указателей и ссылок) – современные.

41. Перегрузка функций. Назначение. Правила использования.

А) Перегрузка функций - способ объявления нескольких одноимённых функций с разным набором данных, осуществляющих однотипные преобразования этих данных, при условии, что их неудобно создавать под разными именами.

Б) Компилятор приводит этим функциям разные идентификаторы и подбирает ту функцию, которая подходит под тип данных. Перегрузку можно использовать для формирования новых версий функций.

В) Преобразование типов осуществляется после выбора наиболее подходящей функции для оставшихся аргументов и их параметров, лучше самостоятельно явно преобразовывать типы, если в одну функцию были внесены правки - надо внести изменения в другие, нельзя объявлять функцию с одним набором типов данных, при необходимости параметров «по умолчанию» необходимо избегать коллизии наборов перезагружаемых функций.

45. Опти

а) ГК-чем дешевле необходимая элементарная база - тем лучше

Б) Построение алгоритмов и программ путём сокращения лишних операций, после использования ячейки передаются другим данным, константы лучше размещать в коде, а не в памяти данных, необходимо выяснить закономерности в данных и вычислять их вместо хранения, ограничение принимаемой информации блоками допустимого объема, выгрузка редко используемых данных в медленную память, уход от типовых структур к индивидуальным, использование подходящих по объему типов данных, использование битов для логических флагов, упрощение логического интерфейса.

42. Оптимизация программного кода. Критерии оптимизации.

А) Оптимизация – поиска решения, дающего лучшего значения каких-либо характеристик.

Б) Минимизация времени разработки, минимизация времени выполнения, минимизация используемой памяти, легкая портретируемость и переносимость, универсальность применения.

43. Оптимизация по времени разработки. Главный критерий и принципы.

А) ГК – чем раньше, тем лучше.

Б) Решайте известными типовыми способами, предварительно демонстрация интерфейса заказчику, в первую очередь разработчик основных функций, использование подпрограмм и функций, решения конкретных задач в ущерб универсальности, обучение пользователя работе программы.

44. Оптимизация по времени выполнения. Главный критерий и принципы.

А) ГК – чем быстрее результат, тем лучше.

Б) Часто возникающие ситуации должны: выполняться за минимальное время, проверять на условие первыми, не ждать более долгих задач в процессе своей обработки. Перестройка алгоритмов, приводящих к уменьшению повторения одинаковых действий, сокращение количества вызовов функций, сокращение количества операций в циклах, сокращение количества проверяемых условий, упрощение визуального интерфейса, распределение программного кода.

45. Оптимизация по объему используемой памяти. Главный критерий и принципы.

А) ГК – чем дешевле необходимая элементарная база – тем лучше.

Б) Построение алгоритмов и программ путем сокращения лишних операций, после использования ячейки передаются другим данным, константы лучше размещать в коде, а не в памяти данных, необходимо выяснить закономерности в данных и вычислять их в места хранения, ограничение принимаемой информации блоками допустимого объема, выгрузка редко используемых данных медленную память, уход от типовых структур к индивидуальным, использование подходящих по объёму тип данных, использование битов для логических флагов, упрощение логического интерфейса.

46. Оптимизация по простоте переносимости. Главный критерий и принципы.

А) Минимизация затрат на переделку, разработка и поддержка единого решения, использование готовых кроссплатформенных сред, использование интерпретируемых языков, использование трансляторов и условной компиляции, уход от системных вызовов, использование эмуляторов, создание платформно-зависимых библиотек и платформно-независимого кода, деление программы на легко заменяемые блоки.

Б) ГК – 5 минут и готово

47. Оптимизация по универсальности применения. Главный критерий и принципы.

А) Для каждой задачи реализуются весь спектр её возможных решений, код программы содержит блоки самоадаптации под среду запуска, возможна реализация отдельного интерфейса для настройки программы, по каждому желанию заказчик все возможные варианты дальнейшее пожеланий, вводятся механизмы поддержки дополнительных модулей расширения, публикуется документация для разработки этих модулей, могут проводиться обучающие курсы.

Б) ГК - лучше день потерять, чем потом за 5 минут доделать.