1. **Опишите одномерную модель памяти абстрактной машины. За что в модели памяти отвечает адрес? Может ли адрес не быть целым?**

Она бесконечна (программа думает, что вся память принадлежит ей), доступ за константное время, представляет собой упорядоченную последовательность ячеек памяти, в которой каждая ячейка имеет уникальный адрес.

Адрес отвечает за указание на область памяти

Адрес может быть только целым

2. **Дайте определение минимальной единице адресации.**

Минимальная единица адресации в программировании обозначает наименьший адресуемый элемент памяти в компьютерной системе. В большинстве систем - это 1 байт - значит каждый байт имеет свой уникальный адрес

3. **Какой адрес считается особым, нулевым? Почему мы выделяем такой адрес среди остальных? Правдив ли он?**

Нулевой адрес (NULL, stddef.h). Он используется для объявления нулевого указателя, чтобы отметить, что этот указатель не указывает на какой-либо адрес в памяти

Не правдив

4. **Как обрабатывается операция взятия адреса?**

Она возвращает адрес операнда

&a[i] == a + i

5. **Есть ли связь между размерами адреса и машинного слова на конкретной машине?**

Они взаимосвязаны.

Машинное слово - минимальный объем данных который можно передать за раз

В основном, размер адреса определяет максимальное количество адресуемых ячеек памяти, а размер машинного слова определяет максимальный объем данных, которые процессор может обрабатывать за один такт ???

6. **Сколько памяти можно адресовать на машине с размером адреса в 1 (2, 3, 4, 8, n) байт?**

Количество адресуемых ячеек = 2 ^ (размер адреса в битах)

7. **Опишите концепцию типа size_t.**

size_t – беззнаковый целый тип, предназначенный для представления размера любого объекта в памяти. Размер типа определяется относительно возможностей аппаратной платформы.

Тип определен в заголовочном файле stddef.h.

Максимальный размер size_t записан в константе SIZE_MAX, определённой в stdint.h.

8. **Опишите концепцию указателя.**

Указатель – это объект, содержащий адрес объекта или функции, либо выражение, обозначающее адрес объекта или функции

Разновидности указателей: типизированный указатель на данные (тип* имя); бестиповой указатель (void* имя); указатель на функцию.

Использование: передача параметров в функцию, обработка областей памяти

9. **Опишите два вида ошибок при работе с памятью — утечку памяти и висячий указатель.**

Утечка памяти или memory leak – это ошибка в исходном коде, при которой выделенная под переменную, массив, объект класса и т. д. динамическая память не освобождается и впоследствии теряется, а данные так и остаются в оперативной памяти до момента закрытия программы.

Висячий указатель или висячая ссылка (англ. Dangling pointer, wild pointer, dangling reference) — указатель, не указывающий на допустимый объект соответствующего типа. Висячие указатели возникают тогда, когда объект удалён или перемещён без изменения значения указателя на нулевое, так что указатель все ещё указывает на область памяти, где ранее хранились данные

10. **Дайте определение операции разыменования.**

Получение значения по адресу

11. **Почему выставлять астериск следует по правому слову при описании указателя?**

Выставление звёздочки (*), как правило, следует по правому слову, чтобы точно визуально определять является ли переменная указателем или нет

12. **Опишите правила адресной арифметики.**

p += n;

новый адрес в p = старый адрес из p + n * sizeof(тип)

p -= m;

новый адрес в p = старый адрес из p - m * sizeof(тип)

13. **Дайте определение агрегированному типу данных.**

Будем называть тип данных агрегированным, если из сущности этого типа можно выделить сущность определённого типа данных, простого или агрегированного

Будем считать, что мы в языке Си подразумеваем под «возможностью выделить сущность» возможность взятия адреса сущности.

14. **Может ли сущность агрегированного типа включать сущность другого агрегированного типа? Того же агрегированного типа?**

Да, может

15. **Дайте определение массиву, обозначив его свойства.**

Массив - агрегированный тип данных, обладающий тремя свойствами

- Линейность - все элементы массива в памяти расположены упорядочено единым неразрывным блоком
- Однородность - все элементы одного типа или одного объема
- Произвольный доступ - скорость доступа к любому элементу массива не зависит от его положения

16. **Могут ли различаться скорости доступа к элементам двух разных массивов?**

Доступ за константное время. Но процессор может кэшировать элементы

17. **Дайте определение статическому массиву.**

Будем называть статическим массив, размер которого известен на момент трансляции

18. **Корректен ли следующий код и почему:**

```
int a[100];
int b[100] = {0};
a = b;
```

Массив - неприсваиваемый тип данных, поэтому код выше некорректен

19. **Раскройте концепцию ключевого слова typedef. Какие задачи можно решить с помощью него? Рекомендуется ли решать те же задачи с помощью макросов?**

Язык С позволяет определять имена новых типов данных с помощью ключевого слова typedef. На самом деле здесь не создается новый тип данных, а определяется новое имя существующему типу. Стандартный вид оператора typedef следующий:

typedef тип имя;

typedef используется для:
- упрощение именования сложных типов данных
- создание псевдонимов для шаблонных типов

Не рекомендуется решать задачи typedef с помощью макросов, так как: typedef создает новый тип данных в области видимости, макросы могут заменяться глобально во всем коде; это безопаснее и удобнее

20. **Дайте определение подпрограмме, точке входа в оную, точке выхода из оной.**

Подпрограмма - именованный участок кода

Точка входа в подпрограмму - первая строка функции, с которой начинается выполнение подпрограммы

Точка выхода из подпрограммы: при помощи оператора return или последняя строчка функции подпрограммы

21. **Дайте определение явному возврату сущности из подпрограммы.**

Будем называть явным возвратом значения применение к значению оператора явного возврата. Оператор явного возврата return

22. **Сущности каких типов можно вернуть из подпрограммы явно? С помощью какого оператора описывается явный возврат в языке Си?**

Из подпрограммы можно явно вернуть одно значение простого типа данных или определенного агрегированного типа, список которых определен стандартом
Массив вернуть нельзя

Явный возврат описывает с помощью оператора return

23. **Дайте определения функции и процедуре.**

Функция - подпрограмма, возвращающая значение явно

Процедура - подпрограмма, не возвращающая значения явно

24. **Дайте определения формальным и фактическим параметрам подпрограммы.**

Формальные параметры указываются при описании подпрограммы, а фактические – непосредственно при её вызове. Фактические параметры часто называют аргументами.

25. **Раскройте концепцию передачи аргумента в подпрограмму по значению или по ссылке. Есть ли передача по ссылке в языке Си?**

При передачи по значению создается локальная копия переданного аргумента. При передачи по ссылке изменяется сама переданная сущность

В Си аргументы передаются по значению. Передача по ссылке имитируется с помощью передачи указателя. Это делается, если нужно изменить переданный аргумент

26. **Можно ли изменять внутри подпрограммы сущность, переданную по значению?**

Только если передается переменная указатель

27. **Чему равно var после исполнения кода:**

```
void proc(int a, int *b)
{
        *b = 4;
        *b = *b + a;
}
…
int var = 3;
proc(var, &var);
```

7, так как внутри функции proc сделается копия переменной в a. Поэтому при изменении значения по ссылке это не отразится на a

28. **Дайте определение области видимости. Как определяется время жизни сущности?**

Область видимости некоторой сущности - те строки кода, из которых можно получить доступ к этой сущности. Область видимости бывает:

- блок - последовательность объявлений, определений и операторов, заключенная в фигурные скобки (переменная, формальные параметры)
- файл - имена описанные за пределами какой-либо функции (имя фукнции)
- функция (метки)
- прототип функции - применяется к именам переменных, которые используются в прототипах функции

Время жизни - интервал времени выполнения программы, в течение которого “программный объект” существует. Бывает: глобальное (существует на протяжении всей программы), локальное (время жизни ограничено блоком), динамическое (с момента выделения памяти до момента ее освобождения)

29. **Дайте определение сигнатуре (прототипу) подпрограммы.**

Сигнатура - запись определенного формата из типа возвращаемого значения, имени подпрограммы и списка аргументов

30. **Дайте определение точке объявления, точке описания, точке вызова, расскажите о единственности каждой из них, опишите возможные варианты взаимного расположения.**

Точка объявления имени - это точка, в которой она становится видимой компилятору

Описание предоставляет компилятору все сведения, необходимые для создания машинного кода при последующем использовании сущности в программе. Располагается после объявления

Точка вызова - место, где происходит вызов. Всегда располагается после описания

Для каждой функции или переменной может быть только одно определение, но несколько объявлений

31. **Дайте определение перегрузке подпрограммы. Есть ли перегрузка подпрограмм в языке Си стандарта C99?**

Перегрузка подпрограмм позволяет создавать несколько подпрограмм с одним и тем же именем, при условии, что все эти подпрограммы имеют разные параметры (или они могут различаться иным образом).

В языке Си стандарта C99 нет перегрузки подпрограмм. Она появилась только в стандарте C11 в качестве дженерик макросов

32. **Дайте определение чистой функции. Различает ли транслятор чистые функции в языке Си**

Чистая функция - функция, которая является детерминированной (для одного и того же набора значений возвращает одинаковый результат) и не обладает побочными эффектами (функция, которая в процессе выполнения своих вычислений не модифицируют значения глобальных переменных и не осуществляют операции ввода и вывода)

Транслятор не различает чистые функции

33. **Дайте определение рекурсивной функции, базе рекурсии. Какая база является полной? Различает ли транслятор рекурсивные функции в языке Си?**

Рекурсивная функция - функция, у которой в теле функции находится точка её же вызова.

База рекурсии - набор условий, определяющих окончание рекурсивных обращений.

Если функция f(n) для любого n из множества N может вернуть некоторое значение, встав на базу, то база является «полной» для N

Транслятор не различает рекурсивные функции

34. **Ограничена ли рекурсивная функция по количеству вызовов?**

Не ограничена. Но если совершается много вызовов, то память может переполнится и произойдет ошибка

35. **Дайте описание следующей функции:**

```
double avg(int a[], size_t len)
{
        double sum = 0.0;
        for (size_t i = 0; i < len; i++)
                sum += a[i];
        return sum / len;
```

```
}
```

Функция считает среднее значение в массиве

36. **Дайте описание следующей функции:**

```
size_t copy_evens(int dst[], int rsc[], size_t rlen)
{
        size_t dlen = 0;
        for(size_t i = 0; i < rlen; i++)
                if(rsc[i] % 2 == 0)
                        dst[dlen++] = rsc[i];
        return dlen;
}
```

Функция добавляет в новый массив только четные числа из старого

37. **Существует ли такое N, при котором массив arr[N] может быть объявлен как глобальная переменная, но не может быть объявлен как локальная переменная?**

Да, такое возможно.

Если N превышает максимальное количество элементов, которое может быть выделено на стеке.

В большинстве современных компиляторов локальные переменные, включая массивы, обычно хранятся на стеке. Размер стека, в свою очередь, ограничен и зависит от операционной системы, компилятора и его настроек. Если размер массива arr[N] слишком большой и превышает доступное пространство на стеке, то компилятор может выдать ошибку при попытке объявления такого массива как локальной переменной.

Однако глобальные переменные, включая глобальные массивы, обычно хранятся в специальных сегментах памяти, и их размер может быть значительно больше, чем доступное пространство на стеке.

38. **Пусть в программе объявлен массив arr[N] как локальная переменная. Существует ли такое положительное N, при котором программа не сможет корректно работать?**

Да, из-за переполнения стека вызовов, программа не будет работать корректно

39. **Раскройте концепцию ключевого слова restrict.**

Объект, доступ к которому осуществляется через указатель restrict, имеет особую связь с этим указателем. Эта связь требует, чтобы все обращения к

этому объекту прямо или косвенно использовали значение этого конкретного указателя. Предполагаемое использование классификатора restrict заключается в оптимизации и удалении всех экземпляров классификатора из всех преобразований предварительной обработки. блоки, составляющие соответствующую программу, не меняют ее смысла (т.е. наблюдаемого поведения).

40. **Является ли, согласно Вашему определению, массив buf статическим:**

```
int func(int a[], size_t len)
{
        int buf[len];
        ...
}
```

Нет, так как длина массива buf может быть разной во время выполнения программы (так как функция может быть вызвана с разными фактическими параметрами)

41. **sizeof – этап трансляции или этап выполнения**

Трансляции

42. **Чему равен x**

```
int x = 3
{
        int x = 2;
}
```

x = 3

43. **Инициализация массива a с количеством элементов, равному размеру int**

```
int a[sizeof(int)];
```

Размер int может быть разным на разных машинах, а sizeof выполняется на этапе трансляции -> не переносимо

44. **Чему равно k**

```
int arr[100];
k = (arr == &arr);
```

arr и &arr указывают на начало массива, поэтому k=1

45. Чему равны p1 и p2

```
int a[100];
int *p1, int *p2;

p1 = a;
p2 = (a+1);
```

p1 – указывает на нулевой элемент, p2 – на первый.

46. Код

```
int func(int n)

{

        int a[n];
        int k = sizeof(a);

}
```