1. Каким образом можно грамотно переписать if с большой вложенностью, чтобы избавиться от нее?

Ответ: разбивать if-else на отдельные функции; находить исключительные ситуации и исключать последующие проверки путем выхода из функции; если встречается проблема, надо ее обработать, а затем проводить последующую проверку, дойдя таким образом до нормального выполнения функции.

1. В чем особенность арифметики указателей? Чем отличается результат i++, если i – указатель на char, float, структуру размером 15 байт?

Ответ: Арифметические действия сложения и вычитания с указателями приводят к увеличению/уменьшению адреса, который хранится в указателе, на размер типа указателя. То есть i++ для char будет увеличивать адрес переменной на 1, в случае float – на 4, а в случае структуры размером 15 байт – на 15.

1. Каким образом можно получить адрес переменной? А значение переменной (или блока памяти), на которую указывает указатель?

Ответ: можно создать указатель, который присвоим к переменной, хранящейся в памяти компьютера, с помощью операции &. y = &x – так указатель \*y может получить адрес переменной x. Далее командой printf(“%p”, y) можем вывести адрес переменной x.

Для получения значения переменной x, на которую указывает указатель \*y, потребуется написать printf(“%d”, \*y).

1. Чем опасно разыменование неинициализированного указателя?

int a;

int \*b;

int a = \*b;

Ответ: Неициализированный указатель дает каждый раз разный адрес. Нужно инициализировать хотя бы с помощью NULL. Статические и глобальные указатели при запуске равны NULL.

1. Что означает операция &ptr, если ptr – это указатель?

Ответ: С помощью &ptr в данном случае мы можем получить адрес указателя

1. Какое значение будет принимать переменная c?

char (\*str)[] = “Hey”;

char \*ptr\_str = str;

int \*b = ptr\_str;

int c = \*b;

Ответ: получим нулевой элемент строки в виде буквы “H”, так как имя массива указывает на нулевой элемент (наверное, это то, как я это понял, пробуя через printf вывести переменную c)

1. Для чего используется указатель void \*? В чем его особенность?

Ответ: void \* позволяет передать адрес, но не передавать кол-во байт (захватывается нисколько байт). Далее можно будет передать указатель на любой тип данных

1. Для чего используются указатели на функции? Каков синтаксис указателя на функцию?

Ответ: тип (void, int и т.д.) затем (\*имя указателя) и (аргументьы): void (\*fn\_math) (void)

Указатели на функцию используются для передачи функции в качестве параметров другим функциях, а также для создания массивов функций.

1. Для чего используется NULL? Зачем применяют проверку указателя на NULL для указателей, переданных в функцию?

Ответ: используется, если необходимо инициализировать указатели нулем. Null – макрос из <stddef.h>, который представляет из себя (void\*)0 нулевой адрес без указания захвата конкретного кол-ва байт.

1. Для чего используется оператор typedef? Как он применяется для базовых типов, структур, для указателей на функции?

Ответ: используется для переименовывания типов. Можно какому-то указателю с помощью данного оператора дать конкретное имя и использовать его в дальнейшем.

Более простой случай typedef float name. Программа будет определять name в дополнение к float для определения типа float

1. typedef float balance;
2. typedef struct {

float x;

int b;

} usart;

1. typedef char (\*calculate)(int);