28. Структуры.

План ответа:
1 понятие «структура»;
2 определение структурного типа;
3 структура и ее компоненты (тэг, поле);
4 определение переменной-структуры, способы инициализации переменной-структуры;
5 операции над структурами;
6 особенности выделения памяти под структурные переменные;
7 структуры с полем переменной длины (flexible array member, C99).

1 Структура представляет собой одну или несколько переменных (возможно, разного типа), которые объединены под одним именем, при этом каждая переменных занимает своё место в памяти. Это позволяет связать в единое целое различные по типу и значению данные, связанные между собой.

2 Формат определения:

```
struct [tag]
{
        type1 field1;
        type2 [field2];
        ...;
        typeN fieldN;
} [variable] = { value1, value2, ..., valueN }
```

3 Тег структуры — имя, позволяющее идентифицировать структуру в самых разных частях программы.
Его передают в качестве спецификатора типа (вместе с ключевым словом struct) формального параметра.
Тег может быть опущен.
Тело структуры представляет собой самостоятельную область видимости: имена в этой области не конфликтуют с именами из других областей.
Поля - перечисленные в структуре переменные
Поля структуры располагаются в памяти в порядке описания.
С целью оптимизации доступа компилятор может располагать поля в памяти не одно за другим, а по адресам кратным, например, размеру поля.
Адрес первого поля совпадает с адресом переменной структурного типа.
Поля структуры могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на него.

4 Инициализация
Для инициализации переменной структурного типа необходимо указать список значений, заключенный в фигурные скобки.
Значения в списке должны появляться в том же порядке, что и имена полей структуры.
Если значений меньше, чем полей структуры, оставшиеся поля инициализируются нулями.

Инициализация в C99

```
struct date exam =
        {.date = 4, .month = 1, .year = 2016};
```

+ Такую инициализацию легче читать и понимать.
+ Значения могут идти в произвольном порядке.
+ Отсутствующие поля получают нулевые значения.
+ Возможна комбинация со старым способом.

```
struct date exam =
        {.date = 4, 1, .year = 2016};
```

5 Операции над структурами:
1) Обращение к полю по переменной: структура.поле
2) Обращение к полю по указателю на переменную: структура->поле
3) Присвоение одной структуре значений другой структуры того же типа.
//корректно передаются и копируются статические массивы
4) Структуры могут передаваться в функцию в качестве параметра и возвращаться как значения.
5) Структуры НЕЛЬЗЯ сравнивать (==, !=)

6 При объявлении переменной типа заданной структуры выделяется память для всех членов структуры.
	При этом память под отдельный член структуры может дополняться до 4 или 8 байт в зависимости от компилятора
	Для того, чтобы память под структуру использовалась эффективно, можно воспользоваться директивой #pragma pack

```
	#pragma pack(push, 1)
	struct
	{
		char a;
		int b;
	} c2;
	#pragma pack(pop)
	sizeof(c2) == 5
```

7 C99: особенности использования структур - flexible array member
	struct {int n, double d[]};
		Подобное поле должно быть последним.
		Нельзя создать массив структур с таким полем.
		Структура с таким полем не может использоваться как член в «середине» другой структуры.
		Операция sizeof не учитывает размер этого поля (возможно, за исключением выравнивания).
		Если в этом массиве нет элементов, то обращение к его элементам — неопределенное поведение.

	Особенность выделения памяти:
		struct s *elem = malloc(sizeof(struct s) + n * sizeof(double));