6.1. Правила описания структур.

6.2. Операции, используемые для выделения элементов структур.

6.3. Выделение элементов структур, адресуемых указателем.

6.4. Назначение функции ***malloc()****.*

6.5. Отличие ***calloc()*** от ***malloc()***.

6.6. Правила инициализации структур при описании.

6.7. Особенности выделения памяти под структуры.

6.8. Назначение функций ***realloc()*** и ***free()***.

6.9. Как передать структуру в функцию?

**6.5** malloc означает **выделение памяти**, тогда как calloc означает **непрерывное выделение**.

1. malloc принимает только **один аргумент**, размер блока, тогда как calloc принимает **два аргумента**, количество блоков, которые должны быть выделены, и размер каждого блока.

**ptr = (cast-type \*) malloc (размер байта)//malloc**

**ptr = (cast-type \*) calloc (нет блоков, размер блока);//calloc**

1. malloc не выполняет инициализацию памяти, и все адреса хранят **значение мусора**, тогда как calloc выполняет инициализацию памяти, а адреса инициализируются как **значениями нуля или нулей**.

**6.8** Функция free освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или reallocосвобождается. То есть освобожденная память может дальше  использоваться программами или ОС.

**6.4** Функция malloc выделяет блок памяти, размером sizemem байт, и возвращает указатель на начало блока.

Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.

**6.9** Лучший способ сделать это - определить структуру глобально, а затем использовать ее ярлык для объявления необходимых структурных переменных и параметров

**6.7** #include <stdlib.h>

struct st \*x = malloc(sizeof \*x);

1. x должен быть указателем
2. литье не требуется
3. включить соответствующий заголовок