Задание 1
Трехмерный массив целых чисел размером 2x3x4 можно описать следующим образом:

```
#include <stdio.h>

int main()
{
   int arr[2][3][4] = {
   {{0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11}},
   {{12, 13, 14, 15}, {16, 17, 18, 19}, {20, 21, 22, 23}}
   };
   return 0;
}
```

В этом массиве есть 2 "слоя", каждый слой содержит 3 "строки", а каждая строка содержит 4 "столбца". Массив может быть заполнен произвольными значениями.

Задание 2
Дамп памяти этого массива, содержащий все значения, может быть представлен в виде последовательности 24 целых чисел, по 4 числа на каждый элемент. Порядок значений в дампе памяти будет зависеть от порядка заполнения массива, но в общем случае он может быть представлен следующим образом:

```
(gdb) x /24xw arr
0xffffffff2a8: 0xfffff498    0x0000ffff    0xf7fa8dd0    0x0000ffff
0xffffffff2b8: 0x00000000    0x00000000    0xfffff420    0x0000ffff
0xffffffff2c8: 0xf7fd41ac    0x0000ffff    0xfffff488    0x0000ffff
0xffffffff2d8: 0x00000001    0x00000000    0xaaab0d80    0x0000aaaa
0xffffffff2e8: 0xf7ffe040    0x0000ffff    0xfffff310    0x0000ffff
0xffffffff2f8: 0xf7e373c0    0x0000ffff    0xfffff488    0x0000ffff
```

Задание 3
Для этого массива компоненты могут быть фиксированы постепенно следующим образом:

```
Двухмерный массив

(gdb) x /12xw arr[0]
```

```
0xffffffff2a8: 0xfffff498      0x0000ffff      0xf7fa8dd0      0x0000ffff
0xffffffff2b8: 0x00000000      0x00000000      0xfffff420      0x0000ffff
0xffffffff2c8: 0xf7fd41ac      0x0000ffff      0xfffff488      0x0000ffff
```

Одномерный массив

```
(gdb) x /4xw arr[0][0]
0xffffffff2a8: 0xfffff498      0x0000ffff      0xf7fa8dd0      0x0000ffff
```

Элемент

```
(gdb) x /1xw arr[0][0]
0xffffffff2a8: 0xfffff498
```

Задание 4

Для работы с трехмерным массивом необходимо использовать указатели на указатели на указатели. Для определенности, рассмотрим массив типа int:

```
int arr[2][3][4];
```

Указатель на первый элемент массива arr будет иметь тип int (*p)[3][4]. Указатель на первый элемент второго измерения массива arr будет иметь тип int (*p)[4]. Указатель на первый элемент третьего измерения массива arr будет иметь тип int *. Размер элемента массива равен sizeof(int) = 4 байта.

```
#include <stdio.h>

int main()
{
    int arr[2][3][4] = {0};

    // Указатель на первый элемент arr
    int (*p)[3][4] = arr;

    // Указатель на первый элемент второго измерения массива
    int (*q)[4] = *p;

    // Указатель на первый элемент третьего измерения массива
    int *r = **p;

    // Использование указателей для обращения к элементам массива
    for (int i = 0; i < 2 * 3 * 4; i++)
```

```
    {
        printf("arr[%d][%d][%d] = %d\n", (i / 12) % 2, (i / 4) % 3, i % 4, *(*(*(p) + (i / 4) % 3) + i % 4));
    }

    return 0;
}
```

Чтобы проверить теоретический расчёт размера элемента трёхмерного массива целых чисел размерами 2, 3 и 4, можно использовать отладчик GDB и вывести размер элемента с помощью команды sizeof.

Вот как это можно сделать:

Создайте программу, объявите трёхмерный массив целых чисел размерами 2, 3 и 4 и присвойте значения его элементам.

```
#include <stdio.h>

int main()
{
    int arr[2][3][4] = {
    {{0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11}},
    {{12, 13, 14, 15}, {16, 17, 18, 19}, {20, 21, 22, 23}}
    };
    return 0;
}
```

Соберите программу с отладочной информацией, используя флаг -g:

```
(gdb) print sizeof(int)
$1 = 4
(gdb) print sizeof(arr)
$2 = 96
(gdb) print sizeof(arr[0])
$3 = 48
(gdb) print sizeof(arr[0][0])
$4 = 16
(gdb) print &arr
```

```
$5 = (int (*)[2][3][4]) 0xfffffffff2a8
(gdb) print &arr[0]
$6 = (int (*)[3][4]) 0xfffffffff2a8
(gdb) print &arr[0][0]
$7 = (int (*)[4]) 0xfffffffff2a8
```

чтобы узнать размер элемента массива

```
(gdb) print sizeof(int)
```

Результат будет равен 4 байта, так как тип int занимает 4 байта на большинстве систем.

размер всего массива

```
(gdb) print sizeof(arr)
```

Результат будет равен 24 * 4 = 96 байт, так как в массиве 2 * 3 * 4 = 24 элемента, каждый из которых занимает 4 байта.

Проверить размер каждой компоненты массива

```
(gdb) print sizeof(arr[0])
(gdb) print sizeof(arr[0][0])
```

Результаты будут равны 12 * 4 = 48 байт и 4 байта соответственно.

адреса первого элемента массива и первого элемента каждой его компоненты

```
(gdb) print &arr
(gdb) print &arr[0]
(gdb) print &arr[0][0]
```

Результаты будут разными, так как каждая компонента занимает разное количество памяти.

Задание 5

```
void processLevel(int level, int arr[][3][4], int size);
```

Параметр “level” — это уровень массива, который мы хотим обработать.

Параметр “arr” представляет собой трехмерный массив, содержащий данные для обработки.

Параметр “size” — это количество элементов на уровне массива.