ЗАДАНИЕ №3.3

Часть 1. Как представлены строки в памяти?

Создадим строку:

```
#include <stdio.h>

#define STR_LEN (19  + 1)

int main(void)
{
    char str[] = "Hello, world!";

    return 0;
}
```

Выведем дамп памяти, содержащий эту строку

```
(gdb) x/20b str
0x7fffffffdfaa:      72      101     108     108     111     44      32      119
0x7fffffffdfb2:      111     114     108     100     33      0       0       51
0x7fffffffdfba:      65      -117    -116    36
```

Как мы видим, первые 13 символов — наша строка «Hello, world!», затем идут два служебных нулевых символа, а оставшаяся память, выделенная под строку, заполнена мусорными значениями.

Часть 2. Представление массива слов с помощью двумерного массива строк

1.

Создадим двумерный массив слов:

```
#include <stdio.h>
```

```
#define STR_LEN (9 + 1)

int main(void)
{
    char str[][STR_LEN] = { "March", "April", "May" };

    return 0;
}
```

Выведем дамп памяти содержащий этот двумерный массив полностью

```
(gdb) x/30b str
0x7fffffffdf90:    77    97    114    99    104    0    0    0
0x7fffffffdf98:    0    0    65    112    114    105    108    0
0x7fffffffdfa0:    0    0    0    0    77    97    121    0
0x7fffffffdfa8:    0    0    0    0    0    0
```

2.

Дамп памяти соответствует формату «байт памяти» - «код символа строки». Можно заметить, что под слово выделяется 10 байт. Сначала записываются коды символов слова, а затем в память записываются нулевые служебные символы, означающие отсутствие полезной информации. Таким образом, под слова выделяется 30 байт, заполненных следующим образом:

M	a	r	c	h	\0	\0	\0	\0	\0
A	p	r	i	l	\0	\0	\0	\0	\0
M	a	y	\0	\0	\0	\0	\0	\0	\0

3.

Структура занимает 30 байт, из которых 13 байт полезных и 17 байт вспомогательных служебных символов «\0»

Часть 3. Представление массива слов с помощью массива указателей на строки

1.

Создадим массив указателей на строки:

```
#include <stdio.h>

int main(void)
{
    const char *str[] = { "March", "April", "May" };

    return 0;
}
```

Выведем дамп памяти содержащий этот массив указателей полностью:

```
(gdb) print sizeof(str)
$1 = 24
(gdb) x/24b *str
0x555555556004: 77  97  114 99  104 0   65  112
0x55555555600c: 114 105 108 0   77  97  121 0
0x555555556014: 1   27  3   59  48  0   0   0
```

2.

Дамп памяти соответствует формату «байт памяти» - «код символа строки». Можно заметить, что размер такого массива указателей = количество слов * размер указателя в байтах = 3 * 8 = 24. Слова записываются в массив последовательно. Разделителем между словами служит символ «\0». Оставшаяся память заполнена мусором. Таким образом, под слова выделяется 24 байта, заполненных следующим образом:

M	a	r	c	h	\0	A	p
r	i	l	\0	M	a	y	\0
Мусор	Мусор	Мусор	Мусор	Мусор	\0	\0	\0

3.

Структура занимает 24 байта, из которых 13 байт полезных, 6 байт вспомогательных служебных символов \0 и 5 мусорных символов.