Задание №5

Представление в памяти строк и массивов строк

Представление строки в языке Си:

Строка хранится в памяти с помощью последовательных ячеек памяти, а строка завершается нулевым символом “\0”(известным как NULL)
Пример: строка “Hello” представлена следующим образом :

‘H’	‘e’	‘l’	‘l’	‘o’	‘\0’

Примечание:
Символы (например, ‘H’,’e’, …) кодируются с помощью кода ASCII.
Таким образом, “H” – это не что иное, как маленькое(1 байт) целое число.

Объявим в программе несколько строк разными способами:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void)
{
    char *str_1="Hello";
    char str_2[10]="Guten tag";
    char str_3[]="Privet";
    char str_4[] = {'J','u','n', 'e', '\0'};
    char str_5[5] = {'J','u','l', 'y', '\0'};
    printf("%s \n", str_1);
    printf("%s \n", str_2);
    printf("%s \n", str_3);
    printf("%s \n", str_4);
    printf("%s \n", str_5);
    return EXIT_SUCCESS;
}
```

Через gdb выведем дамп памяти, который содержит строки полностью:

```
(gdb) x /6xb str_1
0xaaaaaaaaaa08: 0x48    0x65    0x6c    0x6c    0x6f    0x00
(gdb) x /10xb str_2
0xffffffffeed8: 0x47    0x75    0x74    0x65    0x6e    0x20    0x74    0x61
0xffffffffeee0: 0x67    0x00
(gdb) x /7xb str_3
0xffffffffeed0: 0x50    0x72    0x69    0x76    0x65    0x74    0x00
(gdb) x /5xb str_4
0xffffffffeec0: 0x4a    0x75    0x6e    0x65    0x00
(gdb) x /5xb str_5
0xffffffffeec8: 0x4a    0x75    0x6c    0x79    0x00
```

Каждый символ строки представлен в 16-ричной системе счисления. Число в дампе памяти – это код символа из таблицы ASCII.
Например: В str_1 дамп памяти первого элемента равен 48, в таблице ASCII 48 в 16-ричной системе счисления равен символу “H”, второй дамп элемента равный 65 в таблице

ASCII равен “e” и так далее. Последний дамп элемента каждой строки равен 0 (NULL), что означает конец строки.

Способы хранения массива слов:

1 Способ : двумерный массив строк

```
char arr_1[][6] = {"One", "Two", "Three", "Four"};
```

O	n	e	\0	\0	\0
T	w	o	\0	\0	\0
T	h	r	e	e	\0
F	o	u	r	\0	\0

2 Способ: массив указателей на строки (“ragged array”)

```
char *arr_2[] = {"One", "Two", "Three", "Four"};
```

Массив arr_2 на самом деле не содержит строк, он содержит только адреса этих строк.

[0]	➔	O	n	e	\0		
[1]	➔	T	w	o	\0		
[2]	➔	T	h	r	e	e	\0
[3]	➔	F	o	u	r	\0	

1.Представление дампа памяти , который содержит массив полностью:

Пример программы:

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

#define SIZE 4

int main(void)
{
    char arr_1[][6] = {"One", "Two", "Three", "Four"};
    char *arr_2[] = {"One", "Two", "Three", "Four"};

    for (size_t i = 0; i < SIZE; i++)
    {
        printf("%s \n", arr_1[i]);
        printf("%s \n", arr_2[i]);
```

```
    }

    return EXIT_SUCCESS;
}
```

Представление дампа памяти для первого массива arr_1

```
(gdb) x /24xb arr_1
0xffffffffeeb0: 0x4f    0x6e    0x65    0x00    0x00    0x00    0x54    0x77
0xffffffffeeb8: 0x6f    0x00    0x00    0x00    0x54    0x68    0x72    0x65
0xffffffffeec0: 0x65    0x00    0x46    0x6f    0x75    0x72    0x00    0x00
```

Представление дампа памяти для второго массива arr_2

```
(gdb) p arr_2
$1 = {0xaaaaaaaaaad0 "One", 0xaaaaaaaaaad8 "Two", 0xaaaaaaaaaae0 "Three",
0xaaaaaaaaaae8 "Four"}
(gdb) x /19xb *arr_2
0xaaaaaaaaaad0: 0x4f 0x6e 0x65 0x00 0x54 0x77 0x6f 0x00
0xaaaaaaaaaad8: 0x54 0x68 0x72 0x65 0x65 0x00 0x46 0x6f
0xaaaaaaaaaae0: 0x75 0x72 0x00
```

Во втором случае дамп памяти содержит указатели строк.

2. В этих дампах памяти содержится информация о каждом символе строк в 16-ричной системе счисления . Числа в дампах памяти – код символа из таблицы ASCII. При этом “0x00” означает конец строки.

3. В первом случае длина каждой строки в массиве составляет 6 символов.
1 символ типа “char” составляет 1 байт. Так как у нас 4 строки в массиве: 4*6=24 байта.
Убедимся в этом:

```
(gdb) p sizeof(arr_1)
$1 = 24
```

Во втором случае у нас массив указателей, размер указателя постоянен и занимает в зависимости от разрядности операционной системы если 32-бит ОС, то 4 байта, в данном случае ОС 64-битная, поэтому 8 байт, из-за этого размер данного массива постоянен: 8*4=32 байта

```
(gdb) p sizeof(arr_2)
$2 = 32
```

В первом случае размер полезных элементов составляет 19 байт, а вспомогательных 5 байт.
Во втором случае размер полезных элементов составляет 32 байта, а вспомогательных 0 байт, так как их нет, при объявлении массива строк через указатель.