DMP: Дамп памяти

Задача: написать функцию, выводящую на экран шестнадцатеричный дамп блока памяти. Написанную функцию использовать в тестовой программе для вывода дампа двух массивов — целочисленного `a[]` и вещественного `b[]`, содержащих **одинаковые** значения (`a[i] == b[i]`).

0. **Подготовьте массивы (0 балов).** Напишите программу, определяющую два массива (`int` и `double`) размера `N=9` и заполняющую их четными целыми числами, начиная с 600. Выведите эти массивы на экран. Для заполнения и печати напишите отдельные функции:

   ```
   void FillInt(int a[], int size);
   void FillDouble(double a[], int size);

   void PrintInt(int a[], int size);
   void PrintDouble(double a[], int size);
   ```

1. **Посмотрите на память (+1 балл).** Изучите при помощи вашей среды разработки как хранятся в памяти эти массивы. В Visual Studio для этого есть специальное окно, его можно вызвать через меню Debug / Windows / Memory или горячей клавишей `Alt+6`. Поставьте точку останова в подходящем месте, запустите программу в режиме отладки, откройте окно с памятью. В появившемся окне в качестве адреса укажите имя массива, изучите шестнадцатиричное представление байтов, лежащих в памяти здесь и далее.

2. **Выведите HEX самостоятельно (+1 балл).** Напишите функцию, которая печатает последовательность байт указанного блока памяти в шестнадцатеричном виде через пробел, по две цифры на байт (`"%02X"`), последовательно в цикле передвигая переданный указатель дальше по памяти.

   ```
   void MemoryDump(void const* ptr, int size);
   ```

   Вызовите функцию по очереди для обоих массивов, вычисляя их размер через `sizeof()`, чтобы передать в функцию вторым аргументом. Первый массив выглядит примерно так:

   ```
   58 02 00 00 5a 02 00 00 5c 02 00 00 5e 02 00 00 60 02 00 00 62 02 00 00 64 02 00
    00 66 02 00 00 68 02 00 00
   ```

3. **Добавьте удобств (+1 балл).** Модифицируйте функцию для традиционного красивого вывода, разбивая на строки по 16 байт, и предваряя каждую строчку адресом (`"%P"`) первого байта этой строки:

   ```
   00D51970: 58 02 00 00 5a 02 00 00 5c 02 00 00 5e 02 00 00
   00D51970: 60 02 00 00 62 02 00 00 64 02 00 00 66 02 00 00
   00D51970: 68 02 00 00
   ```

   Сравните с тем, что вы видите во время отладки в вашей среде разработки.

4. (*) **Покажите символьное представление (+1 бонус).** Добавьте справа через 2 пробела столбец из 16 букв, которые соответствуют напечатаным в строке байтам. Если код соответствует печатному символу (`isprint()`), то напечатать его (`"%c"`), а непечатные символы заменить точками.

```
00D51970: 58 02 00 00 5a 02 00 00 5c 02 00 00 5e 02 00 00  X...Z...\...^...
00D51970: 60 02 00 00 62 02 00 00 64 02 00 00 66 02 00 00  `...b...d...f...
00D51970: 68 02 00 00                                      h...
```

Создайте разными способами три одинаковые текстовые строчки, изучите, как они хранятся в памяти.

```
char  s0[10] = "Hello!";  // в конце нули
char  s1[10];             // в конце мусор CC CC CC ...
char* s2 = malloc(10);    // в конце мусор CD CD CD ...

strcpy(s1, s0);
strcpy(s2, s0);
```