Семинар #8: Память. Домашнее задание.

Задача 1. Печать разных типов:

Напишите функцию `void polyprint(const char* type, void* p)`, которая должна будет печатать то, на что указывает указатель `p`. Тип того, на что указывает `p`, задаётся с помощью первой переменной и может принимать следующие значения:

- Если `type == "Integer"`, то `p` указывает на целое число типа `int`.
- Если `type == "Float"`, то `p` указывает на вещественное число типа `float`.
- Если `type == "Character"`, то `p` указывает на символ (тип `char`).
- Если `type == "Date"`, то `p` указывает на структуру `Date` (определение этой структуры смотрите выше).
- Если `type == "Movie"`, то `p` указывает на структуру `Movie` (определение этой структуры смотрите выше).
- Если `type == "String"`, то `p` указывает на первый символ строки.
- Если `type == "IntegerArray 15"`, то `p` указывает на первый элемент массива размером 15. Элементы этого массива имеют тип `int`. Нужно распечатать все элементы через пробел. Тут нужно использовать функцию `sscanf`, для того чтобы распарсить строку `type`.
- В ином случае функция должна печатать `Error!`

В любом случае, в конце функция должна печатать символ перехода на новую строку. Для сравнения строк нужно пользоваться функцией `strcmp`. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
struct date
{
    int day, month, year;
};
typedef struct date Date;
struct movie
{
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;
// Тут нужно написать функцию polyprint

int main()
{
    int a = 123;
    polyprint("Integer", &a);
    float b = 1.5;
    polyprint("Float", &b);
    char c = 'T';
    polyprint("Character", &c);

    Date d = {15, 5, 1970};
    polyprint("Date", &d);
```

```
    Movie e = {"Inception", 8.661, {8, 6, 2010}};
    polyprint("Movie", &e);

    char f[] = "Sapere Aude";
    polyprint("String", f);
    int g[] = {10, 20, 30, 40, 50};
    polyprint("IngerArray 5", g);
}
```

Задача 2. Изменить символы:

Напишите функцию `void set_characters(char* begin, char* end, char c)`, которая задаёт символы в строке символом `c`. Начиная с символа, на который указывает `begin` и заканчивая символом на который указывает `end` (но не включая его). Гарантируется, что `end` указывает на символ, находящийся в этой же строке и не левее символа, на который указывает `begin`. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
// Тут нужно написать функции set_characters

int main()
{
    char s[] = "Sapere Aude";
    set_characters(&s[2], &s[8], 'b');
    printf("%s\n", s); // Должно напечатать Sabbbbbbude
    set_characters(s, &s[4], 'a');
    printf("%s\n", s); // Должно напечатать aaaabbbbude
}
```

Задача 3. Просмотр памяти:

Как выглядит память, инициализируемая при создании следующих переменных (в системе с порядком байт Little Endian):

- `int a = 0x11223344;`
- `int b = 65535;`
- `int c = -1;`
- `int array[3] = {10, 2000, 65535};`
- `char str[8] = ''Hello'';`
- `float x = 1.0f;`
- ```
  struct data {
      char str[5];
      int number;
  };
  struct data c = {"Cat", 100000};
  ```

Память представить в виде последовательности 2-значных шестнадцатеричных чисел. Например число $123456 = 1e240_{16}$ будет храниться в памяти как `40 E2 01 00`.

Подсказка: Чтобы проверить, как будет выглядеть память, можно создать указатель типа `char*` на эту память и распечатать каждый байт в виде шестнадцатеричного числа.
  ```