Семинар #7: Память и бинарные файлы. Домашнее задание.

Задача 1. Участок памяти

В следующей программе создан массив содержащий 16 байт.

```
#include <stdio.h>
int main()
{
    unsigned char a[16] = {0x00, 0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70,
                           0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0};
}
```

Произведите с этим массивом следующие операции:

- Напечатайте все байты через пробел. Используйте шестнадцатеричную систему счисления.
- Напечатайте все байты через пробел. Используйте десятичную систему счисления.
- Напечатайте все байты в следующем формате.

  `Index: 04  Hex: 0x40  ASCII: @`

  Если для байта нет печатаемого символа в ASCII, то на месте символа напечатайте символ `.` (точка).
- Присвойте байту с индексом 5 значение `0xE5`.
- Скопируйте первые семь байт на три байта правее. То есть нужно скопировать участок памяти с индексами от 0 до 6 в участок памяти с индексами от 3 до 9. Используйте `memmove`.
- Скопируйте последние пять байт в начало массива. Используйте `memcpy`.
- Занулите байты с идексами от 5 до 10. Используйте `memset`.

Задача 2. Просмотр памяти

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

- `int a = 0x11223344;`
- `int b = 1000000000;`
- `int c = 65535;`
- `int d = -1;`
- `int array[3] = {10, 2000, 65535};`
- `char str[9] = "Cat\nDog\n";`
- `float x = 1.0f;`

Память представить в виде последовательности 2-значных шестнадцатеричных чисел. Например число $123456_{10} = 1E240_{16}$ будет храниться в памяти как `40 E2 01 00`. Чтобы проверить, как будет выглядеть память, можно создать указатель типа `unsigned char*` на эту память и распечатать каждый байт в виде шестнадцатеричного числа. Решение этой задачи – это `.txt` файл, который будет содержать ответы на все подзадачи.

Задача 3. Строка

В программе создан массив из четырёх целых чисел типа `int`. В памяти этот массив занимает 16 байт. Интерпретируйте байты этого массива как строку и напечатайте её.

```
#include <stdio.h>
int main()
{
    int a[4] = {1819043144, 1461726319, 1684828783, 2593};
}
```

Решите эту задачу тремя способами:

(a) Используя приведение указателей.

(b) Используя `union`.

(c) Используя `memcpy`.

Задача 4. Узнать порядок байт:

Напишите функцию `int is_little_endian()` которая будет возвращать `1`, если эта функция запускается на системе с порядком байт Little Endian и `0`, если эта функция запускается на системе с порядком байт Big Endian.

Задача 5. Запись массива чисел в файл

Пусть у нас есть некоторый массив вещественных чисел:

```
int n = 1000;
double* array = (double*)malloc(n * sizeof(double));

for (int i = 0; i < n; ++i)
    p[i] = sin(M_PI * i / n);
```

Напишите следующие функции, которые будут сохранять этот массив в файл:

- `void save_numbers_text(const char* filename, double* array, int n)` – эта функция должна сохранять числа в файл в текстовом формате. Сначала эта функция должна записывать число `n`, а потом все числа с точностью в 15 знаков после запятой. Получившийся файл должен открываться с помощью текстового редактора.
- `void save_numbers_binary(const char* filename, double* array, int n)` – эта функция должна сохранять числа в бинарном формате. Сначала функция должна записывать количество чисел (в первые 4 байта файла), а потом она должна записывать сами числа, по 8 байт на одно число.

Используйте `xxd`, чтобы просмотреть содержимое файлов.

Задача 6. Последние символы

Написать программу, которая будет печатать 3 последних байта в файле. Программа должна принимать название файла из командной строки.

Задача 7. Размер файла

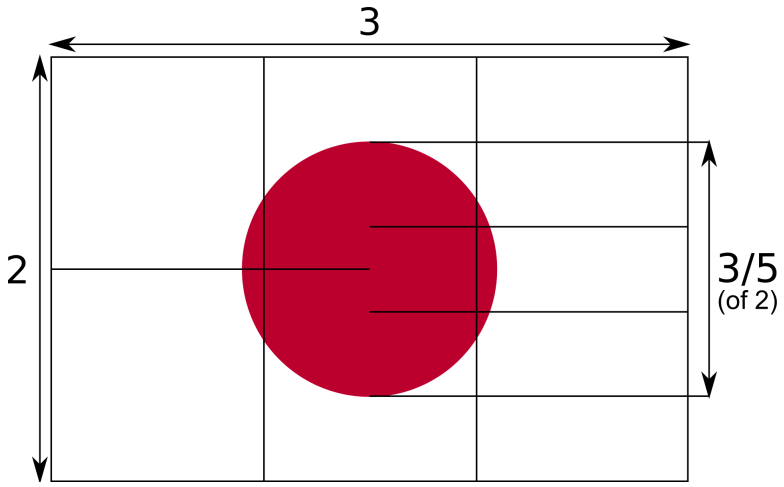
Написать программу, которая будет принимать название файла через аргумент командной строки и печатать его размер в байтах.
Подсказка: Используйте `fseek`, чтобы перейти в конец файла и `ftell`, чтобы узнать позицию.

Рисование в файл изображения

В дальнейших задачах вам понадобится программа для просмотра изображений в формате `ppm`. Если у вас нет программы, которая поддерживает этот формат на компьютере, то советую использовать IrfanView: www.irfanview.com. В этой программе по умолчанию используется сглаживание при приближении. Его можно отключить, чтобы было видно каждый пиксель `View -> Display Options -> Use Resample for zooming` (убрать галочку).

Задача 8. Флаг

В файле `flag.c` содержится пример работы создания изображения в формате `.ppm` в программе. Напишите программу, которая будет рисовать флаг Японии. Изображение должно иметь размер 600 на 400 пикселей. Компоненты белого цвета: `(255, 255, 255)`. Компоненты красного цвета: `(190, 0, 41)`.



Задача 9. Случайные круги

- Напишите функцию
  `void draw_circle(Color* data, int width, int height, int x0, int y0, int r, Color c)`
  которая будет рисовать круг на холсте `data` с центром в точке `(x0, y0)`, радиусом `r` и цветом `c`.
- Напишите программу, которая будет рисовать `n` кругов случайного цвета, расположения. Радиус тоже выбирается случайный в диапазоне от `a` до `b`. Параметры `n`, `a` и `b` передаются через аргументы командной строки. Программа должна создавать изображение `circles.ppm`.

Задача 10. Функция двух переменных

Напишите программу, которая будет рисовать значения функции двух переменных $f(x, y)$ в области $[-1, 1] \times [-1, 1]$.
Значения функции должны сохранятся в изображении размером 500 на 500 пикселей. К примеру, пиксель с координатами `(250, 250)` должен хранить значение функции в точке $(0, 0)$, а пиксель `(0, 400)` - значение в точке $(-1, 0.6)$.
Учтите, что значения пикселей изображения должны лежать в интервале от 0 до 255. Постройте изображения следующих функций:

1. $f(x, y) = k \cdot |x \cdot y|$
2. $f(x, y) = k \cdot |sin(10 \cdot (x^2 + y^2))|$
3. $f(x, y) = k \cdot |sin(5000 \cdot (x^2 + y^2))|$
4. $f(x, y) = k \cdot |cos(10x) \cdot sin(10y)|$
5. $f(x, y) = k\frac{1}{2} \cdot \left|sin\left(\frac{3}{0.1+|x|}\right) + sin\left(\frac{3}{0.1+|y|}\right)\right|$

Параметр $k = 255$ подбирается так, чтобы значения компонент цвета лежало в диапазоне от 0 до 255.

Обработка изображений

В файле `brightness.c` содержится программа, которая увеличивает яркость изображения. Используйте её как пример для решения следующих задач. Компиляция и запуск этой программы осуществляется следующим образом:

```
gcc -std=c99 -o brighter brightness.c
./brighter images/emir.ppm 50
```

Задача 11. Черно-белое изображение

Написать программу, которая принимает на вход файл изображения, считывает его и превращает в чёрно-белое изображение и записывает в файл `result.ppm`. Название изображения должно передаваться через аргументы командной строки.

Задача 12. Перестановка цветов:

Написать программу, которая переставляет местами красную и синюю компоненты цвета. Применить её на файле `emir.ppm`.

Задача 13. Сепия

Написать программу, которая будет применять к изображению эффект сепии.

Формулы для эффекта сепии:

$$r_{new} = 0.393 \cdot r + 0.769 \cdot g + 0.189 \cdot b$$
$$g_{new} = 0.349 \cdot r + 0.686 \cdot g + 0.168 \cdot b$$
$$b_{new} = 0.272 \cdot r + 0.534 \cdot g + 0.131 \cdot b$$

Если какое-то из этих значение станет большим, чем 255, то его нужно приравнять к 255.

Задача 14. Свёртка изображения.

Операция свёртки изображения задаётся следующей формулой:

$$data_{new}[i,j] = \sum_{p=-1}^{1} \sum_{q=-1}^{1} K[p+1, q+1] \cdot data[i+p, j+q]$$

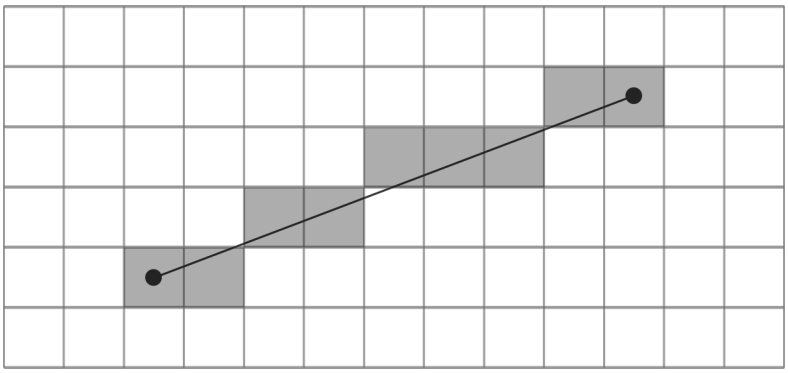
, где K - некоторая матрица 3 на 3. Для размытия эта матрица равна:

$$K = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Подробней о свёртке можно посмотреть тут: `www.youtube.com/watch?v=C_zFhWdM4ic`
Если провести эту операцию 1 раз, то размытие будет небольшое. Чтобы размыть изображение сильнее нужно повторить эту операцию несколько раз. Напишите программу, которая будет размывать изображение `n` раз (`n` передаётся через аргументы командной строки).

Рисование линий. Алгоритм Брезенхема:

Алгоритм построения прямой линии между двумя точками на двумерном холсте называется алгоритмом Брезенхема. В файле `4lines.c` есть реализация этого алгоритма на языке `C`.

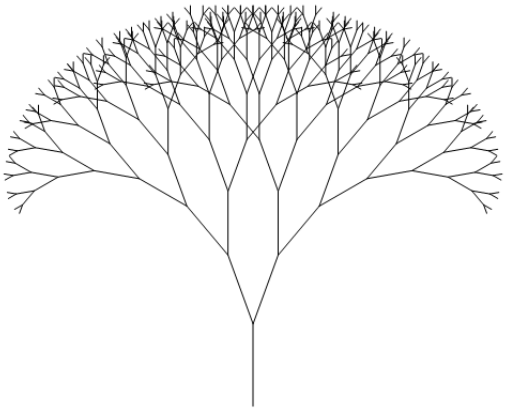


Задача 15. Случайные линии

Создать программу, которая будет рисовать `n` случайных отрезков случайного цвета. `n` должен передаваться через аргументы командной строки. Программа должна создавать файл `randlines.ppm`.

Задача 16. Фрактальное дерево:

Напишите рекурсивную функцию, которая будет рисовать фрактальное дерево:



Работа с изображениями формата `.jpg` с помощью библиотеки stb

Задача 17. Консольный графический редактор:

Объедините все решения предыдущих задач в одну программу `mge`. Выбор эффекта должен задаваться с помощью аргументов командной строки. Например так:

```
mge --sepia image.ppm result.ppm
```

Программа должна применять эффект сепии на изображение `image.ppm` и сохранять результат в `result.ppm`.

А при таком вызове программа должна применять эффект размытия 10 раз:

```
mge --blur 10 image.ppm result.ppm
```

Добавьте ещё опции: `--brighter`, `--bw`, `--changecolors`, `--mirror`.

Необязательные задачи (никак не учитываются)

Задача 1. Числа без указания количества

В файле `numbers.txt` хранятся некоторые целые числа (но не указано их количество). Напишите программу, которая будет считывать все числа из этого файла и печатать их на экран. Если в файле содержится какие-то другие символы кроме цифр и пробельных символов, то программа должна печатать `Error!` и завершаться. *Подсказка:* Для начала нужно узнать количество чисел. Это можно сделать, используя `fgetc`. Затем считываем. Память для чисел выделяем в куче, так как их количество изначально неизвестно и может быть болишим.