|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК6 «Защита информации»***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**«**Изучение различных алгоритмов сортировки.**»**

**Вариант 17**

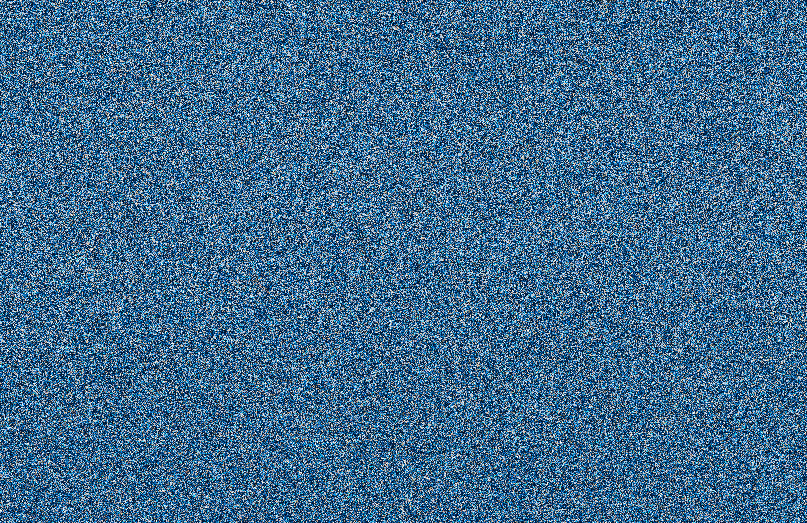
**ДИСЦИПЛИНА: «Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК6-51 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Щербаков В.И )  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Празян К.А )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга , 2021

Дано: bmp файл и связанный с ним txt файл. В текстовом файле каждая цифра соответствует своему пикселю в файле bmp. Цифры в текстовом файле перемешали и по текущему порядку нарисовали bmp файл.

Задача: восстановить bmp файл, отсортировав txt файл.



Картинка для задачи

Для решения данной задачи была выбрана Пирамидальная сортировка.

Алгоритм решения таков:  
1)читаем изначальный Bmp файл из него получаем размеры в пикселях картинки.

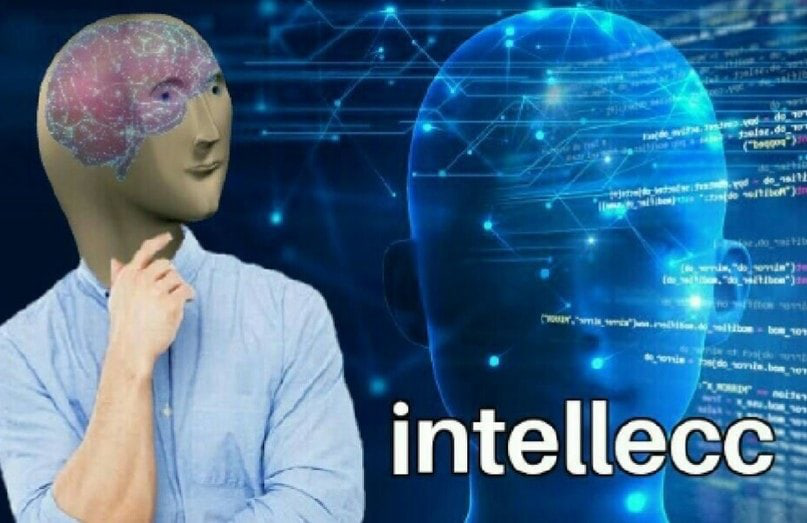
2)создаем одномерные массивы для цветов и для цифр размерностью равной площади картинки

3)читаем txt файл в уже созданный массив для цифр

4)читаем bmp файл в массивы цветов

5)сортируем с помощью пирамидальной сортировки по цифрам связанные с ними цвета, получаем таким образом изначальную картинку

6)записываем полученную картинку в bmp файл.



Результат работы алгоритма

#include "Bmp.h"

using namespace std;

void bmp\_read\_pixels(int size, int\* bleu, int\* green, int\* red, int\* alfa) {//читаем файл bmp и пишем в массивы значения цветов для каждого индекса

BMP pict("C:\\Users\\namel\\source\\repos\\третья лаба\\третья лаба\\17e.bmp");//открываем bmp файл

int channels = pict.bmp\_info\_header.bit\_count / 8;

for (int i = 0; i < size; i++) {//записываем значения цветов в соответсвующие массивы

bleu[i] = pict.data[channels \* i + 0];

green[i] = pict.data[channels \* i + 1];

red[i] = pict.data[channels \* i + 2];

if (channels == 4) {

alfa[i] = pict.data[channels \* i + 3];

}

}

}

void readtxt(int size,int map[]) {//читаем txt файл

string line;

ifstream f;

f.open("17.txt");//открываем файл

if (f.is\_open())

{

for (int i = 0; i < size; i++)

{

f >> map[i];//записываем в массив

}

f.close();

}

else {

cout << "eror" << endl;

}

}

void heapify(int mas\_to\_sort[], int n, int i, int\* bleu, int\* green, int\* red, int\* alfa)

{

int largest = i;

// Инициализируем наибольший элемент как корень

int l = 2 \* i + 1; // левый

int r = 2 \* i + 2; // правый

// Если левый или правый дочерний элемент больше, чем самый большой элемент на данный момент

if (l < n && mas\_to\_sort[l] > mas\_to\_sort[largest])

largest = l;

if (r < n && mas\_to\_sort[r] > mas\_to\_sort[largest])

largest = r;

// Если самый большой элемент не корень

if (largest != i)

{

swap(mas\_to\_sort[i], mas\_to\_sort[largest]);

swap(bleu[i], bleu[largest]);

swap(green[i], green[largest]);

swap(red[i], red[largest]);

swap(alfa[i], alfa[largest]);

// Рекурсивно преобразуем в двоичную кучу затронутое поддерево

heapify(mas\_to\_sort, n, largest,bleu,green,red,alfa);

}

}

//функция выполняющяя пирамидальную сортировку

void heapSort(int\* input, int n, int\* bleu, int\* green, int\* red, int\* alfa)

{

// Построение кучи (перегруппируем массив)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(input, n, i, bleu, green, red, alfa);

// Один за другим извлекаем элементы из кучи

for (int i = n - 1; i >= 0; i--)

{

// Перемещаем текущий корень в конец

swap(input[0], input[i]);

swap(bleu[0], bleu[i]);

swap(green[0], green[i]);

swap(red[0], red[i]);

swap(alfa[0], alfa[i]);

// вызываем процедуру heapify на уменьшенной куче

heapify(input, i, 0, bleu, green, red, alfa);

}

}

void write\_bmp(int x, int y, int\* bleu, int\* green, int\* red, int\* alfa) {

BMP outpict(x, y);

for (int i = 0; i < x; i++) {

for (int j = 0; j < y; j++) {

int b = bleu[i \* y + j];

int g = green[i \* y + j];

int r = red[i \* y + j];

//int a = alfa[i \* x + j];

//cout << b << " " << g << " " << r << " " << " " << endl;

outpict.set\_pixel(i,j, b, g, r, 255);

}

}

outpict.write("result picture2.bmp");

}

int main() {

int x, y, size;

BMP pict("17e.bmp");//открываем файл bmp

x = pict.bmp\_info\_header.width;//получаем ширину файла bmp

y = pict.bmp\_info\_header.height;//получаем высоту файлаbmp

size = x \* y;

//создаём массивы цветов для палитры bmp где на каждый индекс приходится своё значение в нужных цветах

int\* bleu = new int[size];

int\* green = new int[size];

int\* red = new int[size];

int\* alfa = new int[size];

//создаем массив для цифр

int\* input = new int[size];

//читаем txt файл

readtxt(size,input);

//читаем bmp файл

bmp\_read\_pixels(size, bleu, green, red, alfa);

//сортируем

heapSort(input, size, bleu, green, red, alfa);

//создаем новый файл и пишем в него попиксельно цвета

write\_bmp(y,x, bleu, green, red, alfa);

return 0;

}

//известные баги итоговую картинку нужно будет повернуть на 90 градусов

//bmp.h был взят с гитхаба другого человека https://github.com/sol-prog/cpp-bmp-images/blob/master/BMP.h

Вывод: в данной лабораторной работе я понял как можно получить доступ к цветам bmp файла с помощью файла bmp.h в с++ а так же как можно восстановить картинку по цифрам из txt файла.