МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Институт №3 – «Системы управления, информатика и электроэнергетика»

Кафедра 307 – «Цифровые технологии и информационные системы»

Отчёт по практическому заданию №6

«Изображения»

Выполнил:

Студент группы М3О-214Б-20

**Цатурьян Константин Артурович**

Принял:

Преподаватель

**Павлов Олег Владимирович**

Москва 2021

## Цель работы

Разработать программу для сравнения двух одинаковых по разрешению картинок в формате bmp в градациях серого посредством подсчета расстояний между ними.

## Описание алгоритма

Рассмотрим выполнения алгоритма для пары картинок. Программа открывает bmp файлы двух картинок, находящихся на компьютере. Далее программа обрабатывает информацию из заголовка bmp файла, откуда получает размеры картинок. Происходит считывание байтов, количество которых определяется размером картинки. При вычислении расстояния между картинками вычисляется значение равное корню из сумм квадратов разностей значений соответствующих байт.

Итого, в цикле алгоритм считывает значения i-х пикселей каждой картинки, вычисляет между их значениями разность и возводит её в квадрат. Суммирует данные значения и после цикла извлекает из неё квадратный корень. Данная операция схожа с поиском расстояния между точками в N-мерном пространстве, где N – разрешение картинки.

Чем точки ближе, тем меньше между ними расстояние, значит у самых похожих картинок будут наименьшие расстояния.

Для удобства обработки было решено использовать класс Image, который внутри себя будет содержать высоту и ширину картинки, а также массив со значениями пикселей картинки. Для вычисления расстояния используется перегруженный оператор сложения. Также в ходе отладки выяснилось, что значение расстояния получается слишком большим, поэтому было решено извлечь из него корень ещё один раз.

Блок-схема программы представлена на рисунке 1.



Рисунок – Блок-схема программы

## Сравнение яблок

На основе написанной программы были вычислены расстояния между данными картинками яблок.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Яблоко1 | Яблоко2 | Яблоко3 | Яблоко4 |
| Яблоко1 | 0 | 181,055 | 104,397 | 86,384 |
| Яблоко2 | 181,055 | 0 | 177,119 | 178,125 |
| Яблоко3 | 104,397 | 177,119 | 0 | 89,289 |
| Яблоко4 | 86,384 | 178,125 | 89,289 | 0 |

Исходя из полученных расстояний, можно утверждать, что картинки номер 1 и номер 4 имеют наименьшее расстояние, а следовательно, больше похожи друг на друга.

## Сравнение выбранных картинок

Для оценки результата работы программы на другой выборке удобно использовать логотипы компаний, ведь они часто незначительно меняются:

 

|  |  |
| --- | --- |
| *Рисунок 2 – Логотип 1* | *Рисунок 3 – Логотип 2* |



|  |  |
| --- | --- |
| *Рисунок 4 – Логотип 3* | *Рисунок 5 – Логотип 4* |

Результат сравнения для логотипов FireFox:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Логотип1 | Логотип2 | Логотип3 | Логотип4 |
| Логотип1 | 0 | 138,607 | 200,79 | 146,984 |
| Логотип2 | 138,607 | 0 | 201,646 | 152,146 |
| Логотип3 | 200,79 | 201,646 | 0 | 189,538 |
| Логотип4 | 146,984 | 152,146 | 189,538 | 0 |

Итого – картинки 1 и 2 совпадают больше всего.

## Вывод

Была создана программа, выполняющая сравнение картинок на основе поиска расстояния между их соответствующими пикселями.

## Исходный код

1. #include <iostream>
2. #include <fstream>
3. #include <string>
4. using namespace std;
6. class Image
7. {
8. public:
9. fstream img\_src;
10. char\* img\_array;
11. int width, height;
12. static int img\_num;
14. Image()
15. {
16. string filename = "apple\_" + to\_string(img\_num) + ".bmp";
17. img\_src.open(filename);
18. if (!img\_src.is\_open())
19. {
20. cout << "Critical error while opening the image " <<filename << endl;
21. //return;
22. }
23. else cout << "Opened " << filename << endl;
24. img\_num++;
26. char\* info = new char[54];
27. img\_src.read(info, 54);
29. width = \*(int\*)&info[18];
30. height = \*(int\*)&info[22];
32. img\_array = new char[3 \* width \* height];
33. img\_src.read(img\_array, 3 \* width \* height);
34. }
36. ~Image()
37. {
38. delete[] img\_array;
39. img\_src.close();
40. }
41. };
43. int Image::img\_num = 1;
45. double operator+(const Image& im1, const Image& im2)
46. {
47. unsigned long int sq = 0;
48. for (int i = 0; i < 3 \* im1.width \* im1.height; i++)
49. sq += (im1.img\_array[i] - im2.img\_array[i]) \* (im1.img\_array[i] - im2.img\_array[i]);
51. return sqrt(sqrt(sq));
52. }
54. int main()
55. {
56. Image a1, a2, a3, a4;
58. cout << endl;
59. cout << "Image 1 and 1 match = " << a1 + a1 << endl;
60. cout << "Image 2 and 2 match = " << a2 + a2 << endl;
61. cout << "Image 3 and 3 match = " << a3 + a3 << endl;
62. cout << "Image 4 and 4 match = " << a4 + a4 << endl << endl;
64. cout << "Image 1 and 2 match = " << a1 + a2 << endl;
65. cout << "Image 1 and 3 match = " << a1 + a3 << endl;
66. cout << "Image 1 and 4 match = " << a1 + a4 << endl << endl;
68. cout << "Image 2 and 3 match = " << a2 + a3 << endl;
69. cout << "Image 2 and 4 match = " << a2 + a4 << endl << endl;
71. cout << "Image 3 and 4 match = " << a3 + a4 << endl;
72. }