**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MO ЭВМ**

Курсовая РАБОТА

**по дисциплине «ПРОГРАММИРОВАНИЕ»**

Тема: BMP - файлы

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Малышенко Ю.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Малышенко Ю.И. | | |
| Группа 6303 | | |
| Тема работы: BMP-файл | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание структур BMP-файла * Примеры работы программы * Заключение * Список использованных источников * Приложение: Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 10страниц. | | |
| Дата выдачи задания: 28.04.2017 | | |
| Дата сдачи реферата: 2.06.2017 | | |
| Дата защиты реферата: 2.06.2017 | | |
| Студент |  | Малышенко Ю.И. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В результате курсовой работы была реализована программа на языке программирования С, которая находит самый большой белый прямоугольник в BMP-файле и выводит координаты левого верхнего и правого нижнего его углов.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | ОПИСАНИЕ СТРУКТУР BMP-файла | 6 |
| 1.1. | Общее строение | 6 |
| 1.2.  1.3  1.4 | BITMAPFILEHEADER  BITMAPINFOHEADER  RGBTRIPLE | 6  6  7 |
| 2. | Ход работы | 8 |
| 2.1. | Функция squad | 8 |
| 2.2. | Функция search | 9 |
| 2.3.  2.4. | Функция readFile\_  Функция main | 10  11 |
| 3. | ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ | 12 |
| 4. | Заключение | 13 |
|  | Список использованных источников | 14 |
|  | Приложение: Исходный код | 15 |
|  |  |  |

**введение**

* Требуется написать программу, которая находит самый большой белый прямоугольник в BMP-файле и выводит координаты левого верхнего и правого нижнего его углов.
* Программа получает параметры из входного потока и должна проверить их корректность. Параметр:
* input\_file - имя BMP файла
* В случае, если программа получила некорректный параметр, то:
* не создается выходного в файла
* выводится сообщение об ошибке “Fail”.

### **Общие сведения**

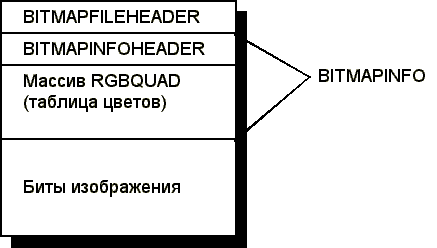
* 24 бита на цвет
* без сжатия
* файл всегда соответствует формату (проверять не нужно)
* обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
* обратите внимание на порядок записи пикселей
* все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

**1. ОПИСАНИЕ СТРУКТУР BMP-файла**

**1.1. Общее строение BMP-файла**

**BMP** (от англ. *Bitmap Picture*) — формат хранения растровых изображений,  разработанный компанией Microsoft.

Для начала приведем графическое представление данных в bmp



**1.2. BITMAPFILEHEADER**

В начале стoит заголовок файла (BITMAPFILEHEADER).

Он описан следующим образом:

typedef struct

{

unsigned short bfType; // Тип файла (должен быть BM)

unsigned int bfSize; // Размер файла

unsigned short bfReserved1; // Зарезервирован (должен быть 0)

unsigned short bfReserved2; // Зарезервирован (должен быть 0)

unsigned int bfOffBits; // Смещение от начала битов

} BITMAPFILEHEADER;

**1.3. BITMAPINFOHEADER**

Дальше идет структура BITMAPINFOHEADER, которая объявлена так:

 typedef struct

{

unsigned int biSize; // Число байтов необходимое для структуры

unsigned int biWidth; // Ширина изображения (в пикселях)

unsigned int biHeight; // Высота изображения (в пикселях)

unsigned short biPlanes; // Число плоскостей (должен быть 1)

unsigned short biBitCount; // Число битов на пиксель (24 бита)

unsigned int biCompression; // Тип сжатия

unsigned int biSizeImage; // Размер изображения (в байтах)

int biXPelsPerMeter; // Разрешающая способность по горизонтали

int biYPelsPerMeter; // Разрешающая способность по вертикали

unsigned int biClrUsed; // Количество используемых цветов

unsigned int biClrImportant; // Количество существенных цветов

} BITMAPINFOHEADER;

**1.4. RGBTRIPLE**

biBitCount = 24 это самый простой формат. Здесь 3 байта определяют 3 компоненты цвета. То есть по компоненте на байт. Просто читаем по структуре RGBTRIPLE и используем его поля rgbtBlue, rgbtGreen, rgbtRed. Они идут именно в таком порядке.

Структура выглядит так:

typedef struct

{

unsigned char rgbBlue; // Синий цвет (от 0 до 255)

unsigned char rgbGreen; // Зеленый цвет (от 0 до 255)

unsigned char rgbRed; // Красный цвет (от 0 до 255)

} RGBTRIPLE;

**2. Ход работы**

**2.1. Функция** squad

Функция squad, находит максимальную прямоугольную площадь в заданной гистограмме и заполняет структуру coordinates, полученными в результате обработки данными.

coordinates squad(int\* histogram, int length)

{

coordinates m\_Rectang;

int maxsquad = 0;

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

{

int lenghtYSize = histogram[i];

for (int j = i; j < length; j++)

{

if (lenghtYSize > histogram[j])

{

lenghtYSize = histogram[j];

}

if ((j - i + 1) \* lenghtYSize >= maxsquad)

{

maxsquad = (j - i + 1) \* lenghtYSize;

m\_Rectang.x0 = i;

m\_Rectang.x1 = j;

m\_Rectang.y1 = lenghtYSize;

}

}

}

m\_Rectang.squad = maxsquad;

return m\_Rectang;

}

**2.2. Функция** **search**

Функция search ищет самый большой белый прямоугольник и выводит координаты левого верхнего угла и правого нижнего.

Для реализации поиска введем вспомогательный одномерный массив соразмерный с длиной данного изображением, в ячейках которого будем хранить 1, если это белый пиксель текущий строки и 0, если не белый.

Идея алгоритма состоит в следующем: последовательно для каждой строки изображения создается гистограмма на основе одномерного массива такой же длины, что и изображение.

Если в i-той строке на j-той позиции встречается 0, тогда i,j элемент гистограммы – 0. Если 1, то прибавляем ее к текущему значению i,j элемента гистограммы.

Получаем гистограмму, в которой необходимо найти наибольшую прямоугольную площадь. Для этого реализуем функцию squad, которая будет возвращать структуру coordinates, в которой будут хранится параметры данной области

void search(RGBTRIPLE\* rgb, int width, int height)

{

int\* addArrayForRGB = (int\*)malloc(sizeof(int\*)\*width);

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

{

if (rgb[i].rgbtBlue == 255 && rgb[i].rgbtGreen == 255

&& rgb[i].rgbtRed == 255)

addArrayForRGB[i] = 1;

else

addArrayForRGB[i] = 0;

}

coordinates m\_Rectang = squad(addArrayForRGB, width);

int result = m\_Rectang.squad;

for (int i = 1; i < height; i++)

{

for (int j = 0; j < width; j++)

if (rgb[j+i\*width].rgbtBlue == 255 && rgb[j + i\*width].rgbtGreen == 255 && rgb[j + i\*width].rgbtRed == 255)

addArrayForRGB[j]++;

else addArrayForRGB[j] = 0;

coordinates newRectang = squad(addArrayForRGB, width);

if (newRectang.squad >= result)

{

m\_Rectang = newRectang;

m\_Rectang.y0 = i;

result = m\_Rectang.squad;

}

}

m\_Rectang.y0 = height - m\_Rectang.y0 - 1;

m\_Rectang.y1 = m\_Rectang.y1 + m\_Rectang.y0;

printf("X0:%d\nY0:%d\nX1:%d\nY1:%d\n", m\_Rectang.x0, height - m\_Rectang.y0 , m\_Rectang.x1, height - m\_Rectang.y1);

free(addArrayForRGB);

}

**2.3. Функция readFile\_**

int readFile\_(char \*bmp)

{

BITMAPFILEHEADER BFH;

BITMAPINFOHEADER BIH;

RGBTRIPLE rgb;

//инициализация структур

FILE \*f = fopen(bmp, "rb");

//открытие файла

if (f == NULL)

{

printf("Fail\n");

return 0;

}

//проверка существования файла

fread(&BFH, 1, sizeof(BFH), f);

fread(&BIH, 1, sizeof(BIH), f);

//считывание заголовков

unsigned int Height = BIH.biHeight;

unsigned int Width = BIH.biWidth;

//инициализация длины и ширины

RGBTRIPLE\* rgbArray = (RGBTRIPLE\*)malloc(Height \* Width \* sizeof(RGBTRIPLE));

int extraBytes = 4 - ((Width \* 3) % 4);

if (extraBytes == 4)

{

extraBytes = 0;

} // Так как количество байт в строке должно быть кратно 4, создаем переменную extraBytes, равную количеству байт, которые необходимо заполнить нулями для выравнивания.

for (unsigned long i = 0; i< Height \* Width; i++)

{

fread(&rgbArray[i].rgbtBlue, 1, 1, f);

fread(&rgbArray[i].rgbtGreen, 1, 1, f);

fread(&rgbArray[i].rgbtRed, 1, 1, f);

if (extraBytes)

{

for (int j = 0; j < extraBytes; j++)

{

char zeroByte;

fread(&zeroByte, 1, 1, f);

}

}

}

search(rgbArray, Width, Height);

fclose(f);

//закрытие файлов

return 0;

}

**2.4. Функция main**

int main()

{

char bmp[15];

printf("Enter the file name ");

fgets(bmp, 15, stdin);

// Считывание строки команд

bmp[strlen(bmp) - 1] = '\0';

int read = readFile\_(bmp);

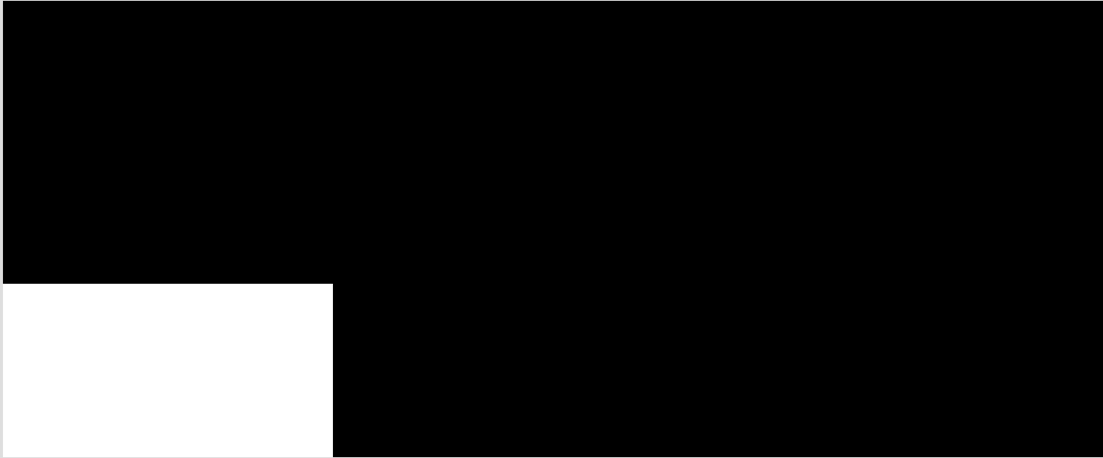
return 0;

}

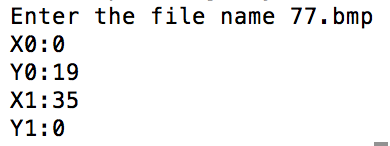
**3.ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ**

**ПРИМЕР 1.**

Исходный файл



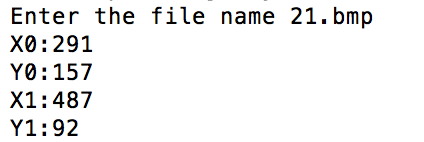
**Результат**

****

**ПРИМЕР 2.**



**Результат**

****

**ЗАКЛЮЧЕНИЕ**

В результате курсовой работы была реализована программа на языке программирования С, которая находит самый большой белый прямоугольник в BMP-файле и выводит координаты левого верхнего и правого нижнего его углов.

**список использованных источников**

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство"Невский Диалект", 2001. 352 с.

2. UNIX. Программное окружение / Керниган Б., Пайк Р. СПб: Символ Плюс, 2003. 416с.

**приложение А**

**Исходный код программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#pragma pack(push, 2)

typedef struct

{

unsigned short bfType;

unsigned int bfSize;

unsigned short bfReserved1;

unsigned short bfReserved2;

unsigned int bfOffBits;

} BITMAPFILEHEADER;

typedef struct

{

unsigned int biSize;

unsigned int biWidth;

unsigned int biHeight;

unsigned short biPlanes;

unsigned short biBitCount;

unsigned int biCompression;

unsigned int biSizeImage;

int biXPelsPerMeter;

int biYPelsPerMeter;

unsigned int biClrUsed;

unsigned int biClrImportant;

} BITMAPINFOHEADER;

typedef struct

{

unsigned char rgbtBlue;

unsigned char rgbtGreen;

unsigned char rgbtRed;

} RGBTRIPLE;

#pragma pack(pop)

typedef struct

{

int x0;

int y0;

int x1;

int y1;

int squad;

} coordinates;

coordinates squad(int\* histogram, int length)

{

coordinates m\_Rectang;

int maxsquad = 0;

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

{

int lenghtYSize = histogram[i];

for (int j = i; j < length; j++)

{

if (lenghtYSize > histogram[j])

{

lenghtYSize = histogram[j];

}

if ((j - i + 1) \* lenghtYSize >= maxsquad)

{

maxsquad = (j - i + 1) \* lenghtYSize;

m\_Rectang.x0 = i;

m\_Rectang.x1 = j;

m\_Rectang.y1 = lenghtYSize;

}

}

}

m\_Rectang.squad = maxsquad;

return m\_Rectang;

}

void search(RGBTRIPLE\* rgb, int width, int height)

{

int\* addArrayForRGB = (int\*)malloc(sizeof(int\*)\*width);

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

{

if (rgb[i].rgbtBlue == 255 && rgb[i].rgbtGreen == 255

&& rgb[i].rgbtRed == 255)

addArrayForRGB[i] = 1;

else

addArrayForRGB[i] = 0;

}

coordinates m\_Rectang = squad(addArrayForRGB, width);

int result = m\_Rectang.squad;

for (int i = 1; i < height; i++)

{

for (int j = 0; j < width; j++)

if (rgb[j+i\*width].rgbtBlue == 255 && rgb[j + i\*width].rgbtGreen == 255 && rgb[j + i\*width].rgbtRed == 255)

addArrayForRGB[j]++;

else addArrayForRGB[j] = 0;

coordinates newRectang = squad(addArrayForRGB, width);

if (newRectang.squad >= result)

{

m\_Rectang = newRectang;

m\_Rectang.y0 = i;

result = m\_Rectang.squad;

}

}

m\_Rectang.y0 = height - m\_Rectang.y0 - 1;

m\_Rectang.y1 = m\_Rectang.y1 + m\_Rectang.y0;

printf("X0:%d\nY0:%d\nX1:%d\nY1:%d\n", m\_Rectang.x0, height - m\_Rectang.y0 , m\_Rectang.x1, height - m\_Rectang.y1);

free(addArrayForRGB);

}

int readFile\_(char \*bmp)

{

BITMAPFILEHEADER BFH;

BITMAPINFOHEADER BIH;

RGBTRIPLE rgb;

FILE \*f = fopen(bmp, "rb");

if (f == NULL)

{

printf("Fail\n");

return 0;

}

fread(&BFH, 1, sizeof(BFH), f);

fread(&BIH, 1, sizeof(BIH), f);

unsigned int Height = BIH.biHeight;

unsigned int Width = BIH.biWidth;

RGBTRIPLE\* rgbArray = (RGBTRIPLE\*)malloc(Height \* Width \* sizeof(RGBTRIPLE));

int extraBytes = 4 - ((Width \* 3) % 4);

if (extraBytes == 4)

{

extraBytes = 0;

}

for (unsigned long i = 0; i< Height \* Width; i++)

{

fread(&rgbArray[i].rgbtBlue, 1, 1, f);

fread(&rgbArray[i].rgbtGreen, 1, 1, f);

fread(&rgbArray[i].rgbtRed, 1, 1, f);

if (extraBytes)

{

for (int j = 0; j < extraBytes; j++)

{

char zeroByte;

fread(&zeroByte, 1, 1, f);

}

}

}

search(rgbArray, Width, Height);

fclose(f);

return 0;

}

int main()

{

char bmp[15];

printf("Enter the file name ");

fgets(bmp, 15, stdin);

bmp[strlen(bmp) - 1] = '\0';

int read = readFile\_(bmp);

return 0;

}