**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Работа с BMP-файлами

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Доброхвалов М.О. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Доброхвалов М.О. | | |
| Группа 6303 | | |
| Тема работы: работа с BMP-файлами | | |
| Исходные данные:  Требуется написать программу на языке С, которая отражает заданную область BMP-файла по вертикали (симметрия относительно вертикальной оси -  Я|R). Результат сохраняет в новом файле. | | |
| Содержание пояснительной записки:  Введение, содержание, реализация программы, описание используемых функций, примеры работы программы, заключение, список используемых источников. | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 20.04.2017 | | |
| Дата сдачи реферата: 02.06.2017 | | |
| Дата защиты реферата: 02.06.2017 | | |
| Студент |  | Доброхвалов М.О. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

Содержание курсовой работы состоит в разработке программы для работы с BMP-файлами. Реализованная программа написана на языке программирования C. Результатом курсовой работы является готовая программа, способная отражать заданную область BMP-файла по вертикали, а затем сохранять изменения в новом файле. Демонстрация работы приведена в виде скриншотов.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Технические требования | 5 |
| 1.1. | Параметры | 5 |
| 1.2. | Общие сведения | 6 |
|  |  |  |
| 2. | Реализация программы | 7 |
| 2.1. | Используемое ПО | 7 |
| 2.2. | Описание используемых функций | 7 |
| 2.3. | Примеры работы программы | 9 |
|  | Заключение | 0 |
|  | Список использованных источников | 0 |
|  | Приложение А. Исходный код программы | 0 |

**введение**

**Целью** является изучение принципов работы с файлами в языке программирования C и создание программы для обработки BMP-файла (отражения заданной области по вертикали).

**Задачи**:

* Изучение материалов о файлах типа BMP
* Разработка основных функций программы
* Сборка и отладка
* Тестирование

**1. Технические требования**

**1.1. Параметры**

* Программа получает параметры из входного потока и должна проверить их корректность. Параметры:
* input\_file
* x0
* y0
* x1
* y1
* input\_file - имя BMP файла
* x0 y0 левый верхний угол области (отсчет с точки 0, 0)
* x1 y1 правый нижний угол области
* В случае, если программа получила некорректные параметры, то:
* не создается выходного в файла
* выводится сообщение об ошибке “Fail with <имя параметра>”

**1.2. 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).

**2. Реализация программы**

**2.1. Используемое программное обеспечение**

* Компилятор: g++
* Среда разработки: Sublime Text 3

**2.2. Описание используемых функций**

1) Функция main

В функции main: выделяется память под чтение потока ввода и название файла, поток ввода разбивается на название файла и координаты вершин, выполняется открытие бинарного файла на чтение и проверка на то, открылся ли файл.

Объявление:

int main().

Тип возвращаемого значения: int.

Функция не имеет аргументов.

При корректном завершении программы функция возвращает 0.

2) Функция newFile

Функция выполняет чтение заголовка BMP-файла и информации о нем. Затем определяется количество байт на один пиксель и количество байт, необходимых для выравнивания. После проверки координат бинарный файл открывается на запись. В новый BMP-файл записывается заголовок и информация о файле. Затем в цикле считывается строка файла, вызывается функция reflection и новая строка записывается в новый файл. Очищается память, выделенная под строку, и открытый файл закрывается.

Объявление:

int newFile(File f, BMP\_HEADER header, BMP\_INFO information) .

Тип возвращаемого значения: int.

Аргументы:

* File f – новый файл
* BMP\_HEADER header - заголовок
* BMP\_INFO information - информация о файле

При корректном выполнении функции возвращается 0. Если введенные координаты превышают размер файла, либо не удалось открыть файл на запись, то возвращаемое значение – 1..5.

3) Функция reflection

Заводится массив для временного хранения одного пикселя при обмене. В цикле по три элемента меняются местами (происходит отражение строки). После завершения работы цикла память, выделенная под временное хранение очищается. Измененная строка возвращается в функцию newFile

Объявление:

char \*reflection(char \*string, int x0, int x1) .

Тип возвращаемого значения: char \*.

Аргументы:

* char \*string – строка
* int x0 – начальная координата
* int x1 – конечная координата

Функция возвращает строку string типа char \*.

**2.3. Примеры работы программы**

|  |  |
| --- | --- |
| Входной поток и файл(если он существует) | Результат работы программы |
| ./coursework  2.bmp 20 20 200 150 |  |
| ./coursework  a.bmp 1 1 2 2 |  |
| ./coursework  2.bmp 1 1 1000 50 |  |
| ./coursework  2.bmp 1 1 1000 50 |  |

**заключение**

В ходе выполнения курсовой работы были получены навыки работы с BMP-файлами. Поставленные задачи полностью выполнены: программа для чтения, отражения и записи BMP-файла разработана, отлажена и протестирована. Таким образом, заданная цель работы полностью выполнена.

**список использованных источников**

1. Евченко А. И. OpenGL и DirectX. Программирование графики (Для профессионалов), 2006 г.
2. <http://www.color.org/specification/ICC1v43_2010-12.pdf>
3. MSDN [Bitmap Header Types](http://msdn.microsoft.com/en-us/library/dd183386(VS.85).aspx)
4. MSDN [BITMAPINFO Structure](http://msdn.microsoft.com/en-us/library/dd183375(VS.85).aspx)
5. ["JPEG and PNG Extensions for Specific Bitmap Functions and Structures"](http://msdn.microsoft.com/en-us/library/dd145023(VS.85).aspx)
6. MSDN: [Windows CE - BITMAPINFOHEADER Structure](http://msdn.microsoft.com/en-us/library/ms959648.aspx)

**приложение А**

**Исходный код программы**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #define BITTOBYTE 8
5. #define WORD 4
6. #define FAILX0 1
7. #define FAILY0 2
8. #define FAILX1 3
9. #define FAILY1 4
10. #define FAILWRITE 5
11. typedef union BMP\_HEADER
12. {
13. char data[14];
14. struct
15. {
16. unsigned short signature;
17. unsigned int size;
18. unsigned int reserved;
19. unsigned int offset;
20. }hdr;
21. }BMP\_HEADER;
22. typedef union BMP\_INFO
23. {
24. char data1[40];
25. struct
26. {
27. unsigned int structSize;
28. unsigned int width;
29. unsigned short height;
30. unsigned short following;
31. unsigned short planes;
32. unsigned short count;
33. unsigned int compression;
34. char asd[28];
35. }info;
36. }BMP\_INFO;
37. typedef struct File
38. {
39. FILE \*oldf;
40. int x0;
41. int y0;
42. int x1;
43. int y1;
44. }File;
45. typedef struct StructOfDimensions
46. {
47. int width;
48. int height;
49. int size\_string;
50. }StructOfDimensions;
51. int newFile(File , BMP\_HEADER, BMP\_INFO);
52. char \*reflection(char \*, int, int);
53. int main()
54. {
55. File f;
56. f.oldf = NULL;
57. BMP\_HEADER header;
58. BMP\_INFO information;
59. char \*str = (char \*)malloc(200);
60. char input\_file[50];
61. int x0=-1, x1=-1, y0=-1, y1=-1;
62. fgets(str, 200, stdin);
63. strcpy(input\_file, strtok(str, " "));
64. f.x0 = atoi(strtok(NULL, " "));
65. f.y0 = atoi(strtok(NULL, " "));
66. f.x1 = atoi(strtok(NULL, " "));
67. f.y1 = atoi(strtok(NULL, " "));
68. f.oldf = fopen(input\_file, "rb");
69. if (f.oldf == NULL)
70. printf("Fail with input\_file\n");
71. else
72. {
73. if (f.x0 > 0)
74. {
75. if (f.y0 > 0)
76. {
77. if (f.x1 > 0 && f.x1>f.x0)
78. {
79. if (f.y1 > 0 && f.y1>f.y0)
80. {
81. int log = 0;
82. log = newFile(f, header, information);
83. if ( log == FAILX0 ) printf("Fail with x0");
84. if ( log == FAILY0 ) printf("Fail with y0");
85. if ( log == FAILX1 ) printf("Fail with x1");
86. if ( log == FAILY1 ) printf("Fail with y1");
87. if ( log == FAILWRITE) printf("Fail with write file\n");
88. fclose(f.oldf);
89. }
90. else
91. printf("Fail with y1\n");
92. }
93. else
94. printf("Fail with x1\n");
95. }
96. else
97. printf("Fail with y0\n");
98. }
99. else
100. printf("Fail with x0\n");
101. }
102. free(str);
103. return 0;
104. }
105. int newFile(File f, BMP\_HEADER header, BMP\_INFO information)
106. {
107. StructOfDimensions Dimensions;
108. fread(header.data, sizeof(char), sizeof(header.data), f.oldf);
109. fread(information.data1, sizeof(char), sizeof(information.data1), f.oldf);
110. Dimensions.width = information.info.width;
111. Dimensions.height = information.info.height;
112. int count = information.info.count/BITTOBYTE;
113. int trash = Dimensions.width\*count % WORD;
114. Dimensions.size\_string = Dimensions.width\*count;
115. if ( trash ) Dimensions.size\_string += (4-trash);
116. if (f.x0 > Dimensions.width) return FAILX0;
117. if (f.y0 > Dimensions.height) return FAILY0;
118. if (f.x1 > Dimensions.width) return FAILX1;
119. if (f.y1 > Dimensions.height) return FAILY1;
120. FILE \*newf = fopen("new.bmp", "w");
121. if (newf == NULL) return FAILWRITE;
122. else
123. {
124. fwrite(header.data, sizeof(char), sizeof(header.data), newf);
125. fwrite(information.data1, sizeof(char), sizeof(information.data1), newf);
126. char \*string = (char \*)malloc(Dimensions.size\_string);
127. for (int i=0; i<Dimensions.height; i++)
128. {
129. fread(string, sizeof(char), Dimensions.size\_string, f.oldf);
130. if ( (Dimensions.height-i)>=f.y0 && (Dimensions.height-i)<=f.y1 )
131. string = reflection(string, f.x0, f.x1);
132. fwrite(string, sizeof(char), Dimensions.size\_string, newf);
133. }
134. free(string);
135. fclose(newf);
136. }
137. return 0;
138. }
139. char \*reflection(char \*string, int x0, int x1)
140. {
141. char \*temp = (char \*)malloc(3);
142. for(int i=0; i < (x1-x0+1)/2; i++)
143. {
144. int j = x0+i-1;
145. int k = x1-(i+1);
146. temp[0] = string[3\*j];
147. temp[1] = string[3\*j+1];
148. temp[2] = string[3\*j+2];
149. string[3\*j] = string[3\*k];
150. string[3\*j+1] = string[3\*k+1];
151. string[3\*j+2] = string[3\*k+2];
152. string[3\*k] = temp[0];
153. string[3\*k+1] = temp[1];
154. string[3\*k+2] = temp[2];
155. }
156. free(temp);
157. return string;
158. }