МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

Курсовая работа
по дисциплине «Программирование»
Тема: Обработка BMP-файла.

Студент гр. 6304 _____ Григорьев И.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Задание на курсовую работу

Студент: Григорьев И.С.

Группа 6304

Тема работы: Обработка BMP-файла

Содержание пояснительной записки:

- Аннотация
- Содержание
- Введение (цель работы, формулировка задачи)
- Описание функций, использованных в ходе выполнения
- Тестирование работоспособности программы
- Разбиение на файлы и работа с Makefile
- Заключение
- Список использованных источников
- Приложение

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: _____

Дата сдачи реферата: _____

Дата защиты реферата: _____

Студент _____ Григорьев И.С.

Преподаватель _____ Берленко Т.А.

Аннотация

В данной работе описывается процесс создания программы для изменения полученного изображения (BMP-файла). Происходит считывание BMP-файла, проверка на корректность полученных данных, процесс разделения области изображения на четыре равных части и перестановка этих частей по диагонали. Результат записывается в новый файл. В документе приведен полный исходный код проекта на языке СИ, а также описание тестирования работоспособности программы с иллюстрациями.

Содержание

Введение

Требуется написать программу, входными данными которой являются BMP-файл и координаты верхнего левого угла и нижнего правого угла области, которую необходимо изменить следующим образом: область должна быть разделена на четыре равных части, а эти части должны быть переставлены местами по диагонали. Также программа должна проверять входные данные на корректность и выводить соответствующие сообщения об ошибке, если данные введены неверно.

Целью данной работы являются изучение структур, описывающих BMP-файл, понимание внутреннего устройства BMP-файла и создание программы для обработки BMP-файла.

Задачи, которые необходимо решить в ходе выполнения данной лабораторной работы:

1. Изучение и освоение структуры BMP-файла.
2. Разработка и реализация функций, необходимых для работы с BMP-файлом.
3. Разбиение проекта на несколько файлов.
4. Тестирование работоспособности программы.

Ход работы

1. Создание отдельных элементов проекта

1.1. Структура BMP-файла

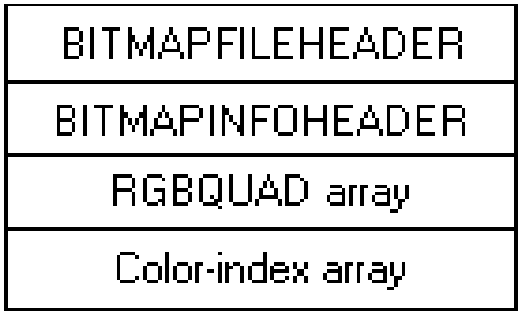


Рис. 1. Представление данных в BMP-файле

В начале стоит заголовок файла BITMAPFILEHEADER (Рис.1.). Он описан следующим образом:

```
typedef struct tagBITMAPFILEHEADER
{
   unsigned short bfType;
   unsigned int bfSize;
   unsigned short bfReserved1;
   unsigned short bfReserved2;
   unsigned int bfOffBits;
} BITMAPFILEHEADER;
```

Данная структура занимает 14 байт в памяти. Роль каждого поля данной структуры:

1. Поле bfType определяет тип файла. По условию нашей задачи тип обрабатываемого файла всегда BMP, значит первыми двумя символами файла являются две буквы “BM”.
2. Поле bfSize отвечает за размер самого файла в байтах.
3. Поля bfReserved1 и bfReserved2 зарезервированы и должны быть нулями.
4. Поле bfOffBits показывает, где начинается битовый массив относительно начала файла (от начала структуры BITMAPFILEHEADER), который описывает изображение.

Далее размещена структура BITMAPINFOHEADER, которая объявлена следующим образом:

```
typedef struct tagBITMAPINFOHEADER
{
   unsigned int biSize;
   unsigned int biWidth;
   unsigned int biHeight;
   unsigned short biPlanes;
   unsigned short biBitCount;
   unsigned int biCompression;
   unsigned int biSizeImage;
   unsigned int biXPelsPerMeter;
   unsigned int biYPelsPerMeter;
   unsigned int biClrUsed;
   unsigned int biClrImportant;
} BITMAPINFOHEADER;
```

Данная структура занимает 40 байт в памяти. Роль каждого поля структуры BITMAPINFOHEADER:

1. Поле biSize – размер самой структуры.
2. Поле biWidth задает ширину изображения.
3. Поле biHeight задает высоту изображения.
4. Поле biPlanes задает количество плоскостей.
5. Поле biBitCount – количество бит на один пиксель.
6. Поле biCompression обозначает тип сжатия.
7. Поле biSizeImage – размер изображения в байтах.
8. Поля biXPelsPerMeter и biYPelsPerMeter обозначают соответственно горизонтальное и вертикальное разрешение (в пикселях на метр) конечного устройства, на которое будет выводиться битовый массив (растр).
9. Поле biClrUsed определяет количество используемых цветов из таблицы.
10. Поле biClrImportant определяет число цветов, которые необходимы для того, чтобы изобразить рисунок.

Следующая позиция Рис. 1. является RGB-структурой, описывающей один пиксель. Так как по условию задачи выделяется 24 бита на цвет, нет необходимости в использовании палитры цветов для решения данной задачи.

```
typedef struct RGBTRIPLE
{
      char  rgbBlue;
      char  rgbGreen;
      char  rgbRed;
} RGBTRIPLE;
```

Последним элементом представления данных в BMP-файле является битовый массив растрового изображения. Количество байт памяти определяется размерами растра и числом бит на пиксель.

Так как поля структур выравниваются по границе кратной своему же размеру, необходимо применить следующие директивы:

```
#pragma pack(push,1)
typedef struct tagBITMAPFILEHEADER
{
   unsigned short bfType;
   unsigned int bfSize;
   unsigned short bfReserved1;
   unsigned short bfReserved2;
   unsigned int bfOffBits;
} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER
{
   unsigned int biSize;
   unsigned int biWidth;
   unsigned int biHeight;
   unsigned short biPlanes;
   unsigned short biBitCount;
   unsigned int biCompression;
   unsigned int biSizeImage;
   unsigned int biXPelsPerMeter;
   unsigned int biYPelsPerMeter;
   unsigned int biClrUsed;
   unsigned int biClrImportant;
} BITMAPINFOHEADER;

#pragma pack(pop)
```

Устанавливается размер выравнивания в 1 байт, описываются структуры и возвращаются предыдущие настройки.

1.2. Проверка на корректность данных

Программа, получая параметры из входного потока, должна проверить их на корректность. Проверка осуществляется специальной функцией AreDataCorrect( ) и условия корректности данных, расположенного в функции main( ).

```
int AreDataCorrect(FILE *bmp_file, int x0, int y0, int x1, int y1)
{
	if (bmp_file==NULL)
	{
		printf("Fail with <BMP_FILENAME>!\n");
		return 0;
	}
	if (x0<0 || y0<0 || x1<0 || y1<0 || (x1-x0)<0 || (y0-y1)<0)
	{
		printf("\nFail with <coordinates>. The entered coordinates are
not correct!\n");
		fclose(bmp_file);
		return 0;
	}
	 if (((x1-x0)%2==0) || ((y0-y1)%2==0))
	{
		printf("\nFail with <coordinates>. Can't devide image on equal
parts!\n");
		fclose(bmp_file);
		return 0;
	}
	return 1;
}
```

Первое условие (bmp_file==NULL) проверяет корректность открытия BMP-файла. Если файл не был открыт или был открыт некорректно, то программа напишет сообщение о соответствующей ошибке. Второе условие проверяет корректный ввод координат. Последнее условие необходимо для корректного разделения области на четыре равных части, если разность соответствующих координат будет четной, то разделить область на четыре равные части не получится.

1.3. Функция получения информации о BMP-файле

Функция, используя информацию, находящуюся в структурах BMP-файла, считывает информацию о файле и получает данные о растре. Сначала происходит корректное выделение памяти: указатель с помощью функции fseek( ) перемещается в конец файла. С помощью функции ftell( ) мы определяем размер изображения. Затем с помощью fseek( ) мы передвигаем указатель обратно. Далее происходит считывание файла в строку и извлечение заголовков. Чтобы правильно извлечь заголовок с информацией о изображении, мы двигаем указатель str на размер заголовка. Также определяется длина строки (учитывается выравнивание и то, что каждый пиксель кодируется 8 байтами). Затем происходит заполнение двумерного массива (растра).

```
char **BmpScan(FILE *bmp_file, BITMAPFILEHEADER *file_head, BITMAPINFOHEADER
*image_head)
{
	int i,j,k=0;

	fseek(bmp_file,0,SEEK_END);
	int image_size=ftell(bmp_file);
	fseek(bmp_file,0,SEEK_SET);

	char* str=(char*)malloc(sizeof(char)*image_size);
	fread(str, sizeof(char), image_size, bmp_file);

	*file_head=*((BITMAPFILEHEADER*)str);
	str+=sizeof(BITMAPFILEHEADER);
	*image_head=*((BITMAPINFOHEADER*)str);
	str-=sizeof(BITMAPFILEHEADER);
	str+=file_head->bfOffBits;

	int str_len=3*(image_head->biWidth)+(image_head->biWidth%4);

	char** raster=(char**)malloc(sizeof(char*)*image_head->biHeight);
	for(i=0;i<image_head->biHeight;i++)
	{
		raster[i]=(char*)malloc(sizeof(char)*str_len);
		for (j=0;j<str_len;j++)
			raster[i][j]=str[k++];
	}

	return raster;
}
```

1.4. Функция разбиения данной области и последующей перестановки частей

Функция принимает в качестве аргументов растр изображения, а также область изображения, которую необходимо изменить. Область в процессе работы данной функции делится на четыре части. Части по ходу меняются местами по диагоналям. Для работы с пикселями строка из символов была приведена к строке из пикселей.

```
char **BmpSwap(char **raster, int x0, int y0, int x1, int y1)
{
	int i,j;
	int x=(x1+x0)/2;
	int y=(y0+y1)/2;

	RGBTRIPLE* pixels_string=NULL;
	RGBTRIPLE temp;

	for (i=y1;i<=y0;i++)
	{
		pixels_string=(RGBTRIPLE*)raster[i];
		for (j=x0;j<=x;j++)
		{
			temp=pixels_string[j];
			pixels_string[j]=pixels_string[x+j+1];
			pixels_string[x+j+1]=temp;
		}
	}

	RGBTRIPLE* second_string=NULL;
	for (i=y1;i<=y;i++)
	{
		pixels_string=(RGBTRIPLE*)raster[i];
		second_string=(RGBTRIPLE*)raster[y+i+1];
		for(j=x0;j<=x1;j++)
		{
			temp=pixels_string[j];
			pixels_string[j]=second_string[j];
			second_string[j]=temp;
		}
	}

	return raster;
}
```

1.5. Функция записи результата в новый файл

По условию задачи, необходимо сохранить результат в новом BMP-файле. Это условие было реализовано в отдельной функции. Функция принимает на вход растр и заголовки соответствующих структур. Создается новый файл, в который записываются ранее упомянутые заголовки. Далее записывается сам растр и нули, необходимые для выравнивания.

```
void CreateChangedBmp(char** raster, BITMAPFILEHEADER* file_head,
BITMAPINFOHEADER*image_head)
{
	int i,j;

	FILE* changed_bmp_file=fopen(NEW_BMP_FILENAME,"wb");

	fwrite(file_head, sizeof(BITMAPFILEHEADER), 1, changed_bmp_file);
	fwrite(image_head, sizeof(BITMAPINFOHEADER), 1, changed_bmp_file);

	int byte_pixelsnum=3*image_head->biWidth;
	int byte_lining=image_head->biWidth%4;

	for(i=0;i<image_head->biHeight;i++)
	{
		fwrite(raster[i], sizeof(char), byte_pixelsnum,
changed_bmp_file);
		for (j=0;j<byte_lining;j++)
			fputc(0, changed_bmp_file);
	}

	fclose(changed_bmp_file);
}
```

1.6. Функция main

В функции main реализован интерфейс, необходимый для удобства работы с программой, а также вызваны все вышеперечисленные функции в необходимой последовательности для нужного результата.

```
int main()
{
	//BMP-file reading
	 FILE* bmp_file=fopen(BMP_FILENAME, "rb");
	 BITMAPFILEHEADER file_head;
	 BITMAPINFOHEADER image_head;

	 //Some interface
	int x0=-1,y0=-1,x1=-1,y1=-1;
	printf("Coordinates of the left upper corner (x0, y0): ");
	scanf("%d %d", &x0, &y0);
	printf("Coordinates of the right bottom corner (x1, y1): ");
	scanf("%d %d", &x1, &y1);

	//Checking data
	if(!AreDataCorrect(bmp_file, x0, y0, x1, y1))
	    return 0;

	//Reading raster from BMP-file
	char** raster=BmpScan(bmp_file, &file_head, &image_head);
	fclose(bmp_file);

	 //if entered area more than image area
	if (image_head.biWidth<(x1+1) || image_head.biHeight<(y0+1))
	{
		printf("Fail with <entered area>\n");
		return 0;
	}

	CreateChangedBmp(BmpSwap(raster, x0, y0, x1, y1), &file_head,
&image_head);
	printf("Bmp file /%s/ successfully changed in the same directory with
new name /%s/\n\n", BMP_FILENAME, NEW_BMP_FILENAME);

  return 0;
}
```

2. Работа с Makefile

Для удобства работы с программой проект был разделен на следующие файлы:

- main.c – основная функция;
- bmpfunctions.c – функции, обрабатывающие BMP-файл;
- bmpfunctions.h – объявление прототипов функций из “bmpfunctions.c”;

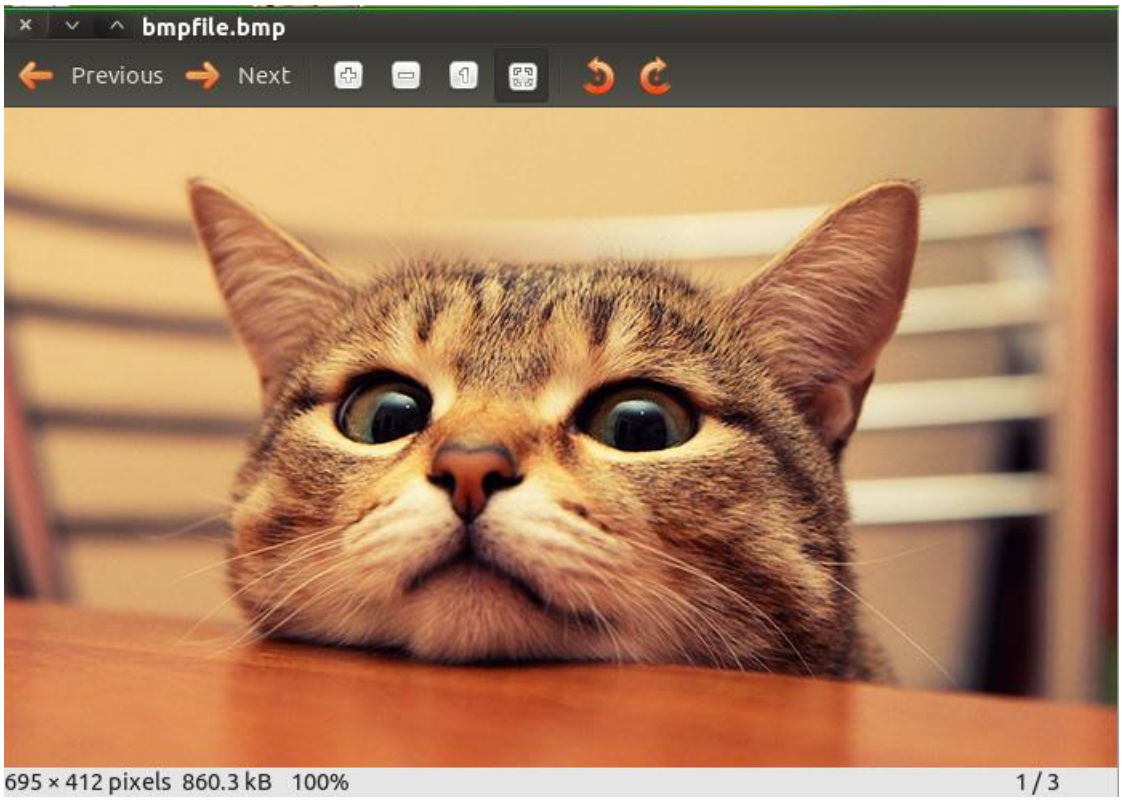
- bmpstructures.h – структуры, необходимы для работы с BMP-файлами.

Для упрощения и ускорения компиляции проекта был создан следующий Makefile:

```
swapqrt: main.o bmpfunctions.o
	gcc main.o -o swapqrt bmpfunctions.o
	 rm *.o
main.o: main.c bmpstructures.h bmpfunctions.h
	 gcc -c main.c
bmpfunctions.o: bmpfunctions.c bmpfunctions.h bmpstructures.h
	gcc -c bmpfunctions.c
```

3. Тестирование программы

- Изображение:



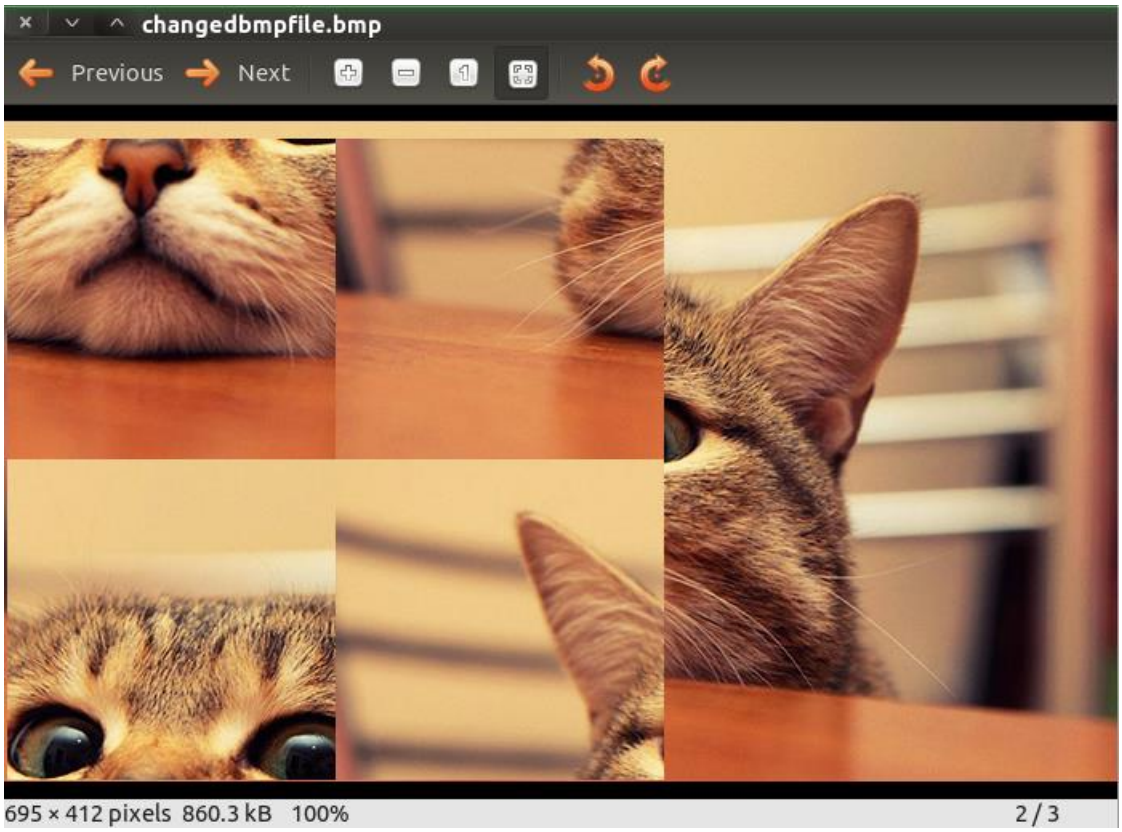
Input:

```
bi@bi-VirtualBox[Desktop] ./swapqrt
Coordinates of the left upper corner (x0, y0): 2 400
Coordinates of the right bottom corner (x1, y1): 411 1
```

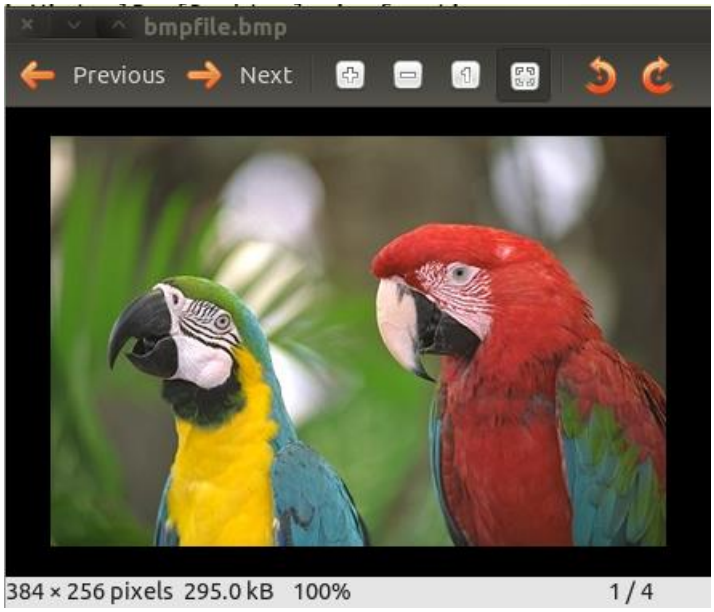
Output:

```
Bmp file /bmpfile.bmp/ successfully changed in the same directory with new name
/changedbmpfile.bmp/
```

Result:



- Изображение:



1)
Input:

```
bi@bi-VirtualBox[Desktop] ./swapqrt
Coordinates of the left upper corner (x0, y0): 2 200
Coordinates of the right bottom corner (x1, y1): 217 1
```

Output:

```
Bmp file /bmpfile.bmp/ successfully changed in the same directory with new name
/changedbmpfile.bmp/
```

Result:

2)

Input:

```
bi@bi-VirtualBox[Desktop] ./swapqrt
Coordinates of the left upper corner (x0, y0): 3 300
Coordinates of the right bottom corner (x1, y1): 506 1
```

Output:

```
Fail with <entered area>
```

Result: -

3)

Input:

```
bi@bi-VirtualBox[Desktop] ./swapqrt
Coordinates of the left upper corner (x0, y0): 4 201
Coordinates of the right bottom corner (x1, y1): 6 21
```

Output:

```
Fail with <coordinates>. Can't devide image on equal parts!
```

Result: -

Заключение

В ходе выполнения данной лабораторной работы были закреплены на практике принципы работы с BMP-файлом, были изучены структуры BMP-файла. Это было выполнено на примере программы, получающей на вход BMP-изображение и его некоторую область, которую надо было разделить на четыре равные части и переставить их по диагонали или же вывести сообщения об ошибке. Также было проведено разбиение проекта на части (отдельные файлы) для удобной работы и дальнейшего возможного расширения проекта.

Список использованных источников

1. Шилдт Г. – Полный справочник по C. – М.: Вильямс, 2004. – 752 с.
2. Wikipedia. – BMP. – 2016. URL: https://ru.wikipedia.org/wiki/BMP
3. Керниган Б., Ритчи Д. Язык программирования Си.\Пер. с англ., 3-е изд., испр. – СПб.: “Невский Диалект”, 2001. – 352 с.

Приложение А. Исходный код

Файл main.c

```
#include <sring.h>
#include <stdio.h>
#include <stdlib.h>

#define NAME_LIMIT 265
#define BMP_FILENAME "bmpfile.bmp"
#define NEW_BMP_FILENAME "changedbmpfile.bmp"

int main()
{
        FILE* bmp_file=fopen(BMP_FILENAME, "rb");
        BITMAPFILEHEADER file_head; //заголовок с информацией о файле
       BITMAPINFOHEADER image_head; //заголовок с информацией о изображении

       int x0=-1,y0=-1,x1=-1,y1=-1;
       printf("Coordinates of the left upper corner (x0, y0): ");
       scanf("%d %d", &x0, &y0);
       printf("Coordinates of the right bottom corner (x1, y1): ");
       scanf("%d %d", &x1, &y1);

       //Проверка введенных данных
       if(!AreDataCorrect(bmp_file, x0, y0, x1, y1))
           return 0;

       /*Создание и заполнение двумерного массива символов информацией о пикселях bmp
изображения*/
       char** raster=BmpScan(bmp_file, &file_head, &image_head);
       fclose(bmp_file);

       if (image_head.biWidth<(x1+1) || image_head.biHeight<(y0+1)) //Если введенная область
больше самого изображения
       {
              printf("Fail with <entered area>\n");
              return 0;
       }

       CreateChangedBmp(BmpSwap(raster, x0, y0, x1, y1), &file_head, &image_head);
       printf("Bmp file /%s/ successfully changed in the same directory with new name
/%s/\n\n", BMP_FILENAME, NEW_BMP_FILENAME);

  return 0;
}
```

Файл bmpfunctions.c

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#include "bmpstructures.h"
#include "bmpfunctions.h"
#define NEW_BMP_FILENAME "changedbmpfile.bmp"

int AreDataCorrect(FILE *bmp_file, int x0, int y0, int x1, int y1)
{

	if (bmp_file==NULL) //проверка на корректность открытия файла
	{
		printf("Fail with <BMP_FILENAME>!\n");
		return 0;
	}
    //проверка на корректность координат
	if (x0<0 || y0<0 || x1<0 || y1<0 || (x1-x0)<0 || (y0-y1)<0)
	{
		printf("\nFai; with <coordinates>. The entered coordinates are not correct!\n");
		fclose(bmp_file);
		return 0;
	}
    /*проверка на то, чтобы разность x и y координат была нечетной, для корректного
      разбиения на 4 равные части*/
   if (((x1-x0)%2==0) || ((y0-y1)%2==0))
	{
		printf("\nFail with <coordinates>. Can't devide image on equal parts!\n");
		fclose(bmp_file);
		return 0;
	}
	return 1;
}

char **BmpScan(FILE *bmp_file, BITMAPFILEHEADER *file_head, BITMAPINFOHEADER *image_head)
{
	int i,j,k=0;

    /*Корректное определение размера файла*/
	fseek(bmp_file,0,SEEK_END); //перемещает указатель в конец файла

	/*возвращает значение указателя текущего положения
	(значение соответствующее количеству байт от начала файла)*/
	int image_size=ftell(bmp_file);

	fseek(bmp_file,0,SEEK_SET); //перемещает указатель обратно в начало
```

```
	char* str=(char*)malloc(sizeof(char)*image_size);
	fread(str, sizeof(char), image_size, bmp_file); //считывание файла

	*file_head=*((BITMAPFILEHEADER*)str); //заголовок с информацией о файле
	str+=sizeof(BITMAPFILEHEADER); //указатель сдвигается на размер заголовка
	*image_head=*((BITMAPINFOHEADER*)str); //заголовок с информацией о изображении
	str-=sizeof(BITMAPFILEHEADER);
	str+=file_head->bfOffBits; //передвигаем указатель на битовый массив, описывающий само
изображение

	int str_len=3*(image_head->biWidth)+(image_head->biWidth%4); //каждый пиксель
кодируется 24 битами + выравнивание

  //заполнение двумерного массива
	char** raster=(char**)malloc(sizeof(char*)*image_head->biHeight);
	for(i=0;i<image_head->biHeight;i++)
	{
		raster[i]=(char*)malloc(sizeof(char)*str_len);
		for (j=0;j<str_len;j++)
			raster[i][j]=str[k++];
	}

	return raster;
}

char **BmpSwap(char **raster, int x0, int y0, int x1, int y1)
{
	int i,j;
	int x=(x1+x0)/2; //x координата половины области
	int y=(y0+y1)/2; //y координата половины области

	RGBTRIPLE* pixels_string=NULL; //указатель на строку из пикселей
	RGBTRIPLE temp; //указатель на пиксель

	for (i=y1;i<=y0;i++)
	{
		pixels_string=(RGBTRIPLE*)raster[i]; //char строка приводится к строке пикселей
		for (j=x0;j<=x;j++)
		{
			temp=pixels_string[j];
			pixels_string[j]=pixels_string[x+j+1];
			pixels_string[x+j+1]=temp;
		}
	}

	RGBTRIPLE* second_string=NULL;
	for (i=y1;i<=y;i++)
```

```
	{
		pixels_string=(RGBTRIPLE*)raster[i];
		second_string=(RGBTRIPLE*)raster[y+i+1];
		for(j=x0;j<=x1;j++)
		{
			temp=pixels_string[j];
			pixels_string[j]=second_string[j];
			second_string[j]=temp;
		 }
	}

	return raster;
}

void CreateChangedBmp(char** raster, BITMAPFILEHEADER* file_head, BITMAPINFOHEADER*
image_head)
{
	int i,j;

	FILE* changed_bmp_file=fopen(NEW_BMP_FILENAME,"wb");

	//запись заголовков
	fwrite(file_head, sizeof(BITMAPFILEHEADER), 1, changed_bmp_file);
	fwrite(image_head, sizeof(BITMAPINFOHEADER), 1, changed_bmp_file);

	int byte_pixelsnum=3*image_head->biWidth; //число байт содержащих информацию о пикселях
	int byte_lining=image_head->biWidth%4; //число байт для выравнивания

	for(i=0;i<image_head->biHeight;i++)
	{
		fwrite(raster[i], sizeof(char), byte_pixelsnum, changed_bmp_file); //запись
растра
		for (j=0;j<byte_lining;j++)
			fputc(0, changed_bmp_file); //дописывание нулей
	}

	fclose(changed_bmp_file);
}
```

Файл bmpfunctions.h

```
int AreDataCorrect(FILE *bmp_file, int x0, int y0, int x1, int y1);

char **BmpScan(FILE *bmp_file, BITMAPFILEHEADER *file_head, BITMAPINFOHEADER *image_head);

char **BmpSwap(char **raster, int x0, int y0, int x1, int y1);

void CreateChangedBmp(char** raster, BITMAPFILEHEADER* file_head, BITMAPINFOHEADER*
image_head);
```

Файл bmpstructures.h

```
#pragma pack(push, 1)
typedef struct tagBITMAPFILEHEADER
{
   unsigned short bfType; //определяет тип файла. Здесь он должен быть BM.
   unsigned int bfSize; //размер самого файла в байтах
   unsigned short bfReserved1; //зарезервирован и должен быть нулем
   unsigned short bfReserved2; //зарезервирован и должен быть нулем
   unsigned int bfOffBits; /*показывает, где начинается сам битовый массив относительно начала
файла
   (или от начала структуры BITMAPFILEHEADER), который и описывает картинку*/
} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER
{
   unsigned int biSize; //размер самой структуры
   unsigned int biWidth; //задает ширину картинки в пикселях
   unsigned int biHeight; //задает высоту картинки в пикселях
   unsigned short biPlanes; //задает количество плоскостей, пока оно всегда устанавливается в
1
   unsigned short biBitCount; //количество бит на один пиксель
   unsigned int biCompression; //тип сжатия, если сжатия нет, то этот флаг надо устанавливать
в BI_RGB
   unsigned int biSizeImage; /*обозначает размер картинки в байтах, если изображение несжато
   (то есть предыдущее поле установлено в BI_RGB), то здесь должен быть записан ноль*/
   unsigned int biXPelsPerMeter; /*горизонтальное разрешение (в пикселях на метр) конечного
устройства,
   на которое будет выводиться битовый массив (растр)*/
   unsigned int biYPelsPerMeter; /*вертикальное разрешение (в пикселях на метр) конечного
устройства,
   на которое будет выводиться битовый массив (растр)*/
   unsigned int biClrUsed; //определяет количество используемых цветов из таблицы
   unsigned int biClrImportant; /*это количество важных цветов. Определяет число цветов,
   которые необходимы для того, чтобы изобразить рисунок.
   Если это значение равно 0 (как это обычно и бывает), то все цвета считаются важными*/
} BITMAPINFOHEADER;

#pragma pack(pop)

typedef struct RGBTRIPLE
{
       char  rgbBlue;
       char  rgbGreen;
       char  rgbRed;
} RGBTRIPLE;
```

Файл makefile

```
swapqrt: main.o bmpfunctions.o
	gcc main.o -o swapqrt bmpfunctions.o
	rm *.o
main.o: main.c bmpstructures.h bmpfunctions.h
	gcc -c main.c
bmpfunctions.o: bmpfunctions.c bmpfunctions.h bmpstructures.h
	gcc -c bmpfunctions.c
```