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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка изображений

Студент гр. 0382 Тюленев Т. В.

Преподаватель Берленко Т. А.

Санкт-Петербург
2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Тюленев Т. В.

Группа 0382

Тема работы: Обработка изображений в формате BMP

Исходные данные:

Вариант 8.

Программа должна реализовывать следующий функционал по обработке bmp-файла

1. Рисование отрезка. Отрезок определяется:
   - координатами начала
   - координатами конца
   - цветом
   - толщиной
2. Отражение заданной области. Этот функционал определяется:
   - выбором оси относительно которой отражать (горизонтальная или вертикальная)
   - координатами левого верхнего угла области
   - координатами правого нижнего угла области
3. Рисование пентаграммы в круге. Пентаграмма определяется:
   - **либо** координатами левого верхнего и правого нижнего угла квадрата, в который вписана окружность пентаграммы, **либо** координатами ее центра и радиусом окружности
   - толщиной линий и окружности
   - цветом линий и окружности

Дата выдачи задания: 25.04.2021

Дата сдачи реферата: 20.05.2021

Дата защиты реферата: 21.05.2021

Студент _____ Тюленев Т. В.

Преподаватель _____ Берленко Т. А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе требовалось написать программу, обрабатывающую изображения в формате BMP с форматом пикселей RGB. Для реализации данной программы использовался язык C.

1. ЦЕЛЬ И ЗАДАНИЕ РАБОТЫ

1.1 Цель работы.

Целью данной работы является разработка программы, обрабатывающей изображения.

1.2 Задание.

Вариант 8.

Программа должна реализовывать следующий функционал по обработке bmp-файла

4. Рисование отрезка. Отрезок определяется:
   - координатами начала
   - координатами конца
   - цветом
   - толщиной
5. Отражение заданной области. Этот функционал определяется:
   - выбором оси относительно которой отражать (горизонтальная или вертикальная)
   - координатами левого верхнего угла области
   - координатами правого нижнего угла области
6. Рисование пентаграммы в круге. Пентаграмма определяется:
   - **либо** координатами левого верхнего и правого нижнего угла квадрата, в который вписана окружность пентаграммы, **либо** координатами ее центра и радиусом окружности
   - толщиной линий и окружности
   - цветом линий и окружности

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1 Чтение / запись изображения

Для чтения и записи изображения были использованны функции *fread* и *fwrite* соответственно. Также была создана структура *Rgb* для удобного представления цветов пикселей обрабатываемого файла.

2.2 Решение подзадачи 1. Рисование отрезка.

Для решения данной задачи был реализован Алгоритм Брезенхема для рисования наклонной прямой.

2.3 Решение подзадачи 2. Отражение заданной области.

При решении данной задачи мы находим центр области, который хотим отразить (вертикально или горизонтально) и относительно найденного центра меняем местами значения цветов пикселей на противоположных сторонах.

2.4 Решение подзадачи 3. Рисование пентаграммы в круге.

В данной задаче пользователю даётся на выбор способ ввода параметров **либо** координатами левого верхнего и правого нижнего угла квадрата, в который вписана окружность пентаграммы, **либо** координатами ее центра и радиусом окружности.

Далее по Алгоритму Брезенхема строится окружность по введенным координатом.

Далее, зная центр окружности и радиус, программа вычисляет точки вершин пентограммы.

Далее, по Алгоритму Брезенхема соединяем найденные точки.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного задания была создана программа для выполнения простейших действий над BMP файлом. Был реализован ввод параметров небходимых для работы через функцию *getopt_long()*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>
#include <getopt.h>

void help(){
        printf("help\n");
        printf("--line      -l\n");
        printf("--mirror    -m\n");
        printf("--pentagram -p\n");
        printf("--start     --s\n");
        printf("--end       --e\n");
        printf("--color     --c\n");
        printf("--asize     --a\n");
        printf("--vertical  --v\n");
        printf("--horizontal -H\n");
        printf("--radius    --r\n");
        printf("--bcentre   --b\n");
        printf("--help      --h\n");
        printf("for example:\n");
        printf("./1 --line --start 10,10 --end 100,150 --color 200,0,0 --asize 9 bmp1.bmp\n");
        printf("./1 -l --s 50,20 --e 100,250 --c 0,200,0 --a 10 bmp1.bmp\n");
        printf("./1 --mirror --horizontal --start 50,50 --end 300,300 bmp1.bmp\n");
        printf("./1 -m -H --s 100,50 --e 200,500 bmp1.bmp\n");
        printf("./1 --pentagram --start 100,50 --end 200,150 --color 0,0,0 --asize 10 bmp1.bmp\n");
        printf("./1 --pentagram --bcentre 100,100 --radius 10 --color red --asize 10 bmp1.bmp\n");
return;
}

#pragma pack (push, 1)
typedef struct
```

```
{
	unsigned short signature;
	unsigned int filesize;
	unsigned short reserved1;
	unsigned short reserved2;
	unsigned int pixelArrOffset;
} BitmapFileHeader;

typedef struct
{
	unsigned int headerSize;
	unsigned int width;
	unsigned int height;
	unsigned short planes;
	unsigned short bitsPerPixel;
	unsigned int compression;
	unsigned int imageSize;
	unsigned int xPixelsPerMeter;
	unsigned int yPixelsPerMeter;
	unsigned int colorsInColorTable;
	unsigned int importantColorCount;
} BitmapInfoHeader;

typedef struct
{
	unsigned char b;
	unsigned char g;
	unsigned char r;
} Rgb;

#pragma pack(pop)

void printFileHeader(BitmapFileHeader header){
	printf("signature:\t%x (%hu)\n", header.signature, header.signature);
	printf("filesize:\t%x (%u)\n", header.filesize, header.filesize);
	printf("reserved1:\t%x (%hu)\n", header.reserved1, header.reserved1);
```

```
        printf("reserved2:\t%x (%hu)\n", header.reserved2, header.reserved2);
        printf("pixelArrOffset:\t%x (%u)\n", header.pixelArrOffset, header.pixelArrOffset);
}

void printInfoHeader(BitmapInfoHeader header){
        printf("headerSize:\t%x (%u)\n", header.headerSize, header.headerSize);
        printf("width:     \t%x (%u)\n", header.width, header.width);
        printf("height:    \t%x (%u)\n", header.height, header.height);
        printf("planes:    \t%x (%hu)\n", header.planes, header.planes);
        printf("bitsPerPixel:\t%x (%hu)\n", header.bitsPerPixel, header.bitsPerPixel);
        printf("compression:\t%x (%u)\n", header.compression, header.compression);
        printf("imageSize:\t%x (%u)\n", header.imageSize, header.imageSize);
        printf("xPixelsPerMeter:\t%x (%u)\n", header.xPixelsPerMeter, header.xPixelsPerMeter);
        printf("yPixelsPerMeter:\t%x (%u)\n", header.yPixelsPerMeter, header.yPixelsPerMeter);
        printf("colorsInColorTable:\t%x (%u)\n", header.colorsInColorTable, header.colorsInColorTable);
        printf("importantColorCount:\t%x (%u)\n", header.importantColorCount,
header.importantColorCount);
}


int main (int argc, char *argv[]){
        char *opts = "SecsCr:lmvHph?";
        static struct option longOpts[] = {
                {"line",      0,0,'l'},
                {"start",     1,0,'s'},
                {"end",       1,0,'e'},
                {"color",     1,0,'c'},
                {"asize",     1,0,'a'},
                {"mirror",    0,0,'m'},
                {"vertical",  0,0,'v'},
                {"horizontal",0,0,'H'},
                {"pentagram", 0,0,'p'},
                {"bcentre",   1,0,'b'},
                {"radius",    1,0,'r'},
                {"help",      0,0,'h'},
                {0,0,0,0}};
```

```
int opt, longIndex;
opterr=0;
opt=getopt_long(argc, argv, opts, longOpts, &longIndex);
int type=-1;
int x,y,xx,yy,size,p,centrex,centrey,colorR,colorG,colorB,i,j,ii,jj;
colorR=255;
colorG=255;
colorB=255;
size=1;
char z[2000][2000];
int fo = 0;
double rad1,rad2;
int signa, signb;
int xo, yo;
int A, B, sign;
int radius=-1;
char *color;
while (opt!=-1){
        switch(opt){
                case 'l':
                type=1;
                break;
                case 'm':
                type=2;
                break;
                case 'p':
                type=3;
                break;
                case 's':
                sscanf(optarg, "%d,%d", &x,&y);
                break;
      case 'e':
                sscanf(optarg, "%d,%d", &xx,&yy);
                break;
         case 'a':
                sscanf(optarg, "%d", &size);
```

```
                break;
        case 'c':
                sscanf(optarg, "%d,%d,%d", &colorR,&colorG,&colorB);
                break;
        case 'v':
                p=0;
                break;
        case 'H':
                p=1;
                break;
        case 'r':
                sscanf(optarg, "%d", &radius);
                break;
        case 'b':
                sscanf(optarg, "%d,%d", &centrex,&centrey);
                break;
      case 'h':
                type=-1;
                break;
      case '?':
                type=-1;
                break;
}opt=getopt_long(argc, argv, opts, longOpts, &longIndex);}

if (type==-1)help();
        argv += optind;
        argc = argc-optind-1;


if(type==1){
        printf("line\n");
        printf("start: %d %d\n",x,y);
        printf("end: %d %d\n",xx,yy);
        printf("color: %d %d %d\n",colorR,colorG,colorB);
        printf("size: %d\n",size);
        printf("file: %s\n",argv[argc]);}
```

```
if(type==2){
        printf("mirror\n");
        if(p==0)printf("vertical\n");else printf("horizontal\n");
        printf("start: %d %d\n",x,y);
        printf("end: %d %d\n",xx,yy);
        printf("file: %s\n",argv[argc]);}
if(type==3){
        printf("pentagram\n");
        if(radius==-1){
        printf("start: %d %d\n",x,y);
        printf("end: %d %d\n",xx,yy);}else{
        printf("centre: %d %d\n",centrex,centrey);
        printf("radius: %d\n",radius);}
        printf("color: %d %d %d\n",colorR,colorG,colorB);
        printf("size: %d\n",size);
        printf("file: %s\n",argv[argc]);}



FILE *f = fopen(argv[argc], "rb");
BitmapFileHeader bmfh;
BitmapInfoHeader bmif;
fread(&bmfh,1,sizeof(BitmapFileHeader),f);
fread(&bmif,1,sizeof(BitmapInfoHeader),f);
        if(type!=-1){
        printf("--------------------------------\n");
        printFileHeader(bmfh);
        printInfoHeader(bmif);}

unsigned int H = bmif.height;
unsigned int W = bmif.width;

Rgb **arr = malloc(H*sizeof(Rgb*));
for(i=0; i<H; i++){
        arr[i] = malloc(W * sizeof(Rgb) + (W*3)%4);
        fread(arr[i],1,W * sizeof(Rgb) + (W*3)%4,f);}
```

```
for (i = 0; i < H; i++){
        for (j = 0; j < W; j++)
                z[i][j] = '-';}

if(type==1){

                A = yy - y;
                B = x - xx;
        if (abs(A) > abs(B)) sign = 1;
                else sign = -1;
        int signa, signb;
        if (A < 0) signa = -1;
                else signa = 1;
        if (B < 0) signb = -1;
                else signb = 1;
        fo = 0;
        z[y][x] = '*';
        xo = x; yo = y;
        if (sign == -1) {
                do {
                        fo += A*signa;
                        if (fo > 0){
                        fo -= B*signb;
                        yo += signa;}
                        xo -= signb;
                        z[yo][xo] = '*';} while (xo != xx || yo != yy);}
        else{
                do {
                        fo += B*signb;
                        if (fo > 0) {
                        fo -= A*signa;
                        xo -= signb;}
                                yo += signa;
                                z[yo][xo] = '*';} while (xo != xx || yo != yy);}
```

```
        for (i = 0; i < W; i++) {
                for (ii = 1; ii <= size; ii++){
                        for (j = 0; j < H; j++){
                                for (jj = 1; jj <= size; jj++){
                                        if(z[i][j]=='*'){
                                                arr[H-i+ii][j+jj].r=colorR;
                                                arr[H-i+ii][j+jj].g=colorG;
                                                arr[H-i+ii][j+jj].b=colorB;}}}}}}

if(type==2){
        int cc,colorT;
        if(p==1){
                if((x+xx)%2==1){
                        cc=(x+xx)/2+1;}
                if((x+xx)%2==0){
                        cc=(x+xx)/2;}}
        if(p==0){
                if((y+yy)%2==1){
                        cc=(y+yy)/2+1;}
                if((y+yy)%2==0){
                        cc=(y+yy)/2;}}
        if(p==1){
                for(i=x;i<cc;i++){
                        for(j=y;j<yy;j++){
                                colorT=arr[H-i][j].r;
                                arr[H-i][j].r=arr[H-cc-cc+i][j].r;
                                arr[H-cc-cc+i][j].r=colorT;

                                colorT=arr[H-i][j].g;
                                arr[H-i][j].g=arr[H-cc-cc+i][j].g;
                                arr[H-cc-cc+i][j].g=colorT;

                                colorT=arr[H-i][j].b;
                                arr[H-i][j].b=arr[H-cc-cc+i][j].b;
                                arr[H-cc-cc+i][j].b=colorT;}}}
        if(p==0){
```

```
                for(i=x;i<xx;i++){
                        for(j=y;j<cc;j++){
                                colorT=arr[H-i][j].r;
                                arr[H-i][j].r=arr[H-i][cc+cc-j].r;
                                arr[H-i][cc+cc-j].r=colorT;

                                colorT=arr[H-i][j].g;
                                arr[H-i][j].g=arr[H-i][cc+cc-j].g;
                                arr[H-i][cc+cc-j].g=colorT;

                                colorT=arr[H-i][j].b;
                                arr[H-i][j].b=arr[H-i][cc+cc-j].b;
                                arr[H-i][cc+cc-j].b=colorT;}}}}

if(type==3){
        if(radius==-1){
                centrex=(x+xx)/2;
                centrey=(y+yy)/2;
                radius=(xx-x)/2;}

        int x1,y1,gap,delta;
        for(jj=radius-1;jj<radius+size;jj++){
        x1 = 0; y1 = jj; gap = 0; delta = (2 - 2 * radius);
while (y1 >= 0){

        arr[H-centrex + x1][centrey + y1].r=colorR;
                arr[H-centrex + x1][centrey + y1].g=colorG;
                arr[H-centrex + x1][centrey + y1].b=colorB;

        arr[H-centrex + x1][centrey - y1].r=colorR;
                arr[H-centrex + x1][centrey - y1].g=colorG;
                arr[H-centrex + x1][centrey - y1].b=colorB;

                arr[H-centrex - x1][centrey - y1].r=colorR;
                arr[H-centrex - x1][centrey - y1].g=colorG;
                arr[H-centrex - x1][centrey - y1].b=colorB;
```

```
            arr[H-centrex - x1][centrey + y1].r=colorR;
            arr[H-centrex - x1][centrey + y1].g=colorG;
            arr[H-centrex - x1][centrey + y1].b=colorB;

gap = 2 * (delta + y1) - 1;
if (delta < 0 && gap <= 0){
            x1++;
            delta += 2 * x1 + 1;
continue;}
if (delta > 0 && gap > 0){
            y1--;
            delta -= 2 * y1 + 1;
continue;}
x1++;
delta += 2 * (x1 - y1);
y1--;}}

for(i=0;i<5;i++){

rad1=1.*(18+226*i)/180;
rad2=1.*(18+226*(i+2))/180;
x=radius*cos(rad1)/1+centrex;
y=radius*sin(rad1)/1+centrey;
xx=radius*cos(rad2)/1+centrex;
yy=radius*sin(rad2)/1+centrey;

A = yy - y;
B = x - xx;
if (abs(A) > abs(B)) sign = 1;
            else sign = -1;
int signa, signb;
if (A < 0) signa = -1;
            else signa = 1;
if (B < 0) signb = -1;
            else signb = 1;
```

```
        fo = 0;
        z[y][x] = '*';
        xo = x; yo = y;
        if (sign == -1) {
                do {
                        fo += A*signa;
                        if (fo > 0){
                        fo -= B*signb;
                        yo += signa;}
                        xo -= signb;
                        z[yo][xo] = '*';} while (xo != xx || yo != yy);}
        else{
                do {
                        fo += B*signb;
                        if (fo > 0) {
                        fo -= A*signa;
                        xo -= signb;}
                                yo += signa;
                                z[yo][xo] = '*';} while (xo != xx || yo != yy);}}
  for (i = 0; i < W; i++) {
                for (ii = 1; ii <= size; ii++){
                        for (j = 0; j < H; j++){
                                for (jj = 1; jj <= size; jj++){
                                        if(z[i][j]=='*'){
                        arr[H-i+ii-centrex+centrey][j+jj-centrex+centrey].r=colorR;
                        arr[H-i+ii-centrex+centrey][j+jj-centrex+centrey].g=colorG;
                        arr[H-i+ii-centrex+centrey][j+jj-centrex+centrey].b=colorB;}}}}}}

FILE *ff = fopen("out.bmp", "wb");

bmif.height = H;
bmif.width = W;
fwrite(&bmfh, 1, sizeof(BitmapFileHeader),ff);
fwrite(&bmif, 1, sizeof(BitmapInfoHeader),ff);
unsigned int w = (W) * sizeof(Rgb) + ((W)*3)%4;
for(int i=0; i<H; i++){
```

```
                fwrite(arr[i],1,w,ff);
        }
        fclose(ff);
        printf("\n");
return 0;
}
```