

Факультет программной инженерии и компьютерной техники

Низкоуровневое программирование

Лабораторная работа №6

Преподаватель:

Логинов Иван Павлович

Выполнил:

Рождественский Никита Сергеевич

Р33113

## Задание:

Повернуть изображение в формате bmp.

Выполнение:

enum read\_status read\_bmp\_header(struct bmp\_header\* bmpHeader,FILE\* bmp\_file){

read\_uint16(&bmpHeader->bfType,bmp\_file);

read\_uint32(&bmpHeader->bfileSize,bmp\_file);

read\_uint32(&bmpHeader->bfReserved,bmp\_file);

read\_uint32(&bmpHeader->bOffBits,bmp\_file);

read\_uint32(&bmpHeader->biSize,bmp\_file);

read\_uint32(&bmpHeader->biWidth,bmp\_file);

read\_uint32(&bmpHeader->biHeight,bmp\_file);

read\_uint16(&bmpHeader->biPlanes,bmp\_file);

read\_uint16(&bmpHeader->biBitCount,bmp\_file);

read\_uint32(&bmpHeader->biCompression,bmp\_file);

read\_uint32(&bmpHeader->biSizeImage,bmp\_file);

read\_uint32(&bmpHeader->biXPelsPerMeter,bmp\_file);

read\_uint32(&bmpHeader->biYPelsPerMeter,bmp\_file);

read\_uint32(&bmpHeader->biClrUsed,bmp\_file);

read\_uint32(&bmpHeader->biClrImportant,bmp\_file);

return READ\_OK;

}

enum read\_status read\_colour\_table(uint32\_t\* colours,FILE\* bmp\_file,struct bmp\_header\* bmpHeader){

for(size\_t i=0;i<(bmpHeader->bOffBits-54)/4;i++){

read\_uint32(&colours[i],bmp\_file);

}

return READ\_OK;

}

enum read\_status read\_pixels(struct pixel\* pixels,FILE\* bmp\_file,struct bmp\_header\* bmpHeader){

size\_t width=bmpHeader->biWidth;

size\_t height=bmpHeader->biHeight;

uint8\_t trash=0;

for(size\_t i=0;i<height;i++){

for(size\_t j=0;j<width;j++){

read\_uint8(&pixels[i\*width+j].b,bmp\_file);

read\_uint8(&pixels[i\*width+j].g,bmp\_file);

read\_uint8(&pixels[i\*width+j].r,bmp\_file);

read\_uint8(&trash,bmp\_file);

}

}

return READ\_OK;

}

enum write\_status write\_header(struct bmp\_header\* bmpHeader,FILE\* bmp\_file){

write\_uint16(&bmpHeader->bfType,bmp\_file);

write\_uint32(&bmpHeader->bfileSize,bmp\_file);

write\_uint32(&bmpHeader->bfReserved,bmp\_file);

write\_uint32(&bmpHeader->bOffBits,bmp\_file);

write\_uint32(&bmpHeader->biSize,bmp\_file);

write\_uint32(&bmpHeader->biWidth,bmp\_file);

write\_uint32(&bmpHeader->biHeight,bmp\_file);

write\_uint16(&bmpHeader->biPlanes,bmp\_file);

write\_uint16(&bmpHeader->biBitCount,bmp\_file);

write\_uint32(&bmpHeader->biCompression,bmp\_file);

write\_uint32(&bmpHeader->biSizeImage,bmp\_file);

write\_uint32(&bmpHeader->biXPelsPerMeter,bmp\_file);

write\_uint32(&bmpHeader->biYPelsPerMeter,bmp\_file);

write\_uint32(&bmpHeader->biClrUsed,bmp\_file);

write\_uint32(&bmpHeader->biClrImportant,bmp\_file);

return WRITE\_OK;

}

enum write\_status write\_colour\_table(uint32\_t\* colours,FILE\* bmp\_file,struct bmp\_header\* bmpHeader){

for(size\_t i=0;i<(bmpHeader->bOffBits-54)/4;i++){

write\_uint32(&colours[i],bmp\_file);

}

return WRITE\_OK;

}

enum write\_status write\_pixels(struct pixel\* pixels,FILE\* bmp\_file,struct bmp\_header\* bmpHeader){

size\_t width=bmpHeader->biWidth;

size\_t height=bmpHeader->biHeight;

uint8\_t trash=255;

for(size\_t i=0;i<height;i++){

for(size\_t j=0;j<width;j++){

write\_uint8(&pixels[i\*width+j].b,bmp\_file);

write\_uint8(&pixels[i\*width+j].g,bmp\_file);

write\_uint8(&pixels[i\*width+j].r,bmp\_file);

write\_uint8(&trash,bmp\_file);

}

}

return WRITE\_OK;

}

struct image\* rotate( const struct image\* source ){

struct image\* new\_image=malloc(sizeof(uint32\_t)\*2+ sizeof(uint8\_t)\*3\*source->height\*source->width);

struct pixel\* pixels= malloc(sizeof(uint8\_t)\*3\*source->height\*source->width);

new\_image->width=source->height;

new\_image->height=source->width;

uint32\_t count=1;

uint32\_t count2=0;

for(size\_t i=0;i<source->height\*source->width;i++){

pixels[i]=source->data[(source->width)\*count++-1-count2];

if(count==source->height+1){

count=1;

count2++;

}

}

new\_image->data=pixels;

return new\_image;

}

Вывод: В результате выполнения данной лабораторной работы было реализовано чтение и запись изображений в формате BMP.