**Задание**

1. Осуществить чтение/запись изображения в формате с типом компоненты цвета К (табл. 7) и форматом данных Ф (табл. 8). Реализовать вызов этой функции через подменю.
2. Осуществить масштабирование изображения согласно заданию М (табл. 9). Добавить соответствующий пункт меню.
3. Осуществить отображение только тех компонент, которые указаны в задании Т (табл. 10) Добавить соответствующий пункт меню.
4. Осуществить инверсию компонент цвета, которые указаны в задании Т (табл. 10). Добавить соответствующий пункт меню.

4 вариант

К4 - Синий компонент цвета с сопутствующими зеленым, красным и альфа-компонентами

М4 - Увеличить изображение в 2 раза

Т3 – B

Ф4 - 16-разрядное целое число со знаком

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <gl\glut.h>

#include <atlbase.h>

#include <atlconv.h>

#include <string>

#include <cmath>

#include <math.h>

#include <Windows.h>

#include <stdlib.h>

#include <glaux/Glaux.h>

#pragma comment(lib, "D:\\Programms\\Microsoft Visual Studio 2022\\VC\\Tools\\MSVC\\14.34.31933\\lib\\x86\\Glaux.lib")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

/\*Структура для хранения заголовка файла изображения\*/

struct Zagolovok

{

GLint shirina; /\*Ширина\*/

GLint vysota; /\*Высота\*/

GLenum formatCveta; /\*Формат представления цвета\*/

GLenum formatKomponenty; /\*Формат данных компоненты цвета\*/

int kol\_voKomponent; /\*Количество компонент цвета\*/

};

/\*Структура для хранения изображения\*/

struct Izobrajenie

{

unsigned char\* pikseli;

Zagolovok zagolovok;

};

/\*Функция чтения изображения из фалйа .kai\*/

Izobrajenie\* ChtenieIzobrajeniyaIzFaila(const char\* szFileName)

{

FILE\* pFile; // Указатель файла

Izobrajenie\* iz = (Izobrajenie\*)malloc(sizeof(Izobrajenie)); //Создание структуры дляхранения изображения

// Открытие файла

pFile = fopen(szFileName, "rb");

if (pFile == NULL) return NULL;

// Считываем заголовок

fread(&(iz->zagolovok), sizeof(Zagolovok), 1, pFile);

//Создание массива для битов

iz->pikseli = (unsigned char\*)malloc(iz->zagolovok.shirina \* iz->zagolovok.vysota \* iz->zagolovok.kol\_voKomponent);

// Считывание битов

fread(iz->pikseli, iz->zagolovok.shirina \* iz->zagolovok.vysota \* iz->zagolovok.kol\_voKomponent, 1, pFile);

// Работа с файлом завершается

fclose(pFile);

// Возвращает указатель на данные изображения

return iz;

}

/\*Функция записи изображения в файл .kai\*/

GLint SohranenieIzobrajeniyaVfail(const char\* szFileName)

{

FILE\* pFile; // Указатель файла

//Создание структуры для хранения изображения

Izobrajenie\* iz = (Izobrajenie\*)malloc(sizeof(Izobrajenie));

GLint iViewport[4]; //Массив для хранения размеров порта просмотра

// Получение размеров порта просмотра

glGetIntegerv(GL\_VIEWPORT, iViewport);

// Считывание битов из буфера цвета

glPixelStorei(GL\_PACK\_ALIGNMENT, 1);

glPixelStorei(GL\_PACK\_ROW\_LENGTH, 0);

glPixelStorei(GL\_PACK\_SKIP\_ROWS, 0);

glPixelStorei(GL\_PACK\_SKIP\_PIXELS, 0);

// Переключение на передний буфер

glReadBuffer(GL\_FRONT);

//Установка параметров изображения

iz->zagolovok.shirina = iViewport[2];

iz->zagolovok.vysota = iViewport[3];

iz->zagolovok.formatCveta = GL\_BLUE;

iz->zagolovok.formatKomponenty = GL\_UNSIGNED\_SHORT;

iz->zagolovok.kol\_voKomponent = 3;

//Выделение памяти для хранения битов

iz->pikseli = (unsigned char\*)malloc(iz->zagolovok.shirina \* iz->zagolovok.vysota \* iz->zagolovok.kol\_voKomponent);

//Чтение битов

glReadPixels(0, 0, iz->zagolovok.shirina / 2, iz->zagolovok.vysota / 2, iz->zagolovok.formatCveta, iz ->

zagolovok.formatKomponenty, iz->pikseli);

// Открытие файла

pFile = fopen(szFileName, "wb");

// Запись заголовка

fwrite(&(iz->zagolovok), sizeof(Zagolovok), 1, pFile);

// Запись данных об изображении

fwrite(iz->pikseli, iz->zagolovok.shirina \* iz->zagolovok.vysota \* iz->zagolovok.kol\_voKomponent, 1, pFile);

//Закрытие файла

fclose(pFile);

// Успех

return 1;

}

Izobrajenie\* izobr; //Текущее изображение

Izobrajenie\* izobr\_bmp; //Изображение из файла .bmp

Izobrajenie\* izobr\_kai; //Изображение из файла .kai

Izobrajenie\* izobr\_ch\_b; //Черно-белое изображение

AUX\_RGBImageRec\* pImage = NULL; //Изображение AUX\_RGB

//Переменная для хранения режима рисования

static GLint rejim = 1;

// Должным образом обновляем флаги в ответ на выбор позиции из меню

void ObrabotkaMenu(int punktMenu)

{

// меняем индекс режима визуализации на индекс,

// соответствующий позиции меню

rejim = punktMenu;

// Активизируем перерисовывание изображения glutPostRedisplay(); }

glutPostRedisplay();

}

int k;

int iz = 0;

void Pererisovka(void)

{

GLint iViewport[4];

GLbyte\* pModifiedBytes = NULL;

GLfloat invertMap[256];

GLint i;

// Очищаем окно текущим цветом очистки

glClear(GL\_COLOR\_BUFFER\_BIT);

// Текущее растровое положение всегда соответствует левому нижнему углу окна

glRasterPos2i(64, 64);

// В зависимости от индекса режима визуализации выполняются необходимые операции с изображением

switch (rejim)

{

case 0: /\*Очистка экрана\*/

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

break;

case 1: /\*Загрузка изображения из файла \*.bmp\*/

pImage = auxDIBImageLoadA("C:\\Users\\Ильдар\\Desktop\\Учеба\\ComputerGraphic\\Labs\\Лаба 6\\texture\_2.bmp");

izobr\_bmp = (Izobrajenie\*)malloc(sizeof(Izobrajenie));

izobr\_bmp->zagolovok.shirina = pImage->sizeX \* 1.5;

izobr\_bmp->zagolovok.vysota = pImage->sizeY \* 1.5;

izobr\_bmp->pikseli = pImage->data;

izobr\_bmp->zagolovok.formatCveta = GL\_BLUE;

izobr\_bmp->zagolovok.formatKomponenty = GL\_UNSIGNED\_SHORT;

izobr\_bmp->zagolovok.kol\_voKomponent = 3;

izobr = izobr\_bmp;

break;

case 2: /\*Инверсия цветов\*/

invertMap[0] = 1.0f;

for (i = 1; i < 256; i++)

invertMap[i] = 1.0f - (1.0f / 255.0f \* (GLfloat)i);

glPixelMapfv(GL\_PIXEL\_MAP\_R\_TO\_R, 255, invertMap);

glPixelMapfv(GL\_PIXEL\_MAP\_G\_TO\_G, 255, invertMap);

glPixelTransferi(GL\_MAP\_COLOR, GL\_TRUE);

break;

default:

break;

}

if (rejim != 0)

// Рисуются пиксели

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glDrawPixels(izobr->zagolovok.shirina, izobr->zagolovok.vysota, izobr->zagolovok.formatCveta, izobr -> zagolovok.formatKomponenty, izobr->pikseli);

}

//Переключает буферы

glutSwapBuffers();

}

void IzmenenieRazmera(int w, int h)

{

//Предотвращает деление на ноль, когда окно слишком маленькое

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

//Система координат обновляется перед модификацией

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

//Точка входа основной программы

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GL\_DOUBLE);

glutInitWindowSize(512, 512);

glutCreateWindow("Операции над пикселями");

glutReshapeFunc(IzmenenieRazmera);

glutDisplayFunc(Pererisovka);

//Создается меню и добавляются опции выбора

glutCreateMenu(ObrabotkaMenu);

glutAddMenuEntry("Очистить экран", 0);

glutAddMenuEntry("Загрузить изображение .bmp", 1);;

glutAddMenuEntry("Инверсия цвета", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

gluOrtho2D(0, 512, 0, 512);

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

rejim = 0;

glutMainLoop();

// Освобождаем исходные данные изображений

free(izobr);

free(izobr\_ch\_b);

free(izobr\_kai);

free(izobr\_bmp);

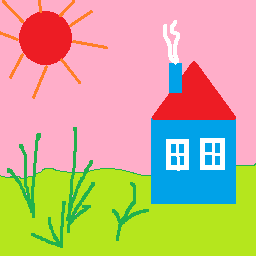
free(pImage);

return 0;

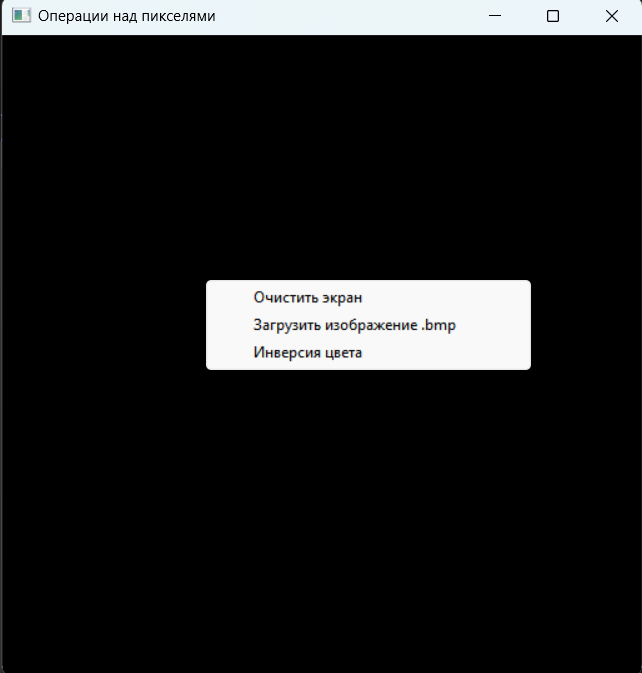
}

**Выполнение программы**

Исходное фото:



Результат работы программы:



Загружаем фотографию по условию задачи:



Применяем инверсию цветов (без синего компонента):

