|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**Лабораторная работа №5**

**«Обработка бинарных файлов»**

**ДИСЦИПЛИНА: «Типы и структуры данных»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Петроченков И. А.\_\_)  (Подпись) (Ф.И.О.) |
| Проверил(а): | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_Пчелинцева\_Н.\_И. )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2023 г. | | |

# Цели работы:

Целью выполнения лабораторной работы является формирование практических навыков создания алгоритмов обработки бинарных файлов.

**Задачи:**

Основными задачами выполнения лабораторной работы являются:

1. Познакомиться со структурой бинарного bmp-файла;
2. Изучить способы программной обработки бинарного файла;
3. Реализовать алгоритм согласно варианту;

**Вариант №18**

**Задания**

* Изменить цвет рисунка
* Обработать/создать/изменить файл, содержание которого предусмотрено вариантом задания.

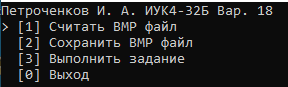


Рис. 1 Меню программы



Рис. 2 Загрузка изображения



Рис. 3 Изменение цвета рисунка



Рис. 4 Сохранение рисунка

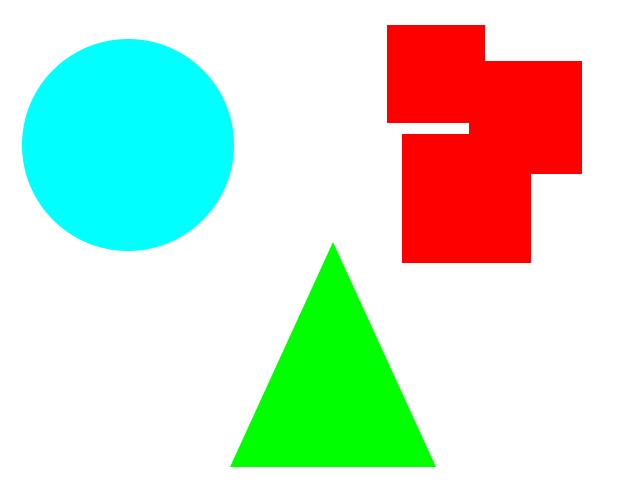


Рис. 5 Исходное изображение

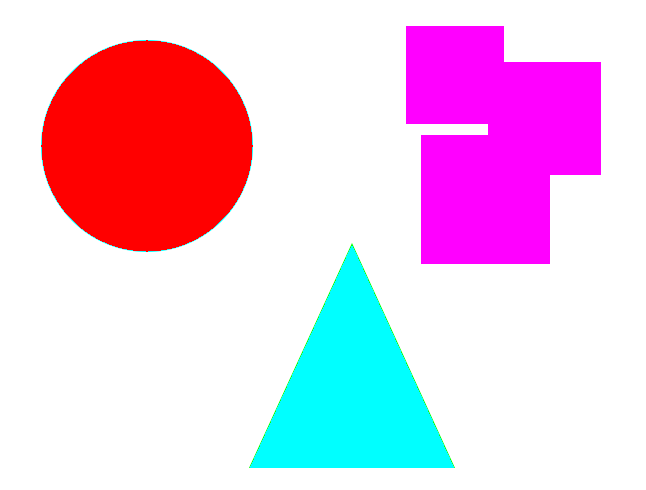


Рис. 6 Измененный рисунок

# Вывод:

Результатами работы являются:

* Программа, реализующая операции над изображениями типа BMP;
* Подготовленный отчет;

**Листинг программы:**

Lab5.cpp

#include <iostream>

#include "Addition.h"

using namespace std;

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

{

setlocale(LC\_ALL, "ru");

if (!Start(argc, argv))

{

Menu menu = Menu("Петроченков И. А. ИУК4-32Б Вар. 18");

menu.AddItem("Считать BMP файл", ReadBMP);

menu.AddItem("Сохранить BMP файл", WriteBMP);

menu.AddItem("Выполнить задание", DoTask);

//menu.AddItem("Генерация изображения", GenerateBMP);

//menu.AddItem("Изменить изображение", ChangeBMP);

menu.RunMenu();

}

}

Addition.h

#pragma once

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\Menu.h"

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\MenuItem.h"

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\Exceptions.h"

#include <iostream>

#include <fstream>

#include <random>

#include <string>

std::string exc = "exception.txt";

//std::string src = "alph.txt";

std::string dst = "changed.bmp";

std::string src = "figures.bmp";

#include <Windows.h>

#include "myBMP.h"

MyBMP bmp\_ = MyBMP();

bool InputError()

{

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

std::cout << "\nОшибка! Введено некорректное значение!\n Попробуйте снова.\n";

return false;

}

bool Start(int argc, char\* argv[])

{

std::string message = "Параметры запуска программы:\nsource.txt - обязательный параметр, место хранения графа\ndestination.txt - обязательный параметр, место записи графа\n";

if (argc <= 1)

{

//std::cout << argc << std::endl;

std::cout << message;

return false;

}

else

{

if (argc == 3)

{

//std::cout << argc << std::endl;

src = argv[1];

dst = argv[2];

std::cout << dst;

return true;

}

else if (argc > 3)

{

//std::cout << argc << std::endl;

std::cout << message << std::endl;

return false;

}

}

}

void ReadBMP()

{

try

{

bmp\_.Load(src.c\_str());

std::cout << "Файл успешно считан!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка считывания файла" << std::endl;

}

}

void WriteBMP()

{

try

{

bmp\_.Save(dst.c\_str());

std::cout << "Файл успешно записан!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка записи файла" << std::endl;

}

}

void DoTask()

{

try

{

bmp\_.DoTask();

std::cout << "Цвет успешно изменен!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка обработки файла" << std::endl;

}

}

BMP.h

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\Menu.h"

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\MenuItem.h"

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ИУК4-32Б\Типы и структуры данных\lab3\lab3\Exceptions.h"

#include <C:\Users\Игорь\Desktop\учеба\LabRab\ЛР ВП 2 Сем\1 ЛР\LR1\LR1\myException.h>

#include "C:\Users\Игорь\Desktop\учеба\LabRab\ЛР ВП 2 Сем\1 ЛР\LR1\LR1\myVector.h"

#include <fstream>

typedef int SDWORD;

// typedef unsigned int DWORD;

typedef short SWORD;

typedef unsigned short WORD;

typedef char SBYTE;

typedef unsigned char BYTE;

#pragma pack(push, 1)

struct BMPHeader

{

WORD type = 0;

DWORD size = 0;

WORD reserved1 = 0;

WORD reserved2 = 0;

DWORD offset = 0;

DWORD headerSize = 0;

SDWORD width = 0;

SDWORD height = 0;

WORD colorPlanes = 0;

WORD bpp = 0;

DWORD compression = 0;

DWORD imgSize = 0;

DWORD hrez = 0;

DWORD vres = 0;

DWORD palette = 0;

DWORD imporant = 0;

};

struct Pixel

{

public:

BYTE r = 255;

BYTE g = 255;

BYTE b = 255;

Pixel() :r{ 255 }, g{ 255 }, b{ 255 } {};

Pixel(BYTE rc, BYTE gc, BYTE bc) :r{ rc }, g{ gc }, b{ bc } {};

bool operator ==(const Pixel& other) const { return this->r == other.r && this->g == other.g && this->b == other.b; };

bool operator !=(const Pixel& other) const { return (\*this) != other; };

Pixel(const Pixel& other) : r{ other.r }, g{ other.g }, b{ other.b } {};

Pixel(Pixel&& other) : r{ other.r }, g{ other.g }, b{ other.b } {};

const Pixel& operator=(const Pixel& other)

{

if (this != &other)

{

this->r = other.r;

this->g = other.g;

this->b = other.b;

}

return \*this;

}

Pixel& operator=(Pixel&& other)

{

if (this != &other)

{

this->r = other.r;

this->g = other.g;

this->b = other.b;

}

return \*this;

}

};

#pragma pack(pop)

class MyBMP

{

private:

BMPHeader bmph{};

unsigned int width = 0;

unsigned int height = 0;

PIA::myVector<Pixel> pixels{};

public:

MyBMP() : bmph{}, width{ 0 }, height{ 0 }, pixels{} {};

MyBMP(const char\* path)

{

Load(path);

}

MyBMP(int width, int height)

{

const SDWORD row\_size = width \* 3 + height % 4;

const DWORD bitmap\_size = row\_size \* height;

this->bmph.type = 0x4D42;

this->bmph.size = bitmap\_size + sizeof(BMPHeader);

this->bmph.reserved1 = 0;

this->bmph.reserved2 = 0;

this->bmph.offset = sizeof(BMPHeader);

this->bmph.size = 40;

this->bmph.width = width;

this->bmph.height = height;

this->bmph.colorPlanes = 1;

this->bmph.bpp = sizeof(Pixel) \* 8;

this->bmph.compression = 0;

this->bmph.imgSize = bitmap\_size;

this->bmph.hrez = 0;

this->bmph.vres = 0;

this->bmph.palette = 0;

this->bmph.imporant = 0;

if (!this->pixels.empty()) this->pixels.clear();

this->pixels = PIA::myVector<Pixel>{ this->bmph.imgSize };

}

void Clear()

{

this->pixels.clear();

}

void DoTask()

{

try

{

PIA::myVector<Pixel> colors;

for (int i = 0; i <= 1; i++)

{

for (int j = 0; j <= 1; j++)

{

for (int k = 0; k <= 1; k++)

{

Pixel t = Pixel{0, 0, 0};

if (i) t.r = 255;

if (j) t.g = 255;

if (k) t.b = 255;

if (!(!i && !j && !k || i && j && k))

colors.pushBack(t);

}

}

}

auto ctr = 0;

for (int i = 0; i < this->width; i++)

{

for (int j = 0; j < this->height; j++)

{

Pixel cur = this->pixels[IX(i, j)];

for (int i = 0; i < colors.getSize(); i++)

{

if (cur == colors[i])

{

cur = colors[(i + 1) % colors.getSize()];

ctr++;

break;

}

}

this->Set(i, j, cur);

}

}

std::cout << "Цвет успешно изменен! Обработано " << ctr << " пикселей!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка обработки файла" << std::endl;

}

}

void Save(const char\* filename)

{

try

{

const int row\_size = this->width \* 3 + this->width % 4;

const unsigned int bitmap\_size = row\_size \* this->height;

if (std::ofstream ofs{ filename, std::ios::binary })

{

ofs.write(reinterpret\_cast<const char\*>(&this->bmph), sizeof(BMPHeader));

PIA::myVector<BYTE> line(row\_size);

for (int y = this->height - 1; y >= 0; --y)

{

int i = 0;

for (int x = 0; x < this->width; ++x)

{

const Pixel& color = this->pixels[IX(x, y)];

line[i++] = color.b;

line[i++] = color.g;

line[i++] = color.r;

}

ofs.write(reinterpret\_cast<const char\*>(line.data()), line.getSize());

}

ofs.close();

}

std::cout << "Изображение сохранено!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка сохранения файла!" << std::endl;

}

}

void Load(const char\* filename)

{

try

{

this->pixels.clear();

if (std::ifstream ifs{ filename, std::ios::binary })

{

ifs.read(reinterpret\_cast<char\*>(&this->bmph), sizeof(BMPHeader));

if (this->bmph.type != 0x4D42)

{

ifs.close();

throw BaseE("Wrong file type");

}

this->width = bmph.width;

this->height = bmph.height;

this->pixels.resize(static\_cast<std::size\_t>(this->width) \* static\_cast<std::size\_t>(this->width), Pixel{255, 255, 255});

const int row\_size = this->width \* 3 + this->width % 4;

PIA::myVector<BYTE> line(row\_size);

for (int y = this->height - 1; y >= 0; --y)

{

ifs.read(reinterpret\_cast<char\*>(line.data()), line.getSize());

int i = 0;

for (int x = 0; x < this->width; ++x)

{

Pixel color{};

color.b = line[i++];

color.g = line[i++];

color.r = line[i++];

this->pixels[IX(x, y)] = color;

}

}

ifs.close();

}

std::cout << "Изображение было загружено!" << std::endl;

}

catch (...)

{

std::cout << "Ошибка загрузки файла!" << std::endl;

}

}

size\_t Width() const { return this->width; }

size\_t Height() const { return this->height; }

void Set(const unsigned int x, const unsigned int y, const Pixel& pixel)

{

if (InBounds(x, y)) this->pixels[IX(x, y)] = pixel;

else throw OutOfRangeE("Pixel coordinated out of range");

}

Pixel& Get(const unsigned int x, const unsigned int y)

{

if (!InBounds(x, y)) throw OutOfRangeE("Pixel coordinated out of range");

return this->pixels[IX(x, y)];

}

private:

unsigned int IX(const unsigned int x, const unsigned int y) { return this->width \* y + x; };

bool InBounds(unsigned int x, unsigned int y) const { return x >= 0 && x < this->width && y >= 0 && y < this->height; }

};

Exceptions.h

#pragma once

#include <exception>

#include <iostream>

#include <fstream>

#include <typeinfo>

#include <chrono>

#include <ctime>

#pragma warning(disable : 4996)

class BaseE : std::exception

{

protected:

const char\* message = "";

public:

BaseE()

{

}

BaseE(const char\* msg)

{

try {

std::ofstream fout;

fout.open("exception.txt", std::ios\_base::app);

if (fout.is\_open()) {

fout << "\n";

std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(std::chrono::system\_clock::now());

fout << msg << ' ' << ctime(&end\_time);

}

fout.close();

}

catch (...) {

}

message = msg;

}

virtual const char\* what() const

{

return message;

}

};

class OutOfRangeE : BaseE

{

private:

public:

OutOfRangeE()

{

}

OutOfRangeE(const char\* msg) : BaseE(msg)

{

}

virtual const char\* what() const override

{

return BaseE(message).what();

};

};

template <class T>

class TypeErrorE : BaseE

{

private:

const char\* targetType = typeid(T).name();

public:

TypeErrorE()

{

targetType = typeid(T).name();

}

TypeErrorE(const char\* msg) : BaseE(msg)

{

targetType = typeid(T).name();

}

const char\* GetTargetType() const

{

return targetType;

}

virtual const char\* what() const override

{

return BaseE(message).what();

};

};

class InputErrorE : BaseE

{

private:

public:

InputErrorE()

{

}

InputErrorE(const char\* msg) : BaseE(msg)

{

}

virtual const char\* what() const override

{

return BaseE(message).what();

};

};

class NodeE : BaseE

{

private:

public:

NodeE()

{

}

NodeE(const char\* msg) : BaseE(msg)

{

}

virtual const char\* what() const override

{

return BaseE(message).what();

};

};

**Список литературы**

1. Алексеев В.Е. Графы и алгоритмы. Структуры данных. Модели вычислений [Электронный ресурс]/ В.Е. Алексеев, В.А. Таланов. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 153 c. — Режим доступа: http://www.iprbookshop.ru/52186.html
2. Вирт Никлаус. Алгоритмы и структуры данных [Электронный ресурс]/ Никлаус Вирт— Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 272 c.— Режим доступа: http://www.iprbookshop.ru/63821.html
3. 3Самуйлов С.В. Алгоритмы и структуры обработки данных [Электронный ресурс]: учебное пособие/ С.В. Самуйлов. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2016. — 132 c.— Режим доступа: http://www.iprbookshop.ru/47275.html