Государственное учреждение высшего профессионального образования

«Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

Лабораторная работа №2

по теме “Чтение и запись графических изображений в формате **BMP**.**”**

**Выполнила:**

 Студент 2 курса 3 группы

Дадыченко Артём Сергеевич

Минск 2016

**Ход работы**

**Задание №1**

Написать функцию для отображения (чтения с диска) в заданной области окна изображения в формате BMP.

***int ShowBitMap(HWND hWnd, RECT& r, HBITMAP hBit, int x, int y****)*

**Код реализации:**

int ShowBitMap(HWND hWnd, HANDLE hBit, int x, int y)

//Функция отображает рисунок в заданной позиции окна

//hWnd - дискриптор окна, куда выводится изображение

//r – область в окне, куда выводится изображение

//hBit - дискриптор рисунка

//(x,y) - координаты левого верхнего угла изображения в окне вывода

{

BITMAP BitMap;

GetObjectW(hBit, sizeof(BITMAP), &BitMap);

int Height = BitMap.bmHeight;

int Width = BitMap.bmWidth;

HDC hdc = GetDC(hWnd);

HDC hdcMem = CreateCompatibleDC(hdc);

HBITMAP OldBitmap = (HBITMAP)SelectObject(hdcMem, hBit);

BitBlt(hdc, x, y, Width, Height, hdcMem, 0, 0, SRCCOPY);

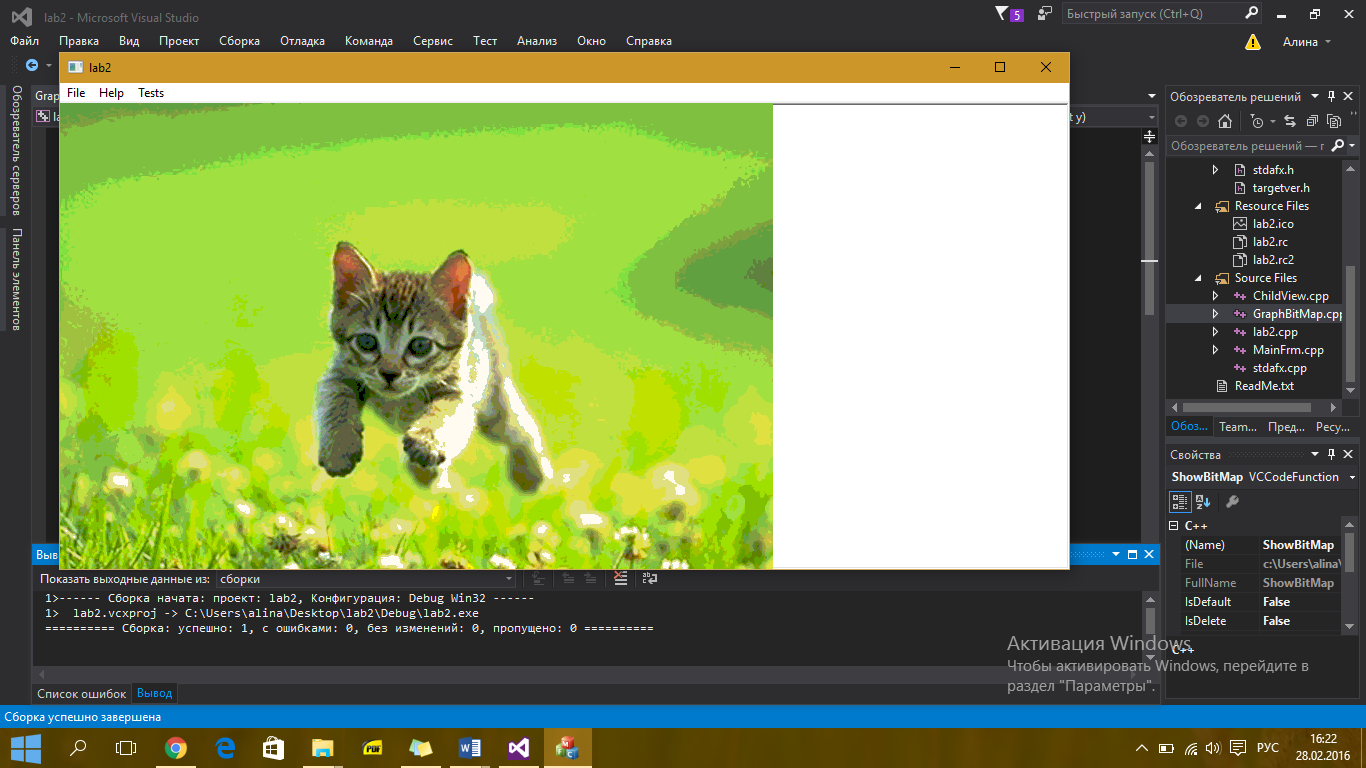
SelectObject(hdcMem, OldBitmap);

ReleaseDC(hWnd, hdc);

return 0;

}

**Результат выполнения функции:**



**Задание №2**

Написать функцию для записи на диск изображения заданной области окна в формате BMP.

**int ClientToBmp(HWND hWnd, RECT& r, char \*Name)**

**Код реализации:**

int ClientToBmp(HWND hWnd, char \*Name)

//Сохранение рабочей области окна в файле Name.bmp

//hWnd - дискриптор окна, рабочая область которого сохраняется

//r – область в окне, которая сохраняется в файле

//Name - имя файла для сохранения

{

RECT r;

GetClientRect (hWnd, &r);

return ClientRectToBmp (hWnd, Name, r);

}

int ClientRectToBmp(HWND hWnd, char\* name, RECT r)

{

HANDLE fh = CreateFile ((LPCWSTR)name, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_SEQUENTIAL\_SCAN, NULL);

if (fh == INVALID\_HANDLE\_VALUE) //не создался

return 2;

BITMAPINFOHEADER bi;//структура содержит описание изображения

ZeroMemory (&bi, sizeof (BITMAPINFOHEADER));//все заполняет нулями

bi.biSize = sizeof (BITMAPINFOHEADER);

bi.biWidth = r.right - r.left;

bi.biHeight = r.bottom - r.top;

bi.biPlanes = 1;

bi.biBitCount = 32; //16 глубина цветов

bi.biSizeImage = (bi.biWidth \* bi.biBitCount + 31)/32\*4\*bi.biHeight;

BITMAPFILEHEADER bmfHdr; //описывает тип файла, размер, смещение области битов

ZeroMemory (&bmfHdr, sizeof (BITMAPFILEHEADER));

bmfHdr.bfType = 0x4D42; //BM ('M'<<8)|'B';

bmfHdr.bfSize = bi.biSizeImage + sizeof (BITMAPFILEHEADER) + bi.biSize;

bmfHdr.bfReserved1 = bmfHdr.bfReserved2 = 0;

bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) + (DWORD)bi.biSize;

HDC hDC = GetDC (hWnd); //контекст изображения

HDC hDCMem = CreateCompatibleDC (hDC); //получаем дискриптор памяти

HBITMAP hBitmap = CreateCompatibleBitmap (hDC, bi.biWidth, bi.biHeight);//создаем битовую карту

HBITMAP oldBitmap = (HBITMAP)SelectObject (hDCMem, hBitmap); // в созданый контекст памяти вносит дискриптор битовой карты

BitBlt (hDCMem, 0, 0, bi.biWidth, bi.biHeight, hDC, r.left, r.top, SRCCOPY); //копирует из памяти картинку, которая в ней находится

hBitmap = (HBITMAP)SelectObject (hDCMem, oldBitmap); //перезаписываем картинку

HANDLE hDIB = GlobalAlloc (GHND /\*GMEM\_FIXED\*/, bi.biSizeImage); //коды цвета в бит формате

char\* lp = (char\*)GlobalLock (hDIB);

GetDIBits (hDC, hBitmap, 0, bi.biHeight, lp, (LPBITMAPINFO)&bi, DIB\_RGB\_COLORS); // не используется?

DWORD dwWritten = sizeof (BITMAPFILEHEADER);//запись файла( заголовочный файл,картинка, карта цветов)

WriteFile(fh, (LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER), &dwWritten, NULL);

dwWritten = sizeof (BITMAPINFOHEADER);

WriteFile (fh, (LPSTR)&bi, sizeof (BITMAPINFOHEADER), &dwWritten, NULL);

dwWritten = bi.biSizeImage;

WriteFile (fh, lp, bi.biSizeImage, &dwWritten, NULL);

GlobalUnlock(hDIB);

GlobalFree(hDIB);

DeleteObject(hBitmap);

//GlobalFree (GlobalHandle (lp));

lp = NULL;

CloseHandle (fh);

ReleaseDC (hWnd, hDCMem);

ReleaseDC (hWnd, hDC);

DeleteDC (hDCMem);

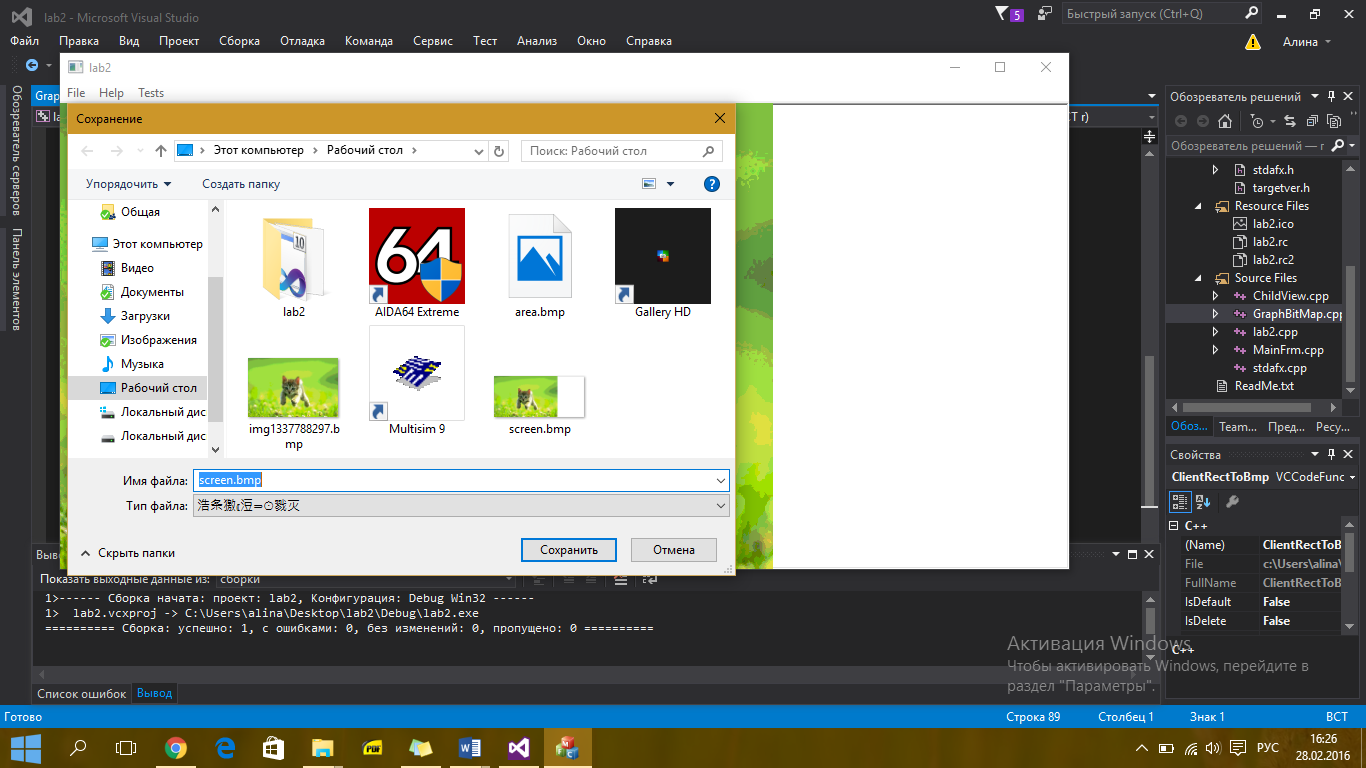
DeleteDC (hDC);

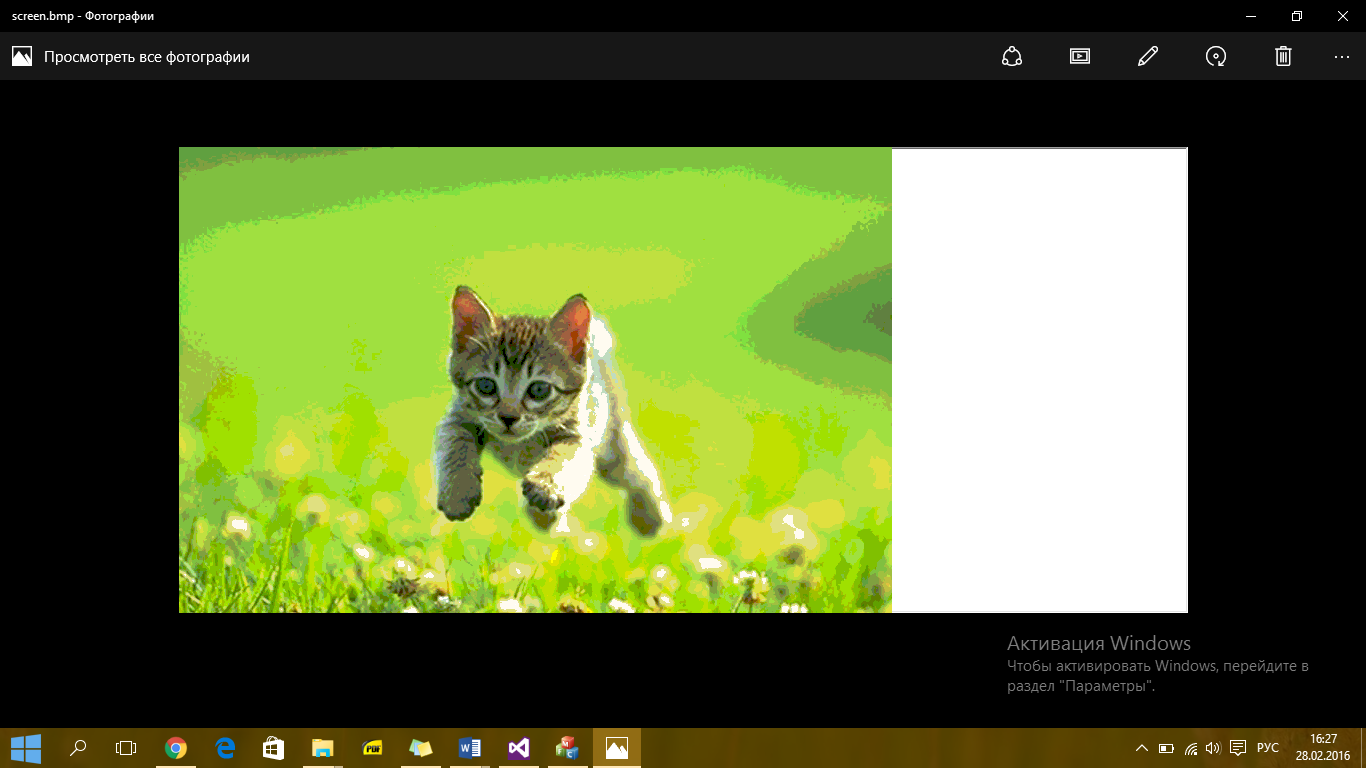
if (dwWritten == 2) return 2;

counter++;

return 0;

**Результат выполнения функции:**





**Приложение**

*Листинг программы:*

#pragma once

#include "stdafx.h"

#include "GraphBitMap.h"

#include <fstream>

int counter = 0;

int ShowBitMap(HWND hWnd, HANDLE hBit, int x, int y)

//Функция отображает рисунок в заданной позиции окна

//hWnd - дискриптор окна, куда выводится изображение

//r – область в окне, куда выводится изображение

//hBit - дискриптор рисунка

//(x,y) - координаты левого верхнего угла изображения в окне вывода

{

BITMAP BitMap;

GetObjectW(hBit, sizeof(BITMAP), &BitMap);

int Height = BitMap.bmHeight;

int Width = BitMap.bmWidth;

HDC hdc = GetDC(hWnd);

HDC hdcMem = CreateCompatibleDC(hdc);

HBITMAP OldBitmap = (HBITMAP)SelectObject(hdcMem, hBit);

BitBlt(hdc, x, y, Width, Height, hdcMem, 0, 0, SRCCOPY);

SelectObject(hdcMem, OldBitmap);

ReleaseDC(hWnd, hdc);

return 0;

}

int ClientToBmp(HWND hWnd, char \*Name)

//Сохранение рабочей области окна в файле Name.bmp

//hWnd - дискриптор окна, рабочая область которого сохраняется

//r – область в окне, которая сохраняется в файле

//Name - имя файла для сохранения

{

RECT r;

GetClientRect (hWnd, &r);

return ClientRectToBmp (hWnd, Name, r);

}

int ClientRectToBmp(HWND hWnd, char\* name, RECT r)

{

HANDLE fh = CreateFile ((LPCWSTR)name, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_SEQUENTIAL\_SCAN, NULL);

if (fh == INVALID\_HANDLE\_VALUE) //не создался

return 2;

BITMAPINFOHEADER bi;//структура содержит описание изображения

ZeroMemory (&bi, sizeof (BITMAPINFOHEADER));//все заполняет нулями

bi.biSize = sizeof (BITMAPINFOHEADER);

bi.biWidth = r.right - r.left;

bi.biHeight = r.bottom - r.top;

bi.biPlanes = 1;

bi.biBitCount = 32; //16 глубина цветов

bi.biSizeImage = (bi.biWidth \* bi.biBitCount + 31)/32\*4\*bi.biHeight;

BITMAPFILEHEADER bmfHdr; //описывает тип файла, размер, смещение области битов

ZeroMemory (&bmfHdr, sizeof (BITMAPFILEHEADER));

bmfHdr.bfType = 0x4D42; //BM ('M'<<8)|'B';

bmfHdr.bfSize = bi.biSizeImage + sizeof (BITMAPFILEHEADER) + bi.biSize;

bmfHdr.bfReserved1 = bmfHdr.bfReserved2 = 0;

bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) + (DWORD)bi.biSize;

HDC hDC = GetDC (hWnd); //контекст изображения

HDC hDCMem = CreateCompatibleDC (hDC); //получаем дискриптор памяти

HBITMAP hBitmap = CreateCompatibleBitmap (hDC, bi.biWidth, bi.biHeight);//создаем битовую карту

HBITMAP oldBitmap = (HBITMAP)SelectObject (hDCMem, hBitmap); // в созданый контекст памяти вносит дискриптор битовой карты

BitBlt (hDCMem, 0, 0, bi.biWidth, bi.biHeight, hDC, r.left, r.top, SRCCOPY); //копирует из памяти картинку, которая в ней находится

hBitmap = (HBITMAP)SelectObject (hDCMem, oldBitmap); //перезаписываем картинку

HANDLE hDIB = GlobalAlloc (GHND /\*GMEM\_FIXED\*/, bi.biSizeImage); //коды цвета в бит формате

char\* lp = (char\*)GlobalLock (hDIB);

GetDIBits (hDC, hBitmap, 0, bi.biHeight, lp, (LPBITMAPINFO)&bi, DIB\_RGB\_COLORS); // не используется?

DWORD dwWritten = sizeof (BITMAPFILEHEADER);//запись файла( заголовочный файл,картинка, карта цветов)

WriteFile(fh, (LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER), &dwWritten, NULL);

dwWritten = sizeof (BITMAPINFOHEADER);

WriteFile (fh, (LPSTR)&bi, sizeof (BITMAPINFOHEADER), &dwWritten, NULL);

dwWritten = bi.biSizeImage;

WriteFile (fh, lp, bi.biSizeImage, &dwWritten, NULL);

GlobalUnlock(hDIB);

GlobalFree(hDIB);

DeleteObject(hBitmap);

//GlobalFree (GlobalHandle (lp));

lp = NULL;

CloseHandle (fh);

ReleaseDC (hWnd, hDCMem);

ReleaseDC (hWnd, hDC);

DeleteDC (hDCMem);

DeleteDC (hDC);

if (dwWritten == 2) return 2;

counter++;

return 0;

}