МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

**(«ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №3

по дисциплине «Технологии программирования»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Щесняк Д. С./

Проверил доцент кафедры ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М. Л./

Киров 2016

1. Задание на лабораторную работу

Используя WinAPI реализовать один родительский и 2 дочерних окна. Родительское окно должно содержать кнопки, при нажатии на которые открывались соотвествующие окна. Одно из дочерних окон должно демонстрировать процесс обучения нейросети, а другое должно позволять протестировать корректность работы нейросети по распознаванию цифр.

1. Словесное описание алгоритма работы

Создается класс окна в котором описывается базовые свойства окна: обработчик, курсор, иконка и тд. Далее на основе этого класса создается само окно при помощи CreateWindow(). На этом этапе полностью создается окно со своим названием, размерами, родительским окном и стилями. Далее запускается цикл получения сообщений. Внутри цикла, в основном потоке, определяется к какому окну пришло сообщение и далее оно передается окну, в котором произошло событие. При получении сообщения, вызывается обработчик сообщений, который в блоке switch обрабатывает сообщения. Работа приложения завершается после прихода последнего сообщения.

1. Экранные формы

Экранные формы представлены на рисунках 1-3.

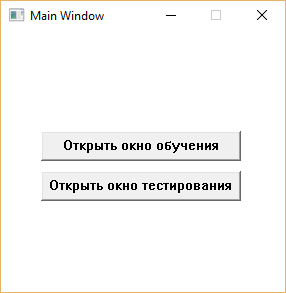


Рисунок 1 – Главное окно программы

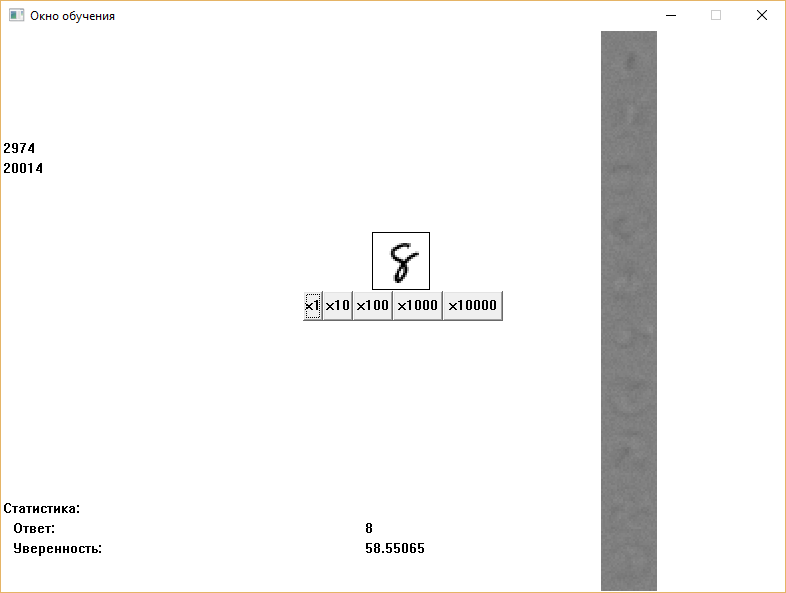


Рисунок 2 – Окно обучения

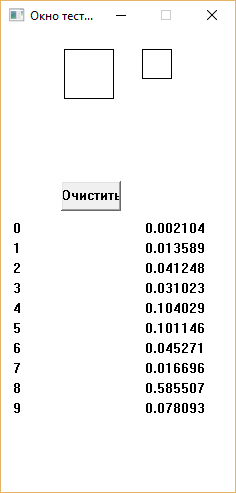


Рисунок 3 – Окно тестирования

1. Исходный код программы

Исходный код программы представлен на рисунке 1.

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <windowsx.h>  #include <windows.h>  #include <stdlib.h>  #include <string>  #include "NeuralNet.h"  using namespace std;  char szClassName[] = "MainClass";  char szTitle[] = "Main Window";  const int nextBut = 0;  const int learnBut = 1;  const int testBut = 2;  const int nextX1But = 3;  const int nextX10But = 4;  const int nextX100But = 5;  const int nextX1000But = 6;  const int nextX10000But = 7;  const int clearBut = 8;  int errors = 0;  int atemps = 0;  bool LBUTTONDOWN = false;  HWND mainWin;  HWND learnWin;  HWND testWin;  template <typename t>  void getData(ifstream &f, t &data) {  data = 0;  for (int i = sizeof(data) - 1; i >= 0; i--) {  data |= f.get() << (8 \* i);  }  }  WNDCLASSEX CreateWinClass(char className[], HINSTANCE hInstance, WNDPROC proc) {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = proc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_APPLICATION));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = NULL;  wcex.lpszClassName = className;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_APPLICATION));  if (!RegisterClassEx(&wcex)) {  MessageBox(NULL, "Can't Register Class", "Error", MB\_OK);  }  return wcex;  }  class ImageBox  {  public:  unsigned char image[28][28];  bool changed;  ImageBox() {  clear();  }  ~ImageBox() {  }  void draw(HDC hdc, int x, int y) {  if (!changed) return;  Rectangle(hdc, x - 1, y - 1, x + 49, y + 49);  byte color;  for (int i = 0; i < 28; i++)  {  for (int j = 0; j < 28; j++)  {  if (image[i][j] != 0) {  color = 255 - image[i][j];  SetPixel(hdc, x + j\*2, y + i\*2, RGB(color, color, color));  SetPixel(hdc, x + j\*2 + 1, y + i\*2, RGB(color, color, color));  SetPixel(hdc, x + j\*2, y + i\*2 + 1, RGB(color, color, color));  SetPixel(hdc, x + j\*2 + 1, y + i\*2 + 1, RGB(color, color, color));  }  }  }  changed = false;  }  void draw28(HDC hdc, int x, int y) {  if (!changed) return;  Rectangle(hdc, x - 1, y - 1, x + 29, y + 29);  byte color;  for (int i = 0; i < 28; i++)  {  for (int j = 0; j < 28; j++)  {  if (image[i][j] != 0) {  color = 255 - image[i][j];  SetPixel(hdc, x + j, y + i, RGB(color, color, color));  }  }  }  }  void clear() {  for (int i = 0; i < 28; i++)  {  for (int j = 0; j < 28; j++)  {  image[i][j] = 0;  }  }  changed = true;  }  byte\* getImage(){  byte\* arr = new byte[28\*28];  for (int i = 0; i < 28\*28; i++)  {  arr[i] = image[i / 28][i % 28];  }  return arr;  }  void onClick(int x, int y) {  int i = y / 2;  int j = x / 2;  image[i][j] = 255;  image[i - 1][j] = 200;  image[i + 1][j] = 200;  image[i][j - 1] = 200;  image[i][j + 1] = 200;  //if ( (i > 0) && (image[i - 1][j] < 192) ) {  // image[i - 1][j] = 192;  //}  //if ( (i < 27) && (image[i + 1][j] < 192) ) {  // image[i + 1][j] = 192;  //}  //if ( (j > 0) && (image[i][j - 1] < 192) ) {  // image[i][j - 1] = 192;  //}  //if ( (j < 27) && (image[i][j + 1] < 192) ) {  // image[i][j + 1] = 192;  //}  changed = true;  }  };  class Image  {  public:  ifstream file;  ifstream labelFile;  int count;  unsigned char image[28][28];  unsigned char label;  Image(char\* filename, char\* labelFilename) {  file.open(filename, ios::binary | ios::in);  file.seekg(4);  getData(file, count);  file.seekg(16);  labelFile.open(labelFilename, ios::binary | ios::in);  labelFile.seekg(8);  next();  }  ~Image() {  file.close();  labelFile.close();  };  void next() {  if (count <= 1) reset();  for(int y = 0; y < 28; y++) {  for (int x = 0; x < 28; x++) {  image[x][y] = 255 - file.get();// +((double)rand() / RAND\_MAX) \* 1.5;  }  }  label = labelFile.get();  count--;  }  void reset() {  file.seekg(4);  getData(file, count);  file.seekg(16);  labelFile.seekg(8);  }  byte\* getImage(){  byte\* arr = new byte[28\*28];  for (int i = 0; i < 28\*28; i++)  {  arr[i] = 255 - image[i / 28][i % 28];  }  return arr;  }  private:  };  Image img("trainBase/images.idx3-ubyte", "trainBase/labels.idx1-ubyte");  ImageBox imgBox = ImageBox();  NeuralNet NN;  HWND learnWindow(HINSTANCE hInstance, HWND parent, int nCmdShow) {  HWND hWnd = CreateWindow("LearnWindowClass", "Окно обучения", WS\_CAPTION | WS\_SYSMENU | WS\_MINIMIZEBOX | SW\_HIDE, CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, parent, NULL, hInstance, NULL);    CreateWindow("button", "x1", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 302, 260, 20, 30, hWnd, (HMENU) nextX1But, hInstance, NULL);  CreateWindow("button", "x10", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 322, 260, 30, 30, hWnd, (HMENU) nextX10But, hInstance, NULL);  CreateWindow("button", "x100", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 352, 260, 40, 30, hWnd, (HMENU) nextX100But, hInstance, NULL);  CreateWindow("button", "x1000", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 392, 260, 50, 30, hWnd, (HMENU) nextX1000But, hInstance, NULL);  CreateWindow("button", "x10000", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 442, 260, 60, 30, hWnd, (HMENU) nextX10000But, hInstance, NULL);  if (!hWnd) {  MessageBox(NULL, "Can't create window", "Error", MB\_OK);  return false;  }  return hWnd;  }  LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {  PAINTSTRUCT ps;  HDC hdc;  int x0 = 372;  int y0 = 202;  switch (msg)  {  case WM\_COMMAND:  {  switch (LOWORD(wParam))  {  case nextX1But:  {  atemps++;  img.next();  int ans = NN.run(img.getImage());  if (ans != img.label) {  NN.learn(img.label);  errors++;  }  break;  }  case nextX10But:  for (int i = 0; i < 10; i++)  {  atemps++;  img.next();  int ans = NN.run(img.getImage());  if (ans != img.label) {  NN.learn(img.label);  errors++;  }  }  break;  case nextX100But:  for (int i = 0; i < 100; i++)  {  atemps++;  img.next();  int ans = NN.run(img.getImage());  if (ans != img.label) {  errors++;  NN.learn(img.label);  }  }  break;  case nextX1000But:  for (int i = 0; i < 1000; i++)  {  atemps++;  img.next();  int ans = NN.run(img.getImage());  if (ans != img.label) {  errors++;  NN.learn(img.label);  }  }  break;  case nextX10000But:  for (int i = 0; i < 10000; i++)  {  atemps++;  img.next();  int ans = NN.run(img.getImage());  if (ans != img.label) {  errors++;  NN.learn(img.label);  }  }  break;  default:  break;  }  InvalidateRect (hWnd, NULL, TRUE);  UpdateWindow (hWnd);  break;  }  case WM\_PAINT: {  hdc = BeginPaint(hWnd, &ps);  Rectangle(hdc, x0 - 1, y0 - 1, x0 + 57, y0 + 57);  for(int y = 0; y < 28; y++) {  for (int x = 0; x < 28; x++) {  SetPixel(hdc, x0 + x\*2, y0 + y\*2, RGB(img.image[x][y], img.image[x][y], img.image[x][y]));  SetPixel(hdc, x0 + x\*2 + 1, y0 + y\*2, RGB(img.image[x][y], img.image[x][y], img.image[x][y]));  SetPixel(hdc, x0 + x\*2, y0 + y\*2 + 1, RGB(img.image[x][y], img.image[x][y], img.image[x][y]));  SetPixel(hdc, x0 + x\*2 + 1, y0 + y\*2 + 1, RGB(img.image[x][y], img.image[x][y], img.image[x][y]));  }  }  NN.draw(hdc, 600, 0);  hdc = GetWindowDC(hWnd);  TextOut(hdc, 10, 500, "Статистика:", 11);  TextOut(hdc, 20, 520, "Ответ:", 6);  TextOut(hdc, 20, 540, "Уверенность:", 12);  TextOut(hdc, 372, 520, std::to\_string(NN.ans).c\_str(), 1);  TextOut(hdc, 372, 540, std::to\_string(NN.ac\*100).c\_str(), std::to\_string(NN.ac).length());  TextOut(GetWindowDC(hWnd), 10, 140, std::to\_string(errors).c\_str(), std::to\_string(errors).length());  TextOut(GetWindowDC(hWnd), 10, 160, std::to\_string(atemps).c\_str(), std::to\_string(atemps).length());  EndPaint(hWnd, &ps);  break;  }  case WM\_DESTROY:  PostQuitMessage(0);  break;  case WM\_CLOSE:  ShowWindow(hWnd, SW\_HIDE);  break;  default:  return DefWindowProc(hWnd, msg, wParam, lParam);  break;  }  return 0;  }  HWND mainWindow(HINSTANCE hInstance, HWND parent, int nCmdShow) {  HWND hWnd = CreateWindow(szClassName, szTitle, WS\_CAPTION | WS\_SYSMENU | WS\_MINIMIZEBOX, CW\_USEDEFAULT, CW\_USEDEFAULT, 300, 300, parent, NULL, hInstance, NULL);  CreateWindow("button", "Открыть окно обучения", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 40, 100, 200, 30, hWnd, (HMENU) learnBut, hInstance, NULL);  CreateWindow("button", "Открыть окно тестирования", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 40, 140, 200, 30, hWnd, (HMENU) testBut, hInstance, NULL);  if (!hWnd) {  MessageBox(NULL, "Can't create window", "Error", MB\_OK);  return false;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return hWnd;  }  LRESULT CALLBACK MainProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {  switch (msg) {  case WM\_COMMAND: {  switch (LOWORD(wParam)) {  case learnBut:  ShowWindow(learnWin, SW\_SHOW);  break;  case testBut:  ShowWindow(testWin, SW\_SHOW);  }  }  default:  return DefWindowProc(hWnd, msg, wParam, lParam);  }  return 0;  }  HWND testWindow(HINSTANCE hInstance, HWND parent, int nCmdShow) {  HWND hWnd = CreateWindow("TestWindowClass", "Окно тестирования", WS\_CAPTION | WS\_SYSMENU | WS\_MINIMIZEBOX | SW\_HIDE, CW\_USEDEFAULT, CW\_USEDEFAULT, 250, 500, parent, NULL, hInstance, NULL);  CreateWindow("button", "Очистить", WS\_CHILD | BS\_PUSHBUTTON | WS\_VISIBLE, 60, 150, 60, 30, hWnd, (HMENU) clearBut, hInstance, NULL);  imgBox = ImageBox();  if (!hWnd) {  MessageBox(NULL, "Can't create window", "Error", MB\_OK);  return false;  }  return hWnd;  }  LRESULT CALLBACK TestProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {  PAINTSTRUCT ps;  HDC hdc = GetWindowDC(hWnd);  int x, y;  switch (msg)  {  case WM\_MOVE:  imgBox.changed = true;  break;  case WM\_PAINT:  imgBox.draw28(hdc, 150, 50);  imgBox.draw(hdc, 72, 50);  for (int i = 0; i < 10; i++)  {  TextOut(hdc, 20, 220 + i\*20, std::to\_string(i).c\_str(), 1);  TextOut(hdc, 152, 220 + i\*20, std::to\_string(NN.out[i]).c\_str(), std::to\_string(NN.out[i]).length());  }  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  case WM\_LBUTTONDOWN:  LBUTTONDOWN = true;  x = GET\_X\_LPARAM(lParam) + 9;  y = GET\_Y\_LPARAM(lParam) + 50 - 18;  if ( (x > 72) && (x < 72 + 48) && (y > 50) && (y < 50 + 48) ) {  imgBox.onClick(x - 72, y - 50);  }  break;  case WM\_LBUTTONUP:  LBUTTONDOWN = false;  NN.run(imgBox.getImage());  break;  case WM\_MOUSEMOVE:  if (LBUTTONDOWN) {  x = GET\_X\_LPARAM(lParam) + 9;  y = GET\_Y\_LPARAM(lParam) + 50 - 18;  if ( (x > 72) && (x < 72 + 48) && (y > 50) && (y < 50 + 48) ) {  imgBox.onClick(x - 72, y - 50);  }  }  break;  case WM\_CLOSE:  ShowWindow(hWnd, SW\_HIDE);  break;  case (WM\_COMMAND):  switch (LOWORD(wParam)) {  case clearBut:  //imgBox = ImageBox();  imgBox.clear();  break;  }  break;  default:  return DefWindowProc(hWnd, msg, wParam, lParam);  break;  }  return 0;  }  int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow) {  NN.run(img.getImage());  WNDCLASSEX mainClass = CreateWinClass(szClassName, hInstance, MainProc);  WNDCLASSEX learnWindowClass = CreateWinClass("LearnWindowClass", hInstance, WndProc);  WNDCLASSEX testWindowClass = CreateWinClass("testWindowClass", hInstance, TestProc);  mainWin = mainWindow(hInstance, NULL, nCmdShow);  learnWin = learnWindow(hInstance, mainWin, nCmdShow);  testWin = testWindow(hInstance, mainWin, nCmdShow);  MSG msg;  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return 0;  } |

Рисунок 1 – Исходный код программы

1. Вывод

В ходе выполнения лабораторной работы были полученные необходимые знания базового синтаксиса языка программирования С++. Была изучена работа с WinAPI. Данные знания являются фундаментальными и необходимы для дальнейшего продолжения изучения различных технологий программирования и языка С++.