Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №4**по дисциплине «Изучение взаимодействия управляемого и неуправляемого кода»

Выполнил:  
Студент Альков В.С.  
Группа И407Б  
  
Преподаватель:  
Кимсанбаев К.А.

Санкт-Петербург  
2021 г.

*Постановка задачи:*

Работа состоит из двух частей - первая посвящена работе с неуправляемым кодом, вторая – графике.

**Первая часть**

В первой части необходимо создать с использованием C или C++ динамическую,библиотеку для работы с массивом, которая позволяет вводить массив, выводить массив и выполнять над ним действие, согласно индивидуальному варианту. Далее необходимо подключить её к программе, написанной на C# и продемонстрировать её возможности, соответствующим образом учтя разницу в используемых типах данных.

В вещественном массиве В (67) заменить все отрицательные числа их квадратами.

**Вторая часть**

Нарисуйте кота или кошку. Программным способом. Использовать готовые изображения нельзя.

**Первая часть:**

Листинг программы на C++:

SquareNegative.h

#pragma once

#ifdef SQUARENEGATIVE\_EXPORTS

#define SQUARENEGATIVE\_API \_\_declspec(dllexport)

#else

#define SQUARENEGATIVE\_API \_\_declspec(dllimport)

#endif

extern "C" SQUARENEGATIVE\_API void SquareNegative(double\* arr, int size);

extern "C" SQUARENEGATIVE\_API void ReadArray(double\* arr, int size);

extern "C" SQUARENEGATIVE\_API void PrintArray(double\* arr, int size);

extern "C" SQUARENEGATIVE\_API double getValue();

SquareNegative.cpp

#include "pch.h"

#include "SquareNegative.h"

#include <iostream>

void SquareNegative(double\* arr, int size)

{

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

if (arr[i] < 0)

arr[i] \*= arr[i];

}

void ReadArray(double\* arr, int size)

{

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

arr[i] = getValue();

}

double getValue()

{

double a;

while (true)

{

system("cls");

std::cin >> a;

if (std::cin.fail())

{

std::cout << "wrong input\n";

std::cin.clear();

std::cin.ignore(32767, '\n');

system("pause");

}

else

return a;

}

}

void PrintArray(double\* arr, int size)

{

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

std::cout<< arr[i] <<" ";

std::cout <<"\n";

}

Листинг программы на C#:

using System;

using System.Runtime.InteropServices;

namespace ConsoleApplication1

{

class Program

{

[DllImport("SquareNegative.dll", CallingConvention = CallingConvention.Cdecl)]

//public static extern void SquareNegative(IntPtr arr, int size);

public static extern void SquareNegative(double[] arr, int size);

[DllImport("SquareNegative.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern void ReadArray(double[] arr, int size);

[DllImport("SquareNegative.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern void PrintArray(double[] arr, int size);

static void Main(string[] args)

{

int count = 0;

bool flag = false;

while(!flag)

{

Console.Write("Введите кол-во элементов массива (>=0): ");

flag = int.TryParse(Console.ReadLine(), out count);

if (count < 0)

flag = false;

}

double[] arr = new double[count];

ReadArray(arr, count);

SquareNegative(arr, count);

Console.WriteLine("Результирующий массив: ");

PrintArray(arr, count);

}

}

}

Результаты работы программы:







**Вторая часть:**

Листинг программы:

**Form.cs**

using System.Drawing;

using System.Windows.Forms;

namespace CatLast

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Pen pen = new Pen(Color.Black, 2);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 221, 190, 224, 162, 240, 165, 240, 191);

e.Graphics.DrawBezier(pen, 221, 190, 229, 231, 244, 191, 240, 191);

e.Graphics.DrawBezier(pen, 290, 190, 297, 164, 307, 168, 312, 192);

e.Graphics.DrawBezier(pen, 290, 190, 297, 164, 307, 168, 312, 192);

e.Graphics.DrawBezier(pen, 290, 190, 293, 218, 304, 211, 312, 192);

e.Graphics.DrawBezier(pen, 290, 190, 297, 164, 307, 168, 312, 192);

e.Graphics.DrawBezier(pen, 284, 166, 299, 159, 311, 163, 324, 180);

e.Graphics.DrawBezier(pen, 240, 162, 217, 151, 207, 166, 199, 183);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 170, 166, 109, 75, 195, 32, 238, 126);

e.Graphics.DrawBezier(pen, 186, 150, 160, 112, 172, 61, 223, 131);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 301, 131, 334, 41, 431, 89, 362, 170);

e.Graphics.DrawBezier(pen, 316, 138, 341, 88, 392, 94, 350, 162);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 252, 204, 251, 222, 282, 223, 278, 205);

e.Graphics.DrawBezier(pen, 252, 204, 261, 207, 267, 205, 278, 205);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 266, 218, 264, 222, 265, 235, 267, 241);

e.Graphics.DrawBezier(pen, 134, 205, 223, 96, 301, 97, 388, 204);

e.Graphics.DrawBezier(pen, 199, 209, 231, 259, 307, 250, 322, 214);

e.Graphics.DrawBezier(pen, 129, 202, 222, 323, 334, 306, 388, 204);

e.Graphics.DrawBezier(pen, 123, 241, 151, 221, 186, 207, 240, 219);

e.Graphics.DrawBezier(pen, 137, 262, 162, 234, 194, 213, 241, 225);

e.Graphics.DrawBezier(pen, 150, 275, 170, 243, 198, 231, 244, 230);

e.Graphics.DrawBezier(pen, 402, 258, 377, 221, 340, 216, 286, 224);

e.Graphics.DrawBezier(pen, 389, 268, 370, 228, 330, 226, 288, 227);

e.Graphics.DrawBezier(pen, 380, 277, 366, 239, 329, 236, 286, 233);

e.Graphics.DrawBezier(pen, 190, 475, 158, 403, 161, 359, 205, 268);

e.Graphics.DrawBezier(pen, 190, 475, 186, 448, 237, 442, 245, 481);

e.Graphics.DrawBezier(pen, 187, 477, 205, 483, 221, 476, 245, 481);

e.Graphics.DrawBezier(pen, 238, 316, 205, 424, 242, 435, 245, 481);

e.Graphics.DrawBezier(pen, 210, 457, 197, 464, 205, 465, 202, 479);

e.Graphics.DrawBezier(pen, 219, 456, 222, 473, 216, 467, 219, 479);

e.Graphics.DrawBezier(pen, 285, 299, 279, 403, 286, 419, 296, 467);

e.Graphics.DrawBezier(pen, 347, 465, 329, 419, 296, 435, 296, 467);

e.Graphics.DrawBezier(pen, 347, 465, 328, 466, 308, 466, 296, 467);

e.Graphics.DrawBezier(pen, 313, 442, 319, 454, 305, 457, 306, 465);

e.Graphics.DrawBezier(pen, 322, 443, 319, 454, 329, 460, 320, 465);

e.Graphics.DrawBezier(pen, 323, 297, 327, 386, 330, 404, 347, 465);

e.Graphics.DrawBezier(pen, 165, 364, 108, 322, 81, 358, 122, 432);

e.Graphics.DrawBezier(pen, 168, 368, 108, 322, 81, 358, 122, 432);

e.Graphics.DrawBezier(pen, 170, 448, 147, 402, 125, 416, 115, 451);

e.Graphics.DrawBezier(pen, 180, 449, 156, 404, 125, 416, 116, 448);

e.Graphics.DrawBezier(pen, 180, 449, 160, 451, 136, 452, 116, 448);

e.Graphics.DrawBezier(pen, 138, 426, 136, 434, 122, 435, 130, 449);

e.Graphics.DrawBezier(pen, 149, 426, 146, 437, 134, 439, 153, 447);

e.Graphics.DrawBezier(pen, 288, 453, 267, 453, 242, 454, 241, 452);

e.Graphics.DrawBezier(pen, 407, 448, 441, 361, 387, 374, 333, 259);

e.Graphics.DrawBezier(pen, 407, 448, 439, 360, 374, 359, 351, 366);

e.Graphics.DrawBezier(pen, 363, 430, 303, 390, 366, 359, 351, 366);

e.Graphics.DrawBezier(pen, 356, 447, 357, 405, 418, 426, 406, 451);

e.Graphics.DrawBezier(pen, 356, 447, 373, 450, 397, 443, 406, 451);

e.Graphics.DrawBezier(pen, 389, 431, 378, 432, 382, 448, 388, 448);

e.Graphics.DrawBezier(pen, 375, 427, 363, 437, 376, 447, 367, 442);

e.Graphics.DrawBezier(pen, 408, 364, 427, 374, 484, 381, 457, 250);

e.Graphics.DrawBezier(pen, 494, 172, 443, 172, 453, 228, 457, 250);

e.Graphics.DrawBezier(pen, 494, 172, 524, 189, 514, 197, 492, 218);

e.Graphics.DrawBezier(pen, 494, 172, 524, 189, 514, 197, 492, 218);

e.Graphics.DrawBezier(pen, 478, 373, 501, 269, 473, 245, 492, 218);

e.Graphics.DrawBezier(pen, 478, 373, 462, 402, 440, 410, 418, 401);

e.Graphics.DrawBezier(pen, 354, 446, 359, 446, 347, 451, 339, 445);

e.Graphics.DrawBezier(pen, 354, 446, 359, 446, 347, 451, 339, 445);

}

}

}

**Designer.cs**

namespace CatLast

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.SuspendLayout();

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Name = "Form1";

this.Text = "Form1";

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.ResumeLayout(false);

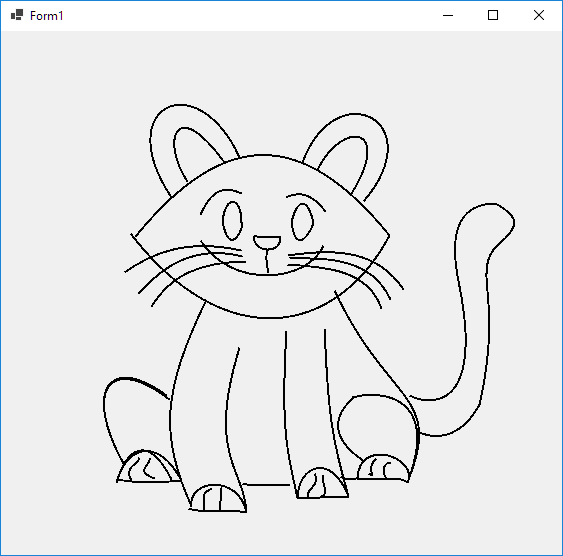
}

#endregion

}

}

Результаты работы программы:



Вывод:

В данной практической работе мы изучили взаимодействие неупрявляемого и управляемого кода в языке C#, написали динамическую библиотеку для работы с массивом на C++ и подключили ее к программе на языке C#.

Использовали во второй части платформу пользовательского интерфейса Windows Forms для отображение рисунка, заданного линиями, не используя готовые изображения.