Московский авиационный институт

(национальный исследовательский университет)

**Институт информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу ООП**

Работу выполнил:

Группа: М8О-203Б-19

ФИО: Авдеев Денис Иванович

Руководитель: Кузнецова С. В.

Содержание

[Тема работы 1](#_Toc59484680)

[Задача 1](#_Toc59484681)

[Программное обеспечение 1](#_Toc59484682)

[Аппаратное обеспечение 1](#_Toc59484683)

[Описание алгоритма на псевдокоде 2](#_Toc59484684)

[Код программы 2](#_Toc59484685)

[Функциональные возможности и интерфейс 10](#_Toc59484686)

[Выводы 15](#_Toc59484687)

# Тема работы

Тема данной работы: Эволюционная стратегия (The (µ, l) Evolution Strategy). Вариант 18 из списка алгоритмов.

Эволюционные стратегии (ЭС) (Evolution Strategies, ES) были разработаны Инго Рехенбергом и Хансом-Паулем Швефелем в Берлинском техническом университете в середине 1960-х. ЭС используют простую процедуру селекции, называемую селекцией усечением (Truncation Selection), а в качестве оператора для улучшения особей применяется (обычно) только мутация.

# Задача

Задачей курсового проекта является разработка приложения, визуализирующего процесс работы эволюционной стратегии, а именно его пошагового кодирования с элементами анимации для большей наглядности.

# Программное обеспечение

Семейство ОС: Windows

Операционная система: Windows 10 LTSC (Enterprise)

Среда разработки: Microsoft Visual Studio Enterprise 2019

Версия: 16.8.3

# Аппаратное обеспечение

Процессор: Intel Core i7-770K 4,9ГГц

ОП: 2 X (DIMM1: Kingston HyperX KHX2133C14/8G) 16 ГБ

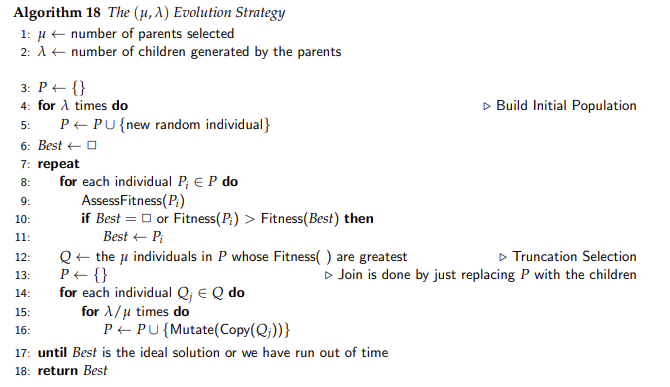
Дисковые устройства:

1. INTEL SSDSC2KW240H6 256 ГБ
2. WDC WD1003FZEX-00K3CA0 1024 ГБ
3. WDC WD800JD-00JRA0 80 ГБ

Монитор: 1920x1080 ViewSonic VA2413wm

# Описание алгоритма на псевдокоде

Говоря по-простому, m – это количество выживающих родительских особей, а λ– количество потомков, создаваемых m родителями. Заметим, что λ должно быть кратно m. На практике для обозначения ЭС обычно используются конкретные значения m и λ. Например, при m = 5 и λ = 20 имеем эволюционную стратегию (5, 20). Вот псевдокод алгоритма:



# Код программы

Общий код программы условно состоит из нескольких объектов. Это обусловлено проектированием по схеме MVC. Таким образом необходимо описать 3 объекта: модель (Models.cs), контроллер (Controllers.cs) и виды (HerosView.cs) и (InputView.cs).

**Models.cs**  
  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EvolutionStrategy

{

public class Models

{

public class HeroModel

{

public int intellegent;

public int strength;

public int agility;

public HeroModel parent1;

public HeroModel parent2;

public string name;

public int resultPower;

public HeroModel()

{

}

public HeroModel(int intellegent, int strength, int agility, string name)

{

this.intellegent = intellegent;

this.strength = strength;

this.agility = agility;

this.name = name;

}

public HeroModel(int intellegent, int strength, int agility, HeroModel parent1, HeroModel parent2, string name)

{

this.intellegent = intellegent;

this.strength = strength;

this.agility = agility;

this.parent1 = parent1;

this.parent2 = parent2;

this.name = name;

}

}

}

}

**Controllers.cs**  
  
using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EvolutionStrategy

{

class Controller

{

public static bool IsInputValid(InputView form)

{

int parents, childes;

try

{

parents = int.Parse(form.Controls["textBox1"].Text);

childes = int.Parse(form.Controls["textBox2"].Text);

if (parents < childes)

{

return false;

}

if (parents > 8)

{

return false;

}

return true;

}

catch (Exception)

{

return false;

}

}

public static List<Models.HeroModel> CreateHeroList(int heroCount)

{

List<Models.HeroModel> heroList = new List<Models.HeroModel>();

var rand = new Random();

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

{

heroList.Add(new Models.HeroModel(rand.Next(0, 11), rand.Next(0, 11), rand.Next(0, 11), "Герой " + i.ToString()));

}

return heroList;

}

public static void SetResultPowers(List<Models.HeroModel> heroList)

{

foreach (Models.HeroModel hero in heroList)

{

hero.resultPower = hero.intellegent \* 5 + hero.agility \* 2 + hero.strength \* 3;

}

}

public static void DrawParents(HerosView form, List<Models.HeroModel> heroList)

{

int i = 0;

foreach (Models.HeroModel hero in heroList)

{

form.Controls.Add(new Label() {

Name = "heroTitle" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 80),

Text = hero.name + ":",

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "heroIntLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 110),

Text = "Интеллект: " + hero.intellegent.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "heroStrLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 130),

Text = "Сила: " + hero.strength.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "heroAgilLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 150),

Text = "Ловкость: " + hero.agility.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "heroPwrLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 180),

Text = "Итого: " + hero.resultPower.ToString(),

AutoSize = true,

});

i++;

}

}

public static async void DrawChilds(HerosView form, List<Models.HeroModel> heroList , List<Models.HeroModel> parentsList)

{

int i = 0;

foreach (Models.HeroModel hero in heroList)

{

form.Controls.Add(new Label()

{

Name = "childTitle" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 240),

Text = "Ребенок " + i.ToString() + ":",

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "childIntLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 270),

Text = "Интеллект: " + hero.intellegent.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "childStrLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 290),

Text = "Сила: " + hero.strength.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "childAgilLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 310),

Text = "Ловкость: " + hero.agility.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "childPwrLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 330),

Text = "Итого: " + hero.resultPower.ToString(),

AutoSize = true,

});

form.Controls.Add(new Label()

{

Name = "childPrntsLabel" + i.ToString(),

Location = new Point(13 + (600 / heroList.Count) \* i, 360),

Text = "Родители:\n" + hero.parent1.name + " и " + hero.parent2.name,

AutoSize = true,

});

i++;

await Task.Delay(1500);

}

form.label2.Text += Controller.FindMiddleResult(heroList) + " против " + Controller.FindMiddleResult(parentsList);

form.label2.Visible = true;

form.button3.Visible = true;

}

public static List<Models.HeroModel> Evolve(List<Models.HeroModel> parentList, int childsCount)

{

List<Models.HeroModel> childeList = new List<Models.HeroModel>();

int newInt, newStr, newAgil, rand1, rand2;

Models.HeroModel parent1, parent2;

var rand = new Random();

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

{

rand1 = rand.Next(0, parentList.Count);

rand2 = rand.Next(0, parentList.Count);

if ((rand1 == rand2) && (rand2 == parentList.Count - 1))

{

rand2--;

}

else if (rand1 == rand2)

{

rand2++;

}

parent1 = parentList[rand1];

parent2 = parentList[rand2];

newInt = parent1.intellegent > parent2.intellegent ? parent1.intellegent : parent2.intellegent;

newStr = parent1.strength > parent2.strength ? parent1.strength : parent2.strength;

newAgil = parent1.agility > parent2.agility ? parent1.agility : parent2.agility;

childeList.Add(new Models.HeroModel(newInt, newStr, newAgil, parent1, parent2, "Ребенок " + i.ToString()));

}

return childeList;

}

public static double FindMiddleResult(List<Models.HeroModel> heroList)

{

double result = 0.0;

foreach (Models.HeroModel hero in heroList)

{

result += hero.resultPower;

}

result /= heroList.Count;

return result;

}

}

}

**HerosView.cs**  
  
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace EvolutionStrategy

{

public partial class HerosView : Form

{

List<Models.HeroModel> parentsList;

List<Models.HeroModel> childsList;

int parentsCount;

int childsCount;

public HerosView()

{

InitializeComponent();

}

public HerosView(List<Models.HeroModel> heroList, int parentsCount, int childsCount)

{

InitializeComponent();

this.parentsList = heroList;

this.parentsCount = parentsCount;

this.childsCount = childsCount;

}

private void button1\_Click(object sender, EventArgs e)

{

this.button1.Hide();

Controller.SetResultPowers(this.parentsList);

Controller.DrawParents(this, this.parentsList);

this.label1.Text = "Вот ваши герои. Хотите посмотреть, во что они эволюционируют?";

this.Controls.Add(new Button()

{

Name = "button2",

Location = new Point(200, 220),

ForeColor = this.button1.ForeColor,

Font = this.button1.Font,

BackColor = this.button1.BackColor,

Size = this.button1.Size,

Text = this.button1.Text,

});

this.Controls["button2"].Click += (sender\_, e\_) => {

this.Controls["button2"].Hide();

this.childsList = Controller.Evolve(this.parentsList, childsCount);

Controller.SetResultPowers(childsList);

Controller.DrawChilds(this, childsList, parentsList);

};

}

private void button3\_Click(object sender, EventArgs e)

{

InputView nextForm = new InputView();

this.Controls.Clear();

this.Hide();

nextForm.Show();

}

}

}

**InputView.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace EvolutionStrategy

{

public partial class InputView : Form

{

List<Models.HeroModel> herosList;

public InputView()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (!Controller.IsInputValid(this))

{

this.label1.Text = "Введите корректные данные (μ должно быть не меньше l):";

}

else

{

int parentsCount = int.Parse(textBox1.Text);

int childsCount = int.Parse(textBox2.Text);

herosList = Controller.CreateHeroList(parentsCount);

HerosView nextForm = new HerosView(herosList, parentsCount, childsCount);

this.Hide();

nextForm.Show();

}

}

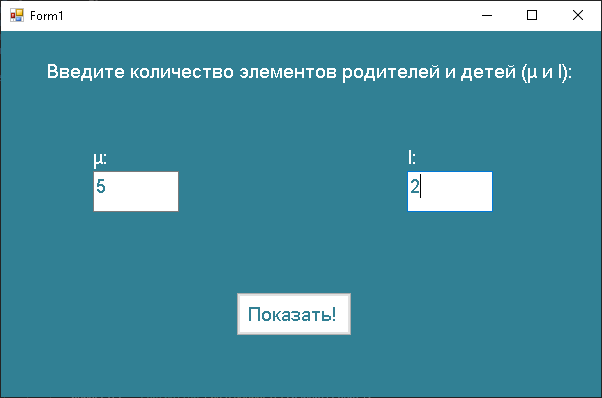
}

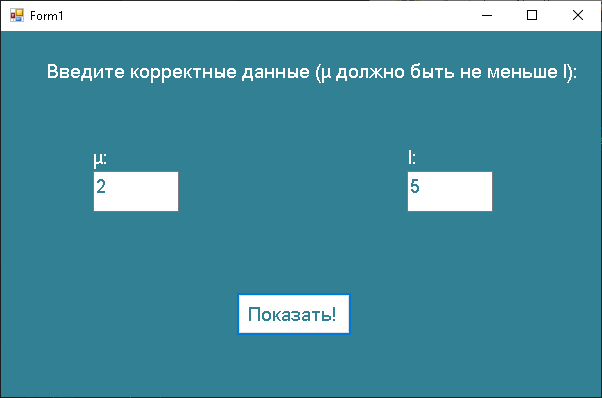
}

# Функциональные возможности и интерфейс

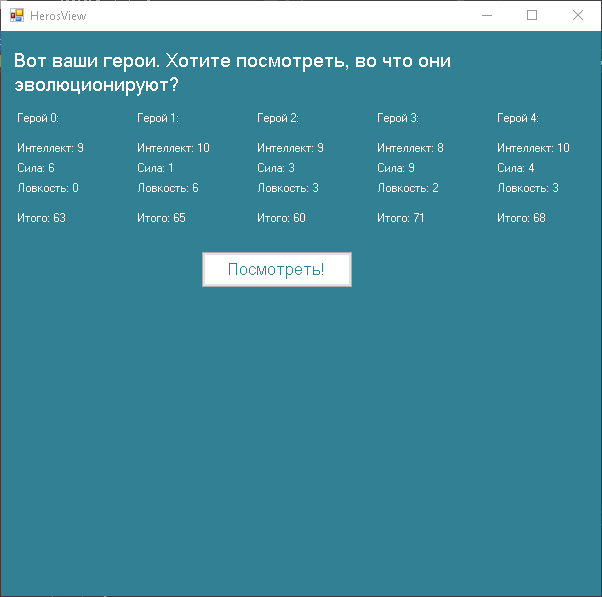
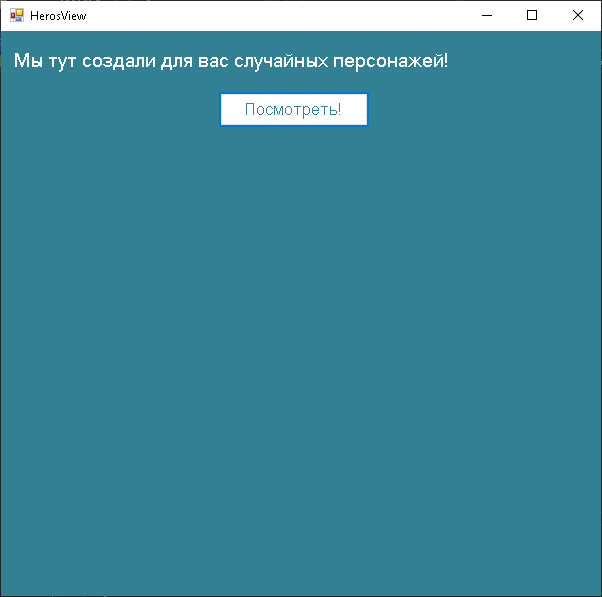
Несколько тестовых примеров для демонстрации работоспособности и возможностей приложения.

Входное числа: 5 и 2

На первой форме мы задаём количество родителей и детей.

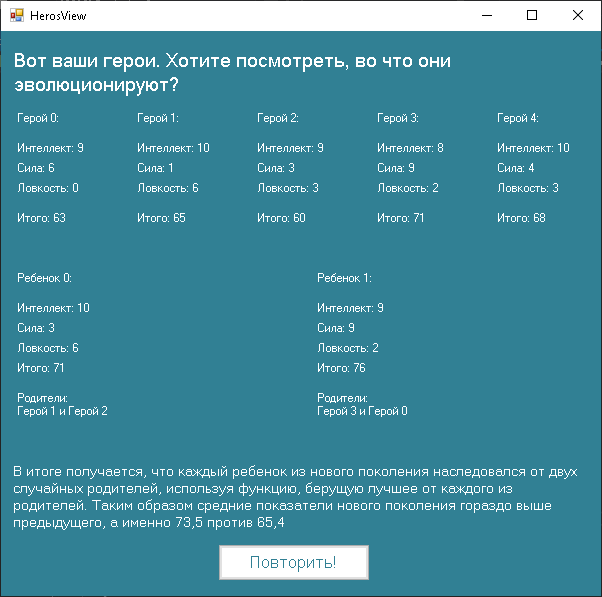
Если введено детей больше, чем родителей или введены символы, программа выдаёт ошибку.

Дальше мы переходим в новую форму в которой по введённым пользователем данным создаются родители со случайными атрибутами (сила, ловкость , интеллект)



Дальше мы с помощью алгоритма “стратеги эволюции” создаём детей, выбирая случайным образом родителей. От родителей наследуются лучшие атрибуты.

Так же выводим на сколько показатели родителей отличаются от показателей детей.



# 

# Выводы

Задача данного курсового проекта во многом потребовала применить объектно-ориентированный подход к созданию графического интерфейса. Обращаясь к конкретике, было необходимо распределить все функции между частями программы с целью получения возможности при необходимости добавить совершенно новый интерфейс. Это и обусловило использование MVC при проектировании. Как можно заметить по коду, все необходимые объекты были успешно реализованы: Модель, Контроллер и Вид. Таким образом выполнить задачу удалось.  
В ходе выполнения было реализовано эволюция стратегии.  
В обязательном порядке была проведена откладка