**Лабораторная работа №10.2**

**Постановка задачи:**

1. Создать класс, который является некоторой произвольной расой в игре. Объявить не менее десяти полей-характеристик для данного класса, используя различные спецификаторы доступа. Реализовать механизмы ввода и вывода данных для одного персонажа этого класса.

*Пример: Класс Псайдак с полями: имя, пол, вес, рост, цвет, сила, физмастерство, уклонение, лечение, арсенал, боевая способность (разгромный удар), не боевая способность (телекинез) и другие.*

1. Для предыдущего задания реализовать все возможные типы конструкторов (не менее трех). Реализовать свойства, позволяющие осуществить проверку на допустимые значения для данных класса. Создать массив персонажей данного класса. Каждый персонаж должен быть уникальным. Вывести этот массив на консоль.
2. Для предыдущего задания реализовать не менее шести **полезных** методов для работы с заданным классом. Также реализовать методы сортировки по каждому имеющемуся полю, вывода максимального и минимального значения поля (где таковое возможно).

*Например, поиск оптимального удара среди ударов ногой, рукой, головой (максимальный урон от удара, который придется соперника) у каждого персонажа этой расы. Методы для расчета ударов ногой, рукой и головой должны быть отдельные и свои.*

**Код:**

using System;

using System.Collections;

namespace Test\_2

{

class People : IComparable

{

private string name;

private string type;

private uint age;

private int hp;

private int stamina;

private int magic;

private int force;

private int heal;

private int level;

private int iq;

private static bool isAlive;

public string Name { get { return name; } set { name = value; } }

public bool IsAlive { get { return isAlive; } set { isAlive = value; } }

public People(string name, uint age, string type)

{

this.name = name; this.age = age; this.type = type;

hp = 0; stamina = 0; magic = 0; force = 0; heal = 0; level = 0; iq = 0; IsAlive = true;

CheckingAge();

CheckingType();

}

public People()

{

name = "Валерий"; age = 54; type = "маг"; IsAlive = true;

hp = 0; stamina = 0; magic = 0; force = 0; heal = 0; level = 0; iq = 0; IsAlive = true;

CheckingAge();

CheckingType();

}

int IComparable.CompareTo(object obj)

{

People p = (People)obj;

return String.Compare(age.ToString(), p.age.ToString());

}

public void CheckingAge()

{

if (age < 30)

{

hp += 20; stamina += 30; magic += 15; force += 20; heal += 10; iq += 80;

}

else

{

hp += 15; stamina += 20; magic += 30; force += 10; heal += 5; iq += 100;

}

}

public void CheckingType()

{

if (type == "мечник")

{

hp += 50; stamina += 20; magic += 5; force += 30; heal += 10; iq += 10;

}

else if (type == "маг")

{

hp += 35; stamina += 10; magic += 50; force += 10; heal += 5; iq += 30;

}

else if (type == "вор")

{

hp += 30; stamina += 40; magic += 5; force += 20; heal += 10; iq += 20;

}

}

public void Battle()

{

hp -= 25 + heal; stamina -= 20; magic -= 20; force += 5; level++; iq++;

if ((hp <= 0) || (stamina <= 0) || (magic <= 0))

{

Console.WriteLine("Смэрть персонажа по имени " + name + "!");

IsAlive = false;

}

}

public void Waiting()

{

hp += 10 + heal; stamina += 20; magic += 20; level++; iq++;

}

public void DisplaySortChoice()

{

Console.Clear();

Console.WriteLine("Выберите параметр для сортировки.");

Console.WriteLine($"\n1. Возраст\n2. HP\n3. Выносливость\n4. Магия\n5. Сила\n6. Уровень\n7. Интеллект\n");

}

public void DisplayParams()

{

Console.WriteLine($"\nИмя: {name} \nВозраст: {age} \nТип: {type} \nHP: {hp} \nВыносливость: {stamina} \nМагия: {magic} \nСила: {force} \nУровень: {level} \nИнтеллект: {iq} \n");

}

public void DisplayCharacters(int i)

{

Console.Write(i + 1 + ".\n");

DisplayParams();

}

public void HandlingInstruction()

{

Console.WriteLine("-------------------------");

Console.WriteLine("Инструкция по управлению! \n0. Показать инструкцию повторно. \n1. Выбрать созданного персонажа. \n2. Ожидание. \n3. В бой! \n4. Статистика всех персонажей. \n5. Статистика текущего персонажа. \n6. Cортировка по возрастанию. \n7. Сортировка по убыванию\n10. Выход.");

Console.WriteLine("-------------------------");

}

//Куча вспомогательных классов для сортировки массива по конкретному полю с использованием интерфейса IComparable

private class SortAgeAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.age > p2.age)

return 1;

if (p1.age < p2.age)

return -1;

else

return 0;

}

}

private class SortAgeDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.age < p2.age)

return 1;

if (p1.age > p2.age)

return -1;

else

return 0;

}

}

private class SortHPAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.hp > p2.hp)

return 1;

if (p1.hp < p2.hp)

return -1;

else

return 0;

}

}

private class SortHPDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.hp < p2.hp)

return 1;

if (p1.hp > p2.hp)

return -1;

else

return 0;

}

}

private class SortStaminaAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.stamina > p2.stamina)

return 1;

if (p1.stamina < p2.stamina)

return -1;

else

return 0;

}

}

private class SortStaminaDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.stamina < p2.stamina)

return 1;

if (p1.stamina > p2.stamina)

return -1;

else

return 0;

}

}

private class SortMagicAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.magic > p2.magic)

return 1;

if (p1.magic < p2.magic)

return -1;

else

return 0;

}

}

private class SortMagicDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.magic < p2.magic)

return 1;

if (p1.magic > p2.magic)

return -1;

else

return 0;

}

}

private class SortForceAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.force > p2.force)

return 1;

if (p1.force < p2.force)

return -1;

else

return 0;

}

}

private class SortForceDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.force < p2.force)

return 1;

if (p1.force > p2.force)

return -1;

else

return 0;

}

}

private class SortLevelAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.level > p2.level)

return 1;

if (p1.level < p2.level)

return -1;

else

return 0;

}

}

private class SortLevelDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.level < p2.level)

return 1;

if (p1.level > p2.level

)

return -1;

else

return 0;

}

}

private class SortIQAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.iq > p2.iq)

return 1;

if (p1.iq < p2.iq)

return -1;

else

return 0;

}

}

private class SortIQDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

People p1 = (People)a;

People p2 = (People)b;

if (p1.iq < p2.iq)

return 1;

if (p1.iq > p2.iq

)

return -1;

else

return 0;

}

}

public static IComparer sortAgeAscending()

{

return new SortAgeAscendingHelper();

}

public static IComparer sortAgeDescending()

{

return new SortAgeDescendingHelper();

}

public static IComparer sortHPAscending()

{

return new SortHPAscendingHelper();

}

public static IComparer sortHPDescending()

{

return new SortHPDescendingHelper();

}

public static IComparer sortStaminaAscending()

{

return new SortStaminaAscendingHelper();

}

public static IComparer sortStaminaDescending()

{

return new SortStaminaDescendingHelper();

}

public static IComparer sortMagicAscending()

{

return new SortMagicAscendingHelper();

}

public static IComparer sortMagicDescending()

{

return new SortMagicDescendingHelper();

}

public static IComparer sortForceAscending()

{

return new SortForceAscendingHelper();

}

public static IComparer sortForceDescending()

{

return new SortForceDescendingHelper();

}

public static IComparer sortLevelAscending()

{

return new SortLevelAscendingHelper();

}

public static IComparer sortLevelDescending()

{

return new SortLevelDescendingHelper();

}

public static IComparer sortIQAscending()

{

return new SortIQAscendingHelper();

}

public static IComparer sortIQDescending()

{

return new SortIQDescendingHelper();

}

}

class MainClass

{

public static void Main(string[] args)

{

try

{

string name;

string type;

uint age;

string choice = "1";

int quantity = 0;

int characterChoice = 0;

Console.WriteLine("Привет! Сколько персонажей ты хочешь сделать?"); quantity = int.Parse(Console.ReadLine());

People[] people = new People[quantity];

People mainCharacter;

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

{

Console.WriteLine($"Дай {i + 1}-му несколько параметров:");

Console.Write($"{i + 1}-е имя: "); name = Console.ReadLine();

while (name == string.Empty)

{

Console.WriteLine("Введи всё же имя персонажу.");

Console.Write($"{i + 1}-е имя: "); name = Console.ReadLine();

}

Console.Write($"{i + 1}-й возраст: "); age = Convert.ToUInt32(Console.ReadLine());

Console.Write($"{i + 1}-й тип(мечник, маг, вор): "); type = Console.ReadLine();

if ((type != "мечник") && (type != "маг") && (type != "вор"))

{

Console.WriteLine("У нас такого не существует, это какая-то магия!");

type = "маг";

}

people[i] = (new People(name, age, type));

}

Console.Clear();

mainCharacter = people[0];

Console.WriteLine("Пока ты управляешь персонажем с именем " + mainCharacter.Name);

Console.WriteLine("Ну что ж, действуй! Выбери, что ты хочешь сделать.");

while ((choice != "10") && (mainCharacter.IsAlive))

{

Console.WriteLine("\nЧто дальше?\n");

people[0].HandlingInstruction();

choice = Console.ReadLine();

if (choice == "0")

{

Console.Clear();

mainCharacter.HandlingInstruction();

}

else if (choice == "1")

{

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

{

people[i].DisplayCharacters(i);

}

Console.WriteLine("И кого же ты хочешь выбрать?"); characterChoice = int.Parse(Console.ReadLine());

mainCharacter = people[characterChoice - 1];

}

else if (choice == "2")

{

mainCharacter.Waiting();

}

else if (choice == "3")

{

mainCharacter.Battle();

}

else if (choice == "4")

{

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "5")

{

mainCharacter.DisplayParams();

}

else if (choice == "6")

{

mainCharacter.DisplaySortChoice();

switch (int.Parse(Console.ReadLine()))

{

case 1: Array.Sort(people, People.sortAgeAscending()); break;

case 2: Array.Sort(people, People.sortHPAscending()); break;

case 3: Array.Sort(people, People.sortStaminaAscending()); break;

case 4: Array.Sort(people, People.sortMagicAscending()); break;

case 5: Array.Sort(people, People.sortForceAscending()); break;

case 6: Array.Sort(people, People.sortLevelAscending()); break;

case 7: Array.Sort(people, People.sortIQAscending()); break;

default:

Console.WriteLine("Попробуй выбрать предложенное!");

break;

}

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "7")

{

mainCharacter.DisplaySortChoice();

switch (int.Parse(Console.ReadLine()))

{

case 1: Array.Sort(people, People.sortAgeDescending()); break;

case 2: Array.Sort(people, People.sortHPDescending()); break;

case 3: Array.Sort(people, People.sortStaminaDescending()); break;

case 4: Array.Sort(people, People.sortMagicDescending()); break;

case 5: Array.Sort(people, People.sortForceDescending()); break;

case 6: Array.Sort(people, People.sortLevelDescending()); break;

case 7: Array.Sort(people, People.sortIQDescending()); break;

default:

Console.WriteLine("Попробуй выбрать предложенное!");

break;

}

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "10")

{

break;

}

else

{

Console.WriteLine("Попробуй выбрать предложенное!");

}

Console.ReadKey();

Console.Clear();

}

Console.WriteLine("До встречи!");

Console.ReadLine();

}

catch (Exception)

{

Console.WriteLine("Ты что-то ввёл не так, попробуй ещё раз!");

}

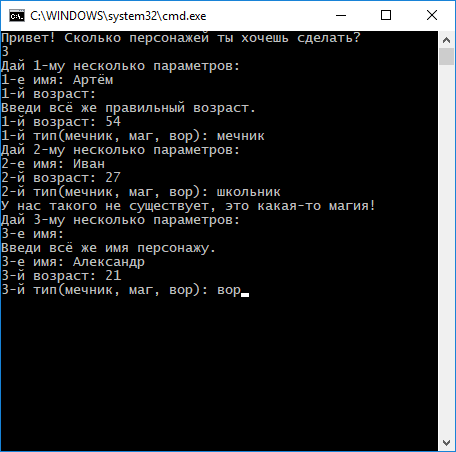
}

}

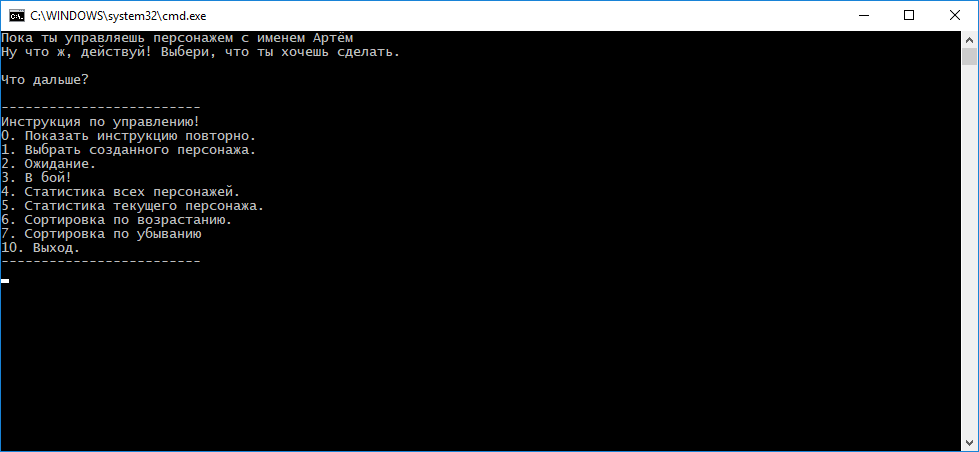
}

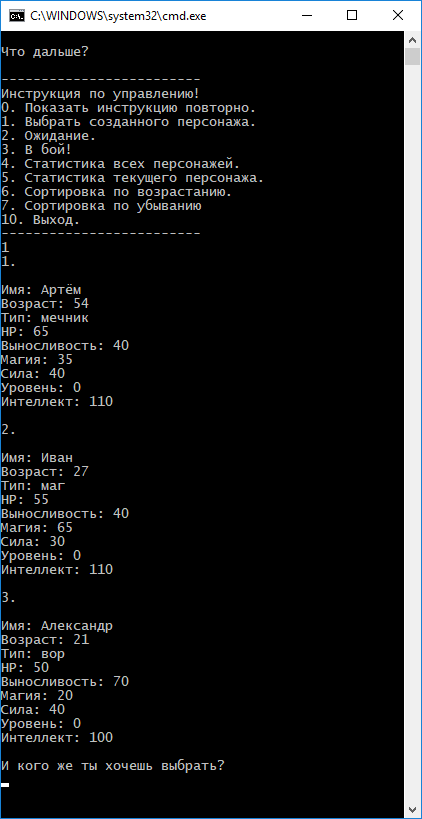
**Скриншоты:**

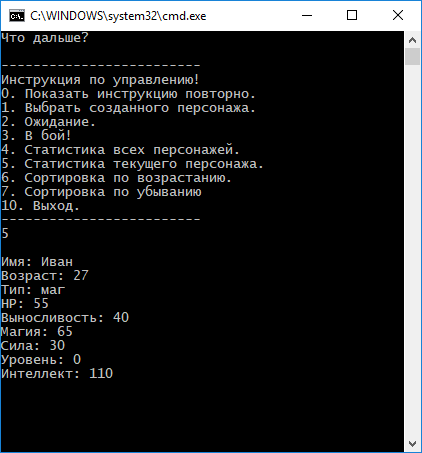
**//**у каждого типа есть свой буст к характеристикам, у мечника буст к хп, у мага к магии и у вора к выносливости соответсвенно, также буст зависит от возраста, чем старше, тем лучше ментальные навыки, чем моложе – физические

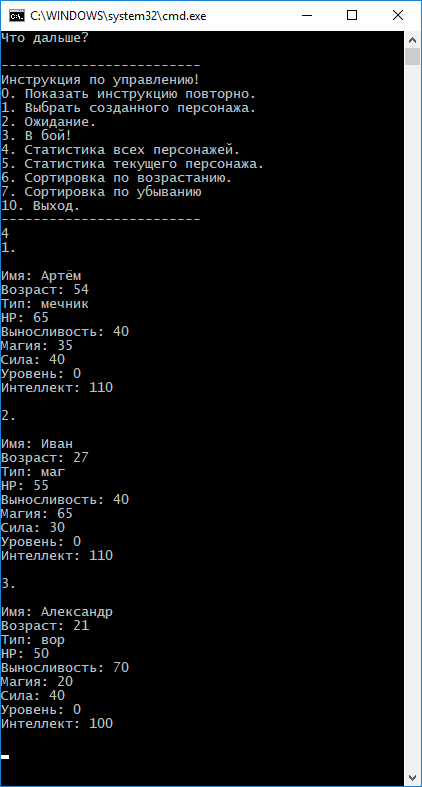


//тыкаем Enter, чтобы продолжить дальше

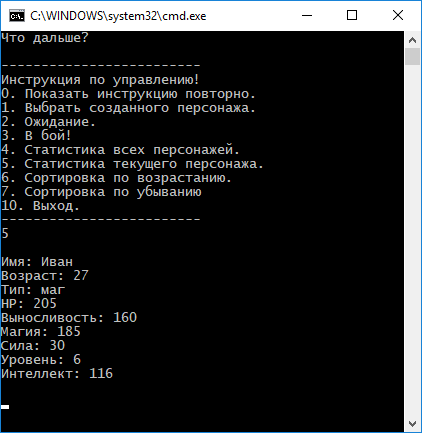




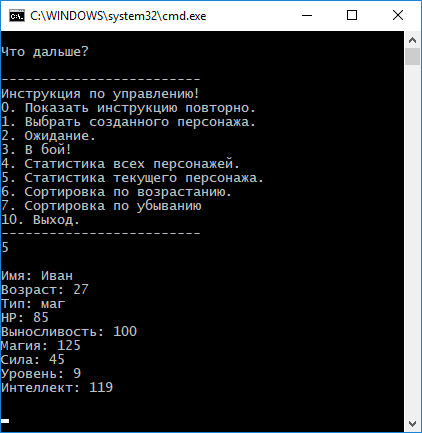




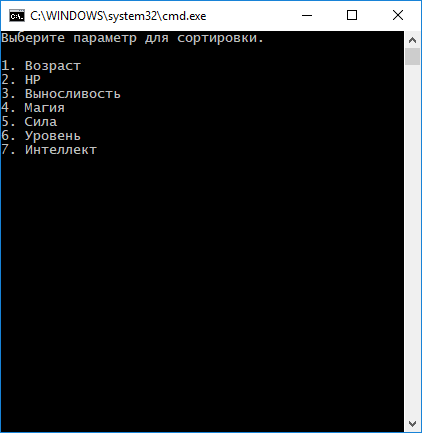
//после Ожидания(2) у персонажа меняются характеристики



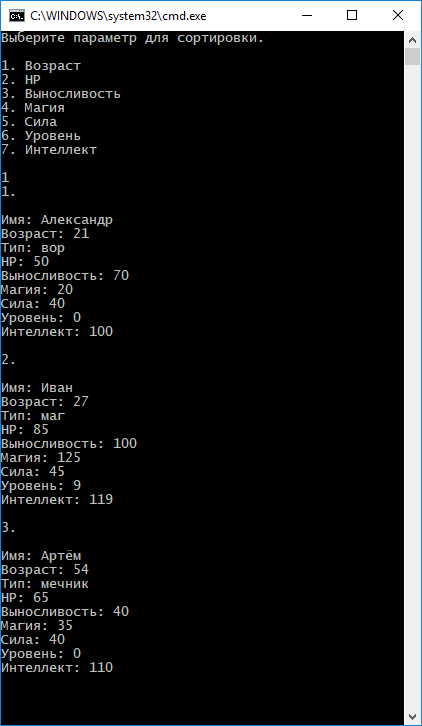
//после В бой!(3) у персонажа отнимается ХП, выносливость и магия, остальные хар-ки повышаются

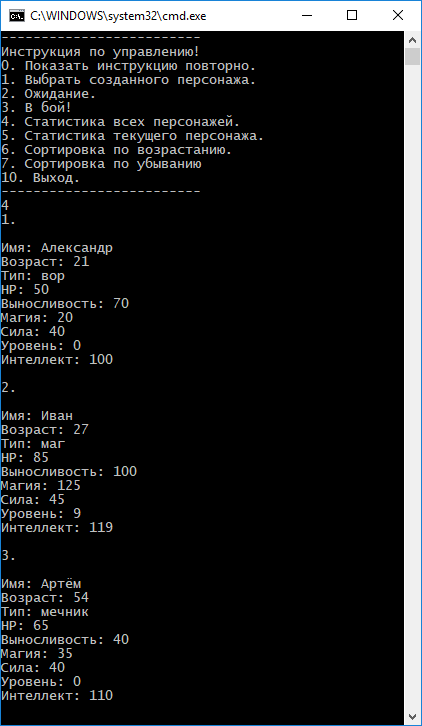


//при выборе сортировки(6 или 7) появляется данное меню с выбором параметра



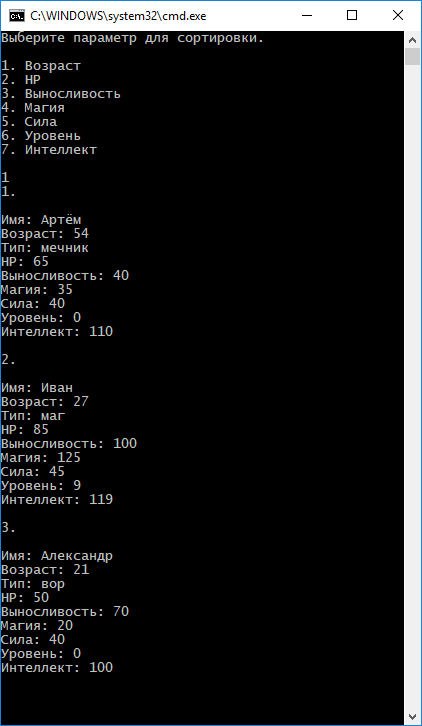
//например, по возрастанию возраста

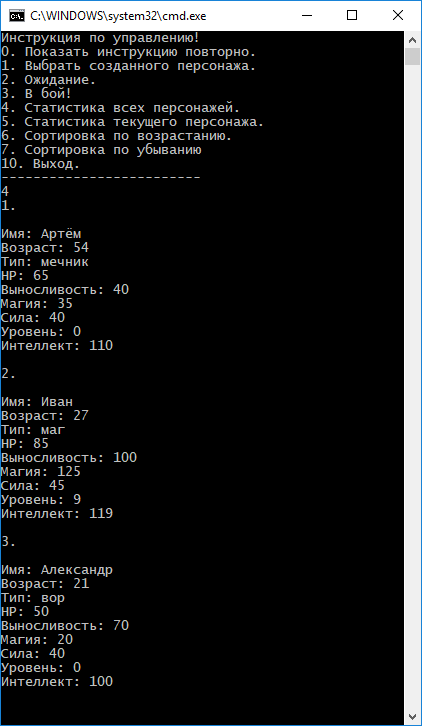


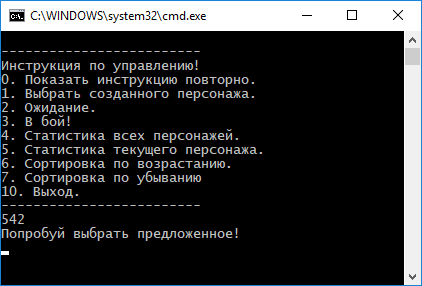


//например,

по убыванию возраста







//условный game over

