Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №1

Тема “ Создание классов.”

Цель работы: Получение навыков в разработке программ с использованием классов.

Постановка задачи: Создать класс <Источник бесперебойного питания>, содержащий поля, которые можно использовать для хранения данных. Предусмотреть инициализацию переменных (полей) класса, помещение данных в переменные и извлечение данных.

Программный код:

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Ёмкость источника бесперебойного питания " + PowerSuply.capcity);

Console.WriteLine("Название фирмы " + PowerSuply.firm);

Console.ReadKey();

}

}

}

public class PowerSuply

{

public static string firm = "Samsung";

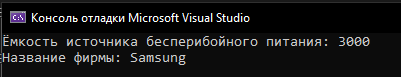
public string mark = "Z310Dual";

public static int capcity = 3000;

private int id;

}

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы я не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №2

Тема “ Создание объектов.”

Цель работы: Получение навыков в разработке программ с использованием объектов.

Постановка задачи: Создать объекты класса<Источник бесперебойного питания>. Объекты должны быть созданы с помощью конструктора по умолчанию и перегруженного конструктора.

Программный код:

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

PowerSuply Ecovolt = new PowerSuply("Ecovolt", "Termo312", 300, 13300);

PowerSuply Myown = new PowerSuply();

Console.WriteLine("Название источника бесперибойного питания из магазина "+Ecovolt.firm + " "+Ecovolt.mark);

Console.WriteLine("Название домашнего источника бесперибойного питания " + Myown.firm + " " + Myown.mark);

Console.ReadKey();

}

}

}

public class PowerSuply

{

public string firm = "Undefind";

public string mark = "Undefind";

public int capcity= 0;

public int cost = 0;

private int id;

public PowerSuply() { }

public PowerSuply(string firm,string mark, int capacity, int cost)

{

this.firm = firm;

this.mark = mark;

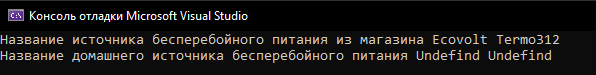
this.capcity = capacity;

this.cost = cost;

}

}

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы я не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №3

Тема “ Наследование, скрытие полей класса и виртуальные методы”

Цель работы: Получение навыков в разработке программ с использованием классов образованных от других классов.

Постановка задачи: Создать класс-наследник от класса <Источник бесперебойного питания>.В классе-наследнике перекрыть некоторые поля класса-родителя и добавить дополнительное поле, связанное со свойствами предмета задания по варианту. При перекрывании полей использовать виртуальные методы. Создать объекты обоих полученных класса и занести в них данные. Организовать обращения родительских методов к замещающим их методам наследника.

Программный код:

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

PowerSuply Ecovolt = new PowerSuply("Ecovolt", "Termo312", 300, 13300);

RezervSuply Luduvi = new RezervSuply("Luduvi", "90000", 90000, 11820, 48);

Console.WriteLine(Ecovolt.GetReport());

Console.WriteLine(Luduvi.GetReport());

Console.ReadKey();

}

}

}

public class PowerSuply

{

public string firm = "Undefind";

public string mark = "Undefind";

public int capcity = 0;

public int cost = 0;

private int id;

public PowerSuply() { }

public PowerSuply(string firm, string mark, int capacity, int cost)

{

this.firm = firm;

this.mark = mark;

this.capcity = capacity;

this.cost = cost;

}

public virtual string GetReport()

{

return $"Источник бесперебойного питания:(Родитель)\nФирма: {firm}\nМарка: {mark}\nЁмкость: {capcity} мАч\nЦена: {cost} руб.\n";

}

}

public class RezervSuply : PowerSuply

{

public int worktime = 0;

public RezervSuply() { }

public RezervSuply(string firm, string mark, int capacity, int cost,int worktime)

{

this.firm = firm;

this.mark = mark;

this.capcity = capacity;

this.cost = cost;

this.worktime = worktime;

}

public override string GetReport()

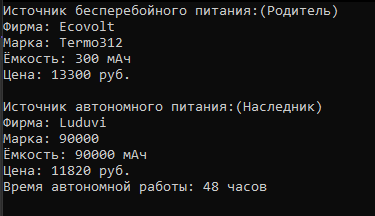
{

return $"Источник автономного питания:(Наследник)\nФирма: {firm}\nМарка: {mark}\nЁмкость: {capcity} мАч \nЦена: {cost} руб.\nВремя автономной работы: {worktime} часов\n";

}

}

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы я не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №4

Тема “ Использование классов для работы с массивами данных”

Цель работы: Получение навыков в разработке программ с использованием классов для создания, хранения и работы с массивами данных.

Постановка задачи: Создать объекты класса <Источник бесперебойного питания>, причем объекты класса должны хранить массивы данных о предметной области, связанные со свойствами предмета задания по варианту. Причем количество элементов массива должно задаваться программно пользователем в интерактивном режиме.

Программный код:

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите кол-во элементов");

int n = Convert.ToInt32(Console.ReadLine());

queues a = new queues();

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

{

a[i] = new PowerSuply();

Console.WriteLine($"Источник автономного питания:\nФирма: {a[i].firm}\nМарка: {a[i].mark}\nИдентифиционный номер: {a[i].id}\n");

}

Console.ReadKey();

}

}

}

class queues

{

PowerSuply[] Queues = new PowerSuply[30];

public PowerSuply this[int pos]

{

get

{

if (pos >= 0 && pos < 30)

return Queues[pos];

else

{

throw new IndexOutOfRangeException("Вне диапазона");

}

}

set

{

Queues[pos] = value;

}

}

}

public class PowerSuply

{

public string firm;

public string mark;

public string id;

private string[] firmArr = new string[6] { "CyberPower", "IPPON", "El-Power", "Philips","Panasonic","samsung" };

private string[] markArr = new string[4] { "Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)", "с двойным преобразованием напряжения(On-Line)","Cломанный" };

public PowerSuply()

{

Random random = new Random();

int t = random.Next(0, firmArr.Length);

int t2 = random.Next(0, markArr.Length);

firm = firmArr[t];

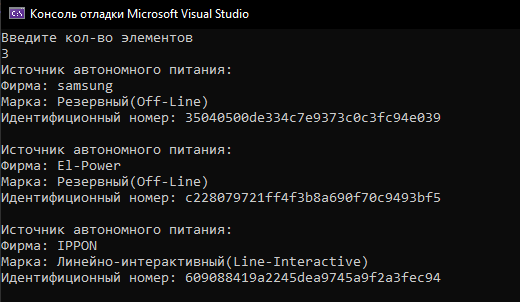
mark = markArr[t2];

id = Guid.NewGuid().ToString("N");

}

}

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы я не столкнулся трудностями и без особого труда написал код ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №5

Тема “ Сохранение текущего состояния объектов в файла”

Цель работы Получение навыков в разработке программ с использованием файлов, сохранения информации в них и работы с ними

Постановка задачи: Создать объекты класса <Источник бесперебойного питания>причем объекты класса должны хранить данных о предметной области, связанные со свойствами предмета задания по варианту. Сохранить данные объекты в файле. Причем сохранить необходимо как отдельные объекты, так объект контейнерного типа, полученный в лабораторной работе 4. Считать сохраненные данные из файла и вывести их на экран.

Программный код:

namespace Lab5.\_3

{

internal class Program

{

[STAThread]

static void Main(string[] args)

{

Console.WriteLine("Введите кол-во элементов");

int n = Convert.ToInt32(Console.ReadLine());

queues q2 = new queues(n);

q2.GetRes();

BinaryFormatter sw = new BinaryFormatter();

using (FileStream fs = new FileStream("File\_queue.txt", FileMode.Create, FileAccess.Write))//Серелизация

{

sw.Serialize(fs, q2);

Console.WriteLine($"Object has been serilized\n");

}

Console.ReadKey();

queues q1 = new();

using (FileStream fs = new FileStream("File\_queue.txt", FileMode.Open, FileAccess.Read))//Десерилизация

{

while (fs.Position < fs.Length)

{

q1 = (queues)sw.Deserialize(fs);

Console.WriteLine($"Object has been deserialized\n");

}

}

q1.GetRes();

Console.ReadLine();

}

}

[Serializable]

public class PowerSuply

{

public string firm;

public string mark;

public string id;

private string[] firmArr = new string[6] { "CyberPower", "IPPON", "El-Power", "Philips", "Panasonic", "samsung" };

private string[] markArr = new string[4] { "Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)", "с двойным преобразованием напряжения(On-Line)", "Cломанный" };

public PowerSuply()

{

Random random = new Random();

int t = random.Next(0, firmArr.Length);

int t2 = random.Next(0, markArr.Length);

firm = firmArr[t];

mark = markArr[t2];

id = Guid.NewGuid().ToString("n");

//only "D", "d", "N", "n", "P", "p", "B", "b", "X" or "x"."

}

}

[Serializable]

public class queues

{

private readonly int n = 1;

PowerSuply[] Queues = new PowerSuply[30];

public queues()

{

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

{

Queues[i] = new PowerSuply();

}

}

public queues(int n)

{

this.n = n;

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

{

Queues[i] = new PowerSuply();

}

}

public PowerSuply this[int pos]

{

get

{

if (pos >= 0 && pos < 30)

return Queues[pos];

else

{

throw new IndexOutOfRangeException("Вне диапазона");

}

}

set

{

Queues[pos] = value;

}

}

public void GetRes()

{

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

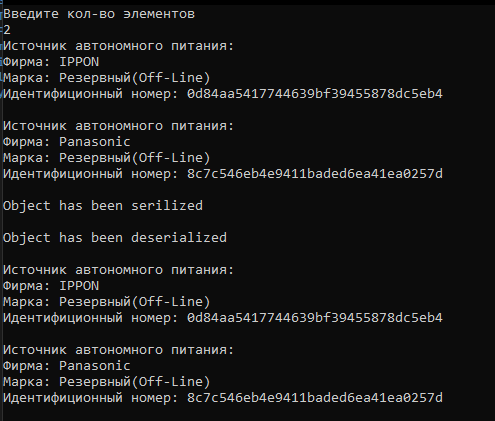
Console.WriteLine($"Источник автономного питания:\nФирма: {Queues[i].firm}\nМарка: {Queues[i].mark}\nИдентифиционный номер: {Queues[i].id}\n");

}

}

}

Результат работы программы:



Вывод: При выполнении работы я не столкнулся трудностями и без особого труда написал код ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №6

Тема “ Сортировка элементов массива.”

Цель работы: Ознакомиться с различными методами сортировки массивов

Постановка задачи: Создать объекты класса «Источники бесперебойного питания», причем объекты класса должны хранить данных о предметной области, связанные со свойствами предмета задания по варианту. В класс контейнерного типа, полученный в лабораторной работе 4, добавить все рассмотренные сортировки в качестве методов. Осуществите сортировку своих данных по разным полям сравните эффективность методов сортировкиПрограммный код:

namespace Lab6

{

[Serializable]

public class PowerSuply

{

public string firm;

public string mark;

public int id;

private string[] firmArr = new string[13] { "CyberPower", "IPPON", "El-Power", "Philips", "Panasonic", "samsung",

"ЗИТ","ИМПУЛЬС","Парус электро", "РОТЕК", "ТК Профэнерджи", "Электромаш", "QTECH" };

private string[] markArr = new string[4] { "Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)",

"с двойным преобразованием напряжения(On-Line)", "Cломанный" };

public PowerSuply()

{

Random random = new Random();

int t = random.Next(0, firmArr.Length);

int t2 = random.Next(0, markArr.Length);

firm = firmArr[t];

mark = markArr[t2];

Random idRan = new Random();

id = idRan.Next(100);

}

}

[Serializable]

public class queues

{

public readonly int n = 1;

public PowerSuply[] Queues;

public queues()

{

Queues = new PowerSuply[n];

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

{

Queues[i] = new PowerSuply();

}

}

public queues(int n)

{

this.n = n;

Queues = new PowerSuply[n];

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

{

Queues[i] = new PowerSuply();

}

}

public PowerSuply this[int pos]

{

get

{

if (pos >= 0 && pos < 30)

return Queues[pos];

else

{

throw new IndexOutOfRangeException("Вне диапазона");

}

}

set

{

Queues[pos] = value;

}

}

public void GetRes()//выводит результат(все поля объекта в строчку

{

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

{

Console.WriteLine($"\nФирма: {Queues[i].firm}; Марка: {Queues[i].mark}; Идентифиционный номер: {Queues[i].id}\n");

}

}

public void GetIds() //выводит поля id

{

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

{

Console.Write($"{Queues[i].id} ");

}

Console.WriteLine();

}

public void GetIds(string str) //выводит поля id

{

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

{

Console.Write($"{Queues[i].id} ");

}

Console.WriteLine(str);

}

public void GetFirm()//выводит поля firm

{

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

{

Console.Write($"{Queues[i].firm} ");

}

Console.WriteLine();

}

public void GetFirm(string str)//выводит поля firm

{

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

{

Console.Write($"{Queues[i].firm} ");

}

Console.WriteLine(str);

}

public void CopyTo(queues obj1)

{

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

{

obj1.Queues[i].id = this.Queues[i].id;

obj1.Queues[i].firm = this.Queues[i].firm;

obj1.Queues[i].mark = this.Queues[i].mark;

}

}

}

public class Sorts

{

public static void DirectChooseId(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

for (int i = 0; i < arr.Queues.Length - 1; i++)

{

int minIndex = i;

for (int j = i; j < arr.Queues.Length; j++)

{

if (arr[j].id < arr[minIndex].id)

{

minIndex = j;

}

}

if (minIndex != i)

{

ghost = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = ghost;

}

}

stopwatch.Stop();

Console.WriteLine($" Метод DirectChoose {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void DirectChooseFirm(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

for (int i = 0; i < arr.Queues.Length - 1; i++)

{

int minIndex = i;

for (int j = i + 1; j < arr.Queues.Length; j++)

{

if (arr[j].firm.CompareTo(arr[minIndex].firm)<0)

{

minIndex = j;

}

}

if (minIndex != i)

{

ghost = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = ghost;

}

}

stopwatch.Stop();

Console.WriteLine($" Метод DirectChoose {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void BubblyId(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

bool swapped;

for (int j = 0; j < arr.Queues.Length-1; j++)

{

swapped = false;

for (int i = 0; i < arr.Queues.Length - 1; i++)

{

if (arr[i + 1].id < arr[i].id)

{

ghost = arr[i + 1];

arr[i + 1] = arr[i];

arr[i] = ghost;

swapped = true;

}

}

if(!swapped) { break; }

}

stopwatch.Stop();

Console.WriteLine($" Метод Bubbly {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void BubblyFirm(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

bool swapped;

for (int j = 0; j < arr.Queues.Length - 1; j++)

{

swapped = false;

for (int i = 0; i < arr.Queues.Length - 1; i++)

{

if (arr[i + 1].firm.CompareTo(arr[i].firm)<0)

{

ghost = arr[i + 1];

arr[i + 1] = arr[i];

arr[i] = ghost;

swapped = true;

}

}

if (!swapped) { break; }

}

stopwatch.Stop();

Console.WriteLine($" Метод Bubbly {stopwatch.Elapsed.TotalMilliseconds} ");

}

public static void DirectConnectionId(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

for (int i = 0; i<arr.Queues.Length; i++)

{

int value = arr[i].id;//// Значение id текущего элемента

int index = i;// Его индекс в массиве

while ((index > 0) && (arr[index - 1].id > value)) // Цикл, выполняющий сдвиг элементов влево, пока предшествующий элемент больше текущего

{

ghost = arr[index];

arr[index] = arr[index - 1];

arr[index - 1] = ghost;

index--;

}

}

stopwatch.Stop();

Console.WriteLine($" Метод DirectConnection {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void DirectConnectionFirm(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

PowerSuply ghost = new PowerSuply();

for (int i = 0; i < arr.Queues.Length; i++)

{

string value = arr[i].firm;//// Значение firm текущего элемента

int index = i;// Его индекс в массиве

while ((index > 0) && (arr[index - 1].firm.CompareTo(value)>0)) // Цикл, выполняющий сдвиг элементов влево, пока предшествующий элемент больше текущего

{

ghost = arr[index];

arr[index] = arr[index - 1];

arr[index - 1] = ghost;

index--;

}

}

stopwatch.Stop();

Console.WriteLine($" Метод DirectConnection {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void ShakerSortId(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

bool swapped = true;// Флаг, означающий, что произошла перестановка элементов

int start = 0;

int end = arr.Queues.Length - 1;

PowerSuply ghost = new PowerSuply();

while (swapped)

{

swapped = false;

// Проход слева направо

for (int i = start; i < end; i++)

{

if (arr[i].id > arr[i + 1].id)// Если предыдущий элемент больше следующего, меняем их местами

{

ghost = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = ghost;

swapped = true;

}

}

if (!swapped)// Если на проходе не было перестановок, массив уже отсортирован

{

break;

}

swapped = false;

end--;

// Проход справа налево

for (int i = end - 1; i >= start; i--)

{

if (arr[i].id > arr[i + 1].id)// Если предыдущий элемент больше следующего, меняем их местами

{

ghost = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = ghost;

swapped = true;

}

}

start++;

}

stopwatch.Stop();

Console.WriteLine($" Метод ShakerSort {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void ShakerSortFirm(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

bool swapped = true;// Флаг, означающий, что произошла перестановка элементов

int start = 0;

int end = arr.Queues.Length - 1;

PowerSuply ghost = new PowerSuply();

while (swapped)

{

swapped = false;

// Проход слева направо

for (int i = start; i < end; i++)

{

if (arr[i].firm.CompareTo(arr[i+1].firm)>0)// Если предыдущий элемент больше следующего, меняем их местами

{

ghost = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = ghost;

swapped = true;

}

}

if (!swapped)// Если на проходе не было перестановок, массив уже отсортирован

{

break;

}

swapped = false;

end--;

// Проход справа налево

for (int i = end - 1; i >= start; i--)

{

if (arr[i].firm.CompareTo(arr[i + 1].firm) > 0)// Если предыдущий элемент больше следующего, меняем их местами

{

ghost = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = ghost;

swapped = true;

}

}

start++;

}

stopwatch.Stop();

Console.WriteLine($" Метод ShakerSort {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void ShallSortId(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

int n = arr.Queues.Length;

PowerSuply ghost = new PowerSuply();

// Выбор шага

int gap = n / 2;

// Пока шаг больше 0

while (gap > 0)

{

// Выполняем сортировку вставками с заданным шагом

for (int i = gap; i < n; i++)

{

ghost = arr[i];

int j = i;

// Сдвигаем элементы на шаг назад, пока не найдем место для вставки

while (j >= gap && arr[j - gap].id > ghost.id)

{

arr[j] = arr[j - gap];

j -= gap;

}

arr[j] = ghost;

}

// Уменьшаем шаг

gap /= 2;

}

stopwatch.Stop();

Console.WriteLine($" Метод ShallSort {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

public static void ShallSortFirm(queues arr)

{

Stopwatch stopwatch = Stopwatch.StartNew();

int n = arr.Queues.Length;

PowerSuply ghost = new PowerSuply();

// Выбор шага

int gap = n / 2;

// Пока шаг больше 0

while (gap > 0)

{

// Выполняем сортировку вставками с заданным шагом

for (int i = gap; i < n; i++)

{

ghost = arr[i];

int j = i;

// Сдвигаем элементы на шаг назад, пока не найдем место для вставки

while (j >= gap && (arr[j-gap].firm.CompareTo(ghost.firm)>0))

{

arr[j] = arr[j - gap];

j -= gap;

}

arr[j] = ghost;

}

// Уменьшаем шаг

gap /= 2;

}

stopwatch.Stop();

Console.WriteLine($" Метод ShallSort {stopwatch.Elapsed.TotalMilliseconds} миллисекунд ");

}

}

}

namespace Lab6

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите кол-во элементов");

int n = Convert.ToInt32(Console.ReadLine());

queues test0 = new(n);

queues test1 = new(n); test0.CopyTo(test1);

queues test2 = new(n); test0.CopyTo(test2);

queues test3 = new(n); test0.CopyTo(test3);

queues test4 = new(n); test0.CopyTo(test4);

queues test5 = new(n);test0.CopyTo(test5);

test1.GetIds();

Sorts.DirectChooseId(test1);

test1.GetIds("\n");

test2.GetIds();

Sorts.BubblyId(test2);

test2.GetIds("\n");

test3.GetIds();

Sorts.DirectConnectionId(test3);

test3.GetIds("\n");

test4.GetIds();

Sorts.ShakerSortId(test4);

test4.GetIds("\n");

test5.GetIds();

Sorts.ShallSortId(test5);

test5.GetIds("\n");

Console.WriteLine("\n");

test1.GetFirm();

Sorts.DirectChooseFirm(test1);

test1.GetFirm("\n");

test2.GetFirm();

Sorts.BubblyFirm(test2);

test2.GetFirm("\n");

test3.GetFirm();

Sorts.DirectConnectionFirm(test3);

test3.GetFirm("\n");

test4.GetFirm();

Sorts.ShakerSortFirm(test4);

test4.GetFirm("\n");

test5.GetFirm();

Sorts.ShallSortFirm(test5);

test5.GetFirm("\n");

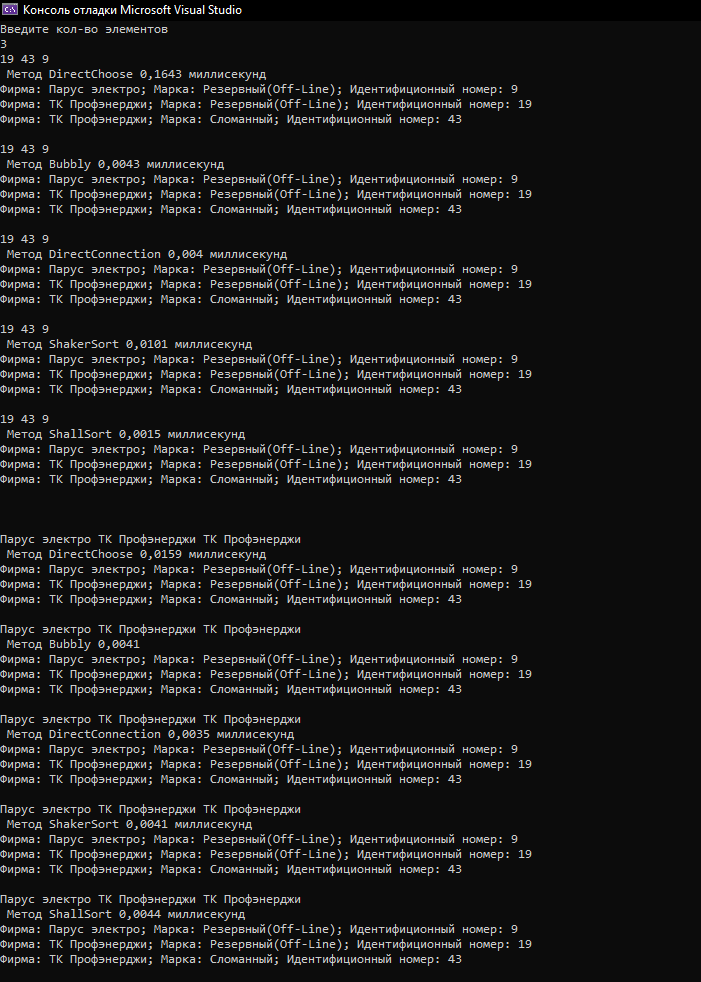
Console.ReadKey();

}

}

}

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу. Исходя из результата работы был сделан вывод, что для сортировки большого объема (30 элементов) типа данных наиболее эффективна сортировка методом Шелла как для строкового типа, так и для числового типа данных. Для сортировки данных меньшего объема (5 элементов) наибольшую эффективность показали метод прямого включения для числового типа данных и метод Шелла для строкового.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

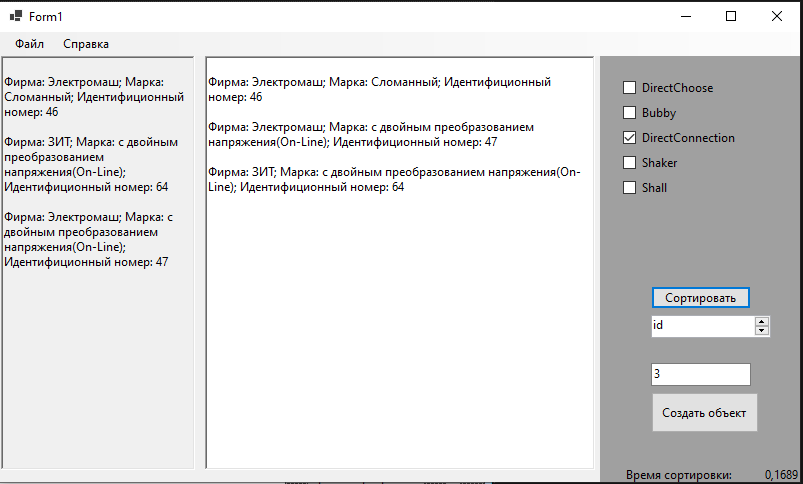
Лабораторная работа №7

Тема “ Создание приложения на Windows Forms”

Цель работы: Получение навыков в разработке программ с с использованием технологии Windows Forms

Постановка задачи: Создать приложение, способное выводить информацию об объектах класса «Источник бесперебойного питания», хранящих массивы данных о предметной области, связанные со свойствами предмета задания по варианту. А также сортировать их используя различные методы сортировки из лабораторной работы №6 и выводить результат.

Результат работы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: Вывод: При выполнении работы не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу. Исходя из результата работы был сделан вывод, что для сортировки большого объема (30 элементов) типа данных наиболее эффективна сортировка методом Шелла как для строкового типа, так и для числового типа данных. Для сортировки данных меньшего объема (5 элементов) наибольшую эффективность показали метод прямого включения для числового типа данных и метод Шелла для строкового.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №1

Тема “ Создание классов.”

Цель работы: Получение навыков в разработке программ с использованием классов.

Постановка задачи: Создать класс <Источник бесперебойного питания>, содержащий поля, которые можно использовать для хранения данных. Предусмотреть инициализацию переменных (полей) класса, помещение данных в переменные и извлечение данных.

Программный код:

class PowerSupply:

firm = "Samsung"

mark = "Z310Dual"

capcity = 3000

def \_\_init\_\_(self):

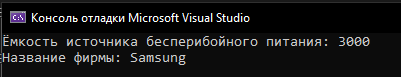
self.id = 0

if \_\_name\_\_ == "\_\_main\_\_":

print("Ёмкость источника бесперибойного питания: ", PowerSupply.capcity)

print("Название фирмы: ", PowerSupply.firm)

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: В ходе выполнения работы с трудностями не столкнулся, кроме того, что пришлось привыкать к особенностям Python.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №2

Тема “ Создание объектов.”

Цель работы: Получение навыков в разработке программ с использованием объектов.

Постановка задачи: Создать объекты класса<Источник бесперебойного питания>. Объекты должны быть созданы с помощью конструктора по умолчанию и перегруженного конструктора.

Программный код:

class PowerSuply:

def \_\_init\_\_(self, firm="Undefind", mark="Undefind", capacity=0, cost=0):

self.firm = firm

self.mark = mark

self.capacity = capacity

self.cost = cost

Myown = PowerSuply()

print("Введите название вашего источника бесперибойного питания")

Myown.firm = input()

print("Введите марку вашего источника бесперибойного питания")

Myown.mark = input()

print("введите ёмкость вашего источника бесперибойного питания")

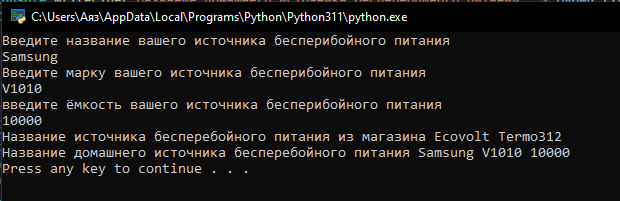
Myown.capacity = int(input())

Ecovolt = PowerSuply("Ecovolt", "Termo312", 300, 13300)

print("Название источника бесперебойного питания из магазина", Ecovolt.firm, Ecovolt.mark)

print("Название домашнего источника бесперебойного питания",Myown.firm, Myown.mark, Myown.capacity)

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы не столкнулся с особыми трудностями и без особого труда написал код на Python ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №3

Тема “ Наследование, скрытие полей класса и виртуальные методы”

Цель работы: Получение навыков в разработке программ с использованием классов образованных от других классов.

Постановка задачи: Создать класс-наследник от класса <Источник бесперебойного питания>.В классе-наследнике перекрыть некоторые поля класса-родителя и добавить дополнительное поле, связанное со свойствами предмета задания по варианту. При перекрывании полей использовать виртуальные методы. Создать объекты обоих полученных класса и занести в них данные. Организовать обращения родительских методов к замещающим их методам наследника.

Программный код:

class PowerSuply:

# Определяем конструктор класса PowerSuply с параметрами firm, mark, capacity и cost

def \_\_init\_\_(self, firm="Undefind", mark="Undefind", capacity=0, cost=0):

# Присваиваем значения переданным параметрам объекта PowerSuply

self.firm = firm

self.mark = mark

self.capacity = capacity

self.cost = cost

def GetReport(self): # Определяем метод GetReport для класса PowerSuply, возвращающий отчет

# Возвращаем отчет о источнике бесперебойного питания

return f"Источник бесперебойного питания:(Родитель)\nФирма: {self.firm}\nМарка: {self.mark}\nЁмкость: {self.capacity} мАч\nЦена: {self.cost} руб.\n"

class RezervSuply(PowerSuply):

# Определяем конструктор класса RezervSuply с параметрами firm, mark, capacity, cost и worktime

def \_\_init\_\_(self, firm="Undefind", mark="Undefind", capacity=0, cost=0, worktime=0):

# Вызываем конструктор родительского класса и передаем ему параметры firm, mark, capacity и cost

super().\_\_init\_\_(firm, mark, capacity, cost)

# Присваиваем значение параметру worktime объекта RezervSuply

self.worktime = worktime

def GetReport(self): # Определяем метод GetReport для класса RezervSuply, возвращающий отчет

# Возвращаем отчет об источнике автономного питания

return f"Источник автономного питания:(Наследник)\nФирма: {self.firm}\nМарка: {self.mark}\nЁмкость: {self.capacity} мАч \nЦена: {self.cost} руб.\nВремя автономной работы: {self.worktime} часов\n"

# Создаем объекты Ecovolt и Luduvi классов PowerSuply и RezervSuply соответственно

Ecovolt = PowerSuply("Ecovolt", "Termo312", 300, 13300)

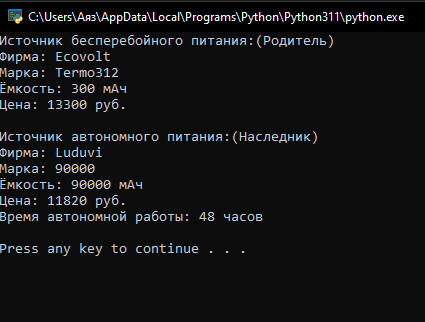
Luduvi = RezervSuply("Luduvi", "90000", 90000, 11820, 48)

# Выводим отчеты о созданных объектах

print(Ecovolt.GetReport())

print(Luduvi.GetReport())

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы я не столкнулся с особыми трудностями и без особого труда написал код на Python ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №4

Тема “ Использование классов для работы с массивами данных”

Цель работы: Получение навыков в разработке программ с использованием классов для создания, хранения и работы с массивами данных.

Постановка задачи: Создать объекты класса <Источник бесперебойного питания>, причем объекты класса должны хранить массивы данных о предметной области, связанные со свойствами предмета задания по варианту. Причем количество элементов массива должно задаваться программно пользователем в интерактивном режиме.

Программный код:

import random # Импортируем модуль random

class PowerSuply:

def \_\_init\_\_(self): # Конструктор класса PowerSuply

self.firmArr = ["CyberPower", "IPPON", "El-Power", "Philips", "Panasonic", "samsung"] # Массив фирм

self.markArr = ["Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)", "с двойным преобразованием напряжения(On-Line)", "Cломанный"] # Массив марок

self.firm = random.choice(self.firmArr) # Выбираем случайную фирму из firmArr

self.mark = random.choice(self.markArr) # Выбираем случайную марку из markArr

self.id = random.randint(0, 29) # Генерируем случайный идентификационный номер

class queues:

def \_\_init\_\_(self, n): # Конструктор класса queues

self.Queues = [PowerSuply() for \_ in range(n)] # Создаем список объектов PowerSuply в количестве n

def \_\_getitem\_\_(self, pos): # Метод для получения элемента по индексу

if pos >= 0 and pos < 30: # Проверяем, находится ли индекс в диапазоне

return self.Queues[pos] # Возвращаем объект из списка по указанному индексу

else:

raise IndexError("Вне диапазона") # Генерируем исключение, если индекс находится вне диапазона

def \_\_setitem\_\_(self, pos, value): # Метод для установки элемента по индексу

self.Queues[pos] = value # Устанавливаем объект в список по указанному индексу

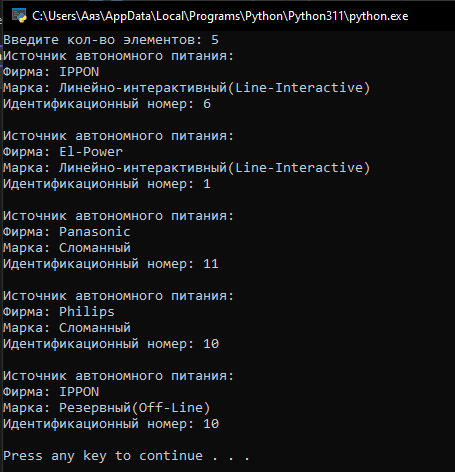
n = int(input("Введите кол-во элементов: ")) # Вводим количество элементов

a = queues(n) # Создаем объект очереди с количеством элементов, введенных пользователем

for i in range(n): # Перебираем каждый объект

print(f"Источник автономного питания:\nФирма: {a[i].firm}\nМарка: {a[i].mark}\nИдентификационный номер: {a[i].id}\n")

Результат работы программы:



|  |  |
| --- | --- |
|  |  |
|  |  |

Вывод: При выполнении работы не столкнулся трудностями и без особого труда написал код на Python ,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №5

Тема “ Сохранение текущего состояния объектов в файла”

Цель работы Получение навыков в разработке программ с использованием файлов, сохранения информации в них и работы с ними

Постановка задачи: Создать объекты класса <Источник бесперебойного питания>причем объекты класса должны хранить данных о предметной области, связанные со свойствами предмета задания по варианту. Сохранить данные объекты в файле. Причем сохранить необходимо как отдельные объекты, так объект контейнерного типа, полученный в лабораторной работе 4. Считать сохраненные данные из файла и вывести их на экран.

Программный код:

import random # Импортируем модуль random для генерации случайных элементов

import uuid # Импортируем модуль uuid для генерации уникальных идентификаторов

import pickle # Импортируем модуль pickle для сериализации и десериализации объектов

class PowerSuply:

def \_\_init\_\_(self): # Конструктор класса PowerSuply

firmArr = ["CyberPower", "IPPON", "El-Power", "Philips", "Panasonic", "samsung"] # Массив фирм

markArr = ["Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)", "с двойным преобразованием напряжения(On-Line)", "Cломанный"] # Массив марок

self.firm = random.choice(firmArr) # Выбираем случайную фирму из firmArr

self.mark = random.choice(markArr) # Выбираем случайную марку из markArr

self.id = str(uuid.uuid4().hex) # Генерируем уникальный идентификатор

class queues:

def \_\_init\_\_(self, n=1): # Конструктор класса queues

self.n = n # Задаем количество элементов

self.Queues = [PowerSuply() for \_ in range(n)] # Создаем список объектов PowerSuply в количестве n

def \_\_getitem\_\_(self, pos): # Метод для получения элемента по индексу

if pos >= 0 and pos < 30: # Проверяем, находится ли индекс в диапазоне

return self.Queues[pos] # Возвращаем объект из списка по указанному индексу

else:

raise IndexError("Вне диапазона") # Генерируем исключение, если индекс находится вне диапазона

def \_\_setitem\_\_(self, pos, value): # Метод для установки элемента по индексу

self.Queues[pos] = value # Устанавливаем объект в список по указанному индексу

def GetRes(self):

for i in range(self.n):

print(f"Источник автономного питания:\nФирма: {self.Queues[i].firm}\nМарка: {self.Queues[i].mark}\nИдентифиционный номер: {self.Queues[i].id}\n")

if \_\_name\_\_ == "\_\_main\_\_":

print("Введите кол-во элементов")

n = int(input())

q2 = queues(n) # Создаем объект queues с количеством элементов, введенных пользователем

q2.GetRes() # Выводим информацию о каждом объекте из Queues

# Сериализация

with open("File\_queue.txt", "wb") as fs:

pickle.dump(q2, fs) # Сериализуем объект q2 и сохраняем его в файл

print("Объект был сериализован\n")

q1 = queues() # Создаем новый объект queues для десериализации

# Десериализация

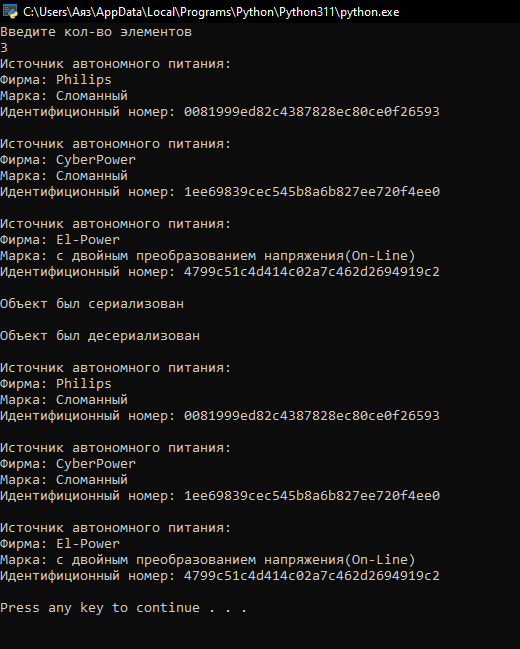
with open("File\_queue.txt", "rb") as fs:

q1 = pickle.load(fs) # Десериализуем объект из файла и сохраняем его в q1

print("Объект был десериализован\n")

q1.GetRes() # Выводим информацию о каждом объекте из q1

Результат работы программы:



Вывод: При выполнении работы не столкнулся трудностями и без особого труда написал код на Python,решающий поставленную задачу.

Выполнил: студ. гр. №2221121

Хайрутдинов Аяз Алмазович

Проверил: ассистент Булатова С.В.

Лабораторная работа №6

Тема “ Сортировка элементов массива.”

Цель работы: Ознакомиться с различными методами сортировки массивов

Постановка задачи: Создать объекты класса «Источники бесперебойного питания», причем объекты класса должны хранить данных о предметной области, связанные со свойствами предмета задания по варианту. В класс контейнерного типа, полученный в лабораторной работе 4, добавить все рассмотренные сортировки в качестве методов. Осуществите сортировку своих данных по разным полям сравните эффективность методов сортировкиПрограммный код:

Программный код:

﻿import random

import time

class PowerSuply:

def \_\_init\_\_(self):

self.firm = ""

self.mark = ""

self.id = 0

self.firmArr = ["CyberPower", "IPPON", "El-Power", "Philips", "Panasonic", "samsung",

"ЗИТ", "ИМПУЛЬС", "Парус электро", "РОТЕК", "ТК Профэнерджи", "Эл��ктромаш", "QTECH"]

self.markArr = ["Резервный(Off-Line)", "Линейно-интерактивный(Line-Interactive)",

"с двойным преобразованием напряжения(On-Line)", "Cломанный"]

self.firm = random.choice(self.firmArr)

self.mark = random.choice(self.markArr)

self.id = random.randint(0, 99)

class queues:

def \_\_init\_\_(self, n=1):

self.n = n

self.Queues = [PowerSuply() for \_ in range(n)]

def \_\_getitem\_\_(self, pos):

if pos >= 0 and pos < 30:

return self.Queues[pos]

else:

raise IndexError("Вне диапазона")

def \_\_setitem\_\_(self, pos, value):

self.Queues[pos] = value

def GetRes(self):

for i in range(self.n):

print(f"Фирма: {self.Queues[i].firm}; Марка: {self.Queues[i].mark}; Идентифиционный номер: {self.Queues[i].id}")

def GetRes(self, str=""):

for i in range(self.n):

print(f"Фирма: {self.Queues[i].firm}; Марка: {self.Queues[i].mark}; Идентифиционный номер: {self.Queues[i].id}", end ="")

print(str)

def GetIds(self, str=""):

for i in range(self.n):

print(f"{self.Queues[i].id} ", end="")

print(str)

def GetFirm(self, str=""):

for i in range(self.n):

print(f"{self.Queues[i].firm} ", end="")

print(str)

def CopyTo(self, obj1):

for i in range(self.n):

obj1.Queues[i].id = self.Queues[i].id

obj1.Queues[i].firm = self.Queues[i].firm

obj1.Queues[i].mark = self.Queues[i].mark

class Sorts:

@staticmethod

def DirectChooseId(arr):

start\_time = time.time()

ghost = PowerSuply()

for i in range(len(arr.Queues) - 1):

minIndex = i

for j in range(i, len(arr.Queues)):

if arr[j].id < arr[minIndex].id:

minIndex = j

if minIndex != i:

ghost = arr[minIndex]

arr[minIndex] = arr[i]

arr[i] = ghost

elapsed\_time = time.time() - start\_time

print(f" Метод DirectChoose {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def DirectChooseFirm(arr):

start\_time = time.time()

ghost = PowerSuply()

for i in range(len(arr.Queues) - 1):

minIndex = i

for j in range(i + 1, len(arr.Queues)):

if arr[j].firm < arr[minIndex].firm:

minIndex = j

if minIndex != i:

ghost = arr[minIndex]

arr[minIndex] = arr[i]

arr[i] = ghost

elapsed\_time = time.time() - start\_time

print(f" Метод DirectChoose {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def BubblyId(arr):

start\_time = time.time()

ghost = PowerSuply()

swapped = True

for j in range(len(arr.Queues) - 1):

swapped = False

for i in range(len(arr.Queues) - 1):

if arr[i + 1].id < arr[i].id:

ghost = arr[i + 1]

arr[i + 1] = arr[i]

arr[i] = ghost

swapped = True

if not swapped:

break

elapsed\_time = time.time() - start\_time

print(f" Метод Bubbly {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def BubblyFirm(arr):

start\_time = time.time()

ghost = PowerSuply()

swapped = True

for j in range(len(arr.Queues) - 1):

swapped = False

for i in range(len(arr.Queues) - 1):

if arr[i + 1].firm < arr[i].firm:

ghost = arr[i + 1]

arr[i + 1] = arr[i]

arr[i] = ghost

swapped = True

if not swapped:

break

elapsed\_time = time.time() - start\_time

print(f" Метод Bubbly {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def DirectConnectionId(arr):

start\_time = time.time()

ghost = PowerSuply()

for i in range(len(arr.Queues)):

value = arr[i].id

index = i

while index > 0 and arr[index - 1].id > value:

ghost = arr[index]

arr[index] = arr[index - 1]

arr[index - 1] = ghost

index -= 1

elapsed\_time = time.time() - start\_time

print(f" Метод DirectConnection {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def DirectConnectionFirm(arr):

start\_time = time.time()

ghost = PowerSuply()

for i in range(len(arr.Queues)):

value = arr[i].firm

index = i

while index > 0 and arr[index - 1].firm > value:

ghost = arr[index]

arr[index] = arr[index - 1]

arr[index - 1] = ghost

index -= 1

elapsed\_time = time.time() - start\_time

print(f" Метод DirectConnection {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def ShakerSortId(arr):

start\_time = time.time()

swapped = True

start = 0

end = len(arr.Queues) - 1

ghost = PowerSuply()

while swapped:

swapped = False

for i in range(start, end):

if arr[i].id > arr[i + 1].id:

ghost = arr[i]

arr[i] = arr[i + 1]

arr[i + 1] = ghost

swapped = True

if not swapped:

break

swapped = False

end -= 1

for i in range(end - 1, start - 1, -1):

if arr[i].id > arr[i + 1].id:

ghost = arr[i]

arr[i] = arr[i + 1]

arr[i + 1] = ghost

swapped = True

start += 1

elapsed\_time = time.time() - start\_time

print(f" Метод ShakerSort {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def ShakerSortFirm(arr):

start\_time = time.time()

swapped = True

start = 0

end = len(arr.Queues) - 1

ghost = PowerSuply()

while swapped:

swapped = False

for i in range(start, end):

if arr[i].firm > arr[i + 1].firm:

ghost = arr[i]

arr[i] = arr[i + 1]

arr[i + 1] = ghost

swapped = True

if not swapped:

break

swapped = False

end -= 1

for i in range(end - 1, start - 1, -1):

if arr[i].firm > arr[i + 1].firm:

ghost = arr[i]

arr[i] = arr[i + 1]

arr[i + 1] = ghost

swapped = True

start += 1

elapsed\_time = time.time() - start\_time

print(f" Метод ShakerSort {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def ShallSortId(arr):

start\_time = time.time()

n = len(arr.Queues)

ghost = PowerSuply()

gap = n // 2

while gap > 0:

for i in range(gap, n):

ghost = arr[i]

j = i

while j >= gap and arr[j - gap].id > ghost.id:

arr[j] = arr[j - gap]

j -= gap

arr[j] = ghost

gap //= 2

elapsed\_time = time.time() - start\_time

print(f" Метод ShallSort {elapsed\_time \* 1000} миллисекунд ")

@staticmethod

def ShallSortFirm(arr):

start\_time = time.time()

n = len(arr.Queues)

ghost = PowerSuply()

gap = n // 2

while gap > 0:

for i in range(gap, n):

ghost = arr[i]

j = i

while j >= gap and arr[j - gap].firm > ghost.firm:

arr[j] = arr[j - gap]

j -= gap

arr[j] = ghost

gap //= 2

elapsed\_time = time.time() - start\_time

print(f" Метод ShallSort {elapsed\_time \* 1000} миллисекунд ")

n = int(input("Введите кол-во элементов: "))

test0 = queues(n)

test1 = queues(n)

test0.CopyTo(test1)

test2 = queues(n)

test0.CopyTo(test2)

test3 = queues(n)

test0.CopyTo(test3)

test4 = queues(n)

test0.CopyTo(test4)

test5 = queues(n)

test0.CopyTo(test5)

test1.GetRes()

Sorts.DirectChooseId(test1)

test1.GetRes("\n")

test2.GetRes()

Sorts.BubblyId(test2)

test2.GetRes("\n")

test3.GetRes()

Sorts.DirectConnectionId(test3)

test3.GetRes("\n")

test4.GetRes()

Sorts.ShakerSortId(test4)

test4.GetRes("\n")

test5.GetRes()

Sorts.ShallSortId(test5)

test5.GetRes("\n")

print()

test1.GetRes()

Sorts.DirectChooseFirm(test1)

test1.GetRes("\n")

test2.GetRes()

Sorts.BubblyFirm(test2)

test2.GetRes("\n")

test3.GetRes()

Sorts.DirectConnectionFirm(test3)

test3.GetRes("\n")

test4.GetRes()

Sorts.ShakerSortFirm(test4)

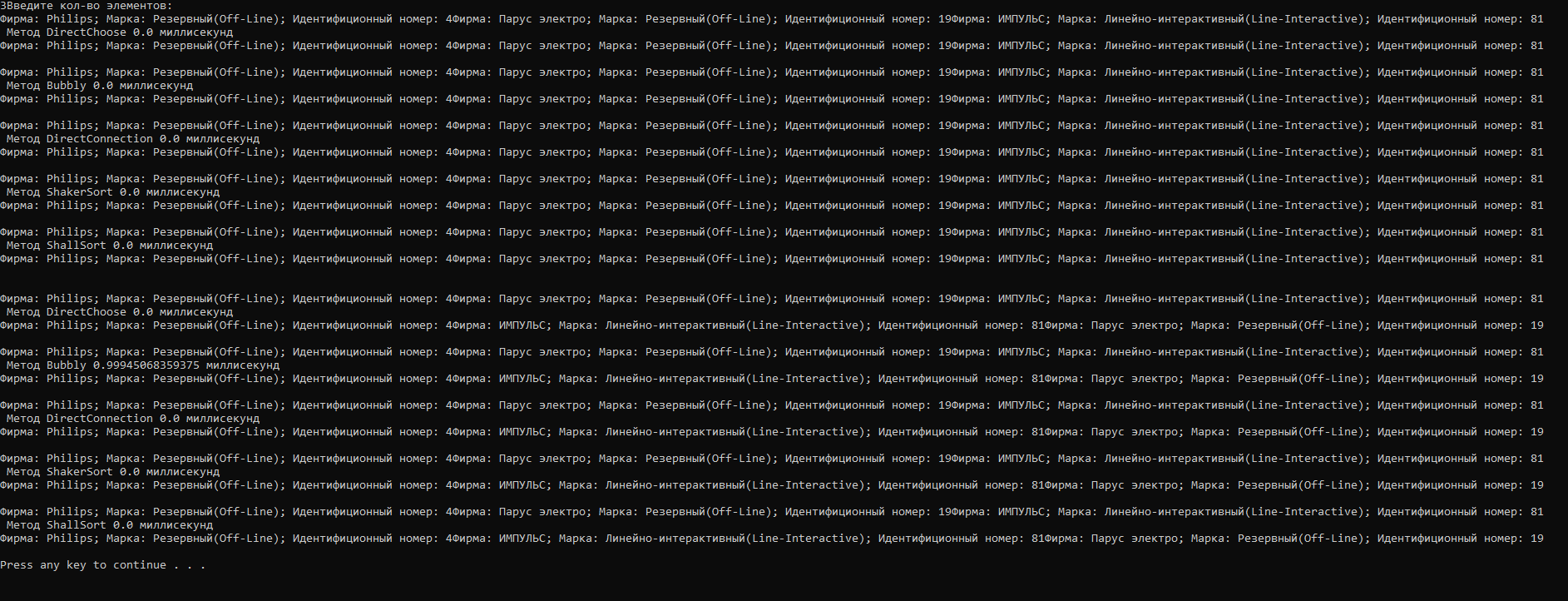
test4.GetRes("\n")

test5.GetRes()

Sorts.ShallSortFirm(test5)

test5.GetRes("\n")

Результат работы программы:



Вывод: Вывод: При выполнении работы не столкнулся с особыми трудностями и без особого труда написал код ,решающий поставленную задачу. Исходя из результата работы был сделан вывод, что для сортировки большого объема (30 элементов) типа данных наиболее эффективна сортировка методом Шелла как для строкового типа, так и для числового типа данных. Для сортировки данных меньшего объема (5 элементов) наибольшую эффективность показали метод прямого включения для числового типа данных и метод Шелла для строкового.