МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«Южно-Уральский государственный университет

(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

**ОТЧЕТ**

о выполнении практического задания № 2

по дисциплине

«Структуры и алгоритмы обработки данных»

|  |  |
| --- | --- |
|  | Выполнил:  студент группы КЭ-203  Старостенок Д.В.  Проверил:  ст. преподаватель кафедры СП  Петрова Л.Н. |

Челябинск – 2022

ОГЛАВЛЕНИЕ

[1. Задача № 1 “Односвязные списки” 2](#_Toc97760271)

[1.1. Физические величины: 2](#_Toc97760272)

[1.2. Область допустимых значений физических величин: 2](#_Toc97760273)

[1.3. Единицы измерения физических величин: 2](#_Toc97760274)

[1.4. Ограничения: 2](#_Toc97760275)

[1.5. Анализ ожидаемых результатов: 2](#_Toc97760276)

[1.6. Листинг программы: 3](#_Toc97760277)

[1.7. Контрольный тест: 12](#_Toc97760278)

[2. Задача № 2 “Стек”. 13](#_Toc97760279)

[2.1. Задача: 13](#_Toc97760280)

[2.2. Физические величины: 13](#_Toc97760281)

[2.3. Область допустимых значений физических величин: 13](#_Toc97760282)

[2.4. Единицы измерения физических величин: 13](#_Toc97760283)

[2.5. Ограничения: 13](#_Toc97760284)

[2.6. Анализ ожидаемых результатов: 13](#_Toc97760285)

[2.7. Листинг программы: 13](#_Toc97760286)

[2.8. Контрольный тест: 15](#_Toc97760287)

[Вывод 16](#_Toc97760288)

1. Задача № 1 “Односвязные списки”

Задача:

Вариант 34. Одна запись по постояльцам гостиницы содержит следующие поля: номер паспорта, ФИО, город прибытия, дата заселения в гостиницу, тип гостиничного номера (одноместный, двухместный или трехместный). Поиск и сортировка – по городу прибытия, типу гостиничного номера, дате заселения.

* 1. Физические величины:

int, string, bool

* 1. Область допустимых значений физических величин:

int – от -2147483648 до 2147483647

string – ~1 миллиард символов

bool – 1 из 2 значений

* 1. Единицы измерения физических величин:

int – целочисленный тип, 4 байта

string – строки, 2 ГБ

bool – 1 байт

* 1. Ограничения:

Поля, типы гостиничного номера, ссылочный тип хранения данных

* 1. Анализ ожидаемых результатов:

Односвязный список с функцией поиска и сортировки по городу прибытия, типу гостиничного номера, дате заселения

* 1. Листинг программы:

namespace singly\_linked\_list

{

class Program

{

class Hotel

{

readonly string[] \_typesOfHotelRooms = { "одноместный", "двухместный", "трехместный" }; // типы комнат

public int PassportId; // номер паспорта

public string FIO; // ФИО

public string ArrivalCity; // город прибытия

public DateTime Date; // дата заселения

public string RoomType; // тип комнаты

public Hotel(int passportId, string fio, string arrivalCity, DateTime date, string roomType)

{

PassportId = passportId;

FIO = fio;

ArrivalCity = arrivalCity;

Date = date;

RoomType = \_typesOfHotelRooms.Any(str => str == roomType.ToLower()) ? roomType : "трехместный";

}

// Ссылка на следующий Hotel

public Hotel Next { get; set; }

// Вывод экземпляра в консоль

public override string ToString()

{

return $"Номер паспорта - {PassportId}; ФИО - {FIO}; Город прибытия - {ArrivalCity}; " +

$"Дата заселения - {Date.Day}.{Date.Month}.{Date.Year}; Тип номера - {RoomType}";

}

}

class LinkedList : IEnumerable<Hotel>

{

// Начало списка

Hotel head = null;

// Конец списка

Hotel tail = null;

// Размер списка

int count = 0;

// Добавление нового элемента в конец списка

public void Add(Hotel hotel)

{

if (head == null) head = hotel;

else tail.Next = hotel;

tail = hotel;

count++;

}

public bool IsEmpty()

{

if (count == 0) return true;

else return false;

}

// Сортировка по городу прибытия

public void SortByArrivalCity()

{

Hotel test;

Hotel last;

Hotel prev;

Hotel curr;

Hotel end = null;

while (end != head)

{

last = prev = head;

while (prev.Next != end)

{

curr = prev.Next;

if (String.Compare(prev.ArrivalCity,curr.ArrivalCity) > 0)

{

prev.Next = curr.Next;

curr.Next = prev;

if (prev != head) last.Next = curr;

else head = curr;

test = curr;

curr = prev;

prev = test;

}

last = prev;

prev = prev.Next;

}

end = prev;

}

}

// Сортировка по типу гостинчного номера

public void SortByRoomType()

{

Hotel test;

Hotel last;

Hotel prev;

Hotel curr;

Hotel end = null;

while (end != head)

{

last = prev = head;

while (prev.Next != end)

{

curr = prev.Next;

if (String.Compare(prev.RoomType,curr.RoomType) > 0)

{

prev.Next = curr.Next;

curr.Next = prev;

if (prev != head) last.Next = curr;

else head = curr;

test = curr;

curr = prev;

prev = test;

}

last = prev;

prev = prev.Next;

}

end = prev;

}

}

// Сортировка по дате прибытия

public void SortByDate()

{

Hotel test;

Hotel last;

Hotel prev;

Hotel curr;

Hotel end = null;

while (end != head)

{

last = prev = head;

while (prev.Next != end)

{

curr = prev.Next;

if (DateTime.Compare(prev.Date, curr.Date) > 0)

{

prev.Next = curr.Next;

curr.Next = prev;

if (prev != head) last.Next = curr;

else head = curr;

test = curr;

curr = prev;

prev = test;

}

last = prev;

prev = prev.Next;

}

end = prev;

}

}

// Поиск по городу прибытия

public List<string> SearchByArrivalCity(string request)

{

List<string> result = new List<string>();

Hotel current = head;

while (current != null)

{

if (current.ArrivalCity == request) result.Add(current.FIO);

current = current.Next;

}

return result;

}

// Поиск по типу гостинчного номера

public List<int> SearchByRoomType(string request)

{

List<int> result = new List<int>();

Hotel current = head;

while (current != null)

{

if (current.RoomType == request) result.Add(current.PassportId);

current = current.Next;

}

return result;

}

// Поиск по дате прибытия

public List<string> SearchByDate(DateTime request)

{

List<string> result = new List<string>();

Hotel current = head;

while (current != null)

{

if (current.Date == request) result.Add(current.ArrivalCity);

current = current.Next;

}

return result;

}

// реализация интерфейса IEnumerable

IEnumerator IEnumerable.GetEnumerator()

{

return ((IEnumerable)this).GetEnumerator();

}

IEnumerator<Hotel> IEnumerable<Hotel>.GetEnumerator()

{

Hotel current = head;

while (current != null)

{

yield return current;

current = current.Next;

}

}

}

static void Main(string[] args)

{

LinkedList list = new();

bool end = true; // Условие выхода из цикла

while (end) // Меню для выбора действия

{

Console.WriteLine("\nВыберите действие: \n" +

" 1 - Добавить новый элемент \n" +

" 2 - Отсортировать по городу прибытия \n" +

" 3 - Отсортировать по типу гостинчного номера \n" +

" 4 - Сортировка по дате прибытия \n" +

" 5 - Поиск по городу прибытия \n" +

" 6 - Поиск по типу гостинчного номера \n" +

" 7 - Поиск по дате прибытия \n" +

" 8 - Закончить программу");

Console.Write(" Ваш выбор - ");

var choice = Console.ReadLine();

switch (choice)

{

// Добавление нового элемента

case "1":

bool check = true;

while (check)

{

Console.Write("Введите номер паспорта: ");

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

Console.Write("Введите ФИО: ");

string fio = Console.ReadLine();

Console.Write("Введите город прибытия: ");

string arrivalCity = Console.ReadLine();

Console.Write("Введите дату заселения: ");

string dateNumbs = Console.ReadLine();

string[] numbs = dateNumbs.Split(".");

DateTime date = new(Convert.ToInt32(numbs[2]), Convert.ToInt32(numbs[1]), Convert.ToInt32(numbs[0]));

Console.Write("Введите тип комнаты (одноместный, двухместный, трехместный):");

string roomType = Console.ReadLine();

list.Add(new(passportId, fio, arrivalCity , date, roomType));

Console.Write("Хотите продолжить?(y/n) ");

if (Console.ReadLine() == "n") check = false;

}

foreach (var item in list) Console.Write($"\n{item}");

break;

// Сортировка по городу прибытия

case "2":

if (!list.IsEmpty())

{

list.SortByArrivalCity();

foreach (var item in list) Console.Write($"\n{item}");

}

else Console.WriteLine("Список пуст");

break;

// Сортировка по типу комнат

case "3":

if (!list.IsEmpty())

{

list.SortByRoomType();

foreach (var item in list) Console.Write($"\n{item}");

}

else Console.WriteLine("Список пуст");

break;

// Сортировка по дате

case "4":

if (!list.IsEmpty())

{

list.SortByDate();

foreach (var item in list) Console.Write($"\n{item}");

}

else Console.WriteLine("Список пуст");

break;

// Поиск по городу прибытия

case "5":

if (!list.IsEmpty())

{

Console.Write("Введите город, который нужно найти: ");

string search = Console.ReadLine();

if (list.SearchByArrivalCity(search).Count != 0) foreach (var item in list.SearchByArrivalCity(search))

Console.WriteLine($"В этот город прилетаю люди с ФИО: {item}");

else Console.WriteLine("Город отсутсвует в списке");

}

else Console.WriteLine("Список пуст");

break;

// Поиск по типу комнат

case "6":

if (!list.IsEmpty())

{

Console.Write("Введите тип комнаты, который нужно найти: ");

string search = Console.ReadLine();

if (list.SearchByRoomType(search).Count != 0) foreach (var item in list.SearchByRoomType(search))

Console.WriteLine($"Номера паспортов зарегестрированные на эти команаты: {item}");

else Console.WriteLine("Тип комнат отсутсвует в списке");

}

else Console.WriteLine("Список пуст");

break;

// Поиск по дате

case "7":

if (!list.IsEmpty())

{

Console.Write("Введите дату заселения, которую нужно найти: ");

string search = Console.ReadLine();

string[] numbsSearch = search.Split(".");

DateTime dateSearch = new(Convert.ToInt32(numbsSearch[2]), Convert.ToInt32(numbsSearch[1]), Convert.ToInt32(numbsSearch[0]));

if (list.SearchByDate(dateSearch).Count != 0) foreach (var item in list.SearchByDate(dateSearch))

Console.WriteLine($"Города, в которые происходит прибытие в эту дату: {item}");

else Console.WriteLine("На эту дату нет нет прибытий в города");

}

else Console.WriteLine("Список пуст");

break;

// Окончание цикла

case "8":

end = false;

break;

// Непредусмотренный ввод

default:

Console.WriteLine("Ошибка ввода");

break;

}

}

}

}

}

* 1. Контрольный тест:

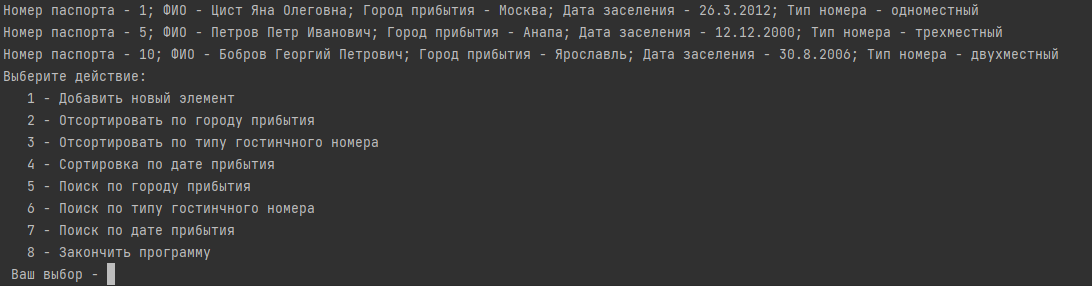


Рис. 1. Меню с заполненными данными

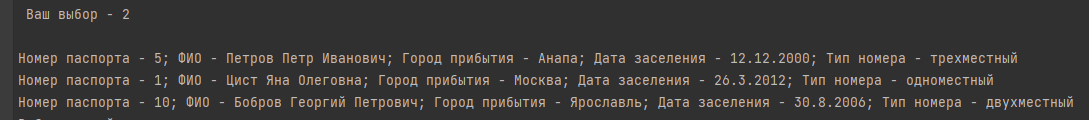


Рис. 2. Сортировка по городу прибытия

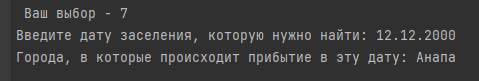


Рис. 3. Поиск по дате прибытия

1. Задача № 2 “Стек”.
   1. Задача:

Вариант 34. Сформировать стек из N чисел. Найти среднее арифметическое элементов стека и поместить его в начало стека

* 1. Физические величины:

int, double

* 1. Область допустимых значений физических величин:

int – от -2147483648 до 2147483647

double – ±5.0\*10-324 до ±1.7\*10308

* 1. Единицы измерения физических величин:

int – целочисленный тип, 4 байта

double – число с плавающей точкой, 8 байта

* 1. Ограничения:

N количество чисел, стек

* 1. Анализ ожидаемых результатов:

Стек, в начале которого находится среднее арифметическое элементов стека

* 1. Листинг программы:

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите количество чисел:");

int n = Convert.ToInt32(Console.ReadLine()); // Считываем количество чисел в стеке

Stack<double> numbers = new Stack<double>(); // Инициализация стека

Stack<double> temp = new Stack<double>(); // Инициализация временного стека

Random rnd = new Random();

double sum = 0; // Сумма чисел в стеке

for (int i = 0; i < n; i++) // Добавление случайных чисел в стек

{

numbers.Push(rnd.Next(-10, 10));

}

Console.WriteLine("Содержимое стека:");

foreach (var number in numbers) // Вывод содержимого стека и подсчет суммы положительных элементов

{

Console.WriteLine(number);

sum += number; // Считаем сумму чисел в стеке

temp.Push(number); // Добавляем числа из основого стека во временный

}

double nToDouble = Convert.ToDouble(n);

double average = sum/nToDouble; // Среднее элементов стека

numbers.Clear(); // Очищаем основной стек

numbers.Push(average); // Добавляем среднее число в основной стек

Console.WriteLine($"Среднее арифметическое стека = {average}");

foreach (var number in temp) // Возврат чисел из временного стека в основной

{

numbers.Push(number);

}

Console.WriteLine("Содержимое стека с добавленным средним:");

foreach (var number in numbers) // Вывод содержимого стека

{

Console.WriteLine(number);

}

}

}

* 1. Контрольный тест:

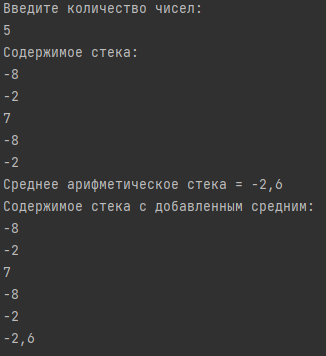


Рис. 4. Тест подсчета среднего с 5 числами

Вывод

В результате проделанной работы были получены навыки реализации и работы с односвязными списками и стеками. По заданию созданы 2 программы, которые помогли закрепить полученные навыки.