|  |  |
| --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-02 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | О |  | Естественнонаучный |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Информационные технологии и программирование | | |

|  |
| --- |
| ПРАКТИЧЕСКАЯ РАБОТА №2 |
| РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РАБОТЫ СО ВСТРОЕННЫМИ |
| КЛАССАМИ, ОБЕСПЕЧИВАЮЩИМИ РАБОТУ С ФАЙЛАМИ, |
| С ИСПОЛЬЗОВАНИЕМ ПОТОКОВ |
| Вариант №24 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | И924Б |
| Фокин В. К. | | | | | |
| Фамилия И.О. | | | | | |
| **Преподаватель:** | | | | | |
| Гладевич А. А. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2023г. |

САНКТ-ПЕТЕРБУРГ

2023 г.

**Постановка задачи**

Написать программу для работы с данными, представленными в виде набора структур, содержимое которых соответствует индивидуальному варианту. Данные считываются из файла и заносятся в указанную в варианте коллекцию из числа стандартных. При старте программы пользователю предлагается ввести имя файла. Если файл не существует, создаётся новый. После этого пользователю предлагается меню для работы с данными. Обязательные пункты меню: отображение содержимого коллекции, добавление нового элемента, удаления элемента с указанным индексом, корректировка элемента, работа с коллекцией, дополнительные пункты указаны в варианте. После работы необходимо сохранять содержимое коллекции в указанные ранее пользователем файлы. Вывод данных на экран должен выполняться в формате таблицы.

Обязательные к реализации возможности списка:

– добавление в начало списка;

– добавление в произвольное место;

– удаление из произвольного места;

– сортировка.

Вариант 24. Цветы.

Коллекция – букет (в виде списка).

Поля данных структуры: название, цвет, количество, цена.

Дополнительные пункты меню:

− вывести общую стоимость букета и преобладающий цвет в нём;

− посчитать количество красных цветов в букете.

**Текст программы**

using System;

using System.IO;

using System.Collections.Generic;

namespace pr2

{

struct flower

{

private string name;

private string color;

private int number;

private double price;

public string Name

{

set { name = value; }

get { return name; }

}

public string Color

{

set { color = value; }

get { return color; }

}

public int Number

{

set { number = value; }

get { return number; }

}

public double Price

{

set { price = value; }

get { return price; }

}

}

internal class Program

{

static void Main(string[] args)

{

string fileName = @"C:\c#\pr2\"; //расположение

string menu, MENU, umen;

List<flower> bouquet = new List<flower>(); //коллекция список

Console.WriteLine("Введите имя файла:");

fileName += Console.ReadLine();

if (File.Exists(fileName)) //если файл существует, то читаем, иначе создадим новый

{

Console.WriteLine("Открытие файла...");

read(fileName, bouquet);

}

else

{

Console.WriteLine("Создание файла...");

}

do

{

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

Console.WriteLine("| МЕНЮ |");

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

Console.WriteLine("| 1. Добавить элемент |");

Console.WriteLine("| 2. Вывод всех элементов коллекции |");

Console.WriteLine("| 3. Удаление элемента с указанным номером |");

Console.WriteLine("| 4. Корректировка элемента |");

Console.WriteLine("| 5. Сортировать список |");

Console.WriteLine("| 6. Узнать общую стоимость букета и преобладающий цвет |");

Console.WriteLine("| 7. Узнать количество красных цветов |");

Console.WriteLine("| 0. Сохранить и выйти |");

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

menu = Console.ReadLine();

switch (menu)

{

case "1":

Console.Clear();

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

Console.WriteLine("| 1. В начало |");

Console.WriteLine("| 2. В произвольное место |");

Console.WriteLine("| 3. В конец |");

Console.WriteLine("| 0. Выход |");

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

MENU = Console.ReadLine();

switch (MENU)

{

case "1":

Console.Clear();

flower flow1 = input();

addBegin(bouquet, flow1);

break;

case "2":

Console.Clear();

flower flow2 = input();

addAnyPlace(bouquet, flow2);

break;

case "3":

Console.Clear();

flower flow3 = input();

bouquet.Add(flow3);

break;

case "0":

break;

default:

Console.WriteLine("Пункта меню не существует");

break;

}

break;

case "2":

Console.Clear();

show(bouquet);

break;

case "3":

Console.Clear();

show(bouquet);

remove(bouquet);

break;

case "4":

Console.Clear();

show(bouquet);

edit(bouquet);

break;

case "5":

Console.Clear();

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

Console.WriteLine("| 1. Сортировать по возрастанию цены |");

Console.WriteLine("| 2. Сортировать по убыванию цены |");

Console.WriteLine("| 3. Сортировать по возрастанию количества цветов |");

Console.WriteLine("| 4. Сортировать по убыванию количества цветов |");

Console.WriteLine("| 0. Выход |");

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

umen = Console.ReadLine();

switch (umen)

{

case "1":

Console.Clear();

sortPriceMore(bouquet);

break;

case "2":

Console.Clear();

sortPriceLess(bouquet);

break;

case "3":

Console.Clear();

sortCountMore(bouquet);

break;

case "4":

Console.Clear();

sortCountLess(bouquet);

break;

case "0":

break;

default:

Console.WriteLine("Пункта меню не существует");

break;

}

break;

case "6":

Console.Clear();

double price = totalPrice(bouquet);

string predominantColor = dominantColor(bouquet);

Console.WriteLine("Общая стоимость букета: " + price);

Console.WriteLine("Преобладающий цвет: " + predominantColor);

break;

case "7":

Console.Clear();

Console.WriteLine("Количество красных цветов в букете: " + сountRed(bouquet));

break;

case "0":

write(fileName, bouquet);

Console.ReadLine();

break;

default:

Console.WriteLine("Пункта меню не существует");

break;

}

} while (menu != "0");

}

static List<flower> read(string filename, List<flower> bouq) //чтение существующего файла

{

try

{

FileStream fs = new FileStream(filename, FileMode.Open); //открываем поток

BinaryReader br = new BinaryReader(fs);

while (fs.Position < fs.Length) //пока не пройдем по всему потоку добавляем в список

{

flower flow = new flower();

flow.Name = br.ReadString();

flow.Color = br.ReadString();

flow.Number = br.ReadInt32();

flow.Price = br.ReadDouble();

bouq.Add(flow);

}

Console.WriteLine("Данные успешно считаны");

fs.Close(); //закрываем

}

catch (Exception ex) //пишет ошибку

{

Console.WriteLine("Произошла ошибка при чтении файла\n" + ex.Message);

}

return bouq;

}

static void write(string filename, List<flower> bouq) //создание и запись в новый файл

{

try

{

FileStream fs = new FileStream(filename, FileMode.OpenOrCreate); //открываем или создаем поток

BinaryWriter bw = new BinaryWriter(fs);

foreach (flower flow in bouq)

{

bw.Write(flow.Name);

bw.Write(flow.Color);

bw.Write(flow.Number);

bw.Write(flow.Price);

}

Console.WriteLine("Данные успешно записаны в файл.");

fs.Close(); //закрываем

}

catch (Exception ex) //пишет ошибку

{

Console.WriteLine("Произошла ошибка при записи файла\n" + ex.Message);

}

}

static flower input() //добавление структуры в список

{

bool norm;

flower flow = new flower();

Console.WriteLine("Введите название цветка: ");

flow.Name = Console.ReadLine();

Console.WriteLine("Введите цвет:");

flow.Color = Console.ReadLine();

do

{

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

if (int.TryParse(Console.ReadLine(), out int number) && number > 0)

{

flow.Number = number;

norm = true;

}

else

{

Console.WriteLine("Введите целое значение больше 0");

norm = false;

}

} while (norm == false);

do

{

Console.WriteLine("Введите цену цветка:");

if (double.TryParse(Console.ReadLine(), out double price) && price > 0)

{

flow.Price = price;

norm = true;

}

else

{

Console.WriteLine("Введите значение больше 0");

norm = false;

}

} while (norm == false);

return flow;

}

static void addBegin(List<flower> bouq, flower flow) //добавление в начало

{

bouq.Insert(0, flow);

Console.WriteLine("Элемент успешно добавлен в начало списка.");

}

static void addAnyPlace(List<flower> bouq, flower flow) //добавление в выбранное место

{

int pos;

do

{

Console.WriteLine("Введите позицию: ");

if (int.TryParse(Console.ReadLine(), out pos) && pos - 1 >= 0 && pos - 1 <= bouq.Count)

{

bouq.Insert(pos - 1, flow);

Console.WriteLine("Элемент добавлен в список на позицию " + pos);

}

else

{

Console.WriteLine("Недопустимая позиция. Элемент не добавлен");

}

} while (pos - 1 < 0 || pos - 1 > bouq.Count);

}

static void show(List<flower> bouq) //вывод списка

{

if (bouq.Count == 0)

{

Console.WriteLine("Список пуст");

return;

}

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

Console.WriteLine("| № | Название | Цвет | Количество | Цена |");

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

for (int i = 0; i < bouq.Count; i++)

{

Console.WriteLine("| " + $"{i + 1,2}" + " | " + $"{bouq[i].Name,10}" + " | " + $"{bouq[i].Color,10}"

+ " | " + $"{bouq[i].Number,6}" + " | " + $"{bouq[i].Price,5}" + " |");

}

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

}

static void remove(List<flower> bouq) //удаление из выбранного места

{

int pos;

Console.WriteLine("Введите позицию для удаления: ");

if (int.TryParse(Console.ReadLine(), out pos) && pos - 1 >= 0 && pos - 1 <= bouq.Count)

{

bouq.RemoveAt(pos - 1);

Console.WriteLine("Элемент удален с позиции " + pos);

}

else

{

Console.WriteLine("Недопустимая позиция. Элемент не удален");

}

}

static void edit(List<flower> bouq) //корректировка элемента

{

int pos;

do

{

Console.WriteLine("Введите позицию: ");

if (int.TryParse(Console.ReadLine(), out pos) && pos - 1 >= 0 && pos - 1 <= bouq.Count)

{

flower flow = input();

bouq[pos - 1] = flow;

Console.WriteLine("Элемент на позиции " + pos + " изменен");

}

else

{

Console.WriteLine("Недопустимая позиция. Элемент не изменен");

}

} while (pos - 1 < 0 || pos - 1 > bouq.Count);

}

static void sortPriceMore(List<flower> bouq) //сортировка по цене

{

bouq.Sort((flower1, flower2) => flower1.Price.CompareTo(flower2.Price));

Console.WriteLine("Список отсортирован по возрастанию цены");

}

static void sortPriceLess(List<flower> bouq) //сортировка по цене

{

bouq.Sort((flower1, flower2) => flower2.Price.CompareTo(flower1.Price));

Console.WriteLine("Список отсортирован по убыванию цены");

}

static void sortCountMore(List<flower> bouq) //сортировка по количеству

{

bouq.Sort((flower1, flower2) => flower1.Number.CompareTo(flower2.Number));

Console.WriteLine("Список отсортирован по возрастанию количества цветов");

}

static void sortCountLess(List<flower> bouq) //сортировка по количеству

{

bouq.Sort((flower1, flower2) => flower2.Number.CompareTo(flower1.Number));

Console.WriteLine("Список отсортирован по убыванию количества цветов");

}

static double totalPrice(List<flower> bouq) //стоимость букета

{

double totalPrice = 0;

foreach (flower flow in bouq)

{

totalPrice += flow.Price \* flow.Number;

}

return totalPrice;

}

public class CaseInsensitiveStringEqualityComparer : IEqualityComparer<string> //для игнорирования регистра

{

public bool Equals(string x, string y)

{

return string.Equals(x, y, StringComparison.OrdinalIgnoreCase);

}

public int GetHashCode(string obj)

{

return obj.ToLower().GetHashCode();

}

}

static string dominantColor(List<flower> bouq) //преобладающий цвет

{

Dictionary<string, int> colorCounts = new Dictionary<string, int>(new CaseInsensitiveStringEqualityComparer());

foreach (flower flow in bouq)

{

string color = flow.Color;

if (colorCounts.ContainsKey(color))

{

colorCounts[color] += flow.Number;

}

else

{

colorCounts[color] = 0;

colorCounts[color] += flow.Number;

}

}

string dominantColor = "";

int maxCount = 0;

foreach (var key in colorCounts)

{

if (key.Value > maxCount)

{

maxCount = key.Value;

dominantColor = key.Key;

}

}

return dominantColor;

}

static int сountRed(List<flower> bouq) //счетчик красных цветов

{

int redCount = 0;

foreach (flower flow in bouq)

{

if (flow.Color.ToLower() == "красный" || flow.Color.ToLower() == "Красный" || flow.Color.ToLower() == "red" || flow.Color.ToLower() == "Red") // Проверяем, является ли цвет красным (может потребоваться настройка регистра)

{

redCount += flow.Number;

}

}

return redCount;

}

}

}

**Результат работы программы**

При запуске программы нужно ввести название файла. Если файл существует, то он будет открыт, если не существует, то произойдет создание файла. После ввода названия файла появится главное меню. Запуск программы показан на рисунке 1.

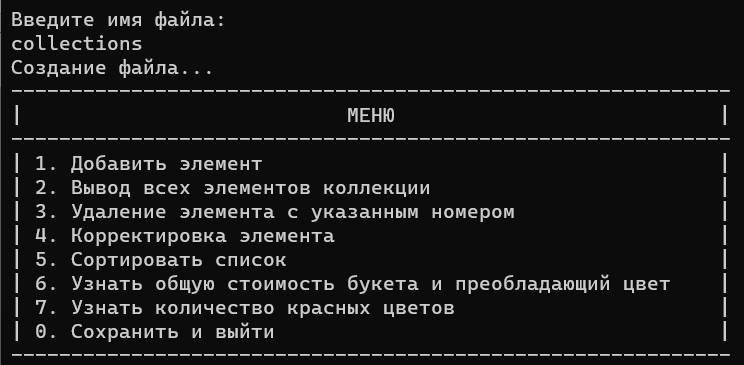


Рисунок 1 − Запуск программы и главное меню

При выборе пункта «1» можно добавить элемент в список. Сначала нужно выбрать, куда добавить элемент. При выборе «1» элемент добавится в начало, при выборе «2» элемент можно добавить в произвольное место, при выборе «3» элемент добавится в конец. Меню выбора показано на рисунке 2.

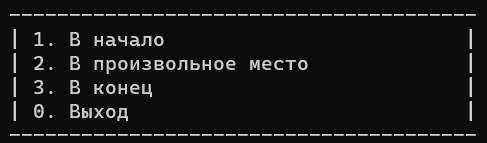


Рисунок 2 – Меню выбора

При добавлении требуется ввести название цветка, его цвет, количество цветов в букете и стоимость цветка. Добавление показано на рисунке 3.

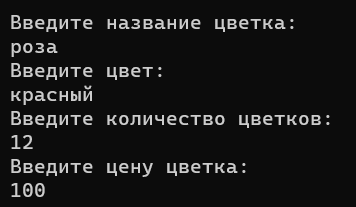


Рисунок 3 – Добавление элемента

При добавлении элемента в произвольное место также нужно ввести номер позиции, куда нужно вставить элемент. Добавление в произвольное место показано на рисунке 4.

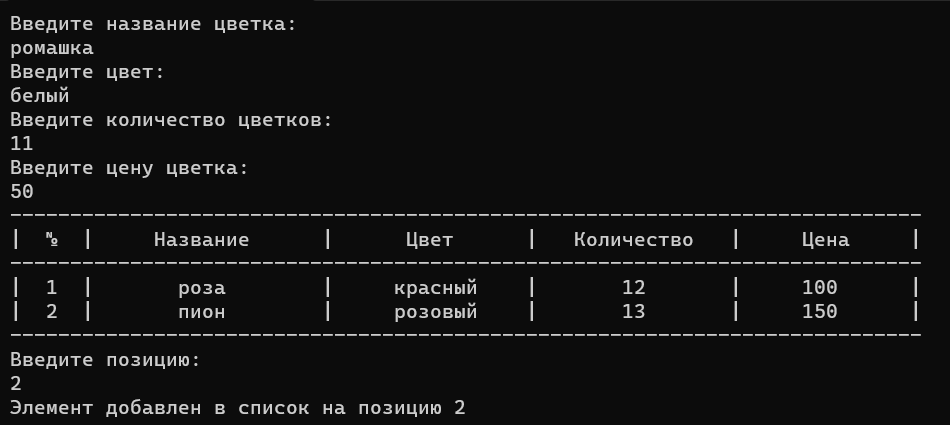


Рисунок 4 – Добавление в произвольное место

Если ввести отрицательный или слишком большой номер позиции, появится сообщение об ошибке. Сообщение показано на рисунке 5.

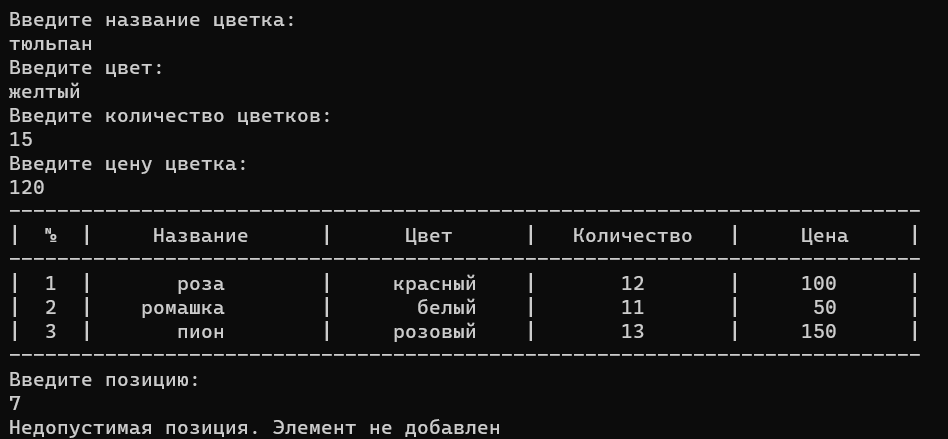


Рисунок 5 – Сообщение об ошибке

При выборе пункта «2» выводится список элементов. Вывод показан на рисунке 6.

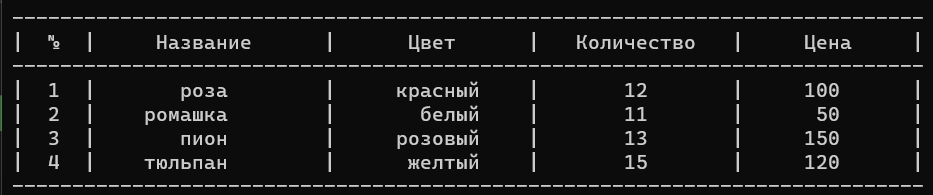


Рисунок 6 – Вывод всех элементов

При выборе пункта «3» можно удалить элемент с выбранного места. Удаление показано на рисунке 7.

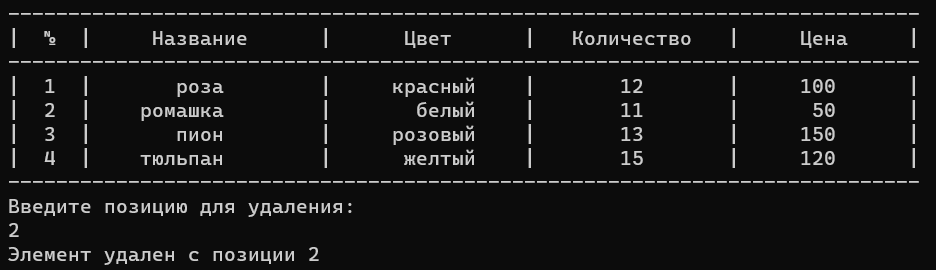


Рисунок 7 – Удаление с выбранной позиции

При выборе пункта «4» можно скорректировать выбранный элемент. Нужно заново заполнить поля. Коррекция элемента показана на рисунке 8.

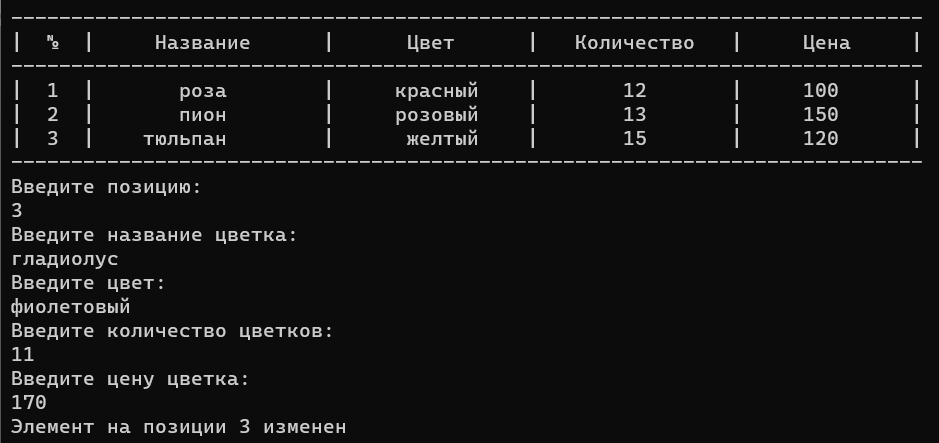


Рисунок 8 – Коррекция элемента

При выборе пункта «5» можно отсортировать список. Нужно выбрать тип сортировки: по возрастанию или убыванию цены, по возрастанию или убыванию количества в списке. Сортировка показана на рисунках 9 и 10.

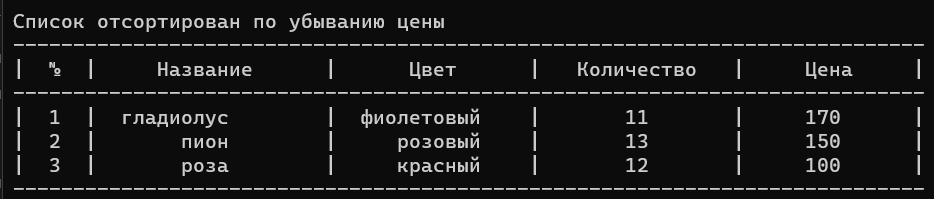


Рисунок 9 – Сортировка по возрастанию

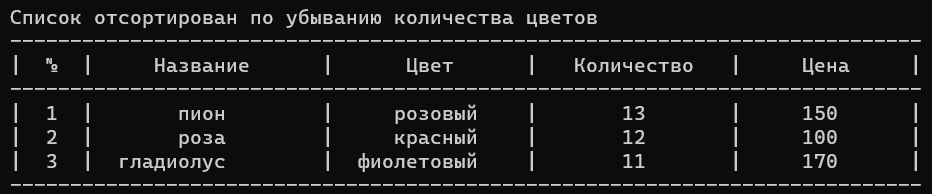


Рисунок 10 – Сортировка по убыванию

При выборе пункта «6» можно узнать общую стоимость букета и преобладающий цвет. Общая стоимость состоит из суммы цен, помноженных на количество. Преобладающий цвет высчитывается по максимальному количеству цветов с цветом. Регистр не имеет значения. Результат показан на рисунке 11.

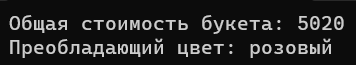


Рисунок 11 – Стоимость букета и преобладающий цвет

При выборе пункта «7» можно узнать количество цветов красного цвета. Результат показан на рисунке 12.



Рисунок 12 – Количество цветов красного цвета

При выборе несуществующего пункта меню выводится сообщение об ошибке. Сообщение показано на рисунке 13.

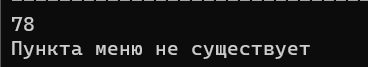


Рисунок 13 – Выбор несуществующего пункта меню

При выборе пункта «0» программа сохранит список в файл и прекратит работу.