1. **Коллекции. Классы-прототипы**

Задание 1. Решить задачи с использованием класса Stack.

Пусть символ # определен в текстовом редакторе как стирающий символ Backspace, т.е. строка abc#d##c в действительности является строкой ac. Дан текст, в котором встречается символ #. Преобразовать его с учетом действия этого символа.

Листинг программы:

namespace Task\_1

{

class Program

{

public static void Main(string[] args)

{

Console.Write("Введите текст: ");

string text = Console.ReadLine();

Console.WriteLine(TextTransformation(text));

}

public static string TextTransformation(string text)

{

Stack<char> stack = new Stack<char>();

foreach (char c in text)

{

if (c == '#')

{

if (stack.Count > 0)

stack.Pop();

}

else

{

stack.Push(c);

}

}

string result = new string(stack.Reverse().ToArray());

return result;

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| При#ушел##и##тик | Прутик |

Анализ результатов:



Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

Задание 2. Решить следующие задачи с использованием класса Queue.

Четные варианты:

Дан файл, содержащий числа. За один просмотр файла напечатать элементы файла в следующем порядке: сначала все числа, из интервала [a,b], потом все числа, меньшие a, потом все числа, большие b, сохраняя исходный порядок в каждой группе чисел.

Листинг программы:

namespace Task\_2

{

class Program

{

public static void Main(string[] args)

{

Console.Write("Введите нижнюю нраницу: ");

int numA = int.Parse(Console.ReadLine());

Console.Write("Введите верзнюю нраницу: ");

int numB = int.Parse(Console.ReadLine());

Queue<int> interval = new Queue<int>();

Queue<int> lessThanA = new Queue<int>();

Queue<int> greaterThanB = new Queue<int>();

string path = @"..\..\..\Numbers.txt";

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

int number = int.Parse(sr.ReadLine());

if (number >= numA && number <= numB)

{

interval.Enqueue(number);

}

else if (number < numA)

{

lessThanA.Enqueue(number);

}

else if (number > numB)

{

greaterThanB.Enqueue(number);

}

}

}

Console.WriteLine("\nИз интервала");

foreach (int number in interval)

{

Console.Write(number + " ");

}

Console.WriteLine($"\nМеньше {numA}");

foreach (int number in lessThanA)

{

Console.Write(number + " ");

}

Console.WriteLine($"\nБольше {numB}");

foreach (int number in greaterThanB)

{

Console.Write(number + " ");

}

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 34 | 34 6 5 7 8 5  3 4 0 -3 -50  56 49 70 |

Анализ результатов:

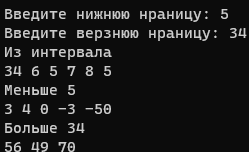


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Создать класс-прототип, с обобщенным методом, с методами добавления, удаления, клонирования экземпляров класса.

В основной программе продемонстрировать:

функциональность созданного класса и всех его методов;

использование обнуляемых типов;

использование абстрактных структур (List<P> Dictionary<Key,Value>. Игра.

Листинг программы:

using System.Collections;

namespace Task\_3

{

class Program

{

public static void Main(string[] args)

{

try

{

Game gameList = new Game();

Dictionary<int, string> dGame = new Dictionary<int, string>(3);

for (int i = 0; i < 3; i++)

{

Score score = new Score();

Console.Write("Введите имя игрока: ");

score.Name = Console.ReadLine();

Console.Write("Введите действие: ");

score.Status = Console.ReadLine();

Console.Write("Введите кол-во очков: ");

score.Scoree = int.Parse(Console.ReadLine());

gameList.Add(score);

dGame[i + 1] = score.status;

}

foreach (KeyValuePair<int, string> keyValue in dGame)

{

Console.WriteLine($"{keyValue.Key} - {keyValue.Value}");

}

Print(gameList);

Console.Write("Вышел из игры игрок под номером: ");

int num = int.Parse(Console.ReadLine());

gameList.RemoveAt(num);

Game game = (Game)gameList.Clone();

Print(game);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static void Print(Game list)

{

foreach (Score score in list)

{

Console.WriteLine(score.GetGame());

}

}

}

public class Gaming

{

public string name;

public string status;

public Gaming()

{

name = Name;

status = Status;

}

public string Name

{

get { return name; }

set

{

if (value == "")

name = "Не указано";

else

name = value;

}

}

public string Status

{

get { return status; }

set

{

if (value == "")

status = "Ожидает";

else

status = value;

}

}

virtual public string GetGame()

{

return $"Игрок {name} сейчас {status}";

}

}

public class Score : Gaming

{

public int score;

public Score() : base()

{

name = Name;

status = Status;

score = Scoree;

}

public int Scoree

{

get { return score; }

set

{

if (value < 0)

score = 0;

else

score = value;

}

}

public override string GetGame()

{

return $"{base.GetGame()}, на счету {score} очко(в).";

}

}

class Game : IEnumerable

{

private List<Score> list;

public Game()

{

list = new List<Score>();

}

public Game(List<Score> list)

{

this.list = list;

}

public void Add(Score score)

{

list.Add(score);

}

public Game Clone()

{

return new Game(list);

}

public void RemoveAt(int i)

{

i--;

list.RemoveAt(i);

}

public void Clear()

{

list.Clear();

}

public IEnumerator GetEnumerator()

{

return list.GetEnumerator();

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Вася Ждет 120 Петя Ходит 101 Игорь 79 2 | Ждет Ходит Ожидает Вася Петя Игорь  Вася Игорь |

Анализ результатов:

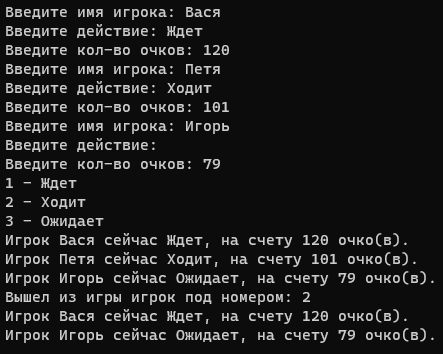


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. Решить задачу с использованием класса HashTable.

Реализовать простейший каталог музыкальных компакт-дисков, который позволяет: добавлять и удалять диски, добавлять и удалять песни, просматривать содержимое целого каталога и каждого диска в отдельности.

Листинг программы:

using System.Collections;

namespace Task\_4

{

class Program

{

public static void Main(string[] args)

{

Catalog catalog = new Catalog();

for (int i = 0; i < 2; i++)

{

CD cd = new CD();

Console.Write("Введите название диска: ");

cd.Title = Console.ReadLine();

Console.Write("Введите исполнителя: ");

cd.Artist = Console.ReadLine();

Console.Write("Введите год: ");

cd.Year = int.Parse(Console.ReadLine());

cd.Songs = GetSetSongs();

catalog.AddCD(cd);

}

Console.WriteLine($"Содержимое каталога\n{catalog.GetCatalog()}");

Console.Write("Введите название для отображения диска: ");

string name = Console.ReadLine();

Console.WriteLine(catalog.GetCD(name));

Console.Write("Введите название диска: ");

name = Console.ReadLine();

Console.Write("Введите название песни для добавления: ");

string song = Console.ReadLine();

catalog.AddSong(name, song);

Console.Write("Введите название диска: ");

name = Console.ReadLine();

Console.Write("Введите название песни для удаления: ");

song = Console.ReadLine();

catalog.RemoveSong(name, song);

Console.Write("Введите название диска для удаления: ");

name = Console.ReadLine();

catalog.RemoveCD(name);

Console.WriteLine($"Содержимое обновленного каталога\n{catalog.GetCatalog()}");

}

public static List<string> GetSetSongs()

{

Console.Write("Введите композицию: ");

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

string s = "";

while (!string.IsNullOrEmpty(s = Console.ReadLine()))

{

list.Add(s);

Console.Write("Введите композицию: ");

}

return list;

}

}

class CD

{

public string title;

public string artist;

public int year;

public List<string> songs;

public CD()

{

title = Title;

artist = Artist;

year = Year;

songs = Songs;

}

public string Title

{

get { return title; }

set

{

if (title == "")

title = "Не указан";

else

title = value;

}

}

public string Artist

{

get { return artist; }

set

{

if (artist == "")

artist = "Не указан";

else

artist = value;

}

}

public int Year

{

get { return year; }

set

{

if (year < 0)

year = 0;

else

year = value;

}

}

public List<string> Songs

{

get { return songs; }

set

{

songs = value;

}

}

}

class Catalog

{

private Hashtable \_cdt;

public Catalog()

{

\_cdt = new Hashtable();

}

public void AddCD(CD cd)

{

\_cdt.Add(cd.Title, cd);

}

public void RemoveCD(string title)

{

\_cdt.Remove(title);

}

public void AddSong(string title, string song)

{

if (\_cdt.ContainsKey(title))

{

CD cd = (CD)\_cdt[title];

cd.Songs.Add(song);

}

else

{

Console.WriteLine("Диск не найден");

}

}

public void RemoveSong(string title, string song)

{

if (\_cdt.ContainsKey(title))

{

CD cd = (CD)\_cdt[title];

cd.Songs.Remove(song);

}

else

{

Console.WriteLine("Композиция не найдена");

}

}

public string GetSong(CD cd)

{

string songs = "";

foreach (string song in cd.songs)

songs += song + " / ";

return songs;

}

public string GetCatalog()

{

string catalog = "";

foreach (DictionaryEntry entry in \_cdt)

{

CD cd = (CD)entry.Value;

catalog += $"Группа: {cd.title}, Исполнители: {cd.artist}, Год: {cd.year}\nКомпозиции: {GetSong(cd)}\n";

}

return catalog;

}

public string GetCD(string title)

{

if (\_cdt.ContainsKey(title))

{

CD cd = (CD)\_cdt[title];

return $"Группа: {cd.title}, Исполнители: {cd.artist}\nГод: {cd.year}, Композиции: {GetSong(cd)}";

}

else

{

return "Композиция не найдена";

}

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Арабеске Настя 1986 Чучунда Выше Кривуля Нирвана Нирвана 1977 Вперед Крутящий момент  Нирвана  Арабеске Вьюга Нирвана Вперед Нирвана | Содержимое каталога  Содержимое диска  Содержимое обновленного каталога |

Анализ результатов:

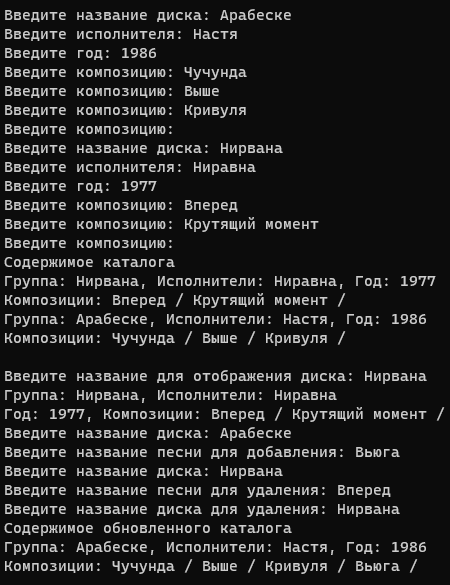


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка