|  |  |
| --- | --- |
| *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 №3 |
| СРАВНЕНИЕ ПОДХОДОВ К ОСВОБОЖДЕНИЮ |
| РЕСУРСОВ В ЯЗЫКАХ C++ И C# |
| 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**

Работа состоит из двух частей: разобраться в предоставленном примере и дополнить программу из предыдущей работы.

**Первая часть**

Рассмотреть представленный ниже пример, построить диаграмму времени жизни объектов и ответить на следующие вопросы.

Какое максимальное поколение объектов в ходе выполнения программы было выявлено? Сколько их в C# всего?

Что будет, если закомментировать строчку GC.Collect(0);? Изменится ли вывод программы, если да, то как и почему?

Что будет, если закомментировать строчку GC.Collect(2)? Изменится ли вывод программы, если да, то как и почему?

Изменить параметр метода GC.GetTotalMemory() с true на false. На что это влияет? В методе MakeSomeGarbage() добавить к объекту vt создание еще одного любого объекта, например класса StringBuilder. Что изменилось в выводе программы?

**Ответы на вопросы**

Максимальное поколение объектов – 2. Всего их три: 0, 1 и 2.

Если закомментировать строчку GC.Collect(0); то сборщик мусора в нулевом поколении не будет вызван явно.

Если закомментировать строчку GC.Collect(2); то сборщик мусора во втором поколении не будет вызван явно.

Если изменить параметр метода GC.GetTotalMemory() с true на false, это означает, что метод вернет текущее количество байтов, выделенных для управления памятью сборщиком мусора.

Если добавить создание дополнительного объекта в метод MakeSomeGarbage(), то общее количество созданных объектов будет увеличено. Это приведет к тому, что больше мусора будет размещено в поколении 0.

**Диаграмма времени жизни объектов**

Диаграмма показывает, сколько памяти освобождается при сборке мусора из Gen 0, Gen 1 и Gen 2. Диаграмма показана на рисунке 1.

Рисунок 1 – Диаграмма времени жизни объектов

**Текст программы**

using System;

using System.Runtime.InteropServices;

namespace GarbageCollectorInCSharp

{

class GCProgram

{

private const long maxGarbage = 1000;

static void Main()

{

GCProgram myGCCol = new GCProgram();

Console.WriteLine("The highest generation is {0}", GC.MaxGeneration);

myGCCol.MakeSomeGarbage();

Console.WriteLine("Generation: {0}", GC.GetGeneration(myGCCol));

Console.WriteLine("Total Memory: {0}", GC.GetTotalMemory(false));

GC.Collect(0);

Console.WriteLine("Generation: {0}", GC.GetGeneration(myGCCol));

Console.WriteLine("Total Memory: {0}", GC.GetTotalMemory(false));

Console.WriteLine("Generation: {0}", GC.GetGeneration(myGCCol));

Console.WriteLine("Total Memory: {0}", GC.GetTotalMemory(false));

GC.Collect(2);

Console.WriteLine("Generation: {0}", GC.GetGeneration(myGCCol));

Console.WriteLine("Total Memory: {0}", GC.GetTotalMemory(false));

Console.Read();

}

void MakeSomeGarbage()

{

Version vt;

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

{

vt = new Version();

}

}

}

}

**Вторая часть. Задание**

Написать класс-для работы с данной библиотекой: при создании объекта класса, он открывает файл по переданному конструктору в качестве параметра пути и хранит дескриптор файла всё время жизни объекта.

Созданный класс должен содержать конструктор, принимающий два параметра: путь к файлу и режим открытия файла. Созданный класс должен закрывать связанный файл при вызове метода Dispose, либо финализации. Созданный класс не должен предусматривать повторное открытие файла.

Предусмотреть обработку всех возможных исключений при работе с файлом.

В качестве тестовых файлов преподаватель выдаст заранее подготовленные файлы, содержащие набор отдельных слов, с возможными повторения.

Написать программу для тестирования созданного класса, которая должна выполнять указанные в варианте операции. Программа должна содержать меню, со следующими функциями: открыть файлы, получить количество слов в файлах, а также указанные в варианте действия над файлами.

Если не сказано иного, то изменяться должны те же файлы, что и были предоставлены.

Использовать иные методы работы с файлом кроме библиотеки запрещено!

**Текст программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

namespace task2

{

public class Lib

{

[DllImport("C:/c#/pr3/task2/file32.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern IntPtr open(string path, bool read);

[DllImport("C:/c#/pr3/task2/file32.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern void close(IntPtr file);

[DllImport("C:/c#/pr3/task2/file32.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern bool read(IntPtr file, int num, StringBuilder word);

[DllImport("C:/c#/pr3/task2/file32.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern void write(IntPtr file, string text);

[DllImport("C:/c#/pr3/task2/file32.dll", CallingConvention = CallingConvention.Cdecl)]

public static extern int length(IntPtr file);

private IntPtr file;

private bool disposed = false;

public Lib(string filePath, bool read)

{

try

{

file = open(filePath, read);

if (file == IntPtr.Zero)

{

throw new InvalidOperationException("Не удалось открыть файл");

}

}

catch

{

throw;

}

}

public string Read(int num)

{

if (disposed)

{

throw new ObjectDisposedException("Lib", "Файл закрыт");

}

try

{

StringBuilder word = new StringBuilder(256);

bool result = read(file, num, word);

if (!result)

{

throw new InvalidOperationException("Не удалось прочитать файл");

}

return word.ToString();

}

catch

{

throw;

}

}

public void Write(string text)

{

if (disposed)

{

throw new ObjectDisposedException("Lib", "Файл закрыт");

}

try

{

write(file, text);

}

catch

{

throw;

}

}

public int Length()

{

if (disposed)

{

throw new ObjectDisposedException("Lib", "Файл закрыт");

}

try

{

return length(file);

}

catch

{

throw;

}

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!disposed)

{

if (disposing)

{

// Закрыть файл и освободить управляемые ресурсы (если таковые есть)

close(file);

}

// Освободить неуправляемые ресурсы (если таковые есть) и пометить объект как утилизированный

disposed = true;

}

}

~Lib()

{

Dispose(false);

}

}

internal class Program

{

static void Main(string[] args)

{

string filePath1 = "24.txt", filePath2 = "25.txt";

Lib fileHandler1 = null, fileHandler2 = null;

bool filesOpened = false;

while (true)

{

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

Console.WriteLine("|================================== МЕНЮ ==================================|");

Console.WriteLine("| 1. Открыть файлы |");

Console.WriteLine("| 2. Получить количество слов в файлах |");

Console.WriteLine("| 3. Оставить только уникальные слова в обоих файлах |");

Console.WriteLine("| 4. Оставить в файлах только десять самых часто встречающихся слов |");

Console.WriteLine("| 5. Закрыть файлы |");

Console.WriteLine("| 0. Выйти |");

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

string choice = Console.ReadLine();

switch (choice)

{

case "1":

if (!filesOpened)

{

try

{

fileHandler1 = new Lib("C:/c#/pr3/task2/" + filePath1, true);

fileHandler2 = new Lib("C:/c#/pr3/task2/" + filePath2, true);

filesOpened = true;

Console.WriteLine($"Файлы '{filePath1}' и '{filePath2}' успешно открыты");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при открытии файлов: {ex.Message}");

}

}

else

{

Console.WriteLine("Файлы уже открыты");

}

break;

case "2":

if (filesOpened)

{

int wordCount1 = fileHandler1.Length();

int wordCount2 = fileHandler2.Length();

Console.WriteLine($"Количество слов в файле '{filePath1}': {wordCount1}");

Console.WriteLine($"Количество слов в файле '{filePath2}': {wordCount2}");

}

else

{

Console.WriteLine("Файлы закрыты");

}

break;

case "3":

if (filesOpened)

{

uniqueWords(fileHandler1, fileHandler2, filePath1, filePath2);

Console.WriteLine("Уникальные слова оставлены в каждом файле");

}

else

{

Console.WriteLine("Файлы не открыты");

}

break;

case "4":

if (filesOpened)

{

commonWords(fileHandler1, filePath1);

commonWords(fileHandler2, filePath2);

Console.WriteLine("Оставлено десять самых часто встречающихся слов в каждом файле");

filesOpened = false;

}

else

{

Console.WriteLine("Файлы закрыты");

}

break;

case "5":

if (filesOpened)

{

fileHandler1.Dispose();

fileHandler2.Dispose();

filesOpened = false;

Console.WriteLine("Файлы успешно закрыты");

}

else

{

Console.WriteLine("Файлы закрыты");

}

break;

case "0":

if (filesOpened)

{

fileHandler1.Dispose();

fileHandler2.Dispose();

}

Console.WriteLine("Программа завершена");

return;

default:

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

break;

}

}

}

static void uniqueWords(Lib fileHandler1, Lib fileHandler2, string filepath1, string filepath2)

{

string content1 = "", content2 = "";

for (int i = 1; i <= fileHandler1.Length() && i <= fileHandler2.Length(); i++)

{

content1 += fileHandler1.Read(i) + " ";

content2 += fileHandler2.Read(i) + " ";

}

List<string> words1 = content1.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Distinct().ToList();

List<string> words2 = content2.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();

string result1 = "", result2 = "";

foreach (string word in words1)

{

if (!words2.Contains(word))

{

result1 += word + " ";

}

}

foreach (string word in words2)

{

if (!words1.Contains(word))

{

result2 += word + " ";

}

}

fileHandler1.Dispose();

fileHandler2.Dispose();

fileHandler1 = new Lib("C:/c#/pr3/task2/" + filepath1, false);

fileHandler2 = new Lib("C:/c#/pr3/task2/" + filepath2, false);

fileHandler1.Write(result1.Trim());

fileHandler2.Write(result2.Trim());

}

static void commonWords(Lib fileHandler, string filepath)

{

string content = "";

for (int i = 1; i <= fileHandler.Length(); i++)

{

content += fileHandler.Read(i) + " ";

}

List<string> words = content.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();

Dictionary<string, int> wordCounts = new Dictionary<string, int>();

foreach (string word in words)

{

if (wordCounts.ContainsKey(word))

{

wordCounts[word]++;

}

else

{

wordCounts[word] = 1;

}

}

var topWords = wordCounts.OrderByDescending(pair => pair.Value).Take(10).Select(pair => pair.Key);

string result = string.Join(" ", topWords);

fileHandler.Dispose();

fileHandler = new Lib("C:/c#/pr3/task2/" + filepath, false);

fileHandler.Write(result);

}

}

}

**Результат работы программы**

При запуске программы открывается главное меню. Меню показано на рисунке 2.

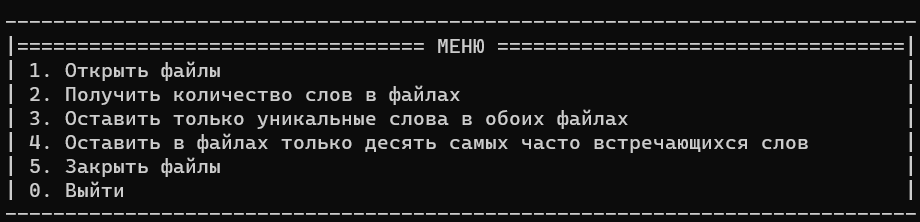


Рисунок 2 – Главное меню

При выборе пункта «1» можно открыть файлы. Выведется сообщение об этом. Если файлы уже открыты, выведется сообщение об этом. Открытие показано на рисунке 3.



Рисунок 3 – Открытие файлов

При выборе пункта «2» можно узнать количество слов в обоих файлах. Расчет показан на рисунке 4.

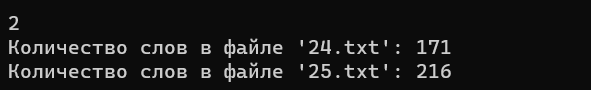


Рисунок 4 – Количество слов в файлах

При выборе пункта «3» в первом файле останутся только те слова, которых нет во втором, а во втором файле останутся только те слова, которых нет в первом. Если файлы закрыты, выведется сообщение об этом. Изменение файлов показаны на рисунках 5, 6 и 7.

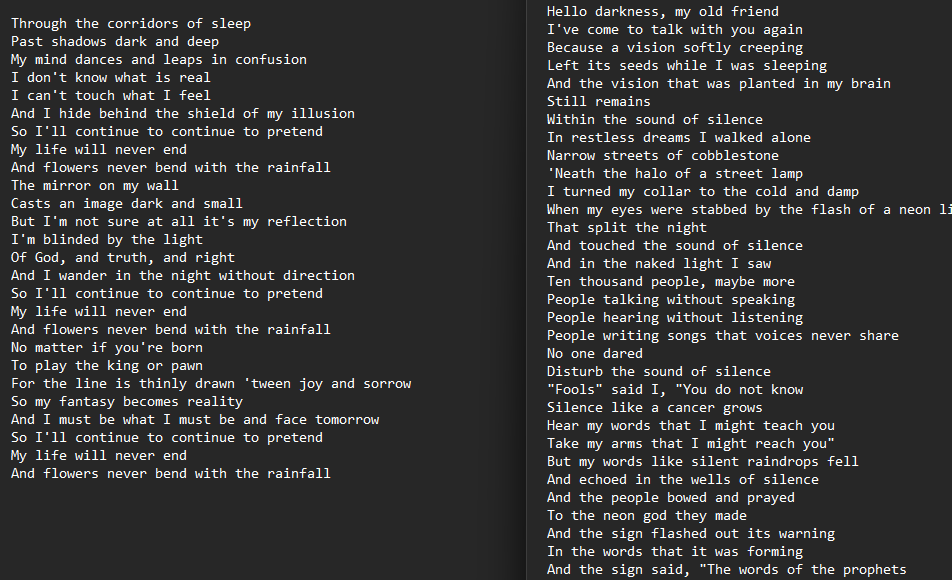


Рисунок 5 – Файлы до изменения



Рисунок 6 – Выбор пункта «3»

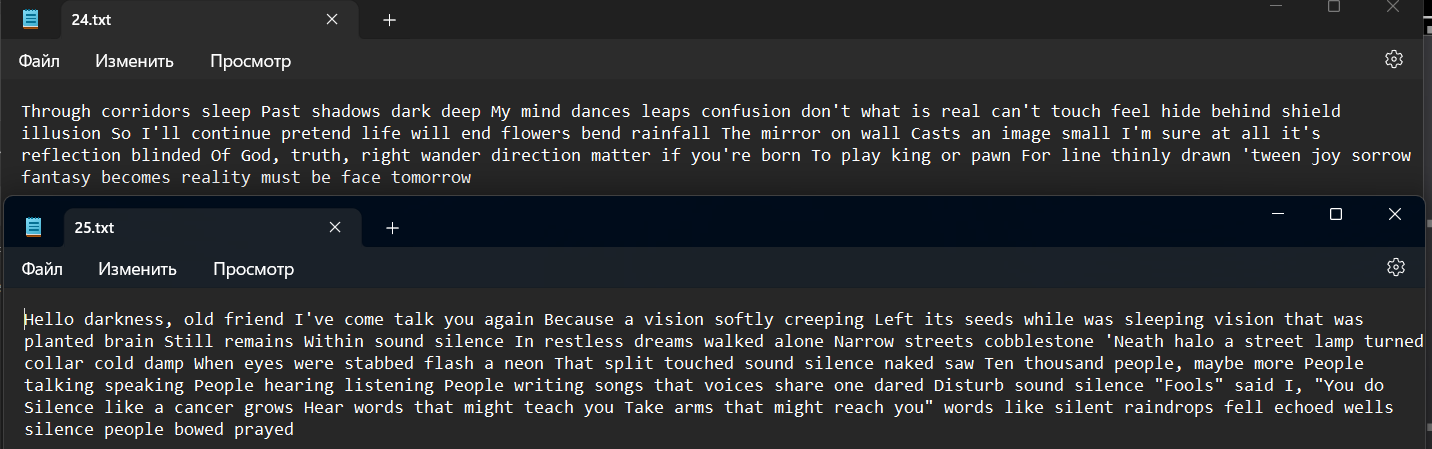


Рисунок 7 – Файлы после изменения

При выборе пункта «4» в каждом из файлов останется по 10 часто встречающихся слов. Если файлы закрыты, выведется сообщение об этом. Результат изменения показан на рисунках 8, 9 и 10.

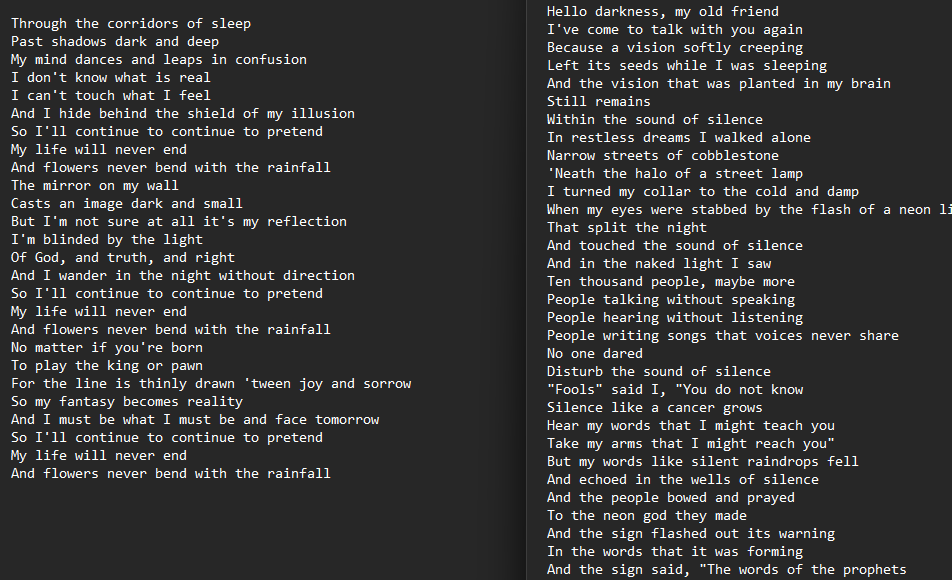


Рисунок 8 – Файлы до изменения

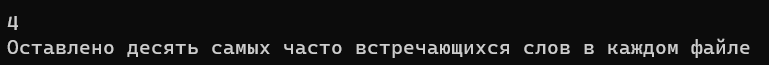


Рисунок 9 – Выбор пункта «4»

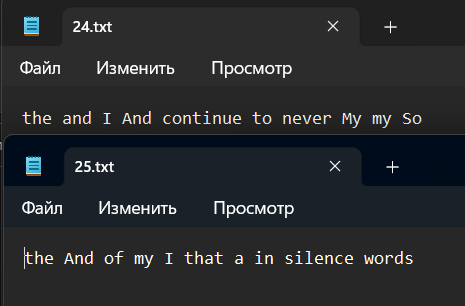


Рисунок 10 – Файлы после изменения

При выборе пункта «5» открытые файлы можно закрыть. Если файл уже закрыт, выведется сообщение об этом. Закрытие файлов показано на рисунке 11.

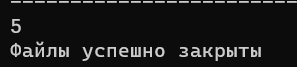


Рисунок 11 – Закрытие файлов

При выборе несуществующего пункта меню выведется сообщение об этом. Ошибка показана на рисунке 12.



Рисунок 12 – Выбор несуществующего пункта меню

При выборе пункта «0» программа завершит свою работу.