**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **“СИНЕРГИЯ”»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и технологии |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | Очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Отчет по практической работе №5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **на тему** |  | Проектирование классов | | |
|  |  | (наименование темы) | | |
|  |  |  | | |
| **по дисциплине** | | |  | Разработка программных модулей |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Дубов А.Н. |  | Не требуется |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | ДКИП-112 прог |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев И.В. |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

381 - Лабораторная работа 4. Простейшие классы.  
395 - Лабораторная работа 8. Классы и операции  
400 - Лабораторная работа 9. Наследование  
405 - Лабораторная работа 10. Структуры

**381 - Лабораторная работа 4. Простейшие классы.**

**Задание**

Описать класс дли работы со строкой, позволяющей хранить только двоичное число и выполнять с пим арифметические операции, Предусмотреть инициализацию с проверкой допустимости значений, В случае недопустимых значений набрасываются исключения, Написать программу, демонстрирующую все разработанные элементы класса.

**Листинг кода:**

using System;

public class BinaryNumber

{

private string binaryValue;

public string Value

{

get { return binaryValue; }

private set

{

if (!IsBinary(value))

{

throw new ArgumentException("Значение должно быть двоичным числом (состоящим только из 0 и 1).");

}

binaryValue = value;

}

}

public BinaryNumber(string binary)

{

Value = binary;

}

private bool IsBinary(string value)

{

foreach (char c in value)

{

if (c != '0' && c != '1')

{

return false;

}

}

return true;

}

public BinaryNumber Add(BinaryNumber other)

{

int decimalSum = Convert.ToInt32(this.Value, 2) + Convert.ToInt32(other.Value, 2);

return new BinaryNumber(Convert.ToString(decimalSum, 2));

}

public BinaryNumber Subtract(BinaryNumber other)

{

int decimalDiff = Convert.ToInt32(this.Value, 2) - Convert.ToInt32(other.Value, 2);

if (decimalDiff < 0)

{

throw new InvalidOperationException("Результат вычитания отрицательный, что недопустимо для двоичных чисел.");

}

return new BinaryNumber(Convert.ToString(decimalDiff, 2));

}

public override string ToString()

{

return binaryValue;

}

}

class Program

{

static void Main(string[] args)

{

try

{

BinaryNumber num1 = new BinaryNumber("1101");

BinaryNumber num2 = new BinaryNumber("1011");

Console.WriteLine($"Первое двоичное число: {num1}");

Console.WriteLine($"Второе двоичное число: {num2}");

BinaryNumber sum = num1.Add(num2);

Console.WriteLine($"Сумма: {sum}");

BinaryNumber difference = num1.Subtract(num2);

Console.WriteLine($"Разность: {difference}");

BinaryNumber invalidNum = new BinaryNumber("102");

}

catch (ArgumentException ex)

{

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

}

catch (InvalidOperationException ex)

{

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

}

catch (Exception ex)

{

Console.WriteLine($"Неизвестная ошибка: {ex.Message}");

}

}

}

**395 - Лабораторная работа 8. Классы и операции**

Описать класс «предметный указатель». Каждый компонент указателя содержит слово и номера страниц, на которых это слово встречается. Количество номеров страниц, относящихся к одному слову, от одного до десяти. Предусмотреть возможность формирования указателя с клавиатуры и из файла, вывода указателя, вывода номеров страниц для заданного слова, удаления элемента из указателя. Написать программу, демонстрирующую все разработанные элементы класса.

Листинг кода:

using System;

using System.Collections.Generic;

using System.IO;

public class IndexEntry

{

public string Word { get; }

public HashSet<int> PageNumbers { get; } = new HashSet<int>();

public IndexEntry(string word)

{

Word = word;

}

public void AddPageNumber(int pageNumber)

{

if (PageNumbers.Count < 10)

PageNumbers.Add(pageNumber);

}

public override string ToString() => $"{Word}: {string.Join(", ", PageNumbers)}";

}

public class Index

{

private readonly Dictionary<string, IndexEntry> entries = new();

public void AddEntry(string word, int pageNumber)

{

if (!entries.ContainsKey(word))

entries[word] = new IndexEntry(word);

entries[word].AddPageNumber(pageNumber);

}

public void RemoveEntry(string word) => entries.Remove(word);

public void PrintIndex()

{

foreach (var entry in entries.Values)

Console.WriteLine(entry);

}

public void PrintPagesForWord(string word)

{

if (entries.TryGetValue(word, out var entry))

Console.WriteLine(entry);

else

Console.WriteLine($"Слово '{word}' не найдено.");

}

public void LoadFromFile(string filePath)

{

foreach (var line in File.ReadLines(filePath))

{

var parts = line.Split(':');

var word = parts[0].Trim();

var pages = parts[1].Split(',', StringSplitOptions.RemoveEmptyEntries);

foreach (var page in pages)

if (int.TryParse(page.Trim(), out var pageNumber))

AddEntry(word, pageNumber);

}

}

}

class Program

{

static void Main()

{

var index = new Index();

index.AddEntry("Программирование", 1);

index.AddEntry("Программирование", 2);

index.AddEntry("Алгоритмы", 3);

Console.WriteLine("Индексы:");

index.PrintIndex();

Console.WriteLine("\nСтраницы для 'Программирование':");

index.PrintPagesForWord("Программирование");

index.RemoveEntry("Алгоритмы");

Console.WriteLine("\nИндексы после удаления 'Алгоритмы':");

index.PrintIndex();

// Загрузка из файла

// string filePath = "index.txt"; // Укажите путь к файлу

// index.LoadFromFile(filePath);

// Console.WriteLine("\nИндексы после загрузки из файла:");

// index.PrintIndex();

}

}

**400 - Лабораторная работа 9. Наследование**

**Листинг кода:**

using System;

public class Mod2Element

{

public int XOR(int a, int b)

{

return a ^ b;

}

}

public class DFlipFlop

{

private int state;

public void Clock(int d)

{

state = d;

}

public int Output()

{

return state;

}

}

public class Register

{

private int[] bits;

public Register(int size)

{

bits = new int[size];

}

public void Load(int[] data)

{

if (data.Length != bits.Length)

throw new ArgumentException("Данные должны соответствовать размеру регистра.");

Array.Copy(data, bits, data.Length);

}

public void Print()

{

Console.Write("Регистровое состояние: ");

foreach (var bit in bits)

Console.Write(bit);

Console.WriteLine();

}

}

class Program

{

static void Main()

{

Console.WriteLine("Демонстрация комбинационного элемента МОД2 (XOR):");

var mod2 = new Mod2Element();

int a = 1;

int b = 0;

Console.WriteLine($"XOR({a}, {b}) = {mod2.XOR(a, b)}");

Console.WriteLine("\nДемонстрация триггера D:");

var dFlipFlop = new DFlipFlop();

dFlipFlop.Clock(1);

Console.WriteLine($"Состояние триггера после установки D=1: {dFlipFlop.Output()}");

dFlipFlop.Clock(0);

Console.WriteLine($"Состояние триггера после установки D=0: {dFlipFlop.Output()}");

Console.WriteLine("\nДемонстрация 10-битного регистра:");

var register = new Register(10);

int[] dataToLoad = { 1, 0, 1, 0, 1, 0, 1, 0, 1, 0 };

register.Load(dataToLoad);

register.Print();

int[] newDataToLoad = { 0, 1, 0, 1, 0, 1, 0, 1, 0, 1 };

register.Load(newDataToLoad);

register.Print();

}

}

**405 - Лабораторная работа 10. Структуры**

Описать структуру с именем NOTE, содержащую следующие поля:

фамилия, имя; номер телефона; дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE (записи должны быть размещены по алфавиту); вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет, вывести соответствующее сообщение).

**Листинг кода:**

using System;

struct NOTE

{

public string LastName;

public string FirstName;

public string PhoneNumber;

public int[] BirthDate;

public NOTE(string lastName, string firstName, string phoneNumber, int day, int month, int year)

{

LastName = lastName;

FirstName = firstName;

PhoneNumber = phoneNumber;

BirthDate = new int[] { day, month, year };

}

}

class Program

{

static void Main()

{

const int size = 8;

NOTE[] notes = new NOTE[size];

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

{

Console.WriteLine($"Введите данные для записи {i + 1}:");

Console.Write("Фамилия: ");

string lastName = Console.ReadLine();

Console.Write("Имя: ");

string firstName = Console.ReadLine();

Console.Write("Номер телефона: ");

string phoneNumber = Console.ReadLine();

Console.Write("День рождения: ");

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

Console.Write("Месяц рождения: ");

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

Console.Write("Год рождения: ");

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

notes[i] = new NOTE(lastName, firstName, phoneNumber, day, month, year);

}

Array.Sort(notes, (x, y) => x.LastName.CompareTo(y.LastName));

Console.Write("Введите номер месяца для поиска (1-12): ");

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

bool found = false;

Console.WriteLine($"\nЛюди с днем рождения в месяце {searchMonth}:");

foreach (var note in notes)

{

if (note.BirthDate[1] == searchMonth) // Проверяем месяц

{

Console.WriteLine($"Фамилия: {note.LastName}, Имя: {note.FirstName}, Номер телефона: {note.PhoneNumber}, Дата рождения: {note.BirthDate[0]}.{note.BirthDate[1]}.{note.BirthDate[2]}");

found = true;

}

}

if (!found)

{

Console.WriteLine("Нет людей с днем рождения в этом месяце.");

}

}

}