**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Институт прикладных информационных технологий и коммуникаций

Направление «Информационные системы и технологии»

Кафедра «Прикладные информационные технологии»

**КОНТРОЛЬНАЯ РАБОТА**

по дисциплине «Технология программирования»

Выполнил студент группы

б-1ИФСТипу12

заочной формы обучения

Волков Андрей Алексеевич

Номер зачетной книжки 233835

Саратов 2024

**Содержание**

[**Задача 1. В массиве целых найти наиболее длинную последовательность чисел одного знака.** 4](#_Toc161595730)

[**Задача 2. Дано целое число. Получить то же число, но записанное цифрами в обратном порядке.** 6](#_Toc161595731)

[**Задача 3. Дан целый массив. Определить число инверсий. Инверсия – это когда большее число стоит левее меньшего.** 7](#_Toc161595732)

[**Задача 4. Два прямоугольника на плоскости заданы координатами левого верхнего и правого нижнего углов. Определить площадь пересечения прямоугольников.** 8](#_Toc161595733)

[**Задача 5. Определить, есть ли в p-ичной записи числа n две одинаковые цифры (p <= 9).** 10](#_Toc161595734)

[**Задача 6. Палиндромом называется число, которое читается одинаково слева направо и справа налево (121; 676; 99). Для данного n определить, является ли число палиндромом в системе счисления с основанием p.** 12](#_Toc161595735)

[**Задача 7. Переставить символы строки так, чтобы в начале строки стояли более часто встречающиеся символы: “барракуда” => “aaaррбкуд”** 14](#_Toc161595736)

[**Задача 8. В текстовой строке вида: “(a [b c {(x q b)}] d r)” проверить правильность скобочной структуры (баланс скобок трех видов и правильность их расстановки)** 16](#_Toc161595737)

[**Задача 9. Дана текстовая строка вида: “XXXZZRRVBVVVVVWW…” (не содержащая символа @). Упаковать ее, заменив группы одинаковых символов: VVVVV => @5V** 18](#_Toc161595738)

[**Задача 10. Из упакованной строки (задача 12) получить исходную строку.** 21](#_Toc161595739)

[**Задача 11. Дан массив X типа int и его размер n. Найти величину максимума массива без использования каких-либо операторов цикла и goto** 23](#_Toc161595740)

[**Задача 12. Дан массив X типа int, его размер n и указатель на функцию одного целого аргумента pf (int). Реализовать функцию map, которая возвращает указатель на массив Y, такой, что Y[i]=pf(X[i]). Операторы цикла не использовать.** 24](#_Toc161595741)

[**Задача 13. Написать рекурсивную функцию вычисления синуса разложением в ряд: sin(x)=x-(x^3/3!)+(x^5/5!)-(x^7/7!)+... с задаваемой точностью eps.** 26](#_Toc161595742)

[**Задача 14. Строка содержит формулу в постфиксной записи вида:** 27](#_Toc161595743)

[**5 6 + 3 \* 22 – написать программу вычисления значения формулы. (В данном случае рез=11)** 27](#_Toc161595744)

[**Задача 15. Дан массив целых. Переместить все отрицательные элементы в начало массива. Внутреннее взаиморасположение отрицательных и положительных чисел не менять. Дополнительный массив не использовать. {1,2,-5,-3,8,-10,7,5} =>{-5,-3,-10,1,2,8,7,5}** 30](#_Toc161595745)

[**Задача 16. На плоскости задан треугольник координатами вершин (x1,y1), (x2,y2), (x3,y3) и произвольная точка (x,y). Составить логическую функцию, проверяющую, лежит ли точка внутри треугольника.** 32](#_Toc161595746)

[**Задача 17. Даны три целых массива, отсортированных по возрастанию. Найти первое число, которые встречается во всех трех массивах.** 35](#_Toc161595747)

[**Задача 18. Дан файл, содержащий текст на русском языке. Отформатировать этот текст “по ширине” w вставкой нужного количества пробелов между словами** 37](#_Toc161595748)

[**Задача 19. Дан целый массив A из n элементов. Построить множество всех подмножеств элементов массива.** 40](#_Toc161595749)

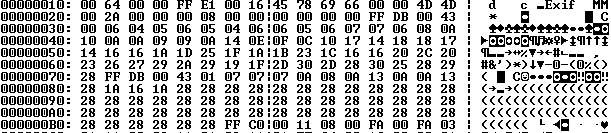
[**для A={1,2} => {{1},{2},{1,2}};** 40](#_Toc161595750)

[**для A={1,2,3} => {{1},{2},{3},{1,2},{1,3}, {2,3},{1,2,3}}** 40](#_Toc161595751)

[**и т.п.** 40](#_Toc161595752)

[**Задача 20. Найти в массиве целых три минимальных числа** 43](#_Toc161595753)

[**Задача 21. Написать программу, которая принимает имя файла, смещение, длину блока и выводит на экран шестнадцатеричный дамп этого блока:** 45](#_Toc161595754)

[**** 45](#_Toc161595755)

[**Задача 22. Дан текстовый файл, содержащий упорядоченные по возрастанию числа одной разрядности и число N. Проверить, присутствует ли число N в файле.** 47](#_Toc161595756)

[**Задача 23. Сколько существует натуральных чисел n, меньших 1000, для которых (2^n)-n делится на 7?** 49](#_Toc161595757)

[**Задача 24. Для данного n выяснить, является ли это n членом последовательности Фибоначчи (1,1,2,3,5,8,13...)** 50](#_Toc161595758)

[**Задача 25. Многочлен с целыми коэффициентами:** 52](#_Toc161595759)

[**P=a0+a1\*x+a2\*x2+…+an\*xn** 52](#_Toc161595760)

[**Задается массивом коэффициентов** 52](#_Toc161595761)

[**P={a0 , a1 , a2 ,…, an }** 52](#_Toc161595762)

[**Разработать программы сложения, умножения и печати многочленов.** 52](#_Toc161595763)

[**Задача 26. Дано выражение вида: “nn+mm-kk+…zz” Операции только “+” и “-” , порядок произволен. nn, mm, kk …zz – целые числа различной разрядности. Вычислить значение выражения.** 55](#_Toc161595764)

## **Задача 1. В массиве целых найти наиболее длинную последовательность чисел одного знака.**

using System;

class Program

{

static void Main()

{

int[] array = { 1, 2, 3, -4, -5, -6, -7, 8, 9, -10, 11, 12, -13, -14, -15 };

int maxLength = 0;

int maxStartIndex = 0;

int currentLength = 0;

int currentStartIndex = 0;

for (int i = 0; i < array.Length; i++)

{

if (i == 0 || (array[i] > 0) == (array[i - 1] > 0))

{

currentLength++;

}

else

{

if (currentLength > maxLength)

{

maxLength = currentLength;

maxStartIndex = currentStartIndex;

}

currentLength = 1;

currentStartIndex = i;

}

}

// Проверяем последнюю последовательность

if (currentLength > maxLength)

{

maxLength = currentLength;

maxStartIndex = currentStartIndex;

}

Console.WriteLine("Наибольшая последовательность одного знака:");

for (int i = maxStartIndex; i < maxStartIndex + maxLength; i++)

{

Console.Write(array[i] + " ");

}

Console.WriteLine();

}

}

## **Задача 2. Дано целое число. Получить то же число, но записанное цифрами в обратном порядке.**

using System;

class Program

{

static void Main()

{

int number = 12345; // Заданное число

// Разбиваем число на цифры и создаем новое число в обратном порядке

int reversedNumber = 0;

while (number > 0)

{

int digit = number % 10; // Получаем последнюю цифру

reversedNumber = reversedNumber \* 10 + digit; // Добавляем цифру к новому числу

number /= 10; // Удаляем последнюю цифру из исходного числа

}

Console.WriteLine("Исходное число: " + number);

Console.WriteLine("Число, записанное цифрами в обратном порядке: " + reversedNumber);

}

}

## **Задача 3. Дан целый массив. Определить число инверсий. Инверсия – это когда большее число стоит левее меньшего.**

using System;

class Program

{

static int CountInversions(int[] arr)

{

int n = arr.Length;

int invCount = 0;

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (arr[i] > arr[j])

{

// Если большее число стоит левее меньшего

invCount++;

}

}

}

return invCount;

}

static void Main()

{

int[] arr = { 1, 5, 4, 3, 2 };

int inversions = CountInversions(arr);

Console.WriteLine("Число инверсий в массиве: " + inversions);

}

}

## **Задача 4. Два прямоугольника на плоскости заданы координатами левого верхнего и правого нижнего углов. Определить площадь пересечения прямоугольников.**

using System;

//Вспомогательный класс прямоугольник

class Rectangle

{

public int TopLeftX { get; set; }

public int TopLeftY { get; set; }

public int BottomRightX { get; set; }

public int BottomRightY { get; set; }

public Rectangle(int topLeftX, int topLeftY, int bottomRightX, int bottomRightY)

{

TopLeftX = topLeftX;

TopLeftY = topLeftY;

BottomRightX = bottomRightX;

BottomRightY = bottomRightY;

}

//Метод для подсчёта площади пересения

public int CalculateIntersectionArea(Rectangle other)

{

int xOverlap = Math.Max(0, Math.Min(this.BottomRightX, other.BottomRightX) - Math.Max(this.TopLeftX, other.TopLeftX));

int yOverlap = Math.Max(0, Math.Min(this.BottomRightY, other.BottomRightY) - Math.Max(this.TopLeftY, other.TopLeftY));

return xOverlap \* yOverlap;

}

}

class Program

{

static void Main()

{

Rectangle rect1 = new Rectangle(0, 0, 3, 3);

Rectangle rect2 = new Rectangle(1, 1, 4, 5);

int intersectionArea = rect1.CalculateIntersectionArea(rect2);

Console.WriteLine("Площадь пересечения прямоугольников: " + intersectionArea);

}

}

## **Задача 5. Определить, есть ли в p-ичной записи числа n две одинаковые цифры (p <= 9).**

using System;

class Program

{

static bool HasDuplicateDigits(int n, int p)

{

if (p > 9)

{

throw new ArgumentException("Основание системы счисления должно быть меньше или равно 9");

}

if (n >= Math.Pow(p, 2))

{

throw new ArgumentException("Число n не может быть представлено в системе счисления с основанием p");

}

bool[] seen = new bool[p];

while (n > 0)

{

int digit = n % p;

if (seen[digit])

{

return true; // Если мы уже видели эту цифру, значит есть дубликат

}

seen[digit] = true;

n /= p;

}

return false; // Дубликаты не найдены

}

static void Main()

{

int n = 321;

int p = 5;

bool hasDuplicates = HasDuplicateDigits(n, p);

if (hasDuplicates)

{

Console.WriteLine($"В {p}-ичной записи числа {n} есть две одинаковые цифры.");

}

else

{

Console.WriteLine($"В {p}-ичной записи числа {n} нет двух одинаковых цифр.");

}

}

}

## **Задача 6. Палиндромом называется число, которое читается одинаково слева направо и справа налево (121; 676; 99). Для данного n определить, является ли число палиндромом в системе счисления с основанием p.**

using System;

class Program

{

static bool IsPalindrome(int n, int p)

{

int reversedNumber = 0;

int originalNumber = n;

//Проходим по цифрам числа и делаем его "обратным"

while (n > 0)

{

int remainder = n % p;

reversedNumber = reversedNumber \* p + remainder;

n /= p;

}

return originalNumber == reversedNumber;

}

static void Main()

{

int n = 1221;

int p = 5;

if (IsPalindrome(n, p))

{

Console.WriteLine($"Число {n} является палиндромом в системе счисления с основанием {p}.");

}

else

{

Console.WriteLine($"Число {n} не является палиндромом в системе счисления с основанием {p}.");

}

}

}

## **Задача 7. Переставить символы строки так, чтобы в начале строки стояли более часто встречающиеся символы: “барракуда” => “aaaррбкуд”**

using System;

class Program

{

static string ReorderString(string input)

{

// Создаем массив для подсчета количества символов в строке (по ASCII таблице)

int[] charCount = new int[256];

// Подсчитываем количество встречающихся символов в строке

foreach (char c in input)

{

charCount[c]++;

}

string reorderedString = "";

// Перебираем символы по частоте их встречаемости (сначала наиболее частые)

for (int count = input.Length; count > 0; count--)

{

for (int c = 0; c < charCount.Length; c++)

{

// Если символ встречается count раз, добавляем его в результирующую строку count раз

if (charCount[c] == count)

{

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

{

reorderedString += (char)c;

}

}

}

}

return reorderedString;

}

static void Main()

{

string input = "барракуда";

string reorderedString = ReorderString(input);

Console.WriteLine("Переупорядоченная строка: " + reorderedString);

}

}

## **Задача 8. В текстовой строке вида: “(a [b c {(x q b)}] d r)” проверить правильность скобочной структуры (баланс скобок трех видов и правильность их расстановки)**

using System;

class Program

{

static bool CheckBrackets(string input)

{

// Создаем стек для отслеживания открытых скобок

Stack<char> stack = new();

foreach (char c in input)

{

// Если текущий символ - открывающая скобка, помещаем ее в стек

if (c == '(' || c == '[' || c == '{')

{

stack.Push(c);

}

// Если текущий символ - закрывающая скобка

else if (c == ')' || c == ']' || c == '}')

{

// Если стек пуст, то не найдена соответствующая открывающая скобка

if (stack.Count == 0)

return false;

// Проверяем соответствие закрывающей скобки последней открытой в стеке

char lastOpenBracket = stack.Pop();

if ((c == ')' && lastOpenBracket != '(') ||

(c == ']' && lastOpenBracket != '[') ||

(c == '}' && lastOpenBracket != '{'))

{

return false;

}

}

}

// Если стек пуст в конце проверки, то все скобки сбалансированы

return stack.Count == 0;

}

static void Main()

{

string input = "(a [b c {(x q b)}] d r)";

if (CheckBrackets(input))

{

Console.WriteLine("Скобочная структура верна.");

}

else

{

Console.WriteLine("Скобочная структура неверна.");

}

}

}

## **Задача 9. Дана текстовая строка вида: “XXXZZRRVBVVVVVWW…” (не содержащая символа @). Упаковать ее, заменив группы одинаковых символов: VVVVV => @5V**

using System;

using System.Text;

class Program

{

static string PackString(string input)

{

StringBuilder packedString = new();

// Инициализируем переменные для хранения текущего символа и его количества повторений

char currentChar = input[0];

int count = 1;

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

{

// Если текущий символ совпадает с предыдущим, увеличиваем счетчик повторений

if (input[i] == currentChar)

{

count++;

}

else

{

// Если символ отличается от предыдущего, добавляем упакованный символ в строку

if (count > 1)

{

packedString.Append($"@{count}{currentChar}");

}

else

{

packedString.Append(currentChar);

}

// Обновляем текущий символ и его количество повторений

currentChar = input[i];

count = 1;

}

}

// Добавляем последний символ в упакованную строку

if (count > 1)

{

packedString.Append($"@{count}{currentChar}");

}

else

{

packedString.Append(currentChar);

}

return packedString.ToString();

}

static void Main()

{

string input = "XXXZZRRVBVVVVVWW";

Console.WriteLine("Упакованная строка: " + PackString(input));

}

}

## **Задача 10. Из упакованной строки (задача 12) получить исходную строку.**

using System;

using System.Text;

class Program

{

static string UnpackString(string input)

{

StringBuilder unpackedString = new StringBuilder();

for (int i = 0; i < input.Length; i++)

{

char currentChar = input[i];

// Если текущий символ - '@', значит следующие два символа содержат информацию о количестве повторений

if (currentChar == '@')

{

// Получаем количество повторений

int count = int.Parse(input[i + 1].ToString());

// Получаем символ, который нужно повторить

char repeatingChar = input[i + 2];

// Добавляем повторяющийся символ нужное количество раз в распакованную строку

for (int j = 0; j < count; j++)

{

unpackedString.Append(repeatingChar);

}

// Увеличиваем индекс, чтобы пропустить количество и повторяющийся символ

i += 2;

}

else

{

// Если текущий символ не '@', добавляем его в распакованную строку

unpackedString.Append(currentChar);

}

}

return unpackedString.ToString();

}

static void Main()

{

string input = "@3XZ@2R@2VB@5W";

string unpackedString = UnpackString(input);

Console.WriteLine("Распакованная строка: " + unpackedString);

}

}

## **Задача 11. Дан массив X типа int и его размер n. Найти величину максимума массива без использования каких-либо операторов цикла и goto**

using System;

class Program

{

static int FindMax(int[] array, int start, int end)

{

// Если массив состоит только из одного элемента, возвращаем его значение

if (start == end)

return array[start];

int current = array[start];

int next = FindMax(array, start + 1, end);

// Возвращаем большее из двух значений

return Math.Max(current, next);

}

static void Main()

{

int[] X = { 4, 7, 2, 9, 5 };

int n = X.Length;

// Вызываем рекурсивную функцию для перебора массива

int max = FindMax(X, 0, n - 1);

Console.WriteLine($"Максимальный элемент массива: {max}");

}

}

## **Задача 12. Дан массив X типа int, его размер n и указатель на функцию одного целого аргумента pf (int). Реализовать функцию map, которая возвращает указатель на массив Y, такой, что Y[i]=pf(X[i]). Операторы цикла не использовать.**

*Реализовано на C++, т.к. в задание работа с указателями*

#include <iostream>

// Определение типа функции pf

typedef int (\*FunctionPointer)(int);

// Функция map принимает массив X, его размер n и указатель на функцию pf

int\* map(int X[], int n, FunctionPointer pf) {

int\* Y = new int[n]; // Создаем массив Y

for (int i = 0; i < n; ++i) {

Y[i] = pf(X[i]);

}

return Y;

}

// Пример функции, которая может быть передана в качестве аргумента pf

int square(int x) {

return x \* x;

}

int main() {

int X[] = { 1, 2, 3, 4, 5 };

int n = sizeof(X) / sizeof(X[0]);

// Вызываем функцию map, передавая массив X, его размер и указатель на функцию square

int\* Y = map(X, n, square);

std::cout << "Mapped array Y: ";

for (int i = 0; i < n; ++i) {

std::cout << Y[i] << " ";

}

std::cout << std::endl;

// Освобождаем память, выделенную для массива Y

delete[] Y;

return 0;

}

## **Задача 13. Написать рекурсивную функцию вычисления синуса разложением в ряд: sin(x)=x-(x^3/3!)+(x^5/5!)-(x^7/7!)+... с задаваемой точностью eps.**

using System;

class Program

{

static double Sin(double x, double eps, int n = 0, int sign = 1)

{

// Вычисляем текущее слагаемое

double term = sign \* Math.Pow(x, 2 \* n + 1) / Factorial(2 \* n + 1);

// Если точность достигнута (слагаемое меньше eps), завершаем рекурсию

if (Math.Abs(term) < eps)

return term;

// Рекурсивно вычисляем оставшуюся часть

return term + Sin(x, eps, n + 1, -sign);

}

static double Factorial(int n)

{

if (n == 0)

return 1;

return n \* Factorial(n - 1);

}

static void Main()

{

double x = 0.5;

double eps = 0.0001;

double sinX = Sin(x, eps);

Console.WriteLine($"sin({x}) = {sinX}");

}

}

## **Задача 14. Строка содержит формулу в постфиксной записи вида:**

## **5 6 + 3 \* 22 – написать программу вычисления значения формулы. (В данном случае рез=11)**

using System;

class Program

{

static double EvaluatePostfixExpression(string expression)

{

Stack<double> stack = new();

// Разбиваем строку на отдельные элементы

string[] elements = expression.Split(' ');

foreach (string element in elements)

{

// Если элемент является числом, помещаем его в стек

if (double.TryParse(element, out double operand))

{

stack.Push(operand);

}

else

{

// Если элемент - оператор, выполняем операцию над операндами из стека

double operand2 = stack.Pop();

double operand1 = stack.Pop();

switch (element)

{

case "+":

stack.Push(operand1 + operand2);

break;

case "-":

stack.Push(operand1 - operand2);

break;

case "\*":

stack.Push(operand1 \* operand2);

break;

case "/":

stack.Push(operand1 / operand2);

break;

default:

throw new ArgumentException("Неверный оператор: " + element);

}

}

}

return stack.Pop();

}

static void Main()

{

string expression = "5 6 + 3 \* 22 -";

Console.WriteLine($"Результат вычисления формулы: {EvaluatePostfixExpression(expression)}");

}

}

## **Задача 15. Дан массив целых. Переместить все отрицательные элементы в начало массива. Внутреннее взаиморасположение отрицательных и положительных чисел не менять. Дополнительный массив не использовать. {1,2,-5,-3,8,-10,7,5} =>{-5,-3,-10,1,2,8,7,5}**

using System;

class Program

{

static void MoveNegativeToFront(int[] array)

{

int negativeIndex = 0; // Индекс, указывающий на следующую позицию, куда нужно поместить отрицательное число

for (int i = 0; i < array.Length; i++)

{

if (array[i] < 0)

{

// Сдвигаем отрицательное число на позицию negativeIndex

int temp = array[i];

for (int j = i; j > negativeIndex; j--)

{

array[j] = array[j - 1];

}

array[negativeIndex] = temp;

// Увеличиваем negativeIndex для указания на следующую позицию

negativeIndex++;

}

}

}

static void PrintArray(int[] array)

{

foreach (int num in array)

{

Console.Write(num + " ");

}

Console.WriteLine();

}

static void Main()

{

int[] array = { 1, 2, -5, -3, 8, -10, 7, 5 };

Console.WriteLine("Исходный массив:");

PrintArray(array);

MoveNegativeToFront(array);

Console.WriteLine("\nМассив после перемещения отрицательных элементов в начало:");

PrintArray(array);

}

}

## **Задача 16. На плоскости задан треугольник координатами вершин (x1,y1), (x2,y2), (x3,y3) и произвольная точка (x,y). Составить логическую функцию, проверяющую, лежит ли точка внутри треугольника.**

using System;

class Point(double x, double y)

{

public double X { get; } = x;

public double Y { get; } = y;

}

class Triangle(Point p1, Point p2, Point p3)

{

public Point P1 { get; } = p1;

public Point P2 { get; } = p2;

public Point P3 { get; } = p3;

// Метод для вычисления площади треугольника, образованного тремя точками

private static double TriangleArea(Point p1, Point p2, Point p3)

{

return Math.Abs((p2.X - p1.X) \* (p3.Y - p1.Y) - (p3.X - p1.X) \* (p2.Y - p1.Y));

}

// Метод для проверки, лежит ли точка внутри треугольника

public bool IsInside(Point point)

{

double triangleArea = TriangleArea(P1, P2, P3);

// Вычисляем площади трёх подтреугольников

double subTriangle1 = TriangleArea(P1, P2, point);

double subTriangle2 = TriangleArea(P2, P3, point);

double subTriangle3 = TriangleArea(P3, P1, point);

// Суммируем площади подтреугольников

double sumSubTriangles = subTriangle1 + subTriangle2 + subTriangle3;

// Если сумма площадей подтреугольников равна площади исходного треугольника, точка внутри треугольника

return Math.Abs(triangleArea - sumSubTriangles) == 0;

}

}

class Program

{

static void Main()

{

// Создаем объект треугольника

Triangle triangle = new(new(1, 1), new(5, 1), new(3, 5));

Point point = new(4, 1);

bool result = triangle.IsInside(point);

if (result)

Console.WriteLine($"Точка ({point.X}, {point.Y}) лежит внутри треугольника ({triangle.P1.X}, {triangle.P1.Y}) ({triangle.P2.X}, {triangle.P2.Y}) ({triangle.P3.X}, {triangle.P3.Y}).");

else

Console.WriteLine($"Точка ({point.X}, {point.Y}) не лежит внутри треугольника ({triangle.P1.X}, {triangle.P1.Y}) ({triangle.P2.X}, {triangle.P2.Y}) ({triangle.P3.X}, {triangle.P3.Y}).");

}

}

## **Задача 17. Даны три целых массива, отсортированных по возрастанию. Найти первое число, которые встречается во всех трех массивах.**

using System;

class Program

{

static bool FindCommonElement(int[] arr1, int[] arr2, int[] arr3, ref int res)

{

int i = 0, j = 0, k = 0;

while (i < arr1.Length && j < arr2.Length && k < arr3.Length)

{

// Если текущие элементы всех трех массивов равны, мы нашли общий элемент

if (arr1[i] == arr2[j] && arr2[j] == arr3[k])

{

res = arr1[i];

return true;

}

// Перемещаем указатели вперед

if (arr1[i] < arr2[j] || arr1[i] < arr3[k])

i++;

else if (arr2[j] < arr1[i] || arr2[j] < arr3[k])

j++;

else

k++;

}

// Если мы дошли до конца какого-либо массива, значит, общих элементов нет

return false;

}

static void Main()

{

int[] arr1 = {1, 5, 10, 15, 17, 20 };

int[] arr2 = {6, 7, 20, 80, 100 };

int[] arr3 = {3, 4, 15, 20, 30, 70, 80, 120 };

int res = 0;

if (FindCommonElement(arr1, arr2, arr3, ref res))

Console.WriteLine("Первое общее число: " + res);

else

Console.WriteLine("Общих чисел нет");

}

}

## **Задача 18. Дан файл, содержащий текст на русском языке. Отформатировать этот текст “по ширине” w вставкой нужного количества пробелов между словами**

using System;

using System.Text;

class Program

{

internal static readonly char[] separator = [' ', '\n', '\r', '\t'];

static string FormatTextWidth(string text, int width)

{

string[] words = text.Split(separator, StringSplitOptions.RemoveEmptyEntries);

// Формирование строк с необходимой длиной

StringBuilder result = new();

StringBuilder currentLine = new();

foreach (string word in words)

{

if (currentLine.Length + word.Length + 1 <= width)

{

// Добавляем слово в текущую строку с одним пробелом после

if (currentLine.Length > 0)

currentLine.Append(' '); // Добавляем пробел между словами

currentLine.Append(word);

}

else

{

currentLine = FormatLineSpace(currentLine, width);

result.AppendLine(currentLine.ToString().Trim());

currentLine.Clear().Append(word);

}

}

currentLine = FormatLineSpace(currentLine, width);

result.AppendLine(currentLine.ToString().Trim());

return result.ToString();

}

static StringBuilder FormatLineSpace(StringBuilder currentLine, int width)

{

// Добавление пробелов в текущую строку, чтобы ее длина была равна width

while (currentLine.Length < width)

{

// Если длина строки уже достигла нужной, прекращаем добавление

if (currentLine.Length == width)

break;

// Добавляем по одному пробелу после каждого слова

for (int i = 0; i < currentLine.Length; i++)

{

if (currentLine[i] == ' ' && currentLine.Length < width)

{

currentLine.Insert(i + 1, ' ');

i++;

}

}

}

return currentLine;

}

static void Main()

{

string inputFile = "input.txt"; // Имя файла с исходным текстом

string outputFile = "output.txt"; // Имя файла для сохранения отформатированного текста

int width = 50; // Ширина текста (нельзя ставить слишком маленькую)

string text = File.ReadAllText(inputFile);

// Отформатировать текст по ширине

string formattedText = FormatTextWidth(text, width);

// Записываем отформатированный текст в файл

File.WriteAllText(outputFile, formattedText);

Console.WriteLine("Текст успешно отформатирован и сохранен в файле output.txt");

}

}

## **Задача 19. Дан целый массив A из n элементов. Построить множество всех подмножеств элементов массива.**

## **для A={1,2} => {{1},{2},{1,2}};**

## **для A={1,2,3} => {{1},{2},{3},{1,2},{1,3}, {2,3},{1,2,3}}**

## **и т.п.**

using System;

class MainClass

{

//Сортировка

private class SubsetComparer : IComparer<List<int>>

{

public int Compare(List<int> x, List<int> y) => x.Count.CompareTo(y.Count);

}

static List<List<int>> GenerateSubsets(int[] array)

{

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

foreach (int item in array)

{

int currentSubsetCount = subsets.Count;

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

{

List<int> newSubset = new List<int>(subsets[i]);

newSubset.Add(item);

subsets.Add(newSubset);

}

subsets.Add(new List<int> { item });

}

return subsets;

}

//Функция вывода

static void PrintSubsets(List<List<int>> subsets)

{

Console.Write("{");

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

{

Console.Write("{");

for (int j = 0; j < subsets[i].Count; j++)

{

Console.Write(subsets[i][j]);

if (j < subsets[i].Count - 1)

{

Console.Write(",");

}

}

Console.Write("}");

if (i < subsets.Count - 1)

{

Console.Write(",");

}

}

Console.Write("}");

}

static void Main()

{

int[] array = { 1, 2, 3, 4 };

List<List<int>> subsets = GenerateSubsets(array);

subsets.Sort(new SubsetComparer());

PrintSubsets(subsets);

}

}

## **Задача 20. Найти в массиве целых три минимальных числа**

using System;

class Program

{

//Сортировка пузырьком

public static void BubbleSort(ref int[] array)

{

int n = array.Length;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (array[j] > array[j + 1])

{

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

static void Main()

{

int[] array = { 10, 5, 20, 3, 7, 15, 25 };

int count = 3;

BubbleSort(ref array);

int[] smallestNumbers = new int[count];

if (array.Length >= count)

{

Array.Copy(array, smallestNumbers, count);

Console.WriteLine("Три наименьших числа в массиве:");

foreach (int number in smallestNumbers)

{

Console.WriteLine(number);

}

}

}

}

## **Задача 21. Написать программу, которая принимает имя файла, смещение, длину блока и выводит на экран шестнадцатеричный дамп этого блока:**

## 

using System;

class Program

{

static void Main()

{

Console.Write("Введите имя файла: ");

string fileName = Console.ReadLine();

Console.Write("Введите смещение: ");

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

Console.Write("Введите длину блока: ");

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

try

{

using (FileStream fs = new FileStream(fileName, FileMode.Open, FileAccess.Read))

{

byte[] buffer = new byte[blockLength];

fs.Seek(offset, SeekOrigin.Begin);

int bytesRead = fs.Read(buffer, 0, blockLength);

if (bytesRead > 0)

{

Console.WriteLine("Шестнадцатеричный дамп блока:");

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

{

Console.Write("{0:X2} ", buffer[i]);

if ((i + 1) % 16 == 0) // Выводим шестнадцатеричные значения по 16 байт в строке

{

Console.WriteLine();

}

}

Console.WriteLine();

}

else

{

Console.WriteLine("Блок не может быть прочитан.");

}

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

}

}

## **Задача 22. Дан текстовый файл, содержащий упорядоченные по возрастанию числа одной разрядности и число N. Проверить, присутствует ли число N в файле.**

using System;

class Program

{

static bool CheckNumberInFile(string filePath, int targetNumber)

{

using StreamReader reader = new(filePath);

if (reader != null)

{

string line;

while ((line = reader.ReadLine()) != null)

{

// Разделение строки на массив чисел по различным разделителям

string[] separators = { ",", " ", Environment.NewLine };

string[] numbers = line.Split(separators, StringSplitOptions.RemoveEmptyEntries);

// Преобразование строковых значений в числа

int[] parsedNumbers = numbers.Select(int.Parse).ToArray();

// Поиск числа в массиве

foreach (int number in parsedNumbers)

{

if (number == targetNumber)

{

return true;

}

}

}

}

return false;

}

static void Main()

{

string filePath = "input2.txt";

int targetNumber = 42;

// Проверка наличия числа в файле

bool numberFound = CheckNumberInFile(filePath, targetNumber);

if (numberFound)

{

Console.WriteLine($"Число {targetNumber} найдено в файле.");

}

else

{

Console.WriteLine($"Число {targetNumber} не найдено в файле.");

}

}

}

## **Задача 23. Сколько существует натуральных чисел n, меньших 1000, для которых (2^n)-n делится на 7?**

using System;

using System.Numerics;

class Program

{

static void Main()

{

int count = 0;

for (int n = 1; n < 1000; n++)

{

// Вычисляем значение выражения (2^n - n)

BigInteger result = (BigInteger)Math.Pow(2, n) - n;

// Проверяем, делится ли результат на 7

if (result % 7 == 0)

{

count++;

Console.WriteLine(n);

}

}

Console.WriteLine($"Количество натуральных чисел n, для которых (2^n - n) делится на 7: {count}");

}

}

## **Задача 24. Для данного n выяснить, является ли это n членом последовательности Фибоначчи (1,1,2,3,5,8,13...)**

using System;

class Program

{

static bool IsFibonacci(int n)

{

// Проверка, является ли число квадратом числа Фибоначчи

return IsPerfectSquare(5 \* n \* n + 4) || IsPerfectSquare(5 \* n \* n - 4);

}

static bool IsPerfectSquare(int x)

{

int sqrt = (int)Math.Sqrt(x);

return sqrt \* sqrt == x;

}

static void Main()

{

int n = 13;

bool isFibonacci = IsFibonacci(n);

if (isFibonacci)

{

Console.WriteLine($"{n} является членом последовательности Фибоначчи.");

}

else

{

Console.WriteLine($"{n} не является членом последовательности Фибоначчи.");

}

}

}

## **Задача 25. Многочлен с целыми коэффициентами:**

## **P=a0+a1\*x+a2\*x2+…+an\*xn**

## **Задается массивом коэффициентов**

## **P={a0 , a1 , a2 ,…, an }**

## **Разработать программы сложения, умножения и печати многочленов.**

using System;

class Program

{

// Функция сложения двух многочленов

static int[] AddPolynomials(int[] poly1, int[] poly2)

{

int maxLength = Math.Max(poly1.Length, poly2.Length);

int[] result = new int[maxLength];

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

{

int coeff1 = (i < poly1.Length) ? poly1[i] : 0;

int coeff2 = (i < poly2.Length) ? poly2[i] : 0;

result[i] = coeff1 + coeff2;

}

return result;

}

// Функция умножения двух многочленов

static int[] MultiplyPolynomials(int[] poly1, int[] poly2)

{

int[] result = new int[poly1.Length + poly2.Length - 1];

for (int i = 0; i < poly1.Length; i++)

{

for (int j = 0; j < poly2.Length; j++)

{

result[i + j] += poly1[i] \* poly2[j];

}

}

return result;

}

// Функция печати многочлена

static void PrintPolynomial(int[] poly)

{

for (int i = 0; i < poly.Length; i++)

{

if (i > 0)

Console.Write(" + ");

Console.Write($"{poly[i]}x^{i}");

}

Console.WriteLine();

}

static void Main()

{

// Примеры массивов коэффициентов для двух многочленов

int[] polynomial1 = { 1, 2, -3, 3 }; // P1 = 1 + 2x + 3x^2

int[] polynomial2 = { 2, -3, 4, 1 }; // P2 = 2 + 3x + 4x^2

Console.WriteLine("Сложение многочленов:");

int[] sum = AddPolynomials(polynomial1, polynomial2);

PrintPolynomial(sum);

Console.WriteLine("\nУмножение многочленов:");

int[] product = MultiplyPolynomials(polynomial1, polynomial2);

PrintPolynomial(product);

}

}

## **Задача 26. Дано выражение вида: “nn+mm-kk+…zz” Операции только “+” и “-” , порядок произволен. nn, mm, kk …zz – целые числа различной разрядности. Вычислить значение выражения.**

using System;

class Program

{

static int EvaluateExpression(string expression)

{

// Удаление пробелов из выражения

expression = expression.Replace(" ", "");

// Инициализация переменных

int result = 0;

int currentNumber = 0;

bool isNegative = false;

// Обход выражения

foreach (char c in expression)

{

if (c == '+' || c == '-')

{

// Добавление текущего числа к результату с учетом знака

result += isNegative ? -currentNumber : currentNumber;

// Сброс текущего числа и флага отрицательного знака

currentNumber = 0;

isNegative = (c == '-');

}

else

{

// Парсинг текущего символа в число

int digit = c - '0'; //Вычитаем код сивола '0', чтобы получить число

currentNumber = currentNumber \* 10 + digit; //Если число имеет больше одной цифры

}

}

// Добавление последнего числа к результату

result += isNegative ? -currentNumber : currentNumber;

return result;

}

static void Main()

{

string expression = "7 + 15 - 7 + 10 - 13 + 1"; // Пример выражения

int result = EvaluateExpression(expression);

Console.WriteLine("Результат выражения: " + result);

}

}