Лабораторная работа №4

Полный перебор

Решить методом полного p-ичного перебора следующие задачи:

Вариант 6: Текстовый файл состоит из двух строк, длинами не более N (N <= 20). Выяснить, можно ли получить вторую строку из первой путем вычеркивания некоторых символов. Прописные и строчные буквы не различаются, т.е. строки abcd и AbcD считаются одинаковыми. Программа должна находить номера вычеркнутых символов и выводить в текстовый файл все возможные варианты.

Результат работы программы представлен в соответствии с рис.1-6:

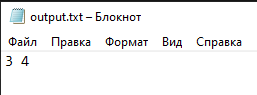
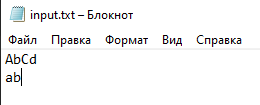


Рис. 1.

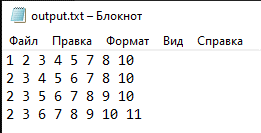
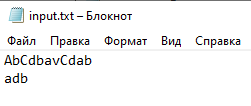


Рис. 2.

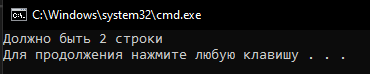
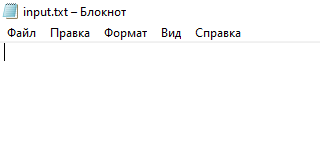


Рис. 3.

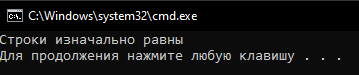
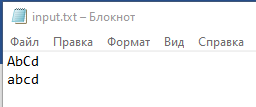


Рис. 4.

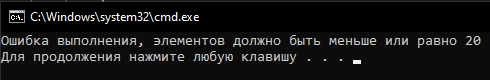
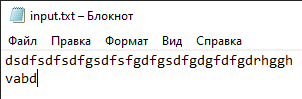


Рис. 5.

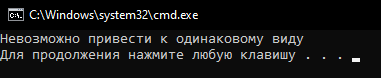
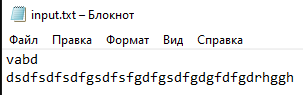


Рис. 6.

Алгоритм работы программы:

* Проверка на наличие строк в файле, если не выполняется условие, то выводится сообщение об ошибке.
* Приводим строки к нижнему регистру.
* Если строки изначально равны, то выводим сообщение об этом и завершаем программу.
* Если длина первой строки будет меньше длины второй строки, то выводим об этом сообщение и завершаем программу.
* Если длина первой строки > 20, то выводим сообщение об этом и завершаем программу.
* Иначе начинаем генерировать все возможные комбинации вычеркнутых символов из первой строки путем вызова метода GenerateCombinations. В методе мы создаем двойной список/список в списке, который будет содержать все сгенерированные комбинации. Через цикл for мы генерируем двоичные представление чисел от 0 до 2^n - 1. Для каждого i в цикле создается новый список, представляющий текущую комбинацию. Далее организуется еще один цикл for от 0 до n-1 для перебора битов числа i. Внутри этого цикла проверяется, установлен ли j-й бит в числе i при помощи поразрядной операции "И", сдвигая 1 влево на j позиций и применяя поразрядное "И" с числом i. Если проверка возвращает ненулевое значение, то j-й бит установлен, и это число включается в текущую комбинацию. По завершению цикла текущая комбинация добавляется в изначальный список. Метод возвращает список combinations, содержащий все сгенерированные комбинации.
* Далее берем комбинации вычеркнутых символов, сортируем комбинация в порядке убывания, чтобы удаление происходило справа налево, чтобы не портить порядок следования символов в строке. На каждой итерации удаляется символ из строки modifiedLine на позиции index. Сравниваем полученные строки, если совпадает, то комбинация номеров вычеркнутых элементов записывается в файл output.txt.

- Чтение строк из файла "input.txt" и помещение их в переменные firstLine и secondLine. Проверка на наличие строк в файле. Если хотя бы одна строка отсутствует, выводится сообщение об ошибке и программа завершается.

- Приведение обеих строк к нижнему регистру с помощью функции ToLowerCase.

- Удаление всех пробелов из обеих строк с помощью функции RemoveSpaces.

- Если строки изначально равны, выводится соответствующее сообщение и программа завершается.

- Если длина первой строки меньше длины второй, выводится сообщение о невозможности приведения строк к одинаковому виду, и программа завершается.

- Если длина первой строки превышает 20 символов, выводится сообщение об ошибке и программа завершается.

- В противном случае начинается процесс генерации всех возможных комбинаций вычеркнутых символов из первой строки путем вызова функции GenerateCombinations.

- Внутри функции вычисляется общее количество комбинаций, равное \(2^n\), где \(n\) — длина первой строки.

- Затем в цикле от 0 до \(2^n - 1\) перебираются все возможные маски. Для каждой маски создается копия первой строки (modifiedLine), в которую будут вноситься изменения.

- Создается массив indices для хранения индексов символов, которые будут вычеркнуты, и переменная count для отслеживания количества вычеркнутых символов.

- Внутри вложенного цикла проверяются биты текущего индекса. Если установленный бит, сохраняется его индекс.

- После завершения цикла по битам осуществляется удаление символов в modifiedLine по индексам из массива indices. Удаление происходит в обратном порядке, чтобы избежать ошибок индексации.

- После удаления символов происходит сравнение modifiedLine с secondLine. Если строки совпадают, индексы вычеркнутых символов записываются в файл output.txt с добавлением 1 для корректной нумерации.

- Программа завершает свою работу после выполнения всех операций и закрытия файловых потоков.

Код программы:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Program

{

static List<List<int>> GenerateCombinations(int n)

{

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

// Генерируем двоичные числа от 0 до 2^n - 1

for (int i = 0; i < Math.Pow(2, n); i++)

{

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

// Перебираем биты числа i

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

{

if ((i & (1 << j)) != 0)

{

combination.Add(j);

}

}

combinations.Add(combination);

}

return combinations;

}

static void Main()

{

// Читаем текстовый файл

string[] lines = File.ReadAllLines("input.txt");

if (lines.Length == 0 || lines.Length == 1)

{

Console.WriteLine("Должно быть 2 строки");

return;

}

// Приводим строки к нижнему регистру и удаляем пробелы

string firstLine = lines[0].ToLower().Replace(" ", "");

string secondLine = lines[1].ToLower().Replace(" ", "");

// Если строки равны, выводим это на консоль и завершаем программу

if (firstLine == secondLine)

{

Console.WriteLine("Строки изначально равны");

return;

}

if (firstLine.Length < secondLine.Length)

{

Console.WriteLine("Невозможно привести к одинаковому виду");

}

int n = firstLine.Length;

if (n > 20)

{

Console.WriteLine("Ошибка выполнения, элементов должно быть меньше или рав-но 20");

}

else

{

// Генерируем все возможные комбинации вычеркнутых символов

List<List<int>> combinations = GenerateCombinations(n);

// Проверяем каждую комбинацию

using (StreamWriter file = new StreamWriter("output.txt"))

{

foreach (List<int> combination in combinations)

{

string modifiedLine = firstLine;

// Вычеркиваем символы по номерам из комбинации

foreach (int index in combination.OrderByDescending(i => i))

{

modifiedLine = modifiedLine.Remove(index, 1);

}

// Если получили вторую строку, записываем комбинацию в текстовый файл

if (modifiedLine == secondLine)

{

// Добавляем +1 к каждому элементу комбинации перед записью в файл

List<string> adjustedCombination = combination.Select(x => (x + 1).ToString()).ToList();

file.WriteLine(string.Join(" ", adjustedCombination));

}

}

}

// Генерация всех возможных комбинаций вычеркнутых символов

}

}

}

С++:

#include <iostream>

#include <fstream>

#include <string>

#include <cmath>

using namespace std;

// Функция для преобразования строки к нижнему регистру

void ToLowerCase(string& str) {

for (size\_t i = 0; i < str.length(); i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

str[i] = str[i] + ('a' - 'A');

}

}

}

// Функция для удаления пробелов из строки

void RemoveSpaces(string& str) {

string result;

for (char c : str) {

if (c != ' ') {

result += c;

}

}

str = result;

}

// Функция для генерации всех возможных комбинаций

void GenerateCombinations(int n, string firstLine, string secondLine, ofstream& outputFile) {

int totalCombinations = pow(2, n); // Общее количество комбинаций

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

string modifiedLine = firstLine;

int indices[20]; // Массив для хранения индексов символов, которые будут вычеркнуты

int count = 0;

// Перебираем биты числа i

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

if (i & (1 << j)) {

indices[count++] = j; // Сохраняем индекс вычеркиваемого символа

}

}

// Вычеркиваем символы по номерам из массива индексов, перебирая его в обратном порядке

for (int j = count - 1; j >= 0; j--) {

modifiedLine.erase(modifiedLine.begin() + indices[j]);

}

// Если полученная строка равна второй строке, записываем комбинацию в текстовый файл

if (modifiedLine == secondLine) {

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

outputFile << (indices[j] + 1) << " "; // +1 для записи номера символа

}

outputFile << endl;

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

// Чтение текстового файла

ifstream inputFile("input.txt");

if (!inputFile.is\_open()) {

cout << "Не удается открыть файл input.txt" << endl;

return 1;

}

string firstLine, secondLine;

getline(inputFile, firstLine);

getline(inputFile, secondLine);

inputFile.close();

if (firstLine.empty() || secondLine.empty()) {

cout << "Должно быть 2 строки" << endl;

return 1;

}

// Приведение строк к нижнему регистру и удаление пробелов

ToLowerCase(firstLine);

RemoveSpaces(firstLine);

ToLowerCase(secondLine);

RemoveSpaces(secondLine);

// Проверка на равенство строк

if (firstLine == secondLine) {

cout << "Строки изначально равны" << endl;

return 0;

}

if (firstLine.length() < secondLine.length()) {

cout << "Невозможно привести к одинаковому виду" << endl;

return 1;

}

int n = firstLine.length();

if (n > 20) {

cout << "Ошибка выполнения, элементов должно быть меньше или равно 20" << endl;

return 1;

}

// Генерация всех возможных комбинаций вычеркнутых символов

ofstream outputFile("output.txt");

if (!outputFile.is\_open()) {

cout << "Не удается открыть файл output.txt" << endl;

return 1;

}

GenerateCombinations(n, firstLine, secondLine, outputFile);

outputFile.close();

return 0;

}