МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**«**Рекурсивные алгоритмы**»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБc-322», «АВТФ»  *Аврамчук Илья Вячеславович*  «23» мая 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевич*  «\_\_\_» \_\_\_\_\_\_ 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

**Цели и задачи работы:** изучение рекурсивного программирования, методов разработки эффективных алгоритмов.

**Методика выполнения работы:**

1) Изучить технологию использования стека при рекурсивном программировании.

2) Написать рекурсивную программу решения поставленной задачи.

3) Протестировать программу.

**Требования к отчету:** отчет по лабораторной работе должен содержать титульный лист, задание, исходный текст программы, тесты.

**Задания:**

1) Задание №1: 1. Дано число m. Требуется в последовательности цифр 1 2 3 4 … 9 расставить знаки «+» и «-» так, чтобы значением получившегося выражения было число m. Например, m=122. Знаки расставляются как 12+34-5-6+78+9.

2) Задание №2: Вариант: 1. Имея входную строку (s) и шаблон (p), реализуйте сопоставление шаблонов с подстановочными знаками с поддержкой '?' и '\*' где: '?' соответствует любому одиночному символу, '\*' соответствует любой последовательности символов (включая пустую последовательность). Сопоставление должно охватывать всю входную строку (не частичную).

**Задание №1:**

**C++:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// функция для вычисления значения выражения

int evaluateExpression(const string &expression) {

int total = 0; // инициализируем итоговую сумму

int currentNumber = 0; // инициализируем текущее число

int sign = 1; // 1 для '+', -1 для '-' (по умолчанию положительное число)

// проходим по каждому символу в строке выражения

for (char c : expression) {

if (isdigit(c)) { // если символ - цифра

currentNumber = currentNumber \* 10 + (c - '0'); // добавляем цифру к текущему числу

} else { // если символ - знак '+' или '-'

total += sign \* currentNumber; // добавляем текущее число с учетом знака к общей сумме

currentNumber = 0; // сбрасываем текущее число

sign = (c == '+') ? 1 : -1; // определяем знак для следующего числа

}

}

total += sign \* currentNumber; // добавляем последнее число к общей сумме

return total; // возвращаем итоговое значение выражения

}

// рекурсивная функция для поиска всех выражений

void findExpressions(string currentExpression, int currentIndex, int target) {

if (currentIndex > 9) {

if (evaluateExpression(currentExpression) == target) {

cout << currentExpression << endl; // если значение выражения равно цели, печатаем выражение

}

return;

}

// добавляем текущую цифру к текущему выражению

findExpressions(currentExpression + to\_string(currentIndex), currentIndex + 1, target);

// добавляем текущую цифру с плюсом

findExpressions(currentExpression + "+" + to\_string(currentIndex), currentIndex + 1, target);

// добавляем текущую цифру с минусом

findExpressions(currentExpression + "-" + to\_string(currentIndex), currentIndex + 1, target);

}

int main() {

int m;

cout << "Введите число m: ";

cin >> m;

// начинаем с первой цифры

findExpressions("1", 2, m);

return 0;

}

**Java:**

import java.util.Scanner;

public class Main {

// функция для вычисления значения выражения

public static int evaluateExpression(String expression) {

int total = 0; // инициализируем итоговую сумму

int currentNumber = 0; // инициализируем текущее число

int sign = 1; // 1 для '+', -1 для '-' (по умолчанию положительное число)

// проходим по каждому символу в строке выражения

for (char c : expression.toCharArray()) {

if (Character.isDigit(c)) { // если символ - цифра

currentNumber = currentNumber \* 10 + (c - '0'); // добавляем цифру к текущему числу

} else { // если символ - знак '+' или '-'

total += sign \* currentNumber; // добавляем текущее число с учетом знака к общей сумме

currentNumber = 0; // сбрасываем текущее число

sign = (c == '+') ? 1 : -1; // определяем знак для следующего числа

}

}

total += sign \* currentNumber; // добавляем последнее число к общей сумме

return total; // возвращаем итоговое значение выражения

}

// рекурсивная функция для поиска всех выражений

public static void findExpressions(String currentExpression, int currentIndex, int target) {

if (currentIndex > 9) {

if (evaluateExpression(currentExpression) == target) {

System.out.println(currentExpression); // если значение выражения равно цели, печатаем выражение

}

return;

}

// добавляем текущую цифру к текущему выражению

findExpressions(currentExpression + currentIndex, currentIndex + 1, target);

// добавляем текущую цифру с плюсом

findExpressions(currentExpression + "+" + currentIndex, currentIndex + 1, target);

// добавляем текущую цифру с минусом

findExpressions(currentExpression + "-" + currentIndex, currentIndex + 1, target);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите число m: ");

int m = scanner.nextInt();

// начинаем с первой цифры

findExpressions("1", 2, m);

scanner.close();

}

}

**Задание №2:**

**C++:**

#include <iostream>

#include <vector>

#include <string>

using namespace std;

bool isMatch(string s, string p) {

int m = s.size(); // инициализируем переменную m длиной строки s

int n = p.size(); // инициализируем переменную n длиной строки p

// создаем двумерный массив для хранения результатов сопоставления

vector<vector<bool>> dp(m + 1, vector<bool>(n + 1, false));

// пустая строка и пустой шаблон сопоставляются

dp[0][0] = true;

// обрабатываем случай, когда шаблон начинается с '\*'

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

if (p[j - 1] == '\*') { // если текущий символ шаблона - '\*'

dp[0][j] = dp[0][j - 1]; // устанавливаем значение в соответствии с предыдущим символом шаблона

}

}

// заполняем таблицу сопоставления

for (int i = 1; i <= m; ++i) {

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

if (p[j - 1] == '?' || s[i - 1] == p[j - 1]) { // если текущий символ шаблона - '?' или совпадает с текущим символом строки

dp[i][j] = dp[i - 1][j - 1]; // устанавливаем значение в соответствии с предыдущими символами

} else if (p[j - 1] == '\*') { // если текущий символ шаблона - '\*'

dp[i][j] = dp[i - 1][j] || dp[i][j - 1]; // устанавливаем значение в соответствии с предыдущими символами и их комбинациями

}

}

}

// возвращаем результат сопоставления для всей строки

return dp[m][n];

}

int main() {

string s1 = "aa", p1 = "a";

string s2 = "aa", p2 = "\*";

string s3 = "cb", p3 = "?a";

cout << "Пример 1: " << (isMatch(s1, p1) ? "true" : "false") << endl;

cout << "Пример 2: " << (isMatch(s2, p2) ? "true" : "false") << endl;

cout << "Пример 3: " << (isMatch(s3, p3) ? "true" : "false") << endl;

return 0;

}

**Java:**

import java.util.Arrays;

public class Main {

public static boolean isMatch(String s, String p) {

int m = s.length(); // инициализируем переменную m длиной строки s

int n = p.length(); // инициализируем переменную n длиной строки p

// создаем двумерный массив для хранения результатов сопоставления

boolean[][] dp = new boolean[m + 1][n + 1];

// пустая строка и пустой шаблон сопоставляются

dp[0][0] = true;

// обрабатываем случай, когда шаблон начинается с '\*'

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

if (p.charAt(j - 1) == '\*') { // если текущий символ шаблона - '\*'

dp[0][j] = dp[0][j - 1]; // устанавливаем значение в соответствии с предыдущим символом шаблона

}

}

// заполняем таблицу сопоставления

for (int i = 1; i <= m; ++i) {

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

if (p.charAt(j - 1) == '?' || s.charAt(i - 1) == p.charAt(j - 1)) { // если текущий символ шаблона - '?' или совпадает с текущим символом строки

dp[i][j] = dp[i - 1][j - 1]; // устанавливаем значение в соответствии с предыдущими символами

} else if (p.charAt(j - 1) == '\*') { // если текущий символ шаблона - '\*'

dp[i][j] = dp[i - 1][j] || dp[i][j - 1]; // устанавливаем значение в соответствии с предыдущими символами и их комбинациями

}

}

}

// возвращаем результат сопоставления для всей строки

return dp[m][n];

}

public static void main(String[] args) {

String s1 = "aa", p1 = "a";

String s2 = "aa", p2 = "\*";

String s3 = "cb", p3 = "?a";

System.out.println("Пример 1: " + (isMatch(s1, p1) ? "true" : "false"));

System.out.println("Пример 2: " + (isMatch(s2, p2) ? "true" : "false"));

System.out.println("Пример 3: " + (isMatch(s3, p3) ? "true" : "false"));

}

}

Результаты работы программ:

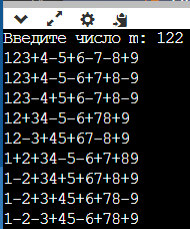


Рисунок 1. Задание №1

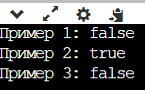


Рисунок 2. Задание №2

Вывод: Мы научились работать с рекурсией.