МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе №7**

**«**Рекурсивные алгоритмы**»**

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

Выполнил:Проверил:

Студент гр. «АБс-324», «АВТФ» *Ассистент кафедры ЗИ*

*Клаус В. А. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

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

(подпись) (подпись)

Новосибирск 2024

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

**Задания:**

1. Задан массив целых чисел. Построить из них любую последовательность таким образом, чтобы последняя цифра предыдущего числа совпадала с первой цифрой следующего. **Вариант 10**.
2. Получив строку s, представляющую допустимое выражение, реализуйте базовый калькулятор для его вычисления и найдите результат вычисления. **Вариант 10**.

**Задание 1**

**C++:**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int lastnum(int num) { // последняя цифра числа**

**return num % 10;**

**}**

**int firstnum(int num) { // первая цифра числа**

**while (num >= 10) {**

**num /= 10;**

**}**

**return num;**

**}**

**void getposledov(vector<int> nums, vector<int> posled, int num, int i) { // получение последовательности**

**int last = lastnum(num);**

**for (int j = 0; j < nums.size(); ++j) { // сравниваем последнюю цифру числа с первыми цифрами всех чисел массива**

**int first = firstnum(nums[j]);**

**if (last == first && num != nums[j]) { // Если найдено, то добавим число в последовательность**

**for (int l = 0; l < nums.size(); ++l) { // убираем повторения**

**if (nums[l] == num) {**

**nums[l] = 0;**

**}**

**}**

**posled.push\_back(num);**

**num = nums[j];**

**break; // выход из цикла, чтобы не было повторов**

**}**

**}**

**if (posled.empty()) { // если последовательность пуста(не нашли подходящие числа для числа)**

**i++; // то добавляем счетчик и не даём уйти за границы**

**if (i <= nums.size()) return getposledov(nums, posled, nums[i], i); // если счётчик впорядке, то запускаем рекурсию**

**else cout << "не найдена"; // ушли за счётчик и последовательность в итоге пуста - нельзя составить последовательность**

**} else {**

**i++; // иначе повторяем тоже самое, только уже для найденного числа последовательности, чтобы продлить последовательность**

**if (i <= nums.size()) return getposledov(nums, posled, num, i);**

**else {**

**posled.push\_back(num);**

**for(int k : posled) cout << k << " ";**

**}**

**}**

**}**

**int main() {**

**setlocale(LC\_ALL, "Rus");**

**vector<int> nums = {101, 51, 15, 107, 89}; // 101 15 51 107**

**vector<int> posled;**

**int i = 0;**

**cout << "Последовательность: ";**

**getposledov(nums, posled, nums[i], i);**

**return 0;**

**}**

**Input:**

**Последовательность: 101 15 51 107**

**Python:**

**def lastnum(num): # последняя цифра числа**

**return num % 10**

**def firstnum(num): # первая цифра числа**

**while num >= 10:**

**num //= 10**

**return num**

**def getposledov(nums, posled, num, i): # получение последовательности**

**last = lastnum(num)**

**for j in range(len(nums)): # сравниваем последнюю цифру числа с первыми цифрами всех чисел массива**

**first = firstnum(nums[j])**

**if last == first and num != nums[j]: # Если найдено, то добавим число в последовательность**

**for l in range(len(nums)): # убираем повторения**

**if nums[l] == num:**

**nums[l] = 0**

**posled.append(num)**

**num = nums[j]**

**break # выход из цикла, чтобы не было повторов**

**if not posled: # если последовательность пуста(не нашли подходящие числа для числа)**

**i += 1 # то добавляем счетчик и не даём уйти за границы**

**if i <= len(nums):**

**getposledov(nums, posled, nums[i], i) # если счётчик впорядке, то запускаем рекурсию**

**else:**

**print("не найдена") # ушли за счётчик и последовательность в итоге пуста - нельзя составить последовательность**

**else:**

**i += 1 # иначе повторяем тоже самое, только уже для найденного числа последовательности, чтобы продлить последовательность**

**if i <= len(nums):**

**getposledov(nums, posled, num, i)**

**else:**

**posled.append(num)**

**for k in posled:**

**print(k, end=" ")**

**nums = [101, 51, 15, 107, 89] # 101 15 51 107**

**posled = []**

**i = 0**

**print("Последовательность: ", end="")**

**getposledov(nums, posled, nums[i], i)**

**Input:**

**Последовательность: 101 15 51 107**

**Задание 2**

**C++:**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int Expression(string &s, int &i);**

**int calculate(string &s) {**

**int i = 0;**

**return Expression(s, i);**

**}**

**int Expression(string &s, int &i) {**

**int result = 0;**

**int num = 0;**

**int sign = 1;**

**string C;**

**while (i < s.length()) {**

**char c = s[i];**

**if (isdigit(c)) { // Если символ - число, то добавляем его в строку и переводим в int**

**C += c;**

**num = stoi(C);**

**} else if ((c == '-' && i == 0) || (c == '-' && s[i+1] == '(')) { // цикл для корректной работы калькулятора( - в самом начале, - перед '(' )**

**sign = -1;**

**} else if (c == '+' && s[i+1] == '(') { // цикл для корректной работы калькулятора( + перед '(' )**

**sign = 1;**

**} else if (c == '+') { // Если +, то...**

**num = stoi(C); // переводим число в строку(ещё раз, чтобы он считал также числа, состоящие из 2+ цифр)**

**C = ""; // обнуляем строку**

**result += sign \* num; // записываем результат**

**num = 0; // обнуляем промежуточную переменную num**

**sign = 1; // обновляем знак**

**} else if (c == '-') { // Если -, то повторяем действия как с +, только меняем знак**

**num = stoi(C);**

**C = "";**

**result += sign \* num;**

**num = 0;**

**sign = -1;**

**} else if (c == '(') { // Если (, то запускаем рекурсию и высчитываем значение в скобках**

**i++; // переход к следующему элементу**

**int resultInSkobka = Expression(s, i); // рекурсия**

**result += sign \* resultInSkobka; // значение рекурсии добавляем в общий результат**

**} else if (c == ')') { // Если ), то рекурсия заканчивается и записывается результат рекурсии**

**result += sign \* num;**

**return result;**

**}**

**i++;**

**}**

**return result + sign \* num;**

**}**

**int main() {**

**setlocale(LC\_ALL, "Rus");**

**string expression; // выражение**

**while (true) {**

**bool check = true;**

**cout << "Введите выражение: ";**

**cin >> expression;**

**if (expression.length()>3\*1e5) { // проверка длины выражения**

**cout << "Слишком много аргументов у выражения!" << endl;**

**} else {**

**int j = 0;**

**while (j < expression.length()) { // проверка на правильный ввод аргументов**

**if (!isdigit(expression[j]) && expression[j] != '+' && expression[j] != '-' && expression[j] != '(' && expression[j] != ')') {**

**check = false;**

**break;**

**}**

**j++;**

**}**

**if (check == true) cout << "Результат: " << calculate(expression) << endl << endl; // Вывод результата, если всё введено верно**

**else cout << "Ошибка в вводе аргументов!" << endl << endl; // Иначе вывод ошибки**

**}**

**}**

**return 0;**

**}**

**Input:**

**Введите выражение: 2+2**

**Результат: 4**

**Введите выражение: 360-54**

**Результат: 306**

**Введите выражение: (1+(4+5+2)-3)+(6+8)**

**Результат: 23**

**Введите выражение: 2\*2**

**Ошибка в вводе аргументов!**

**Python:**

**def expression(s, i):**

**result = 0**

**num = 0**

**sign = 1**

**C = ""**

**while i < len(s):**

**c = s[i]**

**if c.isdigit(): # Если символ - число, то добавляем его в строку и переводим в int**

**C += c**

**num = int(C)**

**elif (c == '-' and i == 0) or (c == '-' and s[i+1] == '('): # цикл для корректной работы калькулятора( - в самом начале, - перед '(' )**

**sign = -1**

**elif c == '+' and s[i+1] == '(': # цикл для корректной работы калькулятора( + перед '(' )**

**sign = 1**

**elif c == '+': # Если +, то...**

**num = int(C) # переводим число в строку(ещё раз, чтобы он считал также числа, состоящие из 2+ цифр)**

**C = "" # обнуляем строку**

**result += sign \* num # записываем результат**

**num = 0 # обнуляем промежуточную переменную num**

**sign = 1 # обновляем знак**

**elif c == '-': # Если -, то повторяем действия как с +, только меняем знак**

**num = int(C)**

**C = ""**

**result += sign \* num**

**num = 0**

**sign = -1**

**elif c == '(': # Если (, то запускаем рекурсию и высчитываем значение в скобках**

**i += 1 # переход к следующему элементу**

**result\_in\_bracket = expression(s, i) # рекурсия**

**result += sign \* result\_in\_bracket # значение рекурсии добавляем в общий результат**

**elif c == ')': # Если ), то рекурсия заканчивается и записывается результат рекурсии**

**result += sign \* num**

**return result**

**i += 1**

**return result + sign \* num**

**def calculate(s):**

**i = 0**

**return expression(s, i)**

**while True:**

**expression\_input = input("Введите выражение: ")**

**if len(expression\_input) > 3 \* 10\*\*5: # проверка длины выражения**

**print("Слишком много аргументов у выражения!")**

**else:**

**check = all(c.isdigit() or c in "+-()" for c in expression\_input)**

**if check:**

**print("Результат:", calculate(expression\_input), "\n")**

**else:**

**print("Ошибка в вводе аргументов!\n")**

**Input:**

**Введите выражение: 2+2**

**Результат: 4**

**Введите выражение: 360-54**

**Результат: 306**

**Введите выражение: (1+(4+5+2)-3)+(6+8)**

**Результат: 23**

**Введите выражение: 2\*2**

**Ошибка в вводе аргументов!**

**Вывод:**

В ходе выполнения лабораторной работы №7 я познакомился с понятием рекурсии и научился реализовывать её на различных примерах. Также реализовал свой первый базовый калькулятор, с помощью рекурсии, на языках программирования C++ и Python.

**Ссылка на репозиторий:** [нажмите](https://github.com/JamesSevil/laba7)