

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 9

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Рекурсивные алгоритмы и их реализация

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

# **Оглавление**

[ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 9 1](#_Toc168313330)

[Оглавление 2](#_Toc168313331)

[1. Требования к выполнению практической работы 3](#_Toc168313332)

[2.Задание 1 3](#_Toc168313333)

[2.1. Требования к выполнению задания 1 3](#_Toc168313334)

[2.2. Разработка алгоритма 4](#_Toc168313335)

[2.3. Код программы 5](#_Toc168313336)

[2.4. Тестирование программы 5](#_Toc168313337)

[3. Задание 2 6](#_Toc168313338)

[3.1. Требования к выполнению задания 2 6](#_Toc168313339)

[3.2. Код программы 6](#_Toc168313340)

[3.3. Тестирование программы для задания 2 8](#_Toc168313341)

[4. Выводы 9](#_Toc168313342)

[5. Информационные источники 10](#_Toc168313343)

# Требования к выполнению практической работы

Требования к выполнению практической работы В данной практической работе требуется реализовать два задания. Варианты задач к заданиям в табл. 20. Первое задание направлено на освоении процесса разработки алгоритма, включающего: определение рекурсивного процесса решения задачи, запись его рекурсивной зависимости, оценка сложности рекурсивного алгоритма подходящим для него методом, реализация алгоритма. В этом задании решается первая задача варианта. Второе задание направлено на закрепление знаний и умений по разработке и реализации более сложных рекурсивных процессов. В этом задании решается вторая задача варианта.

# 2.Задание 1

Реализовать рекурсивную функцию для решения первой задачи варианта.

## 2.1. Требования к выполнению задания 1

1. Выполнить разработку и оценить сложность алгоритма, для этого: − определить, что процесс решения задачи может быть рекурсивным, для этого приведите рекуррентную зависимость процесса; − определить глубину рекурсии, изменяя исходные данные; − определить сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии; − проверить возможность применения основного метода оценки сложности рекурсивного алгоритма, если метод допустим, то проверить, что уже полученное вами решение совпадает с решением по этому методу.

2. Реализовать рекуррентную зависимость, разработав рекурсивную функцию. Примечание. При разработке алгоритма принимайте во внимание результат алгоритма. Если алгоритм формирует одно значение простого или ссылочного типа, то его, возможно удобно реализовать как функцию, возвращающую результат.

3. Подготовить тесты.

4. Разработать программу, демонстрирующую получение решения задачи на подготовленных тестах.

5. Оформить отчет

Персональный вариант №22

*Вычислить с заданной точностью ε>0 значение функции y=cos(x), используя представление ее рядом. Точность считается достигнутой, если очередное слагаемое по модулю меньше ε. Гармоничный ряд вычисления: y=cos(x)=1-x2/2!+x4/4!-…+(-1)nx2n/(2n!)+….*

## 2.2. Разработка алгоритма

Рассмотрим вычисление 𝑦 = *cos(x)* с заданной точностью 𝜀>0 с использованием разложения функции *cos(x)* в ряд Тейлора:

Рекурсивное решение:

1) Рекуррентная зависимость: можно использовать An = , где

A0 = 1,

An =

2) Глубина рекурсии: Глубина рекурсии определяется количеством членов ряда, необходимых для достижения заданной точности *ε*. Алгоритм прекращает вычисления, когда текущий член ряда 𝑎𝑛 по модулю становится меньше *ε*.

|An| < *ε*

3) Алгоритм:

|  |
| --- |
| double cos\_recursive(double x, double epsilon, int n = 0, double term = 1) {  if (std::abs(term) < epsilon) {  return 0;  }  return term + cos\_recursive(x, epsilon, n + 1, -term \* x \* x / ((2 \* n + 2) \* (2 \* n + 1)));  }} |

4) F(x) = 0, если n < epsilon

term + f (x, n + 1, -term \* x \* x / ((2 \* n + 2) \* (2 \* n + 1))) иначе

|  |
| --- |
| exp\_recursive(1, 0, 0)  |  +-- term = 1.0  | +-- exp\_recursive((1,1,−1∗1∗1/(2∗0+2)∗(2∗0+1)))  | |  | +-- term = −0.5  | | +-- exp\_recursive (1,2,0.5∗1∗1/(2∗1+2)∗(2∗1+1))  | | |  | | +-- term = 0.0416667  | | | +-- exp\_recursive(1,3,−0.0416667∗1∗1/(2∗2+2)∗(2∗2+1))  | | | |  | | | +-- term = -0.00138889  | | +-- returns 0.0416667 - 0.00138889  | +-- returns −0.5+ 0.0416667+ 0.00138889  +-- returns 1.0 + −0.5+ 0.0416667+ 0.00138889 |

## 2.3. Код программы

|  |
| --- |
| #include <cmath>  #include <iostream>  // Функция для вычисления cos(x) рекурсивно  double cos\_recursive(double x, double epsilon, int n = 0, double term = 1) {  if (std::abs(term) < epsilon) {  return 0;  }  return term + cos\_recursive(x, epsilon, n + 1, -term \* x \* x / ((2 \* n + 2) \* (2 \* n + 1)));  }  int main() {  double x = 1.0; // значение для которого нужно вычислить cos(x)  double epsilon = 1e-6; // точность  std::cout << "cos(" << x << ") = " << cos\_recursive(x, epsilon) << std::endl;  return 0;  } |

## 2.4. Тестирование программы

|  |  |
| --- | --- |
| Входные данные | Результат тестирования |
| X = 1 | *cos(1) = 0.540303* |
| X = 5 | *cos(5) = 0.283662* |



Результат первого тестирования



Результат второго тестирования

# 3. Задание 2

Разработать рекурсивную функцию (функции) для обработки списковой структуры согласно варианту.

## 3.1. Требования к выполнению задания 2

1. Разработать функцию создания исходного списка. Для создания списка может быть разработана простая или рекурсивная функция по желанию (в тех вариантах, где не требуется рекурсивное создание списка).

2. Разработать рекурсивную функцию (функции), реализующую вторую задачу варианта.

3. Оформить отчет.

Персональный вариант №22

*Дан линейный однонаправленный список, значения узлов которого символы, и*

*его длина. Длина списка четное число. Определить содержит ли список слово*

*палиндром.*

## 3.2. Код программы

|  |
| --- |
| #include <iostream>  using namespace std;  struct Node {  char data;  Node\* next;  };  //Создает однонаправленный список длиной length, запрашивая у пользователя ввод символов для каждого узла.  Node\* createList(int length) {  if (length <= 0) return nullptr;  Node\* head = new Node();  Node\* current = head;  cout << "Введите символы для списка: ";  for (int i = 0; i < length; ++i) {  char ch;  cin >> ch;  current->data = ch;  if (i < length - 1) {  current->next = new Node();  current = current->next;  }  else {  current->next = nullptr;  }  }  return head;  }  //Использует два указателя: left и right. right двигается к концу списка, а left двигается навстречу right, когда стек разворачивается. Проверка осуществляется по парам элементов.  bool isPalindromeHelper(Node\*& left, Node\* right) {  if (right == nullptr) return true;  bool isPal = isPalindromeHelper(left, right->next);  if (!isPal) return false;  bool isCurrentPairEqual = (left->data == right->data);  left = left->next;  return isCurrentPairEqual;  }  bool isPalindrome(Node\* head) {  return isPalindromeHelper(head, head);  }  void deleteList(Node\* head) {  while (head != nullptr) {  Node\* temp = head;  head = head->next;  delete temp;  }  }  int main() {  setlocale(LC\_ALL, "Rus");  int length;  cout << "Введите длину списка: ";  cin >> length;  if (length % 2 != 0) {  cerr << "Длина списка должна быть четной." << endl;  return 1;  }  Node\* head = createList(length);  if (isPalindrome(head)) {  cout << "Этот список является палиндромом." << endl;  }  else {  cout << "Этот список не является палиндромом." << endl;  }  deleteList(head);  return 0;  } |

## 3.3. Тестирование программы для задания 2

|  |  |
| --- | --- |
| Входные данные | Результат тестирования |
| 6 abccba | Этот список является палиндромом. |
| 8  a b c d a b c d | Этот список не является палиндромом. |
| 2  a a | Этот список является палиндромом. |



Результат первого тестирования



Результат второго тестирования



Результат первого тестирования

# 4. Выводы

В ходе самостоятельной работы была изучена технология разработки алгоритмов с использованием рекурсии, а также применение рекурсии на практике на примере математической задачи, а также на примере работы с однонаправленными узлами. Также был изучен механизм оценивания рекуррентного алгоритма.

# 5. Информационные источники

1. Приложение к практическим работам – СДО (online-edu.mirea.ru)

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.