

Занятие 9. Рекурсивные алгоритмы и их реализация

Цель. Получить знания и практические навыки по разработке и реализации рекурсивных процессов

Оглавление

1	Ответить на вопросы.....	1
2	Задание	1
3	Варианты.....	2
4	Форма отчета	3
5	Примеры реализации рекурсивных алгоритмов.....	4
	Рекурсивная функция	4

1 Ответить на вопросы

Дайте определение понятиям

- 1) Определение рекурсивной функции
- 2) Шаг рекурсии -
- 3) Глубина рекурсии -
- 4) Условие завершения рекурсии -
- 5) Виды рекурсии
 - линейная -
 - каскадная –
- 6) Прямая и косвенная рекурсия
- 7) Организация стека рекурсивных вызовов

2 Задание

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта

1) Требования к выполнению первой задачи варианта:

- приведите итерационный алгоритм решения задачи
- реализуйте алгоритм в виде функции и отладьте его
- определите теоретическую сложность алгоритма
- опишите рекуррентную зависимость в решении задачи
- реализуйте и отладьте рекурсивную функцию решения задачи
- определите глубину рекурсии, изменяя исходные данные
- определите сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии
- приведите для одного из значений схему рекурсивных вызовов

- разработайте программу демонстрирующую выполнение обеих функций и покажите результаты тестирования.
- 2) Требования к выполнению второй задачи варианта:
- рекурсивную функцию для обработки списковой структуры согласно варианту. Информационная часть узла – простого типа – целого;
 - для создания списка может быть разработана простая или рекурсивная функция по желанию (в тех вариантах, где не требуется рекурсивное создание списка);
 - определите глубину рекурсии
 - определите теоретическую сложность алгоритма
 - разработайте программу, демонстрирующую работу функций и покажите результаты тестов.
- 3) Составить отчет по выполненному заданию

3 Варианты

Номер	Задачи
1	1. Найти наибольший общий делитель двух целых чисел 2. Создание и вывод линейного однонаправленного списка из n элементов
2	1. Найти n -ое число Фибоначчи. 2. В однонаправленном списке из n элементов найти элемент с заданным значением и вернуть на него указатель.
3	1. Определить делится ли число на каждую из своих цифр. 2. Не используя связанный стек проверить баланс скобок в арифметическом выражении, которое передано как строка.
4	1. Определить является ли текст – палиндромом. 2. Удалить из связанного однонаправленного списка все элементы, равные заданному.
5	1. Дан массив из n элементов вещественного типа. Вычислить среднее значение всех элементов массива. 2. Создание связанного стека из n элементов.
6	1. Сколько квадратов можно отрезать от прямоугольника со сторонами a и b . 2. Удаление связанного стека.
7	1. Найти максимальный элемент в массиве из n элементов. 2. Создание очереди на однонаправленном списке.
8	1. Перевести число из 10-системы счисления в систему с основанием $B(1 < B \leq 10)$ 2. Удаление очереди, реализованной на однонаправленном списке
9	1. Бинарный поиск элемента в массиве 2. Создание двунаправленного списка.
10	1. Вычислить значение цифрового корня для некоторого целого числа N . 2. Найти в двунаправленном списке количество четных элементов.

11	<ol style="list-style-type: none"> 1. Вычислить $x_1(x_2+x_3)(x_4+x_5+x_6)....(x_{46}+x_{47}+...+x_{55})$. 2. Удаление двунаправленного списка
12	<ol style="list-style-type: none"> 1. Сортировка массива по возрастанию 2. Создать новый однонаправленный список из исходного однонаправленного списка, записав его элементы наоборот.
13	<ol style="list-style-type: none"> 1. Дана последовательность из N чисел $X_1, X_2, ..., X_N$. Вычислить значение выражения: $X_n(X_n+X_{n-1})(X_n+X_{n-1}+X_{n-2})(X_n+X_{n-1}+X_{n-2}+X_{n-3})... (X_n+X_{n-1}+X_{n-2}+...+X_1)$. Массив не использовать. 2. Удалить из однонаправленного списка нули.
14	<ol style="list-style-type: none"> 1. Дана строка. Выполнить переворот строки (записать наоборот) на ее же месте в памяти. 2. Определить количество вхождений: положительных, отрицательных, нулевых значений в линейном списке.
15	<ol style="list-style-type: none"> 1. Ханойская башня. 2. Удалить однонаправленный список.
16	<ol style="list-style-type: none"> 1. Прохождение лабиринта 2. Определить симметрично ли число, цифры которого последовательно записаны в узлах двунаправленного списка

4 Форма отчета

1. Титульный лист
2. Ответы на вопросы
3. Отчет по задаче 1
 - 1) Условие задачи
 - 2) Постановка задачи
 - 3) Описание алгоритма – рекуррентная зависимость
 - 4) Коды используемых функций
 - 5) Ответы на задания по задаче 1: список требований к задаче 1
 - 6) Код программы и скриншоты результатов тестирования
4. Отчет по задаче 2
 - 1) Условие задачи
 - 2) Постановка задачи
 - 3) Описание алгоритма – рекуррентная зависимость
 - 4) Коды используемых функций
 - 5) Ответы на задания по задаче 1: список требований к задаче 1
 - 6) Код программы и скриншоты результатов тестирования

5 Примеры реализации рекурсивных алгоритмов

Рекурсивная функция

```
#include "stdafx.h"
```

```
#include "iostream.h"
```

```
// Задача 1. Дана последовательность целых чисел, заканчивающаяся нулем
```

```
// вывести сначала положительные, а затем отрицательные значения
```

```
void rec1();
```

Рекуррентная зависимость

$$f(\quad) = \begin{cases} \text{cin} \gg n & \text{Вывод } n \text{ и шаг в рекурсию при } n > 0 \\ \text{cin} \gg n & \text{Шаг в рекурсию и вывод } n \text{ при } n < 0 \\ \text{cin} \gg n & \text{Выход из рекурсии при } n = 0 \end{cases}$$

```
//Задача 2. Вычислить  $x^n$ . При  $x=0$  и  $n<0$  результат INFINITY
```

$$x^n = \begin{cases} 1 & \text{если } n = 0 \\ x * x^{n-1} & \text{если } n > 0 \\ 1/x^n & \text{если } n < 0 \end{cases}$$

Такое определение алгоритма говорит об его рекурсивной природе

```
int rec2(int x, int n);
```

```
int main()
```

```
{
    rec1();
    cout<<rec2(2,3);

    double rez=rec2(0, -3);
    if (rez == INFINITY)
        std::cout << "zero divide";
    else
        std::cout << rez;
    return 0;
}
```

```
void rec1()
```

```
{int n;
    cin>>n;
    if (n==0)
        return;
    else
```

```

        if(n>0)
        {
            cout<<n;
            rec11();
        }
        else
        {rec11();
        cout<<n;
        }
    }
double  rec2(int x, int n)
{
    if (n==0)
    {
        return 1;
    }
    If (n>0)
        // step recursii rec2(x,n)=x*rec2(x,n-1)
        return x*rec2(x,n-1);
    if(n<0){
        return 1/rec2(x,abs(n));
    }
}

```