|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | МИНОБРНАУКИ РОССИИ | | | | 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 ЛАБОРАТОРНОЙ РАБОТЕ №3** | |
| **по дисциплине** | |
| **«Структуры и алгоритмы обработки данных»**  **По теме: «Рекурсивные функции»**  **Вариант 2** | |
|  | |
| Выполнил студент группы ИВБО-06-17 | Бикеев А. И. |
| Принял преподаватель | Скворцова Л.А. |

Москва 2019

**Оглавление**

[1. Задание 1. Разработать функцию на рекурсии. 3](#_Toc26226122)

[1.1. Вариант задания 3](#_Toc26226123)

[1.2. Рекурентное соотношение 3](#_Toc26226124)

[1.3. Реализация рекурсивной функции 3](#_Toc26226125)

[1.4. Таблица тестов 3](#_Toc26226126)

[1.5. Текст исходного кода (листинг) программы 3](#_Toc26226127)

[2. Задание 2. Разработать функцию на рекурсии, используя структуру списка. 4](#_Toc26226128)

[2.1. Вариант задания 4](#_Toc26226129)

[2.2. Рекурентное соотношение 5](#_Toc26226130)

[2.3. Реализация рекурсивной функции 5](#_Toc26226131)

[1.1. Таблица тестов 5](#_Toc26226132)

[1.2. Текст исходного кода (листинг) программы 5](#_Toc26226133)

[2. Задание 3. Разработать программу тестирования рекурсивных функций. 7](#_Toc26226134)

[2.1. Вариант задания 7](#_Toc26226135)

[2.2. Контрольные прогоны программы 7](#_Toc26226136)

1. Задание 1. Разработать функцию на рекурсии.
   1. Вариант задания

Найти n-ое число Фибоначчи.

Требования:

* 1. Описать рекурсивную зависимость в виде табличной функции
  2. Определить глубину рекурсии.
  3. Рекурентное соотношение

, где n – номер числа Фибоначчи глубина рекурсии

* 1. Реализация рекурсивной функции

int Fibonachi(int n)

{

if(!n) return 0;

else if (n == 1 || n == 2) return 1;

return Fibonachi(n - 1) + Fibonachi(n - 2);

}

* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Исходные данные** | **Эталон результата, глубина рекурсии** |
| **1** | 12 | 144 |
| **2** | 5 | 5 |
| **3** | 6 | 8 |

* 1. Текст исходного кода (листинг) программы

#include "Opeartion.h"

int Fibonachi(int n)

{

    if(!n) return 0;

    else if (n == 1 || n == 2) return 1;

    return Fibonachi(n - 1) + Fibonachi(n - 2);

}

LinkedList\* CreateList(Node\* head, int n)

{

    if (n == 0)

    {

        return nullptr;

    }

    head = (Node\*)malloc(sizeof(Node));

    if (head == nullptr) {

        throw new exception("Не удалось выделить память под хранение указателя на элемент списка\n");

    }

    cout << "Введите значения для"<<n<<" элемента списка: ";

    cin >> head->value;

    head->next = CreateList(head->next, n - 1);

    return head;

}

void ClearList(Node\* head)

{

    if (head->next != nullptr)

    {

        if (head->next->next != nullptr)

        {

            ClearList(head->next);

        }

        free(head->next);

    }

}

Node\* FindNode(Node\* current, int value)

{

    if (current)

    {

        if (current->value == value)

        {

            return current;

        }

        return FindNode(current->next, value);

    }

    return nullptr;

}

1. Задание 2. Разработать функцию на рекурсии, используя структуру списка.
   1. Вариант задания

В однонаправленном списке из n элементов найти элемент с заданным значением и вернуть на него указатель.

* 1. Рекурентное соотношение

, где value– значение, которое должен содержать искомый узел, current – указатель на текущий элемент списка, next – указатель на следующий элемент списка.

* 1. Реализация рекурсивной функции

Node\* FindNode(Node\* current, int value)

{

if (current)

{

if (current->value == value)

{

return current;

}

return FindNode(current->next, value);

}

return nullptr;

}

* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Исходные данные** | **Эталон результата** |
| **1** | **Список из 2 элементов**  **[1, 2]**  **Ищем value = 2** | 006FFBAC |
| **2** | **Список из 3 элементов**  **[1, 2, 3]**  **Ищем value = 3** | 0117FA04 |
| **3** | **Список из 4 элементов**  **[1, 2, 3, 5]**  **Ищем value = 5** | 106CFAAC |

* 1. Текст исходного кода (листинг) программы

#include "Opeartion.h"

int Fibonachi(int n)

{

    if(!n) return 0;

    else if (n == 1 || n == 2) return 1;

    return Fibonachi(n - 1) + Fibonachi(n - 2);

}

LinkedList\* CreateList(Node\* head, int n)

{

    if (n == 0)

    {

        return nullptr;

    }

    head = (Node\*)malloc(sizeof(Node));

    if (head == nullptr) {

        throw new exception("Не удалось выделить память под хранение указателя на элемент списка\n");

    }

    cout << "Введите значения для"<<n<<" элемента списка: ";

    cin >> head->value;

    head->next = CreateList(head->next, n - 1);

    return head;

}

void ClearList(Node\* head)

{

    if (head->next != nullptr)

    {

        if (head->next->next != nullptr)

        {

            ClearList(head->next);

        }

        free(head->next);

    }

}

Node\* FindNode(Node\* current, int value)

{

    if (current)

    {

        if (current->value == value)

        {

            return current;

        }

        return FindNode(current->next, value);

    }

    return nullptr;

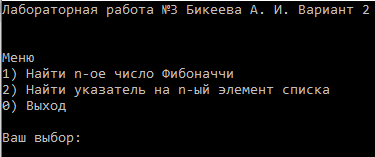
}

1. Задание 3. Разработать программу тестирования рекурсивных функций.
   1. Вариант задания

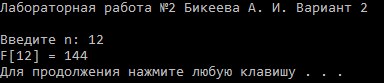
Разработать программу тестирования разработанных рекурсивных функций, которая управляется консольным меню.

* 1. Контрольные прогоны программы

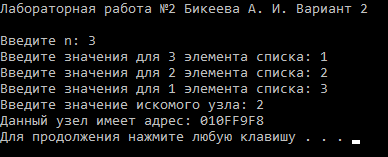
Меню



1 пункт



2 пункт



Вывод

В результате выполнения лабораторной работы были получены навыки по созданию и работе с рекурсивными функциями.