МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

Отчет

Практическая работа №6

Дисциплина Структуры и алгоритмы обработки данных

Тема. Рекурсивные алгоритмы и их реализация.

Выполнил студент Дамарад Д. В.
Фамилия И.О.

Группа ИКБО-13-21
Номер группы

Москва 2022

Вопросы

1) Определение рекурсивной функции:

Рекурсивная функция – функция, которая вызывает внутри своего тела себя же, но с другими исходными данными, в зависимости от условия рекурсии. Имеет 2 исхода – базовый и рекурсивный.

2) Шаг рекурсии:

Шаг рекурсии – это активизация очередного рекурсивного выполнения алгоритма при других исходных данных.

3) Глубина рекурсии:

Глубина рекурсии — это наибольшее одновременное количество рекурсивных вызовов функции, определяющее максимальное количество слоёв рекурсивного стека, в котором осуществляется хранение отложенных вычислений.

4) Условие завершения рекурсии:

Условие завершения рекурсии — это условие, которое определяет завершение рекурсии и формирование конкретного простейшего решения вычислительного процесса.

5) Виды рекурсии:

- линейная - это рекурсия, при которой каждый вызов порождает ровно один новый вызов;

- каскадная – это рекурсия, при которой каждый вызов порождает несколько новых вызовов.

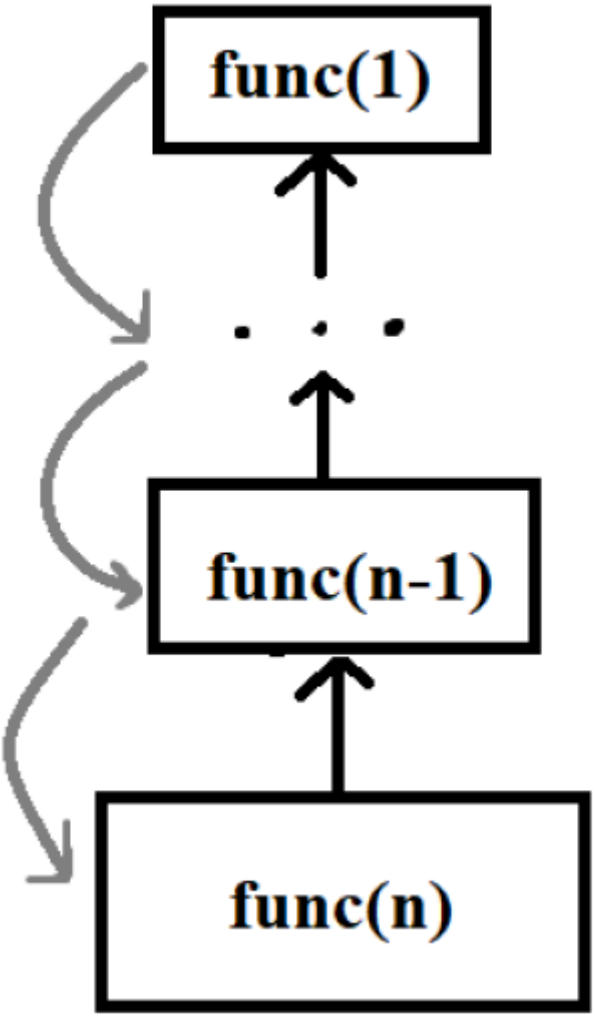
6) Прямая и косвенная рекурсия:

Прямая рекурсия имеет место, если решение задачи сводится к разделению её на меньшие подзадачи, выполняемые с помощью одного и того же алгоритма. Косвенная рекурсия имеет место, если алгоритм А вызывает алгоритм В, и алгоритм В вновь вызывает алгоритм А.

7) Организация стека рекурсивных вызовов:

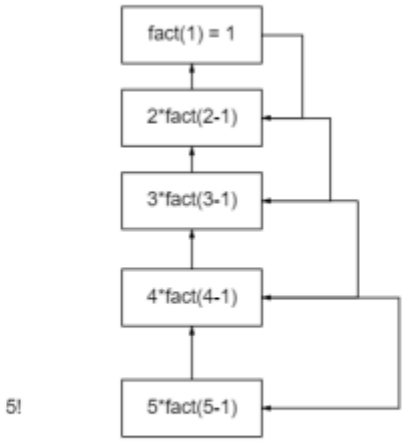
При каждом новом рекурсивном вызове функции в стеке создаётся новое множество локальных переменных и форменных параметров, их имена одинаковы, но они имеют различные значения.

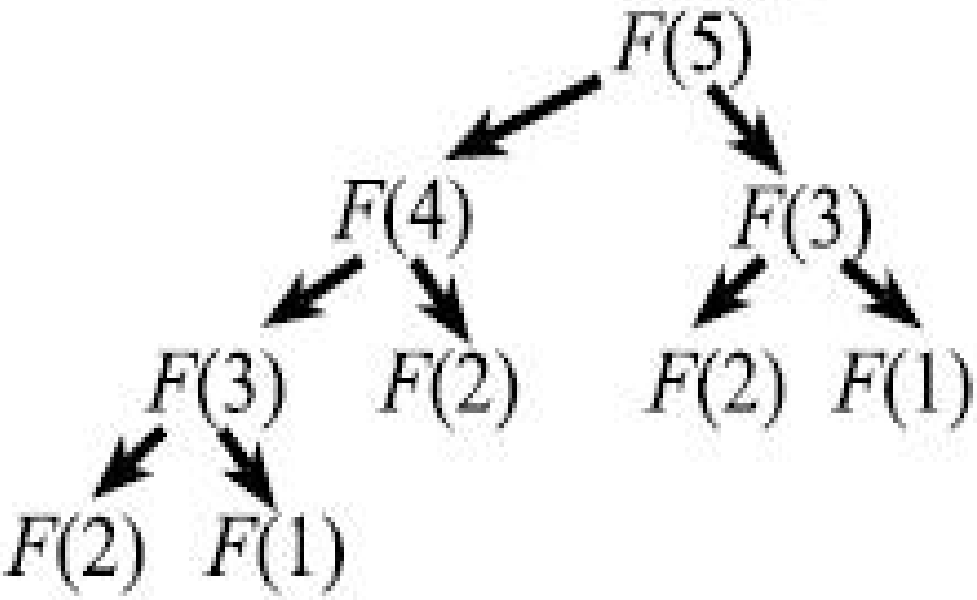
8) Стек рекурсивных вызовов. Модель формирования для алгоритма вычисления n!



Где стрелки чёрного цвета – выполняются в 1 очередь, а стрелки серого цвета – возвращают значение функции.

9) Дерево рекурсии вычисления 5! и пятого числа Фибоначчи.





1. Задача 1:

Дан массив из n элементов вещественного типа. Вычислить среднее значение всех элементов массива.

Декомпозиция:

Задачу следует разбить на следующие подзадачи:

- Ввод элементов массива.
- Вывод элементов массива.
- Вычисление среднего значения массива.

Определение функций:

1. Ввод элементов массива.

Предусловие: double* arr – массив, int n – размер массива.

Постусловие: заполненный массив.

void input_arr (double* arr, int n).

2. Вывод элементов массива.

Предусловие: double* arr – массив, int n – размер массива.

Постусловие: выведенный массив.

void print_arr(double* arr, int n).

3. Вычисление среднего значения массива.

Предусловие: double* arr – массив, int n – размер массива.

Постусловие: среднее значение массива.

double average(double* arr, int n)

Рекуррентная зависимость double average(double* arr, int n):

$$f(arr, n) = \begin{cases} 0 \ \ , \text{если } n = 0 \\ \text{arr[n } - \text{ 1]}, \text{если } n = 1 \\ \frac{\left((\text{n } - \text{ 1}) * \text{ f(arr, n } - \text{ 1}) + \text{ arr[n } - \text{ 1]}\right)}{n}, \text{если } n > 1 \end{cases}$$

2. Задача 2

Создание связанного стека из n элементов. Разработать рекурсивную функцию(функции) для обработки списковой структуры согласно варианту. Для создания списка может быть разработана простая или рекурсивная функция по желанию.

Структура узла:

```
struct Node {
  Node() {};
       Node(int data, Node* link = nullptr) {
          this->data = data;
          this->link = link;
       }
       int data=0;
       Node* link=nullptr;
    };
    Node* top;
         int size;
};
```

Декомпозиция:

Задачу следует разбить на следующие подзадачи:

- Вставить элемент в стек (рекурсивно).
- Вывод элементов стека.

1. Ввод элементов массива.

Предусловие: size – размер стека.

Постусловие: заполненный стек из size элементов.

void push(int size).

2. Вывод элементов стека.

Предусловие: стек с n элементами.

Постусловие: выведенный стек.

void print().

Рекуррентная зависимость void push(int size):

$$f(size) = \begin{cases} return \ \ , \text{если } size = 0 \\ \\ f(\text{size } - \ 1), \text{если } size > 0 \end{cases}$$

3. Код программы:

```
#include <iostream>
using namespace std;
template<typename T>
class stack {
public:
    stack() {
        size = 0;
        top = nullptr;
    }

    int GetSize() { return size; }

    void push(int size) {
        if (size != 0) {
            Node* temp;
            temp = new Node();
            cout << "Enter element: ";
            int info; cin >> info;
            temp->data = info;
            temp->link = top;
            top = temp;
            push(size - 1);

        }
        else {
            return;
        }
    }

    void print() {
        Node* temp;
        temp = top;
        cout << "Stack: " << endl;
        while (temp != NULL) {
            cout << temp->data << endl;
            temp = temp->link;
        }
    }

private:
    struct Node {
```

```
    public:
        Node() {};
        Node(int data, Node* link = nullptr) {
            this->data = data;
            this->link = link;
        }
        int data=0;
        Node* link = nullptr;
    };
    int      size;
    Node* top;
};

void input_arr(double* arr, int n) {
    if (n == 0) {
        cout << "Невозможно ввести элементы, т.к размер массива = 0!" <<
endl;
    }
    else {
        cout << "Введите элементы массива: ";
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
    }
}

void print_arr(double* arr, int n) {
    cout << "Массив: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

double average(double* arr, int n) {
    if (n == 0) {
        return 0;
    }
    else if (n == 1) {
        return arr[n - 1];
    }
    else {
        return ((n - 1) * average(arr, n - 1) + arr[n - 1]) / n;
    }
```

```
}

int main()
{
    setlocale(LC_ALL, "RUS");
    int menu;
    cout << "Введите номер задачи: " << endl;
    cout << "1-Вычисление среднего значения вещественного массива из n
элементов." << endl;
    cout << "2-Создание связанного стека из n элементов." << endl;
    cin >> menu;
    switch (menu) {
    case 1: {
        int size;
        cout << "Введите размер массива: ";
        cin >> size;
        double* arr = new double[size];
        if (size <= 0) {
            input_arr(arr, size);
            break;
        }
        else {
            input_arr(arr, size);
            print_arr(arr, size);
            cout << "Среднее значение массива = " << average(arr, size) <<
endl;;
            break;
        }
    }
    case 2: {
        int size;
        cout << "Enter stack size ";
        cin >> size;
        stack<int> stack;
        stack.push(size);
        if (size == 0) {
            cout << "Stack size = 0" << endl;
        }
        else {
            stack.print();
        }
        break;
    }
    default: {
```

```
            break;
        }
        }
        return 0;
    }
```

4. Результат тестирования программы:

1) Задача 1:

```
Введите номер задачи:
1-Вычисление среднего значения вещественного массива из n элементов.
2-Создание связанного стека из n элементов.
1
Введите размер массива: 5
Введите элементы массива: 1 2 3 4 5
Массив: 1 2 3 4 5
Среднее значение массива = 3
```

2) Задача 2:

```
Введите номер задачи:
1-Вычисление среднего значения вещественного массива из n элементов.
2-Создание связанного стека из n элементов.
2
Enter stack size 5
Enter element: 1
Enter element: 2
Enter element: 3
Enter element: 4
Enter element: 5
Stack:
5
4
3
2
1
```

5. Выводы:

В ходе проделанной работы были получены знания и практические навыки по разработке и реализации рекурсивных процессов.