Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

Рекурсия.

### Цели и задачи

* Научиться работать с рекурсией
* Познакомиться с рекурсеонным подходом решения задачи

### Ожидаемый результат

По результатам занятия Слушатель должен знать:

* Что такое рекурсия и какие у неё преимущества

### Структура занятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  | Таблица 1 |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

**Введение в рекурсию**

«Для того чтобы понять рекурсию, надо сначала понять рекурсию»

Если говорить просто, то рекурсия – это функция, которая сама вызывает себя. Но давайте попробую объяснить на примере.

Представьте, что вы пытаетесь открыть дверь в спальню, а она закрыта. Ваш трехлетний брат появляется из-за угла и говорит, что единственный ключ спрятан в коробке. Вы опаздываете на занятия и Вам действительно нужно попасть в комнату и взять вашу рубашку.

Вы открываете коробку только чтобы найти… еще больше коробок. Коробки внутри коробок и вы не знаете, в какой из них Ваш ключ. Вам срочно нужна рубашка, так что вам надо придумать хороший алгоритм и найти ключ.

Есть два основных подхода в создании алгоритма для решения данной проблемы: итеративный и рекурсивный. Вот блок-схемы этих подходов:

(см слайд 8)

Рекурсивная функция в языке С++ (и не только), это функция, которая вызывает саму себя.

**!Для избегания бесконечного вызова рекурсии, необходимо включить завершающее условие в функцию!**

**Пример рекурсии**

Алгоритм на слайде 10

**Стек вызовов**

Рекурсивные функции используют так называемый «Стек вызовов». Когда программа вызывает функцию, функция отправляется на верх стека вызовов. Это похоже на стопку книг, вы добавляете одну вещь за одни раз. Затем, когда вы готовы снять что-то обратно, вы всегда снимаете верхний элемент.

Для демонстрации работы рекурсии рассмотрим, создадим программу для подсчёта факториала.

В математике, термин факториал означает умножение всех положительных целых чисел, которые меньше, либо равны специальному неотрицательному (n)

Теперь, давайте посмотрим что же происходит, когда вы вызываете fact(3). Ниже приведена иллюстрация в которой шаг за шагом показано, что происходит в стеке. Самая верхняя коробка в стеке говорит Вам, что вызывать функции fact, на которой вы остановились в данный момент: (слайд 14)

**Divide and Conquer**

**(Разделяй и властвуй)**

Разделяй и властвуй в [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) — важная [парадигма](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) разработки [алгоритмов](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), заключающаяся в [рекурсивном](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F) разбиении решаемой задачи на две или более подзадачи того же типа, но меньшего размера, и комбинировании их решений для получения ответа к исходной задаче; разбиения выполняются до тех пор, пока все подзадачи не окажутся элементарными. Типичный пример — алгоритм [сортировки слиянием](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D0%B5%D0%BC). Чтобы отсортировать массив чисел по возрастанию, он разбивается на две равные части, каждая сортируется, затем отсортированные части сливаются в одну. Эта процедура применяется к каждой из частей до тех пор, пока сортируемая часть массива содержит хотя бы два элемента (чтобы можно было её разбить на две части). [Время работы](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0) этого алгоритма составляет **O(logn \* n)** операций, тогда как более простые алгоритмы требуют **O(n^2)** времени, где **n** — размер исходного массива.

**(Если хватает времени можете рассказать об оценке сложности алгоритмов (O-complexity) привести в пример сортировку пузырьком и вставками)**

**Практическая часть**

**Задание 1**

Написать программу для подсчёта факториала числа

(итерационно, т е с помощью циклов)

**Задание 2**

Вывести n-ое число Фибоначчи

(Файл с кодом лежит в папке code/)