Алгоритмы и структуры данных

БЕЛОЗУБОВ АЛЕКСАНДР ВЛАДИМИРОВИЧ

BELOZUBOV@ITMO.RU

Введение

Без знания структур данных и алгоритмов невозможно создать серьезный программный продукт. Ни одна информационная система не обходится без программного обеспечения, она просто не может существовать без этой компоненты. В связи с этим задача данного курса состоит в следующем:

- познакомить со всем разнообразием имеющихся структур данных, показать, как эти структуры реализованы в языках программирования;
- рассмотреть основные операции, которые выполняются над структурами данных;
- показать особенности структурного подхода к разработке алгоритмов, продемонстрировать порядок их разработки.

Основные этапы решения задач на ЭВМ

Решение задачи на ЭВМ сводится к выполнению программы, введённой в память ЭВМ. Решение задачи можно представить в виде следующей последовательности этапов:

- Математическая постановка задачи (формализация).
- Выбор или разработка метода решения (метод решения).
- Разработка алгоритма.
- Написание программы и подготовка её к вводу в ЭВМ (программа).
- Отладка программы и её выполнение (ЭВМ).

Основные этапы решения задач на ЭВМ

1. Математическая постановка задачи и её формализация состоит в определении и уточнении условий и цели задачи. Условие задачи – точное описание исходных данных их характеристик, а также ограничений , записываемых в математической форме.

Если задача математическая и известны метод её решения и соответствующие этому методу уравнения этап формализации задачи может не потребоваться.

Основные этапы решения задач на ЭВМ

2. Выбор или разработка метода решения тесно связан с этапом формализации, т. к. его цель состоит в сведении задачи к некоторой математической модели, для которой известен метод её решения. Но наиболее интересен случай разработки нового метода.

Основные этапы решения задач на ЭВМ

3. Разработка алгоритма. Алгоритмом называется система правил, чётко описывающая последовательность действий, которые необходимо выполнить для решения задачи.

Основные этапы решения задач на ЭВМ

4. Написание программы и подготовка её к вводу в ЭВМ. Цель этого этапа – запись алгоритма на выбранном языке программирования и переносе этого текста на носитель, с которого она может быть введена в ЭВМ. При выборе языка программирования необходимо пользоваться следующими принципами:

а) простота написания программы и сокращения сроков отладки;

б) эффективность: затрачиваемое машинное время и требуемый объём памяти.

Основные этапы решения задач на ЭВМ

5.Отладка программы и её выполнение. Цель – выявление и исправление ошибок в программе. Обычно на отладку программист затрачивает от 20 до 40% времени, отводимого на разработку программы.

Что такое алгоритм?

Алгоритм — это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.

Мухаммед ал-Хорезми
(ок. 783–ок. 850 гг.)

Исполнитель – это устройство или одушёвленное существо (человек), способное понять и выполнить команды, составляющие алгоритм.

Формальные исполнители: не понимают (и не могут понять) смысл команд.

Свойства алгоритма

Дискретность — алгоритм состоит из отдельных команд, каждая из которых выполняется за конечное время.

Детерминированность (определённость) — при каждом запуске алгоритма с одними и теми же исходными данными получается один и тот же результат.

Понятность — алгоритм содержит только команды, входящие в **систему команд исполнителя**.

Конечность (результативность) — для корректного набора данных алгоритм должен завершаться через конечное время.

Корректность — для допустимых исходных данных алгоритм должен приводить к правильному результату.

Способы записи алгоритмов

- **естественный язык**

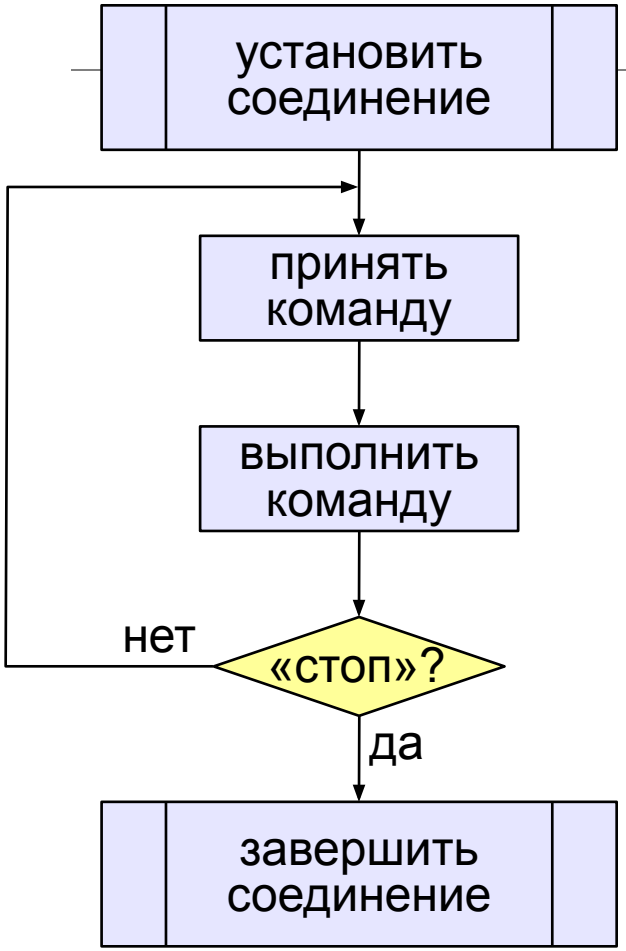
```
установить соединение
пока не принята команда «стоп»
   принять команду
   выполнить команду
завершить сеанс связи
```

- **псевдокод**

```
установить соединение
начало цикла
   принять команду
   выполнить команду
конец цикла при команда = 'stop'
завершить сеанс связи
```

Способы записи алгоритмов

- **блок-схема**



- **программа**

```
установитьСоединение
начало цикла
   cmd:= получитьКоманду
   выполнитьКоманду(cmd)
конец при cmd = 'stop'
закрытьСоединение
```

Структура алгоритмов

В 1969 году известным голландским ученым-программистом **Э. В. Дейкстрой** было доказано, что алгоритм для решения любой логической задачи можно составить только из структур **следование, ветвление, цикл.**

Их называют базовыми алгоритмическими структурами.

Методика программирования, основанная на этой теореме, называется **структурным программированием**.

Приёмы разработки алгоритмов

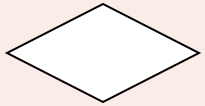
1. Метод пошаговой детализации – при котором первоначально продумывается и фиксируется общая структура алгоритма без детальной проработки отдельных его частей. При этом используются лишь основные структуры алгоритмов. Далее прорабатываются отдельные, не детализированные на предыдущем шаге. Полностью закончив детализацию всех блоков, получаем решение в целом.

2. Программирование «сверху вниз» - модуль может быть вызван только теми модулями, которые принадлежат более высокому уровню иерархии. Модульная программа похожа на конструкцию, собранною из блоков, каждый из которых имеет стандартные средства для связи с другими.

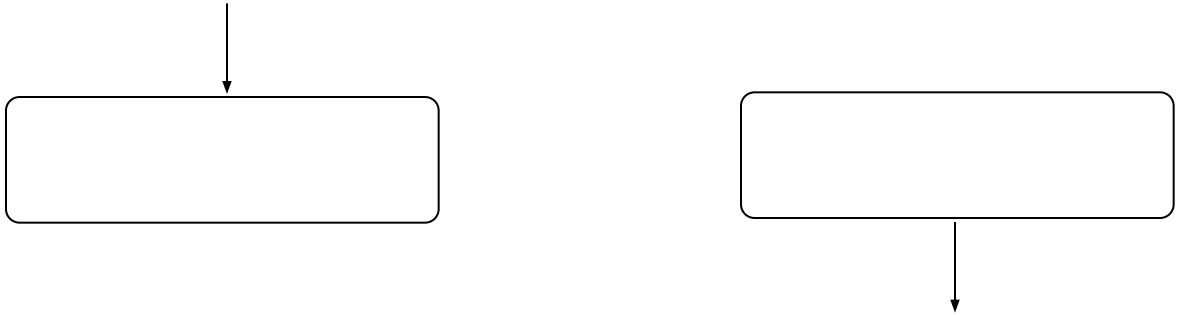
3. Метод разработки программ – «снизу вверх». В этом методе большое внимание уделяется представлению данных. По этой причине программист должен сначала заняться реализацией средств, а затем перейти к разработке структуры всей программы.

4. Метод «от центра по краям». Суть этого метода заключается в том, что сначала выделяется наиболее сложная часть проблемы и ищется её решение, а затем проводятся все оставшиеся работы. В процессе решения проблемы движение происходит как бы от центра по краям.

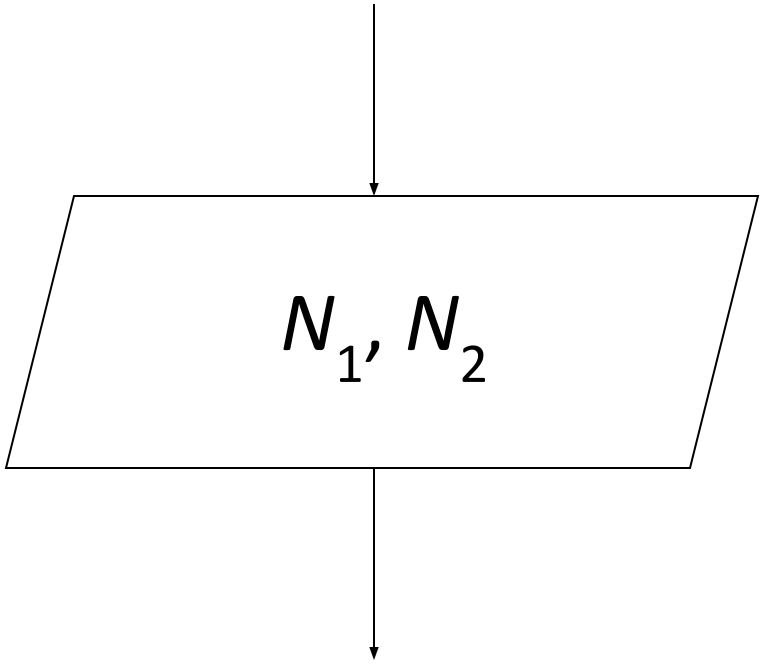
Изображение алгоритма в виде схемы

Процесс		Выполнение операции или группы операций
Решение (ветвление)		Выбор направления выполнения алгоритма или программы в зависимости от выполнения условий
Предопределённый процесс		Использование ранее созданных и отдельно описанных программ (алгоритмов).
Ввод-вывод		Преобразование данных в форму пригодную для обработки.
Линия потока		Линия потока
Соединитель		Указание на связь между прерванными линиями потока.

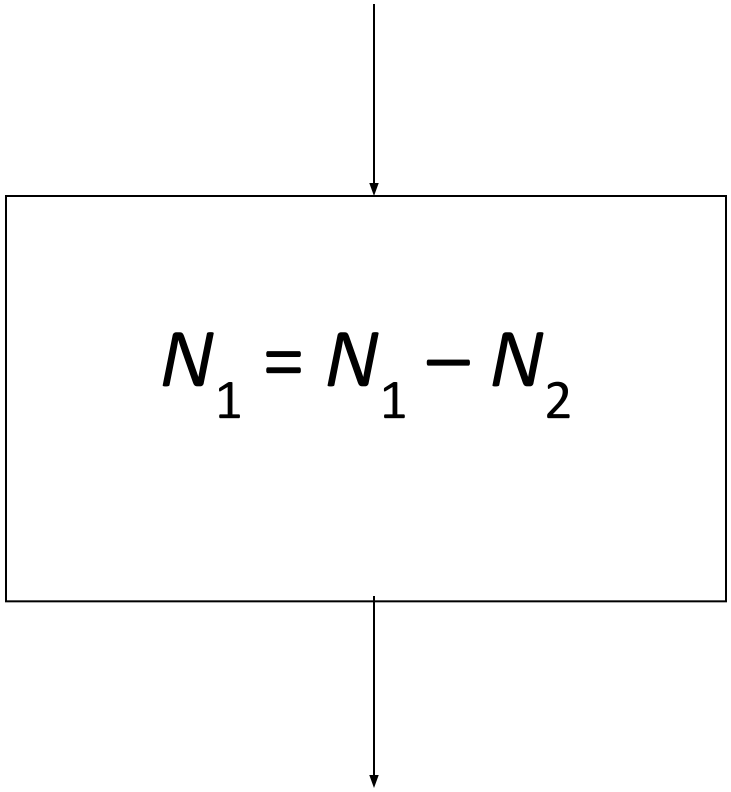
Пуск-останов



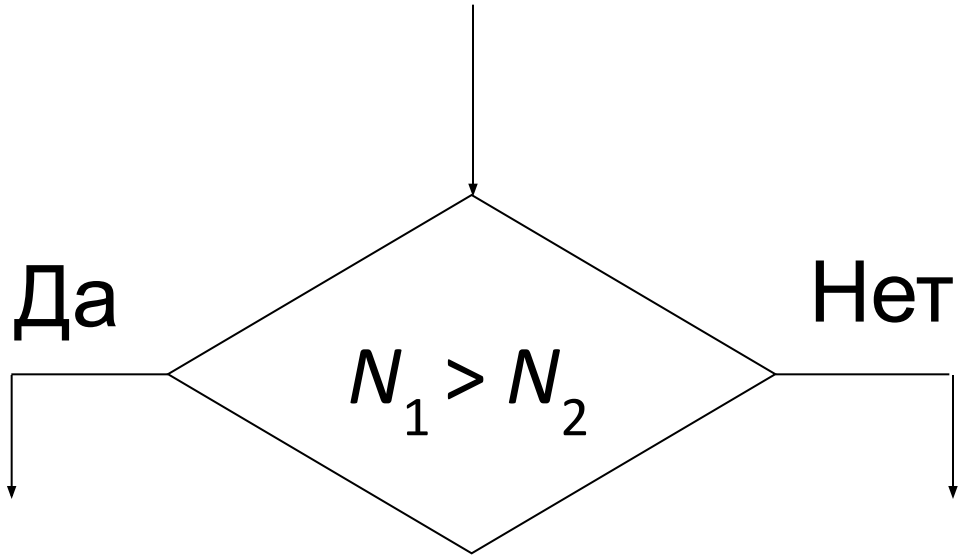
Блок ввода-вывода информации



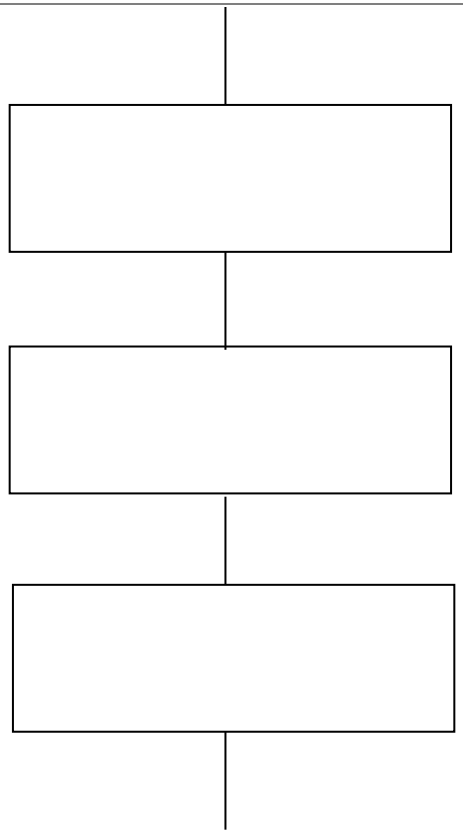
Процесс



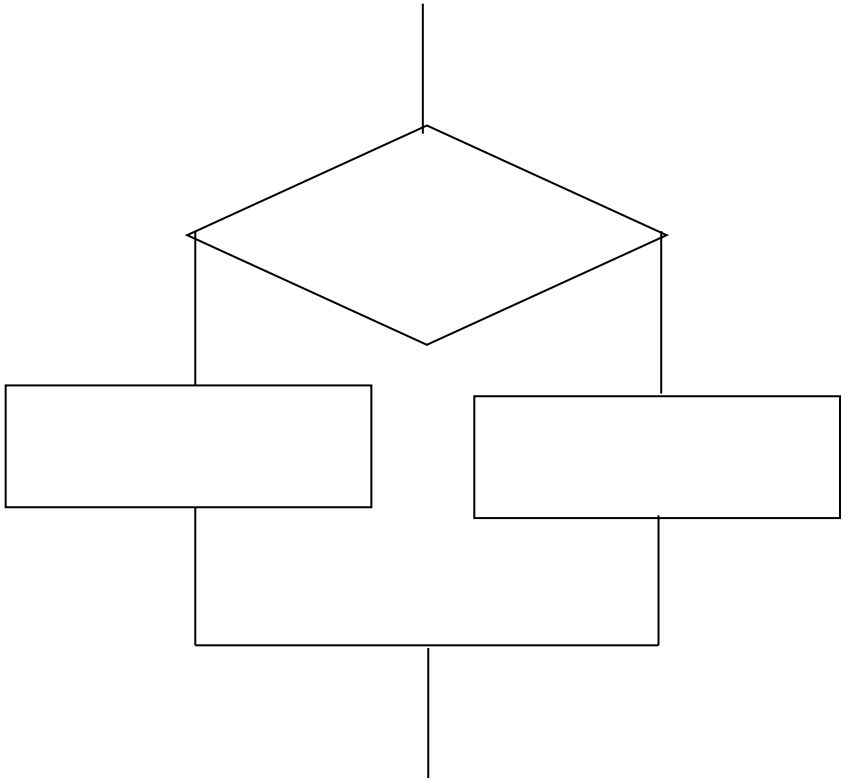
Решение



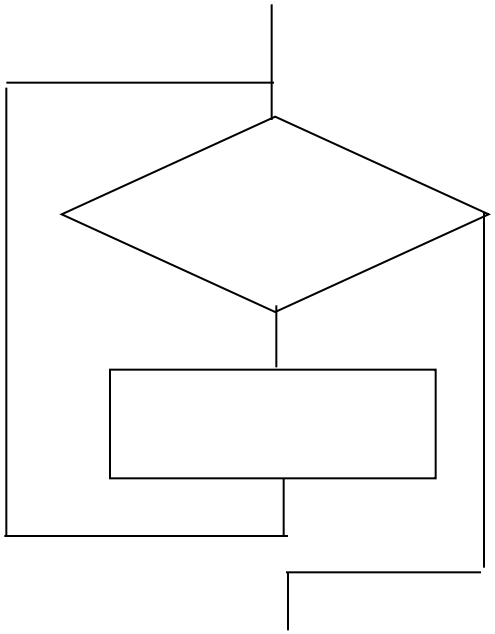
Следование



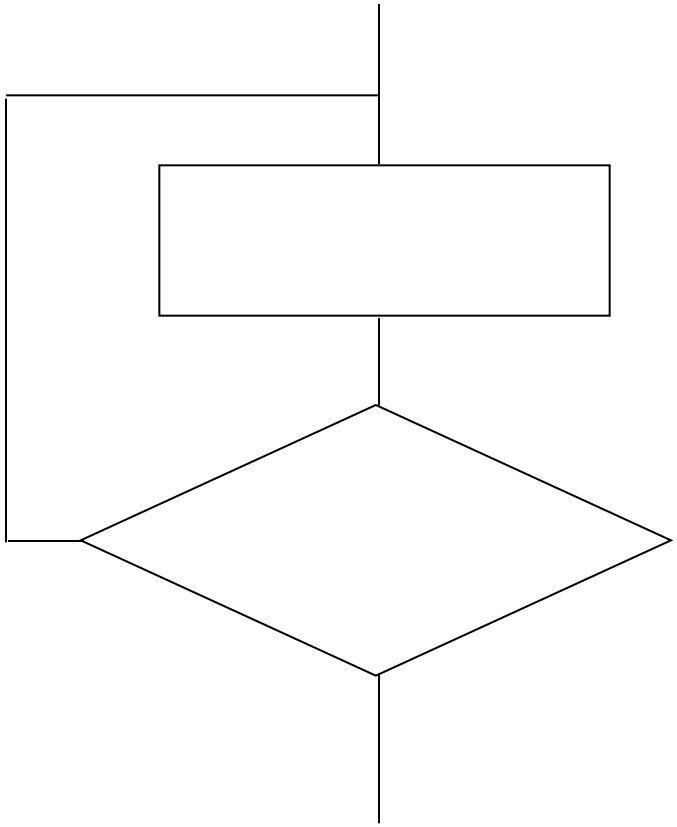
Ветвление



Цикл с предусловием



Цикл с постусловием

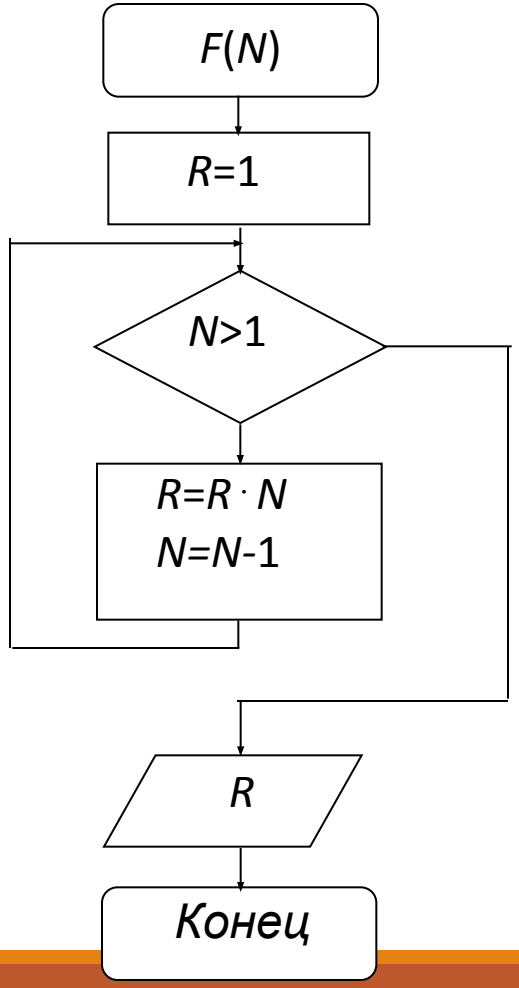


Итерационные и рекурсивные алгоритмы

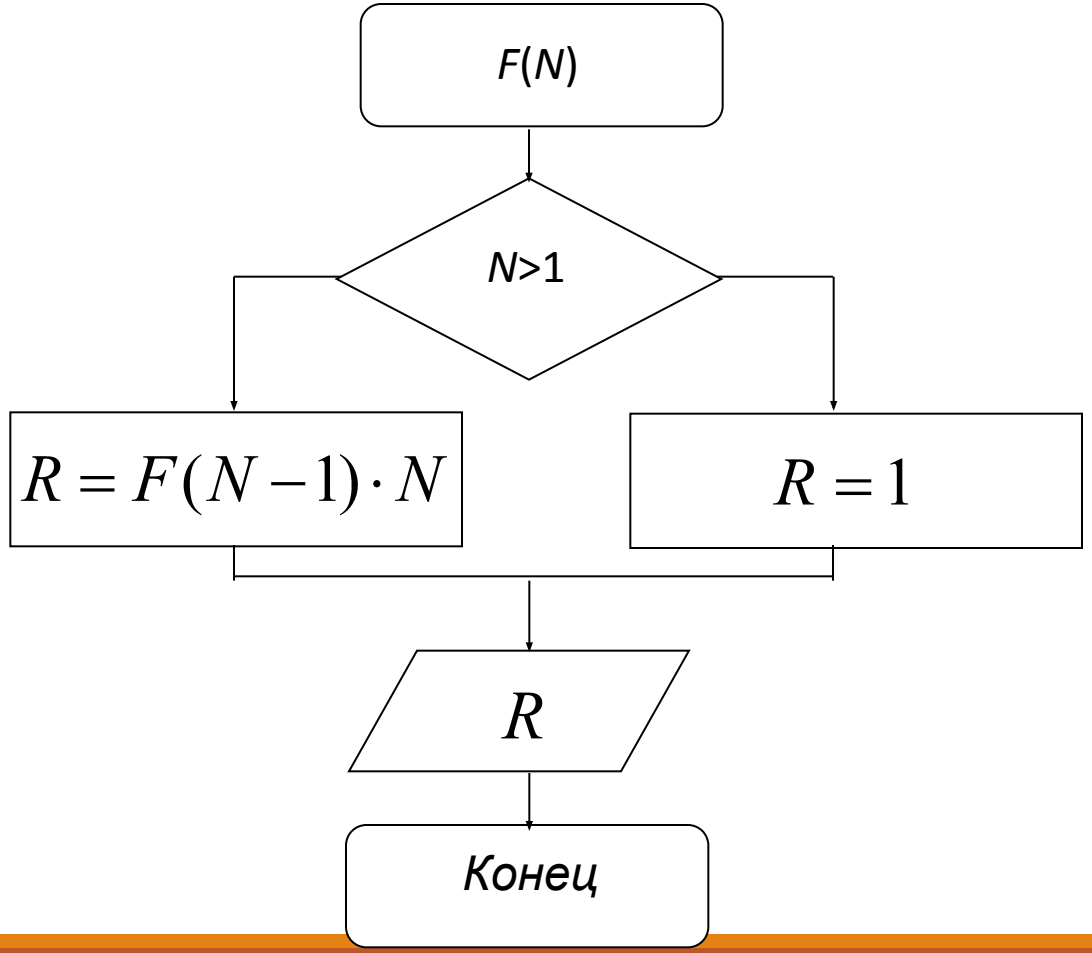
Алгоритм, в состав которого входит цикл, называется **итерационным** (от лат. iteratio - повторение).

Алгоритм называется **рекурсивным** (от лат. recursio - возвращение), если обращение к этому алгоритму может производится из него самого.

Итерационный алгоритм



Рекурсивный алгоритм



Псевдокод

Псевдокод - это система обозначений, предназначенная для неформального представления идей в процессе разработки алгоритмов.

Примеры:

- **if (***Условие***) then {***Действие***} else {***Действие***}**
- **while (***Условие***) do {***Действие***}**
- **do {***Действие***} while (***Условие***)**

Переменные и функции

Имя **:=** *Выражение*;

function *Имя_функции* (*Набор_входных_данных*) {

…

Фрагмент_псевдокода

…

return (*Набор_выходных_данных*);

}

Тип данных

Тип данных определяет:

- интерпретацию конкретных данных;
- операции, которые можно с ними выполнять.

К типам данных относятся **Integer** [целый], **Real** [действительный], **Character** [символьный] и **Boolean** [логический].

Массивы

Массив - это структура, содержащая в себе несколько однотипных элементов. Для упорядочивания элементов массива используются индексы. Индексы записываются в скобках после имени массива. Массив с одним индексом называется одномерным, с двумя - двумерным и т.д.

```
function Example() {
    M[0]:=1;
    M[1]:=1;
    N:=2
    while (N < 10) do {
        M[N]:=M[N-1]+M[N-2];
        N:=N+1;
    };
    return (M[]);
}
```

Записи

Запись - это структура, состоящая из элементов не обязательно одного типа. Отдельные элементы записи называют полями. Поле в свою очередь тоже может быть записью.

```
record Student (
    FirstName,
    LastName,
    Group
)
```

Списки

Список - это множество записей, каждая из которых содержит специальное поле - указатель (**Pointer**).

Виды списков:

односвязные;

двухсвязные;

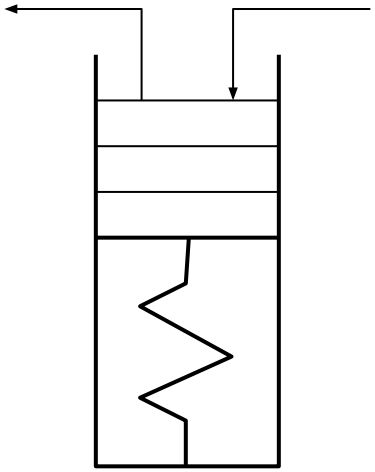
кольцевые.

Деревья

Дерево - это разветвленный список, каждая запись которого может содержать несколько указателей. Записи, входящие в дерево, называются **узлами**. Узлы, у которых все указатели пустые, называются **листьями**. Верхний, начальный узел дерева называется корневым узлом.

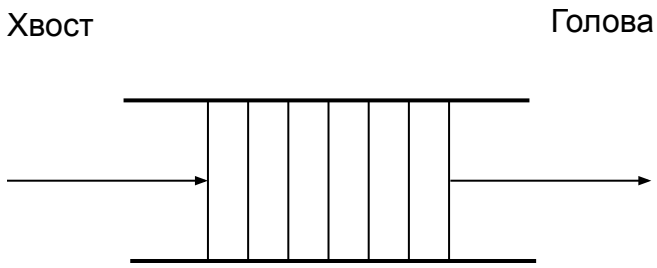
Стек

Стек - это структура данных, организованная по принципу "последним пришел - первым ушел" (Last In - First Out, LIFO).



Очередь

Очередь – это структура данных, организованная по принципу "первым пришел – первым ушел" (First In – First Out, FIFO).



Хэш-таблица

Хеширование - это метод, обеспечивающий прямой доступ к записям без использования каких-либо дополнительных структур.

Символ	ASCII-код
A	01000001
B	01000010
…	…
Z	01011010

Элемент массива
M[1]
M[2]
…
M[26]

Эффективность алгоритмов

Эффективность алгоритма принято оценивать количеством элементарных операций, например сравнений, которые необходимо выполнить для решения задачи, а также количеством памяти, которая требуется для выполнения алгоритма.

Анализ включает изучение ситуаций, в которых алгоритм демонстрирует свои наилучшие свойства, ситуаций, когда его эффективность минимальна, а также оценку его средней производительности.

Эффективность алгоритмов

Название алгоритма	Минимальное число сравнений	Среднее число сравнений	Максимальное число сравнений
Сортировка включением	n-1	$(n^2 + n - 2)/4$	$(n^2 - n)/2$
Сортировка выбором	$(n^2 - n)/2$	$(n^2 - n)/2$	$(n^2 - n)/2$
Обменная сортировка	$(n^2 - n)/2$	$(n^2 - n)/2$	$(n^2 - n)/2$
Последовательный поиск	1	$(n+1)/2$	n
Двоичный поиск	1	$(\log_2(n)+1)/2$	$\log_2(n)$