# АЛГОРИТМЫ В ПРОГРАММИРОВАНИИ.

ПОДГОТОВИЛ: УЧЕНИК 9 КЛАССА ХАЙНОВСКИЙ И.Г.

Роль алгоритмов в программировании. В настоящее время наибольшей популярностью пользуются системы объектно-ориентированного программирования (Visual Basic, Delphi). Разработка программы с помощью такой системы программирования состоит из двух этапов: 1) создание в визуальном режиме элементов графического интерфейса программы; 2) разработка программного кода. Такой подход существенно облегчает создание программ, так как разработка графического интерфейса вручную (в процедурных языках) сложный и трудоёмкий процесс.

Первый шаг к пониманию важности изучения и знания алгоритмов это дать точное определение тому, что понимается под алгоритмом. Алгоритм в программировании- это понятная и точная последовательность действий, записанных на языке программирования. Согласно популярной книге Алгоритмы: построение и анализ (Кормен, Лейзерсон, Ривест, Штайн)"алгоритм (algorithm) - это любая корректно определенная вычислительная процедура, на вход (input) которой подается некоторая величина или набор величин, и результатом выполнения которой является выходная (output) величина или набор значений". Другими словами, алгоритмы похожи на дорожные карты для достижения четко определенной цели. Код, для вычисления членов последовательности Фибоначчи - это реализация конкретного алгоритма. Даже простая функция сложения двух чисел является алгоритмом, хотя и простым.

# ДЛЯ СОЗДАНИЯ АЛГОРИТМА (ПРОГРАММЫ) НЕОБХОДИМО ЗНАТЬ:

- полный набор исходных данных задачи (начальное состояние объекта);
- цель создания алгоритма (конечное состояние объекта);
- систему команд исполнителя (то есть набор команд, которые исполнитель понимает и может выполнить).
- Полученный алгоритм (программа) должен обладать следующим набором свойств:
  - дискретность (алгоритм разбит на отдельные шаги - команды);
  - однозначность (каждая команда определяет единственно возможное действие исполнителя);
  - понятность (все команды алгоритма входят в систему команд исполнителя);
  - результативность (исполнитель должен решить задачу за конечное число шагов).

- Большая часть алгоритмов обладает также свойством  массовости  (с помощью одного и того же алгоритма можно решать множество однотипных задач).

| Приблизительное время выполнения алгоритмов, N = 100 | |
|---|---|
| O(Log(N)) | $10^{-7}$ секунд |
| O(N) | $10^{-6}$ секунд |
| O(N*Log(N)) | $10^{-5}$ секунд |
| $O(N^2)$ | $10^{-4}$ секунд |
| $O(N^6)$ | 3 минуты |
| $O(2^N)$ | $10^{14}$ лет |
| O(N!) | $10^{142}$ лет |

- Некоторые алгоритмы, к примеру, для вычисления последовательности Фибоначчи, являются интуитивно понятными и относятся к врожденным навыкам логического мышления и решения задач. Тем не менее, большинству из нас будет не лишним изучить и сложные алгоритмы, чтобы в будущем можно было использовать их в качестве строительных блоков для более эффективного решения логических задач. В действительности можно удивиться, узнав как много сложных алгоритмов используется людьми при проверке электронной почты или прослушивании музыки. В этой статье представлены некоторые основные идеи анализа алгоритмов с практическими примерами, иллюстрирующими важность изучения алгоритмов.

# ЯЗЫК ПРОГРАММИРОВАНИЯ - НАБОР ПРАВИЛ ЗАПИСИ АЛГОРИТМИЧЕСКИХ СТРУКТУР И ДАННЫХ.

- Одним из наиболее важных аспектов алгоритма является его скорость. Часто бывает легко придумать алгоритм решающий задачу, но если алгоритм слишком медленный, то он возвращается на доработку. Поскольку точная скорость алгоритма зависит от того где запускается алгоритм, а также деталей реализации, компьютерные специалисты обычно говорят о времени выполнения относительно входных данных. Например, если вход состоит из N целых чисел, то алгоритм может иметь время выполнения пропорциональное N2, что представляется как O(N2). Это означает, что если вы запустите реализацию алгоритма на компьютере с входом размером в N, то это займет C*N2 секунд, где C-некоторая константа, которая не меняется с изменением размера входа.

- Тем не менее, время выполнения многих сложных алгоритмов зависит не только от размера входных данных, но и от множества других факторов. Например, алгоритм сортировки множества целых чисел может работать намного быстрее, если это множество уже отсортировано. Принято говорить о наихудшем случае выполнения, и среднем случае выполнения. Наихудшее время выполнения - это максимальное время работы алгоритма при самом "плохом" из всех возможных входов. Средний случай выполнения - это среднее время работы алгоритма на всех возможных входах. Из этих двух типов времени выполнения, легче всего рассуждать о наихудшем случае и поэтому его используют чаще в качестве эталона для заданного алгоритма. Процесс определения наихудшего и среднего случая времени выполнения алгоритма может быть достаточно сложным, т.к. обычно невозможно запустить алгоритм для всех возможных входов.

# СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ

- Выше отмечалось, что один и тот же алгоритм может быть записан по-разному. Можно записывать алгоритм естественным языком. В таком виде мы используем рецепты, инструкции и т.п. Для записи алгоритмов, предназначенных формальным исполнителям, разработаны специальные языки программирования. Любой алгоритм можно описать графически в виде блок-схемы. Для этого разработана специальная система обозначений