**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт №8 «Компьютерные науки**

**и прикладная математика»**

**Реферат**

по теме «Эдсгер Вибе Дейкстра»

1 семестр

**Автор работы:**

студент 1 курса, гр. М8О-113Б-22

Горин Н. А.

**Руководитель проекта:**

Никулин С.П.

**Дата сдачи**



**Эдсгер Вибе Дейкстра**

**Биография**

Э́дсгер Ви́бе Де́йкстра (11 мая 1930— 6 августа 2002) — нидерландский учёный, труды которого оказали влияние на развитие информатики и информационных технологий; один из разработчиков концепции структурного программирования, исследователь формальной верификации и распределённых вычислений. Тьюринговский лауреат (1972).

Родился 11 мая 1930 года в Роттердаме, в семье учёных (отец — химик, мать — математик).

По окончании школы поступил на факультет теоретической физики Лейденского университета.

В 1951 году увлёкся программированием, поступил на трёхнедельные компьютерные курсы в Кембридже, с 1952 года работал программистом в Математическом центре Амстердама под руководством профессора Адриана ван Вейнгаардена, впоследствии — автора одного из способов формального описания грамматики формальных языков — так называемых двухуровневых грамматик ван Вейнгаардена.

Уже в 1952 году принял решение окончательно специализироваться на программировании, но всё же окончил курс теоретической физики.

В 1957 году женился, по собственным воспоминаниям, в графе «профессия» анкеты, которую положено заполнять при бракосочетании, написал «программист» — и его заставили переписывать документы, заявив, что такой профессии не существует, в результате пришлось указать «физик-теоретик».

В 1958—1960 годах принимал участие в разработке языка программирования Алгол, работал в команде по созданию компилятора языка; соревнуясь с датской командой Петера Наура, поклялся не бриться до завершения проекта и победил, написав компилятор за шесть недель, заодно изобретя новое правило компиляции — «вызов по имени».

В 1960-е годы участвовал в создании операционной системы THE, построенной в виде множества параллельно исполняющихся взаимодействующих процессов. Именно в ходе этой работы появились понятия синхронизации процессов, идея семафора, а также была чётко осознана необходимость в структуризации процесса программирования и самих программ.

Длительное время работал в компании Burroughs. В 1970-е годы вместе с Тони Хоаром и Никлаусом Виртом разработал основные положения структурного программирования.

В последние годы жизни преподавал в Техасском университете.

Умер 6 августа 2002 года после долгой борьбы с раком.

**Научная деятельность**

**Собственное определение информатики**

"informatics - научное направление, изучающее модели, методы и средства сбора, хранения, обработки и передачи информации - совокупность дисциплин естественно объединяющихся с целью семантической (смысловой) обработки информации."

**Алгоритм Дейкстры**

Свой известный алгоритм на графах Дейкстра придумал в 1959 году. Он находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS.

**Формулировка задачи:**

Дан взвешенный ориентированный граф G ( V , E ) G(V,E) без дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

**Неформальное объяснение алгоритма:**

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.

Работа алгоритма завершается, когда все вершины посещены.

**Инициализация.**

Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

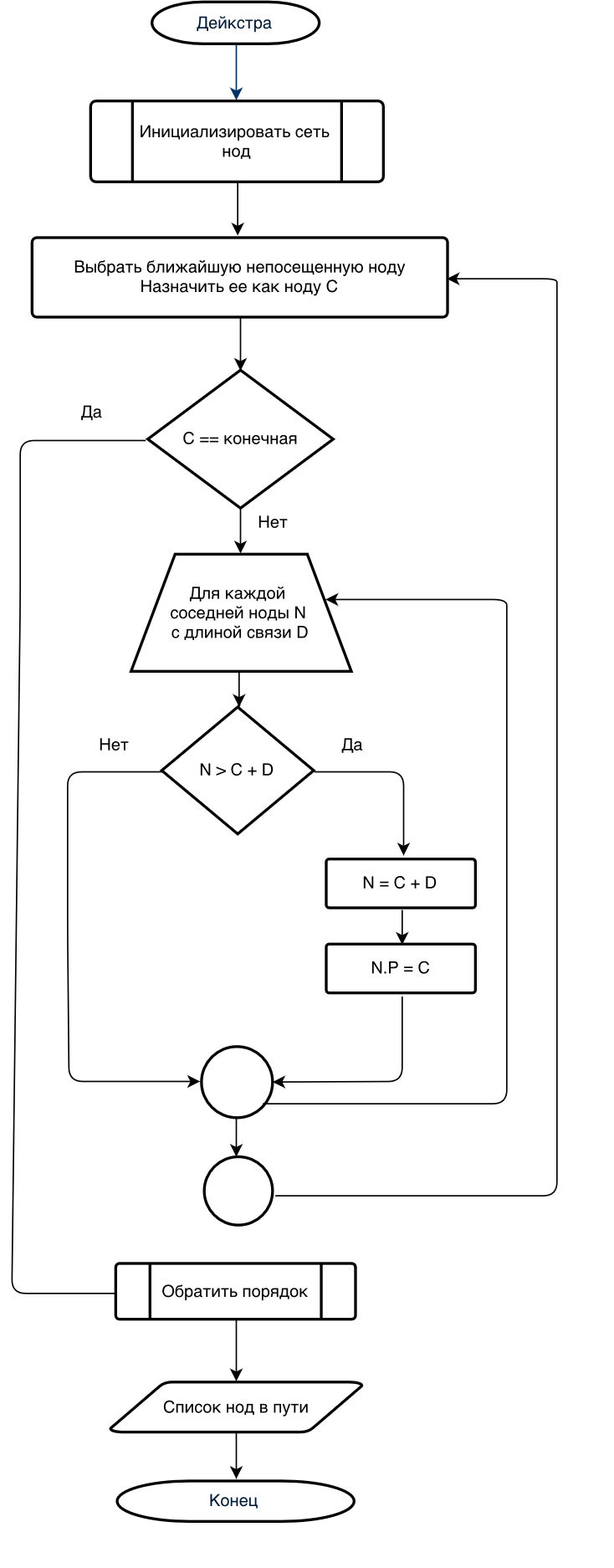
**Шаг алгоритма.**

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.



**Язык программирования Алгол**

Первое время Эдсгер Дейкстра работал на языках программирования низкого уровня. В этот период он и придумал знаменитый алгоритм поиска кратчайшего пути для вершин графов.

В 1957 году появился FORTRAN, Дейкстра перешел на него и… разочаровался. Позже, когда ученый стал преподавателем в университете, он все еще скептически относился к этому языку и даже говорил, что не будет учить студентов со знанием FORTRAN, ведь тот провоцирует развитие «дурных привычек программирования».

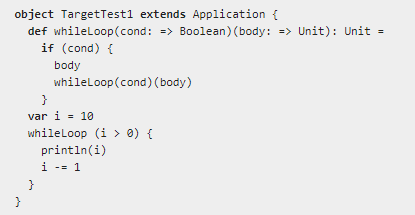
Так что когда в 1958 году на недельной конференции в Швейцарской высшей технической школе Цюриха великие умы собрались придумывать альтернативу FORTRAN — **Алгол**, Эдсгер Дейкстра был среди них.

Ученый работал над компилятором языка. И ему это удалось. Он назывался **«Алгол-60».**

**Изобретение call-by-name**

В рамках работы над компилятором, он изобрел новое правило компиляции – «вызов по имени», когда аргументы не перечисляются перед функцией, а подставляются непосредственно в ее тело.

Наглядно это можно продемонстрировать, например, в языке Scala:



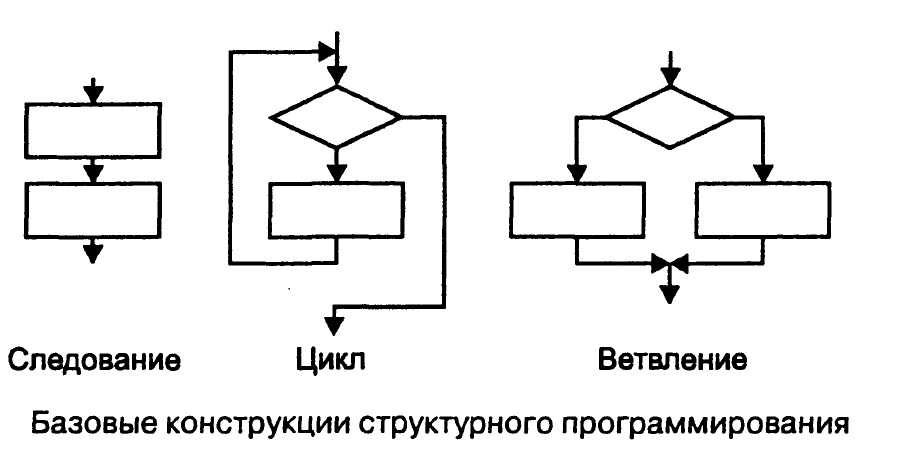
**Огромный вклад в концепцию структурного программирования. Противник goto.**

Структу́рное программи́рование — парадигма программирования, в основе которой лежит представление программы в виде иерархической структуры блоков.

Дейкстра ненавидел оператор goto, поскольку программы с ним при увеличении размеров становились все более запутанными и менее читаемыми, проще говоря, появлялся «спагетти-код».

В 1968 году Дейкстра в пух и прах разнес целесообразность использования goto в своей статье Goto considered harmful («Оператор Goto считается вредным»). В результате это привело к созданию другого подхода, где программа пишется пошагово и в ней есть ветвления, циклы и подпрограммы.

В соответствии с парадигмой, любая программа, которая строится без использования оператора goto, состоит из трёх базовых управляющих конструкций: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».



Методология структурного программирования появилась как следствие возрастания сложности решаемых на компьютерах задач, и соответственно, усложнения программного обеспечения. В 1970-е годы объёмы и сложность программ достигли такого уровня, что традиционная (неструктурированная) разработка программ перестала удовлетворять потребностям практики. Программы становились слишком сложными, чтобы их можно было нормально сопровождать. Поэтому потребовалась систематизация процесса разработки и структуры программ.

Методология структурной разработки программного обеспечения была признана «самой сильной формализацией 70-х годов».

По мнению Бертрана Мейера, «Революция во взглядах на программирование, начатая Дейкстрой, привела к движению, известному как структурное программирование, которое предложило систематический, рациональный подход к конструированию программ. Структурное программирование стало основой всего, что сделано в методологии программирования, включая и объектное программирование».

**Руководитель проекта разработки ОС THE**

THE (Technische Hogeschool Eindhoven) — компьютерная операционная система, разработанная командой под руководством Эдсгера В. Дейкстры, описанная в монографиях в 1965—1966 годах и опубликованная в 1968 году. Система была в основном пакетной системой, которая поддерживала многозадачность; она не была разработана как многопользовательская операционная система.

Система представила первые формы программной выгружаемой виртуальной памяти, освобождая программистов от необходимости использовать фактические физические местоположения в памяти на магнитном барабане.

Это было сделано с помощью модифицированного компилятора (единственный язык программирования, поддерживаемый системой Дейкстры), чтобы автоматически генерировать вызовы системных подпрограмм, которые убедились, что запрошенная информация находится в памяти, а при необходимости меняли местами. Выгружаемая виртуальная память также использовалась для буферизации данных устройств ввода-вывода и значительной части кода операционной системы, а также почти всего компилятора Алгол 60. В самой этой системе семафоры впервые использовались в качестве программной конструкции.

**Понятие семафора**

**Семафор** - примитив синхронизации работы процессов и потоков, в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующейся. Служит для построения более сложных механизмов синхронизации и используется для синхронизации параллельно работающих задач, для защиты передачи данных через разделяемую память, для защиты критических секций, а также для управления доступом к аппаратному обеспечению.

Понятие семафора было введено в 1965 году Эдсгером Дейкстрой, а в 1968 году он предложил использовать два семафора для решения задачи производителя и потребителя.

Семафор представляет собой счётчик, над которым можно выполнять две операции: увеличение на 1 (англ. up) и уменьшение на 1 (англ. down). При попытке уменьшения семафора, значение которого равно нулю, задача, запросившая данное действие, должна блокироваться до тех пор, пока не станет возможным уменьшение значения семафора до неотрицательного значения, то есть пока другой процесс не увеличит значение семафора. Под блокированием задачи понимается изменение состояния процесса или потока планировщиком задач на такое, при котором задача приостановит своё исполнение

В общем виде семафор можно представить как объект, состоящий из:

* переменной-счётчика, хранящей текущее значение семафора;
* списка заблокированных в ожидании сигнального значения семафора задач;
* функций атомарного увеличения и уменьшения значения семафора.

Концепция семафора хорошо подходит для синхронизации потоков, может использоваться для синхронизации процессов, однако совершенно не подходит для синхронизации взаимодействия компьютеров. Семафор является низкоуровневым примитивом синхронизации, поэтому, за исключением защиты критических секций, сам по себе может быть сложен в использовании.

**Библиография**

Автор нескольких книг и множества статей, самые известные публикации — книги «Дисциплина программирования», «Заметки по структурному программированию», статья «О вреде оператора GOTO».

Помимо обсуждения специальных вопросов, в своих статьях и книгах Дейкстра последовательно отстаивал необходимость математического подхода к программированию, который предполагает предварительное точное, всестороннее математическое описание задачи и способа её решения, формальное доказательство правильности выбранного алгоритма и последующую реализацию алгоритма в виде максимально простой, структурированной программы, корректность которой должна быть формально доказана.

По мнению Дейкстры, господствующий в компьютерной индустрии подход к программированию как к процессу достижения результата методом проб и ошибок («написать код — протестировать — найти ошибки — исправить — протестировать — …») порочен, поскольку стимулирует программистов не думать над задачей, а писать код, что при этом совершенно не гарантирует корректность программ, которая не может быть доказана тестированием в принципе.

Многократно предостерегал от попыток превратить разработку программ в некий тривиальный процесс; по его мнению, программирование в сути своей — чрезвычайно сложная научная и инженерная деятельность, и никакие новые методы и инструменты не смогут кардинально изменить это положение — они лишь освобождают программиста от части рутинной работы. Попытки же превратить программирование в простое занятие, доступное каждому, обречены на провал.

В 1975 году на примере положения дел в Германии Дейкстра показал, что развитие программирования как науки, опираясь на один выбранный язык программирования, невозможно.

Результатом такого подхода стал полный разрыв теории и практики программирования. Дейкстра отмечал, что принятие в Германии на правительственном уровне языка ALGOL 68 в качестве фундаментального средства дальнейшего развития имело столь же парализующее действие, как и решение правительства СССР о переходе советской промышленности к копированию модельного ряда IBM/360 в конце 1960 годов, которое учёный назвал величайшей победой Запада в холодной войне.

Факты и цитаты:

* Известно, что он мало заинтересован в приеме на старшие курсы университета, где он преподает, студентов со знанием Фортрана по той причине, что вместе с этими знаниями могли привиться дурные привычки программировании.
* «Для хорошей программы важно выбрать правильный и логичный алгоритм»
* «Решение задачи должно опираться на ее математическое описание»
* «Основная работа программиста — продумать решение, а не писать код»
* «Изящество, ясность и тому подобное в значительной степени определяются количественными аспектами. (Этим владел Моцарт: многие его произведения, от которых замирает дыхание, обманчиво просты; кажется, будто они созданы практически из ничего!)»

**Список использованных источников**

* <https://ru.wikipedia.org>
* <https://www.peoples.ru/science/mathematics/edsger_wybe_dijkstra/>
* <http://db4.sbras.ru/elbib/data/show_page.phtml?76+5>
* <https://highload.today/ne-nravitsya-yazyk-programmirovaniya-sozdaj-svoj-kak-edsger-dejkstra-razveival-mif-chto-programmirovanie-eto-koldovstvo/>
* <https://www.altstu.ru/media/f/Tema-20-Strukturnoe-programmirovanie.pdf>