Основы алгоритмизации и программирования

Пашук Александр Владимирович

pashuk@bsuir.by

Организационные моменты

1. Формат лекций
2. Формат экзамена
3. Бонусы:

   - конспект

   - посещения

О чем этот курс?

1. Общие сведения об алгоритмах
2. Основные элементы языка C++
3. Сложные типы данных (массивы)
4. Распределения памяти, указатели
5. Функции
6. Алгоритмы поиска и сортировки
7. Работа со строками
8. Работа с файлами

Литература

C++:

- Шилдт Г. Теория и практика C++
- Страуструп Б. Язык программирования C++: специальное издание

Алгоритмы:

- Томас Кормен и др. Алгоритмы. Построение и анализ
- Роберт Седжвик. Алгоритмы на C++
- Кнут, Д.Э. Искусство программирования

Содержание лекции

1. Алгоритмы и их свойства
2. Способы записи алгоритмов
3. Базовые алгоритмические структуры
4. Величины в алгоритмах
5. Примеры

Этапы решения задач на ЭВМ

1. Постановка задачи
2. Формализация (математическая постановка)
3. Выбор (или разработка) метода решения
4. Разработка алгоритма.
5. Составление программы
6. Отладка программы
7. Вычисление и обработка результатов.

Постановка задачи

- определить цель и категорию программы (системная, прикладная)
- определить исходные данные и требуемый результат
- проверить, является ли задача хорошо поставленной (должны быть определены все связи между исходными данными и результатом)
- зафиксировать требования к программе в письменной форме

Формализация

- все объекты задачи описываются на языке математики
- выбираются математические абстракции, адекватно, то есть с требуемой точностью и полнотой, представляющие исходные данные и результаты
- выбирается форма хранения данных
- составляются все необходимые формулы

Выбор метода решения. Разработка алгоритма

- анализ существующих методов решения поставленной задачи
- разработка нового метода решения (при необходимости) или определение метода решения задачи – метод преобразования исходных данных в результат
- при необходимости возможен возврат к предыдущему этапы для уточнения моделей данных

Составление программы

- преобразование алгоритма в код, написанным на языке программирования (C, C++, Java, Python etc.)

Вычисление и обработка результатов

- анализ полученных результатов
- внесение правок в алгоритм и/или код программы
- отладка программы

Что такое алгоритм?

Слово «алгоритм» произошло от латинского написания имени ученого из города Хорезма, Абдуллы (или абу Джафара) Мухаммеда бен Муса аль-Хорезми (*Alhorithmi*), жившего в 783 – 850 гг.

Что такое алгоритм?

- **Алгоритм** – это подробное описание последовательности арифметических и логических действий, расположенных в строгом логическом порядке и позволяющих решить конкретную задачу.

- **Шаг алгоритма** – это каждое отдельное действие алгоритма

- **Алгоритмизация** – составление пошагового описания процесса решения задачи.

Свойства алгоритмов

- **Определенность** (детерминированность, точность) – единственность толкования правил выполнения действий и порядка их выполнения;
- **Конечность** – обязательность завершения каждого из действий алгоритма и алгоритма в целом;
- **Результативность** – обязательность получения через определенное число шагов определенных результатов или сообщения о невозможности решения;
- **Массовость** – возможность применения одного и того алгоритма для решения однотипных задач с различными исходными данными;
- **Дискретность** – расчленение вычислительного процесса на отдельные этапы, элементарные операции.

Алгоритмы

Иногда также выделяют дополнительные свойства:

- Эффективность
- Связанность

Каждый алгоритм имеет **вход** и **выход**. Вход алгоритма – это совокупность его исходных данных. Множество допустимых значений переменных на входе алгоритма называются **областью определения алгоритма**.

Выход алгоритма – это совокупность результатов его работы.

Пример задачи

1. **Постановка задачи:**

Разработать алгоритм нахождения наименьшего простого делителя натурального числа k, большего единицы.

2. **Формализация:**

Входные данные: k – натуральное число, *k*>1;

Выходные данные: i – наименьший простой делитель числа *k, i*≠1.

Пример задачи

3. **Метод решения:**

Простым называется число, не имеющее делителей, отличных от единицы и его самого, причем единица во множество простых чисел не входит.

4. **Алгоритм решения задачи:**

1. Положить целое число i равным двум и перейти на шаг 2.
2. Если k делится нацело на i, то завершить работу алгоритма, выдав в качестве результата i, иначе перейти на шаг 3.
3. Увеличить значение i на единицу и перейти на шаг 2.

Пример задачи

Детерминированность. Все действия определены строго и недвусмысленно.

Конечность. Величина i вначале меньше k, ее значение увеличивается на единицу к каждому очередному выполнению шага 2, и поэтому выполнение алгоритма будет прекращено на шаге 2 при i=k, если k – простое число, или ранее при составном k.

Результативность и **массовость**. Этот алгоритм пригоден для нахождения наименьшего простого делителя любого натурального числа k, большего единицы.

Дискретность. Процесс разбит на отдельные этапы, возможность выполнения которых исполнителем (компьютером) не вызывает сомнений.

Способы представления алгоритмов

1. **Словесный** – описание алгоритмов на естественном языке;
2. **Структурно-стилизованный способ** (псевдокоды);
3. **Графический** (с помощью блок-схем);
4. **Программный**.

Словесное описание

- Этап обработки(вычисления)

  V=выражение

  Где V – переменная

- Проверка условия

  Если условие, то идти к N

- Переход к этапу с номером N

  Идти к N

- Конец вычислений

  Конец

Словесное описание: пример

- Дать словесное описание алгоритма решения квадратного уравнения $a \cdot x^2 + b \cdot x + c = 0$

1. $\mathrm{D} = \mathrm{b}^2 - 4 \cdot \mathrm{a} \cdot \mathrm{c}$
2. Если $\mathrm{D} < 0$, идти к 4
3. $\mathrm{x}_1 = (-\mathrm{b} + \sqrt{\mathrm{D}})/(2 \cdot \mathrm{a})$
   $\mathrm{x}_2 = (-\mathrm{b} - \sqrt{\mathrm{D}})/(2 \cdot \mathrm{a})$
4. Вывод данных
5. Конец

Недостаток метода: малая наглядность

Недостатки

- отсутствует наглядность вычислительного процесса, т.к. нет достаточной формализации;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Псевдокод

Псевдокод - позволяет формально изображать логику программы, не заботясь при этом о синтаксических особенностях конкретного языка программирования.

Обычно представляет собой смесь операторов языка программирования и естественного языка.

Является средством представления логики программы, которое можно применять вместо блок-схемы.

Пример

Выбираем первый элемент (i=1)

IF A>xi или xi<B **THEN**

печать сообщения и переход на конец

ELSE переход к следующему элементу(i=i+1)

IF массив не кончился (i<=n) **THEN**

переход на проверку интервала

ELSE печать сообщения, что все элементы входят

в интервал

Конец

Графический способ

Блок-схема – это графическая интерпретация алгоритма, представляющая набор геометрических фигур, каждая из которых изображает какую-либо операцию или действие.

Внутри фигур (блоков) кратко записывают выполняемое действие. Такую схему называют схемой программы. Схема программы – отображает последовательность операций в программе.

Графический способ

Формат блок-схем стандартизирован:

- ГОСТ 19.701-90,
- ISO 5807-85 "Схемы алгоритмов, данных, программ и систем"

Графический способ

4 основных правила:

1. Блок-схема строится сверху вниз.
2. В любой блок-схеме имеется только один элемент, соответствующий началу алгоритма, и один элемент, соответствующий концу алгоритма.
3. Должен быть хотя бы один путь из начала блок-схемы к любому элементу.
4. Должен быть хотя бы один путь от каждого элемента блок-схемы в конец блок-схемы.

Графический способ. Блоки

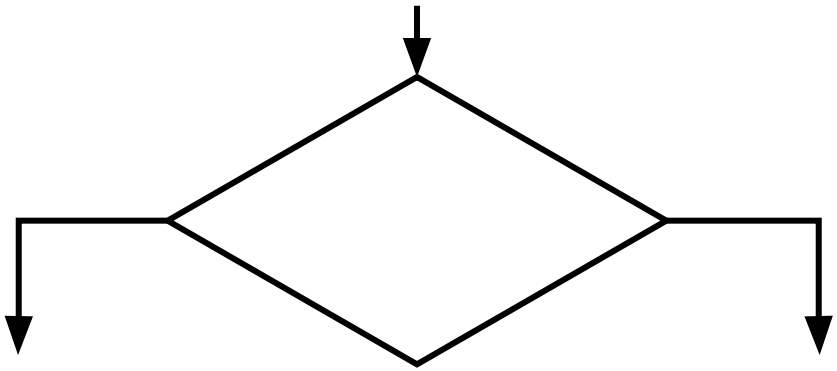
1. Начало и конец вычислительного процесса

2. Блок ввода и вывода

Графический способ. Блоки

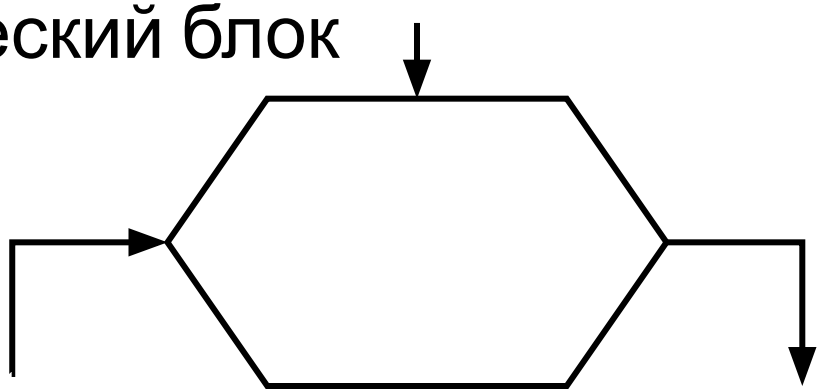
3. Вычислительный блок

4. Логический блок

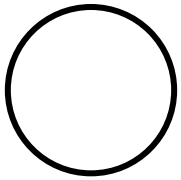


Графический способ. Блоки

5. Циклический блок



6. Соединитель

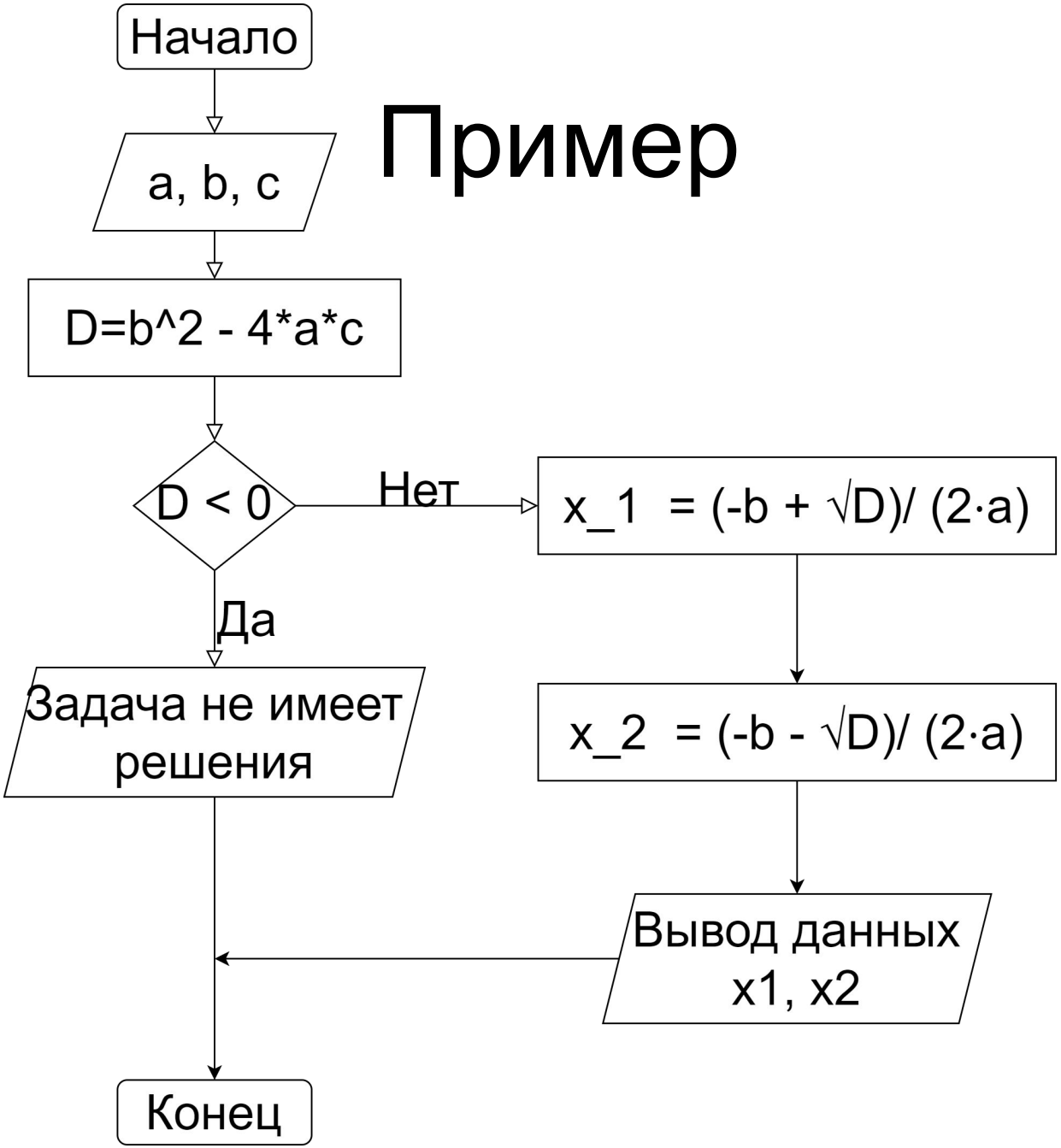


Пример

- $a \cdot x^2 + b \cdot x + c = 0\ (a \neq 0)$:

1. Начало
2. Ввод данных a, b, c
3. Вычислить D = b2 – 4·a·c
4. Если D < 0, то задача не имеет решения, перейти к 8
5. $x_1 = (-b + \sqrt{D})/(2 \cdot a)$
6. $x_2 = (-b - \sqrt{D})/(2 \cdot a)$
7. Вывод данных x_1, x_2
8. Конец

Пример


Начало
a, b, c
D=b^2 - 4*a*c
D < 0
Нет
Да
x_1 = (-b + √D)/ (2·a)
x_2 = (-b - √D)/ (2·a)
Задача не имеет решения
Вывод данных x1, x2
Конец

Основные структуры алгоритмов

Основные структуры алгоритмов – это ограниченный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.

Классификация алгоритмов по структуре:

- Линейный
- Разветвленный (ветвление)
- Циклический (повтор)

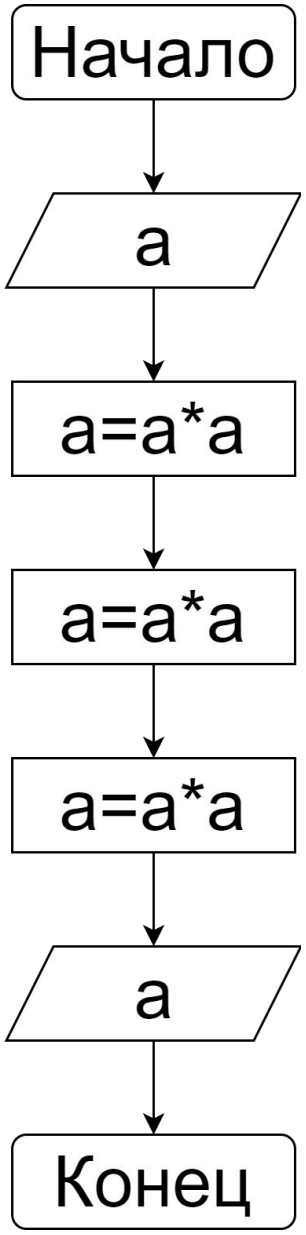
Виды алгоритмов

Алгоритм линейной структуры - это алгоритм, в котором направление вычислений является единственным.

- В алгоритмах с линейной структурой каждый оператор выполняется только одни раз и выполняется весь целиком, т.е. выполняются все ее операторы.
- В них могут использоваться все блоки, за исключением блоков проверки условия и модификации.

Пример

- Дано действительное число a. Не пользуясь никакими операциями, кроме умножения, получить a^8 за три операции.



Виды алгоритмов

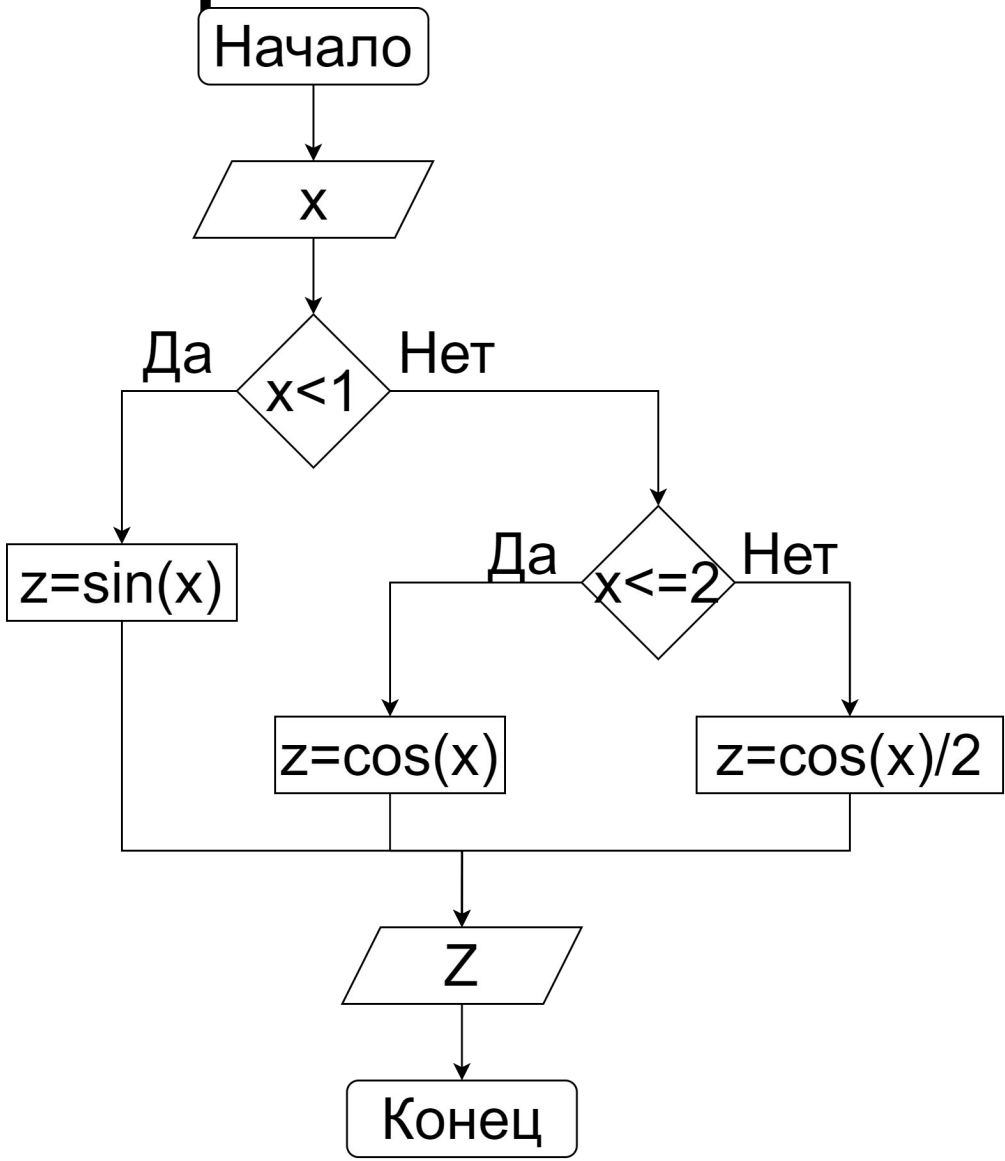
Алгоритм разветвляющейся структуры - это алгоритм, в котором направление вычислений определяется некоторыми условиями

- Каждое возможное направление вычислений называется ветвью.
- Каждая из ветвей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.
- Структура ветвление существует в трех основных вариантах: если–то (неполная форма ветвления); если–то–иначе (полная форма ветвления); выбор.

Пример

- Вычислить значение функции Z по значению x.

$$z = \begin{cases} \sin x, x < 1 \\ \cos x, 1 \le x \le 2 \\ \dfrac{\cos x}{2}, x > 2 \end{cases}$$

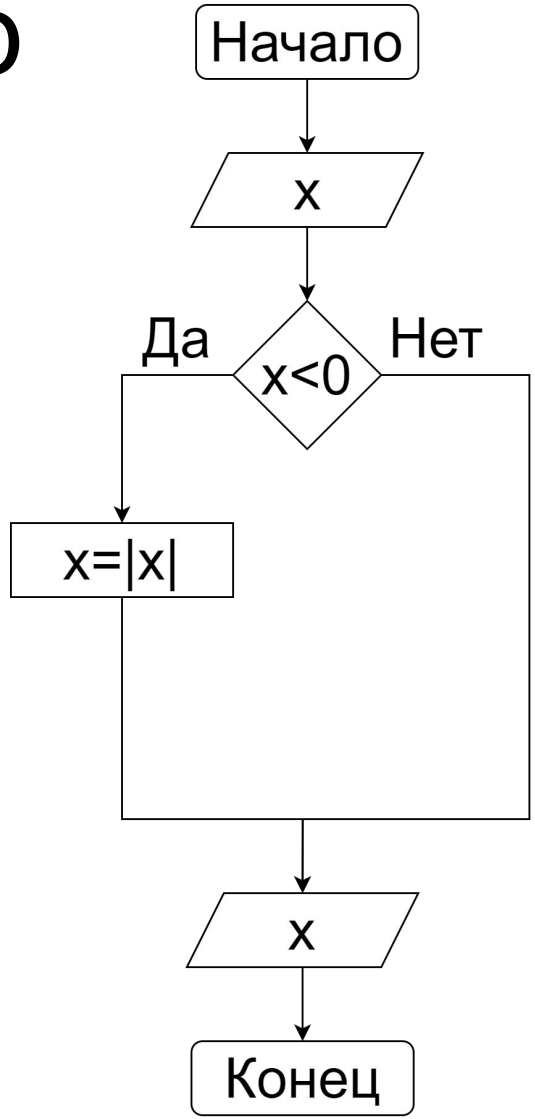


Пример


Начало
a, b, c
D=b^2 - 4*a*c
D < 0
Нет
Да
Задача не имеет решения
D == 0
Нет
Да
x_1 = (-b + √D)/ (2·a)
x_1= x_2 = -b / (2·a)
x_2 = (-b - √D)/ (2·a)
Вывод данных x1, x2
Конец

Пример

Вывести модуль действительного числа x.



Виды алгоритмов

Алгоритм циклической структуры - это алгоритм, в котором отдельные участки вычислений выполняются многократно

- Циклические алгоритмы позволяют существенно сократить объем программы за счет многократного выполнения группы повторяющихся вычислений

Циклы: основные понятия

Цикл – совокупность действий алгоритма, связанная с повторением.

Тело цикла – многократно повторяющиеся действия алгоритма.

Параметр цикла – величина, с изменением которой связано многократное выполнение цикла.

Условие цикла – это некоторое утверждение, которое обязательно принимает одно из значений: истина или ложь.

Циклы

Для организации цикла необходимо выполнить следующие действия:

1. Перед началом цикла задать начальные значения параметров (переменных, используемых в логическом выражении, отвечающем за продолжение или завершение цикла;
2. Внутри цикла изменять переменную (или переменные), которая сменит значение логического выражения, за счет которого продолжается цикл, на противоположное;
3. Вычислять логическое выражение – проверять условие продолжения или окончания цикла;
4. Управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из цикла в противном случае.

Классификация циклов

- **Простые** – циклы, не содержащие внутри себя другие циклы
- **Сложные** - циклы, содержащие внутри себя другие циклы
- **Вложенные (внутренние)** – циклы, входящие в состав других циклов (цикл в цикле)
- **Внешние** – циклы, не являющиеся составной частью других циклов, но содержащие в своем составе внутренние циклы.

Классификация циклов

В зависимости от месторасположения условия выполнения цикла:

- циклы с предусловием (ПОКА)
- циклы с постусловием (ДО)

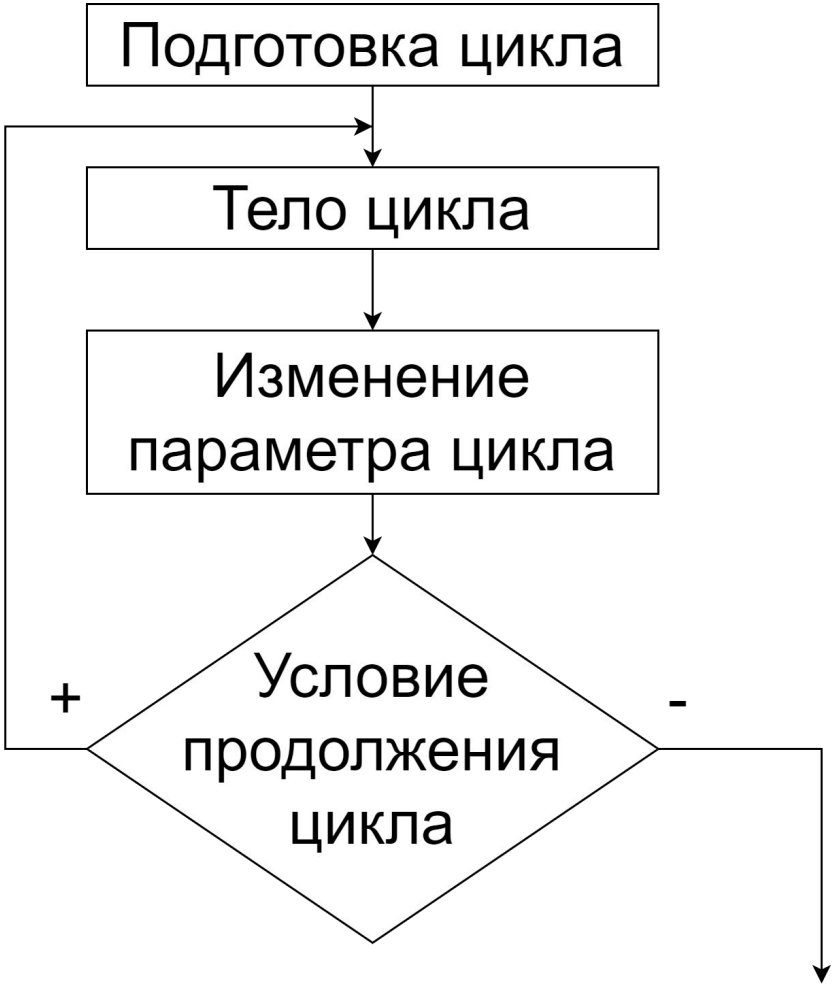
В соответствии с видом условия выполнения:

- циклы с параметром
- итерационные циклы

Циклическая структура «ДО»

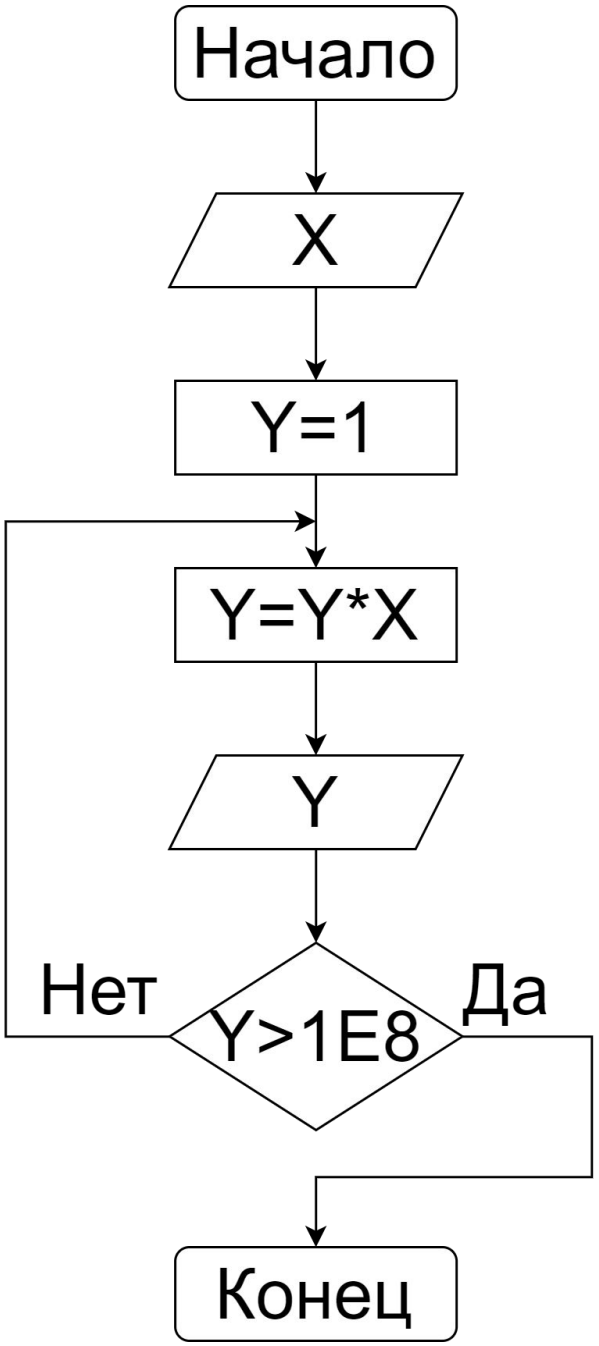
Повторять до тех пор, пока не будет выполнено условие.

Особенность этого цикла состоит в том, что он выполняется хотя бы один раз, так как первая проверка условия происходит после того, как тело цикла выполнено.



Пример

- Дано X>1. Вычислить и вывести степени X до тех пор, пока вычисленное значение не станет больше 10^8.

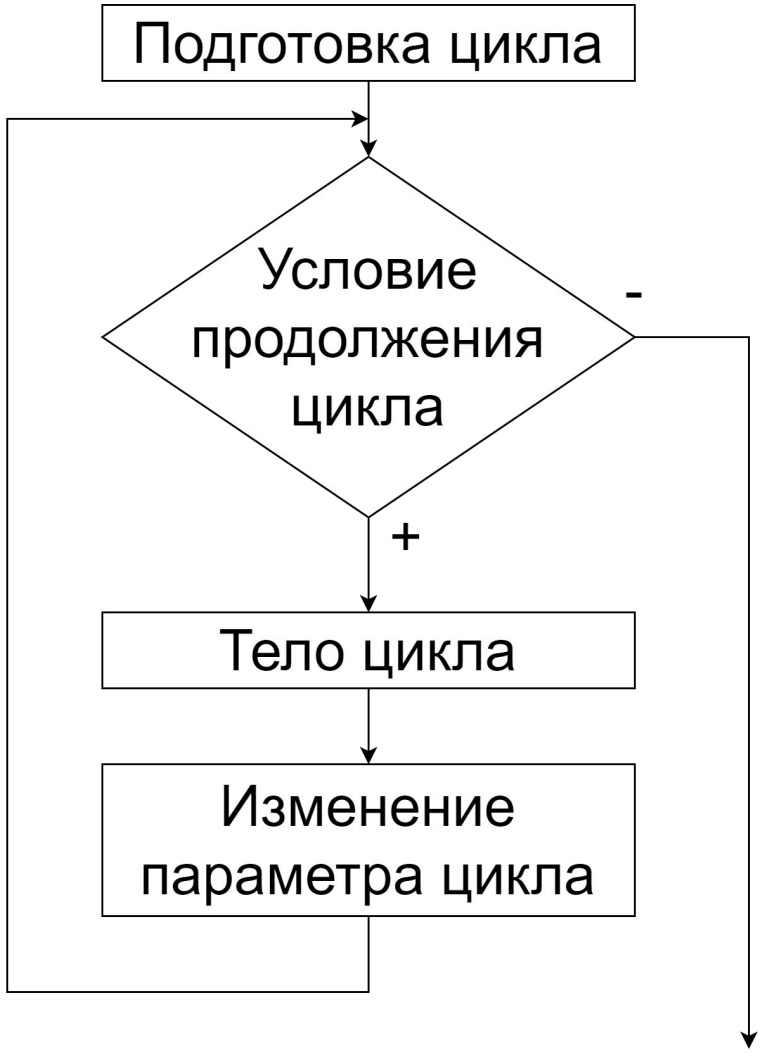


Циклическая структура «ПОКА»

Повторять до тех пор, пока выполняется условие.

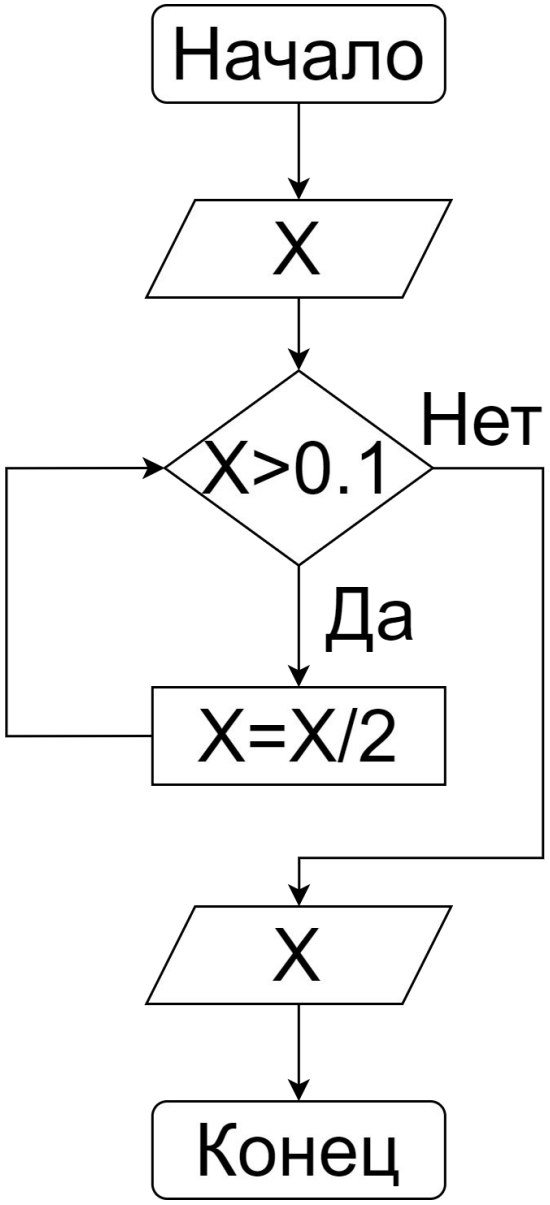
Отличается от цикла «ДО» тем, что здесь проверка условия проводится до выполнения тела цикла.

Если при первой проверке условие выхода из цикла выполняется, то тело цикла не выполнится ни разу.



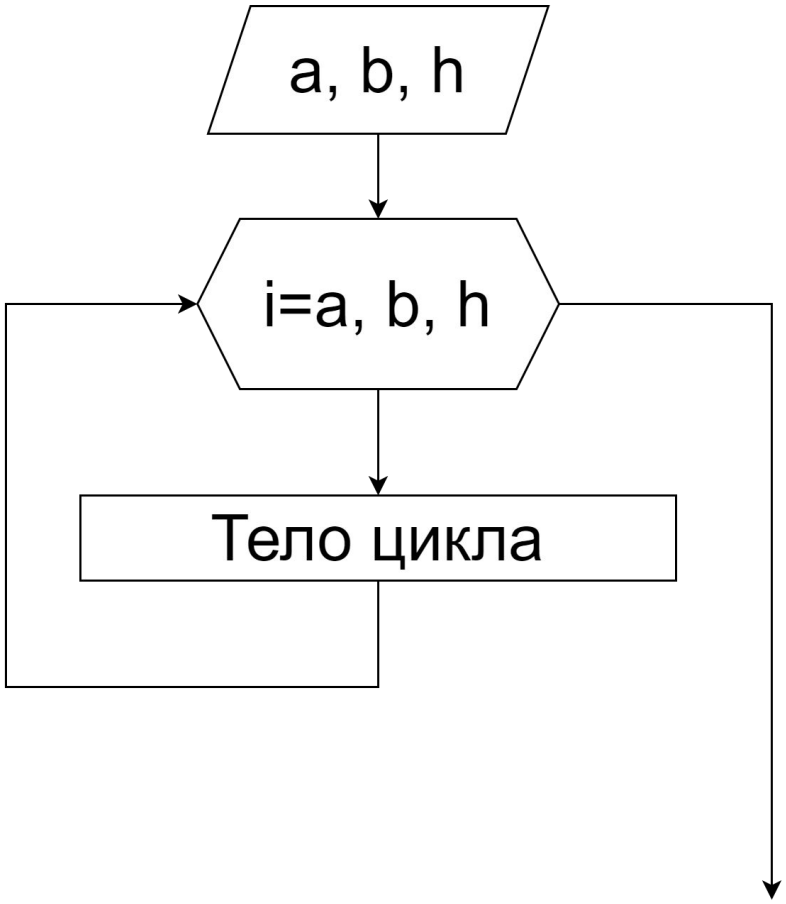
Пример

Дано X. Надо делить его пополам до тех пор, пока X будет больше 0.1.



Циклическая структура «ДЛЯ»

В блоке «модификация» объединяются несколько блоков: подготовка цикла, проверка окончания, изменение параметра цикла.



i – параметр цикла

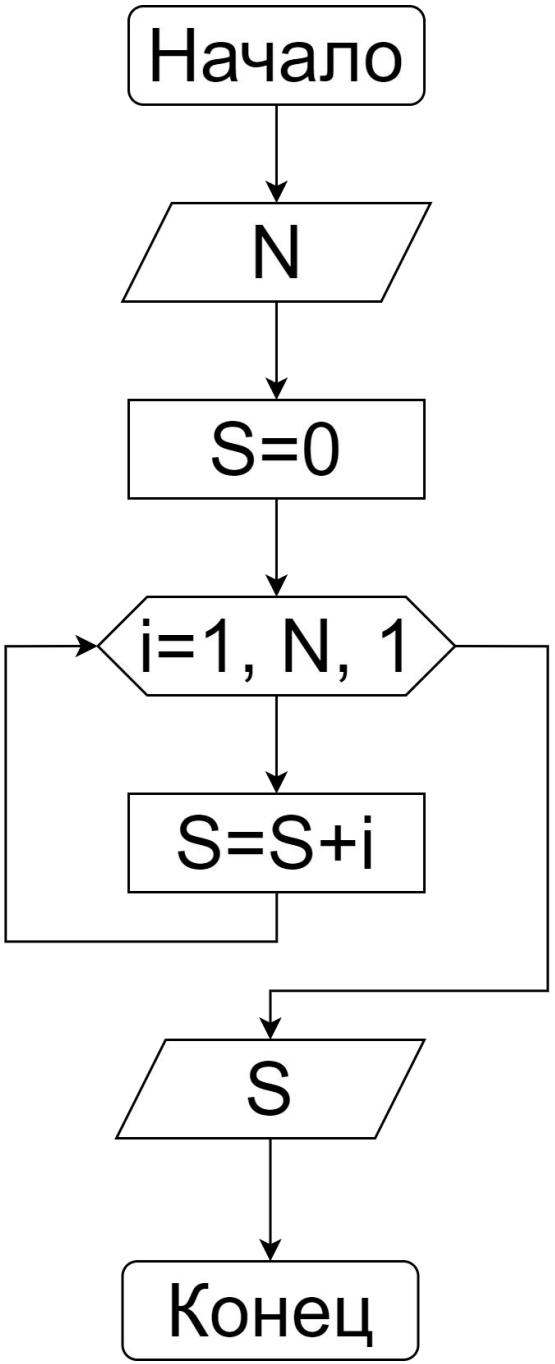
a – начальное значение параметра цикла

b – конечное значение параметра цикла

h – шаг изменения параметра цикла

Пример

Вычислить сумму всех целых чисел от 1 до N.



Величины в алгоритмах

В алгоритмах для данных используются специальные символические обозначения, которые аналогичны обозначениям в математике, представляют имена величин или их **идентификаторы**.

Типы данных: целые, вещественные, логические, текстовые (символьные) и другие.

Величины в алгоритмах

В алгоритмах и программах используются два вида величин:

1. **Константа** – это величина, значение которой не изменяется в процессе выполнения алгоритма и программы.

2. **Переменная** – это величина, значение которой изменяется в процессе выполнения алгоритма и программы

Величины в алгоритмах

Используются простые и индексные переменные (переменные с индексом). Переменная с индексом – это элемент некоторой заданной последовательности значений, которые называются **массивом**.

Массивом называется совокупность однотипных данных, связанных общим именем.

Величины в алгоритмах

Массив может быть одномерным (вектор), количество индексов – один; двумерным, количество индексов – два и т.д. (матрицы).

Количество индексов определяет размерность массива.

Обращение к элементам массива: X(1), Y(2)(3).

Величины в алгоритмах

Основными характеристиками массива являются:

– имя массива;

– тип элементов массива;

– размерность, равная количеству индексов (измерений) массива;

– размер (длина) массива – количество компонентов.