Тема № 1.
Основы алгоритмизации

Понятие алгоритма.

Свойства алгоритма.

Способы записи алгоритмов.

Типы алгоритмов.

Что такое алгоритм?

Алгоритм – конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи.

Задача:

Как положить слона в холодильник?

Положить слона в холодильник:

1. Открыть холодильник
2. Положить туда слона
3. Закрыть холодильник

Задача:

Как положить жирафа в холодильник?

Положить жирафа в холодильник:

1. Открыть холодильник
2. Достать слона
3. Положить туда жирафа
4. Закрыть холодильник

Алгоритм — строго определенная последовательность действий для некоторого исполнителя, приводящая к поставленной цели или заданному результату за конечное число шагов.

Любой алгоритм составляется в расчете на конкретного исполнителя с учетом его возможностей.

Исполнитель — субъект, способный исполнять некоторый набор команд. Совокупность команд, которые исполнитель может понять и выполнить, называется системой команд исполнителя.

Для выполнения алгоритма исполнителю недостаточно только самого алгоритма. Выполнить алгоритм — значит применить его к решению конкретной задачи, т. е. выполнить запланированные действия по отношению к определенным входным данным. Поэтому исполнителю необходимо иметь исходные (входные) данные — те, что задаются до начала алгоритма.


Исходные данные
Алгоритм
Результат

Свойства алгоритмов

Дискретность.

Конечность.

Понятность.

Определенность.

Массовость.

Дискретность.

Процесс решения задачи должен быть разбит на последовательность отдельных шагов — простых действий, которые выполняются одно за другим в определенном порядке. Каждый шаг называется командой (инструкцией). Только после завершения одной команды можно перейти к выполнению следующей.

Конечность.

Исполнение алгоритма должно завершиться за конечное число шагов; при этом должен быть получен результат.

Понятность.

Каждая команда алгоритма должна быть понятна исполнителю. Алгоритм должен содержать только те команды, которые входят в систему команд его исполнителя.

Определенность

Каждая команда алгоритма должна быть точно и однозначно определена. Также однозначно должно быть определено, какая команда будет выполняться на следующем шаге. Результат выполнения команды не должен зависеть ни от какой дополнительной информации. У исполнителя не должно быть возможности принять самостоятельное решение (т. е. он исполняет алгоритм формально, не вникая в его смысл). Благодаря этому любой исполнитель, имеющий необходимую систему команд, получит один и тот же результат на основании одних и тех же исходных данных, выполняя одну и ту же цепочку команд.

Массовость.

Алгоритм предназначен для решения не одной конкретной задачи, а целого класса задач, который определяется диапазоном возможных входных данных.

Способы представления алгоритмов.

Словесная запись.

Блок–схема

Формальные алгоритмические языки

Псевдокод

Словесная запись
(на естественном языке).

Алгоритм записывается в виде последовательности пронумерованных команд, каждая из которых представляет собой произвольное изложение действия.

Положить слона в холодильник:

1. Открыть холодильник
2. Положить туда слона
3. Закрыть холодильник

блок–схема (графическое изображение).

Алгоритм представляется с помощью специальных значков (геометрических фигур) — блоков.


Начало
Открыть холодильник
Положить в холодильник слона
Закрыть холодильник
Конец

Формальные алгоритмические языки.

Для записи алгоритма используется специальная система обозначений (искусственный язык, называемый алгоритмическим).

```
Алг сумма (арг вещ a,b рез вещ result)
нач
    ввод a, b
    result := a/b
    вывод result
кон
```

Псевдокод.

Запись алгоритма на основе синтеза алгоритмического и обычного языков. Базовые структуры алгоритма записываются строго с помощью элементов некоторого базового алгоритмического языка.

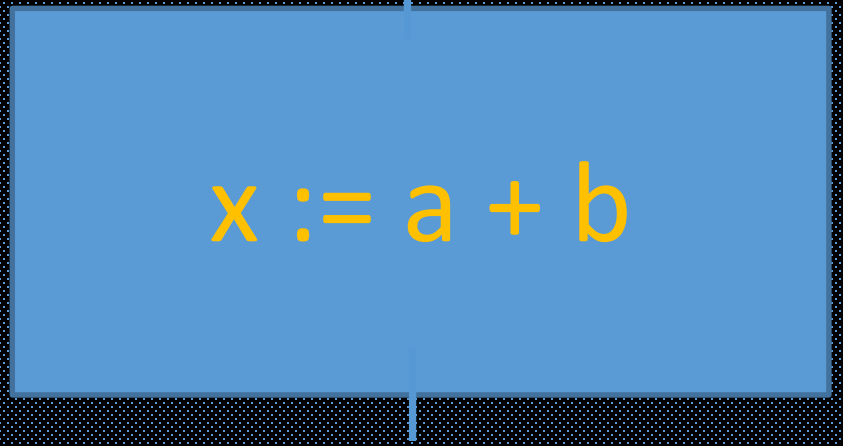
```
Начало
Ввод (a, b);
Если b<>0
То result := a/b;
Иначе result := None;
Вывод(‘Результат=’ result);
Конец.
```

Символы для изображения графических схем алгоритмов

Терминатор

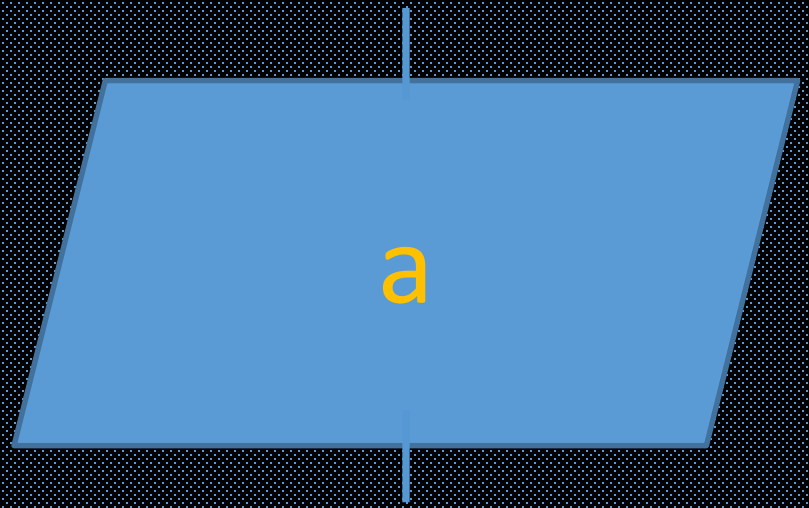
Символ отображает выход во внешнюю среду и вход из внешней среды. Используется для обозначения начала или окончания алгоритма

Процесс



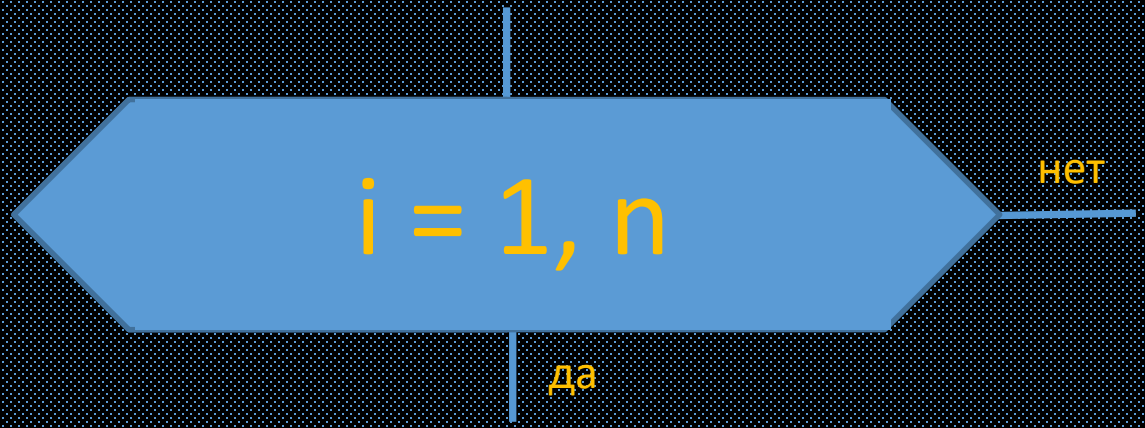
Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации). Используется для обозначения операций присваивания

Данные



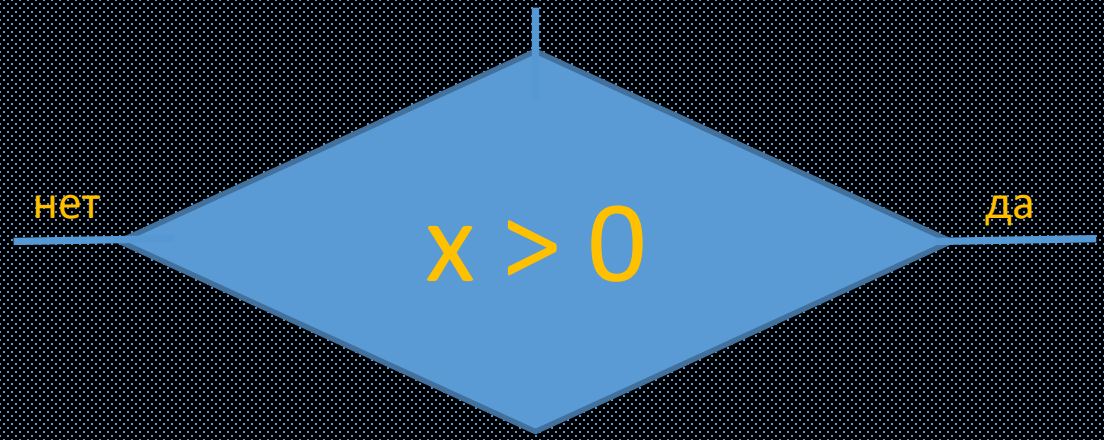
Символ отображает данные, носитель данных не определен. Используется для обозначения операций ввода и вывода данных

Подготовка



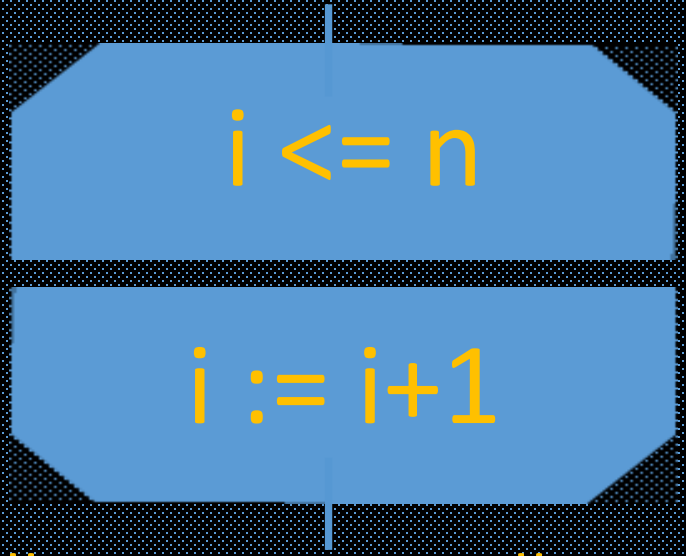
Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы). Может быть использован для обозначения заголовка цикла

Решение



Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.

Граница цикла



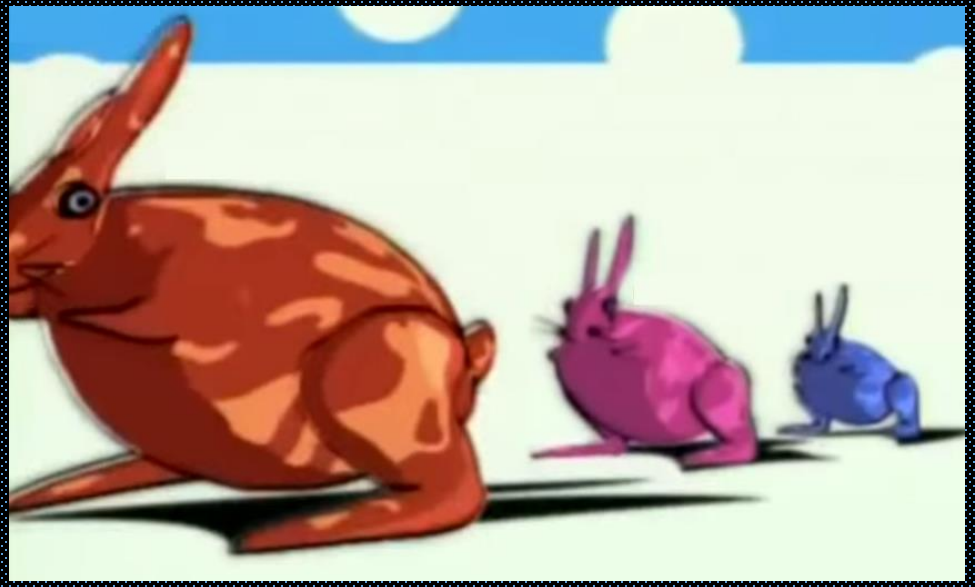
Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или в конце в зависимости от типа цикла.

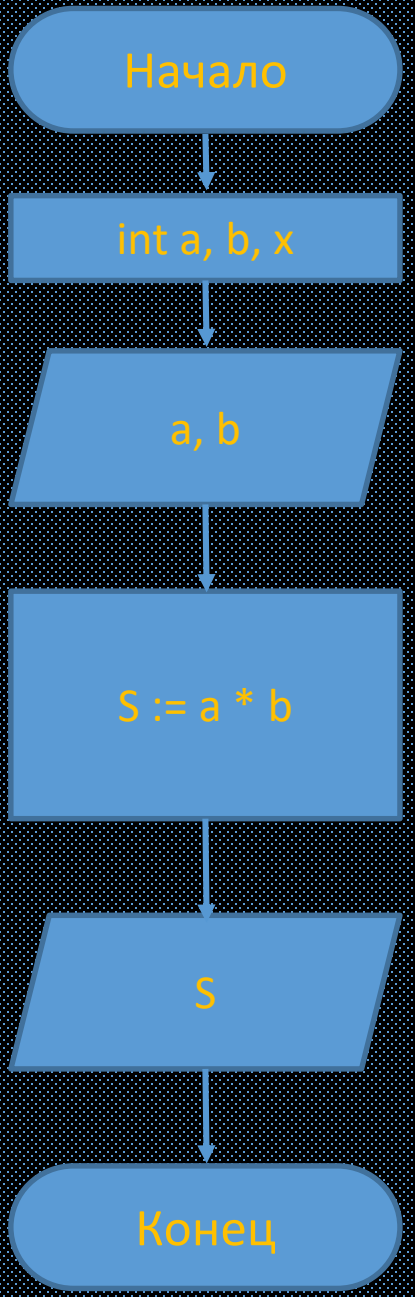
Базовые управляющие структуры алгоритмов

Логическая структура любой программы может быть выражена комбинацией из следующих базовых структур:

1) Композиция (следование);
2) Альтернатива (ветвление);
3) Итерация (цикл).

Композиция, или *следование* – это линейная конструкция алгоритма, составленная из последовательно следующих друг за другом функциональных вершин. Операции, группы операций или базовые структуры алгоритмов выполняются последовательно друг за другом.

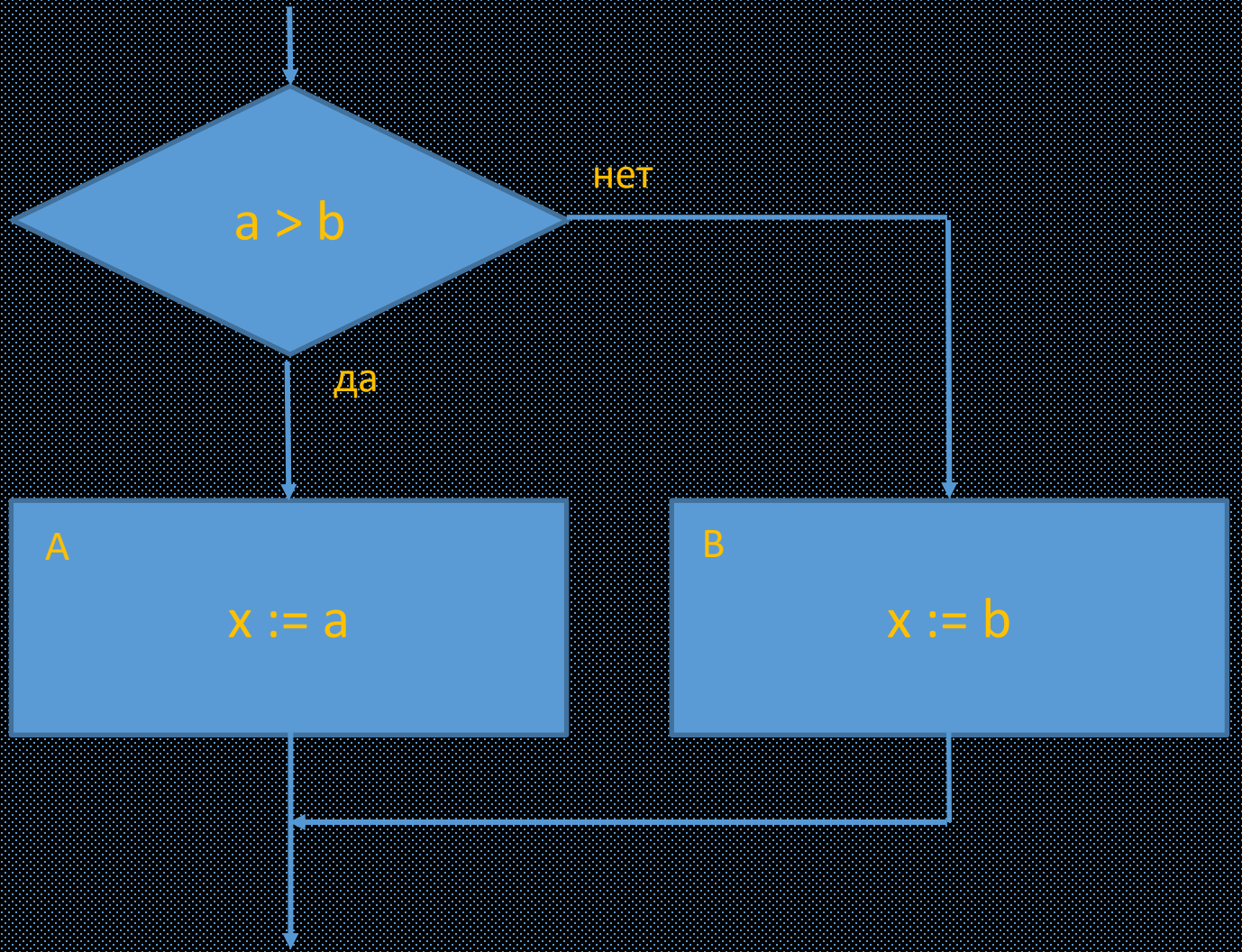




Альтернатива, или *ветвление* – это конструкция ветвления, имеющая предикатную вершину.

Структура обеспечивает выбор между двумя альтернативами: если *условие выполняется, т.е.* ИСТИНА (TRUE), то выполняется структура *A*; если ЛОЖЬ (FALSE) – структура *B* .

При этом происходит разветвление алгоритма.

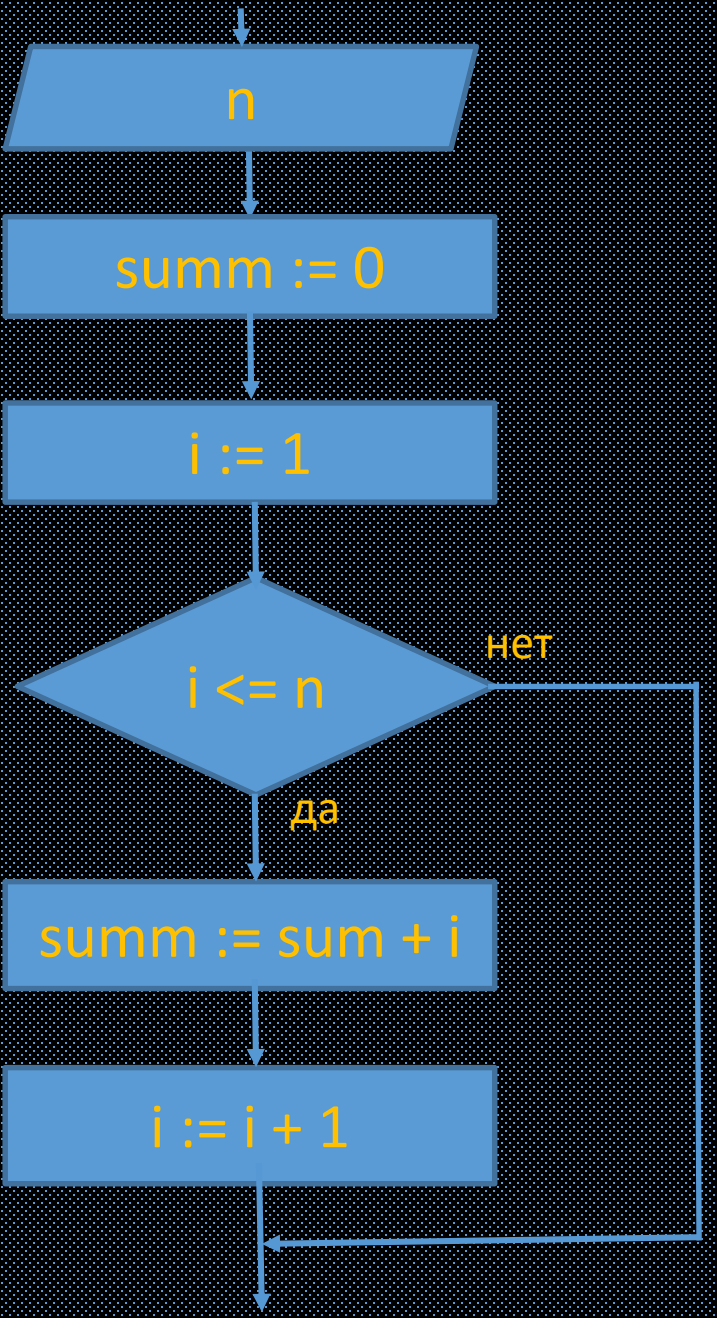


Итерация, или *циклы* – это циклическая конструкция алгоритма, состоящая из композиции и альтернативы.

Цикл представляет собой повторное выполнение определённого набора действий при выполнении некоторого условия. Именно циклы позволяют записывать длинные последовательности операций небольшим числом команд.

Цикл с предусловием

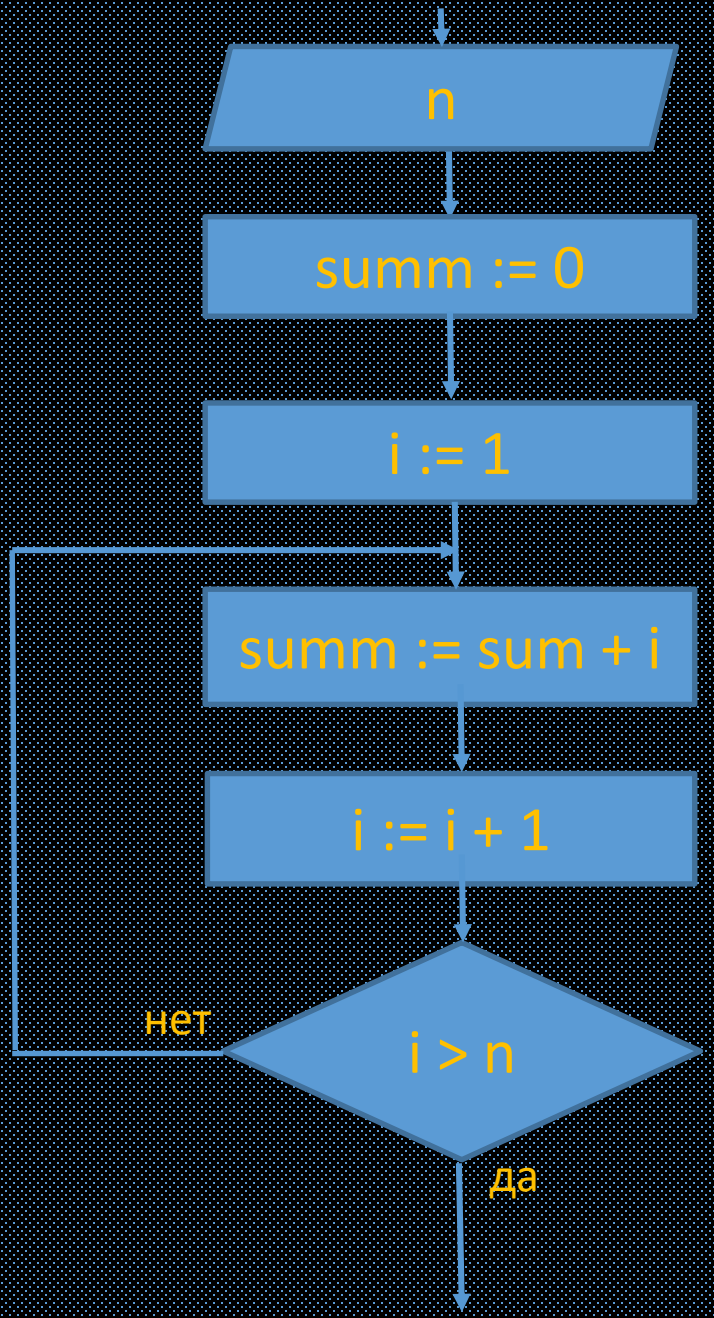
(«Цикл Пока» или «WHILE»)



Цикл с постусловием

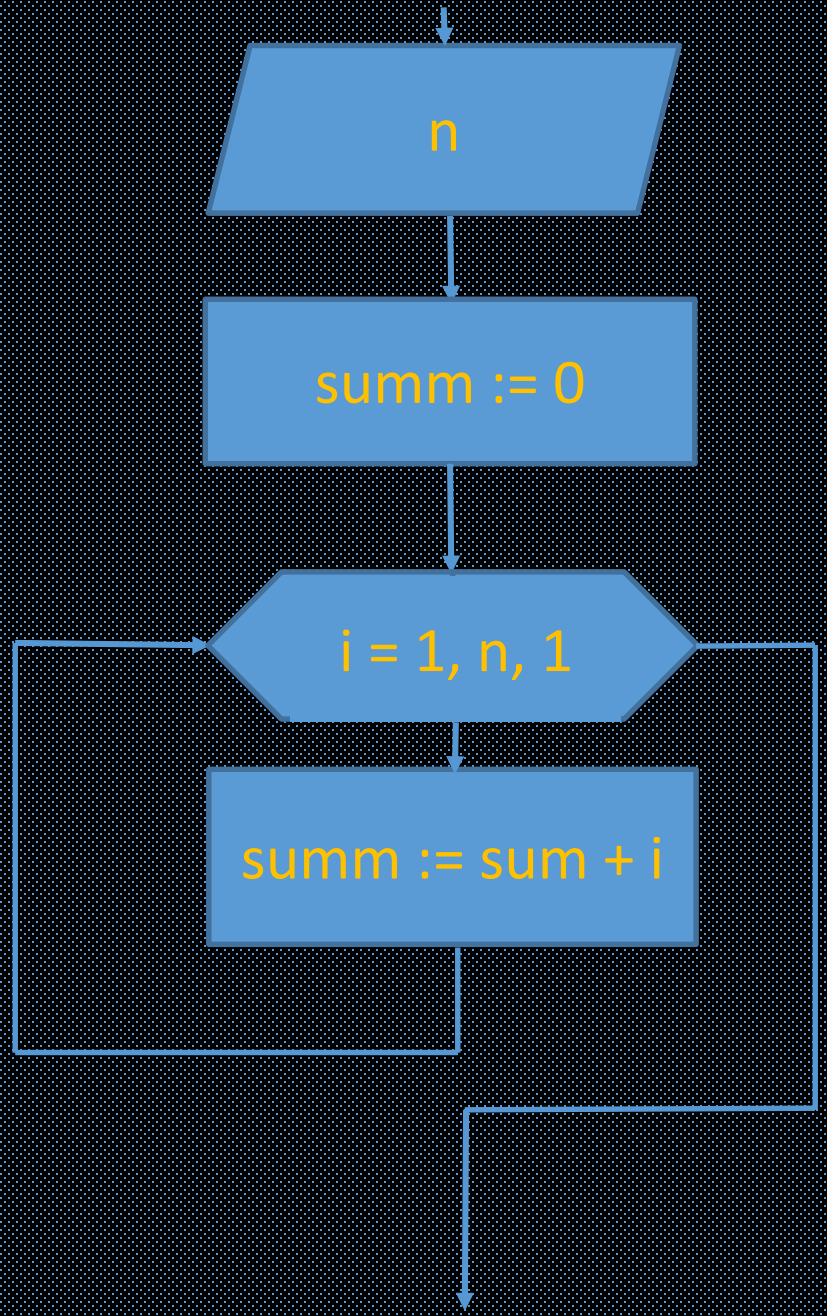
(«Цикл Выполнять До» или

«DO WHILE»)



Цикл с параметром

(«Цикл Для» или «FOR»)



Литература:

Трофимов, В. В. Основы алгоритмизации и программирования : учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2022. — 137 с. — (Профессиональное образование). — ISBN 978-5-534-07321-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/493261 (дата обращения: 26.08.2022).

Соловьева Т.Н. Информатика. Основы алгоритмизации и программирования. Учебное пособие. Пермский государственный национальный исследовательский университет. Пермь. 2018.

https://github.com/ViktorViktorovitsh/lessons

Задание на дом:

1. Разработайте схему алгоритма для расчета площади круга (S = pi * r ^2). Значение радиуса вводится с клавиатуры.
2. В задачу из п.1 добавить проверку, что пользователь ввел положительное значение радиуса круга.
3. Разработайте схему алгоритма для вычисления суммы всех четных чисел из диапазона от 0 до n.