Тема 12. Алгоритмы

1. Определение алгоритма. Свойства.
2. Формы представления алгоритмов.
3. Базовые алгоритмические структуры
4. Программирование.
5. Принципы разработки алгоритмов и программ.

Алгоритм - конечная последовательность команд, исполнителю, приводящая от исходных данных к результату.

Имя великого математика IX века Мухаммеда ибн Мусы аль-Хорезми

- Дискретность
- Понятность
- Определенность
- Результативность
- Массовость

- Словесно-пошаговый
- Учебный алгоритмический язык
- Язык программирования
- Блок-схема
- Таблица

- Линейная
- Разветвляющаяся
- Циклическая
- Вспомогательная

Формальное исполнение алгоритма

Основные характеристики

- Среда
- Система команд исполнителя
- Элементарные действия
- Отказ

Примеры

1. Формальный исполнитель Автомат.

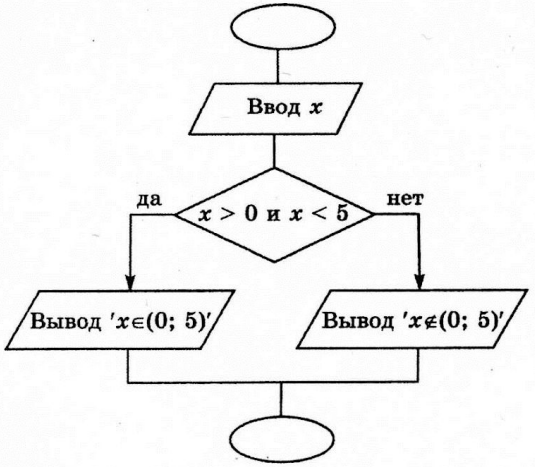
Среда — целые неотрицательные числа

Команда	Действие
Добавить	К числу добавляется 1
Удвоить	Число умножается на 2

Алгоритм получения числа 8:

Алгоритм	
	0
Добавить	1
Удвоить	2
Удвоить	4
Удвоить	8

2. Алгоритм, проверяющий принадлежность числа, введенного с клавиатуры, интервалу (0; 5):



1. Определение алгоритма. Свойства

Алгоритм – четкое описание последовательности действий, которые необходимо выполнить для решения задачи.

Алгоритм описывает последовательный процесс получения результаты из исходных данных.

Свойства алгоритма

1. Дискретность
2. Определенность
3. Результативность
4. Массовость

2. Формы записи алгоритма

1. Словесное описание алгоритма

Коль кругом все будет мирно,
Так сидеть он будет смирно;
Но лишь чуть со стороны
Ожидать тебе войны
Иль набега силы бранной,
Иль другой беды незваной,
Вмиг тогда мой петушок
Приподымет гребешок,
Закричит и встрепенется
И в то место обернется.

А.С. Пушкин

2. Формульно-словесный способ.

Дано Массив a[i], содержащий 10 чисел.

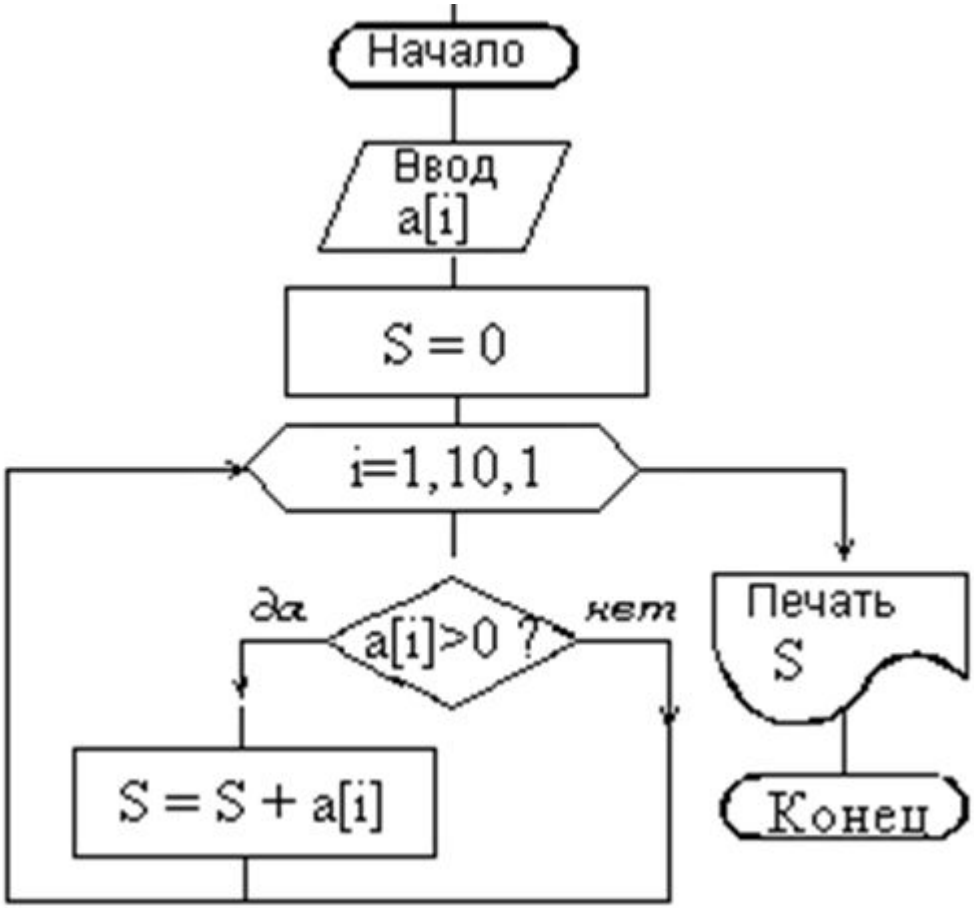
Вычислить сумму положительных элементов массива.

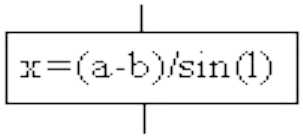
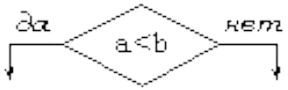
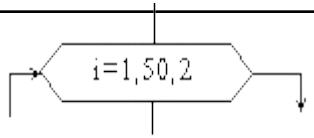
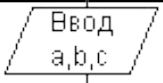
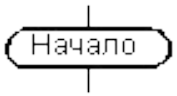
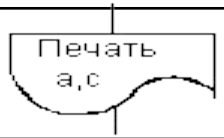
Решение.

Обозначим: S – сумма элементов массива, i – текущий номер элемента массива. Пусть начальное значение S=0.

Если a[i]>0, то S=S+a[i] для $\forall i \in [1...10]$

3. Графический способ (схема алгоритма)



Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

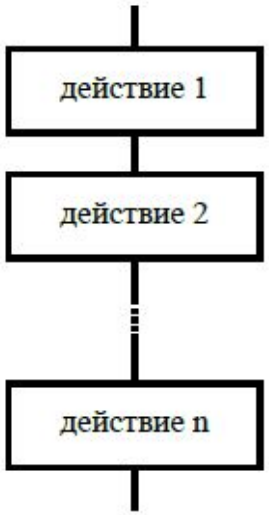
3. Базовые алгоритмические структуры

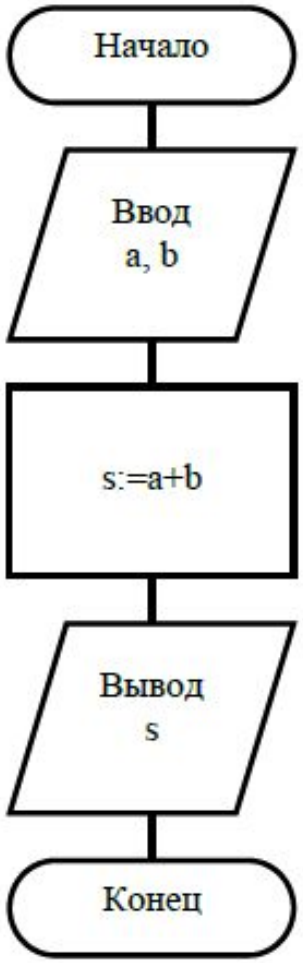
Три базовые структуры алгоритма

1.Линейная
2. Разветвляющаяся
3.Циклическая

Линейная алгоритмическая структура

Образуется последовательностью действий, следующих одно за другим.

Схема алгоритма	Алгоритмический язык
	**алг** **нач** …… действие 1 действие 2 …… действие n …… **кон**




```
алг сумма (арг вещ a, b, рез вещ s)
нач
|       ввод a, b
|       s:=a+b
|       вывод s
кон
```

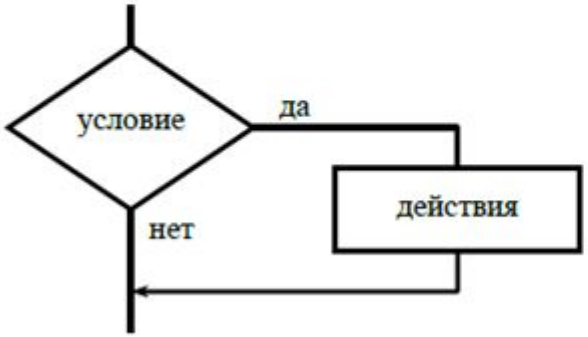
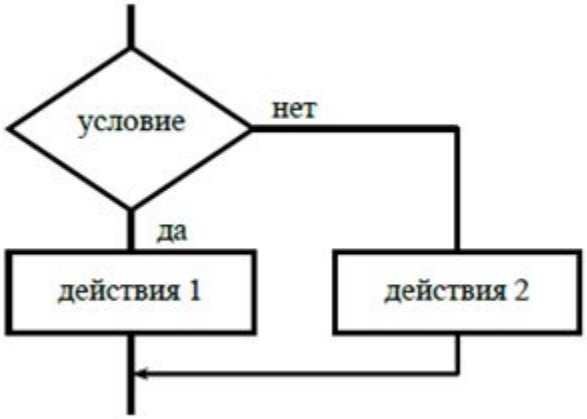
Разветвляющаяся алгоритмическая структура

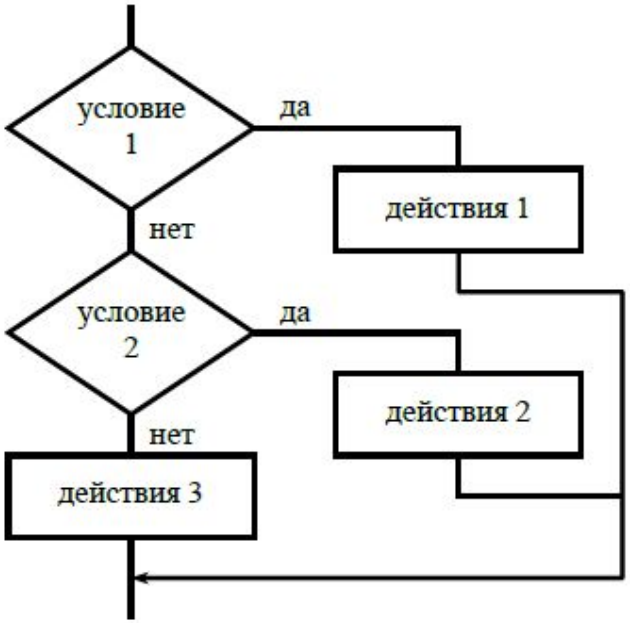
Обеспечивает в зависимости от результата проверки условия (**да** или **нет**) выбор одного из альтернативных путей выполнения алгоритма.

Каждый из путей ведет к **общему выходу**, т.е. работа алгоритма продолжается независимо от выбора пути.

Существует в двух вариантах:

1. Полное ветвление (если – то – иначе).
2. Неполное ветвление (если – то).

Схема алгоритма	Алгоритмический язык
Ветвление (если-то)	
	**алг** **нач** | …… | **если** условие | | **то** действия | **все** | …… **кон**
Ветвление (если-то-иначе)	
	**алг** **нач** | …… | **если** условие | | **то** действия 1 | | **иначе** действия 2 | **все** | …… **кон**




```
алг
нач
|       ..............
|       если условие 1
|       |       то действия 1
|       |       иначе если условие 2
|       |       |               то действия 2
|       |       |               иначе действия 3
|       |               все
|       все
|       ..............
кон
```

Циклическая алгоритмическая структура

Обеспечивает многократное выполнение некоторой совокупности действий, которая называется **телом цикла**. Существует в следующих вариантах:

1. **Цикл с предусловием**. Тело цикла выполняется до тех пор, пока не будет выполнено условие выхода из цикла. Цикл может не выполнится ни разу.
2. **Цикл с постусловием**. Проверка условия выхода из цикла выполняется после того, как будет выполнено тело цикла. Цикл всегда выполняется хотя бы один раз.
3. **Цикл со счетчиком** (с известным количеством повторений). Тело цикл выполняется для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

Переменная , определяющая количество повторений цикла, называется **параметром цикла**.

Организация цикла.

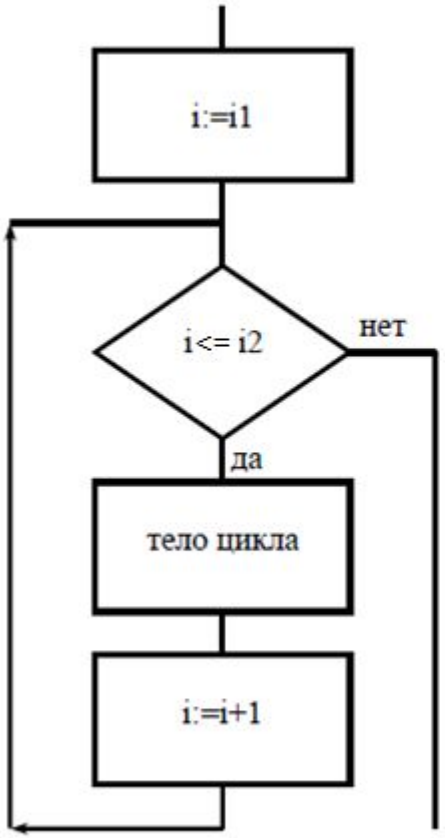
1. Указать начальное значение параметра цикла.
2. Определить тело цикла, т.е. указать команды, которые неоднократно выполняются в цикле.
3. В теле цикла организовать изменение параметра цикла с каким-либо шагом.
4. Для параметра цикла определить условие продолжения тела цикла.

Развёрнутая схема алгоритма	Альтернативное обозначение	Алгоритмический язык
1. Цикл с предусловием		
		**алг** **нач** …… начальные присваивания **нц пока** условие тело цикла (действия) **кц** …… **кон**

2. Цикл с постусловием

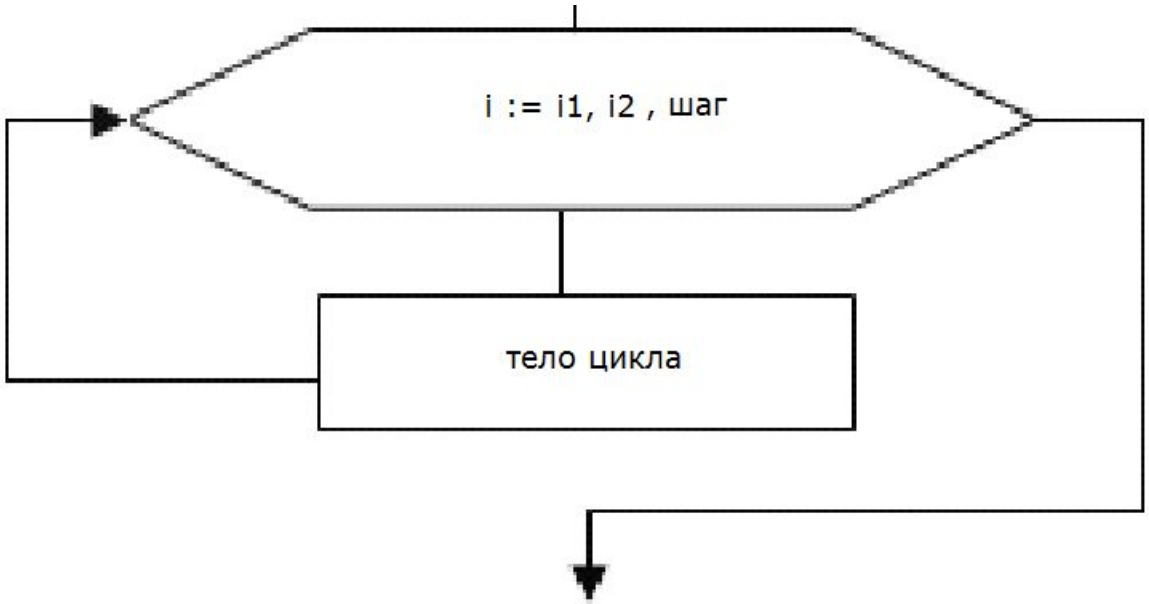

```
алг
нач
|       ..............
|       начальные
|       присваивания
|       нц
|       |       тело цикла
|       |       (действия)
|       кц пока условие
|       ..............
кон
```

3. Цикл со счётчиком




```
алг
нач
|       ..............
|       нц для i от i1 до i2
|       |     тело цикла
|       |     (действия)
|       кц
|       ..............
кон
```

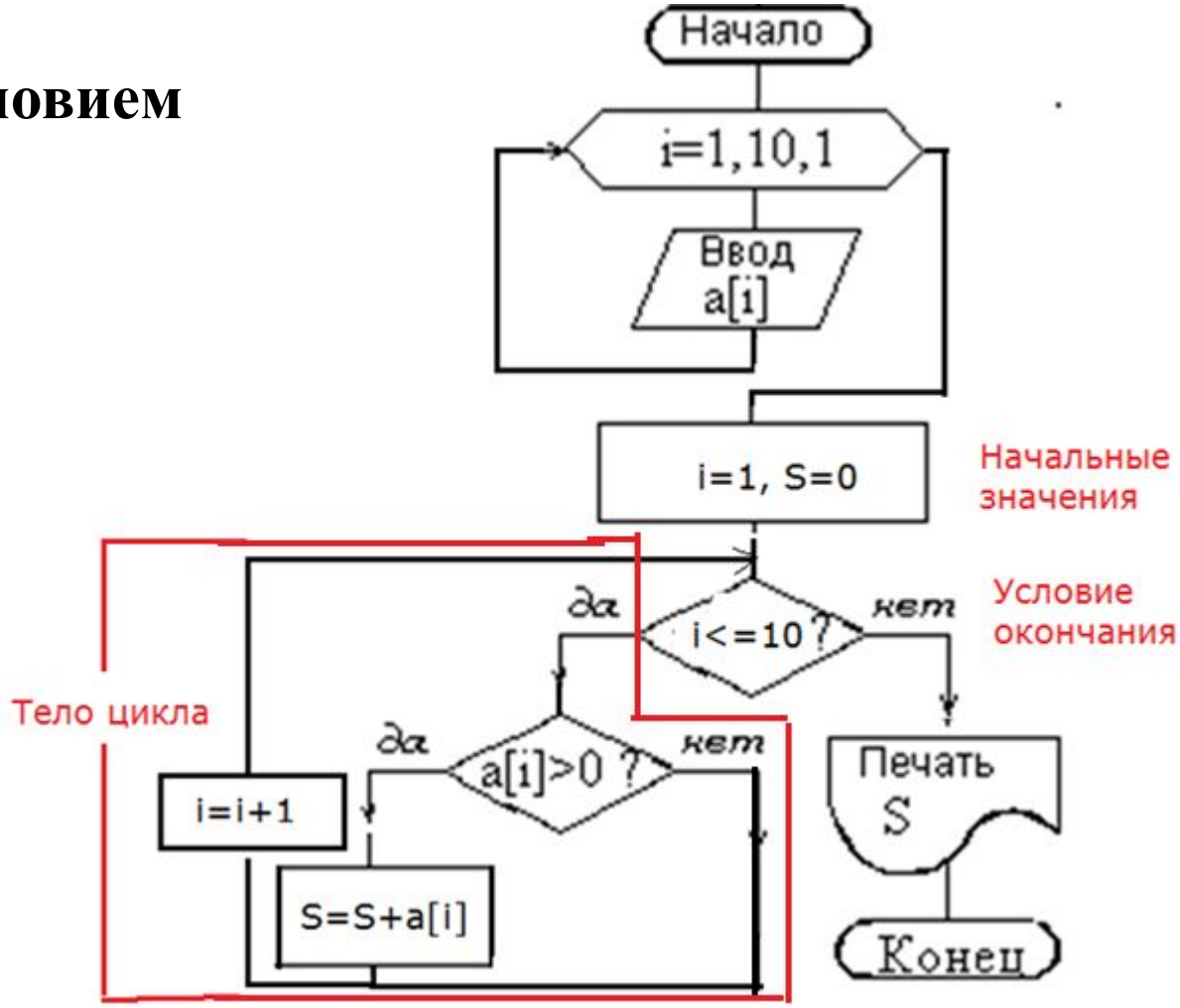
Альтернативное изображение арифметического цикла



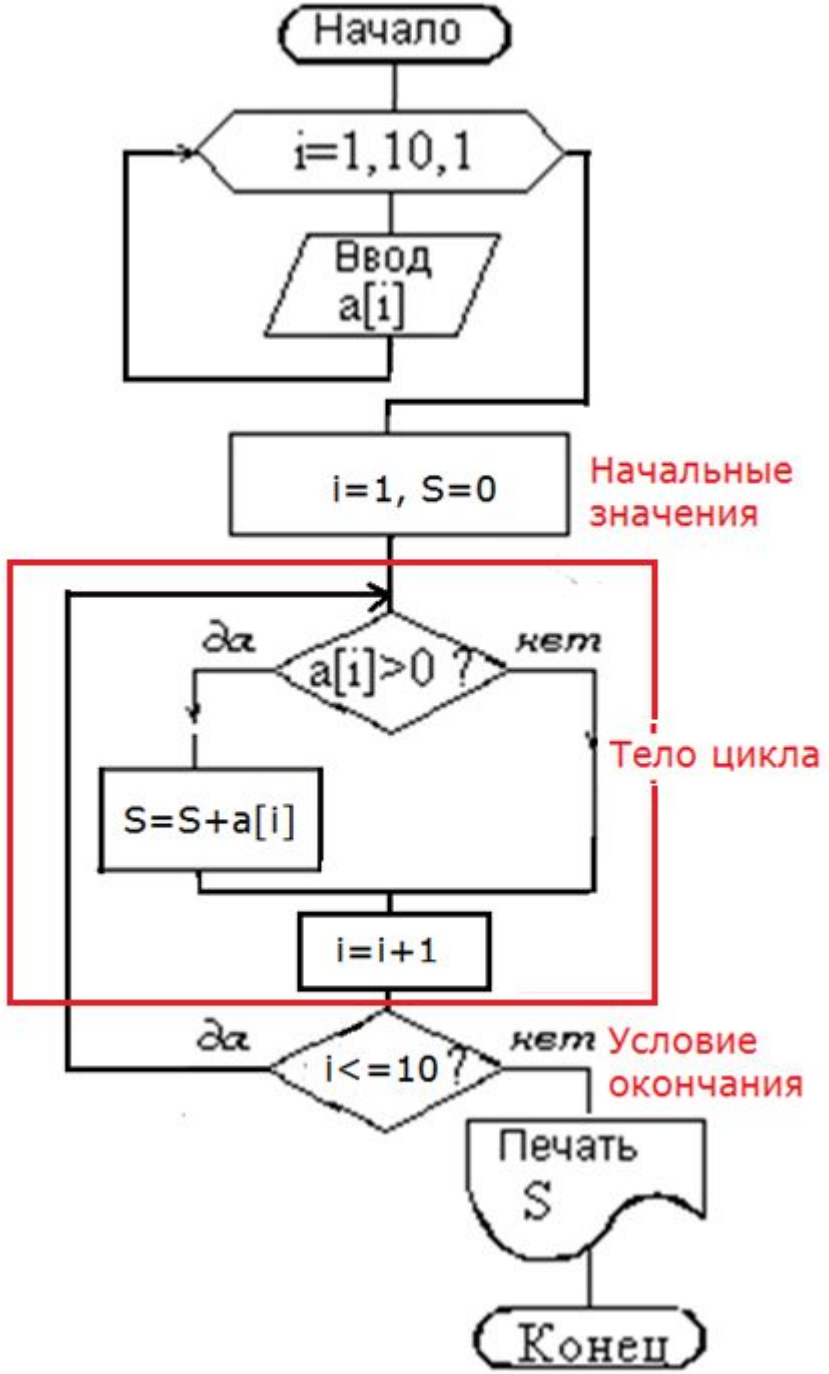
Количество повторений цикла **n=(i2-i1)/ шаг +1**

Задача. Даны 10 чисел. Найти сумму положительных.

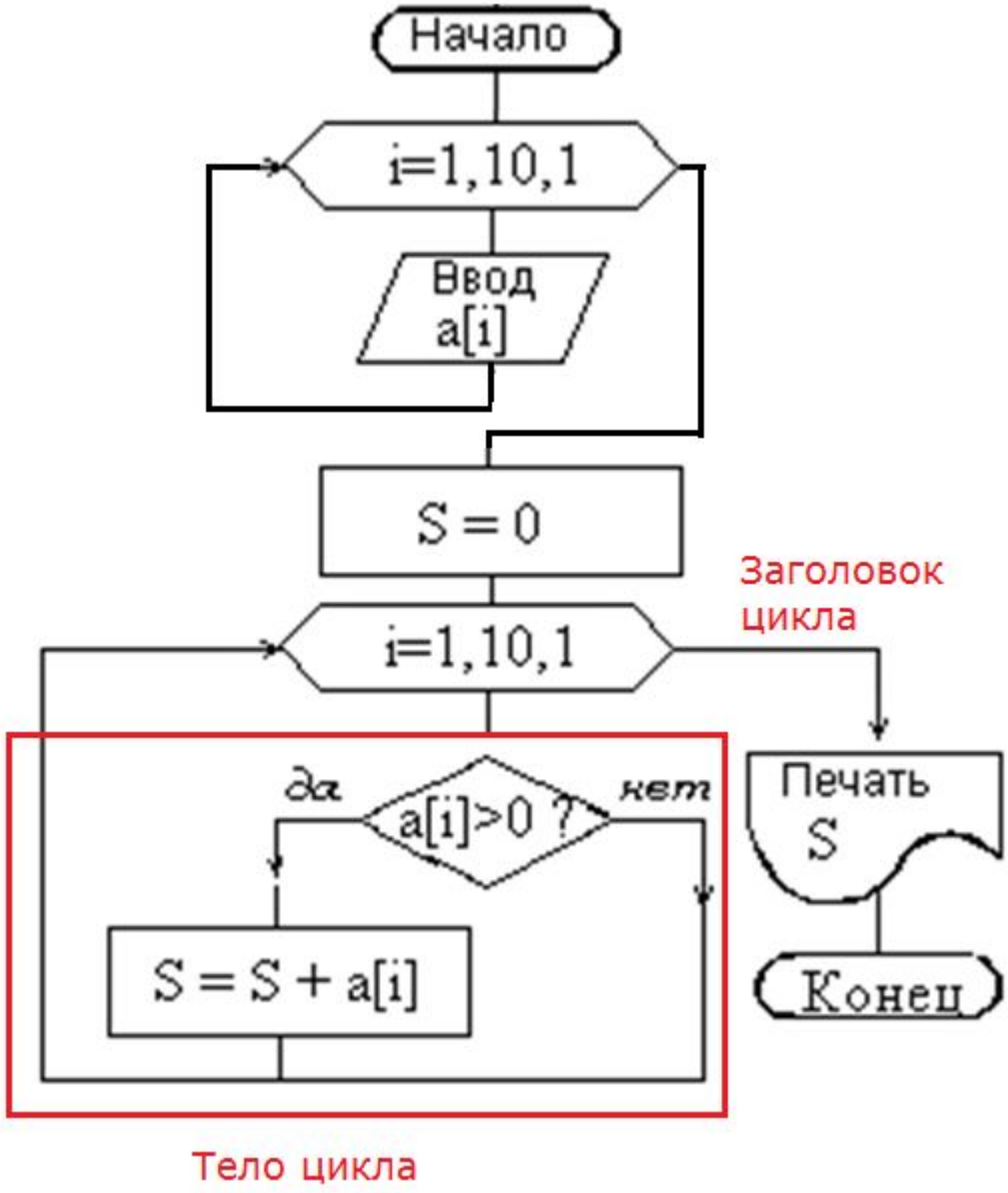
1. Цикл с предусловием

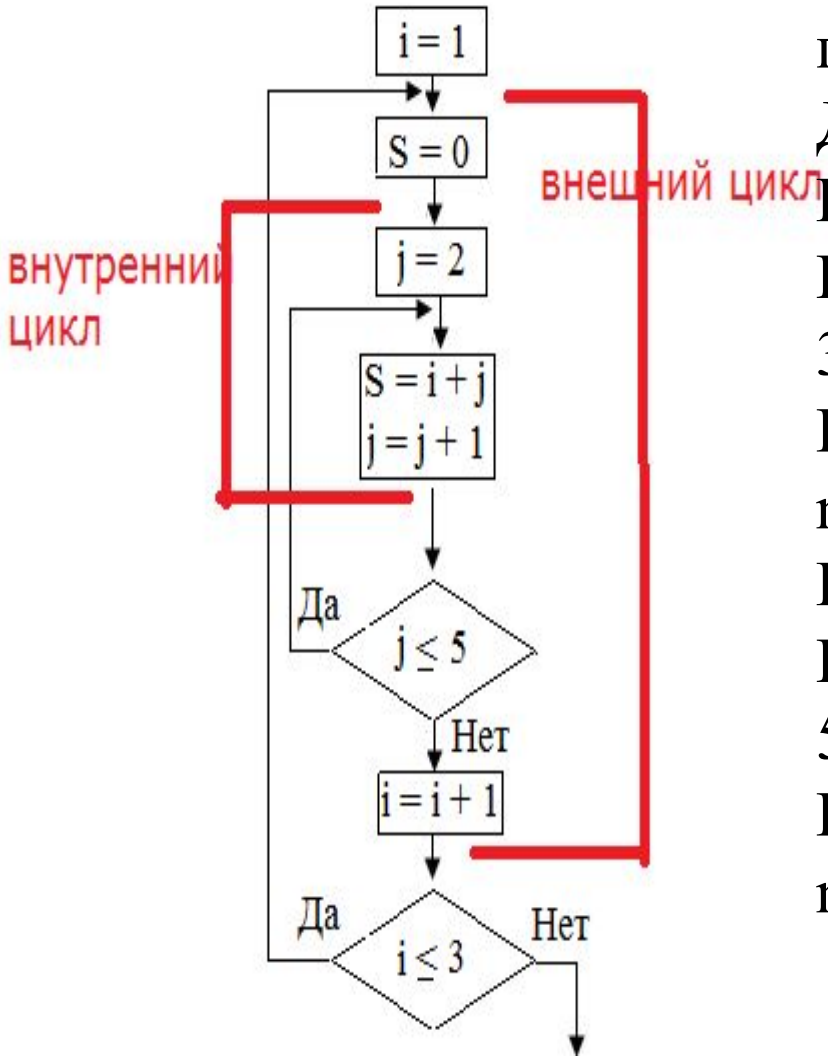


2. Цикл с постусловием



3. Цикл с параметром





Определить значение целочисленной переменной *S*.

Два цикла с постусловием.

Внешний цикл:

Параметр цикла i, изменяется от 1 до 3 с шагом 1.

Количество повторений цикла

n= (3-1)/1+1 = 3

Внутренний цикл:

Параметр цикла j, изменяется от 2 до 5 с шагом 1.

Количество повторений цикла

n= (5-2)/1+1 = 4

S	i	j	j<=5?	i<=3?
0				
	1			
3		2	**2**<=5, да	
4		3	**3**<=5, да	
5		4	**4**<=5, да	
6		5	**5**<=5, да	
7		6	**6**<=5, нет	
	2			**2**<=3, да
0				
4		2	**2**<=5, да	
5		3	**3**<=5, да	
6		4	**4**<=5, да	
7		5	**5**<=5, да	
8		6	**6**<=5, нет	
	3			**3**<=3, да
0				
5		2	**2**<=5, да	
6		3	**3**<=5, да	
7		4	**4**<=5, да	
8		5	**5**<=5, да	
9		6	**6**<=5, нет	
	4			**4**<=3, нет

4. Программирование

Программа – это алгоритм, записанный на языке программирования

Этапы разработки программ

1. **Постановка задачи.**

Задачи этапа:

Формулирование цели решения задачи.

Определение состава и формы представления входной, промежуточной и выходной информации.

Описание контрольного примера, штатных и нештатных ситуаций и ответных действий пользователя.

2. Моделирование задачи и выбор метода ее решения.

Составляется либо математическая, либо информационная модель.

Метод решения должен удовлетворять следующим **требованиям**:

- обеспечивает необходимую точность результатов;
- позволяет использовать готовые стандартные программы;
- ориентирован на минимальный объем исходной информации;
- обеспечивает наиболее полное получение результатов.

3. Составление алгоритма решения задачи.
4. Программирование (кодирование алгоритма) является завершающим этапом технологического процесса разработки программ, предшествующий машинной реализации. Выполняется с использованием языка программирования.

5. Тестирование и отладка – заключительный этап разработки программы.

Тестирование – совокупность действий, предназначенных для демонстрации правильности работы программы в заданных диапазонах изменения внешних условий и режимах эксплуатации программы.

Цель – демонстрация отсутствия ошибок на заранее подготовленном наборе тестовых примерах.

Виды тестирования: функциональное, структурное, бета-тестирование.

Функциональное тестирование: программа рассматривается как **черный ящик** и проверяется соответствие поведения программы ее внешней спецификации.

Структурное тестирование: программа рассматривается как **белый ящик** и проверяется логика работы программы.

Перед выпуском программы проводится **бета-тестирование**, т.е. публичное тестирование бета-версии программы.

Отладка – совокупность действий, направленных на устранение ошибок в программах.

Типы ошибок:

- **синтаксические** – некорректная запись отдельных конструкций языка программирования; выявляются автоматически на этапе трансляции.
- **логические** – ошибки в логике работы программы на исходных данных.

6. Передача программы вместе с документацией в эксплуатацию. Основное назначение документации – обеспечить пользователя необходимыми инструкциями.

Два вида эксплуатации:

- **экспериментальная** – проверка работы программы на реальном объекте.
- **промышленная** эксплуатация.

Все рассмотренные этапы от момента зарождения программы до момента полного отказа от ее эксплуатации составляют **жизненный цикл программы**.

5. Принципы разработки алгоритмов и программ

1. **Процедурный** или **императивный** подход

ориентирован на операции, непосредственно выполняемые компьютером. Использует возможности и особенности конкретной ЭВМ и с развитием компьютерной техники не позволял перейти к массовому промышленному программированию.

Недостатки:
1. Программы трудно читать даже опытным программистам.
2. Команды безусловного и условного перехода приводили к большой и запутанной структуре программы.
3. Уловки для повышения эффективности программы приводили к ненадежности, трудностям в отладке и модификации.
4. Программировать процедурным методом трудно и дорого.

2. **Модульное** проектирование.

Модуль – это последовательность логически взаимосвязанных фрагментов, оформленных как отдельная часть программы. Модули обладают следующими свойствами:

1. К модулю можно обратиться по имени.
2. По завершении работы модуль должен вернуть управление тому модулю, который его вызван.
3. Модуль должен иметь один вход и один выход.
4. Модуль должен иметь небольшой размер

Преимущества модульного проектирования:

- Большую программу могут одновременно разрабатывать несколько исполнителей.
- Появляется возможность создавать библиотеки наиболее употребляемых программы.
- Упрощается процедура загрузки больших программ в оперативную память.
- Обеспечивается более эффективное сопровождение программ.

3. Структурное программирование.

Любую программу можно построить из трёх базовых конструкций:

1. Последовательное исполнение – однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
2. Ветвление – однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
3. Цикл – многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (продолжения цикла).

Принцип структурного программирования требует, чтобы все алгоритмические базовые конструкции имели один вход и один выход

1. Разработка программы ведётся пошагово, методом "сверху вниз« (нисходящее проектирование программ).

Программа разбивается на множество подпрограмм, комбинирование которых и формирует итоговый алгоритм решения задачи.

Подпрограммой называется набор операторов, выполняющих заданное действие и не зависящих от других частей исходного кода.

Отдельные подпрограммы оформляются в виде *модулей*.

4.Объектно-ориентированное программирование

Является развитием структурного подхода. Задача представляется как совокупность взаимодействующих объектов. Каждый объект содержит некоторую структуру данных и доступные только ему процедуры или методы обработки данных. Объединение данных и свойственных им процедур обработки в одном объекте называется инкапсуляцией.

Основные понятия

Объект – представитель некоторого класса однотипных объектов. Объект можно модифицировать, т.е. изменить его состояние. ***Класс*** определяет общие свойства для всех своих объектов.

К ним относятся:

инкапсуляция (способность изменять реализацию любого класса объектов без опасения, что это вызовет нежелательные побочные последствия в программной среде);

наследование (возможность создавать из имеющихся классов новые классы);

полиморфизм (способность объектов выбирать метод обработки).

5. **Автоматизированное программирование** с использованием **CASE-технологии**.

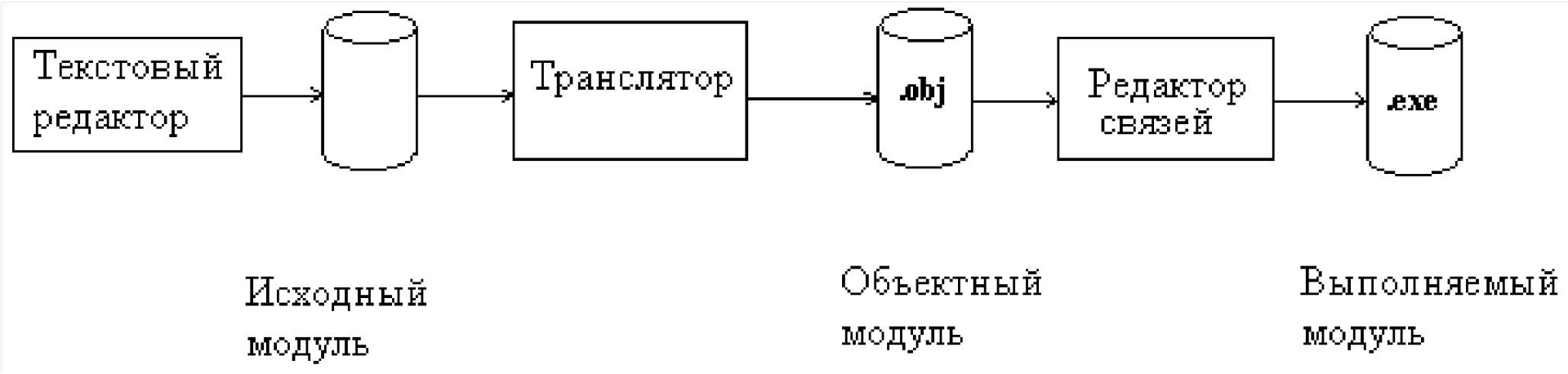
Позволяет генерировать программное обеспечение на основе централизованно хранящихся моделей.

6.Технология RAD – последовательный метод разработки программ в тесном взаимодействии с заказчиком.

Очередной этап создания программы начинается только после завершения предыдущего и не допускает возврата к нему.

7. Программно-инструментальные средства программирования.

Основу составляют системы автоматизации программирования или системы программирования, которые обеспечивают возможность решения задач непосредственно в среде операционной системы.



Текстовый редактор – создание текста программы на языке программирования.

Транслятор – перевод программы в машинный код и обнаружение синтаксических ошибок.

Редактор связей – создание законченной программы.