Переменная в программировании

В алгоритмических языках программирования переменные предназначены для хранения и обработки данных в программах.

Переменной называется величина, значение которой может меняться в процессе выполнения программы.

С понятием переменной связаны следующие характеристики (атрибуты):

Имя	Тип	Значение
Определяет обозначение переменной и ее место в памяти. Имя любой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. Имя переменной должно обязательно начинаться с буквы.	Определяет множество допустимых значений переменной и множество применимых к ней операций, объем занимаемой памяти и способ представления в памяти. Простейший способ задания типа переменной - использование в идентификаторе переменной определенного суффикса (специального значка), который приписывается к имени переменной.	Это динамическая характеристика, которая может меняться многократно в ходе исполнения алгоритма. Во время выполнения алгоритма в каждый конкретный момент величина имеет какое-то значение или не определена. Значениями переменных могут быть данные различных типов (целью или вещественные числа, последовательности символов, логические значения и т. д.).

Операция присваивания.

Любая переменная может получить или изменить свое значение с помощью **оператора присваивания.**

В общем виде оператор присваивания можно записать так:

<имя переменной> : = <выражение>

<имя переменной> : = <выражение>

Оператор выполняется следующим образом:	**Свойства операции присваивания:**
Вычисляется выражение в правой части оператора. После этого переменная, указанная в левой части, получает вычисленное значение. При этом тип выражения должен быть совместим по присваиванию с типом переменной, а значения всех переменных, входящих в выражение, были определены.	1. Пока переменной не присвоено значение, она остается неопределенной; 2. Значение, присвоенное переменной, сохраняется в ней вплоть до выполнения следующего присваивания этой переменной нового значения; 3. Новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Основные понятия алгоритмического программирования (данные, операторы, функции, процедуры и т. д.)

Данные	*Данные* - величины, обрабатываемые программой.	
	Переменные	*Константы*
	Данные, значения которых изменяются в процессе выполнения программы	*Данные, значения которых не могут меняться в процессе выполнения программы*

Основные понятия алгоритмического программирования (данные, операторы, функции, процедуры и т. д.)

Имена (идентификаторы)	Используются для обозначения объектов в программе (переменных, массивов, функций и др.) Каждая переменная имеет свое уникальное имя.
Операции	- арифметические операции +,-,*,/ и др. ; - логические операции и, или, не; - операции отношения <, >, <, >, =, < >; - операция сцепки («присоединения», «конкатенации») символьных значений друг с другом с образованием одной длинной строки; изображается знаком «+».

Основные понятия алгоритмического программирования (данные, операторы, функции, процедуры и т. д.)

Выражения	**Предназначаются для выполнения необходимых вычислений, состоят из констант, переменных, указателей функций (например, exp(x)), объединенных знаками операций.** **Выражения записываются в виде линейных последовательностей символов (без подстрочных и надстрочных символов, «многоэтажных» дробей и т. д.), что позволяет вводить их в компьютер, последовательно нажимая на соответствующие клавиши клавиатуры.**
Операторы (команды)	**Текст любой программы состоит из отдельных предложений. Обычно они называются операторами. Как правило, оператор содержит имя и данные и указывает, какую операцию и над какими величинами надо выполнить. Одна строка программы может содержать один или несколько операторов,** **В состав операторов входят:** **- ключевые слова;** **- данные;** **- выражения и т, д.**

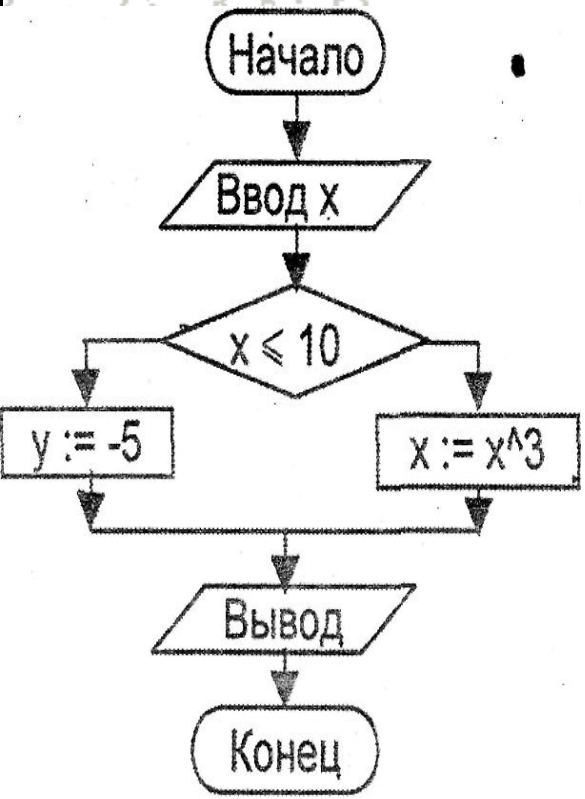
Основные понятия алгоритмического программирования (данные, операторы, функции, процедуры и т. д.)

Функции	**Для наиболее употребительных функций программы их вычисления записаны в память компьютера, в библиотеки программ, а функции включены в состав языков программирования. Такие функции называются встроенными (или стандартными). Для вычисления таких функций в программе достаточно указать имя функции и значение ее аргумента.** **Каждый язык программирования имеет свои стандартные функции.**
Процедуры	**Процедура - это самостоятельная программная единица, которая выполняется по команде из другой программы или процедуры.** **Процедура оформляется определенным образом, к ней можно обращаться из разных точек программы любое число раз. При этом такая процедура может решать каждый раз одну и ту же задачу с разными значениями исходных данных.**

Пример разработки алгоритма и программы для решения задачи, содержащей команды ветвления

Вычисление значения функции y(x) для заданного x:

$$y(x) = \begin{cases} -5, \text{ при } x \leqslant 10 \\ x^3, \text{ при } x > 10 \end{cases}$$

Блок-схема алгоритма	*Паскаль*
	**program zadacha;** **var x, y: real;** **begin** **{вычисление значения функции y(x)};** **writeln ('Введите значение x');** **readln (x);** **if x<= 10 then y:=-5 else y:= x*x*x;** **writeln ('y =', y)** **end.**

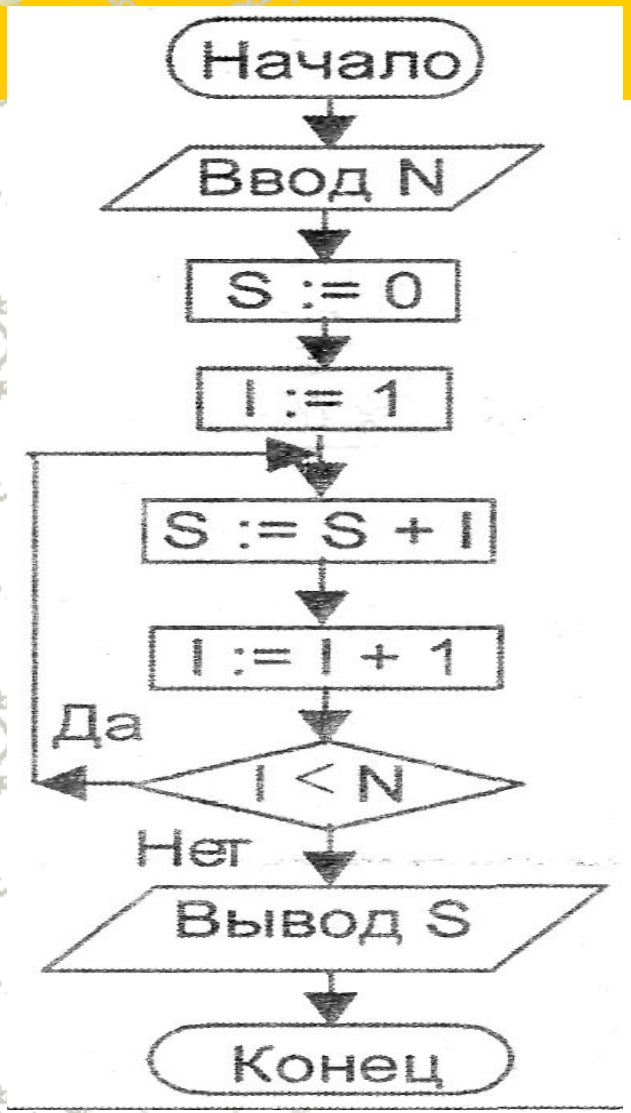
Пример разработки алгоритма и программы для решения задачи, содержащей команду повторения (оператор цикла)

Вычисление суммы натуральных чисел от 1 до N.

Комментарии

Натуральное число N вводится с клавиатуры в процессе выполнения программы. Искомая сумма обозначается идентификатором S. Параметром цикла является переменная i , которая изменяется от 1 до N с шагом 1, за счет чего и происходит перебор всех значений натуральных чисел от 1 до N . По окончании вычислений результат печатается на экране компьютера.

Блок-схема алгоритма	*Паскаль*

	program zadacha;
	var n, i, s : integer;
	begin
	writeln ('Введите натуральное число');
	readln (n);
	s:=0;
	i:= 1;
	Repeat
	begin
	s:= s + i;
	i:= i + 1;
	end
	Until i > n ;
	writeln ('Сумма натуральных чисел s = ', s)
	end.

Задача определения максимального элемента в одномерном массиве целых чисел

Исходные данные: **N - количество элементов в массиве,**

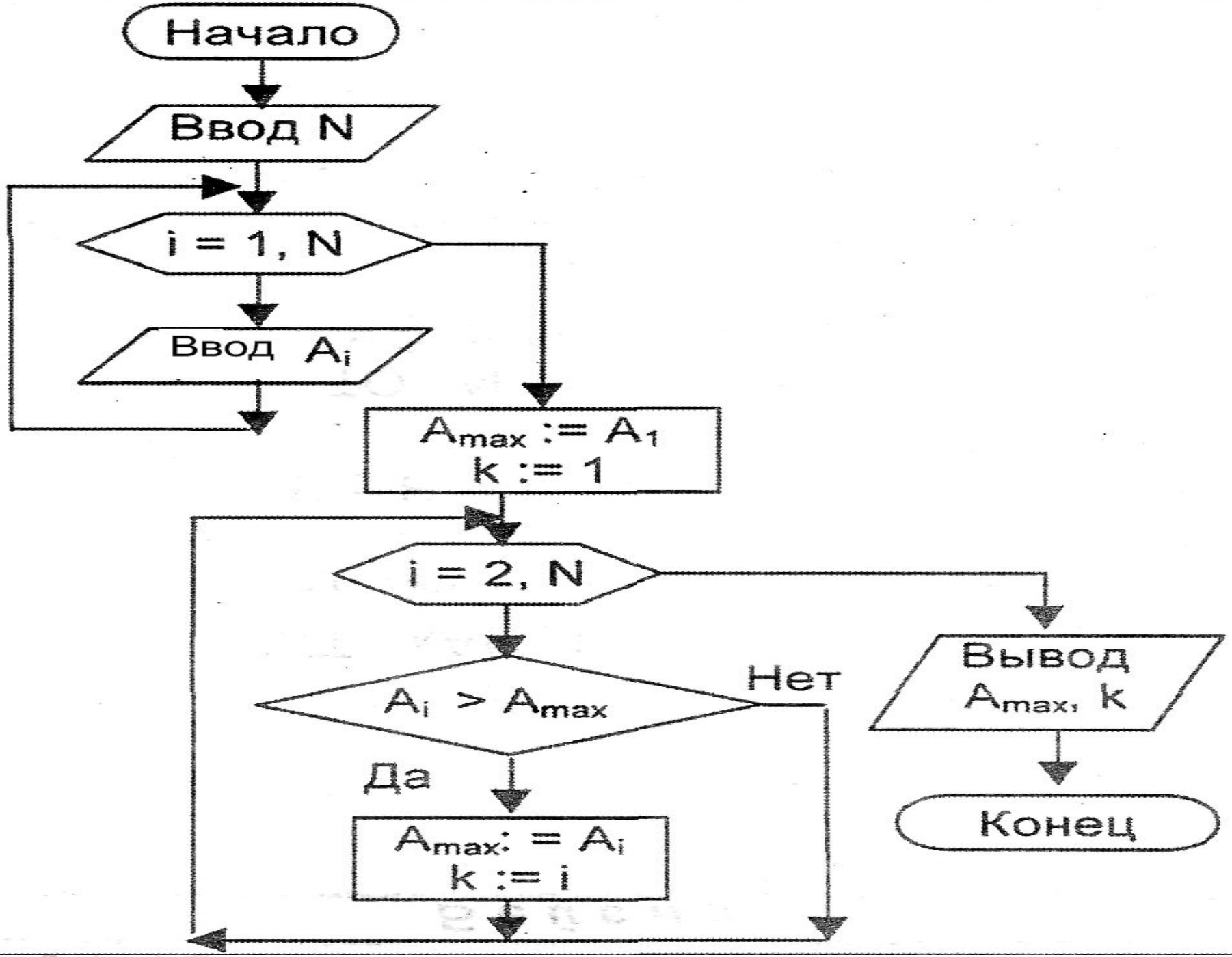
A - сам массив.

Результаты: **K - номер (индекс) того элемента в массиве, который оказался максимальным,**

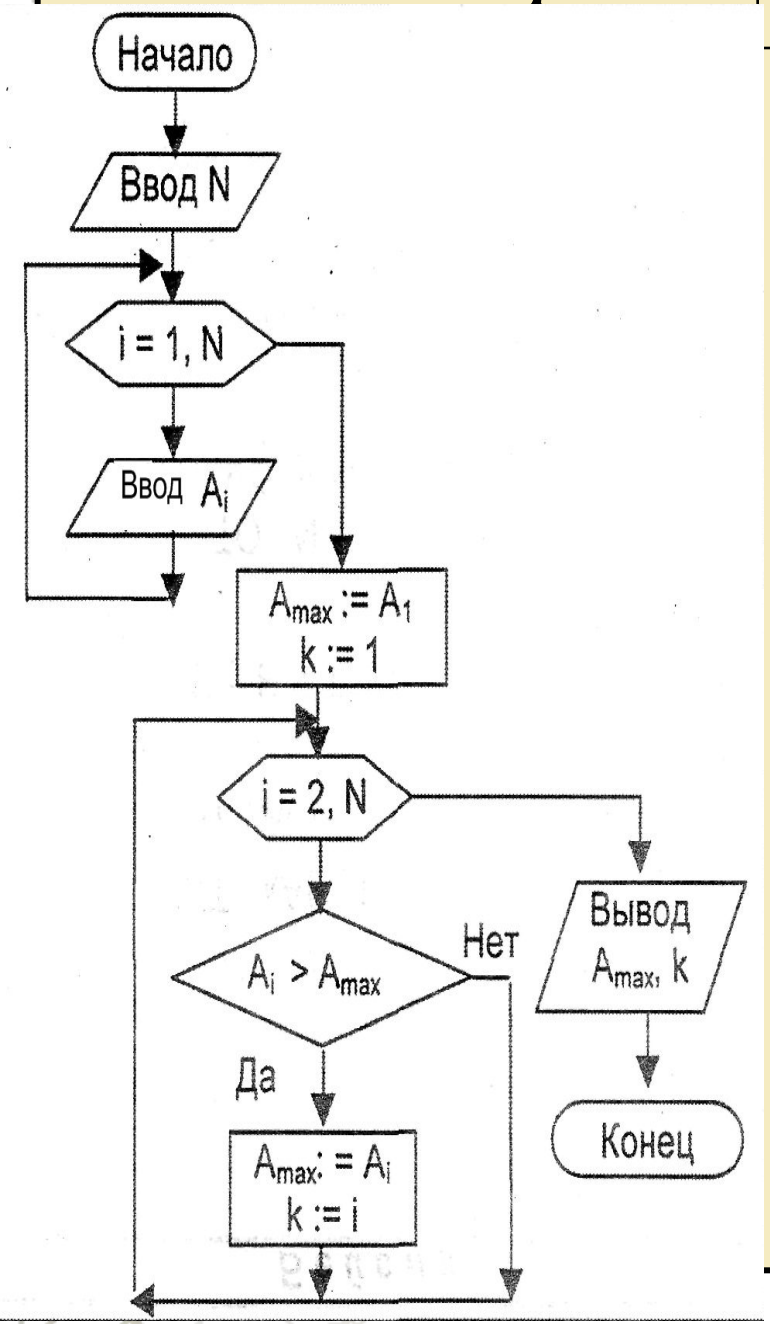
AMAX - значение максимального элемента.

Промежуточные величины:

I - текущий индекс элемента (переменная цикла). Поиск наибольшего значения осуществляется последовательным сравнением значений. В качестве начального значения наибольшего элемента массива выбирается значение первого элемента, а сам цикл, выполняющий сравнение, начинается со второго элемента.


Начало
Ввод N
i = 1, N
Ввод Ai
Amax := A1
k := 1
i = 2, N
Ai > Amax
Нет
Да
Amax: = Ai
k := i
Вывод
Amax, k
Конец

Блок-схема алгоритма



Паскаль

```
program max;
uses crt;
var    a  : array [1..20] of real; i, n, k : integer;
 amax : real;
begin  ClrScr;
write ('Введите N = '); readln(n);
for i := 1 to n do {Ввод элементов массива A}
begin
write ('a[' , i, ']='); readln (a[i]) end;
amax := a[1]; k := 1; {Поиск максимального
элемента} for i :=2 to n do
if a[i] > amax then begin
amax := a[i]; k:=i end; writeln;
writeln ('Наибольший элемент номер ', k);
writeln ('его значение', amax : 5 :1); readln
end.
```

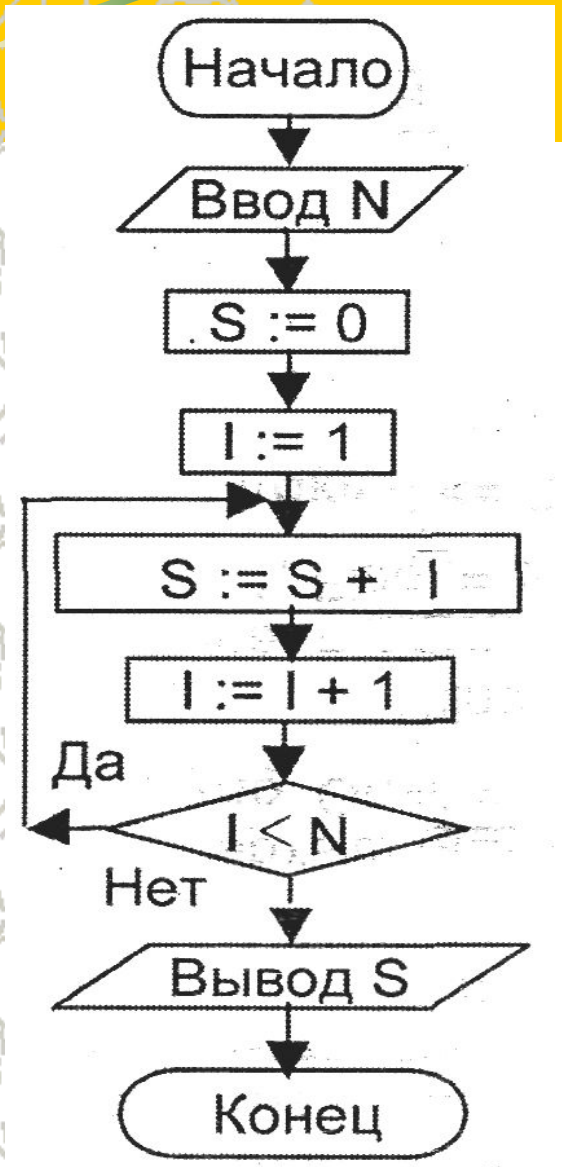
Пример решения расчетной задачи с использованием математических функций

«Вычисление значения выражения $s = 1 + \sqrt{2} + \sqrt{3} + \sqrt{4} + \ldots + \sqrt{n}$ **для заданного натурального числа n».**

Комментарии

Данная задача реализует пример классического циклического алгоритма. Интерес здесь представляет использование стандартной математической функции «корень квадратный».

В языках программирования существуют определенные правила записи таких функций. В частности, языки Бейсик и Паскаль требуют, чтобы рядом с именем стандартной функции (SQR или sqrt - соответственно) в скобках был указан аргумент функции, в нашем случае - это подкоренное выражение.

Блок-схема алгоритма	*Паскаль*
	**program zadacha; {вычисление суммы квадратных корней натуральных чисел}** **var n, i : integer; s : real;** **begin** **writeln ('Введите натуральное число');** **readln (n);** **s:=0;** **for i:= 1 to n do s := s + sqrt(i);** **writeln ('Сумма чисел s = ', s)** **end.**