

---

# Глава 2. Программирование

## §19. Алгоритмы обработки массивов

---

# Алгоритмы обработки массивов



*Практическая работа 19. Алгоритмы обработки массивов.*

## Сумма элементов массива

---

Представьте себе, что в массиве записаны зарплаты сотрудников фирмы, и требуется найти общую сумму, которая будет им выплачена. Для этого нужно сложить все числа, которые находятся в массиве.

Для того чтобы накапливать сумму, нужно ввести переменную, назовём её **sum**.

## Сумма элементов массива

---

Для решения задачи нужно выполнить перебор элементов массива в цикле. На каждом шаге цикла к значению *sum* добавляется значение очередного элемента массива.

Будем считать, что массив уже заполнен. Тогда сумму его элементов можно найти так:

```
sum:= 0
```

```
нц для i от 1 до N
```

```
    sum:= sum + A[i]
```

```
кц
```

```
Вывод sum
```

```
sum:= 0;
```

```
for i:=1 to N do
```

```
    sum:= sum + A[i];
```

```
write(sum);
```

# Сумма элементов массива

Покажем, как работает этот алгоритм для массива А:

	1	2	3	4	5
A	5	2	8	3	1

Выполним «ручную прокрутку» программы. Запишем в таблице выполняемые команды и изменение всех переменных:

	Оператор	i	sum
1	sum:= 0		0
2	i:= 1	1	
3	sum:= sum + A[1]		5
4	i:= i+1	2	
5	sum:= sum + A[2]		7
6	i:= i+1	3	
7	sum:= sum + A[3]		15
8	i:= i+1	4	
9	sum:= sum + A[4]		18
10	i:= i+1	5	
11	sum:= sum + A[5]		19

## Поиск элементов массива, удовлетворяющих условию

---

Во многих задачах нужно найти в массиве все элементы, удовлетворяющие заданному условию, и как-то их обработать. Простейшая из таких задач – подсчёт нужных элементов.

Для подсчёта событий используется переменная-счётчик, назовём её **count**. Перед началом цикла в счётчик записывается ноль (ни одного нужного события не произошло).

Если на очередном шаге цикла произошло нужное событие, значение счётчика увеличивается на единицу.

## Поиск элементов массива, удовлетворяющих условию

---

Подсчитаем количество чётных элементов массива (элементов с чётными значениями). Здесь нужное событие – это появление чётного элемента.

Поэтому условие «элемент  $A[l]$  – чётный» можно сформулировать иначе: «остаток от деления  $A[l]$  на 2 равен нулю»:

<pre>если <math>\text{mod}(A[i],2) = 0</math> то ...   увеличить счётчик все</pre>	<pre>if <math>A[i] \bmod 2 = 0</math> then ... { увеличить счётчик }</pre>
--	--

# Поиск элементов массива, удовлетворяющих условию

---

Теперь можно написать полный цикл:

```
count:= 0
нц для i от 1 до N
  если mod(A[i],2) = 0 то
    count:= count + 1
  все
кц
вывод count
```

```
count:= 0;
for i:=1 to N do
  if A[i] mod 2 = 0 then
    count:= count + 1;
write(count);
```

## Поиск элементов массива, удовлетворяющих условию

---

В массиве записан рост каждого члена баскетбольной команды в сантиметрах. Требуется найти средний рост игроков, которые выше 180 см.

Для решения задачи нам нужно считать и сумму, и количество элементов массива, которые больше 180:

Обратите внимание, что в теле условного оператора находятся две команды, поэтому в программе на языке Паскаль они заключаются в «операторные скобки» – ключевые слова **begin** и **end**.

# Поиск элементов массива, удовлетворяющих условию

---

```
count:= 0
sum := 0
нц для i от 1 до N
  если A[i] > 180 то
    count:= count + 1
    sum:= sum + A[i]
  все
кц
Вывод sum/count
```

```
count:= 0;
sum:= 0;
for i:=1 to N do
  if A[i] > 180 then begin
    count:= count + 1;
    sum:= sum + A[i]
  end;
write(sum/count);
```

---

# Поиск максимального элемента



***Практическая работа 22. Поиск максимального  
элемента.***

## Поиск максимального элемента в массиве

---

Представьте себе, что вы по очереди заходите в  $N$  комнат, в каждой из которых лежит арбуз. Вес арбузов такой, что вы можете унести только один арбуз. Возвращаться в ту комнату, где вы уже побывали, нельзя. Как выбрать самый большой арбуз?

## Поиск максимального элемента в массиве

---

Для хранения максимального элемента выделим в памяти целочисленную переменную  $M$ . Будем в цикле просматривать все элементы массива один за другим.

Если очередной элемент массива больше, чем максимальный из предыдущих, запомним новое значение максимального элемента в  $M$ .

## Поиск максимального элемента в массиве

---

Во-первых, можно записать в переменную  $M$  значение, заведомо меньшее, чем любой из элементов массива.

Например, если в массиве записаны натуральные числа, можно записать в  $M$  ноль.

Если же содержимое массива неизвестно, можно сразу записать в  $M$  значение  $A[1]$  (сразу взять первый арбуз), а цикл перебора начать со второго элемента:

# Поиск максимального элемента в массиве

---

```
M := A[1]  
нц для i от 2 до N  
  если A[i] > M то  
    M := A[i]  
  все  
кц  
вывод M
```

```
M := A[1];  
for i := 2 to N do  
  if A[i] > M then  
    M := A[i];  
write( M );
```

## Поиск максимального элемента в массиве

---

По номеру элемента  $i$  можно всегда определить его значение, оно равно  $A[i]$ . Поэтому достаточно хранить только номер максимального элемента  $nMax$ , тогда его значение равно  $A[nMax]$ :

```
nMax:= 1
нц для i от 2 до N
  если A[i] > A[nMax] то
    nMax:= i
  все
кц
ВЫВОД
'A[' ,nMax,']=',A[nMax]
```

```
nMax:= 1;
for i:= 2 to N do
  if A[i] > A[nMax] then
    nMax:= i;
write('A[' ,nMax,']=',A[nMax])
```

## Выводы:

---

- Для вычисления сумму элементов массива используется дополнительная переменная, в которой накапливается сумма. Начальное значение этой переменной равно нулю. Для добавления к сумме
- очередного элемента массива  $A[l]$  используют оператор вида
- $sum := sum + A[l]$
- При вычислении произведения начальное значение дополнительной переменной должно быть равно 1.
- Для подсчёта количества элементов, удовлетворяющих условию, нужно использовать переменную-счётчик. Начальное значение счётчика должно быть равно нулю. При обнаружении очередного нужного элемента счётчик увеличивается на 1:
- $count := count + 1$
- При поиске максимального значения в массиве используют вспомогательную переменную, в которой хранится максимальное из всех уже просмотренных значений. Сначала в эту переменную записывают значение первого элемента массива, а затем просматривают все элементы, начиная со второго.

# Интеллект-карта

---