

Добрый день, друзья

Рад приветствовать на третьей лекции курса
«Введение в программирование».





Ильнар Шафигуллин

- Методолог программы «Разработчик»
- Преподаватель мехмата Казанского Университета
- Стаж преподавания 10 лет
- Кандидат физико-математических наук



Ещё одна задача

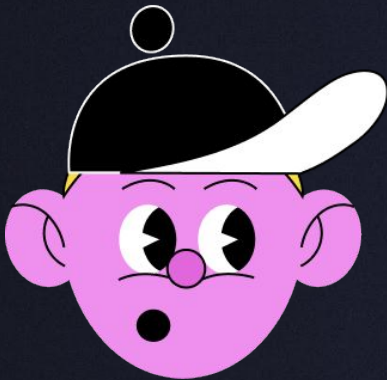
Нужно найти второе максимальное число.

- Представим, что у нас не 5 и не 10 чисел, а 1000 или 100000000

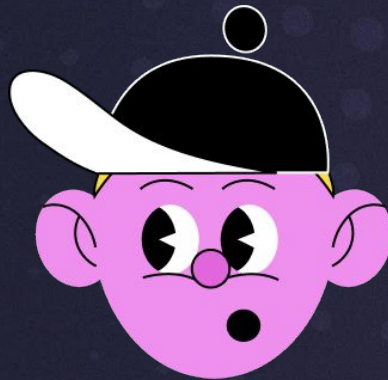


Для начала распишем алгоритм решения:

Находим
наибольшее число
в массиве



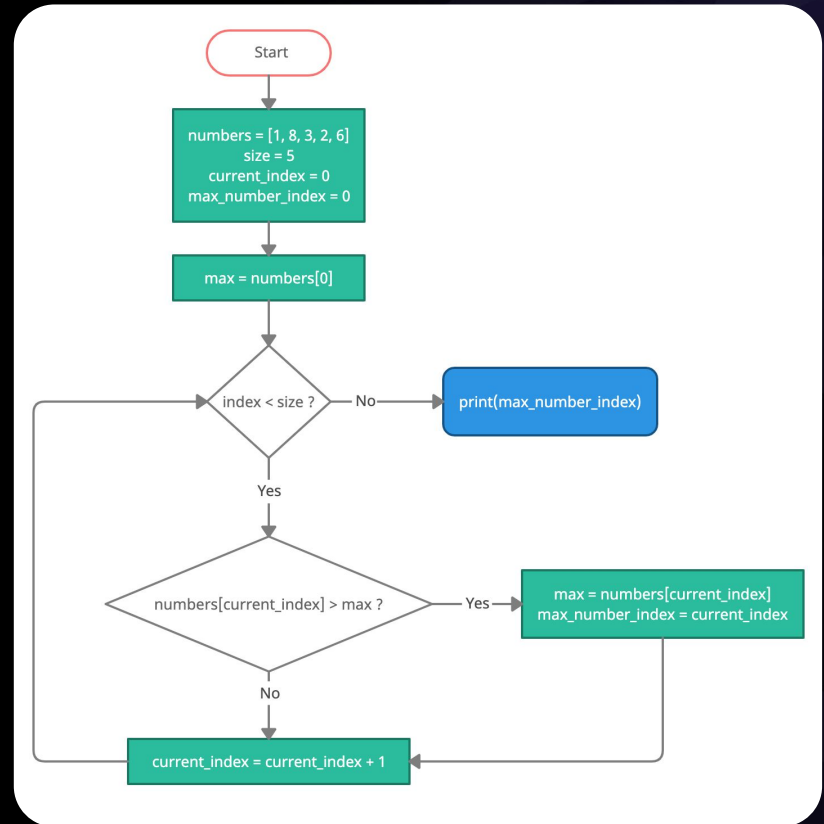
Исключаем
найденное число
из массива



Снова находим
максимальное
число в массиве



Для запоминания
индекса мы можем
записать нашу
программу так:



Для запоминания индекса мы можем записать нашу программу так:

```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 print(max)
12 print(current_index)
```



Самое сложное в программировании — именование переменных




Snake case: `max_number_index`



Camelcase: `MaxNumberIndex`






```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 0
max_number_index = 0
max = 1






```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

```
numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 0
max_number_index = 0
max = 1
numbers[current_index] = 1
```






```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 1
max_number_index = 0
max = 1
numbers[current_index] = 8






```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 1
max_number_index = 0
max = 1
numbers[current_index] = 8





```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10        current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19         current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 1
max_number_index = 1
max = 8
numbers[current_index] = 8



```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 0
max_number_index = 1
max = 8
numbers[current_index] = 1




```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11    current_index = 0
12    second_max = numbers[0]
13    if (max_number_index == 0) then
14        second_max = numbers[1]
15    while (current_index < size) do
16        if (current_index != max_number_index) then
17            if (numbers[current_index] > second_max) then
18                second_max = numbers[current_index]
19            current_index = current_index + 1
20    print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 0
max_number_index = 1
max = 8
numbers[current_index] = 1
second_max = 1



```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 current_index = 0
4 max_number_index = 0
5 max = numbers[0]
6 while (current_index < size) do
7     if (numbers[current_index] > max) then
8         max = numbers[current_index]
9         max_number_index = current_index
10    current_index = current_index + 1
11 current_index = 0
12 second_max = numbers[0]
13 if (max_number_index == 0) then
14     second_max = numbers[1]
15 while (current_index < size) do
16     if (current_index != max_number_index) then
17         if (numbers[current_index] > second_max) then
18             second_max = numbers[current_index]
19     current_index = current_index + 1
20 print(second_max)
```

numbers = [1, 8, 3, 2, 6]
size = 5
current_index = 4
max_number_index = 1
max = 8
second_max = 6



Время потехи или математического юмора

Шутка про чайник
и не очевидную логику.



Решение не всегда может
быть таким простым,
как может показаться
на первый взгляд.

**Костыли программирования
или как создать себе больше
проблем.**



Сложности также бывают разными.

Например, мы завязываем шнурки:

1

Необходимые

Нужно уметь завязать узелок и бантик.

2

Необязательные

Если мы хотим, завязывая шнурки, успеть ещё и попрыгать через скакалку.

3

Случайные

Если вы попытаетесь завязать шнурки в варежках.



Более эффективное решение задач.

Попробуем за один проход цикла решить нашу задачу с нахождением второго максимального числа.

Для начала нужно решить её для двух чисел:

```
1 if (numbers[0] > numbers[1]) then
2     first = numbers[0]
3     second = numbers[1]
4 else
5     first = numbers[1]
6     second = numbers[0]
```



Прекрасно! А теперь для трёх чисел:

```
1 first = numbers[0]
2 second = numbers[0]
3 if (numbers[1] > first) then
4     first = numbers[1]
5 else
6     second = numbers[1]
7 if (numbers[3] > first) then
8     second = first
9     first = numbers[3]
10 else
11     if (numbers[3] > second) then
12         second = numbers[3]
```



Ну и, конечно, теперь мы можем создать программу для любого количества чисел:

```
1 numbers = [1, 8, 3, 2, 6]
2 size = 5
3 first = numbers[0]
4 second = numbers[0]
5 if (numbers[1] > first) then
6     first = numbers[1]
7 else
8     second = numbers[1]
9 current_index = 2
10 while(current_index < size) do
11     if (numbers[current_index] > first) then
12         second = first
13         first = numbers[current_index]
14     else
15         if (numbers[current_index] > second) then
16             second = numbers[current_index]
17     current_index = current_index + 1
18 print(second)
```



Программу любой сложности можно представить в виде комбинации из трёх структур:

1

Следование

Последовательное выполнение инструкций.

2

Ветвление

Условия, которые позволяют перенаправить выполнение нашей программы по одной из имеющихся веток.

3

Цикл

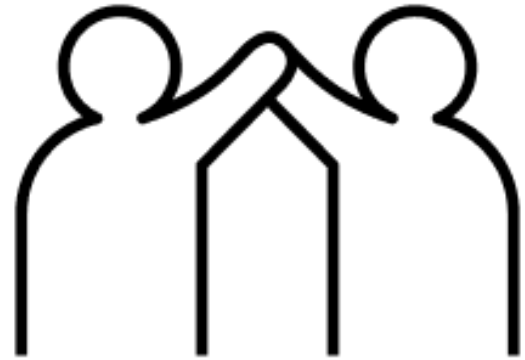
Конструкция, которая позволяет задавать многократное повторение операторов



Функции

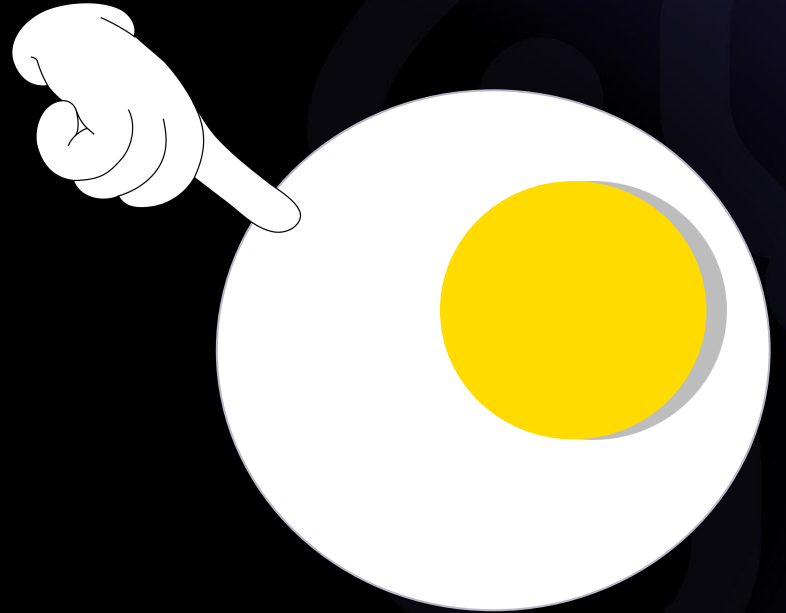
Например, нам нужно найти максимальное значение в двух различных массивах.

Мы будем проводить одну и ту же работу с массивами и результат получим одинаковый (найдем некоторое число)




ГОТОВИМ ЯИЧНИЦУ


- 1) Необходимо взять сковороду.
- 2) Поставить сковороду на плиту и включить огонь.
- 3) Смазать сковороду маслом.
- 4) Аккуратно разбить **несколько** яиц и вылить содержимое на горячую, смазанную маслом сковороду.
- 5) Посолить.
- 6) Накрыть крышкой и следить, чтобы яичница не подгорела.
- 7) Когда яичница будет готова, выключить плиту.



Бывают функции с параметрами и без.
Рассмотрим бытовую программу из вызова
двух последовательных функций:



Приготовь
яичницу



Помой
посуду



Чтобы в реальной жизни пользоваться такими функциями, нам необходимо дать подробную инструкцию. Разберем на примере поиска максимального числа в массиве:

```
1 function find_max(array):  
2   size = array.length  
3   index = 0  
4   max = array[0]  
5   while (index < size) do  
6     if (array[index] > max) then  
7       max = array[index]  
8       index = index + 1  
9   return max
```



Пример использования функции find_max

```
1 numbers = [1, 8, 3, 2, 6]
2 another_numbers = [15, 2, 74, 3, 8, 16, 24]
3 max_number = find_max(numbers)
4 another_max_number = find_max(another_numbers)
```



Все мы пользовались электрическим чайником

Но подавляющее большинство людей не задумывались о том,
что он тоже выполняет функции после включения.



Когда мы пользуемся функцией `find_second_max()`, для нас, как для программистов, не важно, как она реализована.

```
1 numbers = [1, 8, 3, 2, 6]
2 another_numbers = [15, 2, 74, 3, 8, 16, 24]
3 second_max_number = find_second_max(numbers)
4 another_second_max_number = find_second_max(another_numbers)
```



Всем спасибо

