**Практическая работа №7 Алгоритмическая сложность**

**Петухов Боев**

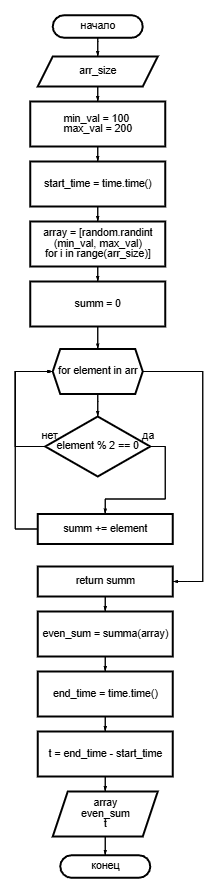
**Вариант 2**

Составить алгоритм поиска суммы четных элементов массива целых чисел. Размерность массива – 20. Заполнение массива осуществить случайными числами от 100 до 200.

**Словесный алгоритм**

1. Ввод размера массива
2. Инициализация диапазона случайных чисел
3. Замер времени начала работы программы
4. Генерация массива из случайных чисел
5. Вычисление суммы четных элементов:
6. Сначала сумма равна 0.
7. Каждый элемент массива проверяется на четность
8. Если элемент четный, он прибавляется к общей сумме.
9. Замер времени окончания работы программы
10. Вывод результатов

**Блок-схема алгоритма**

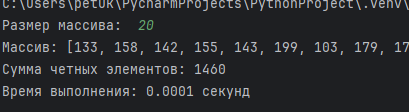


**Программа**

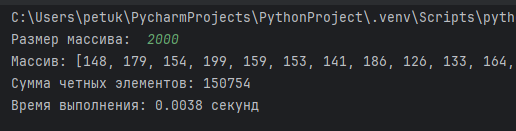
import random  
import time  
  
  
def summa(arr):  
 summ = 0  
 for element in arr:  
 if element % 2 == 0:  
 summ += element  
 return summ  
  
  
def rand():  
  
 arr\_size = int(input('Размер массива: '))  
 min\_val = 100  
 max\_val = 200  
  
 start\_time = time.time()  
  
 array = [random.randint(min\_val, max\_val) for i in range(arr\_size)]  
 even\_sum = summa(array)  
  
 end\_time = time.time()  
 t = end\_time - start\_time  
  
 print("Массив:", array)  
 print("Сумма четных элементов:", even\_sum)  
 print("Время выполнения: {:.4f} секунд".format(t))  
  
  
rand()

**Тестирование программы**

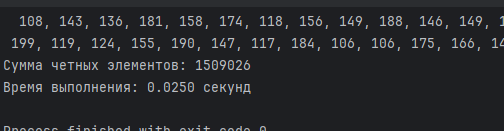
Размер массива: 20 (n)

****

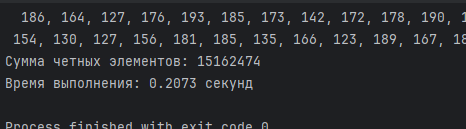
Размер массива: 2000 (n\*100)



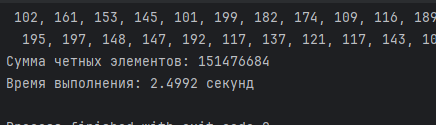
Размер массива: 20000 (n\*1000)

****

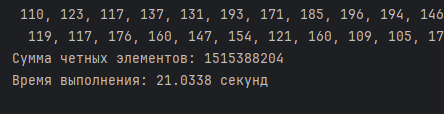
Размер массива: 200000 (n\*10000)



Размер массива: 2000000 (n\*100000)



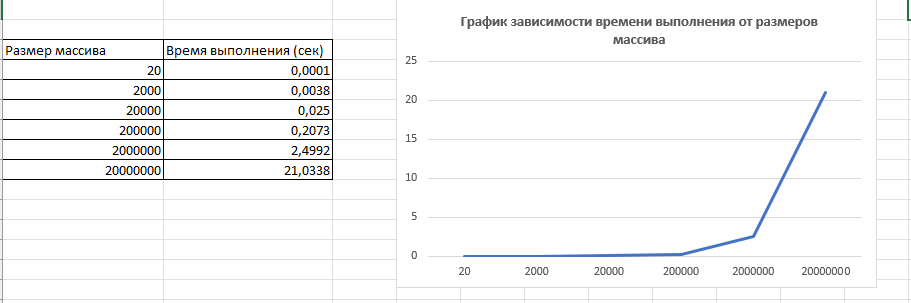
Размер массива: 20000000 (n\*1000000)



**Сложность алгоритма**

Время выполнения алгоритмов генерации чисел и подсчёта суммы четных чисел растет линейно с увеличением размера массива, значит программа имеет O(n) сложность

**График зависимости времени выполнения от размеров массива**

****