Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический   
университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Андреева А.Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

по лабораторной работе № 6

**«РЕКУРСИВНЫЕ ВЫЧИСЛЕНИЯ. МЕМОИЗАЦИЯ»**

по дисциплине «Функциональное и логическое программирование»

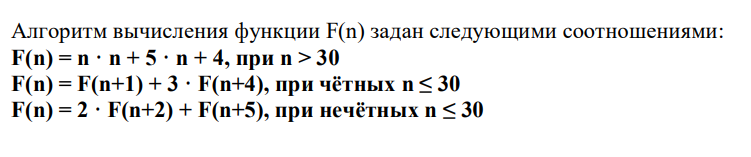
Студент группы ПИ-92 Савиков Д.А.

Преподаватель доцент, к.ф.-м.н. Андреева А.Ю.

Барнаул 2022

Номер в списке – 24

**Задание:** Построить рекурсивную функцию, выполнить замеры расхода памяти и времени .(datetime или timeit, sys.deep\_getsizeof()) Сделать реализацию с мемоизацией. Получить время и память. Сделать реализацию с декоратором @lru\_cache (модуль functools). Получить время и память.



**Решение:**

from functools import lru\_cache  
import timeit  
  
import tracemalloc  
  
def measure\_memory(func):  
 tracemalloc.start()  
  
 func()  
  
 current, peak = tracemalloc.get\_traced\_memory()  
 print(f"\n\033[37mFunction Name :\033[35;1m {func.\_\_name\_\_}\033[0m")  
 print(f"\033[37mCurrent memory usage:\033[36m {current / 10\*\*6}MB\033[0m")  
 print(f"\033[37mPeak :\033[36m {(peak / 10\*\*6) \* 1024}КБ\033[0m")  
 tracemalloc.stop()  
  
def f1(n):  
 *"""реализация рекурсивной функции """* if n > 30:  
 return n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 return 2 \* f1(n + 2) + 3 \* f1(n + 5)  
 *# чет* else:  
 return f1(n + 1) + 3 \* f1(n + 4)  
  
def f2\_wrapper(n):  
 mem\_dict = {}  
  
 def f2(n):  
 *"""реализация рекурсивной функции с мемоизацией"""* if mem\_dict.get(n) is not None:  
 return mem\_dict.get(n)  
  
 if n > 30:  
 mem\_dict[n] = n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 mem\_dict[n] = 2 \* f2(n + 2) + 3 \* f2(n + 5)  
 *# чет* else:  
 mem\_dict[n] = f2(n + 1) + 3 \* f2(n + 4)  
 return mem\_dict[n]  
  
  
@lru\_cache()  
def f3(n):  
 *"""реализация рекурсивной функции с @lru\_cache для мемоизации """* if n > 30:  
 return n \* n + 5 \* n + 4  
 *# нечет* elif n % 2:  
 return 2 \* f3(n + 2) + 3 \* f3(n + 5)  
 *# чет* else:  
 return f3(n + 1) + 3 \* f3(n + 4)  
  
def call\_func(alg\_name, func\_call\_string, import\_string):  
 print(alg\_name)  
 print("время: ", timeit.timeit(func\_call\_string, import\_string))  
  
@measure\_memory  
def profile\_func1():  
 f1(1000000)  
  
@measure\_memory  
def profile\_func2():  
 f2\_wrapper(1000000)  
  
@measure\_memory  
def profile\_func3():  
 f3(1000000)  
  
n=1000000  
  
call\_func("Рекурсивная функция", f"f1({n})", "from \_\_main\_\_ import f1")  
call\_func("Функция с мемоизацией", f"f2\_wrapper({n})", "from \_\_main\_\_ import f2\_wrapper")  
call\_func("Функция с @lru\_cache", f"f3({n})", "from \_\_main\_\_ import f3")  
print(f3.cache\_info())  
**Тесты:**

