|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Языки программирования»  Вариант 3 | | |
|  | | Работу выполнил  студент группы ПМИ-9,2023-2024 1 курса  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Быковец Н.С.  «\_\_\_» \_\_\_\_\_\_\_\_2024 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рубцова М.Б.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 3](#_Toc153130029)

[Код программы 5](#_Toc153130030)

# Постановка задачи

Напечатать в порядке возрастания первых n натуральных чисел, в разложение которых на простые множители входят только числа 2,3,5. Идея решения: введем три очереди x2, x3, x5 в которых будем хранить элементы, которые соответственно в 2, 3, 5 раз больше напечатанных, но еще не напечатаны. Рассмотрим наименьший из ненапечатанных элементов: пусть это x. Тогда он делится нацело на одно из чисел 2, 3, 5; x находится в одной из очередей и является в ней первым элементом (меньшие его уже напечатаны, а элементы очередей не напечатаны). Напечатав x, нужно изъять его из очереди и добавить в очередь кратные ему элементы. Длины очередей не превосходят числа напечатанных элементов. Изначально в очередях хранится по одному числу.

# Алгоритм решения

Инициализация структур данных: Создаются три очереди для хранения множителей 2, 3 и 5.

Главный цикл: Производится n итераций, где n — количество чисел, которые нужно найти. В каждой итерации:

1. Вычисляется минимальное значение среди всех очередей и удаляется из них.
2. Минимальное значение умножается на соответствующий множитель и добавляется обратно в очередь, если оно совпадает с минимальным значением.
3. Вычисляются простые множители минимального значения.
4. Результат выводится на экран.

Вычисление простых множителей: Для минимального значения вычисляются его простые множители, ограничиваясь числами 2, 3 и 5.

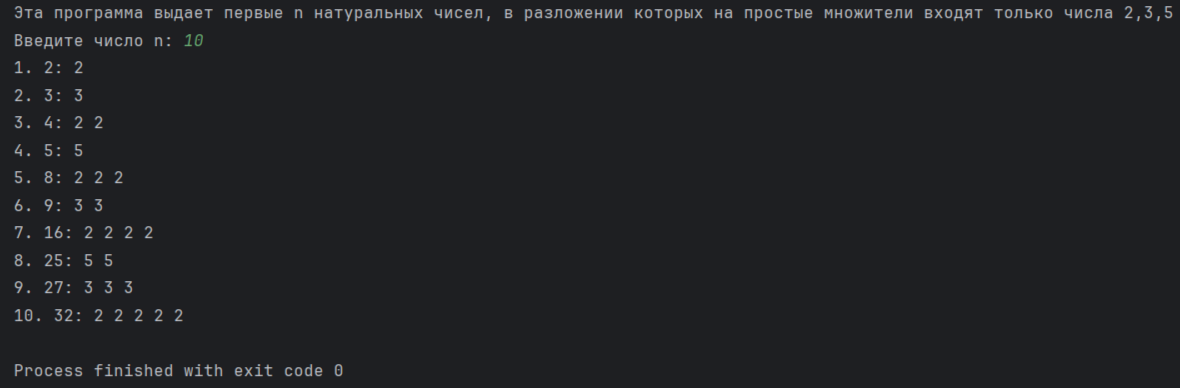
Вывод результатов: Выводятся первые n натуральных чисел, разложение которых на простые множители содержит только числа 2, 3 и 5, вместе с их множителями.

Выбранные структуры и типы данных:

1. Классы. Классы используются для создания пользовательских типов данных, которые могут включать в себя как данные, так и методы для работы с этими данными.
2. Связанные списки. Связанные списки используются для реализации очередей. В отличие от массивов, связанные списки позволяют легко добавлять и удалять элементы из очереди, не перемещая при этом другие элементы.

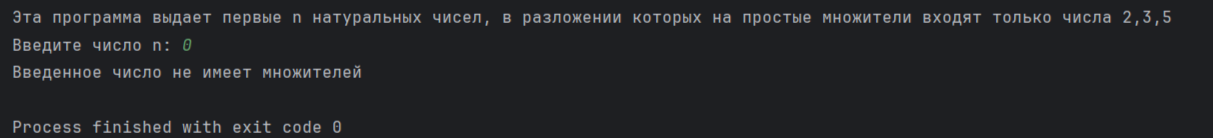
# Тестирование

1. Тестирование при корректных данных:

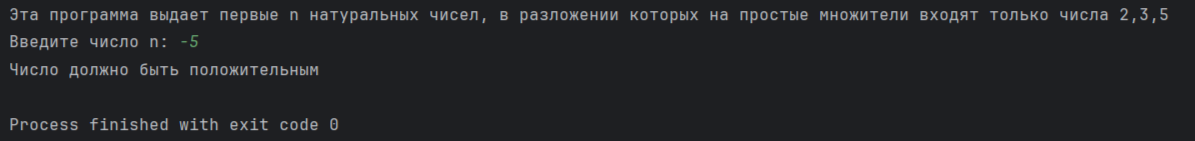


1. Тестирование при некорректных данных:

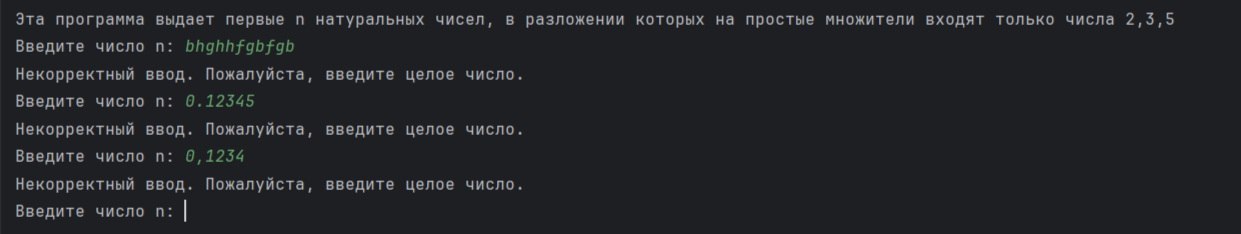
2.1



2.2



2.3



**Код программы**

class Node:  
 def \_\_init\_\_(self, value, next=None):  
 self.value = value  
 self.next = next  
  
  
class Queue:  
 def \_\_init\_\_(self):  
 self.head = None  
 self.tail = None  
  
  
 # Добавление элемента в конец очереди  
 def enqueue(self, value):  
 new\_node = Node(value)  
 if self.tail is None:  
 self.head = self.tail = new\_node  
 else:  
 self.tail.next = new\_node  
 self.tail = new\_node  
  
  
 # Удаление элемента из начала очереди  
 def dequeue(self):  
 if self.head is None:  
 return None  
 value = self.head.value  
 self.head = self.head.next  
 if self.head is None:  
 self.tail = None  
 return value  
  
  
# Функция для получения простых множителей числа  
def prime\_factors(number):  
 factors = [] # Список для хранения множителей  
 divisors = [2, 3, 5] # Множители для проверки  
 for divisor in divisors:  
 while number % divisor == 0:  
 factors.append(divisor)  
 number //= divisor  
 return factors  
  
  
# Функция для обновления очередей и нахождения минимального значения  
def get\_min\_and\_update\_queues(queues, multipliers):  
 # Получаем минимальные значения из очередей и удаляем их  
 min\_values = []  
 for i in range(len(queues)):  
 min\_values.append(queues[i].dequeue())  
  
 # Находим минимальное значение среди полученных  
 min\_value = min(min\_values)  
  
 # Обновляем очереди, добавляя в них новые значения  
 for i in range(len(queues)):  
 value = min\_values[i]  
 if value == min\_value:  
 # Если значение равно минимальному, добавляем в очередь произведение значения и множителя  
 queues[i].enqueue(value \* multipliers[i])  
 else:  
 # Иначе добавляем обратно удаленное значение  
 queues[i].enqueue(value)  
 return min\_value  
  
  
# Функция для вывода чисел с их множителями  
def print\_def(n):  
 # Создаем пустой список для хранения очередей  
 queues = []  
  
 # Добавляем в список три пустые очереди  
 for \_ in range(3):  
 queues.append(Queue())  
  
 # Инициализируем очереди, добавляя в них множители 2, 3 и 5  
 multipliers = [2, 3, 5]  
 for i in range(len(multipliers)):  
 multiplier = multipliers[i]  
 queues[i].enqueue(multiplier)  
  
 count = 1  
 for l in range(n): # Цикл для вывода n чисел  
 min\_value = get\_min\_and\_update\_queues(queues, [2, 3, 5]) # Получение минимального значения  
 factors = prime\_factors(min\_value) # Получение множителей  
 print(f"{count}. {min\_value}: {' '.join(map(str, factors))}")  
 count += 1  
  
  
# Обработка ввода пользователя и запуск функции  
print("Эта программа выдает первые n натуральных чисел, в разложении которых на простые множители входят только числа 2,3,5")  
while True:  
 try:  
 output = int(input("Введите число n: "))  
 if output == 0:  
 print("Введенное число не имеет множителей")  
 break  
 if output < 0:  
 print("Число должно быть положительным")  
 break  
 except ValueError:  
 print("Некорректный ввод. Пожалуйста, введите целое число.")  
  
print\_def(output)