|  |  |  |
| --- | --- | --- |
| 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 «Язык программирования Python»  Вариант 1 | | |
|  | | Работу выполнил  студент группы ИТ/О ИТ-8-2024 НБ 1 курса  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Казаков Н.Е.  «12» \_\_\_06\_\_\_2025 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рубцова М.Б.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc201108784)

[Алгоритм решения задачи. 3](#_Toc201108785)

[Тестирование 3](#_Toc201108786)

[Код программы 3](#_Toc201108787)

# Постановка задачи

Контейнеры. На складе хранятся контейнеры с товарами N различных видов. Все контейнеры составлены в N стопок. В каждой стопке могут находиться контейнеры с товарами любых видов (стопка может быть изначально пустой). Автопогрузчик может взять верхний контейнер из любой стопки и поставить его сверху в любую стопку. Необходимо расставить все контейнеры с товаром первого вида в первую стопку, второго вида — во вторую стопку и т.д. Программа должна вывести последовательность действий автопогрузчика или сообщение о том, что задача решения не имеет. Входные данные: в первой строке задается одно натуральное число N, не превосходящее 500. В следующих N строках описаны стопки контейнеров: сначала записано число ki — количество контейнеров в стопке, а затем ki чисел — виды товара в контейнерах в данной стопке, снизу вверх. В каждой стопке вначале не более 500 контейнеров (в процессе переноса контейнеров это ограничение может быть нарушено). Выходные данные: выведите описание действий автопогрузчика: для каждого действия укажите два числа — из какой стопки брать контейнер и в какую стопку класть. (Обратите внимание, что минимизировать количество операций автопогрузчика не требуется.) Если задача не имеет решения, выдайте одно число 0. Если контейнеры изначально правильно размещены по стопкам, выводить ничего не надо. Комментарий к примеру: изначально в первой стопке лежат четыре контейнера — снизу контейнер с товаром первого вида, над ним — с товаром второго вида, над ним — третьего, и сверху — еще один контейнер с товаром второго вида.

Примеры:

(входные данные)

3

4 1 2 3 2

0

0

(выходные данные)

1 2

1 3

1 2

# Алгоритм решения задачи.

Инициализация

* Получить количество стопок N (от 1 до 500)
* Создать N основных стопок
* Создать N временных стопок

Ввод данных

Для каждой стопки (от 1 до N):

* Получить количество контейнеров и их типы (снизу вверх)
* Заполнить стопку контейнерами

Проверка начального состояния

Если все контейнеры уже на своих местах:

* Завершить работу

Фаза 1: Распределение по временным стопкам

Для каждой основной стопки (i):

* Пока стопка не пуста:

1. Взять верхний контейнер
2. Определить целевую стопку
3. Переместить контейнер во временную стопку
4. Зафиксировать операцию перемещения

Фаза 2: Перемещение в основные стопки

Для каждой временной стопки (i):

* Пока стопка не пуста:
  + 1. Взять верхний контейнер
    2. Переместить его в основную стопку i
    3. Зафиксировать операцию перемещения

Проверка результата

Если все контейнеры на своих местах:

* Вывести последовательность операций

Иначе:

* Сообщить, что решение невозможно

Вывод результатов

* Отобразить все выполненные операции
* Показать конечное расположение контейнеров

# Тестирование

Тест 1: Простой случай

Вход:

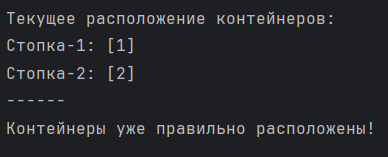
2

1 1

1 2

Ожидаемый вывод: Контейнеры уже правильно расположены

Вывод программы:

****

Тест 2: Пустые стеки

Вход:

3

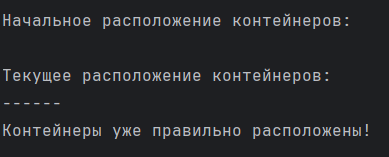
0

0

0

Ожидаемый вывод: Контейнеры уже правильно расположены

Вывод программы:



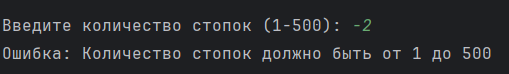
Тест 3: Отрицательный N

Вход:

-2

Ожидаемый вывод: Ошибка

Вывод программы:



Тест 4: Недопустимый тип контейнера

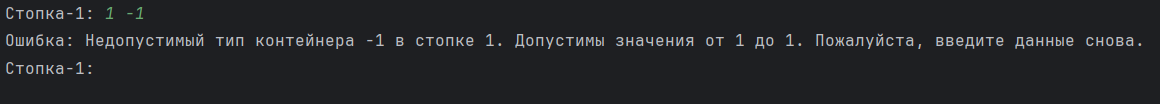
Вход:

1

1 -1

Ожидаемый вывод: Ошибка

Вывод программы:



Тест 5: Отрицательное количество контейнеров

Вход:

1

-1 1

Ожидаемый вывод: Ошибка

Вывод программы:



Тест 6: Некорректное количество контейнеров

Вход:

1

a

Ожидаемый вывод: Ошибка

Вывод программы:



Тест 7: Несоответствие количества контейнеров

Вход:

1

1 2 3

Ожидаемый вывод: Ошибка

Вывод программы:



Тест 8: Большие значения

Вход:

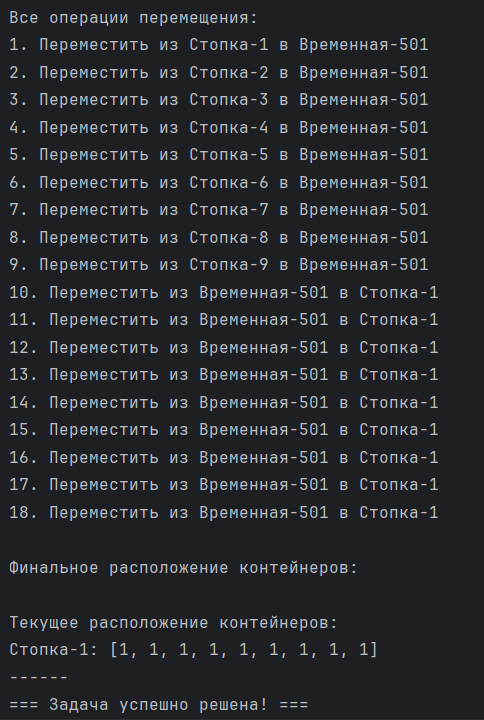
500

Девять строк с 1 1

Остальное 0

Ожидаемый вывод: Последовательность операций

Вывод программы:

****

Тест 9: Правильная задача

Вход:

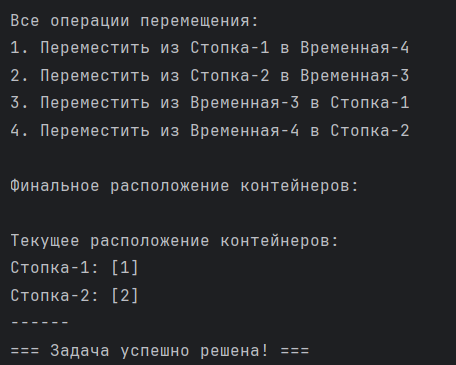
2

1 2

1 1

Ожидаемый вывод: Последовательность операций

Вывод программы:

****

# Код программы

MAX\_STACKS = 500  
MAX\_CONTAINERS = 500  
  
  
class Stack:  
 """Класс для работы со стопками контейнеров."""  
  
 def \_\_init\_\_(self, name):  
 self.items = []  
 self.name = name  
  
 def push(self, item):  
 self.items.append(item)  
  
 def pop(self):  
 if not self.is\_empty():  
 return self.items.pop()  
 raise IndexError("Попытка взять контейнер из пустой стопки")  
  
 def peek(self):  
 if not self.is\_empty():  
 return self.items[-1]  
 return None  
  
 def is\_empty(self):  
 return len(self.items) == 0  
  
 def size(self):  
 return len(self.items)  
  
 def \_\_str\_\_(self):  
 return f"{self.name}: {self.items}"  
  
  
def print\_stacks(main\_stacks, temp\_stacks):  
 """Выводит текущее состояние стопок."""  
 print("\nТекущее расположение контейнеров:")  
 for stack in main\_stacks + temp\_stacks:  
 if not stack.is\_empty():  
 print(f"{stack.name}: {stack.items}")  
 print("------")  
  
  
def validate\_input(parts, stack\_num, num\_stacks):  
 """Проверяет корректность введенных данных."""  
 if not parts:  
 raise ValueError(f"Пустой ввод для стопки {stack\_num}")  
  
 try:  
 ki = int(parts[0])  
 except ValueError:  
 raise ValueError(f"Некорректное количество контейнеров в стопке {stack\_num}")  
  
 if ki < 0:  
 raise ValueError(f"Отрицательное количество контейнеров в стопке {stack\_num}")  
  
 if len(parts) - 1 != ki:  
 raise ValueError(  
 f"Несоответствие количества контейнеров в стопке {stack\_num}. "  
 f"Ожидалось {ki}, получено {len(parts) - 1}"  
 )  
  
 for elem in parts[1:]:  
 try:  
 elem\_int = int(elem)  
 except ValueError:  
 raise ValueError(f"Некорректный тип контейнера в стопке {stack\_num}")  
  
 if elem\_int < 1 or elem\_int > num\_stacks:  
 raise ValueError(  
 f"Недопустимый тип контейнера {elem} в стопке {stack\_num}. "  
 f"Допустимы значения от 1 до {num\_stacks}"  
 )  
  
  
def solve\_containers():  
 """Основная функция решения задачи."""  
 print("=== Программа для расстановки контейнеров ===")  
 print("Условие: нужно расставить контейнеры так, чтобы в i-й стопке")  
 print("находились только контейнеры типа i\n")  
  
 try:  
 num\_stacks = int(input("Введите количество стопок (1-500): "))  
 if num\_stacks < 1 or num\_stacks > MAX\_STACKS:  
 raise ValueError(f"Количество стопок должно быть от 1 до {MAX\_STACKS}")  
 except ValueError as e:  
 print(f"Ошибка: {e}")  
 return  
  
 main\_stacks = [Stack(f"Стопка-{i + 1}") for i in range(num\_stacks)]  
 temp\_stacks = [Stack(f"Временная-{num\_stacks + i + 1}") for i in range(num\_stacks)]  
  
 print("\nВведите данные для каждой стопки:")  
 print("Формат: <количество> <типы контейнеров снизу вверх>")  
 print("Пример: 3 1 2 1 - стопка с 3 контейнерами: 1 (низ), 2, 1 (верх)")  
  
 for i in range(num\_stacks):  
 while True:  
 try:  
 input\_str = input(f"Стопка-{i + 1}: ").strip()  
 parts = list(map(str, input\_str.split())) if input\_str else ['0']  
  
 validate\_input(parts, i + 1, num\_stacks)  
  
 ki = int(parts[0])  
 for elem in parts[1:]:  
 main\_stacks[i].push(int(elem))  
 break  
  
 except ValueError as e:  
 print(f"Ошибка: {e}. Пожалуйста, введите данные снова.")  
 except Exception as e:  
 print(f"Ошибка: {e}. Пожалуйста, введите данные снова.")  
  
 print("\nНачальное расположение контейнеров:")  
 print\_stacks(main\_stacks, temp\_stacks)  
  
 def is\_correct():  
 """Проверяет правильность расположения контейнеров."""  
 for i in range(num\_stacks):  
 for elem in main\_stacks[i].items:  
 if elem != i + 1:  
 return False  
 return True  
  
 if is\_correct():  
 print("Контейнеры уже правильно расположены!")  
 return  
  
 operations = []  
  
 try:  
 # Фаза 1: Распределение по временным стопкам  
 for i in range(num\_stacks):  
 while not main\_stacks[i].is\_empty():  
 elem = main\_stacks[i].pop()  
 if elem < 1 or elem > num\_stacks:  
 raise ValueError(f"Недопустимый тип контейнера {elem}")  
  
 target\_stack = elem - 1  
 temp\_stacks[target\_stack].push(elem)  
 op = f"Переместить из {main\_stacks[i].name} в {temp\_stacks[target\_stack].name}"  
 operations.append(op)  
 print(f"Операция {len(operations)}: {op}")  
 print\_stacks(main\_stacks, temp\_stacks)  
  
 # Фаза 2: Перемещение в основные стопки  
 for i in range(num\_stacks):  
 while not temp\_stacks[i].is\_empty():  
 elem = temp\_stacks[i].pop()  
 main\_stacks[i].push(elem)  
 op = f"Переместить из {temp\_stacks[i].name} в {main\_stacks[i].name}"  
 operations.append(op)  
 print(f"Операция {len(operations)}: {op}")  
 print\_stacks(main\_stacks, temp\_stacks)  
  
 if not is\_correct():  
 print("\nРезультат: задача не имеет решения")  
 return  
  
 print("\nВсе операции перемещения:")  
 for i, op in enumerate(operations, 1):  
 print(f"{i}. {op}")  
  
 print("\nФинальное расположение контейнеров:")  
 print\_stacks(main\_stacks, temp\_stacks)  
 print("=== Задача успешно решена! ===")  
  
 except IndexError as e:  
 print(f"\nОшибка: {e}")  
 except Exception as e:  
 print(f"\nОшибка: {e}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 solve\_containers()