|  |  |  |
| --- | --- | --- |
| 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 \* | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 1 курса  Кулаков М.И.  «10» июня 2024 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 3](#_Toc153130029)

[Код программы 3](#_Toc153130030)

[Инструкция по применению стилей и оформлению работы 4](#_Toc153130031)

# Постановка задачи

Разработать и реализовать программу "Хитрый купец», которая на основании определенного правила, определяет, на какие места купец должен поставить свой товар, что бы не потерять его.

Ввод пользовательский данных осуществляется с клавиатуры в программу реализовать при помощи функций добавления, дальнейшая работа должна быть реализована при помощи методов класса. Интерфейс пользователя удобнее организовать в виде меню с выбором соответствующего действия. В программе обязательно наличие дружественного интерфейса и защиты от некорректного пользовательского ввода. Использовать циклический однонаправленный список

# Алгоритм решения

1. Создание класса RuleData\_List для хранения и использования пользовательских данных.

2. Создание класса Cargo\_Box для работы с отдельным «тюком».

3. Создание класса LinkedList для работы с однонаправленным циклическим списком объектов класса Cargo\_Box.

4. Создание метода adding\_cargo\_box для класса LinkedList, создающего однонаправленный циклический список элементов класса Cargo\_Box .

5. Создание метода removing\_cargo\_box для класса LinkedList, удаляющего из списка половину элементов, которые имеются в списке, используя пользовательские данные из объекта класса RuleData\_List

6. Создание метода counting\_boxes для класса LinkedList, считающего номера «удачных» мест, после удаления, согласно правилу и пользовательским параметрам

7. Создание основной функции main.

8. Создание функции input\_data, для ввода пользователем параметров работы.  
  
9. Создание вложенной функции quanity\_cargo\_box в теле функции input\_data, для ввода пользователем кол-ва «тюков»

10. Создание вложенной функции current\_box в теле функции input\_data, для ввода пользователем нынешнего месяца

11. Создание вложенной функции current\_day в теле функции input\_data, для ввода пользователем дня нынешнего месяца

12. Создание вложенной функции indentation в теле функции input\_data, для промежутка между выкидываемыми «тюками»  
  
13. Тестирование программы, отладка работы, устранение неполадок

# Тестирование

В ходе тестирования, программа получала на вход разные входные данные, и выдавала различные ответы:

|  |  |  |
| --- | --- | --- |
| Вхожные данные | Комментарий | Ответ программы |
| 10  3  3  5 | Корректные входные данные, если месяц не февраль | 2 5 6 7 10 13 14 17 18 19 |
| 15  2  29  1  5 | Корректные входные данные, если выбран февраль, а год — високосный (при не високосном 4 сверху цифра должна быть 0, а 3 — не больше 28) | 1 2 3 6 9 10 12 13 18 19 20 22 24 26 27 |
| Sdgadg  adgag  asgsag  asga | Некорректные входные данные, введены буквы (строки), а не числа. | Введённые вами данные содержат параметры, несовместимые с условиями |
| -10  2  29  1 | Некорректные входные данные, в кол-ве тюков указано отрицательное число |
| 10  99  2  6 | Некорректные входные данные, в номере месяца указано не подходящее по условию число |
| 10  3  -27  3 | Некорректные входные данные, в номере дня указано не подходящее по условию число |
| 10  3  1  100 | Корректные входные данные. Промежуток может быть больше кол-ва всех тюков, берётся остаток от деления на | 11 12 13 14 15 16 17 18 19 20 |
| 10  3  1  -2 | Некорректные входные данные, в промежутке указано отрицательное число, а список однонаправленный | Введённые вами данные содержат параметры, несовместимые с условиями |

# Код программы

""" «Хитрый купец» Два купца отправились торговать за море. Каждый из них

повез по N одинаковых тюков с товаром. В пути корабль попал в шторм и дал

течь. Чтобы корабль не затонул, капитан приказал поднять весь груз на палубу

корабля, расставить его вдоль бортов по периметру и выбросить за борт

половину груза. Груз выбрасывается по следующему правилу:

• каждый тюк получает порядковый номер (+), начиная с тюка, стоящего на

носу корабля; (-)

• их нумерация осуществляется по часовой стрелке; (-)

• номер первого выброшенного тюка соответствует текущему числу

месяца M; (+)

• через K тюков следующий тюк с товаром выбрасывается. Один из

купцов прознал про это правило. Как необходимо расставить ему груз,

чтобы ни один из его тюков не был выброшен? (+)

Использовать циклический однонаправленный список.

"""

class Cargo\_Box: # Класс тюков. Объект имеет след. атрибуты: номер, и указатель на след. тюк (если обходится без указателя на следующий тюк, то можно вообще не создавать этот класс, а это меняет вс. структуру программы)

def \_\_init\_\_(self):

self.number = None

self.next = None

class LinkedList: # Класс связанного, одноправленного списка

def \_\_init\_\_(self):

self.head = None

self.firts\_el = None

self.previous = None

def adding\_cargo\_box(self, RuleData): # Метод добавления в список объекта класса Cargo\_Box

for i in range(RuleData.quanity\*2):

new\_box = Cargo\_Box()

new\_box.number = i+1

#print(new\_box, new\_box.number)

if i == 0:

#print('first')

self.first\_el = new\_box

self.head = new\_box

elif i+1 == RuleData.quanity\*2:

#print('last')

#print(new\_box, self.first\_el, new\_box.next)

self.head.next = new\_box

new\_box.next = self.first\_el

self.head = self.first\_el

#print(new\_box, self.first\_el, new\_box.next, self.head)

else:

#print('medium')

#print(self.head, new\_box)

self.head.next = new\_box

#print(self.head, self.head.next)

self.head = new\_box

def removing\_cargo\_box(self, RuleData): # Метод удаления в список объекта класса Cargo\_Box

check = 0; self.head = self.first\_el

while self.head.number != RuleData.day:

self.head = self.head.next

indentation\_control = 0

while check != RuleData.quanity:

if indentation\_control == 0:

#print ('Удаление')

if self.head == self.first\_el:

self.first\_el = self.first\_el.next

current\_box = self.head

while current\_box.next != self.head:

current\_box = current\_box.next

#print(current\_box.number, self.head.number, current\_box.next.number, current\_box.next.next.number)

current\_box.next = current\_box.next.next

self.head = current\_box.next

indentation\_control = RuleData.indentation

check += 1

else:

#print('Пропуск', self.head.number)

indentation\_control -= 1

self.head = self.head.next

def counting\_boxes(self, RuleData): # Метод подсчета оставшихся (счастливых) мест для тюков.

current\_box = self.first\_el; luck\_box = []; check = 0

""" print( RuleData.quanity,

RuleData.month,

RuleData.day,

RuleData.indentation)

print(self.first\_el, 'первый элемент', RuleData.indentation) """

while check != RuleData.quanity:

#print(current\_box.number, 'То что будет добавлено')

luck\_box.append(current\_box.number)

current\_box = current\_box.next

check += 1

return luck\_box

class RuleData\_List: # Класс, содержащий параметры для определения тюков и выполнения действий с ними

def \_\_init\_\_(self):

self.quanity = None

self.month = None

self.day = None

self.indentation = None

def main(): # Основная функция

def input\_data(): # Функция ввода данных, исходя из которых будут выбрасываться и оставляться тюки

print('\n','\n',

'Вам нужно ввести данные, исходя из которых будут выбираться места для тюков, которые не будут выброшены.\n'

'Сначала вы ввводите все данные, после чего они отправляются на проверку. Если данные подойдут, то программа продолжит работать.\n'

'В противном случае, если какие-то данные не подошли и вы допустили ошибку - программа остановится после общей проверки данных и сообщит вам об том, что вы ввели некорректные данные.\n'

'Внимательно читайте сообщения - они содержат важную информацию, по правилам ввода данных.'

'\n','\n'

'Правила игры:\n'

'• каждый тюк получает порядковый номер (+), начиная с тюка, стоящего на носу корабля;\n'

'• их нумерация осуществляется по часовой стрелке;\n'

'• номер первого выброшенного тюка соответствует текущему числу месяца M;\n'

'• через K тюков следующий тюк с товаром выбрасывается.\n'

'\n','\n',

'---------------------------------------------------------------- \* ------------------------------------------------------------------')

def quanity\_cargo\_box(): # Фунция ввода кол-ва тюков у одного купца

try:

quanity = int(input('Введите целое, неотрицательное число больше нуля, равное количеству тюков у одного купца, у другого будет столько же: '))

except(ValueError, TypeError):

condition = False

quanity = None

else:

if quanity <= 0:

condition = False

quanity = None

else:

condition = True

print('\n')

return quanity, condition

def current\_month(): # Функция выбора месяца

try:

month = int(input('Введите целое число от 1 до 12, чей номер соответствует месяцу М в григорианском календаре: '))

except(ValueError, TypeError):

condition = False

month = None

else:

if month < 1 or month > 12:

condition = False

month = None

else:

condition = True

print('\n')

return month, condition

def current\_day(month, quanity): # Функция выбора дня

try:

day = int(input('Введите текущий день месяца М. Учтите, что в 1, 3, 5, 7, 8, 10 и 12 месяцах - 31 день. В 4, 6, 9, 11 - 30. В 2 месяце в зависимости от года - 28 или 29. Также, номер не должен быть больше, чем кол-во тюков у купцов: '))

except(ValueError, TypeError):

condition = False

day = None

else:

if quanity is not None:

if day >= quanity\*2:

condition = False

day = None

else:

if month in [1, 3, 5, 7, 8, 10, 12]:

if day < 1 or day > 31:

condition = False

else:

condition = True

elif month in [4, 6, 9, 11]:

if day < 1 or day > 30:

condition = False

else:

condition = True

elif month == 2:

try:

leap\_year = int(input('Вы выбрали 2 месяц - это февраль. В зависимости от того, является год висикосным или нет, зависит кол-во дней в месяце. Введите 1 если год високосный, если нет введите 0: '))

except(ValueError, TypeError):

condition = False

else:

if leap\_year == 1:

if day < 1 or day > 29:

condition = False

else:

condition = True

elif leap\_year == 0: # Выбор года - високосный или нет

if day < 1 or day > 28:

condition = False

else:

condition = True

else:

condition = False

else:

condition = False

else:

condition = False

return day, condition

def indentation(quanity): # Функция для определения промежутка К пропущенных тюков

try:

indentation\_value = int(input('Введите целое, неотрицательное К, больше 0 - кол-во тюков, которые будут пропущены, прежде чем следующий будет выкинут за борт: '))

except(ValueError, TypeError):

condition = False

indentation\_value = None

else:

if indentation\_value < 0:

condition = False

else:

if quanity is not None:

if indentation\_value >= quanity\*2:

indentation\_value = indentation\_value % (quanity\*2) # Если промежуток больше, чем кол-во тюков, то нужно брать остаток от деления такого промежутка на кол-во тюков

condition = True

else:

condition = False

indentation\_value = None

return indentation\_value, condition

RuleData = RuleData\_List()

quanity, quanity\_condition = quanity\_cargo\_box()

month, month\_condition = current\_month()

day, day\_condition = current\_day(month, quanity)

indentation\_value, indentation\_condition = indentation(quanity)

print('\n','\n',

'---------------------------------------------------------------- \* ------------------------------------------------------------------',

'\n','\n',)

print(quanity, day, month, indentation\_value)

if quanity\_condition is True and month\_condition is True and day\_condition is True and indentation\_condition is True:

RuleData.quanity = quanity

RuleData.month = month

RuleData.day = day

RuleData.indentation = indentation\_value

return RuleData

else:

return None

print('---------------------------------------------------------------- \* ------------------------------------------------------------------',

'\n','\n','\n',

'Здравствуйте, это игра "Хитрый купец"'

'\n','\n'

'Два купца, имея одинаковое количество абсолютно идентичных тюков с товаром, отправились торговать за море.\n'

'В один из дней, в море разыгрался шторм и корабль, на котором ехали купцы, был повреждён.\n'

'Что бы спасти положение, капитан судна приказал избавится от лишней нагрузки на корабль и выбросить за борт половину товара купцов.\n'

'Для этого, он решил выкидывать тюки по особому правилу, о котором один из купцов прознал и решил спасти свой товар.\n'

'Вы сможете задать в этой программе условия и узнать, как купец должен расположить свой товар так, что бы полностью его спасти.\n'

'\n','\n',

'---------------------------------------------------------------- \* ------------------------------------------------------------------')

status = input\_data()

if status is not None:

print(status.quanity, status.day, status.month, status.indentation)

CargoList = LinkedList()

LinkedList.adding\_cargo\_box(CargoList, status)

LinkedList.removing\_cargo\_box(CargoList, status)

result = (list(dict.fromkeys(LinkedList.counting\_boxes(CargoList, status))))

for i in result:

print(i, end=' ')

print('- Места, на которые хитрому купцу надо поставить свои тюки с товаром',

'\n'

'---------------------------------------------------------------- \* ------------------------------------------------------------------')

return None

else:

print('Введённые вами данные содержат параметры, несовместимые с условиями')

return None

if \_\_name\_\_ == '\_\_main\_\_': # Вызов основной функции

main()

# Инструкция по применению стилей и оформлению работы

Для оформления частей отчёта следует использовать заранее созданные стили. Все стили, которые могут пригодиться начинаются с «ЛР. 1 …».



**ЛР. 1 Обычный** – для оформления текста задания и алгоритма решения.

**ЛР. 1 Текст программы** – для оформления кода программы.

**ЛР. 1 Заголовок 1** – заголовок первого уровня (для того, чтобы озаглавить основные разделы отчета).

**ЛР. 1 Заголовок 2** – заголовок второго уровня (для того, чтобы озаглавить подразделы).

Для того, чтобы перенести текст следующего блока на другую страницу, необходимо воспользоваться инструментов «Разрыв страницы» в разделе «Вставка».

