Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Московский государственный технический университет им. Н.Э. Баумана.»

(МГТУ им.Н.Э.Баумана)



Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_Программное обеспечение ЭВМ и информационные технологии\_\_\_\_\_\_

**Отчет по проектно-технологической практике**

Студент \_\_\_\_\_\_\_\_\_ Андрич Катарина\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Группа \_\_ИУ7-16Б\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_Обработка MIDI-программ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель

Преподаватель\_ \_Кузнецова О.В.\_ \_\_ \_\_\_\_\_\_\_\_\_

Должность ФИО Подпись

Москва, 2019

Содержание:

Условие задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

Схема программи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

Описание алгоритма решения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

Текст программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

Тесты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_15

Введение

В этой работе мы рассмотрим и решим проблему обработки MIDI. Мы собираемся использовать язык программирования Python. Работа содержит текст задачи, код, алгоритм решения и схему.

Условие задачи:

«Обработка MIDI-программ»

Задание:

MIDI (Musical Instrument Digital Interface — Цифровой интерфейс для музыкальных инструментов) - это стандарт для обмена данными между компьютером и музыкальным синтезатором. Часть этого стандарта определяет набор команд, которые заставляют синтезатор начать или закончить проигрывание определенной ноты. В следующем примере три ноты (аккорд из нот с номерами б0, 70 и 80) включаются одновременно и звучат 10 единиц времени (в дальнейшем, для простоты, будем называть эти единицы секундами), после чего сразу включается нота 62, которая звучит 2 секунды.

0 ON 60

0 ON 70

0 ОN 80

10 OFF 60

10 OFF 70

10 OFF 80

10 ON 62

12 OFF 62

Однако, большинство реальных мелодий не могут быть точно переведены в MIDI-программу. Иногда производится попытка включить ноту, которая уже играет. В этом случае, синтезатор не реагирует на эту команду: какой смысл включать ноту, которая уже включена? Например:

0 ON 60

10 ОN 60

12 OFF 60

20 OFF 60

Синтезатор, получив эту программу, проиграет ноту б0 12 секунд, а не 20, как задумано. Мы не услышим две отдельные ноты, а будем слышать их как одну.

В случае, когда включается нота, которая уже играет, программа может быть «исправлена» путем добавления команды OFF за одну секунду до второй команды ОN. Поскольку после команды ON останется две команды OFF, первая из них должна быть удалена. Такая «исправленная» программа будет вызывать эффект проигрывания одной и той же ноты с очень большой скоростью.

Еще одна проблема появляется при включении и выключении ноты в одно и то же время. В зависимости от порядка следования команд включения и выключения, нота либо будет выключена сразу же (как в примере слева), либо перерыв между нотами не будет слышен (справа).

0 ОN 60 0 ON 60

10 ОN 60 10 ОFF 60

10 0FF 60 10 ON 60

20 OFF 60 20 ОFF 60

В примере слева нота не будет сыграна. В примере справа перерыв между нотами недостаточно, чтобы дать слушателю услышать его. В обоих случаях алгоритм исправления одинаков: передвинуть команду ОFF на секунду назад.

Если после такого исправления команда ОFF окажется совпадающей с предыдущей командой ON, она должна быть удалена. Ваша задача - написать программу, которая могла бы читать произвольное количество программ и «исправлять» их.

Дано:

Каждая программа содержит произвольное количество строк. Каждая строка содержит, в соответствующем порядке, время, в которое команда посылается на синтезатор (неотрицательное целое), команду (ON или OFF) и ноту (целое в диапазоне 1-127). Части команды разделены пробелами. Каждая программа, кроме последней, заканчивается числом -1. Последняя программа заканчивается числом -2.

Результат:

Результат работы вашей программы - это исправленная MIDI-программа в том же формате, что и входные данные.

Уточнения:

1. Команды ON и ОFF всегда в верхнем регистре.
2. Время, указанное в команде, не убывает от команды к команде.
3. Изначально все ноты выключены.
4. Если две разные ноты должны быть выключены или включены, порядок следования соответствующих команд неважен.
5. Каждая команда ON имеет свою команду OFF, следующую за ней.

Пример:

Входные данные: Результат:

0 ON 60 0 ON 60

10 ON 60 9 OFF 60

12 OFF 60 10 ON 60

20 OFF 60 20 OFF 60

-1 -1

0 ON 60 0 ON 60

5 ON 70 5 ON 70

10 ON 60 9 OFF 60

10 OFF 60 10 ON 60

15 OFF 70 14 OFF 70

15 ON 70 15 ON 70

20 OFF 60 20 OFF 60

20 OFF 70 20 OFF 70

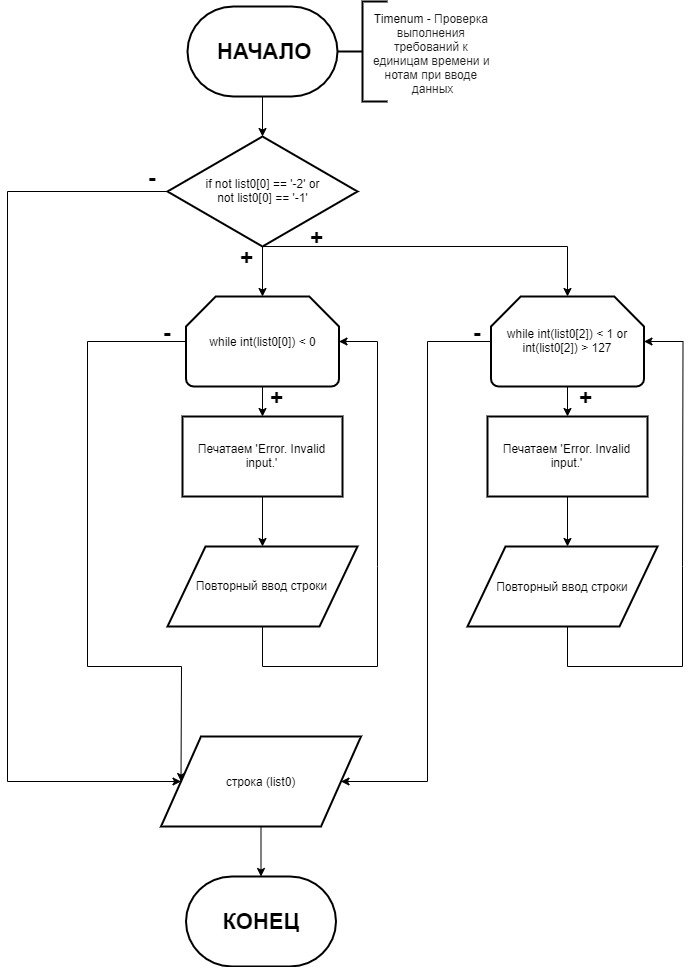
-1 -1

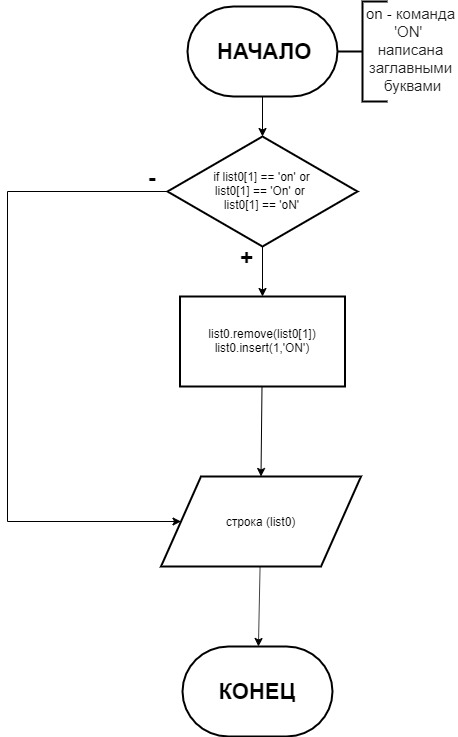
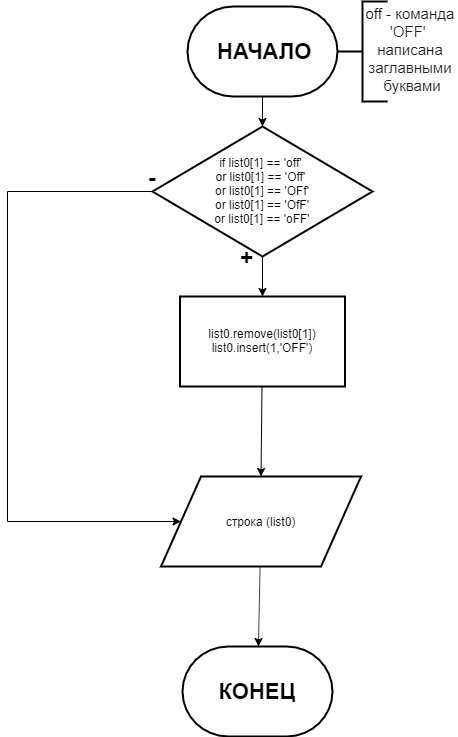
0 ON 60 0 ON 60

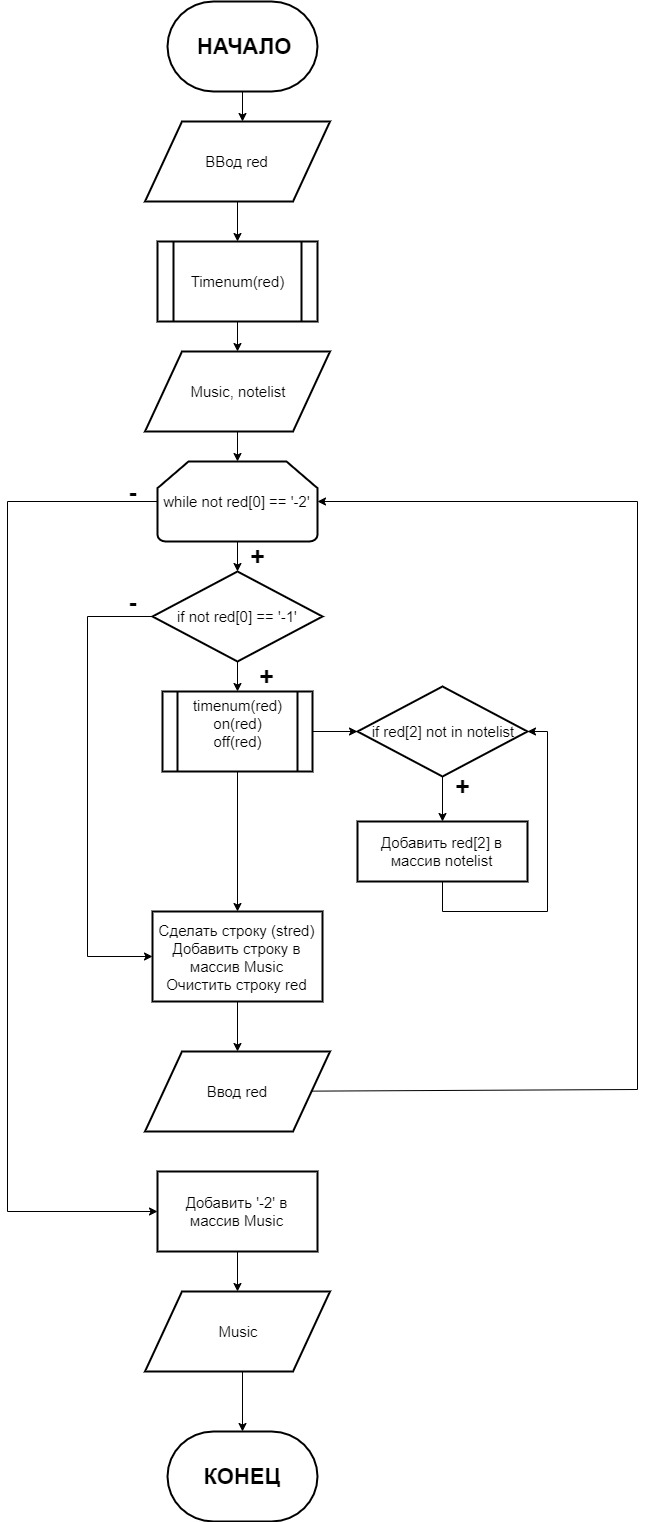
10 OFF 60 10 OFF 60

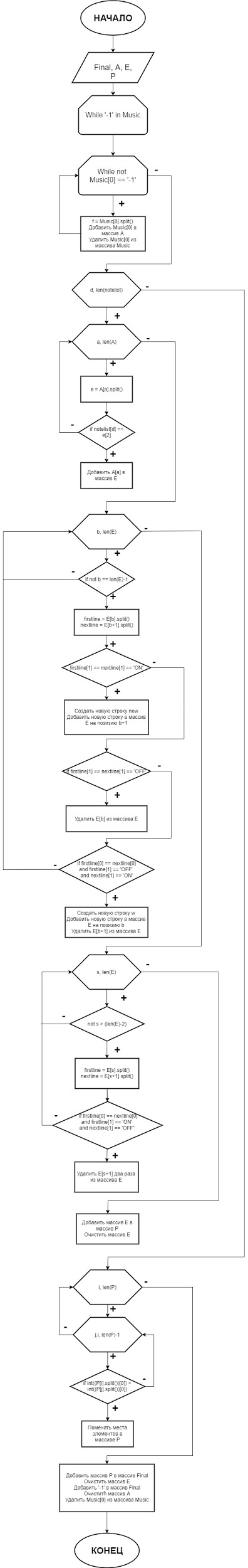
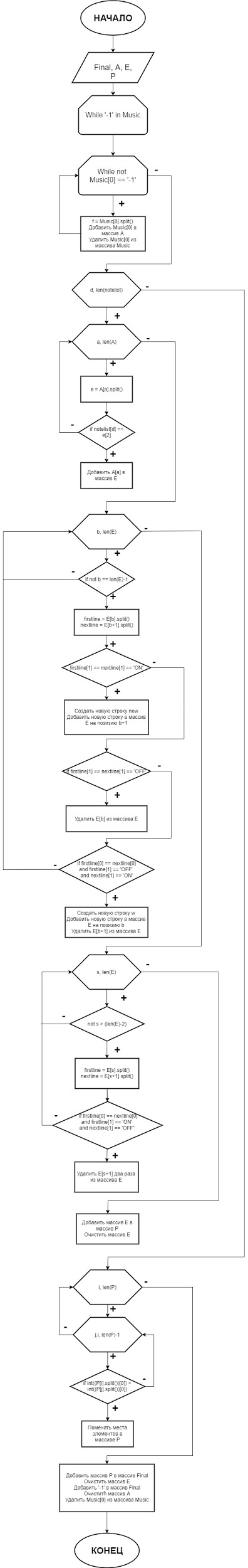
-2 -2

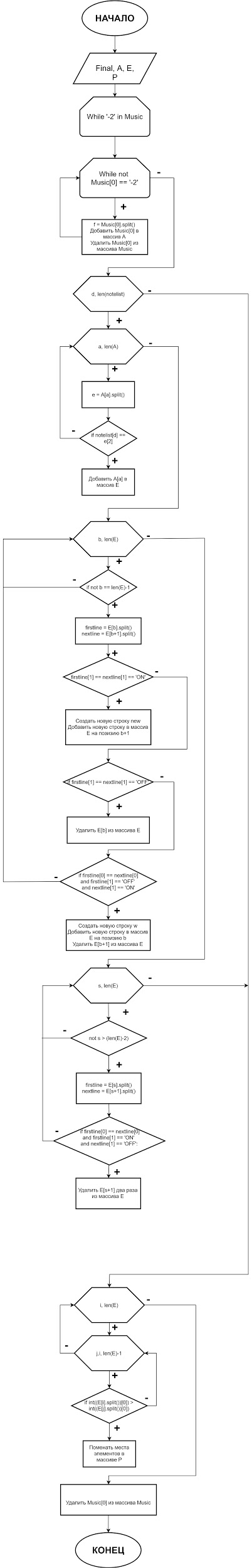
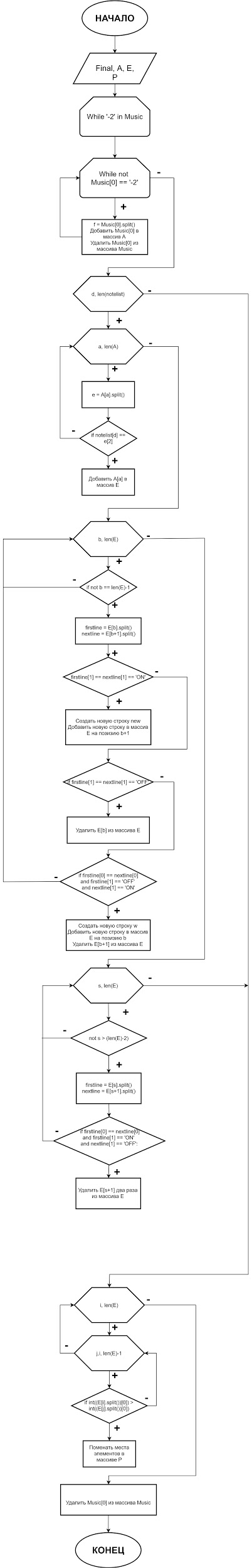
Схема программы:

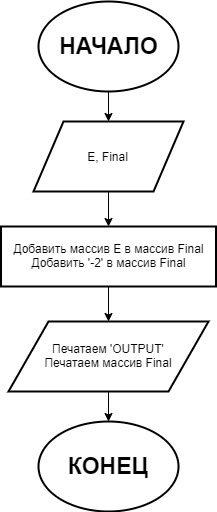






Описание алгоритма решения:

1. Вводим строки
2. Проверяем если первая цифра неотрицательное целое
3. Проверяем если вторая цифра (нота) целая в диапазоне 1-127
4. Если команды ON и OFF не заглавные, мы исправляем
5. Для каждой программы мы разделяем различные ноты, чтобы проверить их и исправить проблемы
   1. Если нота включена два раза подряд, мы отключаем первую на одну единицу времени перед началом второй
   2. Если музыкальная нота выключена два раза подряд, мы удаляем первую команду выключения
   3. Если музыкальная нота выключена, а затем включена в ту же единицу времени, мы выключаем ее на одну единицу времени раньше
6. Прежде чем печатать вывод, мы проверяем, чтобы единица времени находилась в возрастающем порядке

Текст программы:

#Катарина Андрич ИУ7 - 16Б

#Провераем ввод

def timenum(list0):

if not list0[0] == '-2' or not list0[0] == '-1':

while int(list0[0]) < 0:

print('Error. Invalid input.')

list0 = list(input('Try again: ').split())

while int(list0[2]) < 1 or int(list0[2]) > 127:

print('Error. Invalid input.')

list0 = list(input('Try again: ').split())

return list0

#Фиксируем слова ‘on’ и ‘off’

def on(list0):

if list0[1] == 'on' or list0[1] == 'On' or list0[1] == 'oN':

list0.remove(list0[1])

list0.insert(1,'ON')

return list0

def off(list0):

if list0[1] == 'off' or list0[1] == 'Off' or list0[1] == 'OFf' or list0[1] == 'OfF' or list0[1] == 'oFF':

list0.remove(list0[1])

list0.insert(1,'OFF')

return(list0)

#Вводим строку

red = list(input('Line: ').split())

#Провераем ввод

red = timenum(red)

Music = []

notelist = []

#Добавляем строку в массив

while not red[0] == '-2':

if not red[0] == '-1':

red = timenum(red)

on(red)

off(red)

if red[2] not in notelist:

notelist.append(red[2])

stred = ' '.join(map(str,red))

Music.append(stred)

red.clear()

red = list(input('Line: ').split())

Music.append('-2')

Final = []

A = []

E = []

P = []

#Проверяем каждую программу, кроме последней

while '-1' in Music:

while not Music[0] == '-1':

f = Music[0].split()

A.append(Music[0])

Music.remove(Music[0])

#Составляем список всех функций одной музыкальной ноты

for d in range(len(notelist)):

for a in range(len(A)):

e = A[a].split()

if notelist[d] == e[2]:

E.append(A[a])

#проверяем все условия, которые были предоставлены

for b in range(len(E)):

if not b == (len(E)-1):

firstline = E[b].split()

nextline = E[b+1].split()

if firstline[1] == nextline[1] == 'ON':

new = str(int(nextline[0])-1) + ' ' + 'OFF' + ' ' + notelist[d]

E.insert((b+1),new)

if firstline[1] == nextline[1] == 'OFF':

E.remove(E[b])

if firstline[0] == nextline[0] and firstline[1] == 'OFF' and nextline[1] == 'ON':

w = str(int(firstline[0])-1) + ' ' + 'OFF' + ' ' + notelist[d]

E.insert(b,w)

E.remove(E[b+1])

#Сортируем все ноты в порядке возрастания по времени

for s in range(len(E)):

if not s > (len(E)-2):

firstline = E[s].split()

nextline = E[s+1].split()

if firstline[0] == nextline[0] and firstline[1] == 'ON' and nextline[1] == 'OFF':

E.remove(E[s+1])

E.remove(E[s+1])

P.extend(E)

E.clear()

for i in range(len(P)):

for j in range(i,len(P)-1):

if int((P[i].split())[0])>int((P[j].split())[0]):

P[i],P[j] = P[j],P[i]

#Добавляем массив в главный массив

Final.extend(P)

P.clear()

Final.append('-1')

A.clear()

Music.remove(Music[0])

#Проверяем каждую программу, кроме последней

while '-2' in Music:

while not Music[0] == '-2':

A.append(Music[0])

Music.remove(Music[0])

for d in range(len(notelist)):

for a in range(len(A)):

e = A[a].split()

if notelist[d] == e[2]:

E.append(A[a])

#проверяем все условия, которые были предоставлены

for b in range(len(E)):

if not b == (len(E)-1):

firstline = E[b].split()

nextline = E[b+1].split()

if firstline[1] == nextline[1] == 'ON':

new = str(int(nextline[0])-1) + ' ' + 'OFF' + ' ' + notelist[d]

E.insert((b+1),new)

if firstline[1] == nextline[1] == 'OFF':

E.remove(E[b])

if firstline[0] == nextline[0] and firstline[1] == 'OFF' and nextline[1] == 'ON':

w = str(int(firstline[0])-1) + ' ' + 'OFF' + ' ' + notelist[d]

E.insert(b,w)

E.remove(E[b+1])

for s in range(len(E)):

if not s > (len(E)-2):

firstline = E[s].split()

nextline = E[s+1].split()

if firstline[0] == nextline[0] and firstline[1] == 'ON' and nextline[1] == 'OFF':

E.remove(E[s+1])

E.remove(E[s+1])

#Сортируем все ноты в порядке возрастания по времени

for i in range(len(E)):

for j in range(i,len(E)-1):

if int((E[i].split())[0])>int((E[j].split())[0]):

E[i],E[j] = E[j],E[i]

Music.remove(Music[0])

#Добавляем массив в главный массив

Final.extend(E)

Final.append('-2')

print()

#Печатаем решение

print('OUTPUT: ')

for v in range(len(Final)):

print(Final[v])

Тесты

Для тестирования моей программы я использовал следующие примеры:

Позитивный:

Входные данные: Результат:

0 ON 60 0 ON 60

10 ON 60 9 OFF 60

12 OFF 60 10 ON 60

20 OFF 60 20 OFF 60

-1 -1

0 ON 60 0 ON 60

5 ON 70 5 ON 70

10 ON 60 9 OFF 60

10 OFF 60 10 ON 60

15 OFF 70 14 OFF 70

15 ON 70 15 ON 70

20 OFF 60 20 OFF 60

20 OFF 70 20 OFF 70

-1 -1

0 ON 60 0 ON 60

10 OFF 60 10 OFF 60

-2 -2

Негативный:

Входные данные: Результат:

-6 on 10 Error. Invalid input:

Try again:

0 on 200 Error. Invalid input:

Try again:

Заключение:

Основная часть этой программы работает с массивами, а также с циклами while и for. Решая проблему обработки MIDI, мы обеспечиваем, чтобы компьютер мог обрабатывать и воспроизводить музыку, используя алгоритм так же, как люди могут делать это самостоятельно.

Список использованной литературы:

С.Н. Виноградов А.Ф. Кузьмин - “Логика. Учебник для средней школы”

Ален Б. Дауни - “Think Python: An Introduction to Software Design”

Ален Б. Дауни - “How to think like a computer scientist: Learning with Python”

https://python-scripts.com/loops-for-while