Министерство образования и науки Российской Федерации

Московский авиационный институт (национальный исследовательский университет)

ЖУРНАЛ

ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Наименование практики: *вычислительная*
Студент: В. Ю. Юревич
Факультет №8, курс 2, группа 7

Практика с 29.06.21 по 12.07.21

Москва, 2021

ИНСТРУКЦИЯ

о заполнении журнала по производственной практике

Журнал по производственной практике студентов имеет единую форму для всех видов практик.

Задание в журнал вписывается руководителем практики от института в первые три-пять дней пребывания студентов на практике в соответствии с тематикой, утверждённой на кафедре до начала практики. Журнал по производственной практике является основным документом для текущего и итогового контроля выполнения заданий, требований инструкции и программы практики.

Табель прохождения практики, задание, а также технический отчёт выполняются каждым студентом самостоятельно.

Журнал заполняется студентом непрерывно в процессе прохождения всей практики и регулярно представляется для просмотра руководителям практики. Все их замечания подлежат немедленному выполнению.

В разделе «Табель прохождения практики» ежедневно должно быть указано, на каких рабочих местах и в качестве кого работал студент. Эти записи проверяются и заверяются цеховыми руководителями практики, в том числе мастерами и бригадирами. График прохождения практики заполняется в соответствии с графиком распределения студентов по рабочим местам практики, утверждённым руководителем предприятия. В разделе «Рационализаторские предложения» должно быть приведено содержание поданных в цехе рационализаторских предложений со всеми необходимыми расчётами и эскизами. Рационализаторские предложения подаются индивидуально и коллективно.

Выполнение студентом задания по общественно-политической практике заносятся в раздел «Общественно-политическая практика». Выполнение работы по оказанию практической помощи предприятию (участие в выполнении спецзаданий, работа сверхурочно и т.п.) заносятся в раздел журнала «Работа в помощь предприятию» с последующим письменным подтверждением записанной работы соответствующими цеховыми руководителями. Раздел «Технический отчёт по практике» должен быть заполнен особо тщательно. Записи необходимо делать чернилами в сжатой, но вместе с тем чёткой и ясной форме и технически грамотно. Студент обязан ежедневно подробно излагать содержание работы, выполняемой за каждый день. Содержание этого раздела должно отвечать тем конкретным требованиям, которые предъявляются к техническому отчёту заданием и программой практики. Технический отчёт должен показать умение студента критически оценивать работу данного производственного участка и отразить, в какой степени студент способен применить теоретические знания для решения конкретных производственных задач.

Иллюстративный и другие материалы, использованные студентом в других разделах журнала, в техническом отчёте не должны повторяться, следует ограничиваться лишь ссылкой на него. Участие студентов в производственно-технической конференции, выступление с докладами, рационализаторские предложения и т.п. должны заноситься на свободные страницы журнала.

Примечание. Синьки, кальки и другие дополнения к журналу могут быть сделаны только с разрешения администрации предприятия и должны подшиваться в конце журнала.

Руководители практики от института обязаны следить за тем, чтобы каждый цеховой руководитель практики перед уходом студентов из данного цеха в другой цех вписывал в журнал студента отзывы об их работе в цехе.

Текущий контроль работы студентов осуществляется руководители практики от института и цеховыми руководителями практики заводов. Все замечания студентам руководители делают в письменном виде на страницах журнала, ставя при этом свою подпись и дату проверки.

Результаты защиты технического отчёта заносятся в протокол и одновременно заносятся в ведомость и зачётную книжку студента.

Примечание. Нумерация чистых страниц журнала проставляется каждым студентом в своём журнале до начала практики.

С инструкцией о заполнении журнала ознакомились:

« » _____ 2021 г.
(дата)

Студент Юревич В. Ю. _____
(подпись)

ЗАДАНИЕ

кафедры 806 по вычислительной практике: разработать программу для поиска додекафонных серий на основе интервального паттерна, освоить инструменты для создания MIDI файла и его воспроизведения.

Руководитель практики от института:

« » _____ 2021 г. Кухтичев А. А. _____

(дата) (подпись)

ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

Дата	**Содержание или наименование проделанной работы**	**Место работы**	**Время работы**		**Подпись цехового руководителя**
			Начало	Конец	
29.06.2021	Получение задания	МАИ	9:00	18:00	
30.06.2021	Разработка поиска	МАИ	9:00	18:00	
01.07.2021	Разработка поиска. Добавлена возможность использовать преобразования исходного паттерна	МАИ	9:00	18:00	
02.07.2021	Изучение возможностей языка Pure Data для создания MIDI	МАИ	9:00	18:00	
03.07.2021	Изучение возможностей языка MAX для создания MIDI	МАИ	9:00	18:00	
05.07.2021	Изучение теории додекафонных серий	МАИ	9:00	18:00	
06.07.2021	Изучение теории додекафонных серий	МАИ	9:00	18:00	
07.07.2021	Разработка поиска. Добавлен функционал построения серийной матрицы	МАИ	9:00	18:00	
08.07.2021	Поиск библиотек для создания MIDI и воспроизведения	МАИ	9:00	18:00	
09.07.2021	Знакомство с библиотекой Mingus. Разработка функционала генерации MIDI	МАИ	9:00	18:00	
10.07.2012	Поиск утилиты для построения графа серии	МАИ	9:00	18:00	
12.07.2021	Сдача журнала	МАИ	9:00	18:00	

Отзывы цеховых руководителей практики

Презентация защищена на комиссии кафедры 806. Работа выполнена в полном объёме. Рекомендую на оценку « ». Все материалы сданы на кафедру.

ПРОТОКОЛ

ЗАЩИТЫ ТЕХНИЧЕСКОГО ОТЧЁТА

по *производственной практике*

студентами: Юревичем Виталием Юрьевичем

Слушали:	**Постановили:**
Отчёт практиканта	считать практику выполненной и защищённой на Общая оценка: _____

Руководители: Зайцев В. Е. _____

Кухтичев А. А. _____

Дата: 12 июля 2021 г.

МАТЕРИАЛЫ ПО РАЦИОНАЛИЗАТОРСКИМ ПРЕДЛОЖЕНИЯМ

Можно добавить интерфейс для выбора вариантов преобразований паттерна, используемых при поиске. Сделать графический интерфейс. Интересно было бы добавить возможность генерации произведений на основе полученых серий с помощью алгоритмов искусственного интелекта. Очень интересно было бы реализовать поиск и воспроизведение серий в отличных от равномерно темперированного строях.

ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

Описание

Додекафония представляет собой технику музыкальной композиции, придуманную в начале XX века Арнольдом Шёнбергом. Основная идея додекафонии - создание произведений, в основе которых лежит повторение упорядоченной последовательности из двенадцати звуков. Такая последовательность называется додекафонной серией. Так как множество звуков равномерно темперированного строя без учёта октавы изоморфно аддитивной группе остатков от деления целых числел на 12, то мы можем представлять серию как упорядоченный набор двенадцати чисел от 0 до 11. При построении серии будем считать, что первым её элементом всегда является 0. Пользователь вводит интервальный паттерн - упорядоченный набор интервалов, которые последовательно откладываются от первого звука сери. В силу указанного изоморфизма паттерн будет представлять собой упорядоченный набор целых чисел, а построение интервалов - операцию сложения. Программа строит полученный паттерн, а затем, используя поиск в ширину, пытается найти интервалы между повторениями паттерна от различных звуков, чтобы в своей совокупности ипользовались все двенадцать звуков равномерно темперированного строя. Так же осуществляется поиск серий, построенных с помощью паттерна и трёх его преобразований: ракохода, инверсии и ретроверсии. Ракоход - это реверс последовательности чисел, образующих паттерн. Инверсия - это исходный паттерн, но со сменой знака на противоположный у каждого из чисел. Ретроверсия - это ракоход, но не исходного паттерна, а его инверсии. Если введённый пользоватем паттерн образует серии, то программа сохраняет все найденные серии в файл, после чего пользователь может воспроизвести любую из полученных серий, создать её MIDI файл, построить изображение серии на графе или построить серийную матрицу указанной серии. Серийная матрица представляет таблицу 12x12, в которой содержится исходная серия построенная от каждого из двенадцати звуков, а также её преобразования: ракоход, инверсия и ретроверсия, аналогичные преобразованиям паттерна.

Реализация

Файл main.cpp содержит реализацию поиска додекафонной серии и построения серийной матрицы.

```
#include <iostream>
#include <list>
#include <cmath>
#include <map>
#include <utility>
#include <vector>
#include <string>

class ToneRow {
private:
    std::list<unsigned int> rowList;
    bool eachElementOccursOnce;
    unsigned int maskOfRow;

    void CalcMaskOfRow() {
        maskOfRow = 0;
        eachElementOccursOnce = true;
        unsigned int maskInThePreviousStep = 0;
        unsigned int elemOfMask;
        for(unsigned const int& r : rowList) {
            elemOfMask = 1u << r;
            maskOfRow = maskOfRow | elemOfMask;
```

```
            if ((maskOfRow == maskInThePreviousStep) && (eachElementOccursOnce == true)) {
                eachElementOccursOnce = false;
            }
            maskInThePreviousStep = maskOfRow;
        }
    }

public:
    explicit ToneRow(unsigned int firstTone) {
        rowList.push_back(firstTone);
        CalcMaskOfRow();
    }

    ToneRow() : ToneRow((unsigned int)0) {

    }

    explicit ToneRow(std::list<unsigned int>& listOfTones) {
        for(unsigned int t : listOfTones) {
            if (t >= 12) {
                t %= 12;
            }
            rowList.push_back(t);
        }
        CalcMaskOfRow();
    }

    explicit ToneRow(std::string row[]) {
        std::map<std::string, unsigned int> isomorphism {
                {"C", 0},
                {"Db", 1},
                {"D", 2},
                { "Eb", 3},
                { "E", 4},
                {"F", 5},
                {"F#", 6},
                {"G", 7},
                {"Ab", 8},
                {"A", 9},
                {"Bb", 10},
                {"B", 11}
        };
        for(int i = 0; i < 12; ++i) {
            rowList.push_back(isomorphism[row[i]]);
        }
        CalcMaskOfRow();
    }

    bool CheckingTheUniquenessOfEachTone() {
        unsigned int maskForCheck = 0;
        unsigned int maskInThePreviousStep = 0;
        unsigned int elemOfMask;
        for(unsigned const int& r : rowList) {
            elemOfMask = 1u << r;
            maskForCheck = maskForCheck | elemOfMask;
            if (maskForCheck == maskInThePreviousStep) {
                return false;
            }
        }
        return true;
```

```
    }

    bool Unique() const {
        return eachElementOccursOnce;
    }

    size_t SizeOfRow() const {
        return rowList.size();
    }

    void AddToneInRow(unsigned const int& newTone) {
        rowList.push_back(newTone);
        unsigned int maskBeforeAdd = maskOfRow;
        unsigned int newToneMask;
        newToneMask = 1u << newTone;
        maskOfRow = maskOfRow | newToneMask;
        if (maskOfRow == maskBeforeAdd) {
            eachElementOccursOnce = false;
        }
    }

    void CalcNewToneByInterval(int interval) {
        int newTone = (int) rowList.back();
        if (abs(interval) >= 12) {
            interval %= 12;
        }
        if (interval < 0) {
            interval += 12;
        }
        newTone += interval;
        newTone %= 12;
        AddToneInRow((unsigned int) newTone);
    }

    ToneRow& operator=(const ToneRow& rhv) = default;

    std::list<unsigned int> GetRowList() const {
        return rowList;
    }

    friend std::ostream& operator<<(std::ostream& out, const ToneRow& rhv) {
        out << "[ ";
        for(unsigned const int& tone : rhv.rowList) {
            out << tone << " ";
        }
        out << "]";
        return out;
    }

    void CalcRowMatrix() {
        std::list<unsigned int> tmpRow = rowList;
        static std::map<unsigned int, std::string> isomorphism = {
                { 0, "C "},
                { 1, "Db "},
                { 2, "D "},
                { 3, "Eb "},
                { 4, "E "},
                { 5, "F "},
                { 6, "F# "},
                { 7, "G "},
```

```
            { 8, "Ab "},
            { 9, "A "},
            { 10, "Bb "},
            { 11, "B "}
        };
        unsigned int rowMtrx[12][12];
        for(size_t i = 0; i < 12; ++i) {
            rowMtrx[0][i] = tmpRow.front();
            tmpRow.pop_front();
        }
        for(size_t j = 1; j < 12; ++j) {
            rowMtrx[j][0] = 12 - rowMtrx[0][j];
        }
        for(size_t k = 1; k < 12; ++k) {
            for(size_t h = 1; h < 12; ++h) {
                rowMtrx[k][h] = (rowMtrx[0][h] + rowMtrx[k][0]) % 12;
            }
        }
        for(auto& k : rowMtrx) {
            for(unsigned int & h : k) {
                std::cout << "| " << isomorphism[h];
            }
            std::cout << "|" << std::endl;
        }
    }

};

class SearchTwelveToneRows {
public:
    std::list<int> pattern;
    std::pair<std::list<ToneRow>,std::list<std::list<int>>> resultOfSearchRows[4];
    std::map<size_t, std::vector<std::string>> textForPrint;

    explicit SearchTwelveToneRows(std::list<int> pttrn) : pattern(std::move(pttrn)) {

        std::list<int> inversionPattern;
        std::list<int> retrogradePattern;
        std::list<int> retrogradeInversionPattern;
        std::list<ToneRow> lstForSearch;
        CreatePatternInversion(inversionPattern);
        CreatePatternRetrograde(retrogradePattern);
        CreatePatternRetrogradeInversion(retrogradeInversionPattern);
        std::list<std::list<int>> arrayOfPatterns[4];
        for (size_t i = 0; i < 2; ++i) {
            arrayOfPatterns[0].push_back(pattern);
            arrayOfPatterns[0].push_back(pattern);

            arrayOfPatterns[1].push_back(pattern);
            arrayOfPatterns[1].push_back(inversionPattern);

            arrayOfPatterns[2].push_back(pattern);
            arrayOfPatterns[2].push_back(retrogradePattern);

            arrayOfPatterns[3].push_back(pattern);
            arrayOfPatterns[3].push_back(retrogradeInversionPattern);
        }
        for(size_t j = 0; j < 4; ++j) {

            resultOfSearchRows[j].first = std::list<ToneRow>();
```

```
203             resultOfSearchRows[j].second = arrayOfPatterns[j];
204         }
205         textForPrint[2].emplace_back(std::string("Pattern + Pattern + Pattern + Pattern = "));
206         textForPrint[2].emplace_back(std::string("Pattern + Inversion + Pattern + Inversion = "));
207         textForPrint[2].emplace_back(std::string("Pattern + Retrograde + Pattern + Retrograde = "));
208         textForPrint[2].emplace_back(std::string("Pattern + Retroversion + Pattern + Retroversion = "))
                ;
209
210         textForPrint[3].emplace_back(std::string("Pattern + Pattern + Pattern = "));
211         textForPrint[3].emplace_back(std::string("Pattern + Inversion + Pattern = "));
212         textForPrint[3].emplace_back(std::string("Pattern + Retrograde + Pattern = "));
213         textForPrint[3].emplace_back(std::string("Pattern + Retroversion + Pattern = "));
214
215         textForPrint[5].emplace_back(std::string("Pattern + Pattern = "));
216         textForPrint[5].emplace_back(std::string("Pattern + Inversion = "));
217         textForPrint[5].emplace_back(std::string("Pattern + Retrograde = "));
218         textForPrint[5].emplace_back(std::string("Pattern + Retroversion = "));
219
220     }
221
222     void CreatePatternInversion(std::list<int>& result) const {
223         result = pattern;
224         for(int& interval : result) {
225             interval *= -1;
226         }
227     }
228
229     void CreatePatternRetrograde(std::list<int>& result) const {
230         result = pattern;
231         result.reverse();
232     }
233
234     void CreatePatternRetrogradeInversion(std::list<int>& result) const {
235         result = pattern;
236         CreatePatternInversion(result);
237         result.reverse();
238     }
239
240     void FindStep(const ToneRow& originalRow, std::list<ToneRow>& result) {
241         ToneRow tmp;
242         for (int i = 1; i < 12; ++i) {
243             tmp = originalRow;
244             tmp.CalcNewToneByInterval(i);
245             if ((tmp.Unique()) && (tmp.SizeOfRow() <= 12)) {
246                 result.push_back(tmp);
247             }
248         }
249     }
250
251     bool CheckTwelveToneRow(const ToneRow& row) {
252         if ((row.Unique()) && (row.SizeOfRow() == 12)) {
253             return true;
254         }
255         else {
256             return false;
257         }
258     };
259
260     void FindTwelveToneRow(std::list<std::list<int>> patternsList, std::list<ToneRow>& result) {
261         std::list<ToneRow> listForFindStep;
```

```
262         std::list<ToneRow> listForSearch;
263         std::list<ToneRow> bufferForListForSearch;
264         ToneRow tmpRow;
265         std::list<int> currentPattern;
266         int interval;
267 
268         listForSearch.push_back(tmpRow);
269         currentPattern = patternsList.front();
270         patternsList.pop_front();
271         do {
272             if (!currentPattern.empty()) {
273                 interval = currentPattern.front();
274                 currentPattern.pop_front();
275                 while(!listForSearch.empty()) {
276                     tmpRow = listForSearch.front();
277                     listForSearch.pop_front();
278 
279                     tmpRow.CalcNewToneByInterval(interval);
280                     if (CheckTwelveToneRow(tmpRow)) {
281                         result.push_back(tmpRow);
282                     }
283                     else if ((tmpRow.Unique()) && (tmpRow.SizeOfRow() < 12)) {
284                         bufferForListForSearch.push_back(tmpRow);
285                     }
286                 }
287                 listForSearch = bufferForListForSearch;
288                 bufferForListForSearch.clear();
289             }
290             else if (currentPattern.empty()) {
291                 if (!patternsList.empty()) {
292                     currentPattern = patternsList.front();
293                     patternsList.pop_front();
294                 }
295                 while(!listForSearch.empty()) {
296                     FindStep(listForSearch.front(), listForFindStep);
297                     listForSearch.pop_front();
298                     while(!listForFindStep.empty()) {
299                         if (CheckTwelveToneRow(listForFindStep.front())) {
300                             result.push_back(listForFindStep.front());
301                             listForFindStep.pop_front();
302                         }
303                         else {
304                             bufferForListForSearch.push_back(listForFindStep.front());
305                             listForFindStep.pop_front();
306                         }
307                     }
308                 }
309                 listForSearch = bufferForListForSearch;
310                 bufferForListForSearch.clear();
311             }
312             else {
313                 std::cout << "End of pattern" << std::endl;
314                 break;
315             }
316         } while(! listForSearch.empty());
317     }
318 
319     std::string PrintRowByNoteNames(std::list<std::list<int>> lstOfPatterns, const ToneRow& row) {
320         static std::map<unsigned int, std::string> noteIsomorphism = {
321                 { 0, "C"},
```

```
            { 1, "Db"},
            { 2, "D"},
            { 3, "Eb"},
            { 4, "E"},
            { 5, "F"},
            { 6, "F#"},
            { 7, "G"},
            { 8, "Ab"},
            { 9, "A"},
            { 10, "Bb"},
            { 11, "B"}
        };
        std::string result;
        std::list<unsigned int> alreadyPrinted;
        std::list<unsigned int> rowForPrint = row.GetRowList();
        int newNote;
        unsigned int currentOctave;


        alreadyPrinted.push_back(rowForPrint.front());
        rowForPrint.pop_front();
        currentOctave = 4;
        result = "[" + noteIsomorphism[alreadyPrinted.back()] + "-" + std::to_string(currentOctave);

        while(alreadyPrinted.size() != 12) {
            if (lstOfPatterns.front().empty() != true) {
                newNote = (int)(alreadyPrinted.back() + lstOfPatterns.front().front());
                lstOfPatterns.front().pop_front();
                if (newNote >= 12) {
                    ++currentOctave;
                }
                else if (newNote <= 0) {
                    --currentOctave;
                }
                alreadyPrinted.push_back(rowForPrint.front());
                rowForPrint.pop_front();
                result += "," + noteIsomorphism[alreadyPrinted.back()] + "-" + std::to_string(
                    currentOctave);
            }
            else if (lstOfPatterns.front().empty() == true) {
                lstOfPatterns.pop_front();
                currentOctave = 4;
                alreadyPrinted.push_back(rowForPrint.front());
                rowForPrint.pop_front();
                result += "," + noteIsomorphism[alreadyPrinted.back()] + "-" + std::to_string(
                    currentOctave);
            }
        }
        result += "]";
        return result;
    }

    void StartSearch() {
        for(int i = 0; i < 4; ++i) {
            FindTwelveToneRow(resultOfSearchRows[i].second, resultOfSearchRows[i].first);
        }
    }

    void RightWriteCount(size_t& count) {
        if (count < 100) {
```

```
380             std::cout << "0";
381         }
382         if (count < 10) {
383             std::cout << "0";
384         }
385         std::cout << count << ") ";
386     }
387 
388     void PrintAllWhatIFind() {
389         size_t countRow = 0;
390         size_t typeOfMessage;
391         if (pattern.size() >= (size_t)5) {
392             typeOfMessage = 5;
393         }
394         else if ((pattern.size() == (size_t)4) || (pattern.size() == (size_t)3)) {
395             typeOfMessage = 3;
396         }
397         else {
398             typeOfMessage = 2;
399         }
400         for(int i = 0; i < 4; ++i) {
401             std::cout << textForPrint[typeOfMessage][i] << resultOfSearchRows[i].first.size() << std::
                endl;
402             for (auto& patt : resultOfSearchRows[i].second) {
403                 for(auto& p : patt) {
404                     if (p > 0) {
405                         std::cout << "+" << p << " ";
406                     }
407                     else {
408                         std::cout << p << " ";
409                     }
410                 }
411                 std::cout << " ";
412             }
413             std::cout << std::endl;
414             for(auto& r : resultOfSearchRows[i].first) {
415                 ++countRow;
416                 RightWriteCount(countRow);
417                 std::cout << PrintRowByNoteNames(resultOfSearchRows[i].second, r) << std::endl;
418             }
419             std::cout << "\n\n";
420         }
421     }
422 };
423 
424 int main(int argc, char* argv[]) {
425 
426     switch (std::stoi(std::string(argv[1]))) {
427         case 1: {
428             std::list<int> pattern;
429             for (size_t i = 2; i < argc; ++i) {
430                 pattern.push_back(std::stoi(std::string(argv[i])));
431             }
432             SearchTwelveToneRows obj(pattern);
433             obj.StartSearch();
434             obj.PrintAllWhatIFind();
435             break;
436         }
437         case 2: {
438             std::string rowStr[12];
```

```
            for (size_t i = 2; i < 14; ++i) {
                rowStr[i - 2] = std::string(argv[i]);
            }
            ToneRow row(rowStr);
            row.CalcRowMatrix();
            break;
        }
    }

    return 0;
}
```

Файл row.py содержит функционал для создания MIDI файла и воспроизведения найденной серии.

```
from mingus.containers import Note, Bar, Track
from mingus.midi import midi_file_out, fluidsynth
import sys
import argparse

def create_parser ():
    parser = argparse.ArgumentParser()
    parser.add_argument('-p', '--play', nargs='+', default=False)
    parser.add_argument('-m', '--midi', nargs='+', default=False)
    return parser

def create_midi(row):
    trk = Track()
    for note in row:
        trk.add_notes(note)
    midi_file_out.write_Track("row.mid", trk, 65)

def play_row(row):
    trk = Track()
    fluidsynth.init("samples.sf2", "alsa")
    for note in row:
        trk.add_notes(note)
    fluidsynth.play_Track(trk, 1, 65)

def main():
    parser = create_parser()
    namespace = parser.parse_args()


    if (namespace.play != False):
        play_row(namespace.play)
    elif (namespace.midi != False):
        create_midi(namespace.midi)

main()
```

Файл row.sh реализует функционал для взаимодействия вышеописанных модулей с пользователем.

```
#!/bin/bash
g++ -std=c++17 main.cpp -o search.out

if [[ $1 = "-c" ]]; then
  shift
  if [[ $# -eq 1 ]] ; then
    s=C
  else
    s=$2
  fi
```

```
  if [[ "$1" == *","* ]] ; then
    p=$1
    p=`echo $p | tr ',' ' ' ''`
    ./search.out 1 $p > row.t
    cat row.t
  else
    echo -e "Pattern doljen sojerjat' hotyabi 2 intervala"
    exit 1
  fi


elif [[ $1 = "-pr" ]] ; then
  if [[ -r row.t ]] ; then
    cat row.t
  else
    echo -e "Fail row.t ne syshestvyet ilil ne dostypen dlya chtenia"
    exit 1
  fi
elif [[ $1 = "-pl" ]] ; then
  if [[ $# -lt 2 ]] ; then
    echo -e "Dlya proigrivania serii neobhodimo ykazat' nomer serii v faile row.t"
    exit 1
  elif [[ !( -r row.t ) ]] ; then
    echo -e "Fail row.t ne syshestvyet ilil ne dostypen dlya chtenia"
    exit 1
  fi
  shift
  number="${1}"

  if [[ "$number" = "0"* ]]; then
    echo "Nomer serii sledyet ykazivat' bez posicionnih nulei"
    exit 1
  fi
  key1=1
  key2=1
  [[ $number -lt 100 ]] && key1=0
  [[ $number -lt 10 ]] && key2=0

  if [[ key1 -eq 0 ]] && [[ key2 -eq 1 ]] ; then
    number=`echo "0${number}"`
  elif [[ key1 -eq 0 ]] && [[ key2 -eq 0 ]] ; then
    number=`echo "00${number}"`
  fi

  if [[ ")" != "$number"* ]] ; then
    number=`echo "${number})"`
  fi

  row=`grep $number row.t`
  if [[ "$row" = "$number"* ]] ; then
    row=`echo ${row//$number}`
    row=`echo $row | tr ',' ' ' ''`
    row=`echo ${row//[}`
    row=`echo ${row//]}`
    row=`echo ${row//)}`
    r=$row
    r=`echo $r | tr -d [:digit:] | tr -d -`
    echo -e "\nVosproizvedenie serii: $r\n"
    python3 row.py -p $row
  else
```

```
    echo "Seria s ykazanim nomerom otsytstvyet v faile"
  fi


elif [[ $1 = "-md" ]] ; then
  if [[ $# -lt 2 ]] ; then
    echo -e "Dlya proigirivania serii neobhodimo ykazat' nomer serii v faile row.t"
    exit 1
  elif [[ !( -r row.t ) ]] ; then
    echo -e "Fail row.t ne syshestvyet ilil ne dostypen dlya chtenia"
    exit 1
  fi
  shift
  number="${1}"

  if [[ "$number" = "0"* ]]; then
    echo "Nomer serii sledyet ykazivat' bez posicionnih nulei"
    exit 1
  fi
  key1=1
  key2=1
  [[ $number -lt 100 ]] && key1=0
  [[ $number -lt 10 ]] && key2=0

  if [[ key1 -eq 0 ]] && [[ key2 -eq 1 ]] ; then
    number=`echo "0${number}"`
  elif [[ key1 -eq 0 ]] && [[ key2 -eq 0 ]] ; then
    number=`echo "00${number}"`
  fi

  if [[ ")" != "$number"* ]] ; then
    number=`echo "${number})"`
  fi

  row=`grep $number row.t`
  if [[ "$row" = "$number"* ]] ; then
    row=`echo ${row//$number}`
    row=`echo $row | tr ',' ' ' ''`
    row=`echo ${row//[}`
    row=`echo ${row//]}`
    row=`echo ${row//)}`
    r=$row
    r=`echo $r | tr -d [:digit:] | tr -d -`
    python3 row.py -m $row
    echo "MIDI fail s seriei sozdan"
  else
    echo "Seria s ykazanim nomerom otsytstvyet v faile"
  fi
elif [[ $1 = "-mtrx" ]]; then
  if [[ $# -lt 2 ]] ; then
    echo -e "Dlya vichislenia seriinoi matrici neobhodimo ykazat' nomer serii v faile row.t"
    exit 1
  elif [[ !( -r row.t ) ]] ; then
    echo -e "Fail row.t ne syshestvyet ilil ne dostypen dlya chtenia"
    exit 1
  fi
  shift
  number="${1}"

  if [[ "$number" = "0"* ]]; then
```

```
    echo "Nomer serii sledyet ykazivat' bez posicionnih nulei"
    exit 1
  fi
  key1=1
  key2=1
  [[ $number -lt 100 ]] && key1=0
  [[ $number -lt 10 ]] && key2=0

  if [[ key1 -eq 0 ]] && [[ key2 -eq 1 ]] ; then
    number=`echo "0${number}"`
  elif [[ key1 -eq 0 ]] && [[ key2 -eq 0 ]] ; then
    number=`echo "00${number}"`
  fi

  if [[ ")" != "$number"* ]] ; then
    number=`echo "${number})"`
  fi

  row=`grep $number row.t`
  if [[ "$row" = "$number"* ]] ; then
    row=`echo ${row//$number}`
    row=`echo $row | tr ',' ' '`
    row=`echo ${row//[}`
    row=`echo ${row//]}`
    row=`echo ${row//)}`
    row=`echo $row | tr -d [:digit:] | tr -d -`
    ./search.out 2 $row > row_mtrx.t
    cat row_mtrx.t
  else
    echo "Seria s ykazanim nomerom otsytstvyet v faile"
  fi

elif [[ $1 = "-g" ]] ; then
  if [[ $# -lt 2 ]] ; then
    echo -e "Dlya postroenia grafa neobhodimo ykazat' nomer serii v faile row.t"
    exit 1
  elif [[ !( -r row.t ) ]] ; then
    echo -e "Fail row.t ne syshestvyet ilil ne dostypen dlya chtenia"
    exit 1
  fi
  shift
  number="${1}"

  if [[ "$number" = "0"* ]]; then
    echo "Nomer serii sledyet ykazivat' bez posicionnih nulei"
    exit 1
  fi
  key1=1
  key2=1
  [[ $number -lt 100 ]] && key1=0
  [[ $number -lt 10 ]] && key2=0

  if [[ key1 -eq 0 ]] && [[ key2 -eq 1 ]] ; then
    number=`echo "0${number}"`
  elif [[ key1 -eq 0 ]] && [[ key2 -eq 0 ]] ; then
    number=`echo "00${number}"`
  fi

  if [[ ")" != "$number"* ]] ; then
    number=`echo "${number})"`
```

```
  fi

  row=`grep $number row.t`
  if [[ "$row" = "$number"* ]] ; then
    row=`echo $row | tr -d [:digit:] | tr -d -`
    row=`echo ${row//)}`
    row=`echo ${row/[/}`
    row=`echo ${row/]/;}`
    for (( i=1; i <= 12; ++i)) ; do
      row=`echo ${row/,/->}`
    done
    row=`echo ${row/F#/\"F#\"}`
    touch g.gv
    echo -e "digraph g1 {\n\tnode [shape = circle];\n\tedge [color="white"];" >> g.gv
    echo -e '\t"C" -> "Db" -> "D" -> "Eb" -> "E" -> "F";' >> g.gv
    echo -e '\t"F"-> "F#" -> "G" -> "Ab" -> "A" -> "Bb" -> "B" -> "C";\n}' >> g.gv
    circo g.gv > tmp.gv
    tr -d '}' < tmp.gv > tmp_g.gv
    echo -e "\tedge [color="black"];\n\t$row\n}" >> tmp_g.gv
    neato -n tmp_g.gv -Tpng -orow.png
    shotwell "row.png"
    rm g.gv tmp.gv tmp_g.gv
  else
    echo "Seria s ykazanim nomerom otsytstvyet v faile"
  fi

elif [[ $1 = "-rmv" ]] ; then
  rm search.out row.t row_mtrx.t row.mid row.png
elif [[ $# -eq 1 ]] ; then
  echo "Neobhodimo ykazat' odin kluch"
else
  echo "Nevernii kluch"

fi
```

Пример работы программы

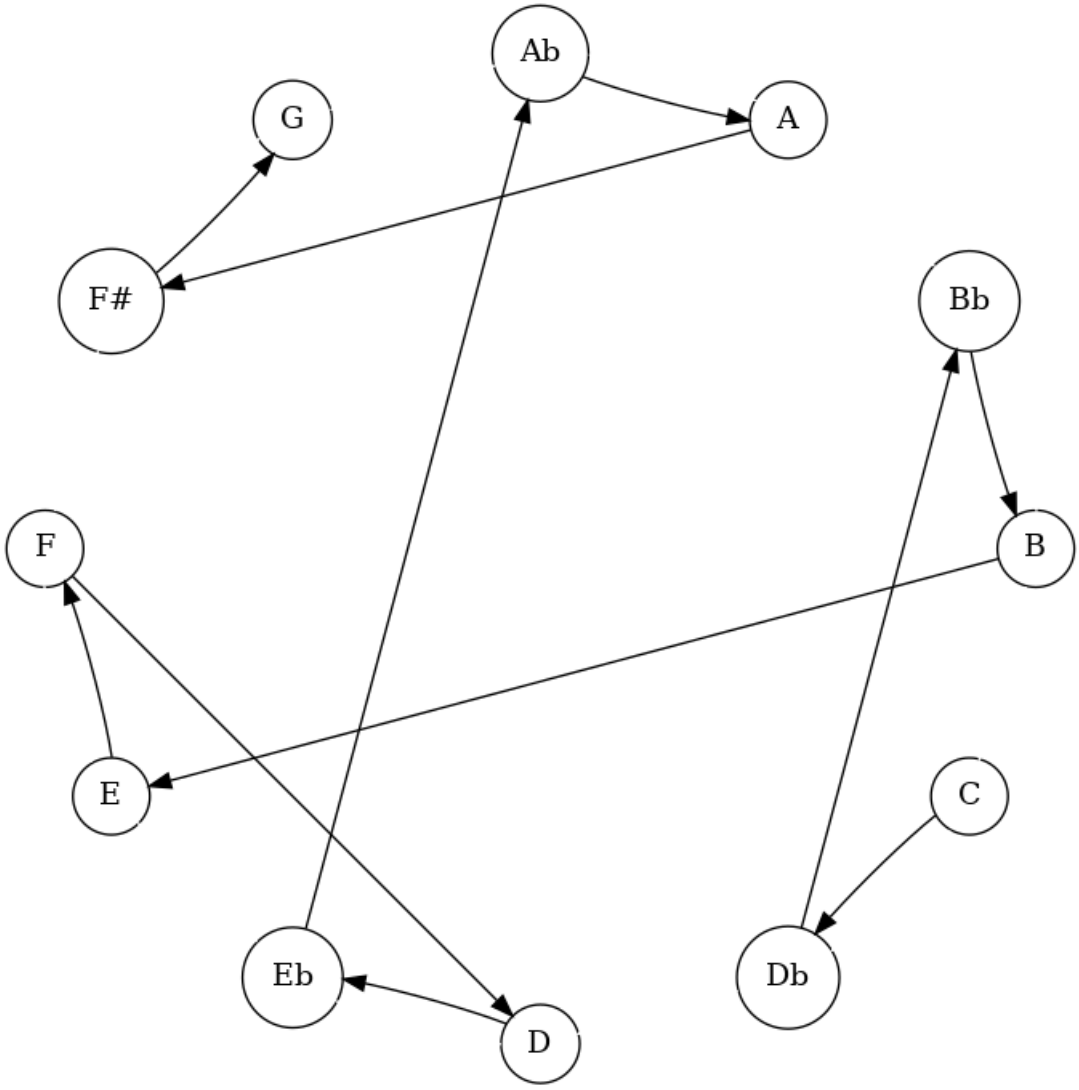
```
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -c 1,-3,1
Pattern + Pattern + Pattern = 2
+1 -3 +1     +1 -3 +1     +1 -3 +1     +1 -3 +1
001) [C-4,Db-4,Bb-3,B-3,E-4,F-4,D-4,Eb-4,Ab-4,A-4,F#-4,G-4]
002) [C-4,Db-4,Bb-3,B-3,Ab-4,A-4,F#-4,G-4,E-4,F-4,D-4,Eb-4]


Pattern + Inversion + Pattern = 2
+1 -3 +1     -1 +3 -1     +1 -3 +1     -1 +3 -1
003) [C-4,Db-4,Bb-3,B-3,Eb-4,D-4,F-4,E-4,Ab-4,A-4,F#-4,G-4]
004) [C-4,Db-4,Bb-3,B-3,G-4,F#-4,A-4,Ab-4,E-4,F-4,D-4,Eb-4]


Pattern + Retrograde + Pattern = 2
+1 -3 +1     +1 -3 +1     +1 -3 +1     +1 -3 +1
005) [C-4,Db-4,Bb-3,B-3,E-4,F-4,D-4,Eb-4,Ab-4,A-4,F#-4,G-4]
006) [C-4,Db-4,Bb-3,B-3,Ab-4,A-4,F#-4,G-4,E-4,F-4,D-4,Eb-4]


Pattern + Retroversion + Pattern = 2
+1 -3 +1     -1 +3 -1     +1 -3 +1     -1 +3 -1
007) [C-4,Db-4,Bb-3,B-3,Eb-4,D-4,F-4,E-4,Ab-4,A-4,F#-4,G-4]
008) [C-4,Db-4,Bb-3,B-3,G-4,F#-4,A-4,Ab-4,E-4,F-4,D-4,Eb-4]


yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -g 1
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $
```



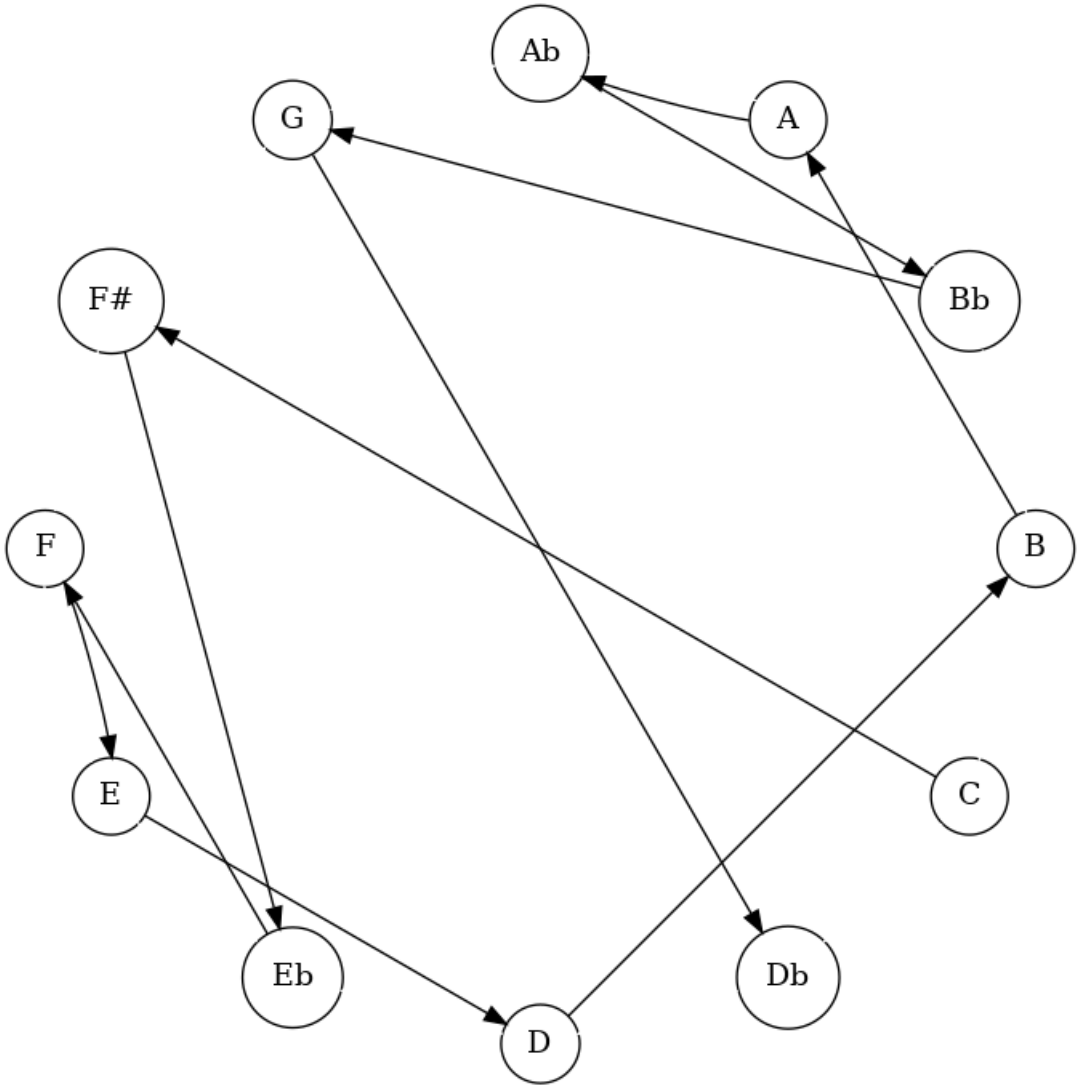
```
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -c 6,-3,2,-1,-2
Pattern + Pattern = 0
+6 -3 +2 -1 -2     +6 -3 +2 -1 -2     +6 -3 +2 -1 -2     +6 -3 +2 -1 -2


Pattern + Inversion = 1
+6 -3 +2 -1 -2     -6 +3 -2 +1 +2     +6 -3 +2 -1 -2     -6 +3 -2 +1 +2
001) [C-4,F#-4,Eb-4,F-4,E-4,D-4,Db-4,G-3,Bb-3,Ab-3,A-3,B-3]


Pattern + Retrograde = 1
+6 -3 +2 -1 -2     -2 -1 +2 -3 +6     +6 -3 +2 -1 -2     -2 -1 +2 -3 +6
002) [C-4,F#-4,Eb-4,F-4,E-4,D-4,B-4,A-4,Ab-4,Bb-4,G-4,Db-5]


Pattern + Retroversion = 0
+6 -3 +2 -1 -2     +2 +1 -2 +3 -6     +6 -3 +2 -1 -2     +2 +1 -2 +3 -6


yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -g 2
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $
```



```
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -c 6,7,-2
Pattern + Pattern + Pattern = 2
+6 +7 -2     +6 +7 -2     +6 +7 -2     +6 +7 -2
001) [C-4,F#-4,Db-5,B-4,E-4,Bb-4,F-5,Eb-5,Ab-4,D-5,A-5,G-5]
002) [C-4,F#-4,Db-5,B-4,Ab-4,D-5,A-5,G-5,E-4,Bb-4,F-5,Eb-5]


Pattern + Inversion + Pattern = 2
+6 +7 -2     -6 -7 +2     +6 +7 -2     -6 -7 +2
003) [C-4,F#-4,Db-5,B-4,E-4,Bb-3,Eb-3,F-3,Ab-4,D-5,A-5,G-5]
004) [C-4,F#-4,Db-5,B-4,Ab-4,D-4,G-3,A-3,E-4,Bb-4,F-5,Eb-5]


Pattern + Retrograde + Pattern = 2
+6 +7 -2     -2 +7 +6     +6 +7 -2     -2 +7 +6
005) [C-4,F#-4,Db-5,B-4,F-4,Eb-4,Bb-4,E-5,Ab-4,D-5,A-5,G-5]
006) [C-4,F#-4,Db-5,B-4,A-4,G-4,D-5,Ab-5,E-4,Bb-4,F-5,Eb-5]


Pattern + Retroversion + Pattern = 2
+6 +7 -2     +2 -7 -6     +6 +7 -2     +2 -7 -6
007) [C-4,F#-4,Db-5,B-4,Eb-4,F-4,Bb-3,E-3,Ab-4,D-5,A-5,G-5]
008) [C-4,F#-4,Db-5,B-4,G-4,A-4,D-4,Ab-3,E-4,Bb-4,F-5,Eb-5]


yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $ ./row.sh -mtrx 4
| C  | F# | Db | B  | Ab | D  | G  | A  | E  | Bb | F  | Eb |
| F# | C  | G  | F  | D  | Ab | Db | Eb | Bb | E  | B  | A  |
| B  | F  | C  | Bb | G  | Db | F# | Ab | Eb | A  | E  | D  |
| Db | G  | D  | C  | A  | Eb | Ab | Bb | F  | B  | F# | E  |
| E  | Bb | F  | Eb | C  | F# | B  | Db | Ab | D  | A  | G  |
| Bb | E  | B  | A  | F# | C  | F  | G  | D  | Ab | Eb | Db |
| F  | B  | F# | E  | Db | G  | C  | D  | A  | Eb | Bb | Ab |
| Eb | A  | E  | D  | B  | F  | Bb | C  | G  | Db | Ab | F# |
| Ab | D  | A  | G  | E  | Bb | Eb | F  | C  | F# | Db | B  |
| D  | Ab | Eb | Db | Bb | E  | A  | B  | F# | C  | G  | F  |
| G  | Db | Ab | F# | Eb | A  | D  | E  | B  | F  | C  | Bb |
| A  | Eb | Bb | Ab | F  | B  | E  | F# | Db | G  | D  | C  |
yurvch@yurvch-ASUS-EXPERTBOOK-P5440FA-P5440FA:~/MAI/SummerPractice2021 $
```

Ссылка на GitHub

https://github.com/vi-yurevich/SummerPractice2021