МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Комбинаторная теория полугрупп**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Алексеев Александр Александрович

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор, д.ф.-м.н. | \_\_\_\_\_\_\_\_\_\_ | В. А. Молчанов |
|  | подпись, дата |  |

Саратов 2022

1 Цель работы и порядок её выполнения­

Цель работы – изучение основных понятий теории полугрупп.

Порядок выполненных работы:

1. Рассмотреть понятие полугруппы, подполугруппы и порождающего множества. Разработать алгоритм построения подполугрупп по таблице Кэли.

2. Разработать алгоритм построения полугрупп бинарных отношений по заданному порождающему множеству.

3. Рассмотреть понятия подгруппы, порождающего множества и определяющих соотношений. Разработать алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям.

2 Теория

2.1 Понятия полугруппы, подполугруппы и порождающего множества

Определение. *Полугруппа* – это алгебра с одной ассоциативной бинарной операцией ·, т.е. выполняется для любых .

Если полугрупповая операция называется умножением, то полугруппу называют *мультипликативной*.

Лемма 1. Для любого непустого множества множество всех бинарных отношений на множестве с операцией композиции является полугруппой. Такая полугруппа называется *симметрической полугруппой бинарных отношений* на множестве и обозначается символом .

Следствие. Множество всех (частичных) преобразований непустого множества *X* с операцией композиции является полугруппой. Такая полугруппа называется *симметрической полугруппой* (*частичных*) *преобразований* множества X и обозначается символом *T*(*X*) (соответственно, *PT*(*X*)).

В случае конечного *n*-элементного множества операция умножения на задаётся *таблицей Кэли* размерности , строки и столбцы которой помечены элементами множества и в которой на пересечении –ой строки и –го столбца стоит произведение элементов , .

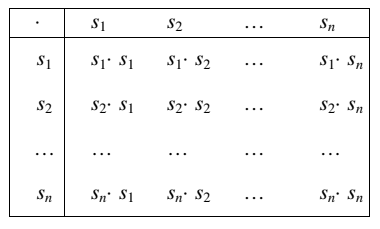


Рисунок 1 – Таблица Кэли операции умножения на множестве S

Определение (Классификация элементов полугруппы).

Элемент полугруппы называется:

1) *нулевым*, если для всех (при мультипликативной записи операции полугруппы такой элемент обозначается символом 0 и называется *нулем*);

2) *нейтральным (единичным)*, если для всех (при мультипликативной записи операции полугруппы такой элемент обозначается символом 1 и называется *единицей*);

3) *обратимым*, если для некоторого выполняется свойство: (такой элемент называется *обратным* для элемента и обозначается символом );

4) *идемпотентом*, если , т.е. .

Если в 1) выполняется лишь равенство для всех , то называется *правым нулем*. Аналогично определяется *левый* *нуль* и *односторонние единицы*.

Лемма 2. Для любой полугруппы *S* справедливы утверждения:

1) если *l* – левый нуль и *r* – правый нуль полугруппы *S*, то эти элементы равны и *S* имеет единственный нуль;

2) если *l* – левая единица и *r* – правая единица полугруппы *S*, то эти элементы равны и *S* имеет единственную единицу;

3) любой левый и правый нуль – идемпотент;

4) любая левая и правая единица – идемпотент.

Лемма 3 (Необходимое условие ассоциативности операции на конечном множестве). Любая конечная полугруппа имеет идемпотент.

Определение. Полугруппа *S* называет *коммутативной*, если умножение в этой полугруппе коммутативно, т.е. выполняется тождество *xy* = *yx* для любых *x*,*y* ∈ *S*.

Определение. Подмножество полугруппы называется *подполугруппой*, если устойчиво относительно операции умножения, т.е. для любых выполняется свойство: .

В силу общего свойства подалгебр пересечение любого семейства подполугрупп полугруппы является подполугруппой и, значит, множество всех подполугрупп полугруппы является системой замыканий. Следовательно, для любого подмножества полугруппы существует наименьшая подполугруппа , содержащая множество . Такая полугруппа обозначается символом и называется *подполугруппой* , *порождённой множеством* . При этом множество называется также *порождающим множеством подполугруппы* .

В частности, если , то называется *порождающим множеством полугруппы* и говорят, что множество порождает полугруппу .

Легко видеть, что полугруппа состоит из всевозможных конечных произведений элементов ,..., , т.е. выполняется равенство:

,...,.

2.2 Алгоритм вычисления подполугруппы <*X*> ⊂ *S*

*Вход*. Полугруппа S с таблицей Кэли и подмножество *X* ⊂ *S*.

*Выход*. Подполугруппа <*X*> ⊂ *S*.

Шаг 1. Положим *i* = 0, *X0* = *X*.

Шаг 2. Для *Xi* вычислим = {*x* ⋅ *y* : *x* *Xi* ˄ *y* *Xi*} и положим *Xi+1* = *Xi* ⋃ *.*

Шаг 3. Вычисляем <*X*> = .

Шаг 4. Выводим <*X*>.

2.3 Алгоритм построения полугруппы бинарных отношений по заданному порождающему множеству

*Вход*. Конечное порождающее множество *X* бинарных отношений (булевых матриц).

*Выход*. Полугруппа <*X*>.

Шаг 1. Положим *i* = 0, *X0* = *X*.

Шаг 2. Для *Xi* вычислим = {*x* ⋅ *y* : *x* *Xi* ˄ *y* *Xi*}, где *x* ⋅ *y* – композиция матриц *x* и *y*, и положим *Xi+1* = *Xi* ⋃ *.*

Шаг 3. Вычисляем <*X*> = .

Шаг 4. Выводим <*X*>.

2.4 Понятия подгруппы, порождающего множества и определяющих соотношений

Определение. Полугруппа с единичным элементом называется *моноидом*. Другими словами, моноид – это алгебра с ассоциативной бинарной операцией и выделенным единичным элементом 1. При этом полугруппа называется *полугруппой моноида* .

Для любой полугруппы канонически определяется моноид по следующему правилу: для некоторого элемента и умножение в на новый элемент 1 определяется по формуле: . Полугруппа моноида обозначается символом и называется *полугруппой с внешне присоединенной единицей*.

Определение. Моноид называется *группой*, если в нем все элементы обратимы.

Лемма. Для любого моноида множество всех обратимых элементов с операцией умножения моноида является группой, т.е. для любых выполняется условие .

Определение. *Подгруппой* полугруппы назовем подполугруппу из , являющуюся группой относительно бинарной операции, определенной в . Это эквивалентно тому, что есть подполугруппа из , в которой для любых существуют такие, что и . Отсюда легко получить, что *подмножество* *полугруппы* *является подгруппой тогда и только тогда, когда* *для любого* .

Единица подгруппа полугруппы является идемпотентом, но не обязательно единицей полугруппы .

Определение. Пусть – произвольное множество, называемое *алфавитом*. Элементы называются *буквами*. *Словом* над алфавитом конечная последовательность букв алфавита .

Обозначим символом множество всех непустых слов над алфавитом и символом - множество слов .

Теорема (О представлении полугрупп словами). Любая полугруппа является фактор-полугруппой некоторой полугруппы слов , т.е. для некоторой конгруэнции полугруппы .

Для любой конечной полугруппы найдется такой конечный алфавит , что для некоторого отображения выполняется равенство и, значит,. В этом случае множество называется *множеством порождающих символов полугруппы* (относительно отображения ).

Если при этом для слов выполняется равенство , т.е. , то говорят, что на *выполняется соотношение* (относительно отображения ).

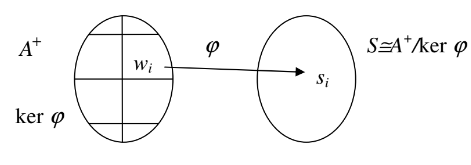


Рисунок 2 – Представление полугруппы словами

Очевидно, что в общем случае множество таких соотношений для всех пар будет бесконечным и не представляется возможности эффективно описать полугруппу в виде полугруппы классов конгруэнции . Однако в некоторых случаях можно выбрать такое сравнительно простое подмножество , которое однозначно определяет конгруэнцию как наименьшую конгруэнцию полугруппы , содержащую отношение , т.е. .

Так как в случае по-прежнему выполняется равенство , то будем писать и называть такие выражения *определяющими соотношениями*.

2.5 Алгоритм нахождения нового элемента для полугруппы

*Вход*. Строка *word*, множество уникальных слов *setOfR*, множество соотношений *R*, полугруппа *S*, размер максимального соотношения *maxRel*.

*Выход*. Полугруппа *S*.

Шаг 1. Если строка *word* есть в множестве *setOfR*, завершаем алгоритм.

Шаг 2. Инициализируем размер подстроки *substr* = 2.

Шаг 3.1. Пока *substr* не превышает *maxRel*, запускаем цикл по *i* от *word*.size() до 1. Для каждого *i* если *substr* > *word*.size() или *substr* > *i* + 1, завершаем цикл. Далее выделяем подстроку строки *word* от *word*[*i* – *substr*] до *word*[*i*] и сохраняем в *part*.

Шаг 3.2.1. Затем если в множестве соотношений *R* есть элемент *part*, то создаём пустую строку *new\_word* и добавляем туда подстроку от *word*[1] до *word*[*i* – *substr*], строку *R*[*part*] и подстроку от *word*[*i*] до *word*[*word*.*size*()].

Шаг 3.2.2. Если *cur\_word*.size() < *word*.size(), добавляем в *setOfR* строку *word* и *new\_word* и завершаем алгоритм. Иначе если *cur\_word* есть в *setOfR*, завершаем алгоритм. Иначе запускаем рекурсивно алгоритм от *cur\_word*.

Шаг 3.3. Если строка *word* есть в *setOfR*, завершаем алгоритм.

Шаг 3.4. После цикла по *i* увеличиваем *substr* на 1 и возвращаемся на шаг 3.1.

Шаг 4. Если *cur\_word*.size() = *word*.size() & *cur\_word* < *word* & строки *word* нет в *setOfR* & строки *cur\_word* нет в *setOfR*, в полугруппу *S* добавляем *cur\_word* и в *setOfR* добавляем *cur\_word* и *word*. Иначе в *S* добавляем *word*, в *setOfR* добавляем *word* и *cur\_word*.

2.6 Алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям

*Вход*. Алфавит *A*, множество определяющим соотношений *R*.

*Выход*. Полугруппа <*A*|*R*> и таблица Кэли.

Шаг 1. Инициализируем *count* = 1 и *S\_cur* = *A*.

Шаг 2.1. Пока *S\_cur* не пустая, увеличиваем *count* на 1 и запускаем цикл по *s* по всем элементам *S\_cur*. Для каждой *s* запускаем цикл по *letter* по всем элементам *A*. Для каждого *letter* запускаем алгоритм нахождения нового элемента для полугруппы от *s* + *letter*.

Шаг 2.2. После цикла по *s* очищаем *S\_cur* и запускаем снова цикл по *s* по всем элементам полугруппы *S*. Если *s*.size() = *count*, добавляем элемент в *S\_cur*. После цикла возвращаемся к шагу 2.1.

Шаг 3. Выводим полугруппу <*A*|*R*> и таблицу Кэли.

3 Результаты работы

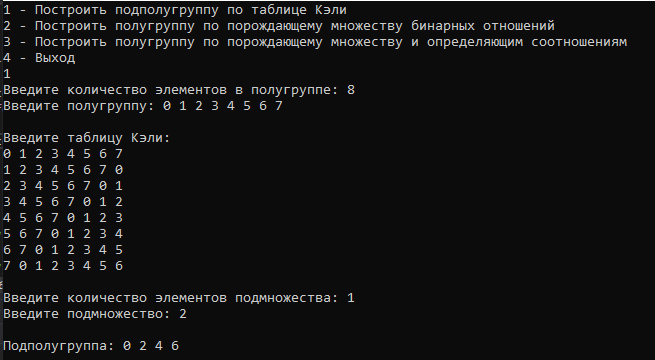
3.1 Оценка временной сложности алгоритмов

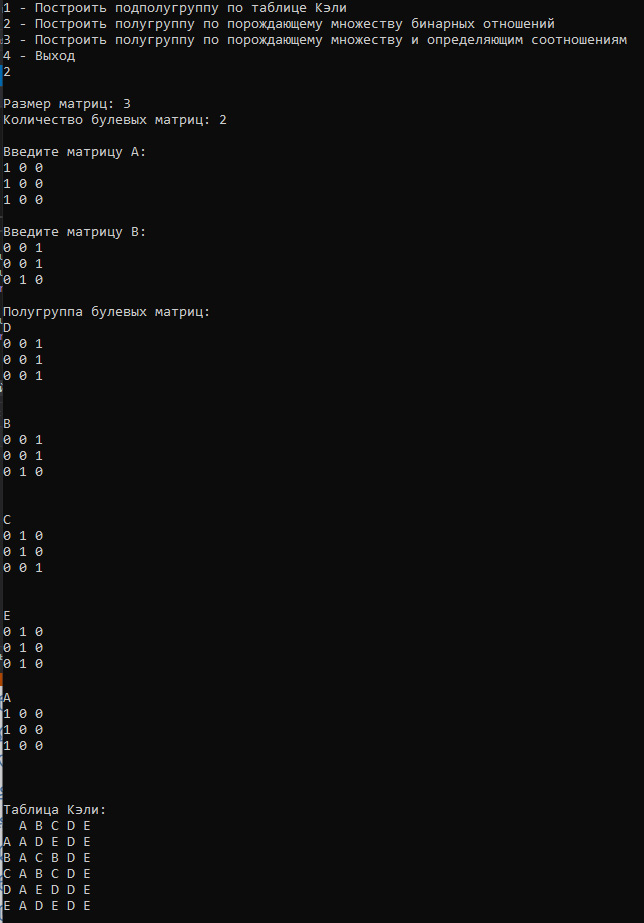
Алгоритм вычисления подполугруппы – O(N3);

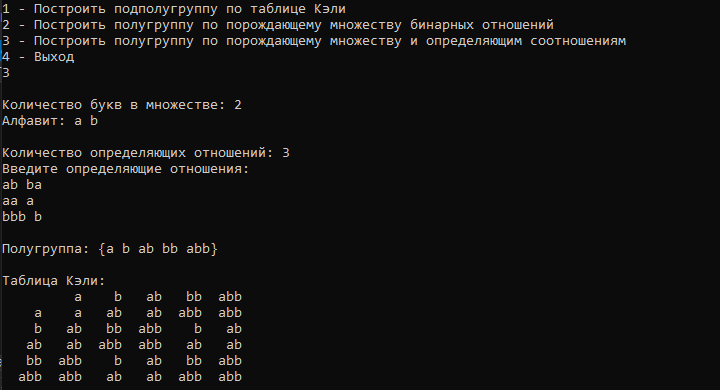
Алгоритм построения полугруппы бинарных отношений по заданному порождающему множеству – O(2N⋅N ⋅ N3);

Алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям – сложность алгоритма невозможно определить, т.к. в данном случае она зависит от того, сколько слов, не эквивалентных относительно конгруэнции ε из <A|R>, будет получаться на каждом шаге цикла.

3.2 Результаты тестирования программы







3.3 Код программы

#include "iostream"

#include "vector"

#include "math.h"

#include "algorithm"

#include "set"

#include "string"

#include "map"

using namespace std;

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Построение подполугруппы \*/

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int findNumber(char word, char\* el, int N) { //Поиск номера буквы в элементах полугруппы

for (int i = 0; i < N; i++)

if (el[i] == word)

return i;

}

bool associativity(char\*\* mas, int N, char\* el) { //Проверка свойства ассоциативности

for (int a = 0; a < N; a++)

for (int b = a; b < N; b++)

for (int c = 0; c < N; c++)

if (mas[findNumber(mas[a][b], el, N)][c] != mas[a][findNumber(mas[b][c], el, N)])

return false;

return true;

}

vector <char> obedSets(vector <char> A, vector <char> B) { //Объединяем 2 множества

vector <char> res;

res.insert(res.end(), A.begin(), A.end());

res.insert(res.end(), B.begin(), B.end());

sort(res.begin(), res.end());

for (int i = 1; i < res.size(); i++)

if (res[i] == res[i - 1]) {

res.erase(res.begin() + i);

i--;

}

return res;

}

void underhalfgroup() { //Находим подполугруппу

int N;

cout << "Введите количество элементов в полугруппе: ";

cin >> N;

char\* halfgroup = new char[N];

cout << "Введите полугруппу: ";

for (int i = 0; i < N; i++)

cin >> halfgroup[i];

char\*\* matrix = new char\* [N];

cout << "\nВведите таблицу Кэли: \n";

for (int i = 0; i < N; i++) {

matrix[i] = new char[N];

for (int j = 0; j < N; j++)

cin >> matrix[i][j];

}

if (!associativity(matrix, N, halfgroup)) {

cout << "\nНе выполняется ассоциативность полугруппы! \n";

return;

}

int M;

cout << "\nВведите количество элементов подмножества: ";

cin >> M;

vector <char> underset;

cout << "Введите подмножество: ";

for (int i = 0; i < M; i++) {

char x;

cin >> x;

underset.push\_back(x);

}

sort(underset.begin(), underset.end());

vector <char> underset\_;

vector <char> undersetNew;

while (true) {

underset\_.resize(0);

undersetNew.resize(0);

int size = underset.size();

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

underset\_.push\_back(matrix[findNumber(underset[i], halfgroup, N)][findNumber(underset[j], halfgroup, N)]);

undersetNew = obedSets(underset, underset\_);

if (underset == undersetNew)

break;

else

underset = undersetNew;

}

cout << "\nПодполугруппа: ";

for (int i = 0; i < underset.size(); i++)

cout << underset[i] << " ";

cout << endl;

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Построение полугруппы матриц \*/

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

vector <vector <int>> multMatrix(vector <vector <int>> A, vector <vector <int>> B, int N) { //Перемножаем булевы матрицы

vector <vector <int>> res(N);

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

int el = 0;

for (int a = 0; a < N; a++)

el += A[i][a] \* B[a][j];

if (el > 0) res[i].push\_back(1);

else res[i].push\_back(0);

}

}

return res;

}

void halfgroupMatrix() { //Полугруппа матриц

int N;

cout << "\nКоличество элементов на множестве: ";

cin >> N;

int M;

cout << "Количество булевых матриц: ";

cin >> M;

set <vector <vector <int>>> sets;

for (int i = 0; i < M; i++) {

vector <vector <int>> matrix(N);

cout << "\nВведите " << i + 1 << "-ю матрицу: \n";

for (int j = 0; j < N; j++)

for (int k = 0; k < N; k++) {

int x;

cin >> x;

matrix[j].push\_back(x);

}

sets.insert(matrix);

}

set <vector <vector <int>>> halfgroup;

halfgroup = sets;

while (true) {

for (auto i = sets.begin(); i != sets.end(); i++)

for (auto j = sets.begin(); j != sets.end(); j++)

halfgroup.insert(multMatrix(\*i, \*j, N));

if (halfgroup.size() == sets.size()) {

cout << "\nПолугруппа булевых матриц: \n";

vector <vector <int>> res;

for (vector <vector <int>> res : sets) {

for (int j = 0; j < N; j++) {

for (int k = 0; k < N; k++)

cout << res[j][k] << " ";

cout << endl;

}

cout << "\n\n";

}

return;

}

else

sets = halfgroup;

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Построение полугруппы по порождающему множеству и определяющим соотношениям \*/

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

set <string> setOfR;

map <string, string> R;

set <string> S\_cur, S, Scopy;

int maxRel = 0;

string cur\_word;

vector <vector <string>> cayley;

vector <string> split(string word) {

vector <string> splitedWord;

for (int i = 0; i < word.size(); i++)

splitedWord.push\_back(word.substr(i, 1));

return(splitedWord);

}

bool comp(string A, string B) {

return (A.size() < B.size()) || (A.size() == B.size() && A < B);

}

void newEl(string word, bool flagCayley = false, int row = 0, int size = -1) {

if (setOfR.find(word) != setOfR.end() && !flagCayley)

return;

vector <string> word\_ = split(word);

int substr = 2;

while (substr <= maxRel) {

for (int i = word\_.size() - 1; i >= 0; i--) {

if (substr > word.size() || substr > i + 1)

break;

string part = "";

for (int j = i; j > i - substr; j--)

part = word\_[j] + part;

if (R.find(part) != R.end()) {

cur\_word = "";

vector <string> part\_ = split(R[part]);

if (!flagCayley)

for (int k = 0; k < i - substr; k++)

cur\_word = cur\_word + word\_[k];

else

for (int k = 0; k < i - substr + 1; k++)

cur\_word = cur\_word + word\_[k];

cur\_word = cur\_word + R[part];

for (int k = i + 1; k < word\_.size(); k++)

cur\_word = cur\_word + word\_[k];

if (cur\_word.size() < word.size() && !flagCayley) {

setOfR.insert(word);

setOfR.insert(cur\_word);

return;

}

else if (setOfR.find(cur\_word) != setOfR.end() && !flagCayley)

return;

else if (flagCayley && Scopy.find(cur\_word) != Scopy.end()) {

if (cayley[row].size() < size)

cayley[row].push\_back(cur\_word);

return;

}

else {

int check = cayley[row].size();

newEl(cur\_word, flagCayley, row, size);

if (flagCayley && cayley[row].size() > check)

return;

}

}

if (setOfR.find(word) != setOfR.end() && !flagCayley)

return;

if (flagCayley && Scopy.find(word) != Scopy.end()) {

if (cayley[row].size() <= size)

cayley[row].push\_back(word);

return;

}

}

substr++;

}

if (cur\_word.size() == word.size() && cur\_word < word && setOfR.find(word) == setOfR.end() && setOfR.find(cur\_word) == setOfR.end()) {

S.insert(cur\_word);

setOfR.insert(word);

setOfR.insert(cur\_word);

}

else {

S.insert(word);

setOfR.insert(word);

setOfR.insert(cur\_word);

}

}

void halfgroupRel() {

int N;

cout << "\nКоличество букв в множестве: ";

cin >> N;

set <string> A;

cout << "Алфавит: ";

for (int i = 0; i < N; i++) {

string a;

cin >> a;

A.insert(a);

S.insert(a);

setOfR.insert(a);

}

int M;

cout << "\nКоличество определяющих отношений: ";

cin >> M;

cout << "Введите определяющие отношения: \n";

for (int i = 0; i < M; i++) {

string r1, r2;

cin >> r1 >> r2;

if (r1.size() > maxRel)

maxRel = r1.size();

if (r2.size() > maxRel)

maxRel = r2.size();

if (r1.size() < r2.size())

R[r2] = r1;

else if (r1.size() > r2.size())

R[r1] = r2;

else {

if (r1 < r2)

R[r2] = r1;

else

R[r1] = r2;

}

}

int count = 1;

S\_cur = A;

while (!S\_cur.empty()) {

count++;

for (string s : S\_cur)

for (string letter : A)

newEl(s + letter);

S\_cur.clear();

for (auto s : S)

if (s.size() == count)

S\_cur.insert(s);

}

vector <string> halfgroup;

for (string s : S)

halfgroup.push\_back(s);

sort(halfgroup.begin(), halfgroup.end(), comp);

int hSize = halfgroup.size();

cout << "\nПолугруппа: {";

for (int i = 0; i < hSize; i++) {

if (i == hSize - 1)

cout << halfgroup[i] << "} \n";

else

cout << halfgroup[i] << " ";

}

cayley.resize(hSize);

Scopy = S;

for (int i = 0; i < hSize; i++)

for (int j = 0; j < hSize; j++)

newEl(halfgroup[i] + halfgroup[j], true, i, hSize);

cout << endl << "Таблица Кэли: \n";

cout << " ";

for (int i = 0; i < hSize; i++)

cout << setw(5) << halfgroup[i];

cout << endl;

for (int i = 0; i < hSize; i++) {

cout << setw(5) << halfgroup[i] << setw(5);

for (int j = 0; j < hSize; j++)

cout << cayley[i][j] << setw(5);

cout << endl;

}

cout << endl;

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Главная функция \*/

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int main() {

setlocale(LC\_ALL, "ru");

for (;;) {

cout << "1 - Построить подполугруппу по таблице Кэли \n2 - Построить полугруппу по порождающему множеству бинарных отношений \n";

cout << "3 - Построить полугруппу по порождающему множеству и определяющим соотношениям \n4 - Выход \n";

int x;

cin >> x;

switch (x) {

case 1:

underhalfgroup();

cout << endl;

break;

case 2:

halfgroupMatrix();

cout << endl;

break;

case 3:

halfgroupRel();

cout << endl;

break;

case 4:

return 0;

default:

cout << "Incorrect. Try again! \n";

}

}

}

ЗАКЛЮЧЕНИЕ

В ходе данной работы были изучены и реализованы алгоритмы построения подполугрупп по таблице Кэли, построения полугрупп бинарных отношений по заданному порождающему множеству и построения полугрупп по порождающему множеству и определяющим соотношениям.