Московский авиационный институт

(национальный исследовательский институт)

Институт «Компьютерные науки и прикладная математика»

**Курсовой проект по курсу**

**«Системы программирования»**

**IV семестр**

**Тема: Реализация оператора «звезда Клини» (\*).**

*Студент:* Михеева К. О.*Группа: М8О-207Б-20*

*Руководитель:* Семёнов А. С.

*Оценка:*

*Дата:*

**Москва. 2022**

**Задача:**

1. Спроектировать теоретико-множественный язык (SETL), описать основные операции над множеством.
2. При помощи функционала языка C# реализовать оператор звезда Клини .
3. Провести тестирование программы.
4. Сформулировать выводы.

**Теоретическая справка**

Язык Сетл был предложен и разработан в начале 70-х годов профессором Нью-Йоркского университета Джекобом Шварцем (Jacob Schwartz). Это стало выдающимся событием: математик с мировым именем смело вошел в область программирования с радикальным практическим предложением. Оно состояло в том, чтобы, базируясь на множествах как основных типах данных, писать программы в терминах высокоуровневых непроцедурных алгоритмов, широко использующих логико-предикатные средства.

Вот два характерных примера записи на Сетле исполняемых алгоритмов.

Напечатать простые числа, меньшие N

Print { n < N | not exist k , 1< k < n | div ( n , k )}

Выборка из базы данных:

Print { rec in DB | rec . age < 30 & rec . salary > good \_ salary }

В 1972 году Дж. Шварц по пути из Китая сделал короткую остановку в Новосибирске и рассказал о Сетле на семинаре в Вычислительном Центре. Организация этого выступления и последующего проекта Сетл в Новосибирске стали еще одним подтверждением выдающейся чуткости А. П. Ершова к фундаментальным тенденциям в программировании.

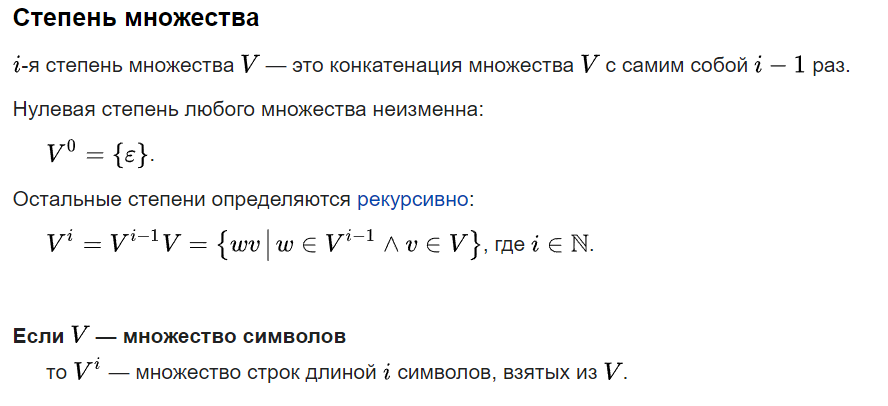
При выполнении курсового проекта за основу взят язык C#, поэтому синтаксис выглядит несколько иначе.

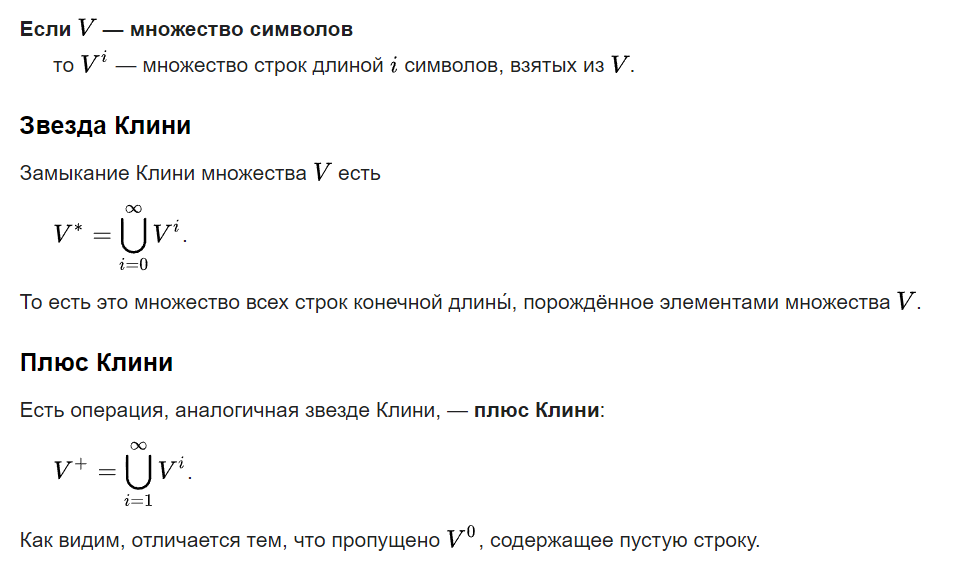
**Описание оператора звезда Клини:**

**Звезда́ Кли́ни** (или **замыка́ние Кли́ни**) в [математической логике](https://ru.wikipedia.org/wiki/Математическая_логика) и [информатике](https://ru.wikipedia.org/wiki/Информатика) — [унарная операция](https://ru.wikipedia.org/wiki/Унарная_операция) над [множеством](https://ru.wikipedia.org/wiki/Множество) [строк](https://ru.wikipedia.org/wiki/Строковый_тип) либо [символов](https://ru.wikipedia.org/wiki/Символьный_тип). Замыкание Клини множества *V* обозначается *V*\*. Широко применяется в [регулярных выражениях](https://ru.wikipedia.org/wiki/Регулярные_выражения).

Если *V*— множество строк, то *V*\*— минимальное [надмножество](https://ru.wikipedia.org/wiki/Надмножество) множества *V*, которое содержит ε ([пустую строку](https://ru.wikipedia.org/wiki/Пустая_строка)) и [замкнуто](https://ru.wikipedia.org/wiki/Замыкание_(алгебра)) относительно [конкатенации](https://ru.wikipedia.org/wiki/Конкатенация). Это также множество всех строк, полученных конкатенацией нуля или более строк из *V*.

Если *V*— множество символов, то *V*\*— множество всех строк из символов из *V* с добавлением пустой строки.





**Описание алгоритма:**

Зададим максимальную длину цепочек равную четырем. Создадим переменную результата, в которой мы будем хранить наши цепочки. Далее мы стеке храним наши терминальные символы, из которых мы в дальнейшем составим цепочки вывода. Первым мы берем верхний элемент стека и сравниваем его с нашей максимальной длинной, если символ меньше максимальной длины, то мы заносим в его в переменную результата. Это продолжается до тех пор, пока мы не составим всем возможные комбинации допустимой длины.

**Пример:**

Вход: T = {a,d,c}

Выход: T\* = {eps, a, d, c, ad, ac, dc, da, ca, cd, aa, bb, cc, adc, acd, cda, cad, dac, dca…}

Изображение выглядит как текст

Автоматически созданное описание

V0 = {e}

V1 = {a,d,c}

V2 = {aa, ad ,ac ,dd...}

top = a

aa

ad

…

V0 = {eps}

foreach i in 1..maxLength

foreach w in Vi-1

foreach v in T

Vi = wv

end

end

end

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KP.SETL

{

public class OpKlini: OpNode

{

private const int \_maxLength = 4;

private OpVariable \_variable;

public OpKlini(OpVariable variable)

{

\_variable = variable;

}

public override DataNode GetResult(Dictionary<string, Variable> variables)

{

return GetKliniChains(variables[\_variable.Name].Data as DataSymbols);

}

private DataChains GetKliniChains(DataSymbols symbols)

{

var result = new DataChains("");

//result.Chains.Add(new DataChain(""));

var strs = new Stack<string>();

foreach(var c in symbols.Symbols)

{

strs.Push(c.ToString());

}

while(strs.Count > 0)

{

var top = strs.Pop();

if(top.Length > \_maxLength)

{

continue;

}

result.Chains.Add(new DataChain(top));

foreach (var c in symbols.Symbols)

{

strs.Push(top + c.ToString());

}

}

return result;

}

}

}

**Результат работы программы:**

Изображение выглядит как текст

Автоматически созданное описание**Вывод:**

При выполнении работы мной был спроектирован алгоритмом звезда Клини. Дальше данный оператор был реализован языке С#.