**Черкаський національний університет імені Богдана Хмельницького**

**Кафедра інформаційних технологій**

(повна назва кафедри)

(повне найменування вищого навчального закладу)

**КУРСОВА РОБОТА**

**з дисципліни «Програмування та алгоритмічні мови»**

**НА ТЕМУ «Емулятор машини Тюрінга для розв’язання простих завдань»**

Студента \_\_2\_\_ курсу, групи \_КН-20\_\_\_\_\_\_\_\_\_

Напряму підготовки «Інформаційні технології»

Спеціальності «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_

Костюк Б.Б.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник Авраменко Валентин Семенович\_\_\_\_

доцент, кандидат фізико-математичних наук\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_

* + 1. Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

(прізвище та ініціали)

* + 1. **\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис)

(прізвище та ініціали)

* + 1. **\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис)

(прізвище та ініціали)

* + 1. **м. Черкаси-2021 рік**

Завідувач кафедри \_\_\_ІТ\_\_\_,

Веретельник В. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

„\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2021 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ на курсову роботу**

Студента Костюк Богдана Борисовича групи КН-20 другого курсу

**ТЕМА** **Емулятор машини Тюрінга для розв’язання простих завдань**

**Вихідні дані до курсової роботи**: Розроблений програмний продукт повинен:\_\_\_\_\_\_\_\_\_\_\_

1) мати зручний графічний інтерфейс;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) правильно обчислювати заданий алгоритм;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) надавати можливість зберігати алгоритм машини Тюрінга;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зміст **Пояснювальної записки** до курсової роботи:

Індивідуальне завдання

Вступ

1. Огляд …
2. Розробка схеми алгоритму …
3. Розробка програми …

Висновки

Список літератури

Додатки (за необхідності)

Перелік наочного матеріалу:

Програмний продукт …, пояснювальна записка, презентація результатів роботи. . **Календарний план виконання роботи:**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапу дипломного проекту (роботи) | Термін виконання етапу: до … | Примітка |
| 1. | Отримання завдання на курсову роботу. | 18.09. 2021 |  |
| 2. | Огляд джерел технічної інформації за темою роботи. | 30.09. 2021 |  |
| 3. | Виконання аналізу методів реалізації завдання | 16.10. 2021 |  |
| 3. | Розробка алгоритму реалізації завдання | 30.10. 2021 |  |
| 4. | Програмування алгоритму, створення програмного продукту | 20.11. 2021 |  |
| 5. | Тестування розробленого програмного продукту | 27.11. 2021 |  |
| 6. | Написання пояснювальної записки | 13.12. 2021 |  |
| 7. | Подача роботи керівнику для написання відгуку. | 19.12. 2021 |  |
| 8. | Корегування роботи за результатами розгляду керівника. Остаточне оформлення пояснювальної роботи. | 26.12. 2021 |  |
| 9. | Написання доповіді, створення слайдів. | 27.12. 2021 |  |
| 10. | Захист курсової роботи | 28.12. 2021 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ / Костюк Б.Б./ Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_ / Авраменко В.С./

(підпис) (Прізвище, ініціали) (підпис) (Прізвище, ініціали)

Завдання видане «18» вересня 2021 року

* + 1. **Зміст**

[Вступ. 4](#_Toc91114622)

[Розділ 1 Порівняння та аналіз аналогів Машини Тюрінга. 6](#_Toc91114623)

[1.1. Нормальні алгоритми Маркова(НАМ). 6](#_Toc91114624)

[1.2. Машина Поста 7](#_Toc91114625)

[1.3. Висновок до першого розділу 9](#_Toc91114626)

[Розділ 2 Проектування модуля для роботи емулятора машини Тюрінга. 11](#_Toc91114627)

[2.1. Обрання способу запису Машини Тюрінга. 11](#_Toc91114628)

[2.2. Реалізація покрокового виконання програми. 12](#_Toc91114629)

[2.3. Блок-схема початку виконання Машини Тюрінга. 12](#_Toc91114630)

[2.4. Блок-схема основного методу з основною роботою. 13](#_Toc91114631)

[2.5. Блок-схема методу який аналізує і виконує операцію. 14](#_Toc91114632)

[2.6. Висновок до другого розділу. 15](#_Toc91114633)

[Розділ 3 Реалізація та тестування емулятора машини Тюрінга. 16](#_Toc91114634)

[3.1. Реалізація допоміжного класу TuringCondition. 16](#_Toc91114635)

[3.2. Реалізація основного класу Turing. 17](#_Toc91114636)

[3.3. Опис реалізації інтерфейсу користувача. 21](#_Toc91114637)

[3.4. Приклад роботи програми. 22](#_Toc91114638)

[3.5. Висновок до третього розділу. 24](#_Toc91114639)

[Висновки 25](#_Toc91114640)

[Розділ 4 Список використаних джерел. 26](#_Toc91114641)

Вступ

Тема роботи є актуальною так як емулятор машини Тюрінга може знадобитися в навчальних цілях щоб навчитися розуміти складніші машини. Машину Тюрінга можна використовувати для шифрування або дешифрування, тобто для вивчення криптографії її теж можна використовувати. Також машину Тюрінга можна використовувати для роботи з строками.

Мета роботи – це написати емулятор машини Тюрінга для розв’язання простих завдань. Емулятор мусить бути в вигляді програми з графічним оформленням.

Постановка задачі курсової роботи:

1. Створення алгоритму для роботи машини Тюрінга.
2. Побудувати блок-схему алгоритму.
3. Реалізувати у вигляді класу модуль для роботи машини Тюрінга.
4. Створити інтерфейс програми.
5. Налаштувати інтерфейс для отримання даних, їх зберігання, виведення на екран нових даних.
6. Підключити модуль для роботи машини Тюрінга в проект і налаштувати його для роботи з графікою.
7. Визначити можливості застосування розробленого програмного модулю.

Потрібно ввести базові поняття про машину Тюрінга.

Машина Тюрінга(МТ) – це математичний об’єкт з нескінченою пам’яттю який має такий алгоритм(програму) який виконує операції над стрічкою яка розділена на комірки. Дії алгоритму описуються станами, алфавітом, і простими операціями. МТ був розроблений для формалізації поняття алгоритму.

Стрічка – це головна частина машини Тюрінга. Вона може бути будь якої довжини. Стрічка розділена на комірки. Стрічку розділяють на дві частини, вони відрізняються нумерацією комірок. Зліва від’ємні комірки а справа додатні. Нумерація комірок не впливає на роботу машини.

Комірка – це місце на стрічці де розміщається окремий символ з слова. Пустий символ позначається Кожна комірка має свій унікальний номер.

Слово – це вхідна інформація яка посимвольно розміщається на стрічці. Всі операції машина буде робити над цим словом, в процесі роботи машини слово буде змінюватися і в кінці ми отримаємо відповідь.

Алфавіт – це множина символів які використовуються в стрічці і в слові. Цю множину позначають літерою – Г. Всі операції програми можуть відбуватися тільки над цими символами. Якщо в алгоритмі є символ якого немає в алфавіті то це неправильний алгоритм.

Стани – це перелік простих операцій. Кожна операція з переліку належить конкретному символу. Два різні символи можуть мати одинакові операції. Станів в машині зазвичай кілька, їх кількість може бути будь якою, але вона мусить бути скінченою. Між станами можна переключатися. Всі стани починаються з літери q далі іде номер стану. Нумерація станів починається з 0. Для виходу з програми використовують назву неіснуючого стану. Множину станів позначають літерою – Q.

Прості операції – це операції які може виконувати машина, а саме: заміна символу на інший, перехід на інший стан, рух по стрічці. Рухатися по стрічці можна лише вліво, вправо і стояти на місці.

Вказівник(голівка, каретка) – вказує на комірку над якою виконуються операції.

1. Порівняння та аналіз аналогів Машини Тюрінга
   1. Нормальні алгоритми Маркова(НАМ)
      1. Як і машина Тюрінга нормальні алгоритми Маркова є одним з способів формального визначення алгоритмів. Між ними є кілька відмінностей в визначеннях які необхідно уточнити.
      2. Алфавіт – за Тюрінгом в алфавіті можуть бути тільки символи, але в Маркова допускається і наявність слів. Якщо є два алфавіти то один з них може належати іншому.
      3. Слова – будь яка послідовність символів. Для зручності їх можна позначати латинськими літерами P, Q, R, або цими ж літерами але з індексами. Одне слово може бути під словом іншого слова. Наприклад слово «тип» є під словом для «нетипова».
      4. Підстановки Маркова. Підстановка, формула – це елементарна дія заміни одного слова на інше. Вона задається впорядкованою парою слів (P,Q). В заданому слові R знаходиться перше входження P і замінюється на Q. P і Q можуть бути будь якими, навіть пустим словом. При підстановці змінюється лише слово P, а права і ліва частина слова R залишаються незмінними. Ліву і праву частину підстановки зазвичай розділяють символом .
      5. Формула застосовна для R – говорять коли ліва частина підстановки є під словом слова R. В протилежному випадку рахують, що формула незастосовна до R.
      6. Підстановка означає видалити з слова P. Важливо уточнити, що це не заміна на пустий символ, а видалення під слова.
      7. Підстановка означає дописати спереду слова Q.
      8. Нормальний алгоритм Маркова це не пуста, закінчена, впорядкований набір формул підстановки. Інколи його називають схемою. Її записують так:
      10. В НАМ не вказується де записується вхідне слово.
      11. НАМ працює таким чином. Записується набір формул підстановки типу (P,Q). Задається вхідне слово, позначимо його буквою R. По порядку з колекції береться підстановка, з неї беремо слово P і шукаємо його перше входження слові R. Якщо таке слово знайшли проводимо заміну і повторюємо перебір схеми з початку. Якщо таке слово не знайдено беремо наступний елемент і продовжуємо шукати. Якщо за повне проходження схеми не відбулося жодної заміни завершуємо роботу.
      12. Слово яке є результатом нормального алгоритму Маркова називається результатом підстановки.
      13. Алгоритм може зациклитися на деяких словах. В такому випадку кажуть, що алгоритм для слова не застосовують, або результат невизначений.
      14. Підстановки (#,P), (P,#), (#,#) є окремими випадками Марковських підстановок.
   2. Машина Поста
      1. Машина Поста – це проста абстрактна обчислювальна машина.
      2. Деталі машини Поста:
      3. Безкінечна стрічка яка розділена на комірки або секції. Стрічку можна умовно розділити на дві частини. Зліва номера комірок від’ємні, а справа додатні. Номера комірок впорядковані за зростанням;
      4. Вказівник(каретка) зчитує або записує інформацію з комірки. Коли каретка стоїть рівно навпроти якоїсь комірки кажуть, що вказівник оглядає секцію, або тримає її в полі зору.

Кожна секція стрічки має два стани, або заповненою, така секція називається пустою, або не заповненою, відмічена. Заповнені комірки позначають символом V, а пусті нічим не позначають. Як альтернатива заповнені – 1, а пусті – 0.

Стан стрічки – це інформація про те які комірки пусті, а які заповнені і де стоїть каретка.

Проста програма машини Поста – конечний непустий список команд. Команди записуються за шаблоном: номер команди(і.), дія вказівника(К), номер наступної команди( Усі команди можна розділити на шість видів:

Рух вправо і перейти в j команду програми: ;

Рух вліво і перейти в j команду програми: ;

Запис мітки і перейти в j команду програми: ;

Знищення мітки і перейти в j команду програми: ;

Якщо секція пуста то перейти в команду , якщо мітка є перейти в команду :

Стоп: !

Після запуску програми можливі три результати:

Перший. Машина дійде до команди яку неможливо виконати. Це або записати мітку в непусту секцію, або стирання мітки з пустої комірки. Така програма зупиниться і тоді відбудеться без результативна зупинка.

Другий. Програма спокійно виконається. Це називається результативна зупинка.

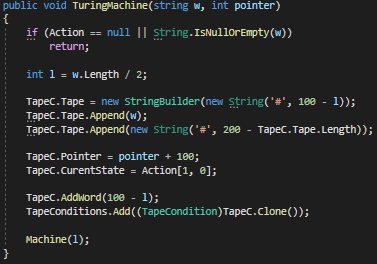
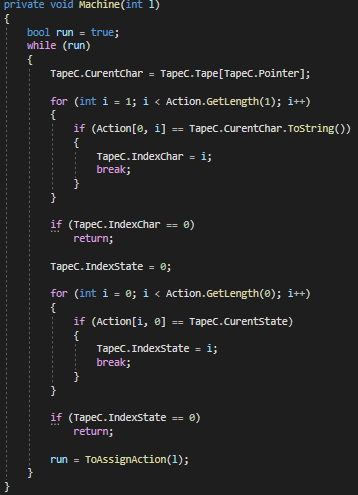
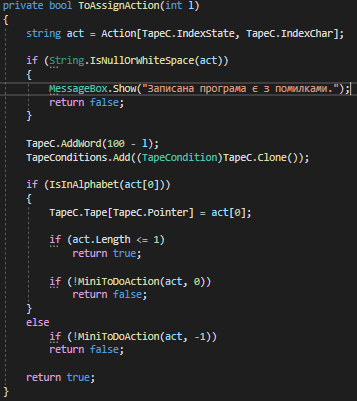
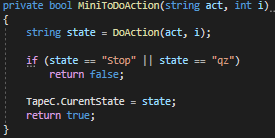
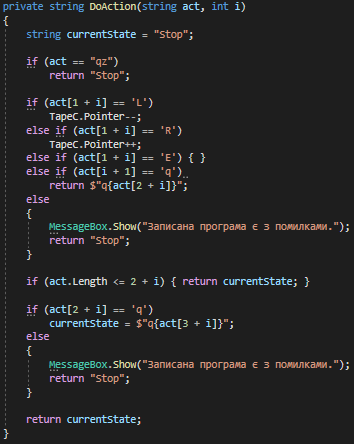
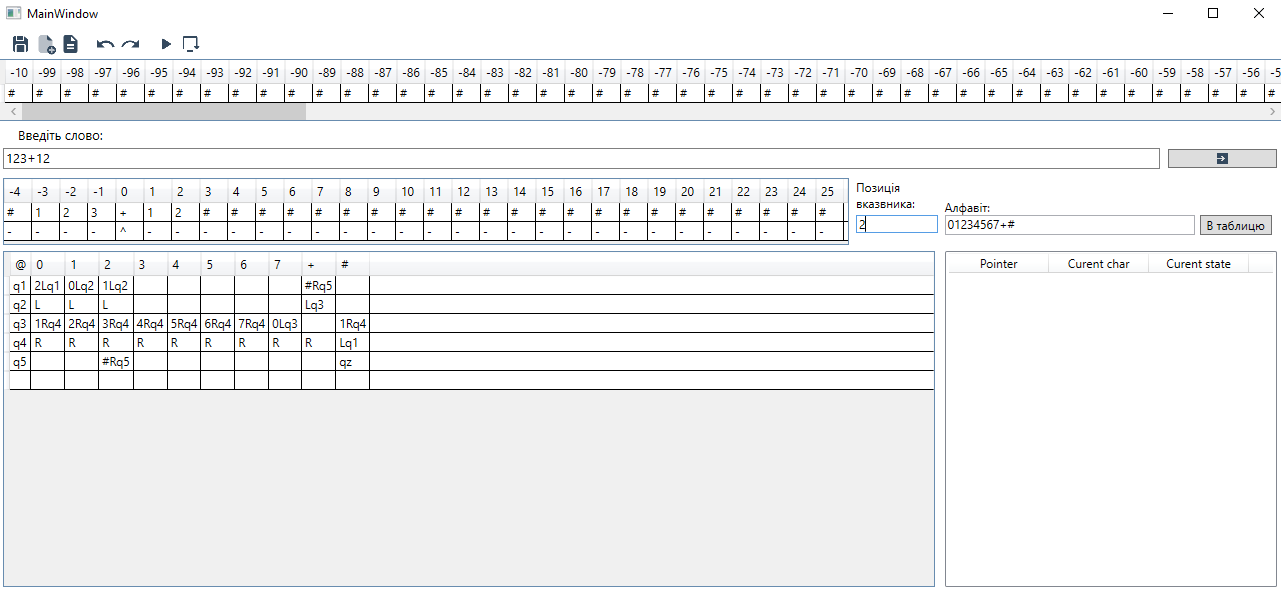
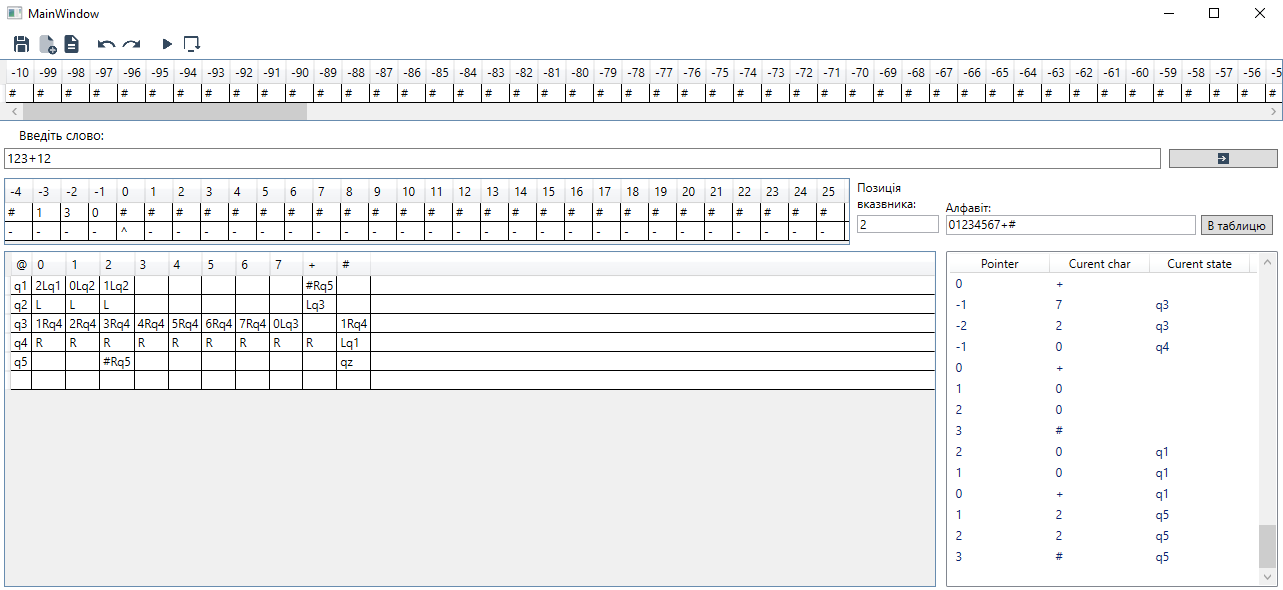
Третій. Програма зациклиться.

* 1. Висновок до першого розділу
     1. Машина Поста і нормальні алгоритми Маркова дуже похожі на машину Тюрінга адже всі вони є спробою узагальнити поняття алгоритм. Всі ці машини є взаємо замінними, тобто кожна з них може розв’язати одну і туж задачу і в кожної з них буде свій власний алгоритм її рішення. Швидкість виконання при цьому буде різна. Один спосіб завжди буде кращим за інший.
     2. Порівняємо машину Поста і машину Тюрінга
     3. В них є обох є безкінечна стрічка, але в МТ можна записувати будь який символ коли в машині поста немає символів, а є лише заповнені і пусті комірки. Так як в комірки нічого не можна записувати немає алфавіту. Відсутність алфавіту дуже сильно ускладнює написання програми для машини Поста і зменшує кількість варіацій для однієї і тієї ж задачі. Частково пробує врятувати ситуацію можливі варіації команд яких є шість і серед яких є команда якщо. Аналогів цієї команди немає ні в машині Тюрінга ні в нормальних алгоритмах Маркова. Ця команда є дуже важливою так як її наявність може спростити написання програми і збільшити можливості машини. При її відсутності в МТ і НАМ потрібно використовувати додаткові символи або стани і якось викручуватися.
     4. Ще однією спільною частиною є наявність станів. Які також дозволяють сильно спростити програмування.
     5. В цілому машина Поста є простішою версією машини Тюрінга. П’ять з шести команд для машини поста з легкість можна замінити в МТ, виключення є команда якщо, але якщо захотіти то і її можливо замінити. З іншої сторони машина Поста не при назначена для роботи з словами, так як відсутній алфавіт, тому її можна застосовувати і цілях відмінних від машини Тюрінга. Але якщо вважати що пусті клітинки позначаються ‘0’, а заповненні ‘1’, то і на машині Тюрінга можна запустити програму для машини Поста.
     6. Порівняємо нормальні алгоритми Маркова і машину Тюрінга.
     7. В машині Тюрінга і нормальних алгоритмах Маркова є багато відмінностей. НАМ як і МТ працює з словами, тому основною спільною деталлю є наявність алфавіту. Але навіть тут є нюанс, так як в алфавіті Маркова можуть бути цілі слова. І цю особливість дуже важко реалізувати в МТ, а без використання додаткових машин майже неможливо. Решта деталей відрізняються і через це одна задача може розв’язуватися швидше і простіше на одній з машин, в той час як на іншій сама програма буде громіздкою і виконуватиметься значно довше.

1. Проектування модуля для роботи емулятора машини Тюрінга
   1. Обрання способу запису Машини Тюрінга
      1. Є кілька способів представлення Машини Тюрінга, а саме: сукупністю команд, графом і таблицею відповідності. Розглянемо кожен з них.
      2. Подання Машини Тюрінга таблицею відповідності. Основна перевага цього способу є те що він найбільш зрозумілий для людини. В першому рядку таблиці записується посимвольно алфавіт, а першому стовбці записуються стани машини, або навпаки. Коли потрібно знайти операцію для символу на якомусь стані ми знаходимо комірку яка стоїть на перетині стовбця і рядка.
      3. Недоліком цього способу є те, що він не зручний для комп’ютера. Дуже часто в програмах для Машини Тюрінга є пусті клітинки, тобто для якогось символу на якомусь стані немає дії і це нормально. Такі пусті клітинки займають пам'ять в комп’ютері.
      4. Подання Машини Тюрінга сукупністю команд. В цьому способі ми маємо послідовність команд. Кожну команду можна розділити на дві частини які для зручності розділяють якимось символом наприклад стрілочкою . В лівій частині записується символ і стан, а в правій поста операція. Наприклад команду можна прочитати так: для символу ‘1’ в стані ‘’ виконати операцію: замінити на символ ‘0’, перейти в стан ‘’ і перемістити вказівник на одну клітинку в право.
      5. Перевагою такого способу є те що комп’ютеру з ним працювати простіше ніж з таблицею відповідності, але людині інколи важко читати такий алгоритм.
      6. Подання машини Тюрінга графом. В цьому способі кожній вершині графа ставиться в відповідність один з станів. А команди позначаються дугами.
      7. Недоліком такого способу є те, що він складний в реалізації. Крім того людині його важко читати.
      8. Для реалізації власного алгоритму я використовував подання програми таблицею відповідності. Він є найпростішим в читані і реалізації. Крім того для сучасних комп’ютерів лишня пам'ять яка буде використовуватися в програмі не є проблемою і це не сказиться на швидкодії емулятора.
      9. При виконанні алгоритму нам потрібно буде зберігати ти деяку інформацію. Це буде така інформація: стан стрічки, позиція вказівника, стан на якому зараз знаходимося, символ під вказівником, програма записана таблицею відповідності. Доцільно буде розділити всю роботу на методи так як кожен з них виконує конкретне завдання.
   2. Реалізація покрокового виконання програми
      1. Реалізовуючи цю частину завдання я не буду створювати додаткові методи або сильно редагувати основний алгоритм. І покрокової реалізації в прямому сенсі цих слів не буде. Але для кінцевого користувача все буде виглядати так ніби алгоритм працює крок за кроком.
      2. Щоб це зробити можна на кожному кроці записувати всі важливу інформацію в окремі колекції. Якщо користувач захоче подивитися на виконання алгоритму детальніше показувати нову інформацію для користувача тільки коли він про це попросить.
   3. Блок-схема початку виконання Машини Тюрінга
      1. Цей метод необхідний щоб отримати вхідні данні, а саме стрічку і позицію вказівника. Також буде перевіряти чи заповнені всі необхідні данні (рис.2.1).
      2. 
      3. Рисунок 2.1 – Блок-схема початку виконання Машини Тюрінга
   4. Блок-схема основного методу з основною роботою
      1. В цьому методі запускається цикл кожний прохід якого рівний одному кроці на стрічці (рис.2.2). На кожній ітерації програма зчитує символ з комірки під вказівником, бере стан на якому зараз знаходиться програма і передає операцію з комірки яка знаходиться на перетині стовбця з символом і рядка з станом в наступний метод.
      2. 
      3. Рисунок 2.2 – Блок-схема основного методу роботи Машини Тюрінга
   5. Блок-схема методу який аналізує і виконує операцію
      1. Цей метод оприділяє дії порівнюючи операцію з шаблоном (рис.2.3). Шаблон може бути таким: символ на який потрібно замінити поточний, крок вліво, вправо або стояти на місці і назва наступного стану. Цей шаблон використовується в цій програмі але він може бути і іншим.



* + 1. Рисунок 2.3 – Блок-схема методу для аналізу і операції і її виконання
  1. Висновок до другого розділу
     1. Всі ці методи мусять знаходитися в одному класі. Також в ньому будуть змінні, які були вже перераховані, крім них можуть знадобитися ще кілька додаткових які необхідні для зручності написання коду. Крім того їх всіх потрібно зберігати в колекціях для реалізації по-крокового виконання. Так як створювати для кожної змінної окрему колекцію і слідкувати за їх відповідністю буде важко доцільно буде створити новий клас або структуру де буде зберігатися вся ця інформація. Тоді нам необхідно буде створити лише одну колекцію. Так як алгоритм роботи програми вже готовий можна приступати до її написання.

1. Реалізація та тестування емулятора машини Тюрінга
   1. Реалізація допоміжного класу TuringCondition
      1. В цьому класі потрібно описати змінні і властивості для них. Це будуть такі приватні змінні:
      * tape – стрічка, так як в стрічці в одній комірці мусить бути один символ доцільно буде використовувати StringBuilder.
      * word типу StringBuilder яка зберігає слово.
      * curentState – стрічка яка зберігає назву стану.
      * curentChar – стрічка яка зберігає символ в комірці під вказівником.
      * indexState і indexChar – ці дві змінні зберігають індекс позиції в таблиці суміжності, для indexState це номер рядка, а для indexChar номер стовбця. Вони необхідні для зручності так як ми не можемо перейти в масиві в клітинку лише за назвами стовбця і строки.
      * pointer – зберігає позицію вказівника.
      1. Для кожної змінної потрібно створити публічну властивість. Також добавимо властивість RealPointer вона буде доступна тільки для читання і повертатиме значення (pointer-100). Ця змінна необхідна щоб показувати значення вказівника на стрічці, яка невідмінну від масиву має, як додатні значення, так і від’ємні.
      2. Також в цьому класі реалізований інтерфейс ICloneable. Його ми використаємо при створені копій екземпляра класу. Добавимо один пустий конструктор в якому створюються нові екземпляри об’єктів tape і word.
   2. Реалізація основного класу Turing
      1. Реалізуємо такі змінні і властивості:
      * string[,] action в ній зберігається таблиця відповідності.
      * tapeC - екземпляр класу TuringCondition. Ця змінна буде зберігати всю необхідну інформацію на кроці.
      * turingConditions – колекція екземплярів класу TuringCondition. Для її реалізації можна використати List<>, але так як елементи цієї колекції мусять відображатися в програмі використаємо колекцію ObservableCollection<>. Її основна відмінність в, тому що при зміні елементів колекції вона повідомить про це елементи які відображають її вміст і вони зразу виведуть його.
      1. Створимо властивості для цих змінних і реалізуємо два конструктори. Перший конструктор не буде приймати ніякої інформації і в тілі буде створювати нові екземпляри для змінних tapeC і tapeConditions. Другий приймає двовимірний масив строк, присвоює його значення змінній action і викликає попередній конструктор.
      2. Перший метод в класі TuringMachine(string w, int pointer). Змінна w – це вхідне слово, а pointer – позиція вказівника (рис 3.1).
      3. В цьому методі спершу відбувається перевірка на наявність програми заданої користувачем і введене ним слово. Далі записуємо отримані значення в змінні властивості поля TapeC. Також додаємо перший крок в TapeConditions. Викликаємо метод Machine(int l).
      4. 
      5. Рисунок 3.1 – Метод перевірки вхідних даних
      6. В метод Machine(int l) (рис. 3.2) передається значення половини довжини слова. Створюємо зміну run якій присвоюємо значення true і запускаємо цикл while і передаємо в нього цю змінну.
      7. Далі знаходимо символ під вказівником, записуємо значення в TapeC.CurrentChar і перевіряємо чи він є в алфавіті якщо є тоді записуємо його позицію в таблиці відповідності в TapeC.IndexChar і виходимо з циклу. Якщо символ не знайдено тоді завершуємо роботу програми.
      8. Шукаємо чи є теперішній стан TapeC.CurrentState в таблиці і якщо знаходимо його то записуємо позицію в TapeC.IndexState і виходимо з циклу, якщо не знайшли то завершуємо роботу програми.
      9. Викликаємо метод ToAssignAction(int l). Цей метод повертає bool тип його значення ми напишемо в зміну run якщо метод повертає true то програма продовжує виконуватися, в іншому випадку метод закінчить виконуватися і програма закінчить свою роботу.
      10. 
      11. Рисунок 3.2 – Основний метод програми
      12. Метод ToAssignAction(int l) (рис.3.3) приймає половину довжини вхідного слова. Використовуючи IndexChar і IndexState знайдемо операцію яку необхідно виконати. Значення операції записуємо в нову змінну act.
      13. Перевіряємо чи є якась операція в цій клітинці, якщо так виконуємо програму, якщо ні закінчуємо роботу методу і повідомляємо користувача про помилку.
      14. TapeConditions добавляємо новий елемент який буде глибоким клоном того який ми маємо зараз.
      15. Далі ми порівнюємо act з шаблоном. Перший символ в шаблоні мусить бути символ на який потрібно замінити. Якщо це такий символ ми замінюємо в стрічці символ під вказівником на новий: TapeC.Tape[TapeC.Pointer] = act[0]. Викликаємо метод який потрібний для уникнення повтору коду MiniToDoAction(string act, int i)(рис. 3.4). Далі з цього методу викликаємо метод DoAction(string act, int i) який продовжить порівнювати act з шаблоном.
      16. Якщо це не символ з алфавіту тоді ми просто викликаємо метод MiniToDoAction(string act, int i).
      17. 
      18. Рисунок 3.3 – Метод для аналізу операції
      19. 
      20. Рисунок 3.4 – Проміжний метод
      21. Метод DoAction(string act, int i) продовжує порівнювати операцію по шаблону. Друга частина шаблону – це зробити крок вліво, вправо, або стояти на місці. Третій - це перейти на інший стан.
      22. В act зберігається операція яка виконується зараз, i потрібна щоб від коректувати порівняння в методі. Якщо в програмі необхідно замінити перший символ тоді в метод замість і передається 0, тоді Алгоритм шукатиме другу частину шаблону на 2 символі, а якщо заміни не було то передається -1, бо в такому випадку перший символ позначатиме куди потрібно перемістити вказівник. Так само з знаходженням наступного стану.
      23. 
      24. Рисунок 3.5 – Метод продовжує аналіз операції і виконує її
   3. Опис реалізації інтерфейсу користувача
      1. Для реалізації інтерфейсу використовується паттерн MVVM. Він розділяє програму на три частини:
      * Model – де розміщена базова логіка програми. Ця частини обов’язково мусить наслідувати один з інтерфейсів INofityPropertyChanged або INofityCollectionChanged.
      * View – ця частина мусить реалізовувати інтерфейс програми. В нашому випадку – це код xaml.
      * ViewModel – в ньому відбувається основна логіка програми. І він є проміжним між Model і View.
      1. Частина View. Інтерфейс програми будемо реалізовувати використовуючи платформу розробки WPF. В програмі використовуватимемо такі елементи: Button, TextBox, TextBlock, DataGrid, ListView. TextBox – потрібен для отримання інформації від користувача, TextBlock – статичне поле яке показує текст, DataGrid – таблиця, яка використовується двома способами: перший – виводить стрічку на екран, другий – в нього записується програма користувача. ListView – виводить інформацію про кожен крок.
      2. Частина Model. Клас ViewModelBase реалізує інтерфейс INofityPropertyChanged.
      3. Частина ViewModel. Клас MainViewModel через цей клас відбуваються всі операції так як в ньому реалізовані метод Click для кожної кнопки. Має кілька змінних і полів до них прив’язується інформація з елементів в xaml. Це інформація про стрічку, таблиця відповідності, використовують тип DataTable, перелік всіх кроків, ViewList. Ще є кілька екземплярів класів, а саме Turing і новий клас WorkWithTable.
      4. WorkWithTable немає полів, але в ньому багато методів які працюють з таблицями. Всі ці методи можна розділити на дві групи, а саме: повертає DataTable з якоїсь колекції і повертає якусь колекцію з DataTable.
   4. Приклад роботи програми
      1. Запишемо програму яка буде знаходити суму двох чисел які розділені знаком ‘+’ зліва буде число записане в вісімковій системі числення, а з права в трійковій. Відповідь буде в вісімковій. На першому скріншоті (рис. 3.6) вже введено алфавіт, програма і слово «123+12». На другому скріншоті (рис. 3.7) видно результат операції і справа знизу можна подивитися історію операцій. Перевіримо чи відповідь правильна. Переведемо спочатку числа в десяткову систему і . Додаймо числа . Переведемо в вісімкову систему . Відповідь зійшлася.
      2. 
      3. Рисунок 3.6 – Вікно програми з заповненими даними
      4. 
      5. Рисунок 3.7 – Вікно програми в якому вже відображається відповідь
   5. Висновок до третього розділу
      1. Я реалізував машину Тюрінга яка може розв’язувати прості завдання. При написанні програми використовував паттерн MVVM який спростив проектування програмного продукту.

Висновки

* + 1. Існує не так багато емуляторів машини Тюрінга, через це комусь буде простіше знайти мою програму і її вона зацікавить. Якщо порівнювати з тими програмами які є вже то більшість з них браузерні і не будуть працювати без підключення до інтернету. Моїй програмі підключення до мережі не потрібне. Крім того частина з них має інший спосіб запису програми. Це може відбуватися через чогось похожого на текстовий редактор і при цьому користувач мусить зробити це правильно бо через один лишній пробіл, або через його відсутність програма просто не запуститься. В моїй реалізації алгоритм записується через таблицю, що є більш зручно для запису програми.
    2. Для реалізації інтерфейсу використовувалася платформа для створення графічного інтерфейсу WPF. Вирішено було використовувати її через те що вона зараз більш розповсюджена ніж WindowsForm, і через це обновляти програму буде не так важко. Також був використаний паттерн MVVM. Він дозволив значно спростити створення програмного продукту.
    3. В першу чергу програму можна використовувати для навчання або для саморозвитку. Також основний клас Turing може знадобитися при роботі з строками. В програму можна покращити додавши кілька способів запису алгоритму, змінити дизайн наприклад видаливши запис алфавіту і зразу записувати його в таблицю, можна задавати вказівник не координатою комірки, по нажаті на саму комірку, або переміщенням повзунка який буде під стрічкою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* + 1. Авраменко В. С.Теорія алгоритмів. Курс лекцій. Черкаси: ЧНУ імені Богдана Хмельницького, 2020. 195 с.
    2. Авраменко В. С., Авраменко А. С.Теорія алгоритмів. Навчальний посібник. Черкаси: ЧНУ імені Богдана Хмельницького, 2019. 200 с.
    3. Теорія алгоритмів Лекція 1 (вводна) [Електронний документ]<http://www.study.cc.ua/files/ta1_1_at/Algorythms_theory.pdf>.
    4. В.Н Пильщитков, Машина Тьюринга и алгоритмы Маркова. Решение задач [Текст]/ В.Н Пильщитков, В.Г. Абрамов, А.А. Вылиток, И.В. Горячая – Издательский отдел факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова, 2006, 47ст.
    5. Машина Тьюринга. Режим доступу: <https://ru.wikipedia.org/wiki/Машина_Тьюринга>.
    6. Нормальный алгоритм. Режим доступу: <https://ru.wikipedia.org/wiki/Нормальный_алгоритм>.
    7. В.А. Успенский, МАШИНА ПОСТА [Текст]/ В.А. Успенский – Москва «Наука», 1979, 96 с.
    8. Т.К. Кацаран, МАШИНА ТЬЮРИНГА И РЕКУРСИВНЫЕ ФУНКЦИИ, [Текст]/ Т.К. Кацаран, Л.Н. Строева – Издательско-полиграфический центр Воронежского государственного университета.