Міністерство освіти і науки України

Житомирський державний технологічний університет

Факультет інформаційно-комп'ютерних технологій

Кафедра програмного забезпечення систем

КУРСОВИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

на тему:

Емулятор «машини Тюрінга»

з дисципліни: «Основи програмної інженерії»

Студентки 2 курсу ПІ-52 групи

напряму підготовки 6.050103

Ксенича Євгенія Геннадійовича

Керівник: к.т.н., доцент, декан факультету інформаційно-комп’ютерних технологій

Морозов А. В.

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_

Оцінка ECTS: \_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_ Морозов А. В.

\_\_\_\_\_\_\_\_ Левківський В. Л

\_\_\_\_\_\_\_\_ Гришкун Є. О.

Житомир – 2016

ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційно-комп'ютерних технологій

Кафедра програмного забезпечення систем

Освітній рівень: бакалавр

Напрям підготовки: 6.050103 «Програмна інженерія»

|  |  |
| --- | --- |
|  | «ЗАТВЕРДЖУЮ» |
|  | Зав. кафедри ПЗС |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Панішев |
|  | “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_р. |

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Ксеничу Євгенію Геннадійовичу

1. Тема роботи: емулятор «машини Тюрінга».

керівник роботи: Морозов Андрій Васильович, к. т. н., доцент, декан факультету інформаційно-комп’ютерних технологій,

затверджені наказом вищого навчального закладу від “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_р. №\_\_\_\_\_

2. Строк подання студентом: “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_р.

3. Вихідні дані до роботи: Розробити програмний продукт емулятор «машини Тюрінга»

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
| 1. Технічне завдання |
| 2. Аналіз аналогічних розробок |
| 3. Математично-логічна модель програмного продукту |
| 3. Алгоритми роботи програми – загальний та функціональний |
| 4. Розробка інтерфейсу програмного продукту |
| 5. Опис роботи програми |
| 6. Програмне дослідження продукту |

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання “18” жовтня 2016р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів випускної роботи | Строк виконання етапів роботи | Примітка |
| 1 | Постановка задачі | 18 жовтня 2016 -  18 жовтня 2016 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 25 жовтня 2016 -  3 листопада 2016 |  |
| 3 | Формулювання технічного завдання | 5 листопада 2016 -  8 листопада 2016 |  |
| 4 | Опрацювання літературних джерел | 10 листопада 2016 -  17 листопада 2016 |  |
| 5 | Проектування структури | 18 листопада 2016-  25 листопада 2016 |  |
| 6 | Написання програмного коду | 26 листопада 2016 -  7 грудня 2016 |  |
| 7 | Відлагодження | 8 грудня 2016 -  10 грудня 2016 |  |
| 8 | Написання пояснювальної записки | 12 грудня 2016 -  22 грудня 2016 |  |
| 9 | Захист |  |  |

|  |  |  |
| --- | --- | --- |
| **Студент** | **\_\_\_\_\_\_\_\_\_** (підпис) | **Є.Г. Ксенич \_**  (прізвище та ініціали) |
| **Керівник роботи** | **\_\_\_\_\_\_\_\_\_** (підпис) | **А.В. Морозов \_**  (прізвище та ініціали) |

РЕФЕРАТ(АНОТАЦІЯ)

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

«Емулятор машини Тюрінга»

Розроб.

Ксенич Є.Г.

Перевір.

Керівник

Морозов А.В.

Н. Контр.

Зав. Каф..

Морозов А.В.

Літ.

Аркушів

ЖДТУ, група ПІ-52

Курсовий проект присвячений розробці алгоритму та зовнішнього вигляду програмного продукту – Емулятор машини Тюрінга. Метою є створення програмного продукту, який задовольнятиме всі поставлені до нього вимоги.

В поданій пояснювальній записці представлено три розділи, які включають в себе повну інформацію про представлений продукт, його проектування, розробку, функціонал. В першому розділі проводиться аналіз проблем, які постають перед написанням продукту, а також створюється підхід до їх подальшого вирішення, формується уявлення про розробку продукту, його інтерфейс та функціональну частину. Наступний розділ є основним і повністю розкриває алгоритм програмного продукту, чітко та послідовно описує етапи його створення. Третій розділ описує роботу вже створеного продукту, показує його функціонал, проводиться остаточне тестування програми та перевіряється відповідність поставленим вимогам.

Текстова частина складається з 35 сторінок, 12 рисунків, 3 додатків та 16 джерел використаної літератури.

.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ………………………………………..

ВСТУП………………………………………………………………………...

1 АНАЛІЗ СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ…………………

* 1. Аналіз аналогічних програм………………………………………
  2. Математично-логічна модель програмного продукту…………..
  3. Технічне завдання**…………………………………………………**

Висновки з першого розділу………………………………………….

2 ПРОЕКТУВАННЯ І РОЗРОБКА ЕМУЛЯТОРА МАШИНИ

ТЮРІНГА ………………………....................................................................

2.1 Загальний алгоритм роботи програмного забезпечення………..

2.2 Функціональний алгоритм програмного забезпечення…………

2.3 Розробка програмного забезпечення……………………………..

2.3.1 Розробка функціональної частини……………………….

2.3.2 Розробка інтерфейсу………………………………………

Висновки з другого розділу…………………………………………..

3 ОПИС РОБОТИ ТА ТЕСТУВАННЯ ПРОГРАМИ……………………….

3.1 Опис роботи додатку……………………………………………….

3.2 Тестування додатку…………………………………………………

Висновки з третього розділу……………………………………………

ВИСНОВКИ……………………………………………………………………

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………..

ДОДАТКИ……………………………………………………………………...

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ

ВСТУП

В даній курсовій роботі було розроблено програму: емулятор машини тюрінга. Машина Тюрінга – це абстрактна обчислювальна машина, що була запропонована англійським математиком Аланом Тюрінгом ще в 1936 році.

В основі машини Тюрінга лежить необмежена у обидві сторони стрічка, що розділена на частини, та керуючий пристрій, що може записувати данні на пусту частину, стирати данні та переписувати вже записані данні.

Керуючий пристрій може переміщатися в будь-якому напрямку по стрічці, але лише керуючись особливими правилами переходів, що також задані у програмі, ці ж правила задають алгоритм дій програми. Правило застосовується відповідно до того, у якому стані зараз машина, та до того, який символ записаний безпосередньо в тій клітинці програми, яка обробляється програмою.

Однією з вирішальних умов є те, що число станів керуючого пристрою чітко задано.

1. АНАЛІЗ СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ
   1. Аналіз аналогічних програм.

Машину Тюрінга було запропоновано ще у 1936 році, тому за цей час

було запропоновано багато варіацій цієї програми.

Розглянемо один з пркладів.

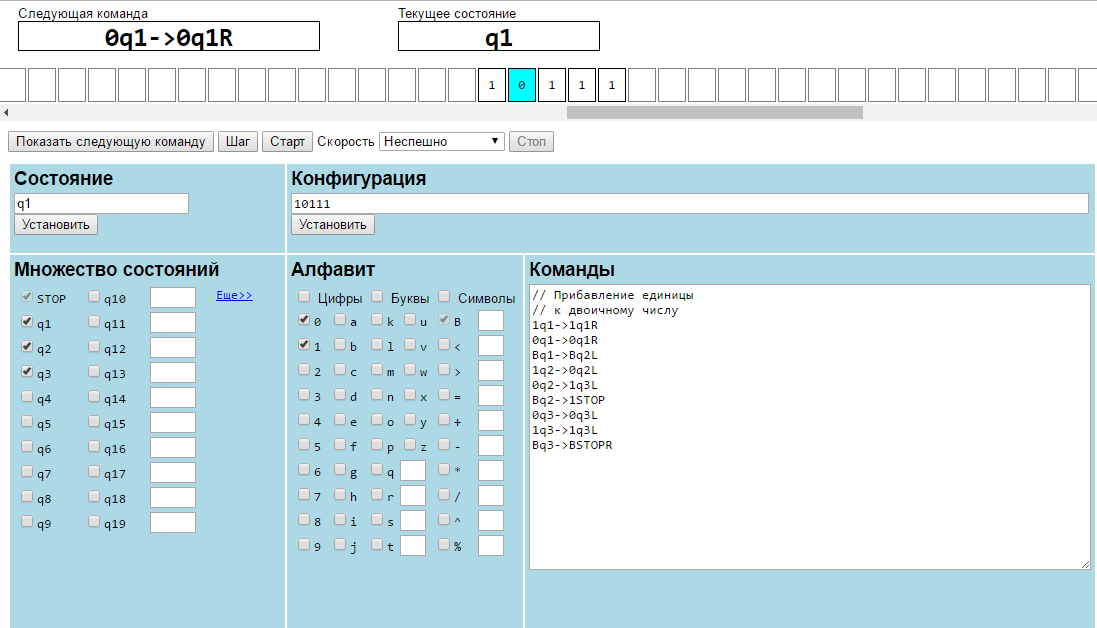


Рис. 1 – Емулятор машини Тюрінга.

У даному прикладі продемонстрована класична машина Тюрінга, в якій є основні структурні елементи, а саме можливість задавати усі можливі стани, можливість задати початкову конфігурацію, встановити початковий стан, встановити можливий алфавіт та можливість задавати команди, які в результаті будуть виконані програмою.

Також до позитивних сторін даної програми можна віднести те, що для зручності користувача в окремі блоки виводиться інформація про поточний стан машини, та про наступну команду, яка буде виконана безпосередньо після поточної команди. Велике значення має можливість вибору між покроковим проходженням, та повним проходженням. А в свою чергу повне проходження має можливість вибору швидкості, та можливість зупинки повного проходження у будь-який момент часу.

Наступний приклад дещо відрізняється від попереднього.

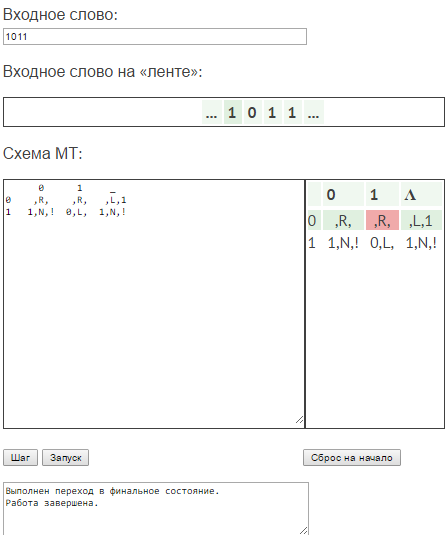


Рис. 2 – Емулятор машини Тюрінга.

У данного прикладу збережені основні признаки машини Тюрінга, однак недоліком є те, що ми не можемо задати множину станів. Також до недоліків можна віднести відсутність вибору швидкості при повному проходженні, вона завжди моментальна.

До переваг можна віднести:

1. можливість вибору між покроковим та повним проходженням по стрічці
2. можливість скидування усіх параматрів машини Тюрінга до початкового стану.
   1. Математично-логічна модель програмного продукту.

Основна мета програми – демонстрація роботи машини Тюрінга.

Основні поняття: Машина Тюрінга, названа на честь англійського математика Алана Тюрінга. Вона містить всі елементи скінченного автомата, крім того, її вхідна стрічка нескінченна в обох напрямках, вона може переміщатися вздовж стрічки в обох напрямках.

За допомогою машини Тюрінга можна моделювати будь-який обчислювальний процес, який можна назвати алгоритмом. Це твердження відоме як тезис Черча-Тьюринга. За допомогою машини Тьюринга можна розпізнавати будь-які мови, визначені граматиками загального виду.

В початковому стані голівка, що читає і пише, встановлюється на ліву з непорожніх комірок. Машина працює так, що її дії визначаються функцією f. Машина зупиняється, якщо для поточної комбінації стану і вхідного символу функція f не визначена.

Може бути задана множина кінцевих станів. Кінцеві стани визначаються, в основному, коли машина моделює розпізнавання рядків.

Машина Тьюрінга може моделювати не тільки розпізнавання рядків мови, але і будь-які інші обчислювальні алгоритми.

* 1. Технічне завдання.

На даному етапі потрібно систематизувати та узагальнити інформацію щодо реалізації ПЗ що розробляється, яка вже була розглянута. А саме потрібно визначити завдання та інтерфейс програми.

Для відображення «активної» частини стрічки машини будемо використовувати компонент «TextBlock», елемент в основному використовується для виведення інформації, що не містить більше 1 абзацу, що цілком підходить для нас, адже нам знадобиться виводити лише по 1 знаку в 1 «TextBlock». в Текстова частина може бути добавлена за допомогою властивості «Text».

Для відображення різного роду написів будемо використовувати компонент «Label»

Для введення текстової інформацій будемо використовувати компонент «TextBox», даний компонент використовується для відображення, або зміни невідформатованого тексту.

На головному вікні програмного продукту будуть розміщені такі елементи:

1. Компонент «TextBlock» для відображення «активної» частини стрічки машини Тьюрінга.
2. Поля «TextBox» для введення деяких початкових значень.
3. Компонент «Label» для виведення підказок щодо елементів управління програмою.
4. Компонент «Button» для встановлення значень машини Тьюрінга.
5. Компонент «Button» для управління процесом обробки данних, а саме старту роботи машини, зупинки роботи машини та покрокового проходження по елементам стрічки.
6. Компонент «CheckBox» для вибору можливих станів та можливого алфавіту для роботи програми.
7. Компонент «RadioButton» для вибору швидкості проходження по стрічці машини.
8. Компонент «Grid» для групування елементів.

Далі визначимо дії, які будуть виконувати вище перераховані елементи.

Компоненти «Button» (4) використовуються для встановлення таких значень, як:

1. Конфігурація машини.
2. Стан машини.
3. Виконувані команди.

Компоненти «Button» (5) використовуються для:

1. Покрокового проходження.
2. Автоматичного проходження по стрічці.
3. Зупинки автоматичного проходження по стрічці.

За допомогою компонент «CheckBox» та «RadioButton» встановлюються значення для:

1. Швидкості проходження по стрічці.
2. Можливого алфавіту машини.
3. Можливих станів у роботі машини.

На вікна з результатами буде розміщено лише Компонент «ListView».

«ListView» - це елемент управління відображаючий список данних.

У ньому буде виводитися фактичний результат роботи програми у покроковому вигляді.

Данний елемент складатиметься з трьох структурних частин, у першій буде виведена фактична кнфігурація машини, у другій буде виведено номер змінюваного елемента, а у третій – команда машини, що буде застосована до даного змінюваного елемента.

Висновки з першого розділу

1. ПРОЕКТУВАННЯ І РОЗРОБКА ЕМУЛЯТОРА МАШИНИ ТЮРІНГА

2.1 Загальний алгоритм роботи програмного забезпечення.

На даному етапі розглянемо основний алгоритм роботи програми.

Загальна схема роботи приведена на рисунку 2.1.

Початок

Введення алфавіту, команд, стану, конфігурації та множини станів.

Обробка вхідних даних.

Перевірка даних.

Так

Ні

Виведення вікна результату.

Кінець

1

2

3

4

5

Рисунок 2.1 – Загальний алгоритм роботи програми.

Опис загального алгоритму ПЗ описано нижче.

Початок роботи програми починається безпосередньо після запуску.

Першим кроком у роботі з програмою є введення таких початкових значень як внесення конфігурації, задання значень алфавіту та множині станів машини Тьюрінга.

Після установлення кожного з значень проводиться перевірка на валідність внесених даних. Якщо данні не відповідають усім вимогам, то буде виведено відповідне повідомлення та буде не можливо проводити обробку даних, якщо ж дані відповідають усім встановленим вимогам, тоді користувач може переходити до обробки введених даних.

Є два варіанти проходження по стрічці:

1. Покрокове. Користувач має можливість пройтися по всій страчці у покроковому режимі, для кожного проходження потрібно натиснути на відповідний елемент «Button».
2. Повне проходження. При натисканні на відповідний елемент «Button» буде здійснено повне проходження по стрічці. Також користувач має можливість зупинити проходження у будь який момент між початком та кінцем роботи обробки данних.

Після того, як обробка даних буде завершена буде показано вікно з результатами. Там у покроковому варіанті буде продемонстровано роботу програми і виведено у зручному для користувача вигляді.

2.2 Функціональний алгоритм програмного забезпечення.

Функціональний алгоритм є більш детальним ніж загальний алгоритм і його складання потребує повного розуміння роботи усіх процесів у роботі даного ПЗ.

Нами будуть розглянуті основні алгоритми у роботі програми, а саме

Алгоритм перетворення одного знаку на стрічці машини, даний алгоритм приведений на рисунку

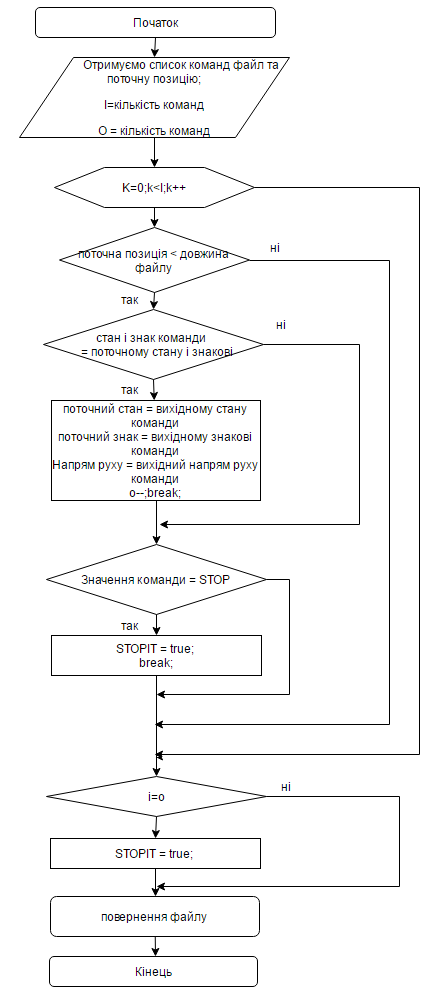


Рис 2. - Алгоритм перетворення одного знаку на стрічці машини.

2.3 Розробка програмного забезпечення.

2.3.2 Розробка інтерфейсу

На данному етапі розробки курсового проекту відбувається безпосередній перехід до розробки ПЗ.

Притримучись технічного завдання та вище описаних положень можно переходти до розробки додатку.

Почнемо безпосереднь зі створення ноаого проекту у середовищі Visual Studio, в якому буде здійснюватися розробка ПЗ. Створення нового проекту продемонстровано на рисунку …

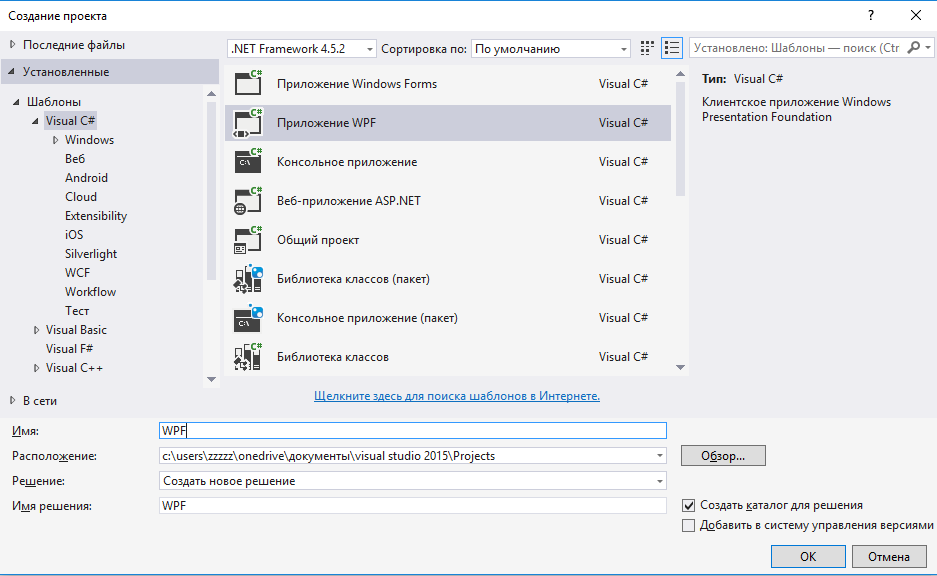


Рис .. – Створення проекту у Visual Studio.

Отримаємо пусту форму, вигляд якої показаний на рисунку… на основі якої буде здійснюватися розробка ПЗ.

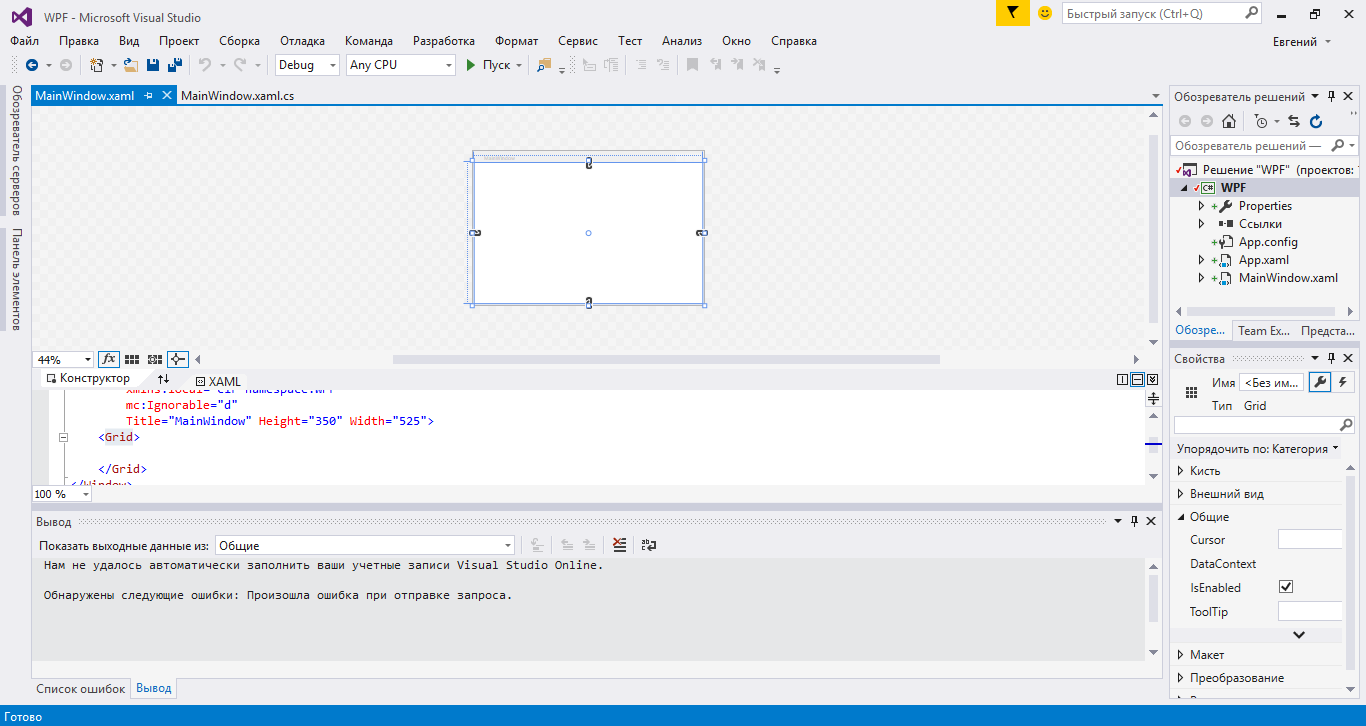
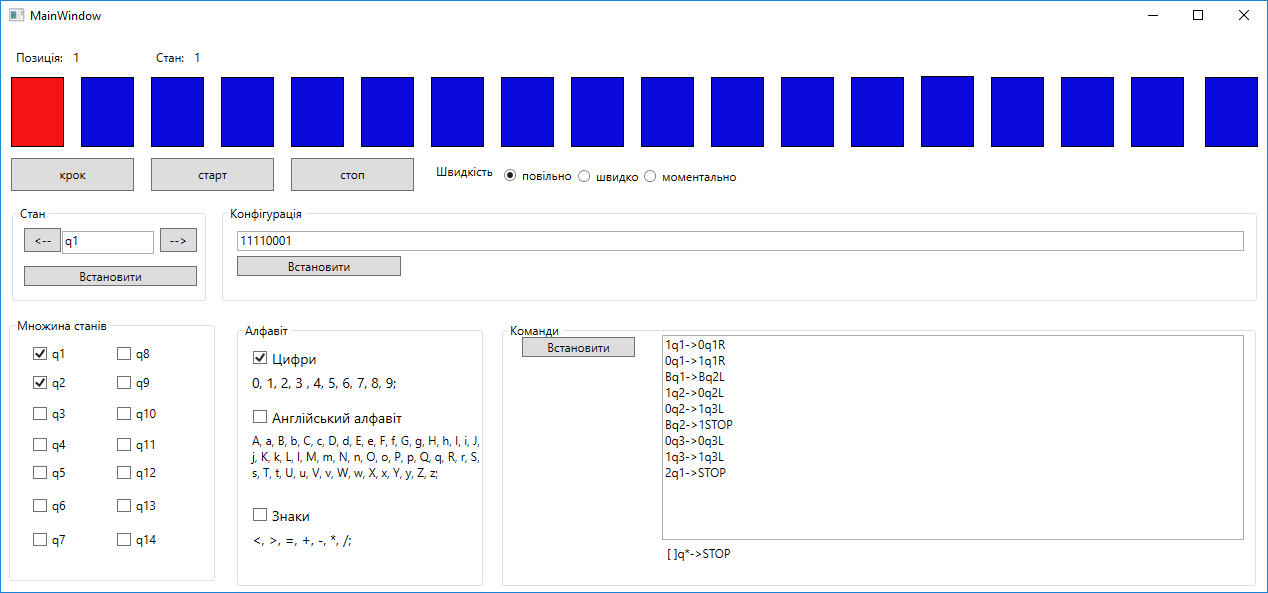
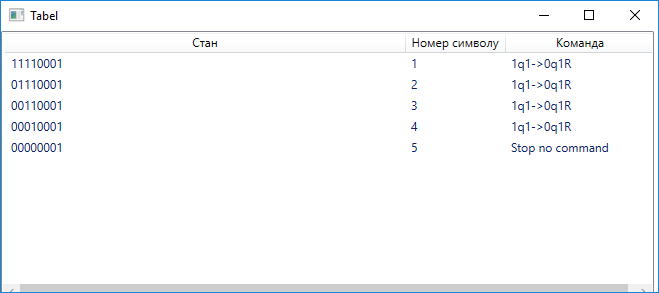


Рис … - Створений новий проект.

Тепер переходимо до розробки ПЗ. А точніше до розробки користувацького інтерфейсу програми. Спочатку створимо головне вікно програмного продукту на якому будуть розміщені усі елементи користувацького інтерфейсу. На рисунку ... приведено вигляд готового вікна даного програмного продукту, на якому розміщені всі елементи які були описані у розділі 1.3.

Після створення головного вікна додатку необхідно створити вікно з виведенням результатів роботи. Вигляд готового вікна приведено на рисунку …





2.3.1 Розробка функціональної частини.

На даному етапі розроблювання ПЗ залишилося лише розробити функціональну частину.

У даному розділі буде приведено основну частину коду додатку, що дасть повне уявлення про процес розроблення коду.

|  |  |
| --- | --- |
| Призначення коду | Програмний код функції |
| Алгоритм перетворення одного знаку на стрічці машини | int LR = position;  int tmp = position;  Regex tm1 = new Regex(@"([\dA-Za-z])q(\d\*)->([\dA-Za-z])(?:(?:q(\d\*))|(STOP))([RLrl])?");  MatchCollection n\_next = tm1.Matches(comand);  int size\_col = n\_next.Count;  int count = size\_col;  for (int i = 0; i < size\_col; i++)  {  if ((position - 1) < file.Length)  {  if (n\_next[i].Groups[1].Value == file[position - 1].ToString() && state.ToString() == n\_next[i].Groups[2].Value)  {  prev\_position = position;  prev\_file = file;  file = file.Remove(position - 1, 1).Insert(position - 1, n\_next[i].Groups[3].Value);  this.comand = n\_next[i].Groups[0].Value;  if (n\_next[i].Groups[6].Value == "R" || n\_next[i].Groups[6].Value == "r") LR++;  if (n\_next[i].Groups[6].Value == "L" || n\_next[i].Groups[6].Value == "l") LR--;  state = int.Parse(n\_next[i].Groups[4].Value);  count--;  break;  }  if (n\_next[i].Groups[5].Value == "STOP" || n\_next[i].Groups[5].Value == "stop" || n\_next[i].Groups[5].Value == "Stop")  {  STOPIT = true;  this.comand = "Stop";  break;  }  }  }  if (count == size\_col)  {  STOPIT = true;  this.comand = "Stop no command";  }  position = LR;  return file; |

Додаткові алгоритми

|  |  |
| --- | --- |
| Призначення коду | Програмний код функції |
| Функція вибірки 18 знаків зі стрічки для відображення | for (int i = 0; i < 18; i++) Out[i] = "";  int size\_f = file.Length - 1;  int more = 0, tmp = curent\_pos;  while (tmp > 18)  {  tmp -= 18;  more++;  }  for (int i = 0; i < 18; i++)  {  if ((more \* 18) + i < file.Length) Out[i] = file[(more \* 18) + i].ToString();  else Out[i] = " ";  }  return Out; |
| Функція відображення 18 знаків. | if(curent\_pos > 18) while (curent\_pos > 18) curent\_pos -= 18;  Dispatcher.Invoke(() => {  textBlock\_1.Text = mas[0];  if(curent\_pos==1)textBlock\_1.Background = new SolidColorBrush(Colors.Red);  else textBlock\_1.Background = new SolidColorBrush(Colors.Blue);  textBlock\_2.Text = mas[1];  if (curent\_pos == 2) textBlock\_2.Background = new SolidColorBrush(Colors.Red);  else textBlock\_2.Background = new SolidColorBrush(Colors.Blue);  textBlock\_3.Text = mas[2];  if (curent\_pos == 3) textBlock\_3.Background = new SolidColorBrush(Colors.Red);  else textBlock\_3.Background = new SolidColorBrush(Colors.Blue);  textBlock\_4.Text = mas[3];  if (curent\_pos == 4) textBlock\_4.Background = new SolidColorBrush(Colors.Red);  else textBlock\_4.Background = new SolidColorBrush(Colors.Blue);  textBlock\_5.Text = mas[4];  if (curent\_pos == 5) textBlock\_5.Background = new SolidColorBrush(Colors.Red);  else textBlock\_5.Background = new SolidColorBrush(Colors.Blue);  textBlock\_6.Text = mas[5];  if (curent\_pos == 6) textBlock\_6.Background = new SolidColorBrush(Colors.Red);  else textBlock\_6.Background = new SolidColorBrush(Colors.Blue);  textBlock\_7.Text = mas[6];  if (curent\_pos == 7) textBlock\_7.Background = new SolidColorBrush(Colors.Red);  else textBlock\_7.Background = new SolidColorBrush(Colors.Blue);  textBlock\_8.Text = mas[7];  if (curent\_pos == 8) textBlock\_8.Background = new SolidColorBrush(Colors.Red);  else textBlock\_8.Background = new SolidColorBrush(Colors.Blue);  textBlock\_9.Text = mas[8];  if (curent\_pos == 9) textBlock\_9.Background = new SolidColorBrush(Colors.Red);  else textBlock\_9.Background = new SolidColorBrush(Colors.Blue);  textBlock\_10.Text = mas[9];  if (curent\_pos == 10) textBlock\_10.Background = new SolidColorBrush(Colors.Red);  else textBlock\_10.Background = new SolidColorBrush(Colors.Blue);  textBlock\_11.Text = mas[10];  if (curent\_pos == 11) textBlock\_11.Background = new SolidColorBrush(Colors.Red);  else textBlock\_11.Background = new SolidColorBrush(Colors.Blue);  textBlock\_12.Text = mas[11];  if (curent\_pos == 12) textBlock\_12.Background = new SolidColorBrush(Colors.Red);  else textBlock\_12.Background = new SolidColorBrush(Colors.Blue);  textBlock\_13.Text = mas[12];  if (curent\_pos == 13) textBlock\_13.Background = new SolidColorBrush(Colors.Red);  else textBlock\_13.Background = new SolidColorBrush(Colors.Blue);  textBlock\_14.Text = mas[13];  if (curent\_pos == 14) textBlock\_14.Background = new SolidColorBrush(Colors.Red);  else textBlock\_14.Background = new SolidColorBrush(Colors.Blue);  textBlock\_15.Text = mas[14];  if (curent\_pos == 15) textBlock\_15.Background = new SolidColorBrush(Colors.Red);  else textBlock\_15.Background = new SolidColorBrush(Colors.Blue);  textBlock\_16.Text = mas[15];  if (curent\_pos == 16) textBlock\_16.Background = new SolidColorBrush(Colors.Red);  else textBlock\_16.Background = new SolidColorBrush(Colors.Blue);  textBlock\_17.Text = mas[16];  if (curent\_pos == 17) textBlock\_17.Background = new SolidColorBrush(Colors.Red);  else textBlock\_17.Background = new SolidColorBrush(Colors.Blue);  textBlock\_18.Text = mas[17];  if (curent\_pos == 18) textBlock\_18.Background = new SolidColorBrush(Colors.Red);  else textBlock\_18.Background = new SolidColorBrush(Colors.Blue);  HEAD();  });  }  void HEAD()  {  label\_out\_position.Content = tr.Position.ToString();  label\_out\_stan.Content = tr.State.ToString();  } |
| Функція проходження 1го кроку по стрічці. | try  {  Dispatcher.Invoke(() =>  {  CheackB();  if (data)  {  Add\_in\_Table();  tr.FILE = tr.Transl(All\_Comand);  Stroka st = new Stroka();  string[] a1 = st.change(tr.FILE, tr.Position);  HeadText(a1, tr.Position);  if (tr.FILE.Length + 1 <= tr.Position || tr.STOP || tr.Position < 1) {  Add\_in\_Table();  stop\_start = false;  Event\_Table();  win2.Show();  tr.Position = 1; tr.STOP = false;  data = true;  win2 = new Tabel();  Event\_Table = null;  Event\_Table += win2.Set\_Table;  HeadText(a1, tr.Position);  tr.Default();  data = false;  }  }  });  }  catch (Exception)  {  System.Windows.MessageBox.Show("Сталася неочікувана помилка", "Помилка");  } |

Висновки з другого розділу

3 Опис роботи та тестування програми.

3.1 Опис роботи додатку

На даному етапі виконання курсового проекту вже реалізовано програмне забезпечення, тому потрібно пояснити принцип роботи з додатком.

Створений додаток складається з основного та допоміжного вікна.

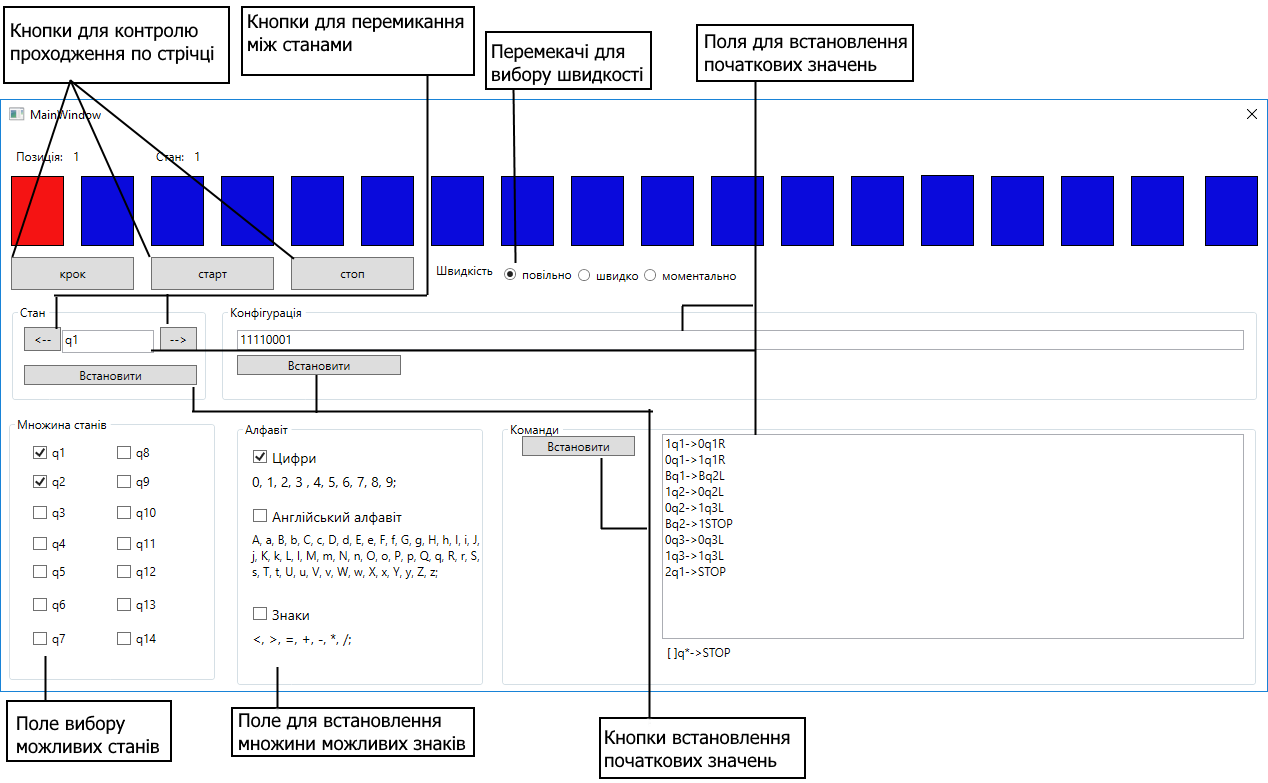


Рис 3.1. – Головне вікно програми.

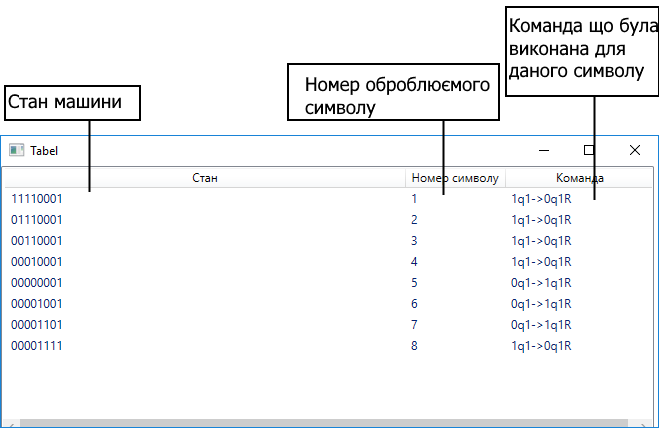


Рис 3.1. – Додаткове вікно програми.

3.2 Тестування додатку

Висновки з третього розділу

ВИСНОВКИ