Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №6»**

“изучение устройства и функциональных особенностей шифровальной машины «Энигма»”

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

Мозолевский Александр Дмитриевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2023

**Разработать приложение-симулятор шифровальной машины, состоящей из клавиатуры, трех роторов и отражателя. Типы роторов (L – M – R) и отражателя Re следует выбрать из рис. 4.5 и 4.6 в соответствии со своим вариантом, представленным в таблице. Крайний правый столбец этой таблицы показывает, на какое число шагов (букв, i) перемещается соответствующий ротор при зашифровании одного (текущего) символа; число 0 означает перемещение соответствующего ротора на один шаг при условии, что расположенный правее ротор совершит один оборот**

Реализация ротора:

|  |
| --- |
| package enigma  type Rotor struct {  Alphabet []rune  Shift int  CountOfShifts int  }  func NewRotor(alphabet []rune, shift int) \*Rotor {  return &Rotor{Alphabet: alphabet, Shift: shift}  }  func (r \*Rotor) ShiftRotor(turnsCount int) int {  turns := 0  if r.CountOfShifts < len(r.Alphabet) {  r.CountOfShifts += r.Shift  } else {  r.CountOfShifts = len(r.Alphabet) - r.CountOfShifts  turns++  }  for i := 0; i < r.Shift+turnsCount; i++ {  temp := ' '  for j := 0; j < len(r.Alphabet)-1; j++ {  if j == 0 {  temp = r.Alphabet[len(r.Alphabet)-1]  r.Alphabet[len(r.Alphabet)-1] = r.Alphabet[j]  r.Alphabet[j] = r.Alphabet[j+1]  } else {  r.Alphabet[j] = r.Alphabet[j+1]  }  if j == len(r.Alphabet)-2 {  r.Alphabet[len(r.Alphabet)-2] = temp  }  }  }  return turns  }  func (r \*Rotor) PickStartPosition(startPosition rune) {  for i := 0; i < len(r.Alphabet); i++ {  if r.Alphabet[i] == startPosition {  r.ShiftRotor(i)  break  }  }  } |

Реализация рефлектора

|  |
| --- |
| package enigma  type Reflector struct {  Alphabet map[rune]rune  }  func NewReflector(alphabet map[rune]rune) \*Reflector {  return &Reflector{Alphabet: alphabet}  }  func (r \*Reflector) GetLetter(letter rune) rune {  for key, value := range r.Alphabet {  if key == letter {  return value  } else if value == letter {  return key  }  }  return 'x'  } |

Результат выполнения:

