

# Приложение для шифрования

Кузнецов Данила



# Цели и задачи


Реализовать программу, с помощью которой пользователь сможет зашифровать информацию различными способами шифрования.



# Описание программы


Моя программа была  
разработана на языке  
Python

```
1  from Vigenere import Vigenere
2  from Caesar import Caesar
3  from Atbash import Atbash
4  import sys
5  from PyQt5.QtWidgets import QWidget, QPushButton, QVBoxLayout, QHBoxLayout, QComboBox, QLabel, QLineEdit, QTextEdit, QApplication
6  from PyQt5.QtGui import QFont
7
8
9  class Encrypt(QWidget):
10     def __init__(self):
11         super().__init__()
12         self.setStyleSheet(
13             "background-color: {}".format('{}'.format('#ffffff')))
14
15         u1 = QVBoxLayout(self)
16         u2 = QHBoxLayout(self)
17         u3 = QHBoxLayout(self)
18         u4 = QHBoxLayout(self)
19
20         self.cypher = QComboBox(self)
21         self.cypher.addItem("Ардем")
22         self.cypher.addItem("Вхенеп")
23         self.cypher.addItem("Уесарп")
24         self.cypher.setFont(QFont("Times", 16))
25         u1.addWidget(self.cypher, 1)
26
27         u1.addLayout(u2, 2)
28         u1.addLayout(u3, 3)
29         u1.addLayout(u4, 4)
30
31         self.label = QLabel(self)
32         self.label.setText("Ключ: ")
33         self.label.setStyleSheet(
34             "background-color: {}".format('{}'.format('#c4d1ff')))
35         self.label.setFont(QFont("Times", 16))
36         u2.addWidget(self.label, 0)
```




## Создан алгоритм работы шифра Атбаш

```
1  class Atbash:
2      def encrypt(text):
3          alphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦШЩЪЫЬЭЮЯ'
4          alpha = 'абвгдеёжзийклмнопрстуфхцшщъыьэюя'
5          result = ""
6          for i in text:
7              if i.islower():
8                  result += Atbash.func(i)
9              else:
10                 result += Atbash.funcC(i)
11          return result
12
13     def func(text):
14         alphabet = 'абвгдеёжзийклмнопрстуфхцшщъыьэюя'
15         reversed_alphabet = alphabet[::-1]
16         atbash_dict = dict(zip(alphabet, reversed_alphabet))
17         result = ''.join(atbash_dict.get(char, char) for char in text)
18         return result
19
20     def funcC(text):
21         alphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦШЩЪЫЬЭЮЯ'
22         reversed_alphabet = alphabet[::-1]
23         atbash_dict = dict(zip(alphabet, reversed_alphabet))
24         result = ''.join(atbash_dict.get(char, char) for char in text)
25         return result
```



## Создан алгоритм работы шифра Виженера

```
1  class Vigenere:
2      def encrypt(message, key_users):
3          key_real = [ord(i) for i in key_users]
4          answer = ''
5          for i in range(len(message)):
6              shift = key_real[i % len(key_real)]
7              answer += chr((ord(message[i]) + shift) % 1200) # Russian lang max
8          return answer
9
10     def decrypt(message, key_users):
11         key_real = [ord(i) for i in key_users]
12         answer = ''
13         for i in range(len(message)):
14             shift = key_real[i % len(key_real)]
15             answer += chr((ord(message[i]) - shift) % 1200) # Russian lang max
16         return answer
```



Создан алгоритм работы шифра  
Цезаря

```
1  ✓ class Caesar:
2  ✓     def encrypt(message, key_users):
3         key_real = int(key_users) % 1200
4         answer = ''
5         for i in range(len(message)):
6             answer += chr((ord(message[i]) + key_real))
7         return answer
8
9  ✓     def decrypt(message, key_users):
10         key_real = int(key_users) % 1200
11         answer = ''
12         for i in range(len(message)):
13             answer += chr((ord(message[i]) - key_real))
14         return answer
```



# Интерфейс программы





# Недостатки

























