МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель	_____	С.Ю. Гуков
должность, уч.степень	подпись, дата	Инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 9

АСИММЕТРИЧНЫЙ АЛГОРИТМ ШИФРОВАНИЯ RSA

По курсу: Алгоритмы и структуры данных

РАБОТУ ВЫПОЛНИЛ

Студент гр. № 4329	27.12.2024	Д.М.Онопричук

Санкт-Петербург

2024

ЦЕЛЬ И ЗАДАНИЕ

Программно реализовать на любом языке программирования ассиметричный алгоритм шифрования RSA. Продемонстрировать на примерах его работу по зашифровыванию и расшифровыванию текста. Разрешается реализовать как версию с пользовательским интерфейсом, так и консольную версию с дружелюбным командно-текстовым интерфейсом.

КРАТКОЕ ОПИСАНИЕ ХОДА РАЗРАБОТКИ

Мой код реализует простое приложение для шифрования и расшифровки текста с использованием алгоритма RSA в графическом интерфейсе на базе библиотеки tkinter. Основные функции включают вычисление наибольшего общего делителя (НОД), проверку чисел на простоту, генерацию случайных простых чисел для формирования ключей, а также сами функции шифрования и расшифровки текста. Публичный и приватный ключи генерируются случайным образом, или могут быть введены пользователем вручную, с обязательной проверкой на совпадение модуля в обоих ключах.

Интерфейс включает текстовые поля для ввода исходного и зашифрованного текста, а также кнопки для выполнения шифрования и расшифровки. Пользователи могут задавать свои ключи, и также поддерживается копирование, вырезание и вставка через контекстное меню. Программа позволяет работать с текстами, шифруя и расшифровывая их с использованием алгоритма RSA, обеспечивая удобный и интуитивно понятный интерфейс для взаимодействия.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import random
import tkinter as tk
from tkinter import messagebox

# Функции RSA

#НОД
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

#проверка на праймовость
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

#генерация праймовых кандидатов
def generate_prime_candidate():
    while True:
        candidate = random.randint(100, 999)
        if is_prime(candidate):
            return candidate

#генерация ключей
def generate_keys():
    p = generate_prime_candidate()
    q = generate_prime_candidate()
    while q == p:
        q = generate_prime_candidate()

    n = p * q
    phi_n = (p - 1) * (q - 1)

    e = random.randint(2, phi_n - 1)
    while gcd(e, phi_n) != 1:
        e = random.randint(2, phi_n - 1)

    d = pow(e, -1, phi_n)

    return ((e, n), (d, n))  # публичный ключ(e, n), закрытый ключ(d, n)

def encrypt(public_key, plaintext):
    e, n = public_key
```

```
    return [pow(ord(char), e, n) for char in plaintext]

def decrypt(private_key, ciphertext):
    d, n = private_key
    return ''.join([chr(pow(char, d, n)) for char in ciphertext])

# поддержка копирования, вырезания и вставки через меню
def create_context_menu(widget):
    menu = tk.Menu(widget, tearoff=0)
    menu.add_command(label="Копировать", command=lambda:
widget.event_generate("<<Copy>>"))
    menu.add_command(label="Вырезать", command=lambda:
widget.event_generate("<<Cut>>"))
    menu.add_command(label="Вставить", command=lambda:
widget.event_generate("<<Paste>>"))

    def show_menu(event):
        menu.post(event.x_root, event.y_root)

    widget.bind("<Button-3>", show_menu)  #как оказалось, баттон 3 это пкм

# интерфейс

#генерация ключей
def generate_keys_ui():
    global public_key, private_key
    public_key, private_key = generate_keys()
    public_key_label.config(text=f"Открытый ключ: {public_key}")
    private_key_label.config(text=f"Закрытый ключ: {private_key}")
    messagebox.showinfo("Ключи сгенерированы", "Ключи успешно сгенерированы!")
    print(public_key,private_key)

#добавление своих ключей
def set_custom_keys():
    try:
        global public_key, private_key
        e, n = map(int, public_key_entry.get().split(','))
        d, n_private = map(int, private_key_entry.get().split(','))
        if n != n_private:
            messagebox.showerror("Ошибка", "Модуль n в открытом и закрытом ключах
должен совпадать!")
            return
        public_key = (e, n)
        private_key = (d, n)
        public_key_label.config(text=f"Открытый ключ: {public_key}")
        private_key_label.config(text=f"Закрытый ключ: {private_key}")
        messagebox.showinfo("Успешно", "Ключи добавлены!")
    except ValueError:
        messagebox.showerror("Ошибка", "Введите ключи в формате: число,число")
        print(public_key,private_key)
```

```
#шифровка текста
def encrypt_text():
    plaintext = input_text.get("1.0", "end-1c")
    if not public_key:
        messagebox.showerror("Ошибка", "Сначала сгенерируйте ключи или задайте
свои!")
        return
    ciphertext = encrypt(public_key, plaintext)
    encrypted_text.delete("1.0", "end")
    encrypted_text.insert("1.0", ' '.join(map(str, ciphertext)))
    print(' '.join(map(str, ciphertext)))

#расшифровка текста
def decrypt_text():
    ciphertext = encrypted_text.get("1.0", "end-1c")
    if not private_key:
        messagebox.showerror("Ошибка", "Сначала сгенерируйте ключи или задайте
свои!")
        return
    try:
        ciphertext_list = list(map(int, ciphertext.split()))
        plaintext = decrypt(private_key, ciphertext_list)
        decrypted_text.delete("1.0", "end")
        decrypted_text.insert("1.0", plaintext)
    except ValueError:
        messagebox.showerror("Ошибка", "Некорректный зашифрованный текст!")

# Создание окна
root = tk.Tk()
root.title("9 лаба")

# прописал интерфейс, на счет "дружелюбности" не уверен,
# ибо он тварь та еще, ему прописывать копирование и вставку отдельно надо,
оказывается, но шо поделать, живем же как-то
frame = tk.Frame(root)
frame.pack(pady=10, padx=10)

generate_button = tk.Button(frame, text="Сгенерировать ключи",
command=generate_keys_ui)
generate_button.grid(row=0, column=0, columnspan=2, pady=5)

public_key_label = tk.Label(frame, text="Открытый ключ: ---")
public_key_label.grid(row=1, column=0, columnspan=2, pady=5)

private_key_label = tk.Label(frame, text="Закрытый ключ: ---")
private_key_label.grid(row=2, column=0, columnspan=2, pady=5)

# ввод своих клбчей
custom_keys_label = tk.Label(frame, text="Задать свои ключи:")
custom_keys_label.grid(row=3, column=0, columnspan=2, pady=5)
```

```
public_key_entry = tk.Entry(frame, width=30)
public_key_entry.insert(0, "Введите открытый ключ (e,n)")
public_key_entry.grid(row=4, column=0, pady=5)

private_key_entry = tk.Entry(frame, width=30)
private_key_entry.insert(0, "Введите закрытый ключ (d,n)")
private_key_entry.grid(row=4, column=1, pady=5)

set_keys_button = tk.Button(frame, text="Добавить ключи",
command=set_custom_keys)
set_keys_button.grid(row=5, column=0, columnspan=2, pady=5)

# шифровка и расшифровка
input_text_label = tk.Label(frame, text="Текст для шифрования:")
input_text_label.grid(row=6, column=0, pady=5)

input_text = tk.Text(frame, height=3, width=40)
input_text.grid(row=6, column=1, pady=5)
create_context_menu(input_text)

encrypt_button = tk.Button(frame, text="Зашифровать", command=encrypt_text)
encrypt_button.grid(row=7, column=0, pady=5)

encrypted_text = tk.Text(frame, height=3, width=40)
encrypted_text.grid(row=7, column=1, pady=5)
create_context_menu(encrypted_text)

decrypt_button = tk.Button(frame, text="Расшифровать", command=decrypt_text)
decrypt_button.grid(row=8, column=0, pady=5)

decrypted_text = tk.Text(frame, height=3, width=40)
decrypted_text.grid(row=8, column=1, pady=5)
create_context_menu(decrypted_text)

# глобальные переменные
public_key = None
private_key = None

# запуск интерфейса
root.mainloop()
```

РЕЗУЛЬТАТЫ РАБОТЫ С ПРИМЕРАМИ

Ниже приведены результаты работы программы с различными наборами входных данных

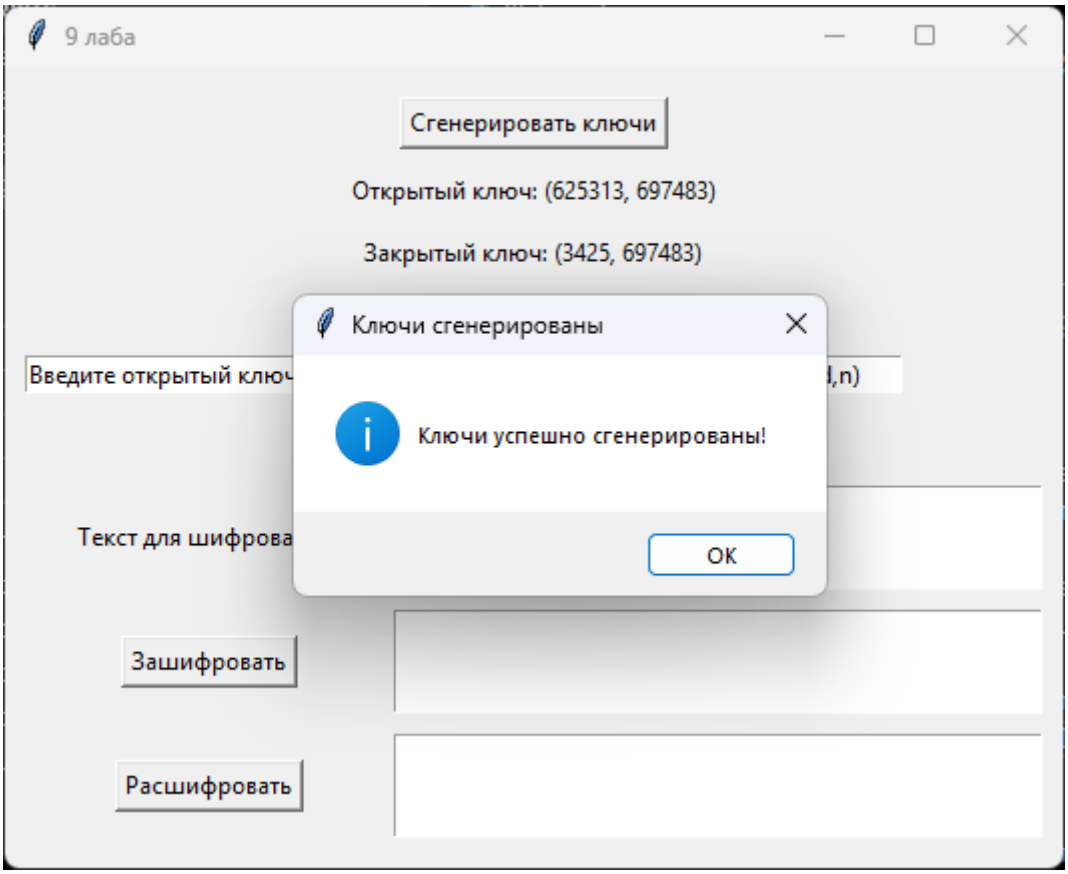


Рисунок 1 – Результаты работы программы

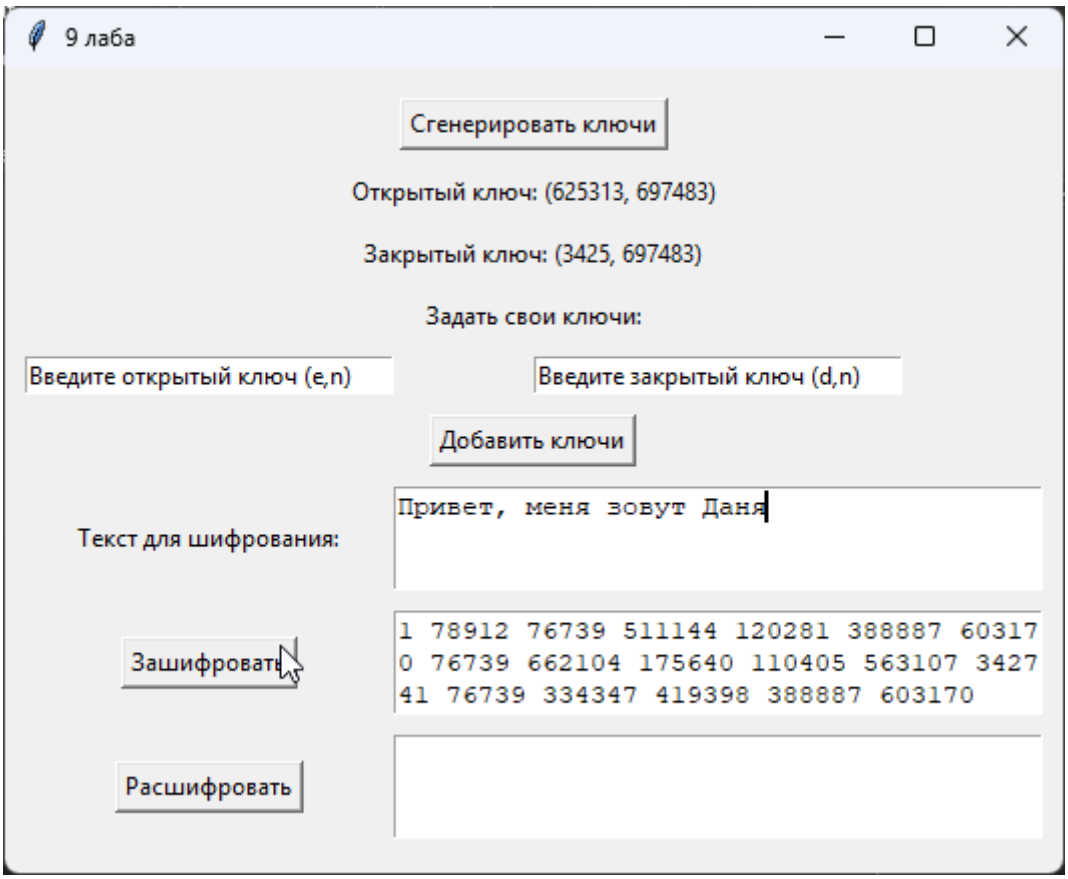


Рисунок 2 – Результаты работы программы

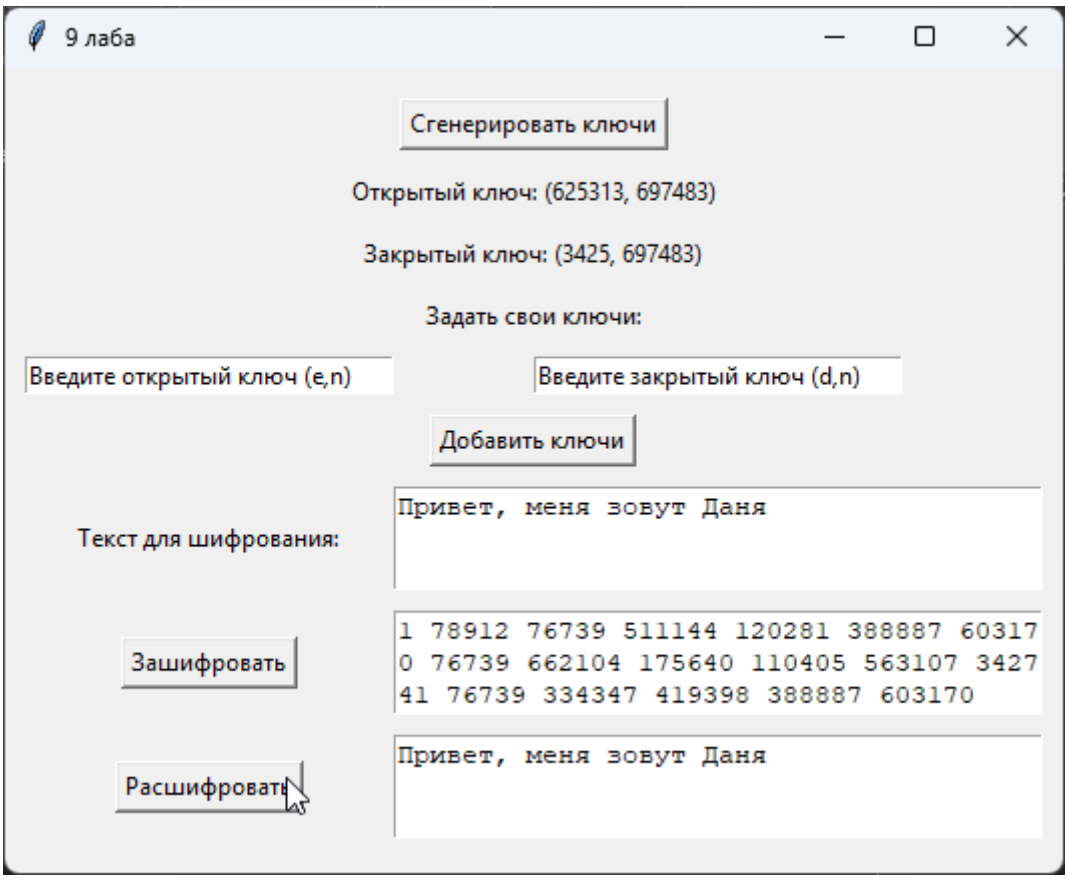
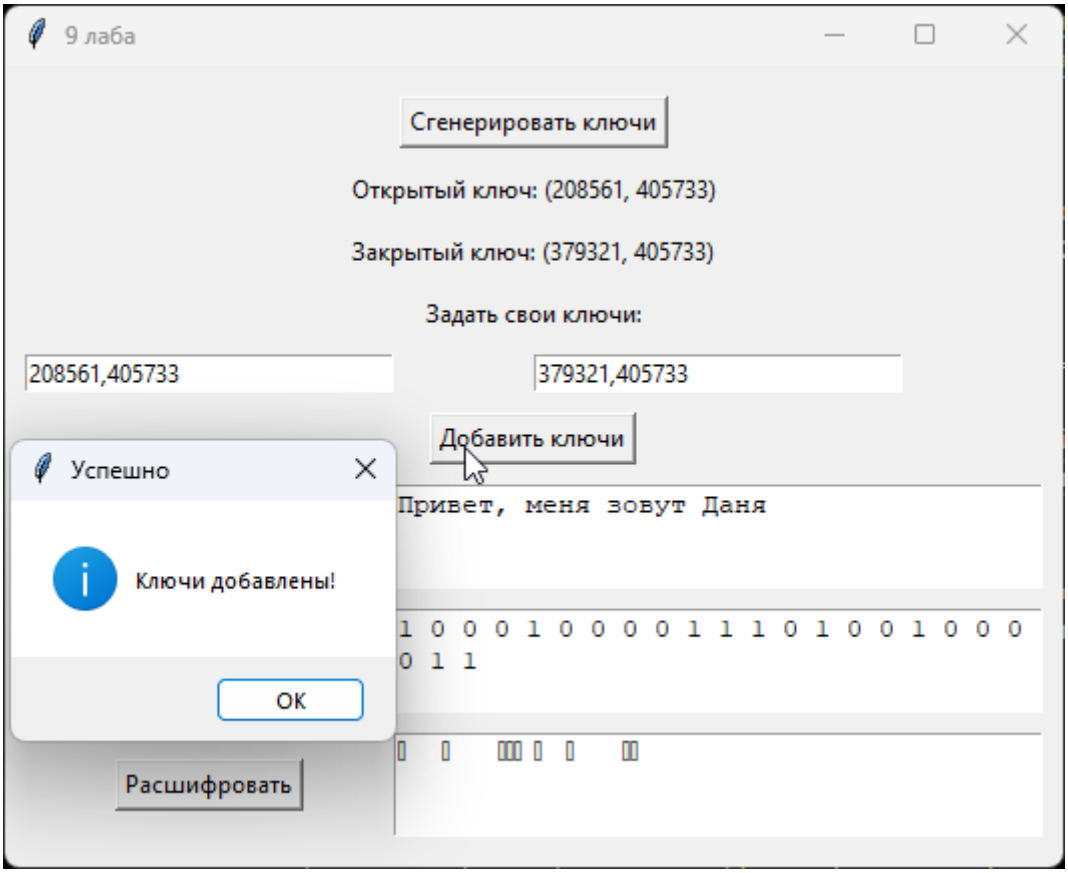


Рисунок 3 – Результаты работы программы



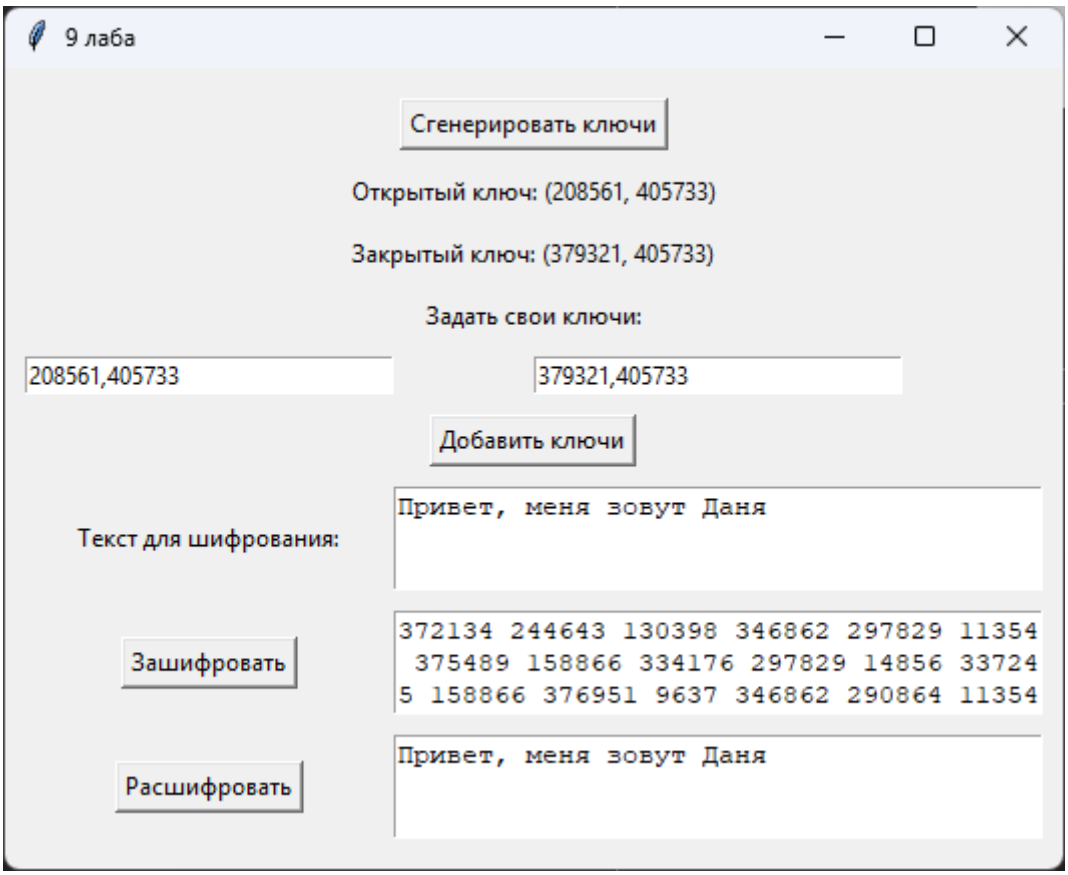


Рисунок 4 – Результаты работы программы

ВЫВОДЫ

В ходе выполнения лабораторной работы удалось реализовать программу, выполняющую шифрование и расшифровку текста с использованием алгоритма RSA. Программа генерирует публичные и приватные ключи, а также позволяет пользователю вводить собственные ключи для шифрования и расшифровки. В интерфейсе реализованы функции для копирования, вырезания и вставки текста, а также кнопки для выполнения операций шифрования и расшифровки. Все операции выполняются с использованием стандартных криптографических методов RSA, что позволяет эффективно обеспечивать безопасность данных.