Izrada programa za šifriranje i dešifriranje teksta uz pomoć Cezarove šifre

Dujlović, Robert

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2022./2023.

Robert Dujlović

Izrada programa za šifriranje i dešifriranje teksta uz pomoć Cezarove šifre

Završni rad

Mentor: doc. dr. sc. Ivan Dunđer

Zagreb, 2023.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Obitelji i mentoru.

Sadržaj

1. Uvod

Ljudi komuniciraju na različite načine već tisućama godina i komunikacija se smatra temeljem civilizacije. Jedan od danas najraširenijih načina komunikacije je komunikacija pismom, tj. porukom, bilo tiskanom ili elektroničkom. Takva poruka sadrži određene informacije koje je pošiljatelj poslao primatelju kako bi ga informirao o nečemu što je on smatrao važnim. Te informacije mogu biti „obične“ informacije koje nisu od velikoga značaja, ali također mogu biti i vrijedne informacije kojima se mora onemogućiti neovlašten pristup. U tom slučaju informacije je potrebno odgovarajuće zaštititi. Jedan od načina zaštite takvih poruka je kriptiranje, tj. šifriranje tih poruka. Time se postiže, ili se barem želi postići, da samo pošiljatelj i primatelj poruke imaju pristup informacijama koje poruka sadržava. Postoje razni, više ili manje sigurni algoritmi za šifriranje teksta kojima se pošiljatelj može poslužiti kako bi šifrirao i time zaštitio svoju poruku.

Glavni cilj ovoga završnog rada bit će razvoj programa koji će imati mogućnost šifriranja i dešifriranja teksta koji korisnik unese u program. Program će biti razvijen u programskom jeziku Python, koji je obrađivan na više kolegija tokom studija.

U radu će biti opisana kriptografija i kratka povijest kriptografije i algoritam koji će se koristiti za šifriranje teksta u sklopu programa. Algoritam koji će se koristiti je Cezarova šifra. Zatim će se ukratko opisati programski jezik Python i modul Tkinter koji je korišten za izradu grafičkog sučelja programa.

Zatim će se opisati program, njegov izgled i funkcionalnosti. Detaljno će se objasniti glavni dio programa i funkcije koje se pozivaju unutar glavnog programa. Na kraju će se predstaviti i određeni nedostaci koje program ima.

2. Kriptografija i kratka povijest kriptografije

„Kriptografija je znanstvena disciplina koja se bavi proučavanjem metoda za slanje poruka u takvom obliku da ih samo onaj kome su namijenjene može pročitati. Sama riječ kriptografija je grčkog podrijetla i mogla bi se doslovno prevesti kao tajnopis“ (Dujella i Maretić, 2007). Kriptografija je, uz kriptoanalizu, grana kriptologije, znanosti koja se bavi izučavanjem metoda za zaštitu informacija i izučavanjem metoda za otkrivanje šifriranih informacija (Wikipedia: Kriptologija, n.d.).

Ukratko, kriptografija se bavi šifriranom komunikacijom. Osnovni zadatak kriptografije je omogućiti pošiljatelju i primatelju komuniciranje preko komunikacijskog kanala na način da treća osoba, koja može nadzirati komunikacijski kanal, ne može razumjeti poruke koje razmjenjuju (Dujella i Maretić, 2007). Tokom povijesti se široko koristila za vojne i diplomatske svrhe (Hrvatska enciklopedija: kriptografija, 2021).

2.1. Osnovni pojmovi

Originalna poruka koju pošiljatelj šalje primatelju se u kriptografiji naziva otvoreni tekst (eng. *plaintext*). Koristeći unaprijed dogovoreni ključ (eng. *key*), pošiljatelj transformira otvoreni tekst i taj se postupak zove šifriranje. Rezultat šifriranja je šifrat (eng. *ciphertext*), koji se šalje preko komunikacijskog kanala i dolazi do primatelja koji koristeći ključ kojim je poruka šifrirana može dešifrirati šifrat i doći do otvorenog teksta (Dujella i Maretić, 2007).

Kriptografski algoritam ili šifra je matematička funkcija koja se koristi za šifriranje. Radi se o dvije funkcije: funkciji za šifriranje i funkciji za dešifriranje. Te funkcije preslikavaju osnovne elemente otvorenog teksta u osnovne elemente šifrata i obratno. Te se funkcije biraju iz određenog skupa funkcija u ovisnosti o ključu. Skup svih mogućih vrijednosti ključeva se naziva prostor ključeva. Kriptosustav se sastoji od kriptografskog algoritma i svih mogućih otvorenih tekstova, šifrata i ključeva (Dujella i Maretić, 2007).

Kriptosustavi se dijele s obzirom na tri kriterija:

- Tip operacija koje se koriste pri šifriranju – postoje supstitucijske šifre, gdje se svaki znak zamjenjuje nekim drugim znakom, i transpozicijske šifre u kojima se poredak znakova unutar teksta premješta.
- Način na koji se obrađuje otvoreni tekst – postoje blokovne šifre, kod kojih se obrađuje jedan po jedan blok teksta koristeći isti ključ K, i protočne šifre (eng. *stream*

cipher) kod kojih se elementi otvorenog teksta obrađuju jedan po jedan koristeći paralelno generirani niz ključeva (eng. *key-stream*).

- Tajnost i javnost ključeva – razlikuju se simetrični kriptosustavi i asimetrični kriptosustavi. Kod simetričnih kriptosustava ključ za dešifriranje se može izračunati pomoću ključa za šifriranje. Kod asimetričnih kriptosustava ključ za dešifriranje se ne može u nekom razumnom vremenu izračunati iz ključa za šifriranje (Dujella i Maretić, 2007).

2.2. Povijest kriptografije

Ljudi su vjerojatno pokušali sakriti informacije u pisanom obliku od vremena kada se razvilo pisanje. Prvi poznati primjer kriptografije je pronađen u natpisu uklesanom oko 1900. godine pr. Kr. u grobnici plemića Khnumhotepa 2. u Egiptu. Natpis je, pored uobičajenih hijeroglifa, sadržavao i neobične simbole koji su izgledali kao hijeroglifi (Sidhpurwala, 2023).

Stari Grci su također koristili kriptografiju. Poznato je da su Spartanci u 5. stoljeću pr. Kr. koristili napravu za šifriranje skital. Naprava se sastojala od štapa, oko kojeg bi se omotala vrpca od pergamene. Na pergamenu bi se okomito upisivala poruka, te bi se nakon upisa vrpca odmotala te bi poredak znakova bio promijenjen. Poruka bi bila čitljiva onima koji su imali štap iste debljine (Dujella i Maretić, 2007; Simmons, n.d.).

Tokom povijesti korišteni su različiti načini šifriranja teksta. Jedan od najpoznatijih načina je Cezarova šifra, koja će biti podrobnije objašnjena u sljedećem poglavlju. Vigenèreova šifra je također jedan od poznatijih načina, a nastala je u 16. stoljeću. Ona spada u polialfabetske šifre, tj. kod nje se svako slovo otvorenog teksta može preslikati u jedno od m mogućih slova (gdje je m duljina ključa), u ovisnosti o svom položaju unutar otvorenog teksta (Dujella i Maretić, 2007). Smatra se prvom šifrom koja je koristila ključ za šifriranje (Sidhpurwala, 2023). Još neki od načina su bili Playfairova šifra, Hillova šifra, stupičasta transformacija i slični (Dujella i Maretić, 2007).

Pojava naprednih naprava za šifriranje je znatno olakšala šifriranje i šifre učinila kompliciranijima i sigurnijima. Jedan od prvih primjera takvih naprava je Jeffersonov kotač za šifriranje. Neki od električnih uređaja za šifriranje koji su se pojavili tokom 20. stoljeća su Hebernov električni stroj za šifriranje, Hagelinov stroj M-209 i drugi. Najpoznatiji primjer takvog uređaja je ENIGMA, razvijena od strane Arthura Scherbiusa 1918. godine, a čiji je naziv postao svojevrstan sinonim za tajnu (Dujella i Maretić, 2007).

2.3. Moderna kriptografija

Do kraja Drugog svjetskog rata kriptografija se najviše koristila u vojne i diplomatske svrhe, ali se u to vrijeme također pojavila želja za komercijalizacijom kriptografije i korištenje u druge, civilne svrhe. Pojavom računala ljudi su počeli veću pozornost obraćati na sigurnost svojih podataka, te su od IBM-a (eng. *International Business Machines*), tada vodeće računalne tvrtke, zahtijevali neku vrstu enkripcije podataka (Sidhpurwala, 2023). Stvorili su Lucifer šifru, koju je NIST (eng. *National Institute of Standards and Technology*), s određenim izmjenama algoritma, prihvatio 1976. godine pod nazivom DES (eng. *Data Encryption Standard*). DES se kao nacionalni standard u Sjedinjenim Američkim Državama koristio sve do ranih 2000-ih godina. 2001. godine je, zbog napretka računalne tehnologije i mogućnosti uspješnog *brute force* napada u razumnom vremenu, a koji je bio smatran nemogućim kada je DES nastao, zamijenjen s algoritmom AES (eng. *Advanced Encryption Standard*) (Simmons, n.d.).

AES je bio razvijen od strane belgijskih kriptografa Joana Daemena i Vincenta Rijmena i originalni naziv je bio Rijndael blokovna šifra. Glavne razlike u odnosu na DES su bile u veličini ključa, koji je bio 128, 192 ili 256 bita, za razliku od DES-ovog 56 bitnog ključa, i veličini blokova koja je kod AES bila 128 bita, dvostruko veća nego kod DES-a (Wikipedia: Advanced Encryption Standard, n.d.).

Sve se veća pozornost obraća i na budućnost kriptografije u eri kvantnih računala. NIST je u tu svrhu 2016. godine objavio poziv u kojem je pozvao svjetske kriptografe na razvoj algoritama za šifriranje koji bi mogli izdržati napade budućih kvantnih računala. 2022. godine objavili su popis odabranih algoritama. Za algoritam enkripcije s javnim ključem odabran je algoritam CRYSTALS-Kyber, dok su za algoritme za digitalne potpise odabrani algoritmi CRYSTALS-Dilithium, FALCON i SPHINCS+ (National Institute of Standards and Technology, 2022).

3. Cezarova šifra

Jedan od najjednostavnijih načina šifriranja teksta je Cezarova šifra, nazvana prema rimskom vojskovođi i državniku Juliju Cezaru. Svetonije je u svome tekstu Dvanaest Cezara zapisao kako je Cezar koristio šifru koja mijenja svako slovo u slovo tri mjesta desno u abecedi (Wikipedia: Cezarova šifra, n.d.), varijanta koja je korištena i u izradi programa za potrebe ovog rada. Danas se svaka šifra s nekim fiksnim pomakom znaka naziva Cezarova šifra (Moore et al., n.d.).

U Cezarovoj šifri osnovni elementi otvorenog teksta su slova (tj. njihovi numerički ekvivalenti), a ključ *K* određuje za koliko mjesta udesno se pomiču slova prilikom šifriranja. Za K=3 dobiva se originalna Cezarova šifra, a postoje i neke naznake da je Cezarov nećak August koristio najjednostavniju verziju ove šifre pomičući slova samo jedno mjesto udesno (Dujella i Maretić, 2007). Razlika je bila u tome što je Cezar vršio rotiranje na početak abecede ako je to bilo potrebno, dok August u svojoj verziji nije vršio rotiranje (Wikipedia: Cezarova šifra, n.d.). Cezarova šifra spada u monoalfabetske šifre, tj. supstitucijske šifre kod kojih svakom slovu otvorenog teksta odgovara jedinstveno slovo šifrata (Dujella i Maretić, 2007).

Korištenje Cezarove šifre je jednostavno i sastoji se od nekoliko koraka:

- odabiranje vrijednosti pomaka,
- raspisivanje jasnopisnog slovoreda i kritopisnog slovoreda, kritopisni slovored se piše ovisno o vrijednosti pomaka koja je izabrana,
- zamjena svakog slova otvorenog teksta s ekvivalentnim slovom kritopisnog slovoreda,
- slanje šifrata primatelju, obratiti pozornost da primatelj mora znati algoritam šifriranja i vrijednost pomaka kako bi došao do otvorenog teksta (Moore et al., n.d.).

3.1. Razbijanje šifre

Cezarova šifra je jednostavna za korištenje, ali isto tako je i jednostavna za kriptoanalizu. U slučaju engleske abecede koja ima 26 slova, postoji 26 različitih kombinacija i kriptoanalitičar treba jednostavno proći kroz svaki mogući pomak i vidjeti je li se pojavila smislena poruka (Moore et al., n.d.). Prema tome, Cezarova šifra danas ne nudi nikakvu kriptografsku zaštitu i većinom se ne koristi za šifriranje poruka, ali se koristi kao korak u

složenijim načinima šifriranja kao npr. Vigenèreovoj šifri ili ROT13 sustavu (Wikipedia: Cezarova šifra, n.d.).

Za razbijanje Cezarove šifre je dovoljan samo šifrat. Tu se mogu razmotriti dva slučaja:

- napadač zna, ili sumnja, da se radi o šifri zamjene, ali ne i da je korištena Cezarova šifra,
- napadač zna da je korištena Cezarova šifra, ali ne zna vrijednost pomaka (Wikipedia: Cezarova šifra, n.d.).

U prvom slučaju, napadač se može poslužiti analizom frekvencije slova. Broji se pojavljivanje svakog slova u šifratu i distribucija slova u šifratu se uspoređuje s poznatim podacima o distribuciji slova u jeziku na kojem je napisan otvoreni tekst. Vrlo je vjerojatno da najfrekventnija slova šifrata odgovaraju najfrekventnijim slovima jezika. Ta je vjerojatnost veća što je šifrat dulji. Napadaču od koristi mogu biti i podaci o najčešćim dvoslovima i troslovima u jeziku (Dujella i Maretić, 2007).

U slučaju da napadač zna da je korištena Cezarova šifra razbijanje je još jednostavnije. Pošto postoji ograničen broj mogućih pomaka, svaki se od njih može ispitati pomoću *brute force* napada. Jedan od načina napada uključuje uzimanje dijela šifrata i stavljanje u tablicu koja sadrži sve moguće pomake. Drugi način je da se ispod svakog slova šifrata ispiše cijela abeceda unatrag, počevši od tog slova. Također se može koristiti i frekvencijska analiza (Wikipedia: Cezarova šifra, n.d.).

3.2. Cezarova šifra u programu

Za potrebe ovog rada korištena je Cezarova šifra s pomakom tri mjesta udesno. Za pomak se koristila ASCII tablica znakova. Svaki znak u unesenom tekstu se pretvarao u brojčanu vrijednost koju ima u ASCII tablici (uz pomoć Python funkcije ord()), te se toj vrijednosti zbrajao broj 3, kako bi se ostvario pomak. Na kraju se nova vrijednost uz pomoć funkcije chr() pretvarala u znak koji predstavlja u ASCII tablici. Za razliku od originalne Cezarove šifre koja vrši rotiranje kroz abecedu, u ovoj verziji šifre ne postoji rotiranje, što znači da će se neka slova šifrirati ili dešifrirati u znakove koji se nalaze ispred, odnosno iza slova u ASCII tablici.

4. O Pythonu

Za izradu programa za potrebe ovog rada je korišten programski jezik Python.

Python je interpretirani, interaktivni, objektu orijentirani programski jezik (Python Software Foundation: General Python FAQ, n.d.). Razvijen je od strane Guida van Rossuma krajem 1980-ih kao nasljednik ABC programskog jezika, a implementacija je započela u prosincu 1989. godine. Na tržištu se pojavio 20. veljače 1991. godine kao Python 0.9.0. Jedan je od najpopularnijih programskih jezika i može se primijeniti za razne svrhe (Wikipedia: Python (programming language), n.d.).

Python kao programski jezik programerima omogućava korištenje više stilova programiranja. Objektu orijentirano programiranje i strukturno programiranje su u potpunosti podržani, a njihove značajke također omogućuju funkcionalno programiranje i aspektno orijentirano programiranje. Drugi stilovi programiranja su podržani pomoću ekstenzija koje se mogu dodati Pythonu (Wikipedia: Python (programming language), n.d.).

Temeljna filozofija Python programskog jezika je sažeta u dokumentu The Zen of Python, autora Tima Petersa, koji je principe za programiranje u Pythonu sažeo u 20 aforizama, od kojih je 19 napisano. Neki od aforizama su:

- lijepo je bolje od ružnog,
- eksplicitno je bolje od implicitnog,
- jednostavno je bolje od kompleksnog,
- kompleksno je bolje od kompliciranog,
- čitljivost se računa,
- greške nikad ne bi trebale proći neprimijećene (osim ako su eksplicitno „utišane“) i slično (Peters, 2004).

4.1. Sintaksa i semantika

Sintaksa programskog jezika obuhvaća pravila za pisanje naredbi u programskom jeziku. Naredba je sintaktički pravilna ukoliko slijedi sva pravila. Semantika programskog jezika obuhvaća značenje koje naredba ima u programskom jeziku (GeeksforGeeks, n.d.).

Python programski jezik je zamišljen da bude lagano čitljiv jezik. Oblikovanje je vizualno čisto i za razliku od drugih programskih jezika često koristi engleske ključne riječi, ne koristi vitičaste zagrade za omeđivanje blokova koda, a znak točka-zarez na kraju naredbe je dopušten, ali rijetko korišten (Wikipedia: Python (programming language), n.d.).

Za omeđivanje blokova koda se koriste uvlake. Uvučeni dio koda označava jedan blok koda, tj. označava da taj kod pripada npr. funkciji ili if naredbi. Prema tome vizualna struktura koda predstavlja ujedno i semantičku strukturu koda. Neki drugi programski jezici koriste uvlake na isti način, ali kod većine uvlaka ne nosi nikakvo semantičko značenje. Python nudi različite načine kontrole toka (pomoću if-else, for, while, try-except i sličnih naredbi), različite operatore (npr. aritmetičke operatore, Booleove operatore i sl.), liste, rječnike, n-torke, metode (funkcije koje su vezane uz klasu objekta) i slično (Wikipedia: Python (programming language), n.d.).

Tipovi podataka koje Python prepoznaje:

- brojevi – npr. int, float, complex,
- nizovi – dijele se na nepromjenjive (stringovi, n-torke) i promjenjive (liste),
- skupovi – set, frozenset,
- rječnici – dict (Wikipedia: Python (programming language), n.d.).

Jedna od velikih prednosti Pythona je Python biblioteka (eng. *Python library*). Python biblioteka sadrži različite module koje korisnik može implementirati kako bi izvršio neki zadatak. Moduli sadrže funkcije prikladne za npr. analizu podataka, manipuliranje bazama podataka, izrade grafičkih sučelja i slično (Wikipedia: Python (programming language), n.d.). Za potrebe ovog rada korišten je modul Tkinter, koji omogućava izradu grafičkih sučelja, a koji će biti podrobnije objašnjen u idućem poglavlju.

4.2. Razvojna sučelja

Python programski jezik sadrži više razvojnih sučelja. Najkorištenija su read-eval-print loop (skraćeno REPL) i integrirano razvojno okruženje (eng. *Integrated Development Environment*, skraćeno IDE), kojih za Python postoji više od deset (Wikipedia: Python (programming language), n.d.). Za potrebe ovog rada korišteno je integrirano okruženje za razvoj i učenje (eng. *Integrated Development and Learning Environment*, skraćeno IDLE), koje je razvio Guido van Rossum, tvorac Pythona.

Read-eval-print loop (REPL) je računalno okruženje koje čita korisnički unos, izvršava ga i vraća rezultat korisniku (Horcasitas, 2021). Ime read-eval-print loop dolazi od Lisp funkcija koje su funkcionirale na taj način (Wikipedia: Read-eval-print loop, n.d.). Svaki kod koji se napiše u Pythonu (i onaj u IDLE sučelju), se izvršava uz pomoć REPL-a i prema tome predstavlja temelj programiranja u Pythonu (Lott, 2015).

Integrirano okruženje za razvoj i učenje (IDLE) je integrirano razvojno okruženje za Python koje postoji od inačice 1.5.2b1. Razvijeno je u Pythonu i Tkinteru. Zamišljeno je kao jednostavno IDE sučelje pogodno za početnike (Wikipedia: IDLE, n.d.). Neke od značajki su isticanje sintakse, automatsko dovršavanje naredbi i pametne uvlake (eng. *smart indents*). Također sadrži program za ispravljanje pogrešaka (eng. *debugger*) (TutorialsTeacher, n.d.).

5. O Tkinteru

Za izradu grafičkog sučelja programa korišten je modul Tkinter.

Tkinter paket je standardna veza Pythona i Tcl/Tk GUI alata (Python Software Foundation: tkinter - Python interface to Tcl/Tk, n.d.). Služi za izradu grafičkih sučelja u Python programskom jeziku. Nije jedini takav alat, ali je daleko najkorišteniji (Python Software Foundation: Tkinter, n.d.).

Tcl/Tk je besplatan, open-source alat koji korisnicima nudi mogućnost izrade grafičkih sučelja u raznim programskim jezicima. Alat je razvijen od strane Johna Ouserhouta kao ekstenzija Tcl skriptnog jezika i objavljen je 1991. godine. Karakteriziraju ga neovisnost o platformi (mogućnost povezivanja i korištenja unutar raznih programskih jezika, a pomoću Tkintera i unutar Pythona), prilagodljivost (svaki *widget* se može urediti) i podesivost (Wikipedia: Tk (software), n.d.).

Tkinter je nastao od strane Steena Lumholta i Guida van Rossuma, stvoritelja Pythona, a kasnije je revidiran od strane Fredrika Lundha. Softver je besplatan i objavljen pod Python licencom (Wikipedia: Tkinter, n.d.).

Kao i druge Tk veze, Tkinter se u Python implementira kao omot (eng. *wrapper*) oko Tcl interpretera i taj Tcl interpreter se ugrađuje u Python interpreter. Radi na način da se Tkinter pozivi (eng. *calls*) prevode u Tcl naredbe koje se izvršavaju uz pomoć Tcl interpretera. Pomoću toga je moguće koristiti i Python i Tcl u istoj aplikaciji (Wikipedia: Tkinter, n.d.).

Ključne karakteristike su:

- jednostavnost,
- potrebno jako malo koda za izradu funkcionalne aplikacije,
- slojevit dizajn,
- neovisnost o operacijskom sustavu,
- dostupan unutar Python biblioteke (Wikipedia: Tkinter, n.d.).

5.1. Osnovni pojmovi

Neki od pojmova su prozor (i prozor najviše razine ili *top-level window*), widget, okvir, odnos roditelj-dijete.

Prozor je najjednostavnije definirati kao prostor na radnoj površini unutar kojeg se nalazi program. Prozor najviše razine je prozor unutar kojeg se nalaze widgeti, okviri i slične stavke koje je korisnik kreirao (Wikipedia: Tkinter, n.d.).

Widgeti su generički naziv za sve stavke koje se koriste prilikom izrade grafičkog sučelja. Obuhvaća stavke poput polja za unos teksta, različitih tipova gumbova, okvira i slično. Također obuhvaća i pop-up dijaloške okvire koji se mogu definirati uz pomoć Tkintera (Wikipedia: Tkinter, n.d.).

U Tkinteru, okviri najčešće služe za organizaciju widgeta unutar aplikacije. Time se uspostavlja odnos roditelj-dijete između okvira i widgeta. Na primjer, kada se unutar okvira kreira polje za unos teksta, okvir je roditelj tom polju (Wikipedia: Tkinter, n.d.).

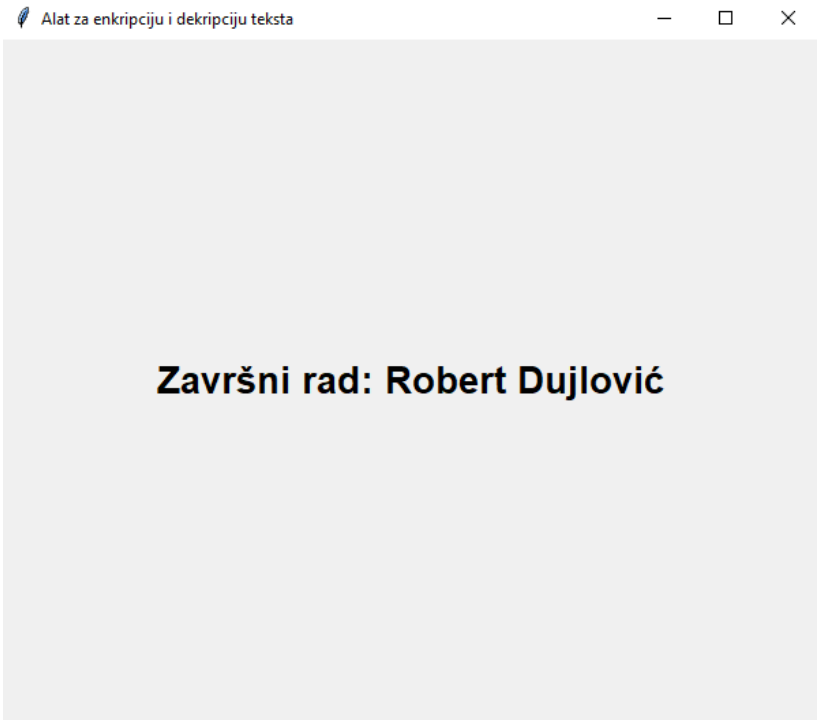
6. Program

Program se sastoji od dvije datoteke: glavnog programa pod nazivom program.py i datoteke s funkcijama pod nazivom funkcije.py.

6.1. Opis programa

Program se može pokrenuti isključivo pomoću glavnog programa. Nije moguće korištenje programa pomoću datoteke s funkcijama. Ukoliko korisnik pokrene datoteku s funkcijama, na ekranu će se ispisati tekst „Pokrenite glavni program: program.py“.

Nakon pokretanja glavnog programa, prvi prozor koji se pojavi je naslovna stranica koja sadrži tekst „Završni rad – Robert Dujlović“ (slika 1). Nakon dvije sekunde zatvara se naslovna stranica i otvara prozor unutar kojeg se nalazi tekst koji korisniku daje uvid u osnovne informacije vezane uz program: autor, mentor, akademska godina nastanka i slično. Prozor se sam zatvara nakon 60 sekundi ili ga korisnik može zatvoriti pritiskom na gumb koji se nalazi na prozoru (slika 2). Nakon toga se otvara prozor s glavnim dijelom programa.



Slika 1. Naslovna stranica

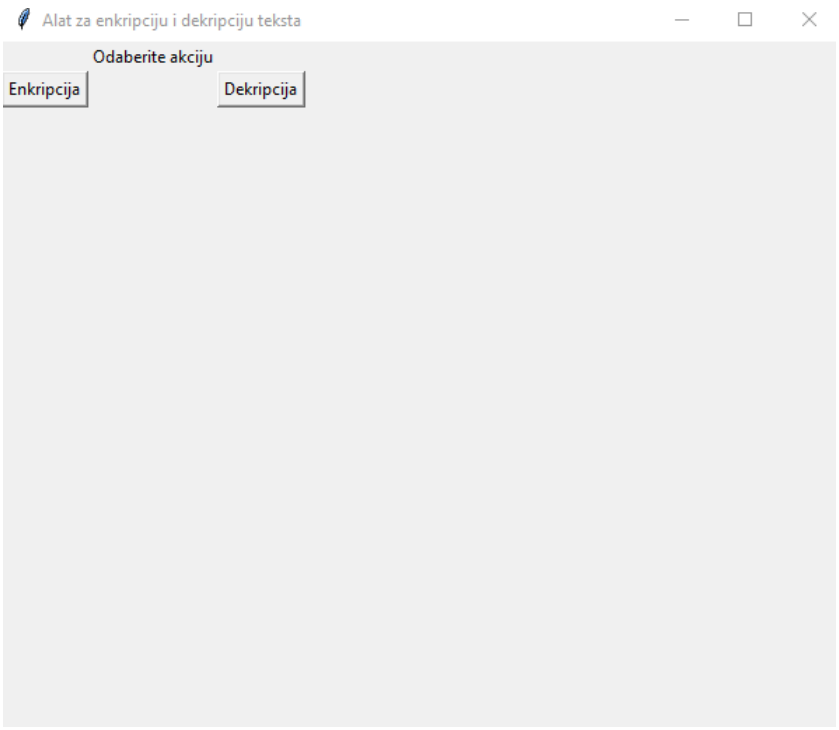
Na početku programa se nalaze dva gumba, „Enkripcija“ i „Dekripcija“ (slika 3), pomoću kojih korisnik odabire želi li enkriptirati ili dekriptirati tekst koji unese u program. Pritiskom na bilo koji gumb pojavljuje se dodatni sadržaj na prozoru: polje za unos teksta i gumb „Enkriptiraj tekst“ ili „Dekriptiraj tekst“ kojim se izvršava akcija koju je korisnik odabrao.

Također se onemogućava odabiranje druge akcije, npr. ako je korisnik odabrao da želi enkriptirati tekst, gumb „Dekriptiraj“ se isključuje, tj. nije ga moguće više pritisnuti. Pomoću polja za unos korisnik može unijeti tekst koji želi enkriptirati ili dekriptirati. Nakon što korisnik unese tekst u polje i pritisne gumb, korisniku se ispisuje enkriptirani ili dekriptirani tekst. Ispod ispisa teksta pojavi se sadržaj koji korisniku nudi mogućnost upisa enkriptiranog ili dekriptiranog teksta u datoteku.

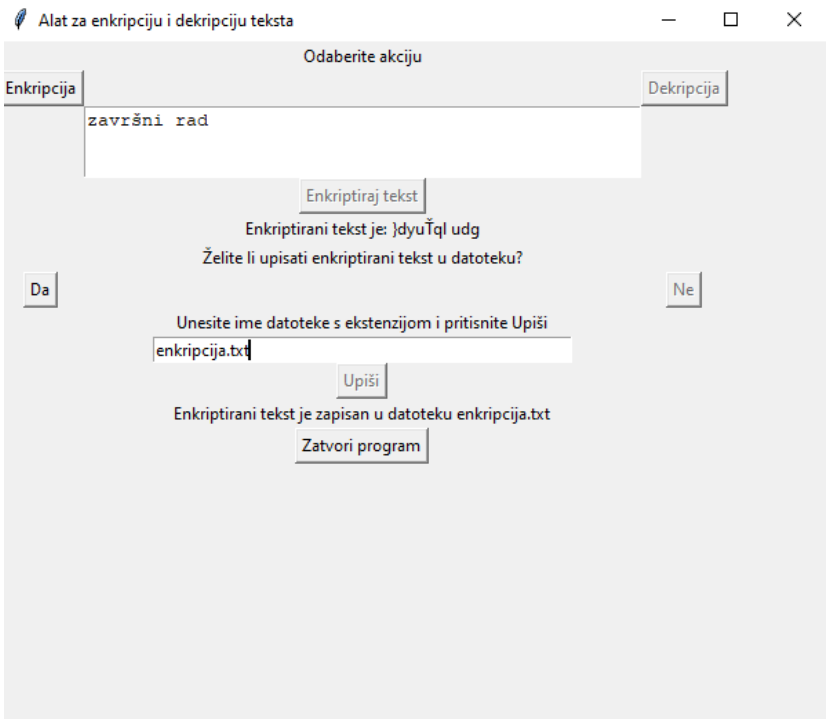
Slika 2. Prozor s informacijama o programu

Ukoliko korisnik odabere da želi upisati tekst u datoteku (pritisne gumb „Da“), stvara se novo polje za unos teksta u koje korisnik mora upisati ime datoteke s ekstenzijom u koju želi upisati tekst. Nakon što unese ime datoteke i pritisne gumb „Upiši“ (slika 4), tekst se upisuje u zadanu datoteku (moguće je korištenje iste datoteke, jer postojeći sadržaj ostaje očuvan i novi sadržaj se dodaje iza postojećeg). Na kraju se pojavljuje tekst koji korisnika obavještava da je tekst upisan u datoteku i gumb kojim se zatvara program.

Ukoliko korisnik odabere da ne želi upisati tekst u datoteku (pritisne gumb „Ne“), pojavljuje se tekst koji korisniku zahvaljuje na korištenju programa i gumb pomoću kojeg se program zatvara.



Slika 3. Početni izgled programa



Slika 4. Izgled programa nakon korištenja

7. Glavni program

Glavni program je datoteka pomoću koje se pokreće program i koja poziva sve potrebne funkcije za izvršenje nekog zadatka. Naziv glavnog programa koji je izrađen za potrebe završnog rada je „program.py“.

Kod:

```
from tkinter import *

prozor = Tk()

prozor.title("Alat za enkripciju i dekripciju teksta")

prozor.geometry("600x500")

platno = Canvas(prozor, width=600, height=500, bg="#F0F0F0")

platno.create_text(300, 250, text="Završni rad: Robert Dujlović",

                   font=("Helvetica 20 bold"))

platno.pack()

platno.after(2000, lambda : platno.destroy())

Tekst = Label(prozor, text="Pozdrav i hvala na korištenju programa. Ovaj program nastao je kao završni rad pod nazivom:\n"\

              "Izrada programa za šifriranje i dešifriranje teksta uz pomoć Cezarove šifre.\n"\

              "Autor završnog rada je Robert Dujlović, a rad je napisan pod mentorstvom doc. dr. sc. Ivana Dunđera\n"\

              "akademske godine 2022./2023.\n\n"\

              "Napomena: Ovaj prozor će se sam zatvoriti nakon 60 sekundi ili nakon pritiska na gumb 'Zatvori prozor'\n"\

              "nakon čega će se otvoriti prozor u kojem se nalazi program.", font=("Helvetica 10")).pack()

GumbZ = Button(

    prozor,

    text="Zatvori prozor",

    state=NORMAL,
```

```
    command= lambda: prozor.destroy())
GumbZ.pack()
prozor.after(60000, lambda : prozor.destroy())
prozor.mainloop()
from funkcije import *
```

Kao i što je vidljivo iz koda, prvo se poziva Tkinter modul koji omogućava izradu grafičkog sučelja. Nakon toga se definira prozor unutar kojeg će se prikazivati naslovna stranica i prozor s podacima o autoru i dodjeljuje mu se naziv „Alat za enkripciju i dekripciju teksta“. Sljedećom naredbom se definira veličina prozora, koja je postavljena na 600x500 piksela. Zatim se kreira platno iste veličine unutar kojeg se prikazuje naslovna stranica. Na tom platnu se prikazuje tekst „Završni rad: Robert Dujlović“. Pomoću metode .after() za platno je definirano da se zatvori nakon 2 sekunde. Zatvaranje se vrši uz pomoć lambda funkcije lambda : platno.destroy(). Nakon naslovne stranice prikazuje se prozor u kojem se prikazuju opće informacije o programu, autoru, mentoru i slično. Taj prozor se automatski pomoću iste metode .after() zatvara nakon 60 sekundi ili pritiskom na gumb koji je u kodu nazvan GumbZ. Naredba prozor.mainloop() je obavezna naredba ukoliko se koristi Tkinter za izradu grafičkog sučelja u Pythonu. .mainloop() metoda služi kako bi Python pokrenuo petlju događaja (eng. *event loop*). Metoda osluškuje događaje, kao npr. pritiskanje gumba. Ukoliko se .mainloop() metoda ne napiše na kraj programa u Python datoteci, Tkinter aplikacija se neće izvesti. Nakon toga se iz modula pod nazivom „funkcije.py“ importiraju funkcije koje obavljaju glavni zadatak.

Za pozicioniranje elemenata unutar prozora naslovne stranice korištena je metoda .pack(), jedna od tri metode koje se mogu koristiti za pozicioniranje elemenata (druge dvije su .grid() i .place()). Metoda postavlja widgete u odnosu na prethodne widgete (GeeksforGeeks, n.d.). Pošto su u glavnom dijelu programa definirani widgeti Tekst i GumbZ, metoda .pack() će ih staviti jedan ispod drugoga po redu po kojem su definirani unutar programa.

8. Funkcije

Unutar datoteke s funkcijama nalaze se funkcije koje obavljaju enkripciju ili dekripciju teksta. Datoteka s funkcijama izrađena za ovaj rad se naziva „funkcije.py“. Funkcije su ugniježđene, tj. unutar jedne funkcije se nalaze druge funkcije. Razlog za to je da se korisniku pojave novi dijelovi programa kada obavi neku akciju (npr. kada korisnik enkriptira tekst, pojavi se novi dio programa koji nudi mogućnost upisa u datoteku). Postoje dvije glavne funkcije unutar kojih se nalaze sve ostale: odabir1() i odabir2(). Funkcija odabir1() omogućuje enkripciju, a odabir2() omogućuje dekripciju.

8.1. Funkcija za odabir akcije: odabir1()

Funkcije odabir1() i odabir2() se izvršavaju prve i pomoću njih se na početku korisniku nudi odabir akcije koju želi izvršiti.

Kod:

```
def odabir1():
    e = Text(root, height=3, width=50)
    e.grid(row=2, column=1)
    Gumb2["state"]=DISABLED
    ...
    GumbE = Button(
        root,
        text="Enkriptiraj tekst",
        command=enkripcija)
    GumbE.grid(row=3, column=1)
```

Funkcija odabir1() je funkcija koja omogućava enkripciju teksta (unutar ove funkcije se nalaze druge funkcije koje to omogućavaju). U kodu iznad se nalaze definicije polja za unos teksta, pozicioniranje tog polja unutar grida i naredba koja isključuje Gumb2 (odnosno gumb „Dekripcija“ koji se nalazi na početku programa). Sadržaj polja za unos se sprema u varijablu e.

Unutar ove funkcije se definira i GumbE, koji izvodi funkciju enkripcija().

8.1.1. Funkcija za enkriptiranje teksta: enkripcija()

Funkcija enkripcija() omogućava enkripciju teksta. Funkcija se nalazi unutar funkcije odabir1().

Kod:

```
def enkripcija():
    x=e.get(1.0, "end-1c")
    niz=""
    if len(x.strip())==0:
       eTekst = Label(root, text="Unesite tekst u polje i pritisnite gumb").grid(row=4, column=1)
    else:
       GumbE["state"]=DISABLED
       for znak in x:
          if znak.strip()=="":
             niz+=znak
          else:
             if ord(znak) in [65531, 65532, 65533]:
                broj=ord(znak)-65533+2
                if broj==0:
                   continue
                niz+=chr(broj)
             else:
                broj=ord(znak)+3
                niz+=chr(broj)
       eTekst = Label(root, text="Enkriptirani tekst je: "+niz, width=50)
       eTekst.grid(row=4, column=1)
       ...
       gUpis = Button(
```

```
        root,
        text="Da",
        state=NORMAL,
        command=upis)
    gUpis.grid(row=6, column=0)

    gNeUpis = Button(
        root,
        text="Ne",
        state=NORMAL,
        command=neupis)
    gNeUpis.grid(row=6, column=2)
```

Kao što je vidljivo iz koda, prva stvar koju funkcija radi je prikupljanje unosa iz polja za unos koje je definirano unutar funkcije odabir1(), tj. prikupljanje sadržaja pohranjenog u varijablu e. Parametri unutar metode .get() označavaju početnu i krajnju točku prikupljanja sadržaja: 1.0 označava da se sadržaj prikuplja od prvog reda i nultog mjesta, end-1c znači da se prikuplja do kraja unesenog teksta i pomoću -1c se briše oznaka novog reda (\n) koju metoda sama dodaje na kraj sadržaja. Prikupljeni sadržaj se pohranjuje u varijablu x. Zatim se definira varijabla niz u koju će se dodavati enkriptirani znakovi.

Prvi uvjet koji se nalazi unutar funkcije definira da korisnik mora unijeti neki tekst u polje za unos teksta, inače program ispisuje tekst „Unesite tekst u polje i pritisnite gumb“. Bjeline se same po sebi ne računaju kao sadržaj, odnosno ako korisnik npr. unese samo tabulator, novi red ili razmake u polje, metoda .strip() će taj sadržaj izbrisati i sadržaj polja će biti jednak vrijednosti 0, što će aktivirati uvjet.

Nakon što korisnik unese tekst, započinje se izvođenje for petlje koja iterira kroz svaki znak unutar unesenog teksta i zamjenjuje ga znakom 3 mjesta desno u ASCII tablici. Jedini znakovi koji se ne mijenjaju su bjeline (razmak, tabulator, novi red), koje se dodaju varijabli niz u svom izvornom obliku (npr. kad bi se enkriptirao tekst „pišem završni rad“ bez navedenog uvjeta enkriptirani tekst bi bio „slŤhp#}dyuŤql#udg“, tj. svaki razmak bi se zamijenio znakom #).

Poseban uvjet se odnosi na znakove na zadnja tri mjesta u ASCII tablici. Ukoliko korisnik pokuša enkriptirati te znakove, oni će se enkriptirati u znakove koji su na prva tri mjesta u ASCII tablici (isključen je jedino znak na nultom mjestu ASCII tablice, tj. NULL znak, jer se u protivnom ispisuje prazan niz). Kada bi korisnik pokušao enkriptirati navedene znakove bez ovoga uvjeta, program ne bi funkcionirao i rezultat bi bila greška.

Na kraju funkcije nalazi se naredba koja ispisuje enkriptirani tekst. Unutar funkcije se također definiraju gumbi gUpis i gNeUpis, koji izvode funkcije upis(), odnosno neupis().

8.1.2. Funkcija za unos imena datoteke u koju korisnik upisuje rezultat: upis()

Funkcija upis() je funkcija koja omogućava unos imena datoteke u koju korisnik želi upisati enkriptirani ili dekriptirani tekst. Jedan je od dva izbora koji su ponuđeni korisniku nakon što program ispiše rezultat enkripcije ili dekripcije (drugi je neupis()). Funkcija upis() i njene podfunkcije su iste i za enkripciju i za dekripciju. Nalazi se unutar funkcije enkripcija().

Kod:

```
def upis():
    gNeUpis["state"]=DISABLED
    Tekst = Label(root, text="Unesite ime datoteke s ekstenzijom i pritisnite Upiši").grid(row=7, column=1)
    u = Entry(root, width=50)
    u.grid(row=8, column=1)
    ...
    gUpisi = Button(
        root,
        text="Upiši",
        command=upis_u_datoteku)
    gUpisi.grid(row=9, column=1)
```

Prva naredba unutar funkcije isključuje gumb „Ne“ koji se pojavljuje unutar programa. Time se onemogućava korisniku da odabere drugu akciju. Zatim se definira polje u koje korisnik unosi ime datoteke u koju želi upisati rezultat izvođenja programa i pozicija polja unutar programa. Sadržaj polja za unos se pohranjuje u varijablu u.

Također u ovu funkciju spada i gumb gUpisi, pomoću kojeg se izvodi funkcija upis_u_datoteku().

8.1.3. Funkcija za upisivanje rezultata u datoteku: upis_u_datoteku()

Funkcija upis_u_datoteku() sadrži naredbe pomoću kojih se rezultat upisuje u datoteku. Nalazi se unutar funkcije upis().

Kod:

```
def upis_u_datoteku():
    d=u.get()
    if len(d.strip())==0:
       uTekst = Label(root, text="Unesite tekst u polje i pritisnite gumb").grid(row=10, column=1)
    else:
       gUpisi["state"]=DISABLED
       with open(d.strip(), "a", encoding="utf-8") as f:
          f.write(eTekst.cget("text")[23:]+"\n")
       f.closed
       Tekst = Label(root, text="Enkriptirani tekst je zapisan u datoteku "+d.strip()).grid(row=10, column=1)
       gZatvori = Button(
           root,
           text="Zatvori program",
           command = lambda : root.destroy()).grid(row=11, column=1)
```

Funkcija prvo dohvaća ime datoteke koju je korisnik upisao u polje iz prethodne funkcije i pohranjuje u varijablu d. Uvjetom je definirano da korisnik mora unijeti sadržaj u polje. Ukoliko korisnik ne unese sadržaj, ispisuje se tekst koji upozorava korisnika da je unos obavezan. Nakon što korisnik unese sadržaj u polje, program otvara datoteku s tim imenom, ili stvara novu ako ne postoji i upisuje rezultat. Korišten je modus „a“ odnosno append, koji omogućava da se novi tekst dodaje nakon postojećeg teksta u datoteci. Pomoću metode .cget() dohvaća se rezultat programa koji je potrebno upisati u datoteku. Metoda .cget() se razlikuje od metode .get() po tome što može dohvatiti sadržaj Label widgeta, koji metoda

.get() ne može dohvatiti. Upisuju se znakovi od 24 znaka do kraja rezultata (jer bi inače u datoteku bio upisan i tekst „Enkriptirani tekst je:“). Sljedeća naredba zatvara datoteku i nakon toga se nalazi naredba koja ispisuje tekst „Enkriptirani tekst je zapisan u datoteku“.

Idućom naredbom unutar funkcije definira se gumb pod nazivom gZatvori, pomoću kojeg se program može zatvoriti. Zatvaranje programa se vrši uz pomoć lambda funkcije i pozivanja metode root.destroy().

8.1.4. Funkcija za završetak programa bez upisa rezultata u datoteku: neupis()

Funkcija neupis() je funkcija koja omogućava završetak izvođenja programa bez upisa rezultata u datoteku. Jedan je od dva izbora koji su ponuđeni korisniku nakon što program ispiše rezultat (drugi je upis()). Funkcija se nalazi unutar funkcije enkripcija().

Kod:

```
def neupis():
    gUpis["state"]=DISABLED
    Tekst = Label(root, text="U redu. Hvala na korištenju programa.").grid(row=7, column=1)
    gZatvori = Button(
        root,
        text="Zatvori program",
        command = lambda : root.destroy()).grid(row=8, column=1)
```

Prva naredba mijenja stanje gumba gUpis, tj. isključuje ga kako ga korisnik više ne bi mogao pritisnuti nakon što je odabrao da ne želi upisati tekst u datoteku. Zatim se idućom naredbom ispisuje tekst zahvale za korištenje programa i definira se gumb kojim se program zatvara. Zatvaranje programa se vrši na isti način kao unutar funkcije upis_u_datoteku(), pomoću lambda funkcije.

8.2. Funkcija za odabir akcije: odabir2()

Drugi izbor na početku programa je funkcija odabir2(). Unutar ove funkcije nalaze se funkcije koje omogućavaju dekripciju teksta koji je enkriptiran pomoću Cezarove šifre.

Kod:

```
def odabir2():
```

```
    e = Text(root, height=3, width=50)
    e.grid(row=2, column=1)
    Gumb1["state"]=DISABLED
    ...
    GumbE = Button(
        root,
        text="Dekriptiraj tekst",
        command=dekripcija)
    GumbE.grid(row=3, column=1)
```

Slično kao i kod funkcije odabir1(), funkcija odabir2() unutar sebe definira polje u koje se unosi tekst i poziciju polja unutar grida. Polje je visine 3 reda i širine 50 znakova. Sadržaj polja se pohranjuje u varijablu e.

Funkcija odabir2(), slično kao i funkcija odabir1() mijenja stanje drugog gumba u DISABLED, čime se onemogućuje naknadno pritiskanje drugog gumba, tj. gumba „Enkripcija“.

Također se unutar funkcije definira i gumb pod nazivom GumbE, koji izvršava funkciju dekripcija().

8.2.1. Funkcija za dekriptiranje teksta: dekripcija()

Funkcija dekripcija() se nalazi unutar funkcije odabir2(). Sadrži naredbe koje su potrebne za dekriptiranje teksta enkriptiranog pomoću Cezarove šifre.

Kod:

```
def dekripcija():
    x=e.get(1.0, "end-1c")
    niz=""
    if len(x.strip())==0:
       eTekst = Label(root, text="Unesite tekst u polje i pritisnite
gumb").grid(row=4, column=1)
    else:
       GumbE["state"]=DISABLED
```

```
        for znak in x:
           if znak.strip()=="":
              niz+=znak
           else:
              if ord(znak) in [0, 1, 2]:
                 broj=ord(znak)-2+65533
                 if broj==0:
                    continue
                 niz+=chr(broj)
              else:
                 broj=ord(znak)-3
                 niz+=chr(broj)
        eTekst = Label(root, text="Dekriptirani tekst je: "+niz,
width=50)
        eTekst.grid(row=4, column=1)
        ...
        gUpis = Button(
            root,
            text="Da",
            state=NORMAL,
            command=upis)
        gUpis.grid(row=6, column=0)


        gNeUpis = Button(
            root,
            text="Ne",
            state=NORMAL,
            command=neupis)
        gNeUpis.grid(row=6, column=2)
```

Funkcija je slična funkciji enkripcija(). Prvom naredbom se prikuplja sadržaj koji je unesen u polje za unos teksta (vrijednost varijable e) i sprema u varijablu x. Zatim se definira varijabla niz u koju se dodaju dekriptirani znakovi.

Algoritam kojim program dekriptira tekst radi na način da svaki znak zamijeni znakom koji se nalazi tri mjesta lijevo u ASCII tablici.

Isto kao i kod funkcije enkripcija() i kod funkcije dekripcija() postoji više uvjeta. Prvi uvjet je da neki znak mora biti upisan u polje za unos teksta. Bjeline se same po sebi neće prihvatiti kao sadržaj i program će ispisati poruku „Unesite tekst u polje i pritisnite gumb“.

Ukoliko je korisnik upisao sadržaj u polje, sadržaj se prikuplja i uz pomoć for petlje se kroz njega iterira, znak po znak. Ukoliko je znak neka vrsta bjeline, npr. razmak, tabulator ili novi red, znak se dodaje u izvornom obliku u varijablu niz (tj. ne vrši se dekripcija tih znakova).

Poseban uvjet se također odnosi na prva tri znaka u ASCII tablici. Ukoliko se radi dekripcija tih znakova, oni se dekriptiraju u znakove koji su na posljednja tri mjesta u ASCII tablici. Kada uvjet ne bi postojao rezultat izvođenja programa bila bi greška. U varijablu niz se također ne dodaje NULL vrijednost, nego se ta vrijednost preskače. Time se izbjegava potpuni gubitak podataka jer rezultat s NULL vrijednosti bi bio prazan niz, ali dovodi do manjeg gubitka podataka jer se ta vrijednost ne upisuje.

Na kraju funkcije nalazi se naredba pomoću koje se dekriptirani tekst ispisuje u programu i definiraju gumbi gUpis, gNeUpis, koji izvode funkcije upis(), odnosno neupis().

Ostale funkcije koje se nalaze unutar funkcije odabir2() (funkcije upis(), upis_u_datoteku() i neupis()) su iste kao i unutar funkcije odabir1().

Na kraju datoteke s funkcijama nalaze se naredbe pomoću kojih se ispisuje tekst „Odaberite akciju“ unutar programa i pomoću kojih se definiraju Gumb1 kojim se izvršava funkcija odabir1() i Gumb2 kojim se izvršava funkcija odabir2().

Kod:

```
myLabel1 = Label(root, text="Odaberite akciju").grid(row=0, column=1)

Gumb1 = Button(

    root,

    text="Enkripcija",
```

```
    state=NORMAL,
    command=odabir1)
Gumb1.grid(row=1, column=0)
Gumb2 = Button(
    root,
    text="Dekripcija",
    state=NORMAL,
    command=odabir2)
Gumb2.grid(row=1, column=2)
root.mainloop()
```

Na kraju datoteke nalazi se naredba root.mainloop(), koja je obavezan dio Tkinter aplikacije, a čija je svrha objašnjena u poglavlju Glavni program.

8.3. Pozicioniranje elemenata

Za pozicioniranje elemenata korištena je metoda .grid(). Metoda omogućava tablično pozicioniranje widgeta, to jest pozicioniranje pomoću redaka i stupaca.

Sintaksa metode je: `widget.grid(grid_options)`

Postoje razne opcije koje se mogu koristiti:

- column – definira stupac u koji se pozicionira widget,
- row – definira red u koji se pozicionira widget,
- rowspan – definira koliko redova zauzima widget,
- colspan – definira koliko stupaca zauzima widget i druge opcije (Tutorials Point, n.d.).

Tablica 1. Skica grida

	myLabel1	
Gumb1		Gumb2
	e	
	GumbE	
	eTekst	
	eUpis	
gUpis		gNeUpis
	Tekst	
	u ili gZatvori	
	gUpisi	
	Tekst	
	gZatvori	

Kao i što je vidljivo iz tablice 1, grid se sastoji od tri stupca i maksimalno jedanaest redaka. Broj redaka u gridu ovisi o izabranim akcijama tokom korištenja programa. Najmanji mogući broj redaka je devet (ukoliko korisnik odabere da ne želi upisati tekst u datoteku), a najveći jedanaest (ukoliko korisnik odabere da želi upisati tekst u datoteku).

Radi preglednosti i jednostavnosti, polja za unos teksta i ispis teksta, kao i pojedini gumbi su postavljeni u centar (središte) prozora, dok se na lijevoj i desnoj strani prozora (tj. prvom i trećem stupcu) nalaze gumbi pomoću kojih korisnik odabire akcije.

9. Nedostaci programa

Program učinkovito služi za šifriranje teksta pomoću Cezarove šifre, jer napadač (barem neko vrijeme) iz šifrata ne može shvatiti o čemu se radi u tekstu. Također jedna od prednosti je ta što će se zadnja tri slova abecede pretvoriti u simbole koji u ASCII tablici slijede nakon tih slova, što će dodatno pridonijeti sigurnosti šifrata jer je potreban ovaj program kako bi se pravilno dešifrirao tekst. Međutim u takvim slučajevima najčešće će se iz konteksta i ostatka riječi moći zaključiti o kojem se slovu radi.

Program ima više nedostataka. Jedan od nedostataka je taj što program neće šifrirati (a sukladno tomu i dešifrirati) tekst na isti način kao što se to radi ručno, odnosno rezultat će biti drukčiji. Razlog za to je što kada se ručno šifrira uz pomoć Cezarove šifre postoji rotacija unutar abecede. Time se postiže da se zadnja tri znaka abecede, u slučaju engleske abecede x, y i z, pretvaraju u a, b i c. Prema tome engleska riječ „happy“ bi se ručno uz pomoć Cezarove šifre šifrirala u „kdssb“, dok bi rezultat programa bio „kdss|“, odnosno zadnje slovo y bi se šifriralo u simbol | koji se u ASCII tablici nalazi tri mjesta desno od slova y. Razlike u šifratima koji se dobiju ručno i koji se dobiju pomoću programa su još veće ako se šifrira tekst na nekom drugom jeziku, upravo zbog razlika u poretku slova unutar abecede, ali i pozicije posebnih slova nekog jezika u ASCII tablici (problem koji je prisutan i u „ručnoj“ verziji Cezarove šifre). Autor programa se nije odlučio na rotaciju unutar abecede jer bi time program uspješno šifrirao i dešifrirao samo tekst na engleskom jeziku.

Drugi nedostatak je gubitak podataka do kojeg dolazi u određenim slučajevima. Jedan od takvih slučajeva je NULL vrijednost, koja može biti rezultat i šifriranja i dešifriranja određenih znakova unutar ASCII tablice. Kao i što se može naslutiti iz naziva, NULL vrijednost je vrijednost koja ne nosi nikakve podatke. Ukoliko se NULL vrijednost nalazi u šifratu ili je rezultat dešifriranja, rezultat izvođenja programa bit će prazan niz, neovisno o tome nalaze li se u rezultatu i neki drugi znakovi. Problem je riješen tako što se uz pomoć posebnog uvjeta unutar algoritma za šifriranje, odnosno dešifriranje, NULL vrijednost ne upisuje u varijablu niz i time se manjim gubitkom podataka izbjegava potpuni gubitak podataka. Drugi slučaj su znakovi bjeline, koji također imaju svoje vrijednosti unutar ASCII tablice. Ukoliko se tekst šifrira pomoću ovog programa ovaj problem se neće pojaviti s obzirom da se bjeline u svom originalnom obliku dodaju varijabli niz. Problem će se pojaviti ako se u tekstu koji korisnik želi dešifrirati nalaze simboli koji se nalaze tri mjesta desno od znakova bjeline, npr. znak # koji se u ASCII tablici nalazi tri mjesta desno od znaka razmaka

(space). Prema tome kada bi korisnik programom dešifrirao tekst „slŤhp#}dyuŤql#udg“, rezultat bi bio „pišem završni rad“. Međutim kada bi otvoreni tekst pokušao ponovo šifrirati, rezultat bi bio „slŤhp }dyuŤql udg“, čime je došlo do gubitka znaka # koji se nalazio u početnome šifratu.

10. Zaključak

Glavni cilj završnog rada bio je razviti program koji će učinkovito šifrirati i dešifrirati tekst koji korisnik unese u program.

U prvom dijelu programa ukratko je predstavljena povijest kriptografije, različiti algoritmi koji su se koristili kroz povijest, ali i neki moderni algoritmi poput DES ili AES algoritma, koji se koriste i danas. Nakon predstavljanja povijesti, detaljnije je objašnjena Cezarova šifra, pomoću koje program šifrira i dešifrira tekst koji korisnik unese u program.

Nakon toga ukratko je opisan programski jezik Python koji je korišten za izradu programa. Ukratko je opisana povijest Pythona, sintaksa i semantika, razvojna sučelja koja se mogu koristiti i slično. Ukratko je opisan i modul Tkinter, koji je korišten za izradu grafičkog sučelja programa.

Zatim je u radu opisan program koji je razvijen za potrebe ovog rada. Predstavljen je opis izgleda programa i funkcije koje ima. Detaljno su opisani glavni dio programa, koji generira naslovnu stranicu i poziva funkcije i datoteka s funkcijama koje omogućavaju glavne funkcionalnosti koje program ima. Opisan je i način na koji je sadržaj strukturiran unutar programa. Na kraju su predstavljeni i neki nedostaci koje program ima.

Kao i što je već spomenuto u završnom radu, program učinkovito služi za šifriranje i dešifriranje korisničkog unosa. Mogućnost upisa šifriranog ili dešifriranog teksta u datoteku je svakako prednost koju valja istaknuti. Program je koncipiran na način da ga bilo koji korisnik, neovisno o informatičkom znanju koje posjeduje može koristiti. Izgled programa je jednostavan i podsjetnici unutar programa pomažu korisniku prilikom korištenja programa.

Program dakako ima svojih nedostataka i tome je posvećeno cijelo poglavlje ovog završnog rada. Mogućnosti za optimizaciju i poboljšanje su gotovo neograničene, od dodavanja novih funkcionalnosti, poput biranja različitih algoritama za šifriranje teksta, šifriranje i dešifriranje datoteka, poboljšanje izgleda programa i slično.

Literatura

1. Dujella, A. & Maretić, M., 2007. *Kriptografija.* Zagreb: Element.
2. GeeksforGeeks, n.d. *Difference Between Syntax and Semantics.* [Mrežno] Dostupno na: https://www.geeksforgeeks.org/difference-between-syntax-and-semantics/ [Pristupljeno 5 Travanj 2023.].
3. GeeksforGeeks, n.d. *Python | pack() method in Tkinter.* [Mrežno] Dostupno na: https://www.geeksforgeeks.org/python-pack-method-in-tkinter/ [Pristupljeno 7 Travanj 2023.].
4. Horcasitas, J., 2021. *What Is REPL?.* [Mrežno] Dostupno na: https://www.digitalocean.com/community/tutorials/what-is-repl [Pristupljeno 5 Travanj 2023.].
5. Hrvatska enciklopedija, mrežno izdanje, 2021. *kriptografija.* [Mrežno] Dostupno na: https://enciklopedija.hr/natuknica.aspx?ID=33988 [Pristupljeno 31 Ožujak 2023.].
6. Lott, S. F., 2015. *Using the Read-Evaluate-Print Loop (REPL).* [Mrežno] Dostupno na: https://subscription.packtpub.com/book/application-development/9781784390341/1/ch01lvl1sec09/using-the-read-evaluate-print-loop-repl [Pristupljeno 5 Travanj 2023.].
7. Moore, K., Satyabrata, D., Eli, R. & Calvin, L., n.d. *Caesar Cipher.* [Mrežno] Dostupno na: https://brilliant.org/wiki/caesar-cipher/ [Pristupljeno 4 Travanj 2023.].
8. National Institute of Standards and Technology, 2022. *NIST Announced First Four Quantum-Resistant Cryptographic Algorithms.* [Mrežno] Dostupno na: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms [Pristupljeno 3 Travanj 2023.].
9. Peters, T., 2004. *PEP 20 - The Zen of Python.* [Mrežno] Dostupno na: https://peps.python.org/pep-0020/ [Pristupljeno 5 Travanj 2023.].

10. Python Software Foundation, n.d. *General Python FAQ.* [Mrežno] Dostupno na: https://docs.python.org/3/faq/general.html [Pristupljeno 5 Travanj 2023.].

11. Python Software Foundation, n.d. *Tkinter.* [Mrežno] Dostupno na: https://wiki.python.org/moin/TkInter [Pristupljeno 6 Travanj 2023.].

12. Python Software Foundation, n.d. *tkinter - Python interface to Tcl/Tk.* [Mrežno] Dostupno na: https://docs.python.org/3/library/tkinter.html [Pristupljeno 6 Travanj 2023.].

13. Sidhpurwala, H., 2023. *A Brief History of Cryptography.* [Mrežno] Dostupno na: https://www.redhat.com/en/blog/brief-history-cryptography [Pristupljeno 1 Travanj 2023.].

14. Simmons, G. J., n.d. *History of cryptology.* [Mrežno] Dostupno na: https://www.britannica.com/topic/cryptology/History-of-cryptology [Pristupljeno 1 Travanj 2023.].

15. Tutorials Point, n.d. *Python - Tkinter grid() Method.* [Mrežno] Dostupno na: https://www.tutorialspoint.com/python/tk_grid.htm [Pristupljeno 10 Travanj 2023.].

16. TutorialsTeacher, n.d. *Python - IDLE.* [Mrežno] Dostupno na: https://www.tutorialsteacher.com/python/python-idle [Pristupljeno 5 Travanj 2023.].

17. Wikipedia, n.d. *Advanced Encryption Standard.* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard [Pristupljeno 3 Travanj 2023.].

18. Wikipedia, n.d. *Cezarova šifra.* [Mrežno] Dostupno na: https://hr.wikipedia.org/wiki/Cezarova_%C5%A1ifra [Pristupljeno 4 Travanj 2023.].

19. Wikipedia, n.d. *IDLE.* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/IDLE [Pristupljeno 5 Travanj 2023.].

20. Wikipedia, n.d. *Kriptologija.* [Mrežno] Dostupno na: https://hr.wikipedia.org/wiki/Kriptologija [Pristupljeno 31 Ožujak 2023.].

21. Wikipedia, n.d. *Python (programming language).* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/Python_(programming_language) [Pristupljeno 5 Travanj 2023.].

22. Wikipedia, n.d. *Read-eval-print loop.* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop [Pristupljeno 5 Travanj 2023.].

23. Wikipedia, n.d. *Tk (software).* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/Tk_(software) [Pristupljeno 6 Travanj 2023.].

24. Wikipedia, n.d. *Tkinter.* [Mrežno] Dostupno na: https://en.wikipedia.org/wiki/Tkinter [Pristupljeno 6 Travanj 2023.].

Popis tablica

Popis slika

Izrada programa za šifriranje i dešifriranje teksta uz pomoć Cezarove šifre

Sažetak

Tekst je jedan od informacijskih oblika kojim se informacije mogu prenijeti od izvora do odredišta. Informacije koje tekst prenosi u nekim slučajevima mogu biti povjerljive i u tom slučaju njih je potrebno odgovarajuće zaštititi, a jedan od načina je šifriranje teksta. Postoje različiti načini za šifriranje teksta, a za potrebe ovog rada koristit će se Cezarova šifra, koja svako slovo u tekstu zamjenjuje slovom tri mjesta desno u abecedi.

Program će se izraditi u programskom jeziku Python. Cilj rada je izrada programa koji će šifrirati tekst koji korisnik unese u program. Za šifriranje teksta koristit će se ASCII tablica znakova. Svaki znak će se mijenjati znakom koji se nalazi tri mjesta desno u ASCII tablici. Program će imati mogućnost dešifriranja teksta šifriranog pomoću Cezarove šifre.

Ključne riječi: program, šifriranje, dešifriranje, funkcije, šifra, programiranje, Python

Creation of a program for text encryption and decryption using Caesar cipher

Summary

Text is one of the information forms by which information can be transferred from source to destination. The information transferred by the text can in some cases be confidential and in that case it needs to be properly protected, and one way is to encrypt the text. There are different ways to encrypt text, and for the purposes of this thesis, the Caesar cipher will be used, which replaces each letter in the text with the letter three places to the right in the alphabet.

The program will be created in the Python programming language. The goal of the thesis is to create a program that will encrypt the user's textual input. The ASCII character table will be used to encrypt the text. Each character will be replaced by the character located three places to the right in the ASCII table. The program will have the ability to decrypt text encrypted using the Caesar cipher.

Key words: program, encryption, decryption, functions, cipher, programming, Python