Programarea Calculatoarelor (2020 - 2021)

Deadline: 15.01.2021 23:55

Password Manager

Responsabili Tema:
Andreea Nica,
Andrei Simescu

Cuprins

Istoric modificari:

- v1.01 - Modificat precizari despre numele fisierelor sursa

1 Let's get started - Ce este Password Manager?

Majoritatea utilizatorilor folosesc parole slabe (`https://howsecureismypassword.net/`) și le reutilizează pentru diferite website-uri. Intrebarea fireasca apare: Cum ar trebui sa generam si, mai greu, sa retinem parole puternice (care contin atat numere, cat si simboluri) si unice pentru toate website-urile pe care le folosim? Solutia: Password Manager.

Password Managerul genereaza si salveaza chei de logare pentru toate website-urile pe care le accesezi urmand ca, mai apoi, sa realizeze logarea automata. Pentru fiecare website in parte este salvata o parola criptata cu ajutorul unei master password, singura pe care va trebui sa o retii.

Tema 2 are ca scop realizarea anumitor functionalitati ce simuleaza comportamentul unui Password Manager.

2 Cerinta I - Criptare Parole : Cifrul Vigenere modificat

Pentru a mentine datele in siguranta, acestea **NU** trebuie pastrate in clar. O metoda de criptare, utilizata inca din 1553, este Cifrul lui Vigenere. Pentru criptare, este nevoie de o cheie, data de catre utilizator. Algoritmul modificat urmeaza urmatorii pasi:

1. Cheia de criptare este multiplicata pana atinge dimensiunea textului ce trebuie cifrat.

2. Litera de pe pozitia i din text va fi deplasata cu un numar de pozitii egal cu deplasarea literei de pe pozitia i a cheii fata de litera 'a'.

3. Literele mari se vor situa in continuarea literelor mici, adica daca deplasarea va depasi litera 'z', se va continua numararea de la litera 'A'. Cand se depaseste litera 'Z', se reia cu litara 'a'

Sarcina voastra este ca, avand la dispozitie textul in clar si o cheie, sa se intoarca textul criptat folosind varianta modificata a Cifrul Vigenere.

Date intrare: un sir de caractere reprezentand cheia, urmat de un sir de caractere reprezentant textul clar. Cele doua siruri de caractere sunt despartite printr-un spatiu si vor fi prelucrate doar daca sunt formate din litere.

Indicatie: Se va scrie o functie care verifica daca un sir contine doar litere. Daca sirul contine alte caractere in afara de litere, atunci sirul va fi considerat invalid. Functia va fi aplicata atat asupra cheii, cat si asupra textului in clar.

Date iesire: un sir de caractere ce reprezinta textul criptat daca sirul dat ca data de intrare este valid sau sirul "INVALID" altfel.

Restrictii: sirurile de caractere folosite vor avea dimensiunea de maxim 100 caractere.

Exemplu 1:

—-Intrare :

bcaA AzYNZbccMn

—-Iesire:

BBYnadcCNp

—-Explicatie:

Deoarece lungimea cheii este mai mica decat lungimea textului, vom multiplica cheia pentru a acoperi intregul text. Cheia va deveni bcaAbcaAbc. Vom incepe cu prima litera: Litera b din cheie se afla la distanta 1 de litera 'a', deci litera 'A' din textul clar va fi inlocuita cu litera 'B'. Litera 'c' din cheie se afla la distanta de 2 de litera 'a', deci litera 'z' din textul clar va fi inlouita de 'B'. Litera 'a' din cheie se afla la distanta 0 de litera 'a', deci 'Y' din textul clar va ramane nemodificat. Litera 'A' din cheie se afla la distanta '26' de litera 'a' (pana la 'z' avem distanta 25, plus inca 1 pana la 'A'). Astfel ca litera 'N' din text va fi inlocuita cu litera 'n' si asa mai departe.

Exemplu 2:

—-Intrare :

bca2 AzYNZbccMn

—-Iesire:

INVALID

—-Explicatie:

Cheia contine caracterul '2', desi sirul este invalid

Exemplu 3:

—-Intrare :

bca AzYNZbcc)n

—-Iesire:

INVALID

—-Explicatie:

Textul in clar contine caracterul ')', deci sirul este invalid

3 Cerinta II - Automatically fill

Password Manager poate fi configurat pentru a completa automat anumite date in formulare, ca de exemplu nume, adresa, numar de telefon, adresa de email etc.

Sarcina voastra este sa implementati aceasta operatie de Autofill pe un text (formular) primit, dandu-se o lista de campuri si valorile cu care vor fi completate.

Date intrare: un numar intreg N, reprezentand numarul de perechi camp - valoare, urmate de N randuri continand denumirea campului urmata de valoarea ce trebuie inserata, sub forma a doua siruri pe acelasi rand, despartite printr-un spatiu. Dupa acestea, urmeaza textul formularului pe care se va aplica functia de Autofill (textul formularului va avea cel mult 1000 de caractere).

Date iesire: Textul formularului cu valorile corespunzatoare inserate.

Restrictii:

- In implementarea voastra trebuie sa folositi urmatoarea structura:

  struct pereche {

  *char *camp, *valoare;*

  };

- Perechile vor fi memorate intr-un vector alocat dinamic, vectorul avand alocata exact atata memorie cate perechi sunt.
- Fiecare sir de caractere din cadrul unei perechi, precum si textul rezultat in urma operatiei de automatically fill trebuie sa aiba alocata exact atata memorie cata este nevoie (nici mai multa nici mai putina).

Exemplu 1:

—-Intrare :

3

NUME Popescu

PRENUME Ion

MAIL ion.popescu@email.com

Numele meu este NUME PRENUME si am adresa de email MAIL.

—-Iesire:

Numele meu este Popescu Ion si am adresa de email ion.popescu@email.com.

4 Cerinta III - Generare Parole

Unul dintre cele mai importante aspecte ale securitatii este siguranta parolelor. O parola trebuie sa fie un cuvant greu de ghicit, ceea ce inseamna ca nu poate fi asociata cu userul (numele persoanelor apropiate, animale de companie). De asemenea, nu trebuie sa contina secvente numerice asociate cu o persoana (numar de telefon, data de nastere). Un Password Manager genereaza parole sigure din punct de vedere computational.

Sarcina voastra este ca, pornind de la o radacina, sa generati o parola urmand urmatorul algoritm:

1. Pornind de la un intreg *seed* primit ca data de intrare, sa se genereze aleator (folosind functia *rand()*) un sir de caractere aleator de o dimensiune precizata. Sirul vostru va contine doar caracterele din lista urmatoare (caracterele cu codurile ASCII cuprinse intre 32 si 125):

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^					
63	3F	077	?	?	95	5F	137	_	_					

2. Pe sirul rezultat, pentru fiecare caracter in parte se va aplica aleator o functie de codificare din urmatoarele trei posibile:

- Codificarea 1: Un caracter va fi inlocuit de sirul format din concatenarea caracterului respectiv cu numarul de aparitii ale caracterului in sir pana la momentul curent (caracterul curent nu este numarat)

  Exemplu:

  Dupa aplicarea codificarii 1 asupra tuturor caracterelor din sirul urmator se obtine:

  nnan ——>*n0n1a0n2*

- Codificarea 2. Asupra caracterului se vor aplica urmatoarele operatii pe biti: se vor inversa bitii de pe pozitia 3 si 6 de la LSB (*Least Significant Bit*), se va extrage numarul de biti de 1 din numarul rezultat, se va face operatia de *SAU* logic cu masca 00100000, dupa care se vor seta la 0 bitul *LSB* (bitul 0). In sir, caracterul va fi inlocuit de sirul rezultat prin concatenarea noului caracter cu numarul de biti de 1 obtinut inaintea aplicarii operatiei de *SAU*

  Exemplu: Pentru caracterul 'c' cu codul ASCII *01100011*

  Inversam bitii de pe pozitia 3 si 6 ——>*00101011*

  Numarul de biti de 1 ——>*4*

  Operatia de *SAU* logic cu masca *00100000* ——>*00101011*

  Setarea la 0 a *LSB* ——>*00101010*

  Rezultatul va fi **4*

- Codificarea 3: Transforma literele mici in litere mari. Asupra altor tipuri de caractere nu se aplica nici o operatie, caracterul ramanand nemodificat

Date intrare: Doua numere intregi, reprezentand seedul de la care se va porni generarea aleatoare urmat de dimensiunea sirului de caractere generat

Date iesire: parola generata

Exemplu 1:

—-Intrare :

10 4

—-Iesire:

*)0\F*3*

—-Explicatie:

Sirul generat aleator va fi: *)\FC*

Pentru caracterul *)* se va aplica codificarea 0, rezultand *)0*

Pentru caracterul \ se va aplica codificarea 2, rezultand \ (cum caracterul nu e litera mica, nu se modifica nimic)

Pentru caracterul *F* se va aplica codificarea 2, rezultand *F*

Pentru caracterul *C* se va aplica codificarea 1, rezultand **3*

Restrictii:

- Generarea parolei va fi separata in mai multe functii: o functie care genereaza stringul aleator, o alta functie care transforma sirul generat, functii pentru fiecare tip de codificare etc.
- Toate codificarile posibile ce pot fi efectuate asupra unui caracter se vor retine intr-un vector de pointeri la functii, vector ce va fi trimis ca argument functiei de codificare a sirului generat aleator
- Alegerea codificarii ce va fi aplicata asupra caracterului curent se va face aleator, una din cele trei posibile, prin generarea aleatoare a unui index (0,1 sau 2) corespunzator unei functii din vectorul de codificari. **Obs**. Indexul 0 corespunde codificarii 1, samd.
- Functiile de codificare vor respecta urmatorul antet: primesc ca argument un caracter si intorc un pointer catre un sir de caractere.
- Pentru calculul numarului de aparitii ale caracterelor se va folosi un vector alocat static in interiorul functiei ce realizeaza codificarea 1. In vector se va completa pentru fiecare caracter (dintre cele posibile: coduri ASCII 32 - 125) numarul de aparitii al caracterului respectiv la un moment dat.
- Nu uitati sa eliberati memoria dupa ce nu mai utilizati un sir alocat dinamic

5 Restrictii si precizari

Separati logica programelor in mai multe functii, asa cum vi s-a cerut in enunturi! Nu se vor puncta sursele in care tot programul este scris in main!

Toate citirile se fac de la tastatura. Nu folositi fisiere. Datele de intrare sunt descrise pentru fiecare cerinta individual si vor fi citite de la tastatura. Pentru a nu fi nevoiti sa scrieti de fiecare data inputul, puteti redirecta un fisier in care sa scrieti datele pe care de obicei le introduceati de la tastatura, si sa rulati folosind formatul urmator:

.nume_executabil < numele_fisierului_de_intrare

Este interzisă folosirea variabilelor globale.

Soluția temei va fi scrisă în C. Nu folosiți sintaxă sau instrucțiuni specifice limbajului C++.

Fiecare problema va fi scrisa intr-un fisier sursa separat, numit problemaX.c, X = {1,2,3}.

Fișierul Makefile va conține doua reguli: **build** (implicit) și **clean**. Regula build va genera 3 executabile, iar clean va șterge toate fișierele create de compilare și execuție.

In **README** precizați cât timp v-a luat implementarea programului vostru și explicați, pe scurt, implementarea temei (comentariile din cod vor documenta mai amănunțit rezolvarea). Este recomandat ca liniile de cod și cele din fișierul README să nu depășească 80 de caractere

Folosiți un coding style astfel încât codul să fie ușor de citit și înțeles. De exemplu:

—-Dați nume corespunzătoare variabilelor și funcțiilor.

—-Nu adăugați prea multe linii libere sau alte spații goale unde nu este necesar (exemplu: nu terminați liniile în spații libere, trailing whitespaces; nu adăugați prea multe linii libere între instrucțiuni sau la sfârșitul fișierului). Principalul scop al spațiilor este identarea.

—-Fiți consecvenți. Coding style-ul are o natură subiectivă. Chiar dacă pentru unele persoane nu pare bun, faptul că îl folosiți consecvent este un lucru bun.

—-Există programe sau extensii pentru editoare text care vă pot formata codul. Deși vă pot ajuta destul de mult, ar fi ideal să încercați să respectați coding style-ul pe măsură ce scrieți codul.

Temele sunt strict individuale. Copierea temelor va fi sancționată. Persoanele cu porțiuni de cod identice, nu vor primi niciun punctaj pe temă. Temele trimise după deadline nu vor fi luate în considerare.

Nu copiati cod de pe Internet! Se poate ajunge la situatia in care doi studenti sa aiba acelasi cod, preluat de pe Internet, caz in care ambele teme vor fi depunctate, desi studentii nu au colaborat direct intre ei.

Veţi trimite o arhivă **ZIP** cu numele de tipul **GRUPANumePrenume.zip** exemplu: **311CCPopescuMariaIoana.zip** , care va conţine fișierele necesare, Makefile și README. Fisierele trebuie să fie în rădăcina arhivei, nu în alte subdirectoare.

Pentru intrebari despre tema se va folosi in mod exclusiv forumul temei, pe care va recomandam sa il vizitati chiar si daca nu aveti intrebari, intrucat este posibil sa aflati informatii noi din intrebarile puse de colegii vostri, respectiv din raspunsurile date de noi.

Compilarea nu ar trebui să producă avertizări (verificați prin adăugarea flagului -Wall la gcc).

Eliberați memoria alocată dinamic. Folosiți comanda de mai jos pentru a verifica dacă memoria este eliberată corect. *valgrind –tool=memcheck –leak-check=full ./passwordManager*

Temele trebuie să fie încărcate pe **vmchecker**. **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.

6 Notare

Cerinta 1 - 50p

Cerinta 2 - 60p

Cerinta 3 - 70p

Coding style, Makefile, README - 20p

- 15p - Coding Style
  - Comentarii
  - Folosirea a mai multor funcții
  - Logică clară
  - Spațiere corectă
  - Stil consecvent
  - Variabile și funcții denumite adecvat
- 5p - Completarea fișierului README