#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<time.h>

#include<ctype.h>

#include<string.h>

#include<math.h>

/\*1. Napisati program koji prvo pročita koliko redaka ima datoteka, tj. koliko ima studenata zapisanih u datoteci. Nakon toga potrebno je dinamički alocirati prostor za niz struktura

studenata (ime, prezime, bodovi) i učitati iz datoteke sve zapise. Na ekran ispisati ime, prezime,apsolutni i relativni broj bodova.

Napomena: Svaki redak datoteke sadrži ime i prezime studenta, te broj bodova na kolokviju. relatvan\_br\_bodova = br\_bodova/max\_br\_bodova\*100 \*/

struct student;

typedef struct student \*pstudent;

struct student

{

char ime[128];

char prezime[128];

int bodovi;

};

float RelativniBrBodova(int,int);

int BrojStudenata(char\*);

int UnesiStudente(char\*,int,pstudent);

void main()

{

int br\_studenata=0;

pstudent p=NULL;

char imedatoteke[128]="";

int i,max=0;

br\_studenata=BrojStudenata(imedatoteke);

p=(pstudent)malloc(sizeof(struct student)\* br\_studenata);

if(p==NULL)

{

printf("Nije se alociralo!");

exit(1);

}

max=UnesiStudente(imedatoteke,br\_studenata,p);

for(i=0;i< 6;i++)

{

printf("Ime : %s\t Prezime : %s\t Bodovi : %d\t Relativni : %f \n",p[i].ime,p[i].prezime,p[i].bodovi,RelativniBrBodova(max,p[i].bodovi));

}

}

float RelativniBrBodova(int maxi,int index)

{

float relativni=1;

relativni=(float)index / (float)maxi;

relativni=relativni\*100;

return relativni;

}

int BrojStudenata(char\* imedatoteke)

{

FILE \*fp;

char ime[128]="",prezime[128]="";

int bod=0,br\_studenata=0;

printf("Unesite ime datoteke: ");

scanf("%s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(!fp)

{

printf("Datoteka se nije otvorila!\n");

exit(1);

}

else

while(!feof(fp))

{

if ( fscanf(fp,"%s %s %d",ime,prezime,&bod) == 3 )

{

br\_studenata++;

}

else continue;

}

fclose(fp);

return br\_studenata;

}

int UnesiStudente(char\* imedatoteke,int br\_studenata,pstudent p)

{

int i=0,max=0;

FILE\* fp;

fp=fopen(imedatoteke,"r");

if(!fp)

{

printf("Datoteka se nije otvorila!\n");

exit(1);

}

while(!feof(fp))

{

fscanf(fp,"%s %s %d",p[i].ime,p[i].prezime,&p[i].bodovi);

if(p[i].bodovi > max)

max=p[i].bodovi;

i++;

}

fclose(fp);

return max;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<time.h>

#include<ctype.h>

#include<string.h>

#include<math.h>

/\*2. Definirati strukturu osoba (ime, prezime, godina rođenja) i napisati program koji:

a) dinamički dodaje novi element na početak liste,

b) ispisuje listu,

c) dinamički dodaje novi element na kraj liste,

d) pronalazi element u listi (po prezimenu)

e) briše određeni element iz liste,

f) dinamički dodaje novi element iza određenog elementa,

g) dinamički dodaje novi element ispred određenog elementa,

h) sortira listu po prezimenima osoba,

i) upisuje listu u datoteku,

j) čita listu iz datoteke.

\*/

struct Osoba;

typedef struct Osoba \* Pos;

struct Osoba{

char ime[128];

char prezime [128];

int godina\_rodjenja;

Pos Next;

};

Pos CreateNode();

void Dodaj(Pos);

void PrintList(Pos);

void DodajNaKraj(Pos);

Pos FindLast(Pos);

Pos Find(Pos,char\*);

Pos FindPrevious(Pos,char\*);

int Izbrisi(Pos,char\*);

void InsertNext(Pos,char\*);

void InsertPrevious(Pos,char\*);

void SortPoPrezimenu(Pos);

void PrintInFile(Pos,char\*);

void ReadFromFile(Pos,char\*);

void Stars();

void Meni();

void main()

{

Pos p=NULL;

int i=1;

char prezime[128]="",imedatoteke[128]="";

char izbor=0;

p=CreateNode();

p->Next=NULL;

Stars();

while(i)

{

Meni();

puts("Koju naredbu zelite izvrsiti(odaberite slovo pored nje):");

scanf(" %c",&izbor);

switch(tolower(izbor))

{

case 'a': Dodaj(p);

break;

case 'b': PrintList(p);

break;

case 'c': DodajNaKraj(p);

break;

case 'd': puts("Po kojem prezimenu zelite traziti?");

scanf(" %[^\n]s", prezime);

printf("Prezime koje ste trazili nalazi se na adresi 0x%x\n",Find(p,prezime));

break;

case 'e': puts("Koji element zelite obrisati? (upisite prezime)");

scanf(" %[^\n]s", &prezime);

Izbrisi(p, prezime);

printf("Izbrisan je element sa prezimenom %s\n", prezime);

break;

case 'f': puts("Upisite prezime nakon kojeg zelite dodati novi element:");

scanf(" %[^\n]s",&prezime);

InsertNext(p,prezime);

break;

case 'g': puts("Upisite prezime ispred kojeg zelite dodati novi element:");

scanf(" %[^\n]s",&prezime);

InsertPrevious(p,prezime);

break;

case 'h': SortPoPrezimenu(p);

break;

case 'i': puts("Upisite ime datoteke u koju zelite upisati listu:");

scanf(" %[^\n]s",&imedatoteke);

PrintInFile(p,imedatoteke);

break;

case 'j': puts("Upisite ime datoteke iz koje zelite iscitati podatke u listu:");

scanf(" %[^\n]s",&imedatoteke);

ReadFromFile(p,imedatoteke);

break;

case 'k': puts("KRAJ PROGRAMA");

i=0;

continue;

break;

default:

puts("Niste unijeli odgovarajuci znak!");

}

}

}

Pos CreateNode()

{

Pos p;

p=(Pos)malloc(sizeof(Osoba));

if(p==NULL)

{

printf("Nije se alociralo!\n");

}

return p;

}

void Dodaj(Pos p)

{

Pos q;

q=CreateNode();

printf("Unesite ime,prezime i godinu rodjenja:\n");

scanf(" %s %s %d",&q->ime,&q->prezime,&q->godina\_rodjenja);

q->Next=p->Next;

p->Next=q;

}

void PrintList(Pos p)

{

if(p->Next==NULL)

{

printf("Lista je prazna!\n");

}

p=p->Next;

while(p!=NULL)

{

printf("Ime: %s\t Prezime: %s\t Godina rodjenja:%d\t",p->ime,p->prezime,p->godina\_rodjenja);

p=p->Next;

}

}

void DodajNaKraj(Pos p)

{

Pos last=FindLast(p);

return Dodaj(last);

// sastoji se od dvi fje a to su nađi zadnji i fje dodaj na početak(nama je početak zapravo taj kraj)

}

Pos FindLast(Pos p)

{

while(p->Next!=NULL)

p=p->Next;

return p;

}

Pos Find(Pos p,char\* prezime)

{

while(p!=NULL)

{

if(strcmp(p->prezime,prezime)==0)

return p;

else p=p->Next;

}

return NULL;

}

Pos FindPrevious(Pos p,char\* prezime)

{

Pos prev=p,curr=p->Next;

while(curr->Next!=NULL)

{

if(strcmp(curr->prezime,prezime)==0)

return prev;

else

{

prev=curr;

curr=curr->Next;

}

}

return prev;

}

int Izbrisi(Pos p,char\* prezime)

{

Pos temp=p;

Pos prev=FindPrevious(p,prezime);

if(prev==NULL)

{

printf("Nema tog elementa!\n");

return -1;

}

else

{

temp=prev->Next;

prev->Next=prev->Next->Next;

free(temp);

}

return 0;

}

void InsertNext(Pos p,char\* prezime)

{

Pos q=NULL;

Pos x=Find(p,prezime);

Pos y=x->Next;

q=CreateNode();

if(q==NULL)

printf("Nije se alociralo!\n");

printf("Unesite ime,prezime i godinu rodjenja:\n");

scanf(" %s %s %d",&q->ime,&q->prezime,&q->godina\_rodjenja);

x->Next=q;

q->Next=y;

}

void InsertPrevious(Pos p,char\* prezime)

{

Pos prev=FindPrevious(p,prezime);

Pos q=CreateNode();

if(q==NULL)

printf("Nije se alociralo!\n");

printf("Unesite ime,prezime i godinu rodjenja:\n");

scanf(" %s %s %d",&q->ime,&q->prezime,&q->godina\_rodjenja);

q->Next=prev->Next;

prev->Next=q;

}

void SortPoPrezimenu(Pos p)

{

Pos prev, curr, between, end;

end = NULL;

while(p->Next != end)

{

prev = p;

between = prev->Next;

curr = between->Next;

while(curr != end)

{

if(\_strcmpi(between->prezime, curr->prezime) > 0)

{

between->Next = curr->Next;

curr->Next = between;

prev->Next = curr;

}

else

prev = between;

between = curr;

curr = curr->Next;

}

end = between;

}

}

void PrintInFile(Pos p,char\* imedatoteke)

{

FILE \*fp;

fp=fopen(imedatoteke,"w");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

exit(1);

}

p=p->Next;

while(p!=NULL)

{

fprintf(fp,"Ime: %s\t Prezime:%s\t Godina rodjenja:%d\n",p->ime,p->prezime,p->godina\_rodjenja);

p=p->Next;

}

fclose(fp);

}

void ReadFromFile(Pos p,char\* imedatoteke)

{

FILE \*fp;

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

exit(1);

}

while(!feof(fp))

{

Pos q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

exit(1);

}

fscanf(fp,"%s %s %d",&q->ime,&q->prezime,&q->godina\_rodjenja);

q->Next=p->Next;

p->Next=q;

p=q;

}

}

void Stars()

{

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*");

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*");

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*");

puts("\* \* \* \* \* \* \* \* \* \* MENU PROGRAMA\* \* \* \* \* \* \* \* \* \* ");

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*");

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*");

puts("\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\n");

}

void Meni()

{

puts("\t\t\tOPCIJE:\n");

puts(" a) dinamicki dodaje novi element na pocetak liste");

puts(" b) ispisuje listu");

puts(" c) dinamicki dodaje novi element na kraj liste");

puts(" d) pronalazi element u listi (po prezimenu)");

puts(" e) brise određeni element iz liste");

puts(" f) dinamicki dodaje novi element iza odredenog elementa");

puts(" g) dinamicki dodaje novi element ispred odredenog elementa");

puts(" h) sortira listu po prezimenima osoba");

puts(" i) upisuje listu u datoteku");

puts(" j) cita listu iz datoteke");

puts(" k) KRAJ programa\n");

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<math.h>

#include<ctype.h>

struct Poli;

typedef struct Poli \* Pos;

struct Poli{

int koef;

int eks;

Pos Next;

};

Pos CreateNode();

int ReadPoli(Pos,char\*);

void Print(Pos);

int ZbrojPoli(Pos,Pos,Pos);

/\*4. Napisati program za zbrajanje i množenje polinoma. Koeficijenti i eksponenti se

čitaju iz datoteke.

Napomena: Eksponenti u datoteci nisu nužno sortirani.\*/

Bez mnozenja napisano

// triba mi fja koja zbraja polinome, koja mnozi polinome i jedna za ispis polinoma , te fja koja će čitat polinome iz datoteke i spremat ih u listu

void main()

{

char p1dat[30]="",p2dat[30]="";

Pos p1=NULL,p2=NULL,z=NULL;

p1=CreateNode();

p2=CreateNode();

z=CreateNode();

p1->Next=NULL;

p2->Next=NULL;

z->Next=NULL;

printf("Unesite ime datoteke iz koje zelite procitati 1.polinom: ");

scanf("%s",p1dat);

ReadPoli(p1,p1dat);

printf("Unesite ime datoteke iz koje zelite procitati 2.polinom: ");

scanf("%s",p2dat);

ReadPoli(p2,p2dat);

Print(p1);

puts("\n");

Print(p2);

printf("\nZbrajanje polinoma: \n");

ZbrojPoli(p1,p2,z);

Print(z);

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(Poli));

if(NULL==q)

return NULL;

else return q;

}

int ReadPoli(Pos p,char\* imedatoteke)

{

Pos q=NULL,temp=NULL;

FILE \*fp;

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

while(!feof(fp))

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

fscanf(fp,"%d %d",&q->koef,&q->eks);

temp=p;

while(temp->Next!=NULL && temp->Next->eks > q->eks)

temp=temp->Next;

if(temp->Next!=NULL && temp->Next->eks==q->eks)

{

temp->Next->koef=temp->Next->koef+q->koef;

free(q);

if(temp->Next->koef==0)

{

q=temp->Next;

temp->Next=temp->Next->Next;

free(q);

}

}

else

{

q->Next=temp->Next;

temp->Next=q;

}

}

fclose(fp);

return 0;

}

void Print(Pos p)

{

if(p->Next==NULL)

printf("Nema polinoma!\n");

else

{

p=p->Next;

while(p!=NULL)

{

printf("+(%d)\*x^%d ",p->koef,p->eks);

p=p->Next;

}

}

}

int ZbrojPoli(Pos p1,Pos p2,Pos z)

{

Pos temp=NULL;

Pos q=NULL;

p1=p1->Next;

p2=p2->Next;

while(p1!=NULL && p2!=NULL)

{

if(p1->eks == p2->eks)

{

q=CreateNode();

if(q==NULL)

return -1;

q->koef=p1->koef+p2->koef;

q->eks=p1->eks;

p1=p1->Next;

p2=p2->Next;

}

else if(p1->eks > p2->eks)

{

q=CreateNode();

if(q==NULL)

return -1;

q->koef=p1->koef;

q->eks=p1->eks;

p1=p1->Next;

}

else if(p1->eks < p2->eks)

{

q=CreateNode();

if(q==NULL)

return -1;

q->koef=p2->koef;

q->eks=p2->eks;

p2=p2->Next;

}

if(q->koef==0)

free(q);

else

{

q->Next=z->Next;

z->Next=q;

z=q;

}

}

if(p1!=NULL)

temp=p1;

else temp=p2;

while(temp!=NULL)

{

q=CreateNode();

if(q==NULL)

return -1;

q->koef=temp->koef;

q->eks=temp->eks;

temp=temp->Next;

q->Next=z->Next;

z->Next=q;

z=q;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<math.h>

#include<ctype.h>

**Bez presjeka i unije**

/\*5. Za dvije sortirane liste L1 i L2 (mogu se pročitati iz datoteke ili unijeti ručno, bitno je

samo da su sortirane), napisati program koji stvara novu vezanu listu tako da računa:

a) L1∪L2,

b) L1∩L2.

Liste osim pokazivača na slijedeću strukturu imaju i jedan cjelobrojni element, po

kojem su sortirane.\*/

struct lista;

typedef struct lista \* Pos;

struct lista

{

int elem;

Pos Next;

};

Pos CreateNode();

int UnosRucno(Pos);

int ReadFromFile(Pos);

void Print(Pos);

void main()

{

Pos p1=NULL,p2=NULL,unija=NULL,presjek=NULL;

char odabir;

p1=CreateNode();

p1->Next=NULL;

p2=CreateNode();

p2->Next=NULL;

unija=CreateNode();

unija->Next=NULL;

presjek=CreateNode();

presjek->Next=NULL;

printf("Ako želite unijeti listu ručno odaberite 1,a ako želite unijeti listu iz datoteke odaberite 2!\n");

scanf(" %c",&odabir);

switch(odabir)

{

case '1':

UnosRucno(p1);

UnosRucno(p2);

break;

case '2':

ReadFromFile(p1);

ReadFromFile(p2);

break;

default:

printf("Krivo ste unijeli!");

exit(1);

break;

}

Print(p1);

puts("");

Print(p2);

}

Pos CreateNode()

{

Pos p=NULL;

p=(Pos)malloc(sizeof(lista));

if(p==NULL)

return NULL;

return p;

}

int UnosRucno(Pos p)

{

int elem=0,i=0;

Pos q=NULL,temp=NULL;

printf("Koliko elemenata zelite unijeti u listu: ");

scanf("%d",&elem);

while(i!=elem)

{

q=CreateNode();

if(q==NULL)

return -1;

scanf("%d",&q->elem);

temp=p;

while(temp->Next!=NULL && temp->Next->elem >= q->elem)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

i++;

}

return 0;

}

int ReadFromFile(Pos p)

{

Pos q=NULL,temp=NULL;

char imedatoteke[30]="";

FILE \*fp;

printf("Unesite ime datoteke: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

while(!feof(fp))

{

q=CreateNode();

if(q==NULL)

{

printf("Memorija se nije alocirala!");

return -1;

}

fscanf(fp,"%d",&q->elem);

temp=p;

while(temp->Next!=NULL && temp->Next->elem >= q->elem)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

}

fclose(fp);

return 0;

}

void Print(Pos p)

{

if(p->Next==NULL)

printf("Lista je prazna!\n");

p=p->Next;

while(p!=NULL)

{

printf("\t%d",p->elem);

p=p->Next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#include<conio.h>

#include<time.h>

#define RANGEMIN 10

#define RANGEMAX 100

#define RANGE 90

struct Lista;

typedef struct Lista \* Pos;

struct Lista{

int elem;

Pos Next;

};

/\*6. Napisati program koji pomoću vezanih listi simulira rad:

a) stoga,

b) reda.

Napomena: Funkcija "push" sprema cijeli broj, slučajno generirani u opsegu od 10 -100.\*/

Pos CreateNode();

int PushP(Pos);

int PushK(Pos);

int PopP(Pos);

void Print(Pos);

void main()

{

char odabir=0;

Pos p=NULL;

int i=1;

srand((unsigned int)time(NULL));

p=CreateNode();

p->Next=NULL;

printf("Zelite li stog odaberite 1, a za red 2\n");

scanf(" %c",&odabir);

switch (odabir)

{

case '1':

while(i)

{

printf("Zelite li push(3) ili pop(4):\n");

scanf(" %c",&odabir);

switch (odabir)

{

case '3':

printf("Stavili ste element %d\n",PushP(p));

Print(p);

break;

case '4':

printf("Skinili ste element %d\n",PopP(p));

Print(p);

break;

default:

printf("Krivo ste unijeli!\n");

i--;

break;

}

}

break;

case '2':

while(i)

{

printf("\nŽelite li push(3) ili pop(4):\n");

scanf(" %c",&odabir);

switch (odabir)

{

case '3':

printf("Stavili ste element %d\n",PushK(p));

Print(p);

break;

case '4':

printf("Skinili ste element %d\n",PopP(p));

Print(p);

break;

default:

printf("Krivo ste unijeli!\n");

i--;

break;

}

}

break;

default:

printf("Krivo ste unijeli!\n");

break;

}

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(Lista));

if(q==NULL)

return NULL;

else return q;

}

int PushP(Pos p)

{

Pos q=NULL;

q=CreateNode();

if(q==NULL)

return -1;

q->elem=rand()%(RANGE+1);

q->elem=q->elem+RANGEMIN;

q->Next=p->Next;

p->Next=q;

return q->elem;

}

int PushK(Pos p) // ili napišem fju find last i pozoven nju i PushP

{

/\*while(p->Next!=NULL)

p=p->Next;

return PushP(p); \*/

Pos q=NULL;

q=CreateNode();

if(q==NULL)

return -1;

q->elem=(rand()%RANGE)+RANGEMIN;

while(p->Next!=NULL)

p=p->Next;

q->Next=p->Next;

p->Next=q;

return q->elem;

}

int PopP(Pos p)

{

int pop=0;

Pos temp=NULL;

if(p->Next==NULL)

{

printf("Lista je prazna!\n");

return -1;

}

temp=p->Next;

pop=temp->elem;

p->Next=temp->Next;

free(temp);

return pop;

}

void Print(Pos p)

{

if(p->Next==NULL)

printf("Prazno!\n");

p=p->Next;

while(p!=NULL)

{

printf("%d\n",p->elem);

p=p->Next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#include<conio.h>

#include<time.h>

struct Lista;

typedef struct Lista \* Pos;

struct Lista{

int elem;

Pos Next;

};

Pos CreateNode();

int PushP(Pos,int);

int PopP(Pos);

int Calculate(char,int,int);

/\*7. Napisati program koji iz datoteke čita postfiks izraz i zatim korištenjem stoga računa

rezultat. Stog je potrebno realizirati preko vezane liste.\*/

int main()

{

FILE \*fp;

char imedatoteke[30]="";

Pos p=NULL;

p=CreateNode();

if(p==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

p->Next=NULL;

printf("Unesite ime datoteke iz koje citate postifx izraz: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

else

{

char buffer[4];

while(!feof(fp))

{

fscanf(fp," %s",buffer);

if(isdigit(buffer[0]))

{

PushP(p,atoi(buffer));

}

else

{

int x=0,y=0,r=0;

x=PopP(p);

y=PopP(p);

r=Calculate(buffer[0],x,y);

PushP(p,r);

}

}

fclose(fp);

}

printf("Rezultat je %d",p->Next->elem);

free(p->Next);

free(p);

return 0;

}

Pos CreateNode()

{

Pos q=(Pos)malloc(sizeof(Lista));

if(q==NULL)

return NULL;

else return q;

}

int PushP(Pos p,int x)

{

Pos q=CreateNode();

if(q==NULL)

return -1;

q->elem=x;

q->Next=p->Next;

p->Next=q;

return 0;

}

int PopP(Pos p)

{

int x=0;

Pos temp=NULL;

temp=p->Next;

x=temp->elem;

p->Next=p->Next->Next;

free(temp);

return x;

}

int Calculate(char oper,int x,int y)

{

int result=0;

switch(oper)

{

case '+':

result=x+y;

break;

case '-':

result=x-y;

break;

case '\*':

result=x\*y;

break;

case '/':

if(y!=0)

result=x/y;

break;

default:

break;

}

return result;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#include<conio.h>

#include<time.h>

/\*9. Napisati program koji omogućava rad s binarnim stablom pretraživanja. Treba omogućiti unošenje novog elementa u stablo, ispis elemenata, brisanje i pronalaženje nekog elementa.\*/

struct Tree;

typedef struct Tree \* Stablo;

struct Tree

{

int elem;

Stablo L;

Stablo D;

};

void Meni();

Stablo CreateStablo();

Stablo Insert(Stablo,int);

void PrintInorder(Stablo);

void PrintPostorder(Stablo);

void PrintPreorder(Stablo);

Stablo Find(Stablo,int);

Stablo FindMax(Stablo);

Stablo FindMin(Stablo);

Stablo Delete(Stablo,int);

void main()

{

Stablo root=NULL,trazi=NULL;

char izbor = 0;

int n = 0;

while(izbor != 'k' && izbor != 'K')

{

Meni();

scanf(" %c", &izbor);

switch(izbor)

{

case '1':

printf("\r\nUnesite broj koji zelite unijeti : ");

scanf(" %d",&n);

root = Insert(root,n);

break;

case '2':

printf("\nUnesite broj koji zelite izbrisati : ");

scanf(" %d",&n);

root = Delete(root,n);

break;

case '3':

printf("\nUnesite broj koji trazite : ");

scanf(" %d",&n);

trazi = Find(root,n);

if(trazi != NULL)

printf("\nTrazeni element %d se nalazi na adresi %x", trazi->elem,trazi);

else

printf("\nTrazeni broj %d nije pronaden.",n);

break;

case '4':

puts("");

if(root == NULL)

printf("\nStablo je prazno");

else

PrintInorder(root);

break;

case 'k':

case 'K':

break;

default:

printf("\nPogresan izbor!\nPokusajte ponovno.\n");

}

}

}

void Meni()

{

printf("\n\* \* \* \* \* \* \* \* BINARNA STABLA ZA PRETRAZIVANJE \* \* \* \* \* \* \* \* \*\n\n");

printf("ODABERITE BROJ POKRAJ PONUĐENOGA:\n");

printf(" 1) unos elementa u stablo \n");

printf(" 2) brisanje elementa iz stabla \n");

printf(" 3) pronalazak elementa u stablu \n");

printf(" 4) ispis stabla \n");

printf(" K) kraj\n");

}

Stablo CreateStablo()

{

Stablo p=NULL;

p=(Stablo)malloc(sizeof(Tree));

if(p==NULL)

return NULL;

else return p;

}

Stablo Insert(Stablo s,int x)

{

if(s==NULL)

{

s=CreateStablo();

s->elem=x;

s->L=NULL;

s->D=NULL;

}

else

{

if(x < s->elem)

s->L=Insert(s->L,x);

else if(x > s->elem)

s->D=Insert(s->D,x);

}

return s;

}

void PrintInorder(Stablo s)

{

if(s!=NULL)

{

PrintInorder(s->L);

printf("%d\n",s->elem);

PrintInorder(s->D);

}

}

void PrintPostorder(Stablo s)

{

if(s!=NULL)

{

PrintPostorder(s->L);

PrintPostorder(s->D);

printf("%d\n",s->elem);

}

}

void PrintPreorder(Stablo s)

{

if(s!=NULL)

{

printf("%d\n",s->elem);

PrintPreorder(s->L);

PrintPreorder(s->D);

}

}

Stablo Find(Stablo s,int x)

{

if(s==NULL)

return NULL;

else

{

if(x < s->elem)

return Find(s->L,x);

else if(x > s->elem)

return Find(s->D,x);

else

return s;

}

}

Stablo FindMax(Stablo s)

{

while(s->D!=NULL)

s=s->D;

return s;

}

Stablo FindMin(Stablo s)

{

while(s->L!=NULL)

s=s->L;

return s;

}

Stablo Delete(Stablo s,int x)

{

Stablo temp=NULL;

temp=CreateStablo();

if(s==NULL)

printf("Nema ga!\n");

else if(x < s->elem)

s->L=Delete(s->L,x);

else if(x > s->elem)

s->D=Delete(s->D,x);

else if(s->L!=NULL && s->D!=NULL)

{

temp=FindMin(s->D);

s->elem=temp->elem;

s->D=Delete(s->D,s->elem);

}

else

{

temp=s;

if(s->L==NULL)

s=s->D;

else

s=s->L;

free(temp);

}

return s;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*

10. Napisati program koji iz datoteke čita postfiks izraz i zatim stvara stablo proračuna. Iz

gotovog stabla proračuna upisati u datoteku infiks izraz.

\*/

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<string.h>

#include<ctype.h>

struct \_Node;

struct \_Stack;

typedef struct \_Node \*NPos;

typedef struct \_Stack \*SPos;

struct \_Node

{

char El[5];

NPos Right;

NPos Left;

};

struct \_Stack

{

NPos NodeElement;

SPos Next;

};

void Push(SPos, NPos);

NPos Pop(SPos);

void InorderPrint(NPos);

void PreorderPrint(NPos);

void PostorderPrint(NPos);

int main(void)

{

FILE \*stream;

char \*ime\_dat;

SPos PosfixStack;

NPos root;

PosfixStack = (SPos)malloc(sizeof(struct \_Stack));

if(PosfixStack == NULL)

return -1;

PosfixStack->Next = NULL;

PosfixStack->NodeElement = NULL;

ime\_dat = (char\*)malloc(sizeof(char)\*1024);

if(ime\_dat == NULL)

return -2;

printf("\r\nUnesite ime datoteke za citanje : \r\n");

scanf\_s(" %s", ime\_dat, 1024);

fopen\_s(&stream, ime\_dat, "r");

if(stream == NULL)

{

printf("\r\nDatoteka %s nije otvorena!", ime\_dat);

return -3;

}

else

{

while(!feof(stream))

{

NPos q = (NPos)malloc(sizeof(struct \_Node));

if(q)

{

//q->El = (char\*)malloc(sizeof(char)\*5);

q->El[4] = '\0';

fscanf\_s(stream, " %s", q->El, 4);

q->Left = NULL;

q->Right = NULL;

if(atoi(q->El) || isalpha(q->El[0]))

{

//Numerička vrijednost trebamo spremiti na stog

Push(PosfixStack, q);

}

else

{

//Operacija je trebamo dobaviti operande sa stoga, povezati ih i ponovno spremiti na stog

q->Right = Pop(PosfixStack);

q->Left = Pop(PosfixStack);

Push(PosfixStack, q);

}

}

}

fclose(stream);

}

root = Pop(PosfixStack);

printf("\r\n\r\n\* Preorder ispis :\t");

PreorderPrint(root);

printf("\r\n\r\n\* Inorder ispis :\t");

InorderPrint(root->Left);

printf(" %s ", root->El);

InorderPrint(root->Right);

printf("\r\n\r\n\* Postorder ispis :\t");

PostorderPrint(root);

printf("\r\n\r\n");

free(PosfixStack);

return 0;

}

void Push(SPos S, NPos N)

{

SPos q;

q = (SPos)malloc(sizeof(struct \_Stack));

if(q)

{

q->NodeElement = N;

q->Next = S->Next;

S->Next = q;

}

}

NPos Pop(SPos S)

{

SPos tmp;

NPos q;

tmp = S->Next;

S->Next = tmp->Next;

q = tmp->NodeElement;

free(tmp);

return q;

}

void InorderPrint(NPos N)

{

if(N != NULL)

{

if(atoi(N->El) || isalpha(N->El[0]))

;

else

printf("(");

InorderPrint(N->Left);

printf(" %s ", N->El);

InorderPrint(N->Right);

if(atoi(N->El) || isalpha(N->El[0]))

;

else

printf(")");

}

}

void PostorderPrint(NPos N)

{

if(N != NULL)

{

PostorderPrint(N->Left);

PostorderPrint(N->Right);

printf(" %s", N->El);

}

}

void PreorderPrint(NPos N)

{

if(N != NULL)

{

printf(" %s", N->El);

PreorderPrint(N->Left);

PreorderPrint(N->Right);

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Napisati program koji čita iz datoteke dio teksta, te kreira binarno stablo pretraživanja. Binarno stablo se kreira na način da se u njega upisuju sve različite riječi iz teksta koji je zapisan u datoteci s pripadajućim

duljinama unesene riječi.

struct \_Cvor{char \*rijec; int length; struct \_Cvor \*Left; struct \_Cvor \*Right;};

Potrebno je izračunati prosječnu duljinu svake riječi zapisane u binarnom stablu.

Napisati program koji iz tako kreiranog binarnog stabla u vezanu listu puni samo one riječi koje su dulje od prosječne duljine svih riječi, te ispisati tu vezanu listu. \*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h.>

#include <string.h>

struct \_Cvor;

typedef struct \_Cvor \* Pos;

struct Lista;

typedef struct Lista \* Position;

struct \_Cvor{

char \* rijec;

int length;

Pos Left;

Pos Right;

}cvor;

struct Lista{

char \* word;

Position Next;

}lista;

Pos CreateCvor();

Pos DodajRijec(Pos,char\*,int);

void PrintList(Position);

Pos CitajIDodaj(Pos,float,Position);

void PrintTree(Pos);

int main()

{

FILE \*fp=NULL;

char \* imedatoteke="";

Pos root=NULL;

Position p=NULL;

float prosjecna\_duljina=0.0;

int brojac=0,ukupna\_duljina=0;

char \*rijec="";

p=(Position)malloc(sizeof(lista));

if(p==NULL)

return -3;

p->Next=NULL;

imedatoteke=(char\*)malloc(sizeof(char)\*30);

if(imedatoteke==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

printf("Unesite ime datoteke iz koje zelite citati rijeci:\n");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -2;

}

else

{

while(!feof(fp))

{

rijec=(char\*)malloc(sizeof(char)\*20);

fscanf(fp," %s",rijec);

brojac++;

ukupna\_duljina += strlen(rijec);

root=DodajRijec(root,rijec,strlen(rijec));

}

fclose(fp);

}

prosjecna\_duljina=(float)ukupna\_duljina/brojac;

printf("Prosjecna duljina rijeci je %f,a ima ukupno %d rijeci\n",prosjecna\_duljina,brojac);

PrintTree(root);

CitajIDodaj(root,prosjecna\_duljina,p);

printf("\n\n\n");

PrintList(p);

}

Pos CreateCvor()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(cvor));

if(q==NULL)

return NULL;

else return q;

}

Pos DodajRijec(Pos s,char \* rijeci,int duljina)

{

if(s==NULL)

{

s=CreateCvor();

if(s==NULL)

printf("Nije se alociralo!\n");

s->Left=NULL;

s->Right=NULL;

s->rijec=rijeci;

s->length=duljina;

}

else

{

if(strcmp(rijeci,s->rijec)<0)

{

s->Left=DodajRijec(s->Left,rijeci,duljina);

}

else if(strcmp(rijeci,s->rijec)>0)

{

s->Right=DodajRijec(s->Right,rijeci,duljina);

}

}

return s;

}

void PrintList(Position p)

{

if(p->Next==NULL)

{

printf("Lista je prazna\n");

}

p=p->Next;

while(p!=NULL)

{

printf("%s\t",p->word);

p=p->Next;

}

}

Pos CitajIDodaj(Pos p,float prosjecna,Position pos)

{

// napravimo varijantu fje trazi za bin.stablo koja ce nam vratit sve one koji imaju duljinu rijeci veću od prosjecne tj. spremit ce ih u listu

Position q=NULL;

if(p!=NULL)

{

CitajIDodaj(p->Left,prosjecna,pos);

if(p->length > prosjecna)

{

q=(Position)malloc(sizeof(lista));

if(q==NULL)

return NULL;

q->word=(char\*)malloc(sizeof(char)\*(p->length+1));

q->word=p->rijec;

q->Next=pos->Next;

pos->Next=q;

}

CitajIDodaj(p->Right,prosjecna,pos);

}

return 0;

}

void PrintTree(Pos p)

{

if(p!=NULL)

{

PrintTree(p->Left);

printf("%s\t",p->rijec);

PrintTree(p->Right);

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

/\*Napisati program koji iz datoteke čita dio teksta, riječ po riječ i unosi u binarno stablo pretraživanja

(maksimalna duljina riječi 1024 znaka). Ispisati binarno stablo na „inorder“ način.

Iz binarnog stabla kreirati dvije datoteke, u prvoj se nalaze sve riječi koje počinju samoglasnikom, a u

drugoj sve riječi koje počinju suglasnikom. Riječi zapisane u datotekama moraju biti sortirane po abecedi\*/

struct Stablo;

typedef struct Stablo\* Pos;

struct Lista;

typedef struct Lista\* Poss;

struct Stablo

{

char \* rijec;

Pos L;

Pos R;

}stablo;

struct Lista

{

char \*rijec;

Poss Next;

}lista;

Pos ReadFromFile(char\*,Pos);

Pos Insert(Pos, char\*);

void Print(Pos);

Poss CreateNode();

void DivideInList1(Pos,Poss);

void DivideInList2(Pos,Poss);

int InsertSort(Poss,Pos);

int WriteToFile(Poss);

void main()

{

Pos root=NULL;

Poss lista1=NULL,lista2=NULL;

char \* filename="";

lista1=CreateNode();

if(lista1==NULL)

exit(1);

lista2=CreateNode();

if(lista2==NULL)

exit(1);

lista1->Next=NULL;

lista2->Next=NULL;

filename=(char\*)malloc(sizeof(char)\*30);

printf("Unesite ime datoteke: ");

scanf(" %s",filename);

root=ReadFromFile(filename,root);

Print(root);

DivideInList1(root,lista1);

DivideInList2(root,lista2);

WriteToFile(lista1);

WriteToFile(lista2);

}

Pos ReadFromFile(char \* filename,Pos s)

{

FILE \*fp=NULL;

char \* rijec="";

fp=fopen(filename,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return NULL;

}

else

{

while(!feof(fp))

{

rijec=(char\*)malloc(sizeof(char)\*1024);

fscanf(fp," %s",rijec);

s=Insert(s,rijec);

}

fclose(fp);

}

return s;

}

Pos Insert(Pos s,char \* rijec)

{

if(s==NULL)

{

s=(Pos)malloc(sizeof(stablo));

if(s==NULL)

{

printf("Nije se alocirala memorija!\n");

return 0;

}

s->rijec=(char\*)malloc(sizeof(char)\*1024);

s->rijec=rijec;

s->L=NULL;

s->R=NULL;

}

else if(strcmp(rijec,s->rijec)>0)

{

s->R=Insert(s->R,rijec);

}

else if(strcmp(rijec,s->rijec)<0)

{

s->L=Insert(s->L,rijec);

}

return s;

}

void Print(Pos s)

{

if(s!=NULL)

{

Print(s->L);

printf(" %s\t",s->rijec);

Print(s->R);

}

}

Poss CreateNode()

{

Poss q=NULL;

q=(Poss)malloc(sizeof(lista));

if(q==NULL)

return NULL;

else return q;

}

int InsertSort(Poss p,Pos s)

{

Poss temp=NULL,q=NULL;

q=CreateNode();

if(q==NULL)

exit(1);

else

{

q->rijec=(char\*)malloc(sizeof(char)\*1024);

q->rijec=s->rijec;

temp=p;

while(temp->Next!=NULL && strcmp(temp->Next->rijec,q->rijec)<0)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

}

return 0;

}

void DivideInList1(Pos s,Poss s1)

{

if(s!=NULL)

{

DivideInList1(s->L,s1);

if(tolower(s->rijec[0])=='a' || tolower(s->rijec[0])=='e' || tolower(s->rijec[0])=='i' || tolower(s->rijec[0])=='o' || tolower(s->rijec[0])=='u')

{

InsertSort(s1,s);

}

DivideInList1(s->R,s1);

}

}

void DivideInList2(Pos s,Poss s2)

{

if(s!=NULL)

{

DivideInList2(s->L,s2);

if(tolower(s->rijec[0])!='a' && tolower(s->rijec[0])!='e' && tolower(s->rijec[0])!='i' && tolower(s->rijec[0])!='o' && tolower(s->rijec[0])!='u')

{

InsertSort(s2,s);

}

DivideInList2(s->R,s2);

}

}

int WriteToFile(Poss p)

{

char filename[30]="";

FILE \*fp;

printf("\nUnesite ime datoteke:(prvi/drugi.txt): ");

scanf(" %s",filename);

fp=fopen(filename,"w");

if(fp==NULL)

return -1;

else

{

p=p->Next;

while(p!=NULL)

{

fprintf(fp,"%s \t",p->rijec);

p=p->Next;

}

fclose(fp);

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Napisati program koji iz datoteke čita podatke o broju bodova koje je jedan student dobio na ispitu

(struct \_student{char ime[128];char prezime[128];int broj\_bodova; int relativni\_broj\_bodova;struct

\_student \*left; ;struct \_student right;}) i sprema ih u binarno stablo pretraživanja.

(Napomena : 0 <broj\_bodova <= 100). Ispisati binarno stablo na „inorder“ način.

U tako kreiranom binarnom stablu potrebno je za svakog od studenata izračunati i relativan broj bodova

po formuli: relativan\_broj\_bodova=( broj\_bodova / MAX(broj\_bodova) ). Ispisati binarno stablo na

„preorder“ način.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#include<conio.h>

struct \_student;

typedef struct \_student \*Pos;

struct \_student

{

char ime[128];

char prezime[128];

int broj\_bodova;

int relativni\_broj\_bodova;

Pos left;

Pos right;

}student;

Pos ReadFromFile(Pos);

Pos Insert(Pos,int,char\*,char\*);

void Inorder(Pos);

void Preorder(Pos);

int Maximum(Pos);

void RelativniBrojBodova(Pos,int);

void main()

{

Pos root=NULL;

int max=0;

root=ReadFromFile(root);

if(root==NULL)

{

printf("Nema elemenata ili nije dobro otvorena datoteka!\n");

exit(1);

}

printf("\n\tInorder stablo:\n");

Inorder(root);

max=Maximum(root);

printf("\nMaximalan broj bodova koji je netko dobio je %d.\n",max);

RelativniBrojBodova(root,max);

printf("\n\tPreorder stablo(s relativnim brojem bodova):\n");

Preorder(root);

}

Pos ReadFromFile(Pos s)

{

FILE \*fp=NULL;

char ime[128]="",prezime[128]="";

char imedatoteke[30]="";

int brojbodova=0;

printf("Unesite ime datoteke: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

return NULL;

else

{

while(!feof(fp))

{

fscanf(fp," %s %s %d",ime,prezime,&brojbodova);

if(brojbodova >= 0 && brojbodova<=100)

s=Insert(s,brojbodova,ime,prezime);

else printf("Nije moguc taj broj bodova!\n");

}

fclose(fp);

}

return s;

}

Pos Insert(Pos p,int bodovi,char \* ime,char \* prezime)

{

if(p==NULL)

{

p=(Pos)malloc(sizeof(student));

if(p==NULL)

return NULL;

p->left=NULL;

p->right=NULL;

strcpy(p->ime,ime);

strcpy(p->prezime,prezime);

p->broj\_bodova=bodovi;

p->relativni\_broj\_bodova=0; // moze i bez toga

}

else if(p->broj\_bodova < bodovi)

{

p->right=Insert(p->right,bodovi,ime,prezime);

}

else if(p->broj\_bodova > bodovi)

{

p->left=Insert(p->left,bodovi,ime,prezime);

}

return p;

}

void Inorder(Pos s)

{

if(s!=NULL)

{

Inorder(s->left);

printf(" %s %s %d\n",s->ime,s->prezime,s->broj\_bodova);

Inorder(s->right);

}

}

void Preorder(Pos s)

{

if(s!=NULL)

{

printf(" %s %s %d %d\n",s->ime,s->prezime,s->broj\_bodova,s->relativni\_broj\_bodova);

Preorder(s->left);

Preorder(s->right);

}

}

int Maximum(Pos s)

{

int max=0;

while(s->right!=NULL)

{

s=s->right;

}

max=s->broj\_bodova;

return max;

}

void RelativniBrojBodova(Pos s,int max)

{

if(s!=NULL)

{

s->relativni\_broj\_bodova=((s->broj\_bodova\*100 )/max);

RelativniBrojBodova(s->left,max);

RelativniBrojBodova(s->right,max);

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<conio.h>

#include<string.h>

#include<time.h>

/\*Napisati program koji iz datoteke čita dio teksta, riječ po riječ i unosi ih u red s prioritetom (maksimalnaduljina riječi 1024 znaka,

prioritet=1 => riječ počinje samoglasnikom, prioritet=2 => riječ počinje suglasnikom;). Riječi u redu moraju biti sortirane po prioritetu.

Iz tako kreiranog reda kreirati dvije datoteke, u prvoj će se nalaziti riječi sa prioritetom=1, a u drugoj riječi

s prioritetom=2; s tim da su te riječi u datoteci zapisane sortirane po abecedi. \*/

struct Red;

typedef struct Red \* Pos;

struct Red

{

char rijec[1024];

int prioritet;

Pos Next;

}red;

Pos CreateNode();

int ReadFromFile(Pos);

int Insert(Pos,char\*);

int Prioritet(char\*);

void Print(Pos);

int Sort(Pos);

int Write(Pos);

void main()

{

Pos p;

p=CreateNode();

if(p==NULL)

{

printf("Memorija se nije alocirala!\n");

exit(1);

}

p->Next=NULL;

ReadFromFile(p);

Print(p);

Sort(p);

Write(p);

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(red));

if(q==NULL)

return NULL;

else return q;

}

int ReadFromFile(Pos p)

{

FILE \*fp=NULL;

char imedatoteke[30]="";

char rijec[1024]="";

printf("Unesite ime datoteke:");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

return -1;

else

{

while(!feof(fp))

{

fscanf(fp," %s",rijec);

Insert(p,rijec);

}

fclose(fp);

}

return 0;

}

int Insert(Pos p,char\* rijec)

{

Pos q=NULL,temp=NULL;

q=CreateNode();

if(q==NULL)

exit(1);

else

{

strcpy(q->rijec,rijec);

q->prioritet=Prioritet(rijec);

if(Prioritet(rijec)==1) // dodaj na kraj, skini s pocetka --> RED

{

temp=p;

while(temp->Next!=NULL && temp->Next->prioritet==1)

{

temp=temp->Next;

}

q->Next=temp->Next;

temp->Next=q;

}

else if(Prioritet(rijec)==2)

{

temp=p;

while(temp->Next!=NULL && temp->Next->prioritet==1)

temp=temp->Next;

while(temp->Next!=NULL && temp->Next->prioritet==2)

{

temp=temp->Next;

}

q->Next=temp->Next;

temp->Next=q;

}

}

return 0;

}

int Prioritet(char \* rijec)

{

if(tolower(rijec[0])=='a' || tolower(rijec[0])=='e' || tolower(rijec[0])=='i' || tolower(rijec[0])=='o' || tolower(rijec[0])=='u')

return 1;

else if(tolower(rijec[0])!='a' && tolower(rijec[0])!='e' && tolower(rijec[0])!='i' && tolower(rijec[0])!='o' && tolower(rijec[0])!='u')

return 2;

else return -1;

}

void Print(Pos p)

{

if(p->Next==NULL)

{

printf("Prazan red!\n");

}

else

{

p=p->Next;

while(p!=NULL)

{

printf(" %s %d \n",p->rijec,p->prioritet);

p=p->Next;

}

}

}

int Sort(Pos p)

{

Pos prev=NULL,curr=NULL,curr2=NULL,t=NULL,temp=NULL;

temp=p;

prev=temp;

curr=temp->Next;

curr2=temp->Next->Next;

while(curr2!=NULL)

{

if(curr2->prioritet==1)

{

if(strcmp(curr->rijec,curr2->rijec)>0)

{

t=curr2->Next;

prev->Next=curr2;

curr2->Next=curr;

curr->Next=t;

}

}

if(prev->prioritet==2 && curr2->prioritet==2)

{

if(strcmp(curr->rijec,curr2->rijec)>0)

{

t=curr2->Next;

prev->Next=curr2;

curr2->Next=curr;

curr->Next=t;

}

}

prev=curr;

curr=curr2;

curr2=curr2->Next;

}

return 0;

}

int Write(Pos p)

{

FILE \*fp=NULL;

FILE \*gp=NULL;

char dat1[30]="prvi.txt";

char dat2[30]="drugi.txt";

fp=fopen(dat1,"w");

if(fp==NULL)

return -1;

p=p->Next;

while(p->Next!=NULL && p->prioritet==1)

{

fprintf(fp," %s %d \n",p->rijec,p->prioritet);

p=p->Next;

}

fclose(fp);

gp=fopen(dat2,"w");

if(gp==NULL)

return -2;

while(p->Next!=NULL)

{

fprintf(gp," %s %d \n",p->rijec,p->prioritet);

p=p->Next;

}

fclose(gp);

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Napisati program koji čita iz datoteka podatke o studentima i ispitima, te kreira dvije vezane liste

struct \_Student{char \*sifra\_studenta; char \*ime; char \*prezime; char \*sifra\_ispita;struct \_Student

\*Next;};

struct \_Ispit {char \*sifra\_ispita; char \*naziv; struct \_Ispit \*Next;};

Potrebno je ispisati koliko je studenata položilo svaki pojedini ispit (po nazivu ispita), te nabrojati koji su

to studenti.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<string.h>

struct \_Student;

typedef struct \_Student \*Pos;

struct \_Ispit;

typedef struct \_Ispit \* Poss;

struct \_Student

{

char \*sifra\_studenta;

char \*ime;

char \*prezime;

char \*sifra\_ispita;

Pos Next;

}student;

struct \_Ispit {

char \*sifra\_ispita;

char \*naziv;

Poss Next;

}ispit;

Pos CreateNode();

Poss CreateNodes();

int ReadFromFile(Pos,Poss);

int PoloziliPrint(Pos,Poss);

void main()

{

Pos p=NULL;

Poss q=NULL;

int x=0;

p=CreateNode();

q=CreateNodes();

if(p==NULL || q==NULL)

{

printf("Nije se alociralo!\n");

exit(1);

}

p->Next=NULL;

q->Next=NULL;

x=ReadFromFile(p,q);

if(x!=0)

exit(1);

PoloziliPrint(p,q);

}

Pos CreateNode()

{

Pos p=NULL;

p=(Pos)malloc(sizeof(student));

if(p==NULL)

return NULL;

else return p;

}

Poss CreateNodes()

{

Poss q=NULL;

q=(Poss)malloc(sizeof(ispit));

if(q==NULL)

return NULL;

else return q;

}

int ReadFromFile(Pos p,Poss q)

{

char dat1[30]="",dat2[30]="";

Pos p1=NULL;

Poss q1=NULL;

FILE \*fp=NULL;

FILE \*qp=NULL;

printf("Unesite ime 1.datoteke u kojoj su studenti: ");

scanf(" %s",dat1);

printf("Unesite ime 2.datoteke u kojoj su ispiti: ");

scanf(" %s",dat2);

fp=fopen(dat1,"r");

qp=fopen(dat2,"r");

if(fp==NULL || qp==NULL)

{

printf("Datoteka nije otvorena!\n");

return -1;

}

else

{

while(!feof(fp))

{

p1=CreateNode();

if(p1==NULL)

return -2;

p1->sifra\_studenta=(char\*)malloc(sizeof(char)\*4);

p1->ime=(char\*)malloc(sizeof(char)\*10);

p1->prezime=(char\*)malloc(sizeof(char)\*10);

p1->sifra\_ispita=(char\*)malloc(sizeof(char)\*3);

fscanf(fp," %s %s %s %s",p1->sifra\_studenta,p1->ime,p1->prezime,p1->sifra\_ispita);

p1->Next=p->Next;

p->Next=p1;

}

fclose(fp);

while(!feof(qp))

{

q1=CreateNodes();

if(p1==NULL)

return -2;

q1->sifra\_ispita=(char\*)malloc(sizeof(char)\*3);

q1->naziv=(char\*)malloc(sizeof(char)\*5);

fscanf(qp," %s %s",q1->sifra\_ispita,q1->naziv);

q1->Next=q->Next;

q->Next=q1;

}

fclose(qp);

}

return 0;

}

int PoloziliPrint(Pos p,Poss q)

{

/\*Potrebno je ispisati koliko je studenata položilo svaki pojedini ispit (po nazivu ispita), te nabrojati koji su to studenti.\*/

// šetamo po listi ispita i gledamo nazive i pamtimo ih,kao i šifre ako su vece od 50 -> preko te šifre tražimo studenta i ispišemo ga

Poss temp=NULL;

Pos tempp=NULL;

p=p->Next;

q=q->Next;

tempp=p;

temp=q;

while(temp!=NULL)

{

if(atoi(temp->sifra\_ispita) > 49)

{

tempp=p;

while(tempp!=NULL)

{

if(strcmp(temp->sifra\_ispita,tempp->sifra\_ispita)==0)

{

printf(" %s %s %s %s\n",tempp->ime,tempp->prezime,temp->naziv,tempp->sifra\_ispita);

}

tempp=tempp->Next;

}

}

temp=temp->Next;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Napisati program koji čita iz datoteke dio teksta, te kreira binarno stablo pretraživanja.

Binarno stablo se kreira na način da se u njega upisuju sve različite riječi iz teksta koji je zapisan u datoteci s pripadajućim

duljinama unesene riječi. Potrebno je izračunati prosječnu duljinu svake riječi zapisane u binarnom stablu.

Napisati program koji iz tako kreiranog binarnog stabla u vezanu listu puni samo one riječi koje su dulje

od prosječne duljine svih riječi, te ispisati tu vezanu listu.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<math.h>

#include<string.h>

#include<time.h>

#include<ctype.h>

struct \_Cvor;

typedef struct \_Cvor \* Pos;

struct \_Node;

typedef struct \_Node \* Poss;

struct \_Cvor

{

char \*rijec;

int length;

Pos Left;

Pos Right;

}cvor;

struct \_Node

{

char \*rijec;

Poss Next;

}node;

// prototipi funkcija

Pos Insert(Pos,char \*,float \*);

Poss CreateNode();

void Dodaj(Poss,Pos,float \*);

void Print(Poss);

void main()

{

Pos root=NULL;

Poss head=NULL;

FILE \*fp=NULL;

char \* imedatoteke=NULL,\* rijec=NULL;

float \*prosjecna=NULL;

prosjecna=(float\*)malloc(sizeof(float)\*5);

head=CreateNode();

if(head==NULL)

printf("Nije se alociralo!\n");

head->Next=NULL;

imedatoteke=(char\*)malloc(sizeof(char)\*30);

printf("Unesite ime datoteke: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

}

while(!feof(fp))

{

rijec=(char\*)malloc(sizeof(char)\*30);

fscanf(fp," %s",rijec);

root=Insert(root,rijec,prosjecna);

}

fclose(fp);

printf("Prosjecna duljina rijeci u datoteci je %f.\n",\*prosjecna);

Dodaj(head,root,prosjecna);

Print(head->Next);

}

Pos Insert(Pos p,char \* rijec,float \*prosjecna)

{

static int i=0;

static float zbroj=0.0;

if(p==NULL)

{

p=(Pos)malloc(sizeof(cvor));

p->rijec=rijec;

p->length=strlen(rijec);

p->Left=NULL;

p->Right=NULL;

i++;

zbroj+=p->length;

\*prosjecna = zbroj / i;

}

else

{

if(strcmp(p->rijec,rijec) > 0)

p->Left=Insert(p->Left,rijec,prosjecna);

else if(strcmp(p->rijec,rijec) < 0)

p->Right=Insert(p->Right,rijec,prosjecna);

}

return p;

}

Poss CreateNode()

{

Poss q=NULL;

q=(Poss)malloc(sizeof(node));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

else return q;

}

void Dodaj(Poss q,Pos p,float\* prosjecna)

{

Poss w=NULL;

if(p!=NULL)

{

Dodaj(q,p->Left,prosjecna);

if(p->length > \*prosjecna)

{

w=CreateNode();

if(q==NULL)

printf("Nije se alociralo!\n");

w->rijec=(char\*)malloc(sizeof(char)\*(p->length+1));

w->rijec=p->rijec;

w->Next=q->Next;

q->Next=w;

}

Dodaj(q,p->Right,prosjecna);

}

}

void Print(Poss p)

{

while(p!=NULL)

{

printf(" %s\t",p->rijec);

p=p->Next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Napisati program koji iz datoteke čita dio teksta, riječ po riječ i unosi u binarno stablo pretraživanja

(maksimalna duljina riječi 1024 znaka). Ispisati binarno stablo na „inorder“ način.

Iz binarnog stabla kreirati dvije datoteke, u prvoj se nalaze sve riječi koje počinju samoglasnikom, a u

drugoj sve riječi koje počinju suglasnikom. Riječi zapisane u datotekama moraju biti sortirane po abecedi.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

struct Tree;

typedef struct Tree \* Pos;

struct Node;

typedef struct Node \* Poss;

struct Tree

{

char rijec[1024];

Pos Left;

Pos Right;

}tree;

struct Node

{

char rijec[1024];

Poss Next;

}node;

Pos ReadFromFile(Pos);

Pos Insert(char\*,Pos);

void PrintInorder(Pos);

void PrintToFiles(Pos);

Poss CreateNode();

void InsertToLists(Pos,Poss,Poss);

void PrintList(Poss,char\*);

// protipi funkcija

int main()

{

Pos root=NULL;

Poss head1=NULL,head2=NULL;

char \* samoglasnici="samoglasnici.txt",\* suglasnici="suglasnici.txt";

head1=CreateNode();

head2=CreateNode();

if(head1==NULL || head2==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head1->Next=NULL;

head2->Next=NULL;

root=ReadFromFile(root);

PrintInorder(root);

// PrintToFiles(root); RADI I OVO I OVO DRUGO IAKO JE OVO DRUGO DA SE STAVI U SORTIRANE LISTE BOLJE

InsertToLists(root,head1,head2);

PrintList(head1,samoglasnici);

PrintList(head2,suglasnici);

return 0;

}

Pos ReadFromFile(Pos p)

{

char imedatoteke[30]="";

FILE \*fp=NULL;

char \*rijec=NULL;

printf("Unesite ime datoteke iz koje zelite procitati rijeci: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return NULL;

}

else

{

while(!feof(fp))

{

rijec=(char\*)malloc(sizeof(char)\*20);

fscanf(fp," %s",rijec);

p=Insert(rijec,p);

}

fclose(fp);

}

return p;

}

Pos Insert(char\* rijec,Pos p)

{

if(p==NULL)

{

p=(Pos)malloc(sizeof(tree));

if(p==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

strcpy(p->rijec,rijec);

p->Left=NULL;

p->Right=NULL;

}

else if(strcmp(p->rijec,rijec) > 0)

p->Left=Insert(rijec,p->Left);

else if(strcmp(p->rijec,rijec) < 0)

p->Right=Insert(rijec,p->Right);

return p;

}

void PrintInorder(Pos p)

{

if(p!=NULL)

{

PrintInorder(p->Left);

printf(" %s\n",p->rijec);

PrintInorder(p->Right);

}

}

void PrintToFiles(Pos p)

{

// idu dvi datoteke jedna za one koje pocinju suglasnicima i jedna za one koe pocinju samoglasnicima i moraju bit ispisani u datoteku abecedno poredani --> preko inordera (OBAVEZNO "a" ZA FILE)

// ako ne onda sortiranje pri upisu u dvi liste i te liste ispisemo ->>> napisat u drugoj funkciji za vjezbu

char \* filename1="samoglasnici.txt",\* filename2="suglasnici.txt";

FILE \*fp1=NULL,\*fp2=NULL;

if(p!=NULL)

{

PrintToFiles(p->Left);

if(tolower(p->rijec[0]=='a' || p->rijec[0]=='e' || p->rijec[0]=='i' || p->rijec[0]=='o' || p->rijec[0]=='u'))

{

fp1=fopen(filename1,"a");

if(fp1==NULL)

{

printf("Datoteka se nije otvorila!\n");

}

fprintf(fp1," %s\n",p->rijec);

fclose(fp1);

}

else

{

fp2=fopen(filename2,"a");

if(fp2==NULL)

{

printf("Datoteka se nije otvorila!\n");

}

fprintf(fp2," %s\n",p->rijec);

fclose(fp2);

}

PrintToFiles(p->Right);

}

}

Poss CreateNode()

{

Poss q=NULL;

q=(Poss)malloc(sizeof(node));

if(q==NULL)

return NULL;

else return q;

}

void InsertToLists(Pos p,Poss a,Poss s)

{

Poss temp1=NULL,temp2=NULL,q=NULL;

if(p!=NULL)

{

InsertToLists(p->Left,a,s);

if(tolower(p->rijec[0]=='a' || p->rijec[0]=='e' || p->rijec[0]=='i' || p->rijec[0]=='o' || p->rijec[0]=='u'))

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

}

strcpy(q->rijec,p->rijec);

temp1=a;

while(temp1->Next!=NULL && temp1->Next->rijec[0] < q->rijec[0])

temp1=temp1->Next;

q->Next=temp1->Next;

temp1->Next=q;

}

else

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

}

strcpy(q->rijec,p->rijec);

temp1=s;

while(temp1->Next!=NULL && temp1->rijec[0] < q->rijec[0])

temp1=temp1->Next;

q->Next=temp1->Next;

temp1->Next=q;

}

InsertToLists(p->Right,a,s);

}

}

void PrintList(Poss p,char \* filename)

{

FILE \*fp=NULL;

fp=fopen(filename,"w");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

}

else

{

p=p->Next;

if(p==NULL)

printf("Lista je prazna!\n");

while(p!=NULL)

{

fprintf(fp," %s\n",p->rijec);

p=p->Next;

}

}

fclose(fp);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

/\*Napisati program koji iz datoteke čita dio teksta, riječ po riječ i unosi ih u red s prioritetom (maksimalna duljina riječi 1024 znaka, prioritet=1 => riječ počinje samoglasnikom, prioritet=2 => riječ počinje

suglasnikom;). Riječi u redu moraju biti sortirane po prioritetu. --->>> RED - skida s početka , stavlja na kraj(ali gledamo prioritete)

Iz tako kreiranog reda kreirati dvije datoteke, u prvoj će se nalaziti riječi sa prioritetom=1, a u drugoj riječi s prioritetom=2; s tim da su te riječi u datoteci zapisane sortirane po abecedi.\*/

// prototipovi funkcija

struct Red;

typedef struct Red \* Pos;

struct Red

{

char rijec[1024];

int prioritet;

Pos Next;

}red;

Pos CreateNode();

int ReadFromFile(Pos);

int Insert(Pos);

void Print(Pos);

int main()

{

Pos head=NULL;

head=CreateNode();

if(head==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head->Next=NULL;

ReadFromFile(head);

Print(head);

return 0;

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(red));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

else return q;

}

int ReadFromFile(Pos p)

{

FILE \*fp=NULL;

char imedatoteke[30]="",rijec[30]="";

Pos temp=NULL,q=NULL;

printf("Unesite ime datoteke: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

else

{

while(!feof(fp))

{

fscanf(fp," %s",rijec);

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -2;

}

strcpy(q->rijec,rijec);

temp=p;

if(rijec[0]=='a' || rijec[0]=='A' || rijec[0]=='e' || rijec[0]=='E' || rijec[0]=='i' || rijec[0]=='I' || rijec[0]=='o' || rijec[0]=='O' || rijec[0]=='u' || rijec[0]=='U')

{

q->prioritet=1;

while(temp->Next!=NULL && temp->Next->rijec[0] < q->rijec[0] && temp->Next->prioritet==q->prioritet)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

}

else

{

q->prioritet=2;

while(temp->Next!=NULL && temp->Next->prioritet==1)

temp=temp->Next;

while(temp->Next!=NULL && temp->Next->rijec[0] < q->rijec[0] && temp->Next->prioritet==q->prioritet)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

}

}

fclose(fp);

}

return 0;

}

void Print(Pos p)

{

char \*dat1="samoglasnici.txt",\*dat2="suglasnici.txt";

FILE \*d1=NULL,\*d2=NULL;

d1=fopen(dat1,"w");

d2=fopen(dat2,"w");

if(d1==NULL || d2==NULL)

{

printf("Datoteka se nije otvorila!\n");

}

p=p->Next;

while(p!=NULL && p->prioritet==1)

{

fprintf(d1," %s\n",p->rijec);

p=p->Next;

}

while(p!=NULL && p->prioritet==2)

{

fprintf(d2," %s\n",p->rijec);

p=p->Next;

}

fclose(d1);

fclose(d2);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

/\*Napisati program koji za tri različita predmeta čita ime i prezime studenta te njegovu ocjenu. Podaci o svakom predmetu se nalaze u odvojenoj datoteci i iz svake datoteke je potrebno kreirati vezanu listu i ispisati je.

a) Nakon toga potrebno je kreirati novu listu koja sadrži imena studenata koji su položili sva tri predmeta.

b) Nadopuniti zadatak pod a) tako da se ispiše nova vezana lista koja osim što sadrži imena studenata, sadrži i njihovu prosječnu ocjenu iz ta tri predmeta. \*/

struct Node;

typedef struct Node \* Pos;

struct Nova;

typedef struct Nova \* Poss;

struct Node

{

char ime[30];

char prezime[30];

int ocjena;

Pos Next;

}node;

struct Nova

{

char ime[30];

float prosjek;

Poss Next;

}ova;

// prototipovi funkcija

Pos CreateNode();

int ReadFromFile(Pos,char\*);

void PrintList(Pos);

int CreateNova(Pos,Pos,Pos,Poss);

Pos Find(Pos,char\*,char\*);

int InsertNova(Poss,char\*,float);

void PrintNova(Poss);

int main()

{

Pos head1=NULL,head2=NULL,head3=NULL;

Poss nova=NULL;

char dat1[10]="",dat2[10]="",dat3[10]="";

head1=CreateNode();

head2=CreateNode();

head3=CreateNode();

nova=(Poss)malloc(sizeof(ova));

if(head1==NULL || head2==NULL || head3==NULL || nova==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head1->Next=NULL;

head2->Next=NULL;

head3->Next=NULL;

nova->Next=NULL;

printf("Unesite imena 3ju datoteka iz kojih zelite procitati: ");

scanf(" %s %s %s",dat1,dat2,dat3);

ReadFromFile(head1,dat1);

ReadFromFile(head2,dat2);

ReadFromFile(head3,dat3);

PrintList(head1->Next);

printf("\n\n\n\n");

PrintList(head2->Next);

printf("\n\n\n\n");

PrintList(head3->Next);

CreateNova(head1,head2,head3,nova);

printf("\n\n\n");

PrintNova(nova->Next);

return 0;

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(node));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

else return q;

}

int ReadFromFile(Pos p,char \* ime)

{

FILE \*fp=NULL;

Pos q=NULL;

fp=fopen(ime,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

else

{

while(!feof(fp))

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -2;

}

fscanf(fp," %s %s %d",q->ime,q->prezime,&q->ocjena);

q->Next=p->Next;

p->Next=q;

p=q;

}

fclose(fp);

}

return 0;

}

void PrintList(Pos p)

{

if(p==NULL)

printf("Lista je prazna!\n");

else

while(p!=NULL)

{

printf("\n %s %s %d ",p->ime,p->prezime,p->ocjena);

p=p->Next;

}

}

int CreateNova(Pos h1,Pos h2,Pos h3,Poss nova)

{

Pos find2=NULL,find3=NULL;

int zbroj=0;

float prosjek=0.0;

h1=h1->Next;

while(h1!=NULL)

{

find2=Find(h2->Next,h1->ime,h1->prezime);

find3=Find(h3->Next,h1->ime,h1->prezime);

if(find2!=NULL && find3!=NULL)

{

if(h1->ocjena > 1 && find2->ocjena > 1 && find3->ocjena > 1)

{

zbroj=h1->ocjena+find2->ocjena+find3->ocjena;

prosjek=(float)zbroj/3;

InsertNova(nova,h1->ime,prosjek);

}

}

h1=h1->Next;

}

return 0;

}

Pos Find(Pos p,char\* ime,char \*prezime)

{

while(p!=NULL && ( strcmp(ime,p->ime)!=0 || strcmp(prezime,p->prezime)!=0 ))

p=p->Next;

return p;

}

int InsertNova(Poss p,char \*ime,float prosjek)

{

Poss q=NULL;

q=(Poss)malloc(sizeof(ova));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

strcpy(q->ime,ime);

q->prosjek=prosjek;

q->Next=p->Next;

p->Next=q;

p=q;

return 0;

}

void PrintNova(Poss p)

{

if(p==NULL)

printf("Prazna lista!\n");

while(p!=NULL)

{

printf("\n %s %f",p->ime,p->prosjek);

p=p->Next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/// Strukture podataka 3-ispitni rok 28.8.2014.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include <time.h>

struct Node;

typedef struct Node \* Pos;

struct Node

{

int elem;

int random;

Pos Next;

}node;

// prototipovi funkcija

int Insert(Pos);

Pos CreateNode();

void Print(Pos);

int Generate(int \*);

int Sort(int\*,Pos);

int main()

{

Pos head=NULL;

int \*a=NULL,\*b=NULL,i;

srand((unsigned)time(NULL));

head=CreateNode();

if(head==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head->Next=NULL;

Insert(head);

Print(head);

printf("\n\nRandom brojevi: \n");

a=(int\*)malloc(sizeof(int)\*5);

Generate(a);

for(i=0;i<5;i++)

printf(" %d\t ",a[i]);

printf("\nPRVI PROLAZ:\n");

Sort(a,head);

Print(head);

b=(int\*)malloc(sizeof(int)\*5);

for(i=0;i<5;i++)

b[i]=a[5-i-1];

printf("\nDRUGI PROLAZ:\n");

Sort(b,head);

Print(head);

return 0;

}

int Insert(Pos p)

{

Pos q=NULL;

int i=1;

srand((unsigned)time(NULL));

while(i!=20) // SMANJI BROJ JER INAČE DUGO VRTI WHILE U FUNKCIJI SORT

{

q=CreateNode();

if(q==NULL)

return -1;

q->elem=rand()%51+100;

q->random=rand()%20;

q->Next=p->Next;

p->Next=q;

p=q;

i++;

}

return 0;

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(node));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

else return q;

}

void Print(Pos p)

{

p=p->Next;

if(p==NULL)

printf("Lista je prazna!\n");

while(p!=NULL)

{

printf(" %d\t%d\n",p->random,p->elem);

p=p->Next;

}

}

int Generate(int \*a)

{

int i=0,j=0,isti=0;

srand((unsigned)time(NULL));

for(i=0;i<5;i++)

{

a[i]=rand()%20;

}

while(isti==0)

{

isti=1;

for(i=0;i<4;i++)

for(j=i+1;j<5;j++)

{

if(a[i]==a[j])

isti=0;

}

if(!isti)

{

for(i=0;i<5;i++)

a[i]=rand()%20;

}

}

return 0;

}

int Sort(int \*a,Pos p)

{

Pos temp=NULL,prev=NULL,pocetak=p,pomocna=NULL;

int i=0;

for(i=0;i<5;i++)

{

prev=p;

temp=prev->Next;

while(temp!=NULL)

{

if(temp->random==a[i])

{

pomocna=pocetak->Next;

pocetak->Next=temp;

prev->Next=temp->Next;

temp->Next=pomocna;

prev=temp;

temp=temp->Next;

}

prev=temp;

temp=temp->Next;

}

}

return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Strukture podataka – 3 ispitni rok - dodatni 11.9.2014.

Napisati program koji generira slučajne brojeve u opsegu od 50-80. Ukoliko je generirani broj paran sprema se u vezanu listu parnih brojeva

i to u padajućem redoslijedu (od najvećeg prema najmanjem).

Ukoliko je broj neparan sprema se u vezanu listu neparnih brojeva, također u padajućem redoslijedu.

Niti jedna vezana lista ne smije sadržavati dva ista elementa.

Unos traje sve dok obje liste ne sadrže 10 elemenata. Treba ispisivati koji se broj generira i na kraju ispisati obje liste.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

// prototipovi funkcija

struct Node;

typedef struct Node \* Pos;

struct Node

{

int elem;

Pos Next;

}node;

Pos CreateNode();

int CheckElements(Pos,int);

int Insert(Pos,Pos);

int Place(Pos,int);

void Print(Pos);

int main()

{

Pos head1=NULL,head2=NULL;

srand((unsigned)time(NULL));

head1=CreateNode();

head2=CreateNode();

if(head1==NULL || head2==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head1->Next=NULL;

head2->Next=NULL;

Insert(head1,head2);

printf("\nParni: \n");

Print(head1);

printf("\nNeparni: \n");

Print(head2);

return 0;

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(node));

if(q==NULL)

{

printf("Nije se alociralo!\n");

return NULL;

}

else return q;

}

int CheckElements(Pos p,int x)

{

p=p->Next;

if(p==NULL)

{

return 1;

}

while(p!=NULL)

{

if(x==p->elem)

return 0;

p=p->Next;

}

return 1;

}

int Insert(Pos p,Pos n)

{

int i=0,j=0,x=0;

srand((unsigned)time(NULL));

while(i<10 || j<10)

{

x=rand()%31+50;

if(x%2==1 && CheckElements(n,x) && i<10)

{

i++;

printf("Izgeneriran je broj %d!\n",x);

Place(n,x);

}

else if(x%2==0 && CheckElements(p,x) && j<10)

{

j++;

printf("Izgeneriran je broj %d!\n",x);

Place(p,x);

}

}

return 0;

}

int Place(Pos p,int x)

{

Pos q=NULL,temp=NULL;

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

q->elem=x;

temp=p;

while(temp->Next!=NULL && temp->Next->elem > x)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

return 0;

}

void Print(Pos p)

{

p=p->Next;

if(p==NULL)

{

printf("Prazna lista!\n");

}

while(p!=NULL)

{

printf(" %d\t ",p->elem);

p=p->Next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Napisati program koji iz dvije datoteke (jedna sadrži ocjenu 1. kolokvija,a druga ocjenu 2. kolokvija) kreira vezanu listu studenata koji su položili ispit

i ta se nova vezana lista slaže po ocjenama. Način pretvorbe bodova u ocjenu je dan u tablici.

Bodovi Ocjena

50 – 59.99% 2 60 – 73.99% 3 74 – 86.99% 4 87 - 100 5 \*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

struct Node;

typedef struct Node \* Pos;

struct Nova;

typedef struct Nova \* Poss;

struct Node

{

char ime[30];

char prezime[30];

int bodovi;

Pos Next;

}node;

struct Nova

{

char ime[30];

char prezime[30];

int ocjena;

Poss Next;

}nova;

Pos CreateNode();

Poss CreateNova();

int ReadFromFile(Pos,char\*);

int InsertNova(Poss,Pos,Pos);

void Print(Poss);

Pos Find(Pos,char\*,char\*);

int Ocjena(float);

// --> Napravit 2 liste za svaki kolokvij i 3.listu i onda tu listu pri unosu sortiramo po ocjenama

int main()

{

Pos k1=NULL,k2=NULL;

Poss head=NULL;

char \*f1=NULL,\*f2=NULL;

k1=CreateNode();

k2=CreateNode();

head=CreateNova();

if(k1==NULL || k2==NULL || head==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

k1->Next=NULL;

k2->Next=NULL;

head->Next=NULL;

printf("Unesite ime 2 datoteke iz koje citate ocjene: ");

f1=(char\*)malloc(sizeof(char)\*10);

f2=(char\*)malloc(sizeof(char)\*10);

scanf(" %s %s",f1,f2);

ReadFromFile(k1,f1);

ReadFromFile(k2,f2);

InsertNova(head,k1,k2);

Print(head);

return 0;

}

Pos CreateNode()

{

Pos p=NULL;

p=(Pos)malloc(sizeof(node));

if(p==NULL)

return NULL;

else return p;

}

Poss CreateNova()

{

Poss p=NULL;

p=(Poss)malloc(sizeof(nova));

if(p==NULL)

return NULL;

else return p;

}

int ReadFromFile(Pos p,char \* imedatoteke)

{

FILE \*fp=NULL;

Pos q=NULL;

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

else

{

while(!feof(fp))

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

if(fscanf(fp," %s %s %d",q->ime,q->prezime,&q->bodovi)==3)

{

if(q->bodovi >-1 && q->bodovi < 101)

{

q->Next=p->Next;

p->Next=q;

p=q;

}

else

{

printf("Krivo unesen broj bodova!\n");

}

}

}

fclose(fp);

}

return 0;

}

int InsertNova(Poss p,Pos k1,Pos k2)

{

Poss q=NULL,temp=NULL;

Pos findk2=NULL;

float ukbodovi=0.0;

k1=k1->Next;

while(k1!=NULL)

{

findk2=Find(k2,k1->ime,k1->prezime);

if(findk2!=NULL)

{

if(k1->bodovi > 49 && findk2->bodovi > 49)

{

ukbodovi=(float)(k1->bodovi+findk2->bodovi)/2;

q=CreateNova();

if(q==NULL)

return -1;

strcpy(q->ime,k1->ime);

strcpy(q->prezime,k1->prezime);

q->ocjena=Ocjena(ukbodovi);

if(q->ocjena==-1)

{

printf("Niste dobro unijeli bodove!\n");

return -2;

}

temp=p;

while(temp->Next!=NULL && temp->Next->ocjena > q->ocjena)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

}

}

k1=k1->Next;

}

return 0;

}

void Print(Poss p)

{

if(p->Next==NULL)

{

printf("Lista je prazna!\n");

}

p=p->Next;

while(p!=NULL)

{

printf("\n %s %s %d\t",p->ime,p->prezime,p->ocjena);

p=p->Next;

}

}

Pos Find(Pos k,char \*ime,char \*prezime)

{

k=k->Next;

while(k!=NULL && ( strcmp(k->ime,ime)!=0 || strcmp(k->prezime,prezime)!=0 ) )

k=k->Next;

return k;

}

int Ocjena(float ukbodovi)

{

if(ukbodovi>=50.0 && ukbodovi<=59.99)

return 2;

else if(ukbodovi>=60.0 && ukbodovi<=73.99)

return 3;

else if(ukbodovi>=74.0 && ukbodovi<=86.99)

return 4;

else if(ukbodovi>=87.0 && ukbodovi<=100.0)

return 5;

else return -1;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*2.ZADATAK Napisati program koji iz datoteke cita rijeci,te ih sprema u vezanu listu s tim ukoliko se ista rijec ponavlja tada se ona ne zapisuje u vezanu listu nego se brojac uvecava.

a) Ispisati tako kreiranu listu s tim da je redoslijed rijeci obrnut od redoslijeda kojim se pojavljuju u datoteci.

b) Iz tako kreirane liste kreirati novu listu,na nacin, da su elementi u ovoj listi sortirani prvo po broju ponavljanja , a zatim po abecednom redu.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

struct Node;

typedef struct Node \* Pos;

struct Node

{

char rijec[20];

int brojac;

Pos Next;

}node;

// prototipovi funkcija

Pos CreateNode();

int ReadFromFile(Pos);

Pos Find(Pos,char\*);

void Print(Pos);

int CreateNewList(Pos,Pos);

int main()

{

Pos head=NULL,nova=NULL;

head=CreateNode();

nova=CreateNode();

if(head==NULL || nova==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head->Next=NULL;

nova->Next=NULL;

ReadFromFile(head);

Print(head);

CreateNewList(head,nova);

Print(nova);

return 0;

}

Pos CreateNode()

{

Pos q=NULL;

q=(Pos)malloc(sizeof(node));

if(q==NULL)

{

return NULL;

}

else return q;

}

int ReadFromFile(Pos p)

{

FILE \*fp=NULL;

char imedatoteke[20]="",rijec[20]="";

Pos find=NULL;

printf("Unesite ime datoteke: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return -1;

}

else

{

while(!feof(fp))

{

Pos q=NULL;

q=CreateNode();

if(q==NULL)

return -1;

fscanf(fp," %s",rijec);

find=Find(p,rijec);

if(find==NULL)

{

strcpy(q->rijec,rijec);

q->brojac=1;

q->Next=p->Next;

p->Next=q;

}

else

{

find->brojac++;

}

}

fclose(fp);

}

return 0;

}

Pos Find(Pos p,char \*rijec)

{

while(p!=NULL && strcmp(p->rijec,rijec)!=0)

p=p->Next;

return p;

}

void Print(Pos p)

{

if(p->Next==NULL)

{

printf("Lista je prazna!\n");

}

p=p->Next;

printf("\n\nRijec \t Broj ponavljanja");

while(p!=NULL)

{

printf("\n %s\t%d",p->rijec,p->brojac);

p=p->Next;

}

}

int CreateNewList(Pos p,Pos nova)

{

Pos q=NULL,temp=NULL;

p=p->Next;

while(p!=NULL)

{

q=CreateNode();

if(q==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

strcpy(q->rijec,p->rijec);

q->brojac=p->brojac;

temp=nova;

while(temp->Next!=NULL && temp->Next->brojac < q->brojac)

temp=temp->Next;

while(temp->Next!=NULL && temp->Next->rijec[0] < q->rijec[0] && temp->Next->brojac==q->brojac)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

p=p->Next;

}

return 0;

}

// PRIMJER bila mama bio tata imali su maloga juju zaljubljeni i odrasli su tata i mama tata tata bio zubar

// PRIMJER2 naranca mandarina jabuka kruska limun limun naranca kivi kivi kivi jabuka kivi voce

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*1.ZADATAK Napisati program koji iz datoteke cita znak po znak te ih sprema u binarno stablo pretrazivanja (struct \_treeNode {char znak; int brojac; struct\_treeNode \* left; struct \_treeNode \* Right; };)

Pri spremanju znaka po znak u stablo, znakovi se usporeduju po svojim ASCII vrijednostima. Ukoliko se naide na isti znak, potrebno je povećati brojac(inace je 1).

a) Ispisati sve znakove s brojem ponavljanja u tekstu po ASCII vrijednostima

b) Ispisati sve znakvoe s brojem ponavljanja u tekstu sortitrane po broju ponavljanja od najveceg ka najmanjem,te zatim po ASCII vrijednosti znaka.\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

struct TreeNode;

typedef struct TreeNode \* Pos;

struct Node;

typedef struct Node \* Poss;

struct TreeNode

{

char znak;

int brojac;

Pos Left;

Pos Right;

}treenode;

struct Node

{

char znak;

int brojac;

Poss Next;

}node;

Pos ReadFromFile(Pos);

Pos Insert(Pos,char);

Pos Find(Pos,char);

void PrintInorder(Pos);

Poss CreateNode();

void CreateList(Pos,Poss);

int InsertInList(Pos,Poss);

void PrintList(Poss);

// prototipovi funkcija

int main()

{

Pos root=NULL;

Poss head=NULL;

head=CreateNode();

if(head==NULL)

{

printf("Nije se alociralo!\n");

return -1;

}

head->Next=NULL;

root=ReadFromFile(root);

printf("\nZNAK\tASCII VRIJEDNOST\tBROJ PONAVLJANJA\n");

PrintInorder(root);

CreateList(root,head);

printf("\nZNAK\tASCII VRIJEDNOST\tBROJ PONAVLJANJA\n");

PrintList(head);

return 0;

}

Pos ReadFromFile(Pos p)

{

FILE \*fp=NULL;

char imedatoteke[20]="",znak;

Pos find=NULL;

printf("Unesite ime datoteke iz koje zelite citati znakove: ");

scanf(" %s",imedatoteke);

fp=fopen(imedatoteke,"r");

if(fp==NULL)

{

printf("Datoteka se nije otvorila!\n");

return NULL;

}

else

{

while(!feof(fp))

{

if(fscanf(fp,"%c",&znak)==1)

{

find=Find(p,znak);

if(find==NULL)

{

p=Insert(p,znak);

}

else if(find!=NULL)

{

find->brojac++;

}

}

}

fclose(fp);

}

return p;

}

Pos Insert(Pos p,char znak)

{

if(p==NULL)

{

p=(Pos)malloc(sizeof(treenode));

if(p==NULL)

{

printf("Nije se alociralo!\n");

return p;

}

p->znak=znak;

p->brojac=1;

p->Left=NULL;

p->Right=NULL;

}

else if((int)p->znak < (int)znak)

p->Right=Insert(p->Right,znak);

else if((int)p->znak > (int)znak)

p->Left=Insert(p->Left,znak);

return p;

}

Pos Find(Pos p,char znak)

{

if(p==NULL)

return NULL;

else if((int)p->znak < (int)znak)

return Find(p->Right,znak);

else if((int)p->znak > (int)znak)

return Find(p->Left,znak);

else return p;

}

void PrintInorder(Pos p)

{

if(p!=NULL)

{

PrintInorder(p->Left);

if((int)p->znak==32)

printf("\n\\n\t\t%d\t\t%d",(int)p->znak,p->brojac);

else if((int)p->znak==10)

printf("\n\\b\t\t%d\t\t%d",(int)p->znak,p->brojac);

else printf("\n%c\t\t%d\t\t%d",p->znak,(int)p->znak,p->brojac);

PrintInorder(p->Right);

}

}

Poss CreateNode()

{

Poss q=NULL;

q=(Poss)malloc(sizeof(node));

if(q==NULL)

return NULL;

else return q;

}

void CreateList(Pos tree,Poss p)

{

if(tree!=NULL)

{

CreateList(tree->Left,p);

InsertInList(tree,p);

CreateList(tree->Right,p);

}

}

int InsertInList(Pos tree,Poss p)

{

Poss q=NULL,temp=NULL;

q=CreateNode();

if(q==NULL)

return -1;

q->brojac=tree->brojac;

q->znak=tree->znak;

temp=p;

while(temp->Next!=NULL && temp->Next->brojac > q->brojac)

temp=temp->Next;

while(temp->Next!=NULL && temp->Next->brojac==q->brojac && (int)(temp->Next->znak)>(int)q->brojac)

temp=temp->Next;

q->Next=temp->Next;

temp->Next=q;

return 0;

}

void PrintList(Poss p)

{

p=p->Next;

if(p==NULL)

printf("Lista je prazna!\n");

while(p!=NULL)

{

if((int)p->znak==32)

printf("\n\\n\t\t%d\t\t%d",(int)p->znak,p->brojac);

else if((int)p->znak==10)

printf("\n\\b\t\t%d\t\t%d",(int)p->znak,p->brojac);

else printf("\n%c\t\t%d\t\t%d",p->znak,(int)p->znak,p->brojac);

p=p->Next;

}

}