Algoritmi i strukture podataka
1. blic – primjeri i objašnjenja nekih pitanja
ak. god. 2007/08

by Mad Mat

4) Nakon izvršavanja naredbe fprintf, što će se nalaziti zapisano u datoteci (gledajući binarno):
fprintf(fp, "%c\n", '0');

a. 00110000
b. **00110000 00001010**
c. 00110000 00000000
d. 00000000 00001010
e. 00000000
s lijeva na desno čitaš 00110000 = 48 = '0' 00001010 = 10 = '\n'

---

5) Što će biti zapisano u datoteku sljedećim programskim odsječkom:
int i = 4; FILE *datIzlaz;
...
fprintf(datIzlaz, "%3d", i);

a. 00000000 00000000 00110100
b. 00000100
c. 00000000 00000000 00000100
d. 00100000 00100000 00110100
e. 01100100
Odgovor: 00001010 - to je ascii kod za \n
00100000 - ascii kod za razmak
a brojevi se zapisuju kao ascii kod za char tog broja
zato 4 nije 00000100, već je 52, tj. 00110100

Neka se ključevi zapisa nekom metodom transformacije u postupku raspršenog adresiranja transformiraju u vrijednosti iz intervala [0,999]. Koji će raspon vrijednosti smjestiti u pretinac s adresom 3, ako su adrese pretinaca iz intervala od [0,199]? U koji će pretinac otići zapis s ključem koji se transformira u vrijednost 56?
1000/200=5

0.pretinac: [0,4]
1.pretinac: [5,9]
2.pretinac: [10,14]
3.pretinac: [15, 19]

a za 56:
znači znaš da taj traženi pretinac ima raspon vrijednosti [55,59] a njegova adresa je onda 55:5=11

Što će biti zapisano u datoteku sljedećim programskim odsječkom:

```
int i = 4; FILE *datIzlaz;
...
fprintf(datIzlaz, "%3d", i);
```

a.00000000 00000000 00110100
b.00000100
c.00000000 00000000 00000100
d.00100000 00100000 00110100
e.01100100
Odgovor: Razmak, razmak, ASCII 4.

```
void main() {
int a, b, *p;
a=10; b=5; p=&a;
*p = b++;
printf("a=%d b=%d", a, b);
}
```

a) a=5 b=5
b) a=11 b=6
c) a=5 b=6
d) a=10 b=6
e) a=11 b=5
odgovor: zbog b++, da je ++b onda bi bilo 6, 6. Ovak prvo pridruži 5 pa poveća za 1

```
char p, char r, char *pp, char *pr, char *pom;
p='p';pp=&p;
r='r';pr=&r;
pom=pp;
pp=pr;
pr=pom;
p='r';r='p';
printf("%c %c %c %c",r,*pp,p,*pr);
```

a) p r r r
b) r p p r
c) p p r r
d) r p r p
e) r r p p
znaci imas varijable p,r
onda imas pokazivace pp, pr i pom
prvo stavi 'p' u varijablu p i stavi u pp adresu od p
onda stavi 'r' u varijablu r i stavi u pr adresu od r
u pom stavi pp (znaci adresu od p)
u pp stavi pr (znaci adresu od r)
u pr stavi pom (znaci adresu od p)
kaze da je p='r' i r='p'
i sad samo citas sta trebas
trazi se prvo r, on je 'p'
onda se trazi *pp, a to znaci ono sto je na adresi pp, potrazis pp i vidis da je tu adresa od r, znaci u *pp je 'p'
onda se trazi p, u p je 'r'
onda se trazi *pr, a to znaci ono sto je na adresi pr, potrazis pr i vidis da je tu adresa od p, a u p se nalazi 'r'

11) Što će biti zapisano u datoteku sljedećim programskim odsječkom?

```
...
int i = 2; FILE *datIzlaz;
...
fprintf(datIzlaz, "%2d", i);
...
```

a. 00000000 00110010
b. 01100010
c. 00000000 00000010
d. 00000010
e. 00100000 00110010
Ovdje je teksutalna(fprintf)..dakle razmak+ASCII od 2 pa je rješ: e

---

Za koliko bajta maksimalno naraste stog tijekom izvođenja odsječka

```
...
y = g(10);
...
```

ako su definirane funkcije

```
long f(long a, long b) { return a + b;}
double g(long x) { return 2.*f(x,x); }
```

Napomena: u obzir uzeti samo argumente, lokalne varijable i povratnu adresu
Zašto je tu odgovor 20? Zato jer: long je 4 bajta, povratna adresa također,
dakle a+b+pov.adr.(f)+x+pov.adr.(g)=20 bajtova

Kolika će biti vrijednost varijable n nakon izvođenja sljedećeg programskog odsječka:

```
f = fopen ("ulaz", "r");
n = 1;
while (fgets(s, 512, f)) {
n++;
}
```

ako se datoteka ulaz uspješno otvori, a sadrži sljedeći niz bajtova:
a a \n b \n c \n d d d d d d d \n
zasto 5
n je prvo 1, fgets čita do znaka \n, u datoteci se nalaze 4 takva znaka dakle n+=4 ->5

ili
String se učitava dok se ne dođe do znaka za novi red ('\n') ili se ne učita maksimalni broj znakova (512 u ovom zadatku). Dakle ovdje se učitaju četiri stringa: a a; b; c; d d d d d d d

---

Formatirani ispis se čita s lijeva na desno bajt po bajt. Svaki bajt sadrži ASCII vrijednost.
fprintf(file, "%d %3.2f", 72, 14.321f);

00000111 00000010 00100000 00000001 00000100 00101110 00000011

7 2 space 1 4 . 3
Napomena: navodno ipak treba biti umjesto binarnih 7, 2, 1, 4 i 3 ići zapravo binarni 55, 50, 49, 52 i 51 jer su to ASCII vrijednosti?

Neformatirani ispisi se prvo rastavljaju na dijelove čije veličine ovise veličini podatka koji se zapisuje (char - 1; short - 2, int, long, float - 4, double - 8). Dijelovi se čitaju s lijeva na desno dok se sadržaj jednog dijela čita sa desna na lijevo.
short polje[2] = { 1, 7 };

fwrite(polje, sizeof(short), 2, file);

00000001 00000000 00000111 00000000

1 7

Adresa pohranjena u pokazivaču deklariranom kao int *p; povećat će se za 1 bajt sljedećom naredbom:

a. p=(int *)(p+1);

b. p=(int *)((char *)p + 1);

c. p=*p+1;

d. p=*(p+1);

e. p=(char *)((int *)p + 1);
Odgovor: b je tocan zato sto je prvo pokazivac na int prevori u pokazivac na char i onda se doda ovaj 1*sizeof(char)..a to je 1 bajt

---

Kako bismo rezervirali memorijski prostor za znakovni niz duljine 8?
a. (char*)malloc(8*sizeof(char));
b. **(char*)malloc(9);**
c. (int*)malloc(8)*sizeof(int));
d. (char*)realloc(9*sizeof(char));
e. realloc(p, 8*sizeof(char));
Odgovor: jer znakovni niz završava sa '\0' pa moramo sačuvati jedno mjesto i za to

---

```
void f( int x) {
if (x>3) return;
f(x-1);
printf("%d",x);
}
```

ako je poziv funkcije f(2);
rješenje: prepunit će se stog prije bilo kakvog ispisa
odgovor: pa znači budući da je x<3 pozvat će se f(x-1) i onda će se opet u toj funkciji pozvati ponovno f(x-1) itd...(jer x nikad neće biti veći od 3) pa će se onda zato valjda prenapuniti stog.
ali mislim da to spada pod rekurzije pa ne bi trebal doći sada u ovom blicu

---

```
int main (void)
{
long int *broj1, *broj2;
int broj;
broj1 = broj2 = (long int *) malloc (4 * sizeof( long int));
broj2 += 8;
broj = (int) ( (char *) broj2 - (char *) broj1 );
printf("%d", broj );
return 0;
}
```

koja je fora s ovim zadatkom?? odgovor kazu da je 32...ako netko eventualno zna i ima volje objasnit zasto??
Odgovor: broj1 i broj2 ti zbog malloca zauzimaju 16 byte. Zbog broj2+=8, broj2 zauzima 24 byte. E sad nakon toga imaš broj = (int) ((char *) broj2 - (char *) broj1);
(char*)broj2=24, (char*)broj1=16, (char*)broj2-(char*)broj1=8.
E sad imaš (int) od 8 (4*8), a to je 32.

---

2) Ako je sadržaj formatirane datoteku ulaz 012012 što će ispisati sljedeći programski odsječak:

```
char c;
int s=0;
FILE *in=fopen("ulaz","r");
while(fscanf(in,"%c",&c)){
s+=c-'1';
}
printf("%d\n",s);
```

a. 49
b. 0
c. 48
d. 288
e. 6

Računaš po redu, s tim da ovu ASCII jedinicu koju se oduzima pretvoriš u dekadski 49 pa imaš:

s = 0 + 48 (48 je dekadska vrijednost ove prve ulazne 0) - 49 = -1
s = -1 + 49 (49 je dekadska vrijednost od 1 koji dolazi iza 0) - 49 = -1
s = -1 + 50 (sad kužiš, jel, to je ASCII 2) - 49 = 0
s = 0 + 48 - 49 = -1
s = -1 +49 - 49 = -1
s = -1 + 50 - 49 = 0

---

1) Uzmimo da su podaci o studentima pohranjeni u datoteku. Podaci su sortirani prema JMBAG-u (kao na slici). Koliko će čitanja biti obavljeno za dohvaćanje podataka o studentu čiji je JMBAG 35 ako se koristi čitanje po blokovima, a veličina bloka je 4?
1
2
3
5
8
12
20
25
30
35
40
41
42
43

a. 4
b. 7
c. 5
d. 1
e. 6
Odgovor: uglvnom rjesenje je 6.citanje po blokovima funkcionira da podjelis ovo kaj je zadano na blokove od po 4 zapisa..
znaci 1.blok (1235)
2.blok (8 12 20 25)
3.blok (30 35 40 41)
4.blok (42 43....)

i sad prvo ies na prvi blok i gleas dal je veci od prvog zapisa onaj zapis koji trazis,veci je...ides na drugi blok i gledas dal je 35 vece od 8,,je vece je...sad ides na 3 blok i gledas dal je vece od 30,je vece je ...onda ides na 4 blok (to je 4. korak) i vidis da j e 35 manje od 42 i vratis se na blok prije (to je 5 korak,)i sad ides po tombloku i dodes na 35=35 i toje 6 korak....

---

2) Uzmimo da su podaci o studentima pohranjeni u datoteku. Podaci su sortirani prema JMBAG-u (kao na slici). Koliko će čitanja biti obavljeno za dohvaćanje podataka o studentu čiji je JMBAG 35 ako se koristi binarno adresiranje?

1,2,3,5,8,12,20,25,30,35,40,41,42,43
a.3
b.2
c.4
d.7
e.6

znači binarno se pretražuje tako da ideš na pola nekog niza , pa opet na pola i tako do traženog elementa. ovdje je 14 elemenata, pola je 7 , pa se pozicioniraš na polovicu, da ima 15 članova ti bi se pozicionirala na 8. član tako da sa obje strane tog člana bude 7 elemenata, sad se ne možeš pozicionirat točno na pola pa se pozicioniraš na lijevi (iliti kako je u zadatku zadano -> gornji) element

1,2,3,5,8,12,20,25,30,35,40,41,42,43

tražiš broj 35, 20 je manje od 35, znači da sad idemo desno, opet na pola od ostatka niza

1,2,3,5,8,12,20,**25,30,35,40,41,42,43**

40 je veće od 35, znači da sad idemo lijevo, opet na pola od ostatka niza

1,2,3,5,8,12,20,**25,30,35,**40,41,42,43

30 je manje, znači desno idemo

1,2,3,5,8,12,20,25,30,**35**,40,41,42,43

i to je 4 koraka

```
void f(int x,int *y){
x%=2;
*y * x;
}

void main(){
int a=8,b=10;…….greška v zadatku; piše 2 a treba 10
printf("%d %d",a,b);
f(a,&b);
printf("->%d %d",a,b);
}
```

RJ: 8 10 -> 8 10

i to je to, ispiše ti 8 10
onda pozove void funkciju koja ništa ne promjeni pa opet ispiše 8 10

---

11. Što će se ispisati programom:

```
int fun (char *c, int i) {
(*c)++; i--;
return (*c) * i;
}
main () {
char c = '0'; int i = 1, j;
j = fun (&c, i);
printf ("%d %d %d", c, i, j);
}
```

a) 49 1 0
Odgovor: U ASCII-ju ti je nula pod brojem 48, ovdje u funkciji povecas taj 48 za 1 [(*c)++], dakle bit ce 49. i se ne mijenja (i=1), a posto u funkciji imas i-- = 0 (pazi, ovo ne utjece na i u glavnom programu) i kad mnozis s bilo cim bit ce nula -> return (*c) * i = return (*c)*0 = return 0, pa ce j biti nula

Ili opširnije
c='0' i=1 i j su zadani
pozoveš fun(...)

(*c)++ ti poveća c za 1 -> **c='1'** , smanji se i -> **i=0**
i onda return vrati u glavni program *c *i , znači vrati 0 -> **j=0**

i sad, pošto si prenijela **ADRESU** od C promjenio si vrijednost c-a u funkciji (call by reference)
a **i** si prenijela kao vrijednost običnu pa bilo kakva promjena **i**-a u funkciji ne utječe na to koliko iznosi i u glavnom programu

znači sada se returnom vraćaš u glavni program, **c** je promjenjeno i sad je c='49' , **i** je **NE**promjenjeno!
i=1, a **j=0**

pa ti ispiše 49 1 0

49 ispiše sato što vrijedi '1' = 49 dekadski

Za koliko bajta maksimalno naraste stog tijekom izvođenja odsječka
...
y = g(10);
...
ako su definirane funkcije

```
long f(long a, long b) {
return a + b;
}

double g(long x) {
return 2.*f(x,x);
}
```

Napomena: u obzir uzeti samo argumente, lokalne varijable i povratnu adresu.
JEL ZNA NETKO KAKO SE DOBIJE OVDJE REZ 20Byta
Odgovor: odozgo prema dolje sam pisao stog
long=4
povratna adresa=4?
---pozove se 2. funkcija----
long=4
long=4
povratna adresa=4?
i vrati se a+b
ako je povratna adresa 4 bajta onda je 20

U datoteku organiziranu po načelu raspršenog adresiranja pohranjuju se zapisi koji sadrže šifru iz intervala [100000, 500000] i naziv (19 +1 znak). Fizički blok na disku je veličine 512 okteta. Koliki je maksimalni broj zapisa po pretincu?

a. 22

b. 25

c. 26

d. 21

e. 27

odgovor; Šifra ide u int (4 bajta), ime u polje znakova (20 bajtova). Ukupno 24 bajta. U blok od 512 bajtova stane 512 / 24 = 21 zapis.
ili opširnije
U datoteku organiziranu po načelu raspršenog adresiranja pohranjuju se zapisi koji sadrže šifru iz intervala [100000, 500000] i naziv (19 +1 znak). Fizički blok na disku je veličine 512 okteta. Koliki je maksimalni broj zapisa po pretincu?

veličina strukture je **24** bajta ---> naziv (19 +1 znak) = **20** bajta + šifru iz intervala [100000, 500000] = **4** bajta

veličina pretinca = 512 / 24 = 21.33333 = 21 (zaokružuješ na manje jer ne može postojat 21 i po pretinac a ni 22 pretinca jer nema dovoljno mjesta za to)

---

6) U datoteku organiziranu po načelu raspršenog adresiranja pohranjuju se zapisi koji sadrže šifru iz intervala [100000, 500000] i naziv (19 +1 znak). Fizički blok na disku je veličine 512 okteta. Broj zapisa koje treba pohraniti je 10000. Broj pretinaca je zbog očekivanog preljeva veći za 30%. Koliki je u tom slučaju broj pretinaca?

a. ~ 476
b. ~ 620
c. ~ 48
d. ~ 143
e. ~ 470
Odgovor: Ako u jedan pretinac stane 21 zapis onda će nam za 10000 zapisa idealno trebati 10000 / 21 = 476 pretinaca, povećano za 30% = 619 pretinaca.
Ili opširnije
U datoteku organiziranu po načelu raspršenog adresiranja pohranjuju se zapisi koji sadrže šifru iz intervala [100000, 500000] i naziv (19 +1 znak). Fizički blok na disku je veličine 512 okteta. Broj zapisa koje treba pohraniti je 10000. Broj pretinaca je zbog očekivanog preljeva veći za 30%. Koliki je u tom slučaju broj pretinaca?
opet je 24 bajta veličina strukture
veličina pretinca = 512 / 24 = 21.33333 = 21
broj pretinaca = (10 000 / 21)* 1,3 = **619,04** i to zaokružuješ na veći broj zbog nečega, ne znam čega hehe
dakle, ~ 620 je odgovor

---

5)Što će biti zapisano u datoteku sljedećim programskim odsječkom:
```
int i = 4; FILE *datIzlaz;
...
fprintf(datIzlaz, "%3d", i);
```
a.00000000 00000000 00110100
b.00000100
c.00000000 00000000 00000100
d.00100000 00100000 00110100
e.01100100
Odgovor: %3d znači da se moraju ispisati najmanji tri znaka (uključujući broj). U slučaju da broj ima manje od tri znamenke prvo će se ispisati odgovarajući broj jedinica.
4" = 00100000 00100000 00110100 (ASCII kodovi)

---

12) Koja od sljedećih tvrdnji nije istinita, ako imamo naredbu: *p=7;:

a. Pokazivač p mora prethodno biti inicijaliziran da bi naredba bila logički ispravna.

b. Nova adresa na koju pokazuje pokazivač p nakon naredbe
p=p+broj;
može se dobiti na sljedeći način:
nova_adresa=stara_adresa+broj*sizeof(long);

Odgovor: Zato što p = p + broj znači: povečaj p tako da pokazuje broj long-ova naprijed.
```
p = p + 4;              /* p pokazuje 4 long-a (16 bajtova) naprijed */

p = p + sizeof(long) * 4;   /*p pokazuje 4 * 4 long-a (64 bajta) naprijed */
```

c. Ako je deklarirano polje int a[5] sljedeće naredbe, kojima pristupamo trećem elementu polja, su ekvivalentne:
*(a+2)
a[2]

d. Pokazivače se može uspoređivati.

e. Pokazivaču se može oduzeti i dodati cijeli broj.

---

8) Što će biti zapisano u neformatiranu datoteku dat.txt sljedećim programskim odsječkom:
```
FILE *fp;
short int a=9;
fp=fopen("dat.txt", "w+");
fwrite(&a, sizeof(a), 1, fp);
```

a. 00001001 00000000
b. 00000000 00001001
c. 00001001 00000000 00000000 00000000
d. 00000000 00000000 00001001 00000000
e. 00000000 00000000 00000000 00001001

---

11) Što će biti zapisano u datoteku sljedećim programskim odsječkom?
...
int i = 2; FILE *datIzlaz;
...
fprintf(datIzlaz, "%2d", i);
...
a. 00000000 00110010
b. 01100010
c. 00000000 00000010
d. 00000010
e. 00100000 00110010
koja je razlika u ispisu formatiranih i neformatiranih datoteka.
Odgovor: Formatirane sadrže ASCII kodove. Neformatirane sadrže brojeve u "sirovom" little endian binarnom obliku.

---

39. Koja od sljedeæih tvrdnji vezanih uz stog je istinita?

a) interni stog raèunala koristi se samo pri deklaraciji globalnih varijabli C programa
Baš obrnuto.

b) pojedina operacija dodaj (push) i brisi (pop) zahtijeva jednako vremena bez obzira na broj pohranjenih podataka
Istina: Push i Pop se svode na smanjivanje tj. povećanje jednog pokazivača.

c) stog je programska struktura u koju se dodaju i brišu elementi po naèelu FIFO (First In First Out)
LIFO (Last In First Out): sjeti se analogije sa tanjurima.

d) za programsku realizaciju stoga moguæe je iskljuèivo koristiti stog koji je definiran kao polje
Može se riješiti i na druge načine (npr. vezanim listama).

e) na stog se mogu pohraniti isljuèivo cjelobrojni (int) podaci
Sve što postoji u programu može na stog.

---

7) Što će biti zapisano u neformatiranu datoteku slijedećom naredbom:
char c='A';
FILE *datoteka;
datoteka=fopen("test.dat","wb");
fwrite(&c, sizeof(c), 1, datoteka);

a. 01000001
b. 00000000 00000000 00000000 01000001
c. 01000001 00000000 00000000 00000000
d. 00000000 01000001
e. 00000000 00000000 00000000 01000001

8) Što će biti zapisano u neformatiranu datoteku dat.txt sljedećim programskim odsječkom:
FILE *fp;
short int a=9;
fp=fopen("dat.txt", "w+");
fwrite(&a, sizeof(a), 1, fp);

a. 00001001 00000000
b. 00000000 00001001
c. 00001001 00000000 00000000 00000000
d. 00000000 00000000 00001001 00000000
e. 00000000 00000000 00000000 00001001
zašto i u sedmom nema još niz nula ko u 8.? bi će da me taj little endian mući...

odgovor:u ovom prvom ti je c char, (1 bajt), a u drugom ti je a short int (2 bajta)

Koliko je prosječno asimptotsko vrijeme izvođenja funkcije pot, ako se broje samo pozivi funkcije i povratci iz funkcije:
long pot(long x, long y){
if (y<=0) return 1;
*else return x*pot(x, y-1);*
}
a. y
b. 2y
c. y^2
d. ylog(y)
e. y/2
odgovor: ovdje ti je poziv funkcije ujedno i povratak iz funkcije, a to se događa y puta i puta 2 zbog poziva i povratka.

---

zna li netko pravilo kod datoteka,kada se unosi ASCII vrijednost a kada normalna????
Odgovor: ASCII uvijek kod formatiranih, što god da dobiješ, slovo, broj, razmak, sve to gledaš kao ASCII pa pretvaraš u dekadsku vrijednost pa u binarnu.

Kod neformatiranih unosiš ono što i dobiješ. Ako dobiješ ASCII, onda opet pretvaranje u dekadsku pa u binarnu, a ako dobiješ integer, onda normalno samo ga pretvorit u binarnu.