Glava 1

Digitalizacija

Današnji računari su *digitalni*. To znači da su svi podaci koji su u njima zapisani – zapisani kao nizovi brojeva. Zapisivanje tekstova, slika, zvuka i filmova u vidu brojeva zahteva pogodnu reprezentaciju koja ponekad znači i gubitak dela polaznih informacija. Kada su podaci predstavljeni u vidu brojeva, te brojeve potrebno je zapisati u računarima. Dekadni brojevni sistem koji ljudi koriste u svakodnevnom životu nije pogodan za zapis brojeva u računarima jer zahteva azbuku od 10 različitih simbola (cifara). Bilo da se radi o elektronskim, magnetnim ili optičkim komponentama, tehnologija izrade računara i medijuma za zapis podataka koristi elemente koji imaju dva diskretna stanja, što za zapis podataka daje azbuku od samo dva različita simbola. Tako, na primer, ukoliko između dve tačke postoji napon viši od određenog praga, onda se smatra da tom paru tačaka odgovara vrednost 1, a inače mu odgovara vrednost 0. Takođe, polje hard diska može biti ili namagnetisano što odgovara vrednosti 1 ili razmagnetisano što odgovara vrednosti 0. Slično, laserski zrak na površini kompakt diska „buši rupice“ kojim je određen zapis podataka pa polje koje nije izbušeno predstavlja vrednost 0, a ono koje jeste izbušesno predstavlja vrednost 1. U nastavku će biti pokazano da je azbuka od samo dva simbola dovoljna za zapisivanje svih vrsta brojeva, pa samim tim i za zapisivanje svih vrsta digitalnih podataka.

1.1 Analogni i digitalni podaci i digitalni računari

Kontinualna priroda signala. Većina podataka koje računari koriste nastaje zapisivanjem prirodnih signala. Najznačajniji primeri signala su zvuk i slika, ali se pod signalima podrazumevaju i ultrazvučni signali, EKG signali, zračenja različite vrste itd.

Signali koji nas okružuju u prirodi u većini slučajeva se prirodno mogu predstaviti neprekidnim funkcijama. Na primer, zvučni signal predstavlja promenu pritiska vazduha u zadatoj tački i to kao neprekidnu funkciju vremena. Slika se može opisati intenzitetom svetlosti određene boje (tj. određene talasne dužine) u datom vremenskom trenutku i to kao neprekidna funkcija prostora.

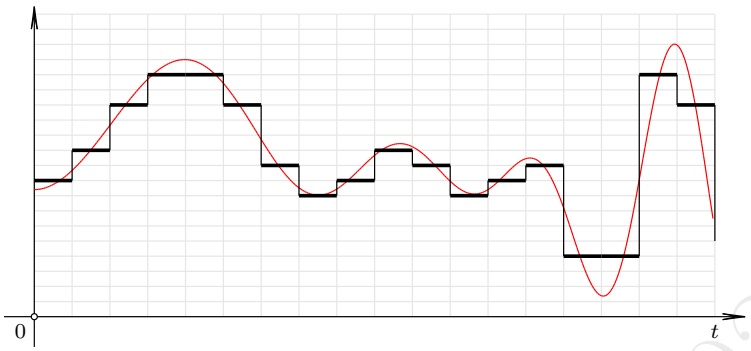
Analogni zapis. Osnovna tehnika koja se primenjuje kod analognog zapisa signala je da se kontinualne promene signala koji se zapisuje opišu kontinualnim promenama određenog svojstva medijuma na kojem se signal zapisuje. Tako, na primer, promene pritiska vazduha koji predstavlja zvučni signal direktno odgovaraju promenama nivoa namagnetisanja na magnetnoj traci na kojoj se zvuk analogno zapisuje. Količina boje na papiru direktno odgovara intenzitetu svetlosti u vremenskom trenutku kada je fotografija bila snimljena. Dakle, analogni zapis uspostavlja *analogiju* između signala koji je zapisan i određenog svojstva medijuma na kome je signal zapisan.

Osnovna prednost analogne tehnologije je da je ona obično veoma jednostavna ukoliko se zadovoljimo relativno niskim kvalitetom (još su drevni narodi mogli da naprave nekakav zapis zvuka uz pomoć jednostavne igle prikačene na trepereću membranu).

Osnovni problem analogne tehnologije je što je izrazito teško na medijumu napraviti veran zapis signala koji se zapisuje i izrazito je teško napraviti dva identična zapisa istog signala. Takođe, problem predstavlja i inherentna nestalnost medijuma, njegova promenljivost tokom vremena i podložnost spoljašnjim uticajima. S obzirom na to da varijacije medijuma direktno dovode do varijacije zapisanog signala, vremenom neizbežno dolazi do pada kvaliteta analogno zapisanog signala. Obrada analogno zapisanih signala je obično veoma komplikovana i za svaku vrstu obrade signala, potrebno je da postoji uređaj koji je specijalizovan za tu vrste obrade.

Digitalni zapis. Osnovna tehnika koja se koristi kod digitalnog zapisa podataka je da se vrednost signala izmeri u određenim vremenskim trenucima ili određenim tačkama prostora i da se onda na medijumu zapišu

izmerene vrednosti. Ovim je svaki digitalno zapisani signal predstavljen nizom brojeva koji se nazivaju *odbirci* ili *semplovi* (engl. *sample*). Svaki od brojeva predstavlja vrednost signala u jednoj tački diskretizovanog domena. S obzirom na to da izmerene vrednosti takođe pripadaju kontinualnoj skali, neophodno je izvršiti i diskretizaciju kodomena, odnosno dopustiti zapisivanje samo određenog broja nivoa različitih vrednosti.



Slika 1.1: Digitalizacija zvučnog signala

Digitalni zapis predstavlja diskretnu aproksimaciju polaznog signala. Važno pitanje je koliko često je potrebno vršiti merenje da bi se polazni kontinualni signal mogao verno rekonstruisati. Odgovor daje tvrđenje o odabiranju (tzv. Najkvist-Šenonova teorema), koje kaže da je signal dovoljno meriti dva puta češće od najviše frekvencije koja sa u njemu javlja. Na primer, pošto čovekovo uho čuje frekvencije do 20kHz, dovoljno je da frekvencija odabiranja (semplovanja) bude 40kHz. Dok je za analogne tehnologije za postizanje visokog kvaliteta zapisa potrebno imati medijume visokog kvaliteta, kvalitet reprodukcije digitalnog zapisa ne zavisi od toga kakav je kvalitet medija na kome su podaci zapisani, sve dok je medijum dovoljnog kvaliteta da se zapisani brojevi mogu razaznati. Dodatno, kvarljivost koja je inherentna za sve medije postaje nebitna. Na primer, papir vremenom žuti što uzrokuje pad kvaliteta analognih fotografija tokom vremena. Međutim, ukoliko bi papir sadržao zapis brojeva koji predstavljaju vrednosti boja u tačkama digitalno zapisane fotografije, činjenica da papir žuti ne bi predstavljala problem dok god se brojevi mogu razaznati.

Digitalni zapis omogućava kreiranje apsolutno identičnih kopija što dalje omogućava prenos podataka na daljinu. Na primer, ukoliko izvršimo fotokopiranje fotografije, napravljena fotokopija je daleko lošijeg kvaliteta od originala. Međutim, ukoliko umnožimo CD na kojem su zapisani brojevi koji čine zapis neke fotografije, kvalitet slike ostaje apsolutno isti. Ukoliko bi se dva CD-a pregledala pod mikroskopom, oni bi izgledali delimično različito, ali to ne predstavlja problem sve dok se brojevi koji su na njima zapisani mogu razaznati.

Obrada digitalno zapisanih podataka se svodi na matematičku manipulaciju brojevima i ne zahteva (za razliku od analognih podataka) korišćenje specijalizovanih mašina.

Osnovni problem implementacije digitalnog zapisa predstavlja činjenica da je neophodno imati veoma razvijenu tehnologiju da bi se uopšte stiglo do iole upotrebljivog zapisa. Na primer, izuzetno je komplikovano napraviti uređaj koji je u stanju da 40 hiljada puta izvrši merenje intenziteta zvuka. Jedna sekunda zvuka se predstavlja sa 40 hiljada brojeva, za čiji je zapis neophodna gotovo cela jedna sveska. Ovo je osnovni razlog zbog čega se digitalni zapis istorijski javio kasno. Kada se došlo do tehnološkog nivoa koji omogućava digitalni zapis, on je doneo mnoge prednosti u odnosu na analogni.

1.2 Zapis brojeva

Proces digitalizacije je proces reprezentovanja (raznovrsnih) podataka brojevima. Kako se svi podaci u računarima reprezentuju na taj način — brojevima, neophodno je precizno definisati zapisivanje različitih vrsta brojeva. Osnovu digitalnih računara, u skladu sa njihovom tehnološkom osnovom, predstavlja *binarni* brojevni sistem (sistem sa osnovom 2) . U računarstvu se koriste i *heksadekadni* brojevni sistem (sistem sa osnovom 16) a i, nešto ređe, *oktalni* brojevni sistem (sistem sa osnovom 8), zbog toga što ovi sistemi omogućavaju jednostavnu konverziju između njih i binarnog sistema. Svi ovi brojevni sistemi, kao i drugi o kojima će biti reči u nastavku teksta, su *pozicioni*. U pozicionom brojnom sistemu, udeo cifre u celokupnoj vrednosti zapisanog broja zavisi od njene pozicije.

Treba naglasiti da je zapis broja samo konvencija a da su brojevi koji se zapisuju apsolutni i ne zavise od konkretnog zapisa. Tako, na primer, zbir dva prirodna broja je uvek jedan isti prirodni broj, bez obzira na to u kom sistemu su ova tri broja zapisana.

S obzirom na to da je svaki zapis broja u računaru ograničen, ne mogu biti zapisani svi celi brojevi. Ipak, za cele brojeve zapisive u računaru se obično govori samo *celi brojevi*, dok su ispravnija imena *označeni celi brojevi* (engl. *signed integers*) i *neoznačeni celi brojevi* (engl. *unsigned integers*), koji podrazumevaju konačan zapis. Ni svi realni brojevi (sa potencijalno beskonačnim decimalnim zapisom) ne mogu biti zapisani u računaru. Za zapis zapisivih realnih brojeva (koji su uvek racionalni) obično se koristi konvencija zapisa u pokretnom zarezu. Iako je jedino precizno ime za ove brojeve *brojevi u pokretnom zarezu* (engl. *floating point numbers*), često se koriste i imena *realni* ili *racionalni brojevi*. Zbog ograničenog zapisa brojeva, rezultati matematičkih operacija nad njima sprovedenih u računaru, neće uvek odgovarati rezultatima koji bi se dobili bez tih ograničenja (zbog takozvanih *prekoračenja*). Naglasimo još i da, za razliku od matematike gde se skup celih brojeva smatra podskupom skupa realnih brojeva, u računarstvu, zbog različitog načina zapisa, između zapisa ovih vrsta brojeva ne postoji direktna veza.

1.2.1 Neoznačeni brojevi

Pod *neoznačenim brojevima* podrazumeva se neoznačeni zapis nenegativnih celih brojeva i znak se izostavlja iz zapisa.

Određivanje broja na osnovu datog zapisa. Pretpostavimo da je dat pozicioni brojevni sistem sa osnovom b, gde je b prirodan broj veći od 1. Niz cifara $(a_n\, a_{n-1}\, \ldots\, a_1\, a_0)_b$ predstavlja zapis[1] broja u osnovi b, pri čemu za svaku cifru a_i važi $0 \leq a_i < b$.

Vrednost broja zapisanog u osnovi b definiše se na sledeći način:

$$(a_n\, a_{n-1}\, \ldots\, a_1\, a_0)_b = \sum_{i=0}^{n} a_i \cdot b^i = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots a_1 \cdot b + a_0$$

Na primer:
$(101101)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2 + 1 = 32 + 8 + 4 + 1 = 45$,
$(3245)_8 = 3 \cdot 8^3 + 2 \cdot 8^2 + 4 \cdot 8 + 5 = 3 \cdot 512 + 2 \cdot 64 + 4 \cdot 8 + 5 = 1536 + 128 + 32 + 5 = 1701$.
Navedena definicija daje i postupak za određivanje vrednosti datog zapisa:

```
x := 0
za svako i od 0 do n
    x := x + a_i · b^i
```

ili, malo modifikovano:

```
x := a_0
za svako i od 1 do n
    x := x + a_i · b^i
```

Za izračunavanje vrednosti nekog $(n+1)$-tocifrenog zapisa drugim navedenim postupkom potrebno je n sabiranja i $n + (n-1) + \ldots + 1 = \frac{n(n+1)}{2}$ množenja. Zaista, da bi se izračunalo $a_n \cdot b^n$ potrebno je n množenja, da bi se izračunalo $a_{n-1} \cdot b^{n-1}$ potrebno je $n-1$ množenja, itd. Međutim, ovo izračunavanje može da se izvrši i efikasnije. Ukoliko se za izračunavanje člana b^i iskoristi već izračunata vrednost b^{i-1}, broj množenja se može svesti na $2n$. Ovaj način izračunavanja primenjen je u sledećem postupku:

```
x := a_0
B := 1
za svako i od 1 do n
    B := B · b
    x := x + a_i · B
```

Još efikasniji postupak izračunavanja se može dobiti korišćenjem *Hornerove sheme*:

$$(a_n\, a_{n-1}\, \ldots\, a_1\, a_0)_b = (\ldots((a_n \cdot b + a_{n-1}) \cdot b + a_{n-2}) \ldots + a_1) \cdot b + a_0$$

Korišćenjem ove sheme, dolazi se do sledećeg postupka za određivanje vrednosti broja zapisanog u nekoj brojevnoj osnovi:

[1] Ako u zapisu broja nije navedena osnova, podrazumeva se da je osnova 10.

```
x := 0
za svako i od n unazad do 0
    x := x · b + a_i
```

Naredni primer ilustruje primenu Hornerovog postupka na zapis $(9876)_{10}$.

i		3	2	1	0
a_i		9	8	7	6
x	0	$0 \cdot 10 + 9 = 9$	$9 \cdot 10 + 8 = 98$	$98 \cdot 10 + 7 = 987$	$987 \cdot 10 + 6 = 9876$

Međurezultati dobijeni u ovom računu direktno odgovaraju prefiksima zapisa čija se vrednost određuje, a rezultat u poslednjoj koloni je traženi broj.

Navedeni postupak može se primeniti na proizvoljnu brojevnu osnovu. Sledeći primer ilustruje primenu postupka na zapis $(3245)_8$.

i		3	2	1	0
a_i		3	2	4	5
x	0	$0 \cdot 8 + 3 = 3$	$3 \cdot 8 + 2 = 26$	$26 \cdot 8 + 4 = 212$	$212 \cdot 8 + 5 = 1701$

Navedena tabela može se kraće zapisati na sledeći način:

	3	2	4	5
0	3	26	212	1701

Hornerov postupak je efikasniji u odnosu na početni postupak, jer je u svakom koraku dovoljno izvršiti samo jedno množenje i jedno sabiranje (ukupno $n + 1$ sabiranja i $n + 1$ množenja).

Određivanje zapisa datog broja. Za svaku cifru a_i u zapisu broja x u osnovi b važi da je $0 \leq a_i < b$. Dodatno, pri deljenju broja x osnovom b, ostatak je a_0 a celobrojni količnik je broj čiji je zapis $(a_n\, a_{n-1} \ldots a_1)_b$. Dakle, izračunavanjem celobrojnog količnika i ostatka pri deljenju sa b, određena je poslednja cifra broja x i broj koji se dobija uklanjanjem poslednje cifre iz zapisa. Ukoliko se isti postupak primeni na dobijeni količnik, dobija se postupak koji omogućava da se odrede sve cifre u zapisu broja x. Postupak se zaustavlja kada tekući količnik postane 0. Ako se izračunavanje ostatka pri deljenju označi sa `mod`, a celobrojnog količnika sa `div`, postupak kojim se određuje zapis broja x u datoj osnovi b se može formulisati na sledeći način:

```
i := 0
dok je x različito od 0
    a_i := x mod b
    x := x div b
    i := i + 1
```

Na primer, $1701 = (3245)_8$ jer je $1701 = 212 \cdot 8 + 5 = (26 \cdot 8 + 4) \cdot 8 + 5 = ((3 \cdot 8 + 2) \cdot 8 + 4) \cdot 8 + 5 = (((0 \cdot 8 + 3) \cdot 8 + 2) \cdot 8 + 4) \cdot 8 + 5$. Ovaj postupak se može prikazati i tabelom:

i	0	1	2	3	
x	1701	1701 div 8 = 212	212 div 8 = 26	26 div 8 = 3	3 div 8 = 0
a_i	1701	1701 mod 8 = 5	212 mod 8 = 4	26 mod 8 = 2	3 mod 8 = 3

Prethodna tabela može se kraće zapisati na sledeći način:

1701	212	26	3	0
5	4	2	3	

Druga vrsta tabele sadrži celobrojne količnike, a treća ostatke pri deljenju sa osnovom b, tj. tražene cifre. Zapis broja se formira tako što se dobijene cifre čitaju unatrag.

Ovaj algoritam i Hornerov algoritam su međusobno simetrični u smislu da se svi međurezultati poklapaju.

Direktno prevođenje između heksadekadnog i binarnog sistema.
Osnovni razlog korišćenja heksadekadnog sistema je mogućnost jednostavnog prevođenja brojeva između binarnog i heksadekadnog sistema. Pri tome, heksadekadni sistem omogućava da se binarni sadržaj memorije zapiše kompaktnije (uz korišćenje manjeg broja cifara). Prevođenje se može izvršiti tako što se grupišu četiri po četiri binarne cifre, krenuvši unazad, i svaka četvorka se zasebno prevede u odgovarajuću heksadekadnu cifru na osnovu sledeće tabele:

heksa	binarno	heksa	binarno	heksa	binarno	heksa	binarno
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Na primer, proizvoljni 32-bitni sadržaj može se zapisati korišćenjem osam heksadekadnih cifara:
$(1011\,0110\,0111\,1100\,0010\,1001\,1111\,0001)_2 = (B67C29F1)_{16}$

Zapisi fiksirane dužine U računarima se obično koristi fiksirani broj binarnih cifara (sačuvanih u pojedinačnim *bitovima*) za zapis svakog broja. Takve zapise označavamo sa $(\ldots)_b^n$, ako se koristi n cifara. Ukoliko je broj cifara potrebnih za zapis broja kraći od zadate dužine zapisa, onda se broj proširuje vodećim nulama. Na primer, $55 = (0011\,0111)_2^8$. Ograničavanjem broja cifara ograničava se i raspon brojeva koje je moguće zapisati (u binarnom sistemu) i to na raspon od 0 do $2^n - 1$. U sledećoj tabeli su dati rasponi za najčešće korišćene dužine zapisa:

broj bitova	raspon
8	od 0 do 255
16	od 0 do 65535
32	od 0 do 4294967295

1.2.2 Označeni brojevi

Označeni brojevi su celi brojevi čiji zapis uključuje i zapis znaka broja (+ ili −). S obzirom na to da savremeni računari koriste binarni brojevni sistem, biće razmatrani samo zapisi označenih brojeva u binarnom brojevnom sistemu. Postoji više načina zapisivanja označenih brojeva od kojih su najčešće u upotrebi *označena apsolutna vrednost* i *potpuni komplement*.

Označena apsolutna vrednost. Zapis broja se formira tako što se na prvu poziciju zapisa unapred fiksirane dužine n, upiše znak broja, a na preostalih $n-1$ pozicija upiše zapis apsolutne vrednosti broja. Pošto se za zapis koriste samo dva simbola (0 i 1), konvencija je da se znak + zapisuje simbolom 0, a znak − simbolom 1. Ovim se postiže da pozitivni brojevi imaju identičan zapis kao da su u pitanju neoznačeni brojevi. Na primer, $+100 = (0\ 1100100)_2^8$, $-100 = (1\ 1100100)_2^8$.

Osnovni problem zapisa u obliku označene apsolutne vrednosti je činjenica da se osnovne aritmetičke operacije teško izvode ukoliko su brojevi zapisani na ovaj način.

Potpuni komplement. Zapis u potpunom komplementu (engl. two's complement) označenih brojeva zadovoljava sledeće uslove:

1. Nula i pozitivni brojevi se zapisuju na isti način kao da su u pitanju neoznačeni brojevi, pri čemu u njihovom zapisu prva cifra mora da bude 0.

2. Sabiranje se sprovodi na isti način kao da su u pitanju neoznačeni brojevi, pri čemu se prenos sa poslednje pozicije zanemaruje.

Na primer, broj +100 se u potpunom komplementu zapisuje kao $(0\,1100100)_2^8$. Nula se zapisuje kao $(0\,0000000)_2^8$. Zapis broja −100 u obliku $(\ldots)_2^8$ se može odrediti na sledeći način. Zbir brojeva −100 i +100 mora da bude 0.

	binarno	dekadno
	????????	-100
+	01100100	+100
1	00000000	0

Analizom traženog sabiranja cifru po cifru, počevši od poslednje, sledi da se -100 mora zapisati kao $(10011100)_2^8$. Do ovoga je moguće doći i na sledeći način. Ukoliko je poznat zapis broja x, zapis njemu suprotnog broja je moguće odrediti iz uslova da je $x+(-x) = (1\,00\ldots00)_2^{n+1}$. Pošto je $(1\,00\ldots00)_2^{n+1} = (11\ldots11)_2^n+1$, zapis broja $(-x)$ je moguće odrediti tako što se izračuna $(11\ldots11)_2^n - x + 1$. Izračunavanje razlike $(11\ldots11)_2^n - x$ se svodi na *komplementiranje* svake pojedinačne cifre broja x. Tako se određivanje zapisa broja -100 može opisati na sledeći način:

	01100100	$+100$
	10011011	komplementiranje
$+$	1	
	10011100	

Kao što je traženo, zapisi svih pozitivnih brojeva i nule počinju cifrom 0 dok zapisi negativnih brojeva počinju sa 1.

Broj -2^{n-1} je jedini izuzetak u opštem postupku određivanja zapisa u potpunom komplementu. Zapis broja $(100\ldots00)_2^n$ je sam sebi komplementaran, a pošto počinje cifrom 1, uzima se da on predstavlja zapis najmanjeg zapisivog negativnog broja, tj. broja -2^{n-1}. Zahvaljujući ovoj konvenciji, u zapisu potpunog komplementa $(\ldots)_2^n$ moguće je zapisati brojeve od -2^{n-1} do $2^{n-1}-1$. U sledećoj tabeli su dati rasponi za najčešće korišćene dužine zapise u potpunom komplementu:

broj bitova	raspon
8	od -128 do $+127$
16	od -32768 do $+32767$
32	od -2147483648 do $+2147483647$

Kao što je rečeno, za sabiranje brojeva zapisanih u potpunom komplementu može se koristiti opšti postupak za sabiranje neoznačenih brojeva (pri čemu se podrazumeva da može doći do prekoračenja, tj. da neke cifre rezultata ne mogu biti upisane u raspoloživ broj mesta). To ne važi samo za sabiranje, već i za oduzimanje i množenje (pri čemu kod realizacije množenja u procesoru tj. u mašinski zavisnim jezicima postoje određene razlike između množenja označenih i neoznačenih brojeva). Ovo pogodno svojstvo važi i kada je jedan od argumenata broj -2^{n-1} (koji se je jedini izuzetak u opštem postupku određivanja zapisa). Sve ovo su važni razlozi za korišćenje potpunog komplementa u računarima.

1.2.3 Zapis realnih brojeva

Pored celobrojnih dostupni u programskim jezicima su po pravilu podržani i realni brojevni tipovi. Realne brojeve je u računaru mnogo komplikovanije predstaviti nego cele brojeve. Obično je moguće predstaviti samo određeni podskup skupa realnih brojeva i to podskup skupa racionalnih brojeva. Interni zapis celog broja i njemu odgovarajućeg realnog se ne poklapaju (nule i jedinice kojima se oni zapisuju nisu iste), čak i kada se isti broj bitova koristi za njihov zapis. Fiksiranjem bitova koji će se odvojiti za zapis nekog realnog broja, određuje se i broj mogućih različitih brojeva koji se mogu zapisati. Svakoj kombinaciji nula i jedinica pridružuje se neki realan broj. Kod celih brojeva odlučivano je da li će se takvim kombinacijama pridruživati samo pozitivni ili i pozitivni i negativni brojevi i kada je to odlučeno, u principu je jasno koji skup celih brojeva može biti zapisan (taj skup vrednosti je uvek neki povezan raspon celih brojeva). U zapisu realnih brojeva stvari su komplikovanije jer je potrebno napraviti kompromis između širine raspona brojeva koji se mogu zapisati (slično kao i kod celih brojeva), ali i između preciznosti brojeva koji se mogu zapisati. Dakle, kao i celobrojni tipovi, realni tipovi imaju određen raspon vrednosti koji se njima može predstaviti, ali i određenu preciznost zapisa (nju obično doživljavamo kao broj decimala, mada to tumačenje, kao što ćemo uskoro videti, nije uvek u potpunosti tačno).

Osnovni načini zapisivanja realnih brojeva su zapis u *fiksnom zarezu* i zapis u *pokretnom zarezu*. Zapis u fiksnom zarezu podrazumeva da se posebno zapisuje znak broja, zatim ceo deo broja i zatim njegov razlomljeni deo (njegove decimale). Broj cifara za zapis celog dela i za zapis razlomljenog dela je fiksiran i jednak je za sve brojeve u okviru istog tipa koji koristi zapis u fiksnom zarezu. Zapis u pokretnom zarezu podrazumeva oblik $\pm m \cdot b^e$. Vrednost b je osnova koja se podrazumeva (danas je to obično 2, mada se nekada koristila i vrednost 16, dok se u svakodnevnoj matematičkoj praksi često brojevi izražavaju na ovaj način uz korišćenje osnove 10), m se naziva mantisa, a vrednost e naziva se eksponent. Broj cifara (obično binarnih) za zapis mantise i broj cifara za zapis eksponenta je fiksiran i jednak je za sve brojeve u okviru istog tipa koji koristi zapis u pokretnom zarezu. Zapisivanje brojeva u pokretnom zarezu propisano je standardom *IEEE 754* iz 1985. godine, međunarodne asocijacije *Institut inženjera elektrotehnike i elektronike*, IEEE (engl. *Institute of Electrical and Electronics Engineers*). Ovaj standard predviđa da se određene kombinacije bitova odvoje za zapis tzv. specijalnih vrednosti (beskonačnih vrednosti, grešaka u izračunavanju i slično).

Ilustrujmo osnovne principe zapisa realnih brojeva na dekadnom zapisu nenegativnih realnih brojeva (kao što smo rekli, u računaru se ovakav zapis interno ne koristi, već se koristi binarni zapis). Zamislimo da imamo tri dekadne cifre koje možemo iskoristiti za zapis. Dakle, imamo 1000 brojeva koji se mogu zapisati. Prva mogućnost je da se određeni broj cifara upotrebi za zapis celog dela, a određeni broj cifara za zapis decimala i takav način zapisa predstavlja zapis u fiksnom zarezu. Na primer, ako se jedna cifra upotrebi za decimale moći ćemo zapisivati vrednosti 00,0, 00,1, ..., 99,8 i 99,9. Ako se se za decimale odvoje dva mesta, onda možemo zapisivati vrednosti 0,00, 0,01, ..., 9,98 i 9,99. Ovim smo dobili bolju preciznost (svaki broj je zapisan sa dve, umesto sa jednom decimalom), ali smo to platili mnogo užim rasponom brojeva koje možemo zapisati (umesto širine oko 100, dobili smo raspono širine oko 10). U računaru se fiksni zarez često ostvaruje binarno (određeni broj bitova kodira ceo, a određeni broj bitova kodira razlomljeni deo), pri čemu se uvek jedan bit ostavlja za predstavljanje znaka broja čime se omogućava zapis i pozitivnih i negativnih vrednosti.

Umesto fiksnog zareza, u računarima se mnogo češće koristi pokretni zarez. U takvom zapisu, naše tri dekadne cifre podelićemo tako da dve odlaze za zapis mantise, a jednu za zapis eksponenta (pa će zapis biti oblika $m_1m_2e_3$). Ako su prve dve cifre m_1 i m_2, tumačićemo da je mantisa $0,m_1m_2$. Ako je treća cifra u zapisu e_3, tumačićemo da eksponent $e = e_3 - 4$ (ovim postižemo da vrednosti eksponenta mogu da budu između -4 i 5 tj. da mogu da budu i pozitivne i negativne). Pogledajmo neke zapise koje na taj način dobijamo.

zapis	vrednost	zapis	vrednost	...	zapis	vrednost	zapis	vrednost
010	$0{,}01 \cdot 10^{-4} = 0{,}000001$	011	$0{,}01 \cdot 10^{-3} = 0{,}00001$	...	018	$0{,}01 \cdot 10^{4} = 100{,}0$	019	$0{,}01 \cdot 10^{5} = 1\,000{,}0$
020	$0{,}02 \cdot 10^{-4} = 0{,}000002$	021	$0{,}02 \cdot 10^{-3} = 0{,}00002$	...	028	$0{,}02 \cdot 10^{4} = 200{,}0$	029	$0{,}02 \cdot 10^{5} = 2\,000{,}0$
980	$0{,}98 \cdot 10^{-4} = 0{,}000098$	981	$0{,}98 \cdot 10^{-3} = 0{,}00098$	...	988	$0{,}98 \cdot 10^{4} = 9800{,}0$	989	$0{,}98 \cdot 10^{5} = 98\,000{,}0$
990	$0{,}99 \cdot 10^{-4} = 0{,}000099$	991	$0{,}99 \cdot 10^{-3} = 0{,}00099$	...	998	$0{,}99 \cdot 10^{4} = 9900{,}0$	999	$0{,}99 \cdot 10^{5} = 99\,000{,}0$

Raspon je prilično širok (od $0,000001$ do $99\,000,0$ tj. od oko 10^{-6} do oko 10^5) i mnogo širi nego što je to bilo kod fiksnog zareza, ali gustina zapisa je neravnomerno raspoređena, što nije bio slučaj kod fiksnog zareza. Kod malih brojeva preciznost zapisa je mnogo veća nego kod velikih. Na primer, kod malih brojeva možemo zapisivati veliki broj decimala, ali nijedan broj između 98 000 i 99 000 nije moguće zapisati (brojevi iz tog raspona bi se morali zaokružiti na neki od ova dva broja). Ovo nije preveliki problem, jer nam obično kod velikih brojeva preciznost nije toliko bitna koliko kod malih (matematički gledano, relativna greška koja nastaje usled zaokruživanja se ne menja puno kroz ceo raspon). Dve važne komponente koje karakterišu svaki zapis u pokretnom zarezu su raspon brojeva koji se mogu zapisati (u našem primeru to je bilo od oko 10^{-6} do oko 10^5) i on je skoro potpuno određen širinom eksponentna, kao i broj dekadnih značajnih cifara (u našem primer, imamo dve dekadne značajne cifre) i on je skoro potpuno određeno širinom mantise.

1.3 Zapis teksta

Međunarodna organizacija za standardizaciju, *ISO* (engl. *International Standard Organization*) definiše tekst (ili dokument) kao *„informaciju namenjenu ljudskom sporazumevanju koja može biti prikazana u dvodimenzionalnom obliku. Tekst se sastoji od grafičkih elemenata kao što su karakteri, geometrijski ili fotografski elementi ili njihove kombinacije, koji čine sadržaj dokumenta“*. Iako se tekst obično zamišlja kao dvodimenzioni objekat, u računarima se tekst predstavlja kao jednodimenzioni (linearni) niz karaktera koji pripadaju određenom unapred fiksiranom skupu karaktera. U zapisu teksta, koriste se specijalni karakteri koji označavaju prelazak u novi red, tabulator, kraj teksta i slično.

Osnovna ideja koja omogućava zapis teksta u računarima je da se svakom karakteru pridruži određen (neoznačeni) ceo broj (a koji se interno u računarima zapisuje binarno) i to na unapred dogovoreni način. Ovi brojevi se nazivaju *kodovima karaktera* (engl. *character codes*). Tehnička ograničenja ranih računara kao i neravnomeran razvoj računarstva između različitih zemalja, doveli su do toga da postoji više različitih standardnih tabela koje dodeljuju numeričke kodove karakterima. U zavisnosti od broja bitova potrebnih za kodiranje karaktera, razlikuju se 7-bitni kodovi, 8-bitni kodovi, 16-bitni kodovi, 32-bitni kodovi, kao i kodiranja promenljive dužine koja različitim karakterima dodeljuju kodove različite dužine. Tabele koje sadrže karaktere i njima pridružene kodove obično se nazivaju *kodne strane* (engl. *code page*).

Postoji veoma jasna razlika između karaktera i njihove grafičke reprezentacije. Elementi pisanog teksta koji najčešće predstavljaju grafičke reprezentacije pojedinih karaktera nazivaju se *glifovi* (engl. *glyph*), a skupovi glifova nazivaju se *fontovi* (engl. *font*). Korespondencija između karaktera i glifova ne mora biti jednoznačna. Naime, softver koji prikazuje tekst može više karaktera predstaviti jednim glifom (to su takozvane *ligature*, kao na primer glif za karaktere „f“ i „i“: fi), dok jedan isti karakter može biti predstavljen različitim glifovima u zavisnosti od svoje pozicije u reči. Takođe, moguće je da određeni fontovi ne sadrže glifove za određene karaktere i u tom slučaju se tekst ne prikazuje na željeni način, bez obzira što je ispravno kodiran. Fontovi koji

se obično instaliraju uz operativni sistem sadrže glifove za karaktere koji su popisani na takozvanoj *WGL4* listi (*Windows Glyph List 4*) koja sadrži uglavnom karaktere korišćene u evropskim jezicima, dok je za ispravan prikaz, na primer, kineskih karaktera, potrebno instalirati dodatne fontove. Specijalnim karakterima se najčešće ne pridružuju zasebni grafički likovi.

Englesko govorno područje. Tokom razvoja računarstva, broj karaktera koje je bilo poželjno kodirati je postajao sve veći. Pošto je računarstvo u ranim fazama bilo razvijano uglavnom u zemljama engleskog govornog područja, bilo je potrebno predstaviti sledeće karaktere:

- Mala slova engleskog alfabeta: `a, b, ... , z`
- Velika slova engleskog alfabeta: `A, B, ... , Z`
- Cifre `0, 1, ..., 9`
- Interpunkcijske znake: `, . : ; + * - _ ( ) [ ] { }` ...
- Specijalne znake: kraj reda, tabulator, ...

Standardne tabele kodova ovih karaktera su se pojavile još tokom 1960-ih godina. Najrasprostranjenije od njih su:

- *EBCDIC* - IBM-ov standard, korišćen uglavnom na mejnfrejm računarima, pogodan za bušene kartice.
- *ASCII* - standard iz koga se razvila većina danas korišćenih standarda za zapis karaktera.

ASCII. *ASCII (American Standard Code for Information Interchange)* je standard uspostavljen 1968. godine od strane *Američkog nacionalnog instituta za standarde*, (engl. *American National Standard Institute*) koji definiše sedmobitan zapis koda svakog karaktera što daje mogućnost zapisivanja ukupno 128 različitih karaktera, pri čemu nekoliko kodova ima dozvoljeno slobodno korišćenje. *ISO* takođe delimično definiše ASCII tablicu kao deo svog standarda *ISO 646 (US)*.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	NUL	STX	SOT	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Slika 1.2: ASCII tablica

Osnovne osobine ASCII standarda su:

- Prva 32 karaktera – od $(00)_{16}$ do $(1F)_{16}$ – su specijalni kontrolni karakteri.
- Ukupno 95 karaktera ima pridružene grafičke likove (engl. printable characters).
- Cifre 0-9 predstavljene su kodovima $(30)_{16}$ do $(39)_{16}$, tako da se njihov ASCII zapis jednostavno dobija dodavanjem prefiksa 011 na njihov binarni zapis.
- Kôdovi velikih i malih slova se razlikuju u samo jednom bitu u binarnoj reprezentaciji. Na primer, A se kodira brojem $(41)_{16}$ odnosno $(100\,0001)_2$, dok se a kodira brojem $(61)_{16}$ odnosno $(110\,0001)_2$. Ovo omogućava da se konverzija veličine slova u oba smera može vršiti efikasno.
- Slova su poređana u *kolacionu sekvencu*, u skladu sa engleskim alfabetom.

Različiti operativni sistemi predviđaju različito kodiranje oznake za prelazak u novi red. Tako operativni sistem Windows podrazumeva da se prelazak u novi red kodira sa dva kontrolna karaktera i to CR (carriage return) predstavljen kodom $(0D)_{16}$ i LF (line feed) predstavljen kodom $(0A)_{16}$, operativni sistem Unix i njegovi derivati (pre svega Linux) podrazumevaju da se koristi samo karakter LF, dok MacOS podrazumeva korišćenje samo karaktera CR.

Nacionalne varijante ASCII tablice i ISO 646. Tokom 1980-ih, *Jugoslovenski zavod za standarde* definisao je standard *YU-ASCII (YUSCII, JUS I.B1.002, JUS I.B1.003)* kao deo standarda ISO 646, tako što su kodovi koji imaju slobodno korišćenje (a koji u ASCII tabeli uobičajeno kodiraju zagrade i određene interpunkcijske znakove) dodeljeni našim dijakriticima:

YUSCII	ASCII	kôd	YUSCII	ASCII	kôd
Ž	@	$(40)_{16}$	ž	`	$(60)_{16}$
Š	[	$(5B)_{16}$	š	{	$(7B)_{16}$
Đ	\	$(5C)_{16}$	đ	|	$(7C)_{16}$
Ć	]	$(5D)_{16}$	ć	}	$(7D)_{16}$
Č	^	$(5E)_{16}$	č	~	$(7E)_{16}$

Osnovne mane YUSCII kodiranja su to što ne poštuje abecedni poredak, kao i to da su neke zagrade i važni interpunkcijski znaci izostavljeni.

8-bitna proširenja ASCII tabele. Podaci se u računaru obično zapisuju bajt po bajt. S obzirom na to da je ASCII sedmobitni standard, ASCII karakteri se zapisuju tako što se njihov sedmobitni kôd proširi vodećom nulom. Ovo znači da jednobajtni zapisi u kojima je vodeća cifra 1, tj. raspon od $(80)_{16}$ do $(FF)_{16}$ nisu iskorišćeni. Međutim, ni ovih dodatnih 128 kodova nije dovoljno da se kodiraju svi karakteri koji su potrebni za zapis tekstova na svim jezicima (ne samo na engleskom). Zbog toga je, umesto jedinstvene tabele koja bi proširivala ASCII na 256 karaktera, standardizovano nekoliko takvih tabela, pri čemu svaka od tabela sadrži karaktere potrebne za zapis određenog jezika odnosno određene grupe jezika. Praktičan problem je što postoji dvostruka standardizacija ovako kreiranih kodnih strana i to od strane ISO (International Standard Organization) i od strane značajnih korporacija, pre svega kompanije *Microsoft*.

ISO je definisao familiju 8-bitnih kodnih strana koje nose zajedničku oznaku *ISO/IEC 8859* (kodovi od $(00)_{16}$ do $(1F)_{16}$, $(7F)_{16}$ i od $(80)_{16}$ do $(9F)_{16}$ ostali su nedefinisani ovim standardom, iako se često u praksi popunjavaju određenim kontrolnim karakterima):

ISO-8859-1	Latin 1	većina zapadnoevropskih jezika
ISO-8859-2	Latin 2	centralno i istočnoevropski jezici
ISO-8859-3	Latin 3	južnoevropski jezici
ISO-8859-4	Latin 4	severnoevropski jezici
ISO-8859-5	Latin/Cyrillic	ćirilica većine slovenskih jezika
ISO-8859-6	Latin/Arabic	najčešće korišćeni arapski
ISO-8859-7	Latin/Greek	moderni grčki alfabet
ISO-8859-8	Latin/Hebrew	moderni hebrejski alfabet

Kompanija Microsoft definisala je familju 8-bitnih strana koje se označavaju kao *Windows-125x* (ove strane se nekada nazivaju i *ANSI*). Za srpski jezik, značajne su kodne strane:

Windows-1250	centralnoevropski i istočnoevropski jezici
Windows-1251	ćirilica većine slovenskih jezika
Windows-1252	(često se neispravno naziva i ANSI) zapadnoevropski jezici

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8																
9																
A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	

Slika 1.3: ISO-8859-1 tablica

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
8																
9																
A	NBSP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SHY	Ž	Ż
B	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	

Slika 1.4: ISO-8859-2 tablica

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
8	€		‚		„	…	†	‡		‰	Š	‹	Ś	Ť	Ž	Ź
9		‘	’	“	”	•	–	—		™	š	›	ś	ť	ž	ź
A		ˇ	˘	Ł	¤	Ą	¦	§	¨	©	Ş	«	¬		®	Ż
B	°	±	˛	ł	´	µ	¶	·	¸	ą	ş	»	Ľ	˝	ľ	ż
C	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	

Slika 1.5: Windows-1250 tablica

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
8																
9																
A	NBSP	Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ	SHY	Ў	Џ
B	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
C	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
D	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	№	ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ	§	ў	

Slika 1.6: ISO-8859-5 tablica

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
8	Ђ		‚		„	…	†	‡		‰	Љ	‹	Њ	Ќ	Ћ	Џ
9		‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
A		Ў	ў	J	¤	Ґ	¦	§	Ё	©	Є	«	¬		®	Ї
B	°	±	I	i	ґ	µ	¶	·	ё	№	є	»	j	S	s	ї
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	

Slika 1.7: Windows-1251 tablica

Unicode. Iako navedene kodne strane omogućavaju kodiranje tekstova koji nisu na engleskom jeziku, nije moguće, na primer, u istom tekstu koristiti i ćirilicu i latinicu. Takođe, za azijske jezike nije dovoljno 256 mesta za zapis svih karaktera. Pošto je kapacitet računara vremenom rastao, postepeno se krenulo sa standardizacijom skupova karaktera koji karaktere kodiraju sa više od jednog bajta. Kasnih 1980-ih, dve velike organizacije započele su standardizaciju tzv. univerzalnog skupa karaktera (engl. Universal Character Set — UCS). To su bili ISO, kroz standard *10646* i projekat *Unicode*, organizovan i finansiran uglavnom od strane američkih kompanija koje su se bavile proizvodnjom višejezičkog softvera (Xerox Parc, Apple, Sun Microsystems, Microsoft, ...).

ISO 10646 zamišljen je kao četvorobajtni standard. Prvih 65536 karaktera koristi se kao osnovni višejezički skup karaktera, dok je preostali prostor ostavljen kao proširenje za drevne jezike, naučnu notaciju i slično.

Unicode je za cilj imao da bude:

- univerzalan (UNIversal) — sadrži sve savremene jezike sa pismom;
- jedinstven (UNIque) — bez dupliranja karaktera - kodiraju se pisma, a ne jezici;
- uniforman (UNIform) — svaki karakter sa istim brojem bitova.

Početna verzija Unicode standarda svakom karakteru dodeljuje dvobajtni kôd (tako da kôd svakog karaktera sadrži tačno 4 heksadekadne cifre). Dakle, moguće je dodeliti kodove za ukupno $2^{16} = 65536$ različitih karaktera. S vremenom se shvatilo da dva bajta neće biti dovoljno za zapis svih karaktera koji se koriste na planeti, pa je odlučeno da se skup karaktera proširi i Unicode danas dodeljuje kodove od $(000000)_{16}$ do $(10FFFF)_{16}$ podeljenih u 17 tzv. ravni, pri čemu svaka ravan sadrži 65536 karaktera. U najčešćoj upotrebi je *osnovna višejezička ravan* (engl. *basic multilingual plane*) koja sadrži većinu danas korišćenih karaktera (uključujući i CJK — Chinese, Japanese, Korean — karaktere koji se najčešće koriste) čiji su kodovi između $(0000)_{16}$ i $(FFFF)_{16}$.

Vremenom su se pomenuta dva projekta UCS i Unicode združila i danas postoji izuzetno preklapanje između ova dva standarda.

U sledećoj tabeli je naveden raspored određenih grupa karaktera u osnovnoj višejezičkoj ravni:

0020-007E	ASCII printable
00A0-00FF	Latin-1
0100-017F	Latin extended A (osnovno proširenje latinice, sadrži sve naše dijakritike)
0180-027F	Latin extended B
...	
0370-03FF	grčki alfabet
0400-04FF	ćirilica
...	
2000-2FFF	specijalni karakteri
3000-3FFF	CJK (Chinese-Japanese-Korean) simboli
...	

Unicode standard u suštini predstavlja veliku tabelu koja svakom karakteru dodeljuje broj. Standardi koji opisuju kako se niske karaktera prevode u nizove bajtova se definišu dodatno.

UCS-2. ISO definiše UCS-2 standard koji svaki Unicode karakter osnovne višejezičke ravni jednostavno zapisuje sa odgovarajuća dva bajta.

UTF-8. Latinični tekstovi kodirani korišćenjem UCS-2 standarda sadrže veliki broj nula. Ne samo što te nule zauzimaju dosta prostora, već zbog njih softver koji je razvijen za rad sa dokumentima u ASCII formatu ne može da radi bez izmena nad dokumentima kodiranim korišćenjem UCS-2 standarda. *Unicode Transformation Format (UTF-8)* je algoritam koji svakom dvobajtnom Unicode karakteru dodeljuje određeni niz bajtova čija dužina varira od 1 do najviše 3. UTF je ASCII kompatibilan, što znači da se ASCII karakteri zapisuju pomoću jednog bajta, na standardni način. Konverzija se vrši na osnovu sledećih pravila:

raspon	binarno zapisan Unicode kôd	binarno zapisan UTF-8 kôd
0000-007F	00000000 0xxxxxxx	0xxxxxxx
0080-07FF	00000yyy yyxxxxxx	110yyyyy 10xxxxxx
0800-FFFF	zzzzyyyy yyxxxxxx	1110zzzz 10yyyyyy 10xxxxxx

Na primer, karakter A koji se nalazi u ASCII tabeli ima Unicode kôd $(0041)_{16} = (0000\,0000\ 0100\,0001)_2$, pa se na osnovu prvog reda prethodne tabele u UTF-8 kodiranju zapisuje kao $(01000001)_2 = (41)_{16}$. Karakter Š ima Unicode kôd $(0160)_{16} = (0000\,0001\ 0110\,0000)_2$. Na njega se primenjuje drugi red prethodne tabele i dobija se da je njegov UTF-8 zapis $(1100\,0101\ 1010\,0000)_2$ tj. $(C5A0)_{16}$. Opisani konverzioni algoritam omogućava da se čitanje samo početka jednog bajta odredi da li je u pitanju karakter zapisan korišćenjem jednog, dva ili tri bajta.

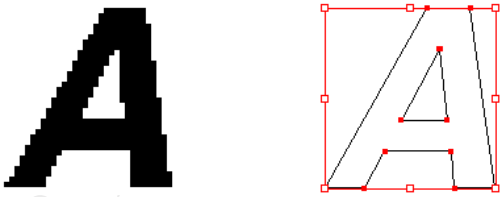
1.4 Zapis multimedijalnih sadržaja

Računari imaju sve veću ulogu u većini oblasti svakodnevnog života. Od mašina koje su pre svega služile za izvođenje vojnih i naučnih izračunavanja, računari su postali i sredstvo za kućnu zabavu (gledanje filmova, slušanje muzike), izvor informacija (internet) i nezaobilazno sredstvo u komunikaciji (elektronska pošta (engl. e-mail), ćaskanje (engl. chat, instant messaging), video konferencije, telefoniranje korišćenjem interneta (VoIP), ...). Nagli razvoj i proširivanje osnovne namene računara je posledica široke raspoloživosti velike količine multimedijalnih informacija (slika, zvuka, filmova, ...) koje su zapisane u digitalnom formatu. Sve ovo je posledica tehnološkog napretka koji je omogućio jednostavno i jeftino digitalizovanje signala, skladištenje velike količine digitalno zapisanih informacija, kao i njihov brz prenos i obradu.

1.4.1 Zapis slika

Slike se u računaru zapisuju koristeći *vektorski* zapis, *rasterski* zapis ili *kombinovani zapis*.

U vektorskom obliku, zapis slike sastoji se od konačnog broja geometrijskih figura (tačaka, linija, krivih, poligona), pri čemu se svaka figura predstavlja koncizno svojim koordinatama ili jednačinom u Dekartovoj ravni. Slike koje računari generišu često koriste vektorsku grafiku. Vektorski zapisane slike često zauzimaju manje prostora, dozvoljavaju uvećavanje (engl. zooming) bez gubitaka na kvalitetu prikaza i mogu se lakše preuređivati, s obzirom na to da se objekti mogu nezavisno jedan od drugoga pomerati, menjati, dodavati i uklanjati.



Slika 1.8: Primer slike u rasterskoj (levo) i vektorskoj (desno) grafici

U rasterskom zapisu, slika je predstavljena pravougaonom matricom komponenti koje se nazivaju pikseli (engl. pixel - PICture ELement). Svaki piksel je individualan i opisan jednom bojom. Raster nastaje kao rezultat digitalizacije slike. Rasterska grafika se još naziva i bitmapirana grafika. Uređaji za prikaz (monitori, projektori), kao i uređaji za digitalno snimanje slika (fotoaparati, skeneri) koriste rasterski zapis.

Modeli boja. Za predstavljanje crno-belih slika, dovoljno je boju predstaviti intenzitetom svetlosti. Različite količine svetlosti se diskretizuju u konačan broj nivoa osvetljenja čime se dobija određeni broj nijansi sive boje. Ovakav model boja se naziva *grayscale*. Ukoliko se za zapis informacije o količini svetlosti koristi 1 bajt, ukupan broj nijansi sive boje je 256. U slučaju da se slika predstavlja sa ukupno dve boje (na primer, crna i bela, kao kod skeniranog teksta nekog dokumenta) koristi se model pod nazivom *Duotone* i boja se tada predstavlja jednim bitom.

U RGB modelu boja, kombinovanjem crvene (R), zelene (G) i plave (B) komponente svetlosti reprezentuju se ostale boje. Tako se, na primer, kombinovanjem crvene i zelene boje reprezentuje žuta boja. Bela boja se reprezentuje maksimalnim vrednosti sve tri osnovne komponente, dok se crna boja reprezentuje minimalnim vrednostima osnovnih komponenti. U ovom modelu, zapis boje čini informacija o intenzitetu crvene, plave i zelene komponente. RGB model boja se koristi kod uređaja koji boje prikazuju kombinovanjem svetlosti (monitori, projektori, ...). Ukoliko se za informaciju o svakoj komponenti koristi po jedan bajt, ukupan broj bajtova za zapis informacije o boji je 3, te je moguće koristiti $2^{24} = 16777216$ različitih boja. Ovaj model se često naziva i *TrueColor* model boja.

Za razliku od aditivnog RGB modela boja, kod kojeg se bela boja dobija kombinovanjem svetlosti tri osnovne komponente, u štampi se koristi subtraktivni CMY (Cyan-Magenta-Yellow) model boje kod kojeg

se boje dobijaju kombinovanjem obojenih pigmenata na belom papiru. Kako se potpuno crna teško dobija mešanjem drugih pigmenata, a i kako je njena upotreba obično daleko najčešća, obično se prilikom štampanja uz CMY pigmente koristi i crni pigment čime se dobija model CMYK.

Za obradu slika pogodni su HSL ili HSV (poznat i kao HSB) model boja. U ovom modelu, svaka boja se reprezentuje Hue (H) komponentom (koja predstavlja ton boje), Saturation (S) komponentom (koja predstavlja zasićenost boje odnosno njenu „jarkost“) i Lightness (L), Value (V) ili Brightness (B) komponentom (koja predstavlja osvetljenost).

Formati zapisa rasterskih slika. Rasterske slike su reprezentovane matricom piksela, pri čemu se za svaki piksel čuva informacija o njegovoj boji. Dimenzije ove matrice predstavljaju tzv. *apsolutnu rezoluciju* slike. Apsolutna rezolucija i model boja koji se koristi određuju broj bajtova potrebnih za zapis slike. Na primer, ukoliko je apsolutna rezolucija slike 800×600 piksela, pri čemu se koristi RGB model boje sa 3 bajta po pikselu, potrebno je ukupno $800 \cdot 600 \cdot 3B = 1440000B \approx 1.373MB$ za zapis slike. Da bi se smanjila količina informacija potrebnih za zapis slike, koriste se tehnike kompresije i to *(i)* kompresije bez gubitka (engl. lossless), i *(ii)* kompresije sa gubitkom (engl. lossy). Najčešće korišćeni formati u kojima se koristi tehnike kompresije bez gubitka danas su GIF i PNG (koji se koriste za zapis dijagrama i sličnih računarski generisanih slika), dok je najčešće korišćeni format sa gubitkom JPEG (koji se obično koristi za fotografije).

1.4.2 Zapis zvuka

Zvučni talas predstavlja oscilaciju pritiska koja se prenosi kroz vazduh ili neki drugi medijum (tečnost, čvrsto telo). Digitalizacija zvuka se vrši merenjem i zapisivanjem vazdušnog pritiska u kratkim vremenskim intervalima. Osnovni parametri koji opisuju zvučni signal su njegova amplituda (koja odgovara „glasnoći“) i frekvencija (koja odgovara „visini“). Pošto ljudsko uho obično čuje raspon frekvencija od 20Hz do 20kHz, dovoljno je koristiti frekvenciju odabiranja 40kHz, tj. dovoljno je izvršiti odabiranje oko 40 000 puta u sekundi. Na primer, AudioCD standard koji se koristi prilikom snimanja običnih audio kompakt diskova, propisuje frekvenciju odabiranja 44.1kHz. Za postizanje još većeg kvaliteta, neki standardi (miniDV, DVD, digital TV) propisuju odabiranje na frekvenciji 48kHz. Ukoliko se snima ili prenosi samo ljudski govor (na primer, u mobilnoj telefoniji), frekvencije odabiranja mogu biti i znatno manje. Drugi važan parametar digitalizacije je broj bitova kojim se zapisuje svaki pojedinačni odbirak. Najčešće se koristi 2 bajta po odbirku (16 bitova), čime se dobija mogućnost zapisa $2^{16} = 65536$ različitih nivoa amplitude.

Da bi se dobio prostorni osećaj zvuka, primenjuje se tehnika višekanalnog snimanja zvuka. U ovom slučaju, svaki kanal se nezavisno snima posebnim mikrofonom i reprodukuje na posebnom zvučniku. *Stereo* zvuk podrazumeva snimanje zvuka sa dva kanala. *Surround* sistemi podrazumevaju snimanje sa više od dva kanala (od 3 pa čak i do 10), pri čemu se često jedan poseban kanal izdvaja za specijalno snimanje niskofrekvencijskih komponenti zvuka (tzv. bas). Najpoznatiji takvi sistemi su 5+1 gde se koristi 5 regularnih i jedan bas kanal.

Kao i slika, nekomprimovan zvuk zauzima puno prostora. Na primer, jedan minut stereo zvuka snimljenog u AudioCD formatu zauzima $2 \cdot 44100 \frac{sample}{sec} \cdot 60sec \cdot 2 \frac{B}{sample} = 10584000B \approx 10.1MB$. Zbog toga se koriste tehnike kompresije, od kojeg je danas najkorišćenija tehnika kompresije sa gubitkom MP3 (MPEG-1 Audio-Layer 3). MP3 kompresija se zasniva na tzv. psiho-akustici koja proučava koje je komponente moguće ukloniti iz zvučnog signala, a da pritom ljudsko uho ne oseti gubitak kvaliteta signala.

1.4.3 Zapis video sadržaja

Video zapis u računaru predstavlja kompleksan proces koji uključuje efikasno povezivanje slike i zvuka, uz primenu različitih tehnika kompresije i sinhronizacije. Video se sastoji od niza pojedinačnih *frejmova*, pri čemu je svaki frejm digitalna slika koja se kombinuje sa odgovarajućim segmentom zvučnog zapisa. *Koderi i dekoderi* (eng. codecs) igraju ključnu ulogu u ovom procesu. Koder je odgovoran za kombinovanje slike i zvuka u jedinstven *tok* (eng. stream) tokom snimanja, pri čemu se informacije komprimuju da bi se smanjila veličina fajla. Naime, i u zapisu videa, kao i zvuka javlja se redundacija podataka. Redundancija se odnosi na višak podataka koji nisu neophodni za reprodukciju, ali zauzimaju prostor i usporavaju prenos, te se uklanjanjem redundancije značajno povećava efikasnost.

Koderi (npr. *H.264, H.265, AV1*) su algoritmi koji vrše kompresiju video i audio podataka tokom snimanja. Koristi se tehnike poput vremenskog (eng. inter–frame) i prostornog (eng. intra–frame) kodiranja kako bi se smanjila redundancija. U toku ovog procesa posmatraju se uzastopni frejmovi i analiziraju sličnosti između njih. Koderi koriste različite algoritme, poput diskretne kosinusne transformacije (eng. Discrete Cosine Transform, DCT) i kompenzacije pokreta (eng. Motion Compensation) za smanjenje veličine podataka. Glavna ideja iza DCT je da se podaci rastave na različite frekvencije, pri čemu je većina u niskim frekvencijama. Niske frekvencije se mogu zadržati, dok se visoke frekvencije, koje predstavljaju manje značajne detalje, mogu delimično ili

potpuno ukloniti, čime se smanjuje veličina podataka bez značajnog gubitka vizuelnog kvaliteta. Kod tehnike kompenzacije pokreta glavna ideja je da se pokretni elementi u video zapisu efikasno predstave pomoću modela predviđanja. Umesto da se kompletan frejm kodira iznova, kodira se samo razlika između trenutnog i prethodnog frejma, koristeći vektore pokreta za opisivanje kretanja objekata. *H.265*, poznat i kao *HEVC*, koristi naprednije tehnike predviđanja i particionisanja kako bi omogućio efikasniju kompresiju u odnosu na stariji *H.264*, dok *AV1*, kao projekat otvorenog koda, donosi još bolju kompresiju.

Prilikom reprodukcije, dekoder razdvaja podatke i rekonstruiše originalne frejmove slike zajedno sa odgovarajućim audio segmentima. Koriste se sofisticirani algoritmi za dekompresiju i rekonstrukciju podataka, odvajanjem frejmova slike od odgovarajućih audio segmenata i pritom osiguravajući da se slika i zvuk *sinhronizovano* i precizno prikažu korisniku. Sinhronizacija između zvuka i slike postiže se pomoću *vremenskih oznaka* (eng. timestamp) koji osiguravaju da se svaki deo zvučnog zapisa reprodukuje u tačno definisanom trenutku tokom prikazivanja određenog frejma. Ovaj proces je ključan za postizanje prirodnog osećaja pokreta i zvuka, jer svaki nesklad može izazvati percepcijski disonant kod korisnika. Dekoderi koriste tehnike kao što su kompenzacija kretanja zasnova na blokovima (eng. Block-based Motion Compensation) i inverzna kosinusna transformacija (eng. Inverse Discrete Cosine Transform, IDCT) kako bi povratili originalni kvalitet slike i zvuka.

Moderni video formati kao što su *MP4*, *MKV*, ili *MOV* koriste tzv. kontejnere koji integrišu video i audio tokove, ali i druge podatke, kao što su prevodi (eng. title) ili metapodaci. Kontejneri služe kao *omoti* koji omogućavaju simultano čitanje i reprodukciju svih ovih tokova na uređajima za reprodukciju, osiguravajući sinhronizaciju između zvučnih i vizuelnih elemenata. Na ovaj način, tokom reprodukcije, dekoder u realnom vremenu analizira, odvaja i dekodira frejmove videa i odgovarajuće segmente zvučnog zapisa, koristeći sofisticirane algoritme za predviđanje i rekonstrukciju kako bi održao preciznu sinhronizaciju, čak i u slučajevima gubitka podataka ili varijabilnih brzina prenosa. *MP4* je široko podržan format i često se koristi za uživo puštanje sadržaja zbog balansa između kvaliteta i veličine fajla. *MKV* pruža veću fleksibilnost i može integrisati više tokova (npr. više audio zapisa ili titlova), dok *MOV* format, razvijen od strane kompanije Apple, nudi visok nivo kvaliteta i često se koristi u profesionalnim okruženjima za uređivanje videa.

Jedan od ključnih izazova u zapisu videa je balans između kvaliteta i efikasnosti. Pojmovi kao što su *HD*, *Full HD*, i *4K* označavaju različite rezolucije video zapisa koje direktno utiču na kvalitet slike. *HD (High Definition)* se odnosi na rezoluciju od 1280×720 piksela, dok *Full HD* ima rezoluciju od 1920×1080 piksela, što se često označava kao $1080p$ na platformama poput YouTube servisa. Ovo p označava progresivno skeniranje, što znači da se svaki frejm prikazuje u celosti, što doprinosi boljem kvalitetu slike. U manjim formatima ili u slučaju upletenog (eng. interlaced) skeniranja, frejmovi se ne prikazuju svi odjednom, već se deli na polja, gde se prvo prikazuje polovina slike, a zatim druga polovina, što može dovesti do smanjenja kvaliteta slike ili povećanja *treperenja*. Više rezolucije, kao što su *4K* (3840×2160 piksela), pružaju još veću oštrinu i detalje, ali zahtevaju i veći protok podataka i veću memoriju za skladištenje.

Brzina bita (brzina protoka, eng. bitrate), izražen u megabitima po sekundi (Mbps), označava količinu podataka koja se prenosi ili obrađuje u jedinici vremena tokom reprodukcije videa. Viša brzina bita rezultira boljim kvalitetom slike, ali takođe zahteva veći kapacitet za skladištenje i širu mrežnu propusnost, dok niži brzina bita može uzrokovati gubitak detalja i pojavu vizuelnih nepravilnosti u slici. Brzina bita je u korelaciji sa rezolucijom i korišćenim kodekom – veće rezolucije, kao što su *4K*, obično zahtevaju veću brzinu bita za očuvanje kvaliteta. Na primer, za *Full HD* ($1080p$) video često se preporučuje brzina bita između 8-12 Mbps, dok za *4K* rezoluciju je potrebno da brzina bita bude između 35-45 Mbps kako bi se očuvao visok nivo detalja i kvaliteta slike. Takođe, moderniji kodeci, kao što su *H.265* i *AV1*, omogućavaju bolju kompresiju i kvalitet slike pri nižoj brzini bita u odnosu na starije kodeke, kao što je *H.264*. Na primer, *H.264* za *Full HD* (1080p) kvalitet može zahtevati brzinu bita od 8-12 Mbps, dok *H.265* može postići sličan kvalitet pri 4-6 Mbps, a *AV1* može još više smanjiti potrebnu brzinu bita, omogućavajući sličan kvalitet slike pri brzina bita od 3-5 Mbps za *Full HD* (1080p), što ga čini pogodnim za reprodukovanje sadržaja preko mreža sa ograničenom propusnošću.

Kompresija sa gubicima (eng. lossy compression) koristi psihovizuelne modele kako bi eliminisala informacije koje ljudsko oko manje primećuje, čime se drastično smanjuje veličina fajla. Na primer, visoke frekvencije boja i svetlosti koje oko teško registruje često se odstranjuju iz zapisa, što omogućava efikasniju upotrebu memorije. Pored toga, adaptivne metode kodiranja kao što je kvantizacija omogućavaju dinamičko prilagođavanje nivoa detalja u zavisnosti od kompleksnosti scene, čime se optimizuje kompresija u realnom vremenu. Ove tehnike zajedno omogućavaju da moderni video zapisi zadrže visok nivo vizuelnog kvaliteta uz relativno mali prostor za skladištenje, što je od ključnog značaja za primenu u prenosu podataka i skladištenju na savremenim uređajima.

1.4.4 Jezici za obeležavavanje

Postoje dva osnovna pristupa za kreiranje multimedijalnih dokumenata: pristup "Ono što vidiš je ono što dobijaš"(WYSIWYG) i pristup pomoću jezika za označavanje. Metoda WYSIWYG omogućava korisnicima da odmah vide kako će finalni dokument izgledati, koristeći grafičke alate za uređivanje. Ovaj pristup je intuitivan

i jednostavan za korišćenje, što ga čini pogodnim za korisnike koji preferiraju vizuelne povratne informacije dok kreiraju sadržaj. Međutim, WYSIWYG uređivači ponekad mogu ograničiti nivo preciznosti i mogućnost prilagođavanja, posebno kada su u pitanju složene potrebe za formatiranjem.

Nasuprot tome, jezici za označavanje nude drugačiji način kreiranja dokumenata, naglašavajući strukturu i preciznost. Jezik za označavanje je sistem za anotiranje dokumenta na način koji razlikuje anotacije od stvarnog sadržaja. Te anotacije, ili 'oznake', pružaju strukturu i instrukcije za formatiranje neophodne za efektivno predstavljanje informacija. Jezici za označavanje omogućavaju autorima da sistematski organizuju sadržaj, odrede njegov vizuelni prikaz, pa čak i dodaju metapodatke za obradu. Oni su osnovni u kreiranju i predstavljanju različitih oblika sadržaja, od jednostavnih beleški i članaka do složenih naučnih dokumenata, knjiga i web stranica.

Ključna prednost upotrebe jezika za označavanje je nivo kontrole koji pružaju nad strukturom i prezentacijom dokumenta. Za razliku od WYSIWYG uređivača dokumenata, jezici za označavanje omogućavaju detaljno prilagodjavanje, što ih čini idealnim za složene dokumente gde je konzistentno formatiranje od suštinskog značaja. Korišćenjem ova dva pristupa, autori mogu odabrati onaj koji najbolje odgovara njihovim potrebama — WYSIWYG za jednostavnost i trenutni prikaz ili jezike za označavanje za fleksibilnost, preciznost i doslednost.

Jezici za označavanje imaju svoje korene u izdavačkoj industriji, gde su urednici koristili posebne notacije za označavanje instrukcija za formatiranje i slaganje teksta. Ovi koncepti su adaptirani u digitalne formate, počevši od SGML-a (Standardni Generalizovani Jezik za Označavanje), koji je služio kao meta-jezik za definisanje drugih jezika za označavanje. Rane verzije HTML-a su bile definisane unutar okvira SGML-a, čineći HTML jednom od prvih široko korišćenih primena SGML-a. Jezici za označavanje definisani unutar SGML-a često se nazivaju SGML aplikacijama, što naglašava fleksibilnost i proširivost koje je SGML pružio kao osnovu za razvoj jezika za označavanje. Ova evolucija je utrla put modernim jezicima kao što su HTML za web sadržaj i LaTeX za akademsko pisanje.

Opšta struktura jezika za označavanje

Jezici za označavanje razdvajaju logičku strukturu dokumenta od njegove vizuelne prezentacije. Logička struktura podrazumeva organizovanje dokumenta u manje jedinice, poput poglavlja i paragrafa, kao i označavanje naglašenih delova teksta, kao što su citati i definicije. Vizuelna prezentacija, s druge strane, uključuje definisanje stila, poput tipova fonta, veličina, razmaka između redova i boja koje se koriste u različitim delovima dokumenta.

Primer: Opšta struktura jezika za označavanje Evo primera koji pokazuje jednostavan jezik za označavanje za organizovanje kolekcije recepata:

```
<!DOCTYPE kolekcija SYSTEM "recipe-collection.dtd">
<kolekcija>
  <recept author="Marko Peric" title="Cokoladna torta">
    <sastojci>
      <sastojak>2 šolje brašna</sastojak>
      <sastojak>1 šolja šećera</sastojak>
      <sastojak>1/2 šolje kakao praha</sastojak>
    </sastojci>
    <instrukcije>
      <korak>Ugrejte rernu na 350°F (175°C).</korak>
      <korak>Pomešajte sve suve sastojke.</korak>
      <korak>Pecite 30 minuta.</korak>
    </instrukcije>
  </recepti>
</kolekcija>
```

U ovom primeru su prikazani osnovni koncepti jezika za označavanje: elementi, oznake (tagovi) i atributi. Elementi, kao što su `<recipe>` i `<ingredients>`, predstavljaju logičke delove dokumenta, dok atributi, kao što su author i title, pružaju dodatne informacije o elementima.

Definicija tipa dokumenta (`DOCTYPE`) Jedan važan aspekt jezika za označavanje je definicija tipa dokumenta, koja osigurava da dokument poštuje određenu strukturu i pravila. Svaki jezik za označavanje koji se bazira na SGML-u (Standardni Generalizovani Jezik za Označavanje) koristi `DOCTYPE` deklaraciju da bi specificirao tip dokumenta koji se kreira. DOCTYPE deklaracija pruža informacije o Definiciji tipa dokumenta (`DTD`), koja definiše dozvoljene elemente, njihove odnose i atribute.

Na primer:

```
<!DOCTYPE collection SYSTEM "recipe-collection.dtd">
```

Ova deklaracija ukazuje na to da je tip dokumenta kolekcija, a njegova struktura je definisana fajlom `recipe-collection.dtd`. `DTD` specificira koje oznake (tagovi) su dozvoljene, njihova značenja i kako mogu biti ugnježdene jedna u drugu.

Ispod je detaljniji primer DTD pravila za definisanje strukture dokumenta:

```
<!ELEMENT kolekcija - - (recept+)>
<!ELEMENT recept - - (sastojci, instrukcije)>
<!ELEMENT sastojci - O (sastojak+)>
<!ELEMENT instrukcije - O (korak+)>
<!ELEMENT sastojak - O (#PCDATA)>
<!ELEMENT korak - O (#PCDATA)>
```

U ovom primeru, `<ELEMENT>` se koristi za definisanje elemenata kao što su kolekcija, recept, sastojci i instrukcije. Simbol + označava da se element mora pojaviti bar jednom, dok `#PCDATA` označava parsirane karaktere, što znači da je tekst dozvoljen unutar tog elementa.

Jezici za označavanje koji koriste SGML, kao što je HTML, definišu sopstveni skup oznaka, njihovo značenje i moguće odnose među njima. Ova specifikacija tipa dokumenta određuje sintaksu i strukturu dokumenta, osiguravajući da se dokument kreira na osnovu unapred definisanih pravila.

Pregled različitih jezika za označavanje

Uobičajeni jezici za označavanje: Najšire korišćeni jezici za označavanje su HTML, LaTeX, Markdown i XML. Svaki od njih služi različitoj svrsi u zavisnosti od konteksta u kojem se koriste.

HTML (HyperText Markup Language): Pretežno se koristi za strukturiranje web stranica i web aplikacija. Definiše osnovne elemente web stranice, kao što su naslovi, paragrafi, linkovi, slike i drugo. HTML je osnova za web sadržaj i često se kombinuje sa CSS-om i JavaScript-om kako bi se kreirale vizuelno privlačne i interaktivne stranice.

LaTeX: Sistem za pripremu dokumenata koji se najčešće koristi za slaganje složenih dokumenata, naročito u akademskim i naučnim radovima. Odlikuje se izvanrednim formatiranjem matematičkih formula, naučnih notacija i bibliografija. Za razliku od programa za obradu teksta, LaTeX osigurava dosledno formatiranje, što je ključno za tehničke radove i disertacije.

Markdown: Lagan jezik za označavanje koji je jednostavan za pisanje i čitanje. Markdown se često koristi za kreiranje dokumentacije, README fajlova i blog postova. Njegova jednostavna sintaksa omogućava korisnicima brzo formatiranje teksta pomoću plain-text editora, čineći ga pristupačnim i tehničkim i netehničkim korisnicima.

XML (eXtensible Markup Language): Dizajniran za skladištenje i prenos podataka, XML se fokusira na strukturu i semantiku podataka, a ne na njihovu prezentaciju. XML se koristi u različitim aplikacijama, uključujući web servise, konfiguracione fajlove i razmenu podataka između sistema.

HTML (HyperText Markup Language)

Kratka istorija. Poreklo HTML-a datira iz ranih dana interneta 1991. godine. Razvoj HTML-a napredovao je prilično haotično (različite verzije HTML-a su razvijane nekoordinisano od strane različitih pojedinaca i grupa, što je dovelo do nedoslednosti). Od verzije 3.2, upravljanje HTML-om je preuzeo World Wide Web Konzorcijum (W3C) (http://www.w3c.org/). Trenutna verzija standarda je HTML5 (HTML 5.2).

HTML (HyperText Markup Language) je standardni jezik za kreiranje i strukturiranje web stranica. Na početku HTML dokumenta nalazi se DOCTYPE deklaracija koja označava verziju standarda koja se koristi; za HTML5, to se piše kao <!DOCTYPE html>. HTML dokument se sastoji od elemenata kao što su ceo dokument, paragrafi, tabele, slike i drugi.

Osnovni elementi. Elementi se označavaju pomoću tagova, koji se nazivaju i oznake ili markeri, i koji označavaju početak i kraj elementa. Većina elemenata ima početni tag u obliku <ime-elementa> i završni tag u obliku </ime-elementa>, koji obuhvataju sadržaj elementa.

Neki elementi nemaju sadržaj, i za njih se koriste samougašeni tagovi, kao što je `<ime-elementa/>`.

Elementi takođe mogu imati atribute koji pružaju dodatne informacije o njima. Atributi se navode unutar početnog taga u obliku ime-attributa="vrednost-attributa".

Primer: Kreiranje jednostavne web stranice HTML se koristi za kreiranje osnovnih delova web stranice. Evo osnovnog primera HTML dokumenta:

```
<!DOCTYPE html>
<html>
  <head><title>Moja prva stranica</title></head>
  <body>
    <h1>Dobrodošli!</h1>
    <p>Ovo je primer HTML označavanja.</p>
  </body>
</html>
```

Kompletna web stranica u HTML-u je predstavljena jednim `html` elementom. Sadržaj `html` elementa sastoji se od dva glavna dela `head` i `body`. Sekcija `<head>` pruža važne informacije i linkove koji utiču na ceo dokument, dok sekcija `<body>` sadrži sadržaj koji korisnici vide na stranici.

`head`: Sekcija head sadrži meta-informacije o dokumentu, kao što su naslov, skup karaktera i linkovi ka spoljnim resursima poput CSS-a za stilizovanje ili JavaScript-a za funkcionalnost.

```
<head>
  <title>My First Page</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
  </head>
```

`<title>`: Određuje naslov web stranice, koji se prikazuje na kartici pregledača i koriste ga pretraživači. Precizan i relevantan naslov doprinosi boljem rangiranju stranice u rezultatima pretrage.

Meta tagovi, `<meta>` se koriste u sekciji `<head>` HTML dokumenta za pružanje metapodataka – informacija o web stranici, kao što su kodiranje karaktera, opis, ključne reči i postavke prikaza (eng. viewport). Oni igraju ključnu ulogu u SEO optimizaciji, mobilnoj prilagodljivosti i pravilnom prikazu specijalnih karaktera. Još jedan primer meta taga je `<meta name="viewportčontent="width=device-width, initial-scale=1.0»`, koji se koristi za kontrolu izgleda na mobilnim pregledačima i osigurava da stranica bude responzivna na različitim uređajima.

Tag `<link>` u HTML-u koristi se za povezivanje sa spoljnim resursima, kao što su stilovi ili fontovi. Najčešće se postavlja u sekciju `<head>` HTML dokumenta. Tag `<link>` je ključan za uključivanje stilova (npr. CSS fajlova), ikonica i drugih resursa koji unapređuju izgled i funkcionalnost web stranice. To je samougašeni tag i ne zahteva završni `</link>` tag. U gornjem primeru dat je link ka eksternom CSS fajlu koji definise stil stranice, a u drugom je uključen Google font Roboto koji će biti korišćen na stranici. Na taj način osiguravamo da se stranica prikazuje pravilno i zadržava isti izgled, čak i ako font Roboto nije instaliran na datom računaru, što omogućava dosledan prikaz bez obzira na uređaj sa kog se stranica pregleda.

`body`: Sekcija body sadrži sav sadržaj koji se prikazuje na stranici, kao što su tekst, slike, tabele i linkovi. U narednom delu biće prikazani najčešće korišćeni tagovi u okviru sekcije body.

Elementi za naslove. HTML obezbeđuje šest nivoa elemenata za naslove, od `<h1>` do `<h6>`. Ovi elementi se koriste za definisanje naslova različite važnosti.

- `<h1>` se koristi za najvažniji naslov, obično glavni naslov stranice.
- `<h2>` do `<h6>` se koriste za podnaslove, pri čemu je `<h6>` najmanje važan.

Evo primera kako se ovi tagovi za naslove mogu koristiti:

```
<body>
  <h1>Glavni naslov stranice</h1>
  <h2>Naslov sekcije</h2>
  <h3>Naslov podsekcije</h3>
  <h4>Manji naslov podsekcije</h4>
  <h5>Još manji naslov</h5>
  <h6>Najmanje važan naslov</h6>
</body>
```

Ovi naslovi pomažu u organizaciji sadržaja web stranice, čineći strukturu stranice jasnijom za korisnike. Takođe su važni za pretraživače i pristupačnost, jer pružaju jasnu hijerarhiju sadržaja.

Element paragrafa. Element `<p>` koristi se za definisanje paragrafa u HTML dokumentu. Paragrafi su blokovi teksta odvojeni od ostalog sadržaja, što olakšava čitanje i organizaciju informacija na stranici.

Evo primera kako se <p> tag može koristiti:

```
<body> <p>Ovo je prvi paragraf na web stranici. On pruža uvod
u sadržaj koji je predstavljen.</p> <p>Ovo je još jedan paragraf, koji se
može koristiti za detaljnije objašnjenje, davanje primera ili pružanje
dodatnih informacija.</p> </body>
```

Element `<p>` pomaže u razdvajanju teksta na delove, poboljšavajući čitljivost i olakšava korisnicima da lakše prate informacije. Često se paragrafi obeležavaju različitim stilovima korišćenjem css-a, što dodatno povećava čitljivost.

Elementi liste. HTML nudi različite vrste elemenata za organizovanje sadržaja u liste:

- `<ul>` **(Neuređena lista)**: Ovaj element se koristi za kreiranje lista gde redosled stavki nije bitan. Stavke su obično označene tačkama.
- `<ol>` **(Uređena lista)**: Ovaj element se koristi za kreiranje lista gde je redosled stavki važan. Stavke su numerisane brojevima, slovima, rimskim brojevima i slično.
- `<dl>` **(Lista opisa)**: Ovaj element se koristi za kreiranje lista termina i njihovih opisa.

Evo primera kako se ovi elementi lista mogu koristiti:

```
<h2>Primer neuređene liste</h2>
<ul>
  <li>Stavka 1</li>
  <li>Stavka 2</li>
  <li>Stavka 3</li>
</ul>

<h2>Primer uređene liste</h2>
<ol>
  <li>Prvi korak</li>
  <li>Drugi korak</li>
  <li>Treći korak</li>
</ol>

<h2>Primer liste opisa</h2>
<dl>
  <dt>HTML</dt>
  <dd>Jezik za označavanje za kreiranje web stranica.</dd>
  <dt>CSS</dt>
  <dd>Jezik stilskih tablica koji se koristi za opisivanje izgleda i formatiranja HTML dokumenta.</dd>
</dl>
```

Elementi `<ul>` i `<ol>` pomažu u organizaciji sadržaja u obliku tačkica ili numerisanih lista, čineći sekvence ili kolekcije stavki lakšim za praćenje. Element `<dl>` se koristi za davanje definicija ili objašnjenja, što ga čini korisnim za bilo koju situaciju gde treba povezati termine sa opisima.

Tekstualni elementi. HTML nudi razne tekstualne elemente za naglašavanje ili promenu prikaza teksta:

- `<i>`: Koristi se za italic stil teksta, često za strane reči ili tehničke izraze.
- `<b>`: Koristi se za podebljanje teksta, obično radi isticanja ključnih reči ili važnih fraza.
- `<u>`: Podvlači tekst. Iako se ranije često koristio u HTML-u, danas se koristi ređe jer CSS pruža bolje opcije za stilizovanje.
- `<em>`: Naglašava tekst, obično prikazan u italic stilu. Takođe ukazuje na naglasak ili važnost za čitače ekrana.
- `strong>`: Označava jako naglašavanje, obično prikazano podebljano. Koristi se za označavanje važnosti i dostupno je čitačima ekrana.
- `<small>`: Koristi se za prikaz manjeg teksta, često za napomene ili sitno štampane informacije.
- `<br>`: Ubacuje prelazak u novi red. Za razliku od tagova za paragraf, koristi se za početak novog reda bez kreiranja novog bloka teksta.

- `Komentari (<!- komentar ->)`: Koriste se za dodavanje beleški unutar HTML koda koje nisu prikazane u pregledaču. Komentari su korisni za dokumentaciju koda ili privremeno onemogućavanje delova HTML-a.

Evo primera kako se ovi tagovi mogu koristiti:

```
<p>Ovo je paragraf sa <i>italic</i>, <b>podebljanim</b> i
<u>podvučenim</u> tekstom.</p> <p><em>Naglašen tekst</em> i <strong>snažan
tekst</strong> su važni za pristupačnost i prenošenje značenja.</p>
<p><small>Ovo je napomena u malom tekstu.</small></p> <p>Ovo je red
teksta.<br>Ovo je sledeći red, napravljen korišćenjem preloma reda.</p> <!--
Ovo je komentar koji neće biti vidljiv na prikazanoj stranici -->
```

različitih nivoa naglaska ili važnosti, poboljšavajući vizuelnu strukturu i pristupačnost web stranice.

Linkovi. Linkovi ili hiperveze koriste se za povezivanje dva resursa na internetu. Hiperveza je element na web stranici koji korisnici mogu aktivirati, obično klikom, što pretraživaču nalaže da učita novu stranicu ili resurs. Linkovi su opisani pomoću `<a>` elementa, a sadržaj ovog elementa predstavlja klikabilno područje koje aktivira link.

Atribut `href` koristi se za određivanje URL-a resursa koji treba da se prikaže kada se link aktivira.

```
<a href="http://www.matf.bg.ac.rs">Matematički fakultet,
Beograd</a>
```

Podrazumevano, linkovi se otvaraju u istoj kartici, ali može se koristiti atribut `target="\_blank"` kako bi se otvorio link u novoj kartici.

```
<a href="http://www.matf.bg.ac.rs"
target="_blank">Matematički fakultet, Beograd (otvori u novoj kartici)</a>
```

Atribut `href` može sadržati apsolutnu ili relativnu adresu. **Apsolutni URL** uključuje kompletnu URL adresu, počevši od protokola kao što je `http://` ili `https://`.

```
<a href="http://www.example.com">Poseti Example</a>
```

Relativni URL se koristi za linkove unutar istog sajta i ne uključuju protokol niti naziv domena.

```
<a href="/about.html">O nama</a>
```

se poziva na putanje relativne u odnosu na trenutnu stranicu ili osnovni URL.

Moguće je i povezivanje na specifičan deo web stranice pomoću **fragment identifikatora**. Ovi identifikatori se definišu korišćenjem atributa `id` na elementu.

```
<h2 id="kontakt">Kontakt</h2>
...
<a href="#kontakt">Idi na sekciju Kontakt</a>
```

sa id-jem kontakt.

Hiperveze su osnovna funkcionalnost interneta, omogućavajući korisnicima da lako prelaze između povezanih resursa.

Tabele. Tabele se u HTML-u koriste za prikaz podataka u strukturiranom tabelarnom formatu. Sastoje se od redova i kolona, što olakšava organizovano prikazivanje informacija. Tabela se definiše pomoću elementa `<table>`, dok se redovi definišu pomoću elementa `<tr>`, a ćelije se predstavljaju pomoću `<td>` (standardna ćelija) za obične ćelije ili `<th>` (zaglavlje) za ćelije zaglavlja.

```
<table border="1">
  <tr>
    <th>Stavka</th>
    <th>Količina</th>
  </tr>
  <tr>
    <td>Jabuke</td>
    <td>10</td>
  </tr>
  <tr>
```

```
    <td>Narandže</td>
    <td>15</td>
  </tr>
</table>
```

definiše red, `<th>` se koristi za ćelije zaglavlja (koje su podrazumevano prikazane podebljano i centrirano), a `<td>` se koristi za standardne ćelije podataka.

Atribut tabele `border` može se koristiti za dodavanje ivice tabeli. U gornjem primeru, `border="1"` dodaje ivicu širine 1 piksel tabeli.

Za spajanje više kolona ili redova, možete koristiti atribute `colspan` i `rowspan`, respektivno.

```
    <table border="1">
    <tr>
      <th colspan="2">Zalihe voća</th>
    </tr>
    <tr>
      <th>Stavka</th>
      <th>Količina</th>
    </tr>
    <tr>
      <td>Jabuke</td>
      <td>10</td>
    </tr>
  </table>
```

U ovom primeru, atribut `colspan="2"` koristi se da zaglavna ćelija obuhvati dve kolone.

```
<table border="1">
  <tr>
    <th rowspan="2">Kategorija</th>
    <th>Stavka</th>
    <th>Količina</th>
  </tr>
  <tr>
    <td>Jabuke</td>
    <td>10</td>
  </tr>
  <tr>
    <td>Voće</td>
    <td>Narandže</td>
    <td>15</td>
  </tr>
</table>
```

Ovde, `rowspan="2"` koristi se da zaglavlje "Kategorija"obuhvati dva reda.

Element `<caption>` koristi se za dodavanje naslova tabeli, koji se podrazumevano prikazuje iznad nje.

```
<table border="1">
  <caption>Zalihe voća</caption>
  <tr>
    <th>Stavka</th>
    <th>Količina</th>
  </tr>
  <tr>
    <td>Jabuke</td>
    <td>10</td>
  </tr>
</table>
```

Ovaj naslov pruža tabeli objašnjenje, pomažući korisnicima da lakše razumeju šta predstavljaju podaci.

Slike. Web stranice takođe mogu uključivati multimedijalni sadržaj kao što su slike, audio i video klipovi. Za umetanje slika koristi se `<img>` element, koji je prazan tag (tj. nema završni tag).

Atribut `src` predstavlja URL slike i obavezan je. Preporučuje se korišćenje relativnih putanja za slike kada je to moguće.

Atribut `alt` pruža alternativni tekst koji će se prikazati ako pregledač ne može da prikaže sliku. Ovaj atribut je važan za pristupačnost, jer omogućava čitačima ekrana da opišu sadržaj slike korisnicima sa oštećenjem vida.

```
<img src="images/competition.jpg" alt="Učesnici na matematičkom takmičenju" title="Matematičko takm
```

Atribut `title` pruža prikaz teksta koji se pojavljuje kada korisnik pređe mišem preko slike. Atributi `width` i `height` definišu dimenzije slike, koje mogu biti postavljene u pikselima ili kao procenat u odnosu na kontejner elementa. Preporučuje se da ove vrednosti odgovaraju stvarnim dimenzijama slike kako bi se izbegli problemi sa skaliranjem.

Spoljni sadržaj. Osim slika, HTML omogućava ugrađivanje drugih tipova multimedijalnog sadržaja, kao što su YouTube video zapisi ili Google Mape, pomoću `<iframe>` taga.

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ" title="YouTube video p
```

omogućavajući korisnicima da reprodukuju video bez napuštanja sajta.

Primer ugrađivanja Google Mapa:

```
<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3151.8354345093644!2d144.955651315899
```

Ovaj primer ugrađuje interaktivnu Google Mapu, omogućavajući korisnicima da direktno na stranici pregledaju lokacije.

Opšti elementi. HTML pruža dva opšta elementa koja se često koriste za grupisanje sadržaja bez dodavanja posebnog značenja: `<div>` i `<span>`.

Element `<div>` je blokovski kontejner koji se koristi za grupisanje većih delova sadržaja. Često se koristi za obuhvatanje sekcija web stranice, kao što su navigacioni meniji, bočne trake ili čitavi segmenti stranice. Posebno je koristan kada se kombinuje sa globalnim atributima id i class kako bi se stilizovale ili manipulisale te sekcije pomoću CSS-a ili JavaScript-a.

```
<div id="galerija">
  <h2>Galerija slika</h2>
  <img src="images/photo1.jpg" alt="Fotografija 1" />
  <img src="images/photo2.jpg" alt="Fotografija 2" />
  <img src="images/photo3.jpg" alt="Fotografija 3" />
</div>
```

U ovom primeru, `<div>` element se koristi za grupisanje skupa slika i njihovog naslova, omogućavajući lako stilizovanje i kontrolu galerije kao celine.

Element `<span>` je jednolinijski kontejner koji se koristi za grupisanje manjih delova teksta ili drugih jednolinijiskih elemenata. Često se koristi kada je potrebno stilizovati ili dodati drugačije ponašanje određenom delu teksta bez stvaranja novog bloka.

```
<p>Glavne komponente računara su <span class="highlight">procesor</span>, <span class="highlight">m
```

Ovde se `<span>` koristi za isticanje pojedinačnih komponenti unutar paragrafa. Atribut `class="highlight"` može se koristiti za primenu specifičnog stila (npr. promenu boje teksta) na ove termine pomoću CSS-a.

HTML je ključan za kreiranje strukture web stranica. U kombinaciji sa CSS-om za stilizovanje (iako CSS nije jezik za označavanje, već jezik stilskih tablica koji se koristi za primenu stilova, kao što su boje, fontovi i raspored, kako bi se poboljšao izgled tog sadržaja) i JavaScript-om za interaktivnost, HTML služi kao osnova za sav web razvoj.

LaTeX

Kratka istorija. Poreklo LaTeX-a datira iz ranih 1980-ih. Razvio ga je Lesli Lamport (Leslie Lamport) na osnovu sistema za slaganje teksta TeX Donalda Knuta (Donald Knuth), s ciljem da obezbedi jednostavniji način za kreiranje kvalitetno formatiranih dokumenata. LaTeX je brzo stekao popularnost u akademskim krugovima, posebno u matematici, računarstvu i inženjerstvu, zbog svoje sposobnosti da obradi složene sadržaje kao što su matematičke formule i bibliografije. Danas je LaTeX de facto standard za akademsko izdavaštvo u mnogim naučnim oblastima, zahvaljujući preciznosti u formatiranju i snažnim mogućnostima za upravljanje referencama i jednačinama.

Kako se .tex dokument prevodi Da bi se .tex dokument preveo u čitljiv format (kao što je PDF), potrebno je koristiti LaTeX kompajler. Najčešće korišćeni alati za prevođenje su: - **pdflatex**: Direktno prevodi .tex fajl u PDF format. Na primer, u terminalu možete pokrenuti:

```
pdflatex naziv_fajla.tex
```

Ova komanda će generisati PDF fajl iz vašeg LaTeX dokumenta. - **xelatex** ili **lualatex**: Ovi kompajleri omogućavaju korišćenje Unicode karaktera i jednostavan rad sa različitim fontovima. Pokreću se slično kao pdflatex.

Takođe postoje integrisana razvojna okruženja kao što su **TeXShop**, **TeXworks**, ili **Overleaf** (online alat), koji omogućavaju jednostavno uređivanje i kompajliranje LaTeX dokumenata sa samo nekoliko klikova.

Osnovni elementi. LaTeX se koristi za generisanje visokokvalitetnih dokumenata koristeći jednostavnu sintaksu za definisanje elemenata. Dokument započinje deklaracijom tipa dokumenta kao što je `\documentclass` koja definiše osnovne karakteristike dokumenta, npr. članak, knjigu ili prezentaciju.

```
\documentclass{article}
  \title{Moj prvi LaTeX dokument}
  \author{Danijela Simic}
  \date{\today}

\begin{document}

  \maketitle

  Dobrodošli u svet LaTeX-a!
\end{document}
```

U ovom primeru kreiramo osnovni dokument sa naslovom, autorom i datumom, koji će biti automatski generisan.

- `\documentclass{article}`: Ova komanda definiše tip dokumenta. Umesto `article`, mogu se koristiti i drugi tipovi kao što su `report` (izveštaj), `book` (knjiga), ili `beamer` (prezentacija). Mogu se dodati i opcije poput veličine fonta (npr. `12pt`, `11pt`, `10pt`) ili formata (npr. `twocolumn`, `landscape`). Primer:

```
\documentclass[12pt, a4paper]{article}
```

Ovde definišemo veličinu fonta od 12pt na A4 papiru.

- ** **: Postavlja naslov dokumenta. - ** **: Definiše autora dokumenta. - ** **: Postavlja datum dokumenta. Komanda `7. novembar 2024.` automatski koristi današnji datum, ali možete navesti i specifičan datum, npr. . - **

1. januar 2024

: Ova komanda generiše stranicu sa naslovom, autorom i datumom na osnovu prethodno definisanih vrednosti. - **`documentclass`: Kao što je već pomenuto, ova komanda definiše tip dokumenta, što utiče na osnovni izgled i ponašanje dokumenta. - **Veličina fonta i format stranice**: Možete odrediti veličinu fonta (npr. `10pt`, `12pt`) i format stranice (npr. `a4paper`, `letterpaper`) unutar `documentclass` opcija. Primer:

```
\documentclass[12pt, a4paper]{article}
```

Ovde se podešava veličina fonta i format papira. - **Paketi**: Nakon `documentclass` linije, možete koristiti `usepackage` komande da uključite različite pakete koji proširuju funkcionalnost LaTeX-a. Na primer:

```
\usepackage{graphicx}  % Za rad sa slikama
\usepackage{amsmath}   % Za napredne matematičke formule
```

- **Podešavanje fontova i stilova**: Paket `fontspec` (u XeLaTeX i LuaLaTeX) omogućava korišćenje različitih fontova, dok osnovni LaTeX koristi standardne fontove.

Kada su sva podešavanja obavljena, započinje glavni deo dokumenta sa `begindocument`. Sve što se nalazi unutar `begin document` i `enddocument` predstavlja sadržaj dokumenta, uključujući tekst, slike, matematičke formule, tabele i ostalo.

Struktura dokumenta. Dokumenti u LaTeX-u su strukturirani pomoću različitih nivoa naslova: `section`, `subsection`, `subsubsection`, itd. Ovo omogućava hijerarhijsko organizovanje sadržaja, što olakšava preglednost i navigaciju.

```
\section{Uvod}
Ovo je uvodni deo dokumenta.

\subsection{Motivacija}
Ovde objašnjavamo zašto koristimo LaTeX.
```

Liste. LaTeX podržava neuređene i uređene liste koje olakšavaju organizaciju sadržaja.

```
\begin{itemize}
    \item Prva stavka.
    \item Druga stavka.
\end{itemize}

\begin{enumerate}
    \item Prva stavka.
    \item Druga stavka.
\end{enumerate}
```

Tabele. Kreiranje tabela u LaTeX-u omogućava organizovano predstavljanje podataka.

```
\begin{table}[h]
  \centering
  \begin{tabular}{|c|c|}
    \hline
    Stavka & Količina \\
    \hline
    Jabuke & 10 \\
    Narandže & 15 \\
    \hline
  \end{tabular}
  \caption{Zalihe voća}
  \label{tab:zalihe}
\end{table}
```

Slike. Slike se u LaTeX-u uključuju pomoću paketa `graphicx` i naredbe `\includegraphics`. Preporučuje se da slike budu u PDF, PNG ili JPEG formatu.

```
\usepackage{graphicx}
\begin{figure}[h]
  \centering
  \includegraphics[width=0.5\textwidth]{slika.png}
  \caption{Primer slike}
  \label{fig:primer_slika}
\end{figure}
```

Tekstualni elementi. LaTeX omogućava formatiranje teksta korišćenjem naredbi kao što su za podebljavanje, za kurziv, i za podvlačenje. Komentari se mogu dodati korišćenjem znaka procenta (%), koji označava da ostatak linije neće biti procesuiran.

```
\textbf{Ovo je podebljani tekst}, dok je \textit{ovo kurziv}.
% Ovo je komentar u LaTeX kodu
```

LaTeX je moćan alat za stvaranje tehničkih i naučnih dokumenata, nudeći visoku preciznost i fleksibilnost pri formatiranju matematičkih formula, tabela, slika i složenih struktura teksta. U kombinaciji sa BibTeX-om za citiranje i različitim paketima za napredne funkcionalnosti, LaTeX ostaje ključni alat za akademsko i naučno pisanje.

Matematičke formule. Jedna od glavnih prednosti LaTeX-a je jednostavno i efektivno pisanje matematičkih formula. Matematički izrazi mogu biti ubačeni unutar teksta koristeći dvostruke zagrade `$ ... $` ili kao zasebni blok koristeći

...

.

Primeri matematičkih formula:

- Jednostavna inline formula: `$ a^2 + b^2 = c^2 $`
- Zasebna formula:

```
\[
  E = mc^2
\]
```

- Složena formula sa integracijom:

```
\begin{equation}
  \int_a^b x^2 \, dx = \frac{b^3 - a^3}{3}
\end{equation}
```

- Matrica reda 2×2:

```
\[
  \begin{bmatrix}
    a & b \\
    c & d
  \end{bmatrix}
\]
```

- Formula za sumu:

```
\[
  \sum_{i=1}^n i = \frac{n(n+1)}{2}
\]
```

- Diferencijalna jednačina:

```
\begin{equation}
  \frac{d^2y}{dx^2} + 3\frac{dy}{dx} + 2y = 0
\end{equation}
```

- Integral sa trigonometrijom:

```
\begin{equation}
  \int_0^{\pi} \sin(x) \, dx = 2
\end{equation}
```

- Razlomak:

```
\[
  \frac{a+b}{c+d}
\]
```

- Složena suma:

```
\[
  \sum_{i=1}^n \frac{1}{i^2}
\]
```

- Proizvod:

```
\[
  \prod_{i=1}^n i = n!
\]
```

- Limiti:

```
\begin{equation}
  \lim_{x \to 0} \frac{\sin x}{x} = 1
\end{equation}
```

Kreiranje sadržaja. Da bi se kreirao sadržaj u LaTeX dokumentu, koristi se komanda `\tableofcontents`. Ova komanda se postavlja na mesto u dokumentu gde želite da se sadržaj prikaže, obično neposredno nakon komande `\maketitle`. Sadržaj se automatski generiše na osnovu naslova kao što su `\section`, `\subsection`, i `\subsubsection` koji su definisani u dokumentu.

```
\documentclass{article}
\begin{document}
  \title{Moj LaTeX dokument}
  \author{Danijela Simić}
  \date{\today}
  \maketitle
  \tableofcontents

  \section{Uvod}
  Ovo je uvodni deo dokumenta.

  \subsection{Motivacija}
  Ovde objašnjavamo zašto koristimo LaTeX.
\end{document}
```

Važno je napomenuti da je potrebno dva puta prevesti (kompajlirati) dokument da bi se sadržaj ispravno generisao. Prvi put LaTeX kreira pomoćni fajl sa informacijama o stranicama naslova, a drugi put koristi taj fajl da bi ispravno prikazao sadržaj u PDF-u. Dakle, u terminalu treba pokrenuti komandu pdflatex naziv_fajla.tex dva puta zaredom.

Markdown (Jezik za označavanje)

Kratka istorija. Markdown je razvijen 2004. godine od strane Johna Grubera, u saradnji sa Aaronom Swartzom, kao lagani jezik za označavanje koji je dizajniran za brzo formatiranje teksta. Cilj Markdown-a bio je da bude čitljiv i lako razumljiv čak i u izvornom obliku, kao i da se jednostavno konvertuje u HTML. Danas se Markdown koristi širom interneta, posebno na platformama kao što su GitHub, Reddit, i mnogi blogovi, zbog svoje jednostavnosti i fleksibilnosti.

Kako se koristi Markdown. Markdown omogućava brzo kreiranje formatiranih dokumenata pomoću jednostavne sintakse koja koristi razne simbole za kreiranje elemenata kao što su naslovi, liste, linkovi, slike i kod blokovi.

Osnovni elementi. **Naslovi** Naslovi se kreiraju pomoću znaka `#`, pri čemu broj znakova označava nivo naslova (od 1 do 6):

```
# Naslov 1
## Naslov 2
### Naslov 3
#### Naslov 4
##### Naslov 5
###### Naslov 6
```

Paragrafi Paragrafi se formiraju jednostavnim pisanjem teksta, a za razdvajanje paragrafa koristi se jedan ili više praznih redova.

Podebljani i italic tekst - *Italic* tekst se označava stavljanjem teksta između jedne zvezdice (`*`) ili donje crte (`_`):

```
*italic* ili _italic_
```

- **Podebljani** tekst se označava sa dve zvezdice (`**`) ili dve donje crte (`__`):

```
**podebljani** ili __podebljani__
```

- Kombinacija podebljanog i italics moguća je kombinovanjem zvezdica ili donjih crta:

```
**_italic i podebljani_**
```

Liste Markdown podržava dve vrste listi: neuređene i uređene.
- **Neuređene liste** kreiraju se pomoću zvezdica (*), znakova plus (+) ili crtica (-):

```
* Stavka 1
* Stavka 2
  * Podstavka 2.1
  * Podstavka 2.2
```

- **Uređene liste** kreiraju se brojevima praćenim tačkom:

```
1. Prva stavka
2. Druga stavka
   1. Podstavka 2.1
   2. Podstavka 2.2
```

Linkovi Linkovi se kreiraju korišćenjem uglastih zagrada za tekst linka, a zatim obličnih zagrada za URL:

```
[Posetite W3C](http://www.w3c.org/)
```

Slike Sintaksa za slike je slična sintaksi za linkove, ali sa uzvičnikom (!) na početku:

```
![Alternativni tekst](url_slike)
```

Kod blokovi Markdown omogućava umetanje koda unutar linije koristeći znakove backtick (`).

```
Ovo je ‘inline kod‘ primer.
```

Za višeredne blokove koda koriste se tri backtick znaka:

```
‘‘‘
function pozdrav() {
    console.log("Zdravo, svetu!");
}
‘‘‘
```

Takođe, mogu se dodati jezici nakon ‘‘‘ za naglašavanje sintakse, npr.:

```
‘‘‘python
print("Hello, World!")
‘‘‘
```

Citatni blokovi Citatni blokovi se kreiraju korišćenjem znaka > ispred teksta:

```
> Ovo je citatni blok.
```

Horizontalna linija Horizontalne linije se kreiraju koristeći tri ili više zvezdica (***), crtica (---) ili donje crte (___):

```
---
```

Linkovi ka naslovima u dokumentu Markdown omogućava kreiranje linkova ka određenim delovima dokumenta koristeći id-ove naslova. Na primer:

```
[Idi na Naslov 2](#naslov-2)
```

Zaključak. Markdown je jednostavan, lagan jezik za označavanje koji omogućava brzo kreiranje formatiranih dokumenata koji su čitljivi i u izvornom obliku. Zahvaljujući svojoj jednostavnosti i fleksibilnosti, Markdown je postao veoma popularan, posebno među programerima i autorima tehničkih dokumenata.

XML (eXtensible Markup Language)

XML je svestrani jezik za označavanje koji se primarno koristi za skladištenje i prenos podataka. Za razliku od HTML-a, koji je dizajniran za prikazivanje podataka, XML se fokusira na definisanje struktura podataka i semantike. Omogućava programerima da kreiraju prilagođene oznake (tagove) za opisivanje značenja podataka, pružajući fleksibilan i lako čitljiv format. XML se široko koristi u web servisima, razmeni podataka i konfiguracionim fajlovima.

Primer XML-a

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="fiction">
    <title lang="en">The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <year>1925</year>
    <price>10.99</price>
  </book>
  <book category="non-fiction">
    <title lang="en">Sapiens: A Brief History of Humankind</title>
    <author>Yuval Noah Harari</author>
    <year>2011</year>
    <price>14.99</price>
  </book>
</bookstore>
```

U ovom primeru, XML se koristi za opisivanje inventara knjižare. Svaki element `<book>` sadrži ugnježdene oznake koje pružaju detalje o pojedinačnim knjigama, kao što su naslov, autor, godina i cena. Atribut `category` pruža dodatne informacije o vrsti knjige, dok atribut `lang` označava jezik naslova knjige.

XML se obično koristi u sledećim scenarijima:

- **Web servisi**: Kao format za razmenu podataka između različitih sistema, kao što je SOAP (Simple Object Access Protocol).
- **Konfiguracioni fajlovi**: Mnoge aplikacije koriste XML za svoje konfiguracione postavke, obezbeđujući doslednu strukturu koja je jednostavna za čitanje i izmene.
- **Reprezentacija podataka**: XML se koristi za predstavljanje složenih struktura podataka na način koji je čitljiv i za ljude i za računare, što ga čini idealnim za interoperabilnost između različitih aplikacija i sistema.

Pitanja i zadaci za vežbu

Pitanje 1.1. *Kako su u digitalnom računaru zapisani svi podaci? Koliko se cifara obično koristi za njihov zapis?*

Pitanje 1.2. *Koje su osnovne prednosti digitalnog zapisa podataka u odnosu na analogni? Koji su osnovni problemi u korišćenju digitalnog zapisa podataka?*

Pitanje 1.3. *Šta je Hornerova shema? Opisati Hornerova shemu pseudo-kodom.*

Pitanje 1.4. *Koje su prednosti zapisa u potpunom komplementu u odnosu na zapis u obliku označene apsolutne vrednosti?*

Pitanje 1.5. *Šta je to IEEE 754?*

Pitanje 1.6. *Šta je to glif, a šta font? Da li je jednoznačna veza između karaktera i glifova? Navesti primere.*

Pitanje 1.7. *Koliko bitova koristi ASCII standard? Šta čini prva 32 karaktera ASCII tabele? Kako se određuje binarni zapis karaktera koji predstavljaju cifre?*

Pitanje 1.8. *Navesti barem dve jednobajtne kodne strane koje sadrže ćirilične karaktere.*

Pitanje 1.9. *Nabrojati bar tri kodne sheme u kojima može da se zapiše reč* računarstvo.

Pitanje 1.10. *Koliko bitova koristi ASCII tabela karaktera, koliko YUSCII tabela, koliko ISO-8859-1, a koliko osnovna Unicode ravan? Koliko različitih karaktera ima u ovim tabelama?*

Pitanje 1.11. *Koja kodiranja teksta je moguće koristiti ukoliko se u okviru istog dokumenta želi zapisivanje teksta koji sadrži jedan pasus na srpskom (pisan latinicom), jedan pasus na nemačkom i jedan pasus na ruskom (pisan ćirilicom)?*

Pitanje 1.12. *U čemu je razlika između Unicode i UTF-8 kodiranja?*

Pitanje 1.13. *Prilikom prikazivanja filma, neki program prikazuje titlove tipa* `"raèunari æe biti..."`. *Objasniti u čemu je problem.*

Pitanje 1.14. *Šta je to piksel? Šta je to sempl?*

Zadatak 1.1. *Prevesti naredne brojeve u dekadni brojevni sistem:*
(a) $(10111001)_2$ *(b)* $(3C4)_{16}$ *(c)* $(734)_8$
Zadatak uraditi klasičnim postupkom, a zatim i korišćenjem Hornerove sheme. √

Zadatak 1.2. *Zapisati dekadni broj* 254 *u osnovama* 2, 8 *i* 16. √

Zadatak 1.3. *(a)*
Registar ima sadržaj

10101011010010001111010101010110011010111010101011100010100100 11.

Zapisati ovaj broj u heksadekadnom sistemu.

(b) Registar ima sadržaj $A3BF461C89BE23D7$. *Zapisati ovaj sadržaj u binarnom sistemu.*
√

Zadatak 1.4. *Na Vebu se boje obično predstavljaju šestocifrenim heksadekadnim kodovima u RGB sistemu: prve dve cifre odgovaraju količini crvene boje, naredne dve količini zelene i poslednje dve količini plave. Koja je količina RGB komponenti (na skali od 0-255) za boju* `#35A7DC`*? Kojim se kodom predstavlja čista žuta boja ako se zna da se dobija mešanjem crvene i zelene? Kako izgledaju kodovi za nijanse sive boje?* √

Zadatak 1.5. *U registru se zapisuju neoznačeni brojevi. Koji raspon brojeva može da se zapiše ukoliko je širina registra u bitovima:*
(a) 4 *(b)* 8 *(c)* 16 *(d)* 24 *(e)* 32 √

Zadatak 1.6. *Ukoliko se koristi binarni zapis neoznačenih brojeva širine* 8 *bitova, zapisati brojeve:*
(a) 12 *(b)* 123 *(c)* 255 *(d)* 300 √

Zadatak 1.7. *U registru se zapisuju brojevi u potpunom komplementu. Koji raspon brojeva može da se zapiše ukoliko je širina registra u bitovima:*
(a) 4 *(b)* 8 *(c)* 16 *(d)* 24 *(e)* 32 √

Zadatak 1.8. *Odrediti zapis narednih brojeva u binarnom potpunom komplementu širine* 8 *bitova:*
(a) 12 *(b)* −123 *(c)* 0 *(d)* −18 *(e)* −128 *(f)* 200 √

Zadatak 1.9. *Odrediti zapis brojeva -5 i 5 u potpunom komplementu dužine 6 bitova. Odrediti, takođe u potpunom komplementu dužine 6 bitova, zapis zbira i proizvoda ova dva broja.* √

Zadatak 1.10. *Ukoliko se zna da je korišćen binarni potpuni komplement širine* 8 *bitova, koji su brojevi zapisani?*
(a) 11011010 *(b)* 01010011 *(c)* 10000000 *(d)* 11111111
(e) 01111111 *(f)* 00000000
Šta predstavljaju dati zapisi ukoliko se zna da je korišćen zapis neoznačenih brojeva? √

Zadatak 1.11. *Odrediti broj bitova neophodan za kodiranje 30 različitih karaktera.* √

Zadatak 1.12. *Znajući da je dekadni kôd za karakter A 65, navesti kodirani zapis reči* `FAKULTET` *ASCII kodovima u heksadekadnom zapisu. Dekodirati sledeću reč zapisanu u ASCII kodu heksadekadno: 44 49 53 4B 52 45 54 4E 45.* √

Zadatak 1.13. *Korišćenjem ASCII tablice odrediti kodove kojima se zapisuje tekst:* `"Programiranje 1"`. *Kôdove zapisati heksadekadno, oktalno, dekadno i binarno. Šta je sa kodiranjem teksta* `Matematički fakultet`*?* √

Zadatak 1.14. *Za reči* `računarstvo`, `informatika`, *navesti da li ih je moguće kodirati narednim metodima i, ako jeste, koliko bajtova zauzimaju:*
(a) ASCII (b) Windows-1250 (c) ISO-8859-5
(d) ISO-8859-2 (e) Unicode (UCS-2) (f) UTF-8 √

Zadatak 1.15. *Odrediti (heksadekadno predstavljene) kodove kojima se zapisuje tekst* `kružić` *u UCS-2 i UTF-8 kodiranjima. Rezultat proveriti korišćenjem HEX editora.* √

Zadatak 1.16. *HEX editori su programi koji omogućavaju direktno pregledanje, kreiranje i ažuriranje bajtova koji sačinjavaju sadržaja datoteka. Korišćenjem HEX editora pregledati sadržaj nekoliko datoteka različite vrste (tekstualnih, izvršivih programa, slika, zvučnih zapisa, video zapisa, . . .).*

Zadatak 1.17. *Uz pomoć omiljenog editora teksta (ili nekog naprednijeg, ukoliko editor nema tražene mogućnosti) kreirati datoteku koja sadrži listu imena 10 vaših najomiljenijih filmova (pisano latinicom uz ispravno korišćenje dijakritika). Datoteka treba da bude kodirana kodiranjem:*
(a) Windows-1250 (b) ISO-8859-2 (c) Unicode (UCS-2) (d) UTF-8
Otvoriti zatim datoteku iz nekog pregledača Veba i proučiti šta se dešava kada se menja kodiranje koje pregledač koristi prilikom tumačenja datoteke (obično meni `View->Character encoding`*). Objasniti i unapred pokušati predvideti ishod (uz pomoć odgovarajućih tabela koje prikazuju kodne rasporede).*

Zadatak 1.18. *Za datu datoteku kodiranu UTF-8 kodiranjem, korišćenjem editora teksta ili nekog od specijalizovanih alata (na primer,* `iconv`*) rekodirati ovu datoteku u ISO-8859-2. Eksperimentisati i sa drugim kodiranjima.*

Zadatak 1.19. *Korišćenjem nekog naprednijeg grafičkog programa (na primer, GIMP ili Adobe Photoshop) videti kako se boja* `#B58A34` *predstavlja u CMY i HSB modelima.*