UVOD U ORGANIZACIJU I ARHITEKTURU RAČUNARA

Zapis teksta u računaru

Danijela Simić

Zapis teksta u računaru

Tekst je *"informacija namenjena ljudskom sporazumevanju koja može biti prikazana u dvodimenzionom obliku. Tekst se sastoji od grafičkih elemenata kao što su karakteri, geometrijski ili fotografski elementi ili njihove kombinacije, koji čine sadržaj dokumenta"* (ISO definicija).

Iako obično tekst zamišljamo kao dvodimenzioni objekat, u računarima se tekst predstavlja kao jednodimenzioni (linerani) niz karaktera.
Potrebno je uvesti specijalne karaktere koji označavaju prelazak u novi red, tabulator, kraj teksta i slično.

Računari su zasnovi na binarnoj aritmetici, cele broje je moguće predstaviti u binarnom sistemu, pa je osnovna ideja svakom karakteru pridružiti ceo broj na unapre dogovoren način. Ove brojeve zovemo **kodovima karaktera**.

Tokom razvoja računarstva broj karaktera je postajao sve veći. Pošto je u početku razvoja englesko govorno područje bilo dominantno osnovno je bilo predstaviti sledeće karaktere:

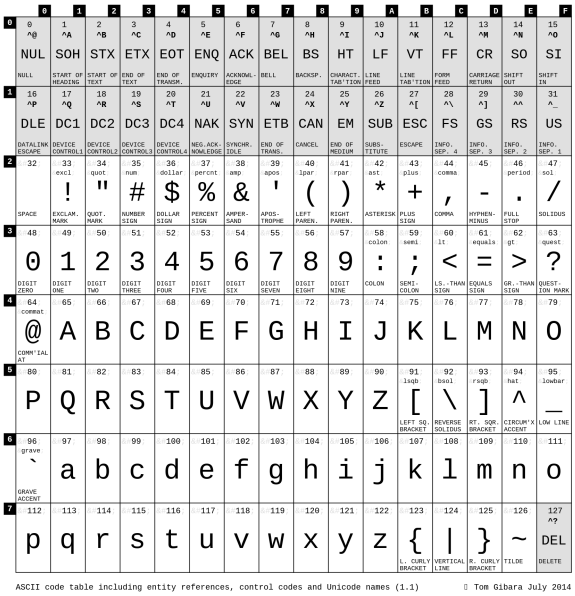
1. Velika slova engleskog alfabeta : A,B,...,Z
2. Mala slova engleskog alfabeta : a,b,...,z
3. Cifre : 0,1,...,9
4. Interpunkcijske znake : $.,:;?+*-$ _ i slično
5. Specijalne znake : kraj reda, tabulator i slično

Sedamdesetih godina su se pojavile tabele standardnih karakterskih kodova dovoljne za zapis pomenutih karaktera. Najpoznatiji su:

1. **EBCDIC** (*Extended Binary Coded Decimal Interchange Code*): IBM-ov standard, korišćen uglavnom na mainframe računarima, pogodan za bušene kartice. U ovom kode se može predstaviti 256 različitih karaktera pri čemu se svaki karakter predstavlja jednoznačnom niskom od 8 binarnih cifara. Karakteri kodirani u EBCDIC kodu čuvaju se i prenose u grupama od po 8 bita.
   Karakteri u EBCDIC kodu se mogu podeliti na kontrolne karaktere i karaktere koji imaju grafičku reprezentaciju.
2. **ASCII** (*American National Standard code for Information Interchange*). Uspostavljen 1968. godine od strane Američkog nacionalnog instituta za standarde (*engl.* American National Standard Institute).

   To je sedmobitini kod, tj. u njemu se može predstaviti 128 različitih karaktera. Karakteri u ASCII kodu se čuvaju i prenose u grupi od 8 bita pri čemu se bit najveće težine koristi za kontrolu parnosti. Kao i u EBCDIC kodnoj strani, karakteri u ASCII kodu se mogu podeliti na kontrolne karaktere i karaktere koji imaj grafičku reprezentaciju.

   Karakter A se zapisuje kao $(41)_{16}$, tj. Ox41 što je $(65)_{10}$ tj. $(1000001)_2$.
   Karakter a se zapisuje kao $(61)_{16}$, tj. Ox61 što je $(97)_{10}$ tj. $(1100001)_2$.
   Karakter 0 se zapisuje kao $(30)_{16}$, tj. Ox30 što je $(48)_{10}$ tj. $(110000)_2$.



Slika 1: ASCII tablica

Kontrolni znakovi u ASCII tabeli (znakovi sa kodovima od 0 do 31 i 127) služe različitim svrhama u računarstvu i komunikacionim sistemima. Ovi znakovi često nisu vidljivi na ekranu ili papiru, što znači da ne predstavljaju vidljive simbole. Umesto toga, koriste se za kontrolu i koordinaciju različitih zadataka. Evo nekoliko uobičajenih svrha kontrolnih znakova u ASCII tabeli:

- Kontrolni znakovi su prvobitno razvijeni za teleteks i rane računarske komunikacione sisteme. Pomažu u kontroli prenosa podataka, kao što su početak i zaustavljanje toka podataka (npr. Početak zaglavlja - SOH, Kraj teksta - ETX).
- Oblikovanje teksta: Neki kontrolni znakovi se koriste za oblikovanje teksta i kontrolu rasporeda. Na primer, Line Feed (LF) i Carriage Return (CR) se koriste za premeštanje kursora na početak sledećeg reda ili na početak trenutnog reda. Tab (HT) se koristi za horizontalnu tabulaciju, dok se Vertical Tab (VT) koristi za vertikalnu tabulaciju.
- Struktura podataka: Određeni kontrolni znakovi se koriste za definisanje strukture podataka. Na primer, Start of Text (STX) i End of Text (ETX) se koriste za razgraničavanje tekstualnih podataka. Record Separator (RS) i Unit Separator (US) se mogu koristiti za razdvajanje zapisa i podataka unutar toka.
- Kontrola uređaja: Kontrolni znakovi se mogu koristiti za interakciju sa spoljnim uređajima, kao što su štampači i displeji. Na primer, Bell (BEL) može izazvati zvučno upozorenje ili vizuelni signal na nekim uređajima.
- Kraj datoteke: End of Transmission (EOT) se koristi za označavanje kraja prenosa ili kraja fajla.
- Upravljanje greškama: Neki kontrolni znakovi se koriste za upravljanje i ispravku grešaka, kao što je Error(ERR) znak.
- Nekarakteristični znakovi: Null (NUL) se često koristi kao zamena ili za popunjavanje podataka. Delete (DEL) se može koristiti za označavanje izbrisanih ili nevažećih karaktera.

Važno je napomenuti da se upotreba i tumačenje kontrolnih znakova može razlikovati u zavisnosti od konteksta i konkretnog sistema. Savremeni računarski sistemi često obrađuju mnoge od ovih zadataka na drugačiji način, ali kontrolni znakovi ostaju deo istorijskog nasleđa kodiranja karaktera.

Oznaka za kraj reda se ne zapisuje isto u svim operativnim sistemima. Pod Windows operativnim sistemom, ova se oznaka se zapisuje sa dva karaktera (CR LF), 0xD 0xA tj. 13 10 - istorijski razlozi (stari štampači). Unix koristi samo karakter CR tj. 0xD. Ista konvencija za zapis novog reda je korišćena u okviru MacOS operativnog sistema.

8-bitna proširenja ASCII tabele

Razvojem računarstva se javlja potreba kodiranja tekstova i na drugim jezicima. Kroz istoriju su postojala mnoga rešenja, od kojih su se neka zadržala, a neka su nestala.

Pod **kodnom stranom** (Code page) tj. **skupom karaktera** (Character set, charset) podrazumevamo uređenu listu karaktera predstavljenih svojim karakterskim kodovima.

Podaci se u računarima obično zapisuju bajt po bajt. ASCII je sedmobitni standard. ASCII karakteri se zapisuju tao što se u svakom bajtu bit najveće težine postavi na 0. To ostavlja prostor za novih 128 karaktera čiji binarni zapis počinje sa 1. Ovaj prostor se može popuniti na razne načine. Rešenje nije univerzalno, jer na svetu postoji više od 256 različitih karaktera. Postavljeni su razni standardi dopunjavanja ovih 128 karaktera. Svim ovim kodnim stranama je zajedničko prvih 128 karaktera i oni se poklapaju sa ASCII karakterima.

Ovako napravljene kodne strane obično omogućavaju kodiranje tekstova na više srodnih jezika (obično i geografski bliskih). Nama su uglavnom važne kodne strane napravljene za centralno-evropske (Central European) latinice, kao i ćirilične kodne strane. Najčešće korišćene kodne strane kod nas su:

1. ISO 8859-1 (Latin1)
2. ISO 8859-2 (Latin2)
3. ISO 8859-5 (Ćirilična)
4. Windows 1250
5. Windows 1251 (Ćirilična)

Prve tri su delo međunarodne organizacije za standardizaciju (International Standard Organization), dok su naredne dve standarni napravljeni od strane kompanije Microsoft. ISO 8859-1 se veoma često postavlja kao podrazumevana kodna strana. Ona se koristi za zapis tekstova na zapadno evropskim jezicima (Western European).

Višebajtne kodne strane

Iako navedene kodne strane omogućavaju kodiranje tekstova koji nisu na engleskom jeziku nije moguće u istom tekstu pisati i ćirilicu i našu latinicu. Azijskim jezicima nije dovoljno 256 mesta za zapis svih karaktera. Zbog toga se uvode **višebajtni karakterski kodovi**.

Pre svega zbog potreba istočno azijskih korisnika uvedeni su tzv. višebajtni skupovi karaktera tj. Multi-Byte Character Sets (MBCS).

Ideja je u tome da se najčešće korišćeni karakteri zapisuju koristeći samo jedan bajt, dok se ostali karakteri zapisuju koristeći dva bajta, tj. koristi se mešavina jednobajtnih i dvobajtnih karakterskih kodova (pod UNIX-om nekad čak i trobajtnih). Ovo značajno otežava tumačenje podataka.

Kasnih osamdesetih, dve velike organizacije su pokušale standardizaciju tzv. *Univerzalnog skupa karaktera* (Universal Character Set - UCS).

To su bili **ISO 10646** i projekat **UNICODE** organizovan i finansiran uglavnom od strane američkih firmi koje su se bavile proizvodnjom višejezičkog softvera.

ISO 10646 je zamišljen kao 4 bajtni standard. Pri tome se prvih 65536 karaktera koriste kao osnovni višejezični skup karaktera dok je ostali prostor ostavljen kao proširenje za drevne jezike, celokupnu naučnu notaciju i slično. Vremenom su se pomenuta dva projekta združila i nastao je jedinstven dvobajtni standard koji jednostavno nazivamo **UNICODE**. Početna verzija UNICODE standarda svakom karakteru dodeljuje dvobajtni kod.

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 izmedu $(0000)_{16}$ i $(FFFF)_{16}$.

Prvih 128 karaktera se poklapaju sa ASCII standardom, dok su sledećih 128 napravljeni tako da se pokalapaju sa Latin1 standardom.

020-007E	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) simbol
...	

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

Standardi koji opisuju kako se niske karaktera prevode u nizove bajtova

ISO definiše **UCS-2** standard koji jednostavno svaki UNICODE karakter prevodi u odgovarajuća dva bajta.

Tekstovi kodirani preko UCS-2 standarda sadrže veliki broj nula, koje obično u operativnim sistemima poput UNIX-a i u programskom jeziku C imaju specijalno značenje.

Iz istog razloga softver koji je razvijen za rad sa dokumentima u ASCII formatu ne može da radi bez izmena nad dokumentima kodiranim preko UCS-2 standarda.

UTF (*A Unicode transformation format*) je algoritam koji svakom UNICODE karakteru dodeljuje odredjeni niz bajtova čija dužina varira od 1 do najviše 6.

UTF je ASCII kompatibilan, što znači da se ASCII karakteri zapisuju pomoću jednog bajta, na standardni način.

Najčešće korišćena varijanta ovog agloritma je **UTF-8** koja je dovoljna za zapis svih dvobajtnih UNICODE karaktera. Njegova dužina varira od 1 do najviše 3 bajta i konverzija se vrši po sledećim pravilima:

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

Pored ovoga ISO uvodi i UTF-16, UTF-32, kao i standard UCS-4.

Primer: Tekstualna datoteka sadrži 512 ASCII karaktera.

1. Koliko bajtova datoteka zauzima?
2. Koliko bajtova će zauzimati isti tekst ukoliko se koristi UNICODE (UCS-2) kodiranje?
3. Koliko bajtova će zauzimati isti tekst ukoliko se koristi UTF-8 kodiranje?
4. Koliko bajtova će zauzimati isti tekst ukoliko se koristi ISO 8859-2 kodiranje?
5. Koliko bajtova će zauzimati isti tekst ukoliko se koristi ISO 8859-5 kodiranje?
6. U kojim od navedenih formata je moguće ispravno kodirati tekst koji sadrži naša latinična i ćirilična slova?

Rešenje:

1. Datoteka zauzima 512 bajtova.
2. Ako se koristi UCS-2 tekst zauzima 1024 bajta.
3. Datoteka zauzima po 512 bajtova.
4. Datoteka zauzima po 512 bajtova.
5. Datoteka zauzima po 512 bajtova.
6. Tekst koji sadrži naša latinična i ćirilična slova moguće je ispravno kodirati samo u UCS-2 i UTF-8.

Uticaj na programiranje, C primeri

Određivanje enkodiranja znakova tekstualne datoteke može biti složen zadatak jer tekstualni fajlovi obično ne sadrže eksplicitne informacije o enkodiranju. Jedan od pristupa može biti proverava da li sadržaj tekstualna datoteka odgovara uobičajenim obrascima za UTF-8 enkodiranje. Ipak, napomena je da se radi o osnovnoj heuristici i da možda ovakav pristup neće uvek tačno identifikovati enkodiranje, posebno za datoteke sa neodređenim ili mešanim enkodiranjem. Detekcija enkodiranja sa visokom tačnošću često zahteva sofisticirane metode.

U modernim programskim jezicima postoji podrška za rad sa karakterima i niskama enkodiranim u različitim kodnim stranama.

U narednom delu videćemo neke primere iz programskog jezika C u kojima se manipuliše karakterima zapisanim u UTF-8 standardu.

1. U programskom jeziku C koristi se biblioteka `wchar.h` za rad sa karakterima koji mogu biti enkodirani u kodnoj strani koja nije ASCII. Za ASCII karaktere smo koristi tip `char`, ali u opštem slučaju to nije moguće imajući na umu da neki karakteri zauzimaju više od 1 bajta. Zato se kao tip karaktera koristi `wchar_t`.

   ```c
   #include <stdio.h>
   #include <wchar.h>

   int main() {
       wchar_t wideChar = L'Š';
       wprintf(L"Široki karakter: %lc\n", wideChar);
       return 0;
   }
   ```

   ```
   Izlaz iz ovog programa:
   ?iroki karakter: D
   ```

   Za definisanje konstante karaktera koji nije ASCII kodiran se ispred znaka navodnika navodi prefiks `L`. Bez ovog prefiksa tip konstante je `char`.

   Za ispis teksta koji sadrži karaktere van osnovne ASCII tabele potrebno je korisiti funkciju `wprintf`. Prilikom ispisa koristi se kvalifikator `%lc`.

2. Veličina tipa `wchar_t` zavisi od kompajlera i platforme. Na većini modernih sistema, tip `wchar_t` je obično 2 bajta (16 bita) ili 4 bajta (32 bita). Ipak, ovo nije striktno definisano u C standardu.

   ```c
   #include <stdio.h>
   #include <wchar.h>

   int main() {
       printf("Velicina tipa wchar_t: %lu bajta\n", sizeof(wchar_t));
       return 0;
   }
   ```

   ```
   Izlaz iz ovog programa:
   Velicina tipa wchar_t: 4 bajta
   ```

Na sistemu na kome je ovaj program pokrenut, tip `wchar_t` je zauzeo 4 bajta.

Važno je poznavati veličinu ovog tipa, kao i platforme koje se koriste da bi napisani program bio prenosiv i da bi ispravno radio na različitim sistemima.

3. Pored bibilioteke `wchar.h` potrebno je podesiti i **lokalizaciju** da bi rad sa karakterima zapisanim u različitim kodiranjima bio ispravan. To postižemo korišćenjem biblioteke `locale.h`

```c
#include <stdio.h>
#include <wchar.h>
#include <locale.h>

int main() {
    setlocale(LC_ALL, "en_US.UTF-8");

    wchar_t wideChar = L' ';
    wprintf(L"Široki karakter: %lc\n", wideChar);
    return 0;
}
```

```
Izlaz iz ovog programa:
Široki karakter:
```

Funkcija `setlocale` služi da se za tekući program postavi lokalizacija. Lokalizacija predstavlja skup podešavanja koji određuju kako će se podaci kao što su brojevi, datumi, vreme i karaktersko kodiranje formatirati i interpretirati.

Prethodnom primeru parametri lokalizacije su:

- `LC_ALL` je konstanta koja označava na koja podešavanja funkcija utiče. U ovom primeru, ova konstanta označava da su sva podešavanja prema postavljenom lokalu, a to označava i karaktersko enkodiranje, formati datuma i vremena itd.
- `en_US.UTF-8` je niska kojom se zadaje željena lokalizacija. U ovom primeru, jezik je engleski, oznaka zemlje je Sjedinjene Američke države i koristi se UTF-8 za kodiranje karaktera.

```c
#include <stdio.h>
#include <wchar.h>
#include <locale.h>

int main() {
    if (setlocale(LC_ALL, "sr_RS.UTF-8") == NULL) {
        perror("Failed to set the locale");
        return 1;
    }

    wchar_t wideChar = L' '; //'Ω'
    wprintf(L"Široki karakter: %lc\n", wideChar);
    return 0;
}
```

U ovom primeru, kao jezik je postavljen srpski (ćirilica), za kod zemlje je postavljena Srbija, a kodiranje karaktera je prema standardu UTF-8.

U programskom jeziku C `setlocale` se može pozvati više puta u toku rada programa, ali treba imati na umu da svaki poziv ove funkcije utiče na kompletno ceo program i njegova podešavanja. Ako je funkcija pozvana više puta, onda je poslenji poziv jedini koji ima efekat.

Ako funkcija `setlocale` iz bilo kog razloga ne može da postavi traženu lokalizaciju onda ona vraća `NULL`.

4. U narednim primerima možemo videti kako različite lokalizacije utiču na ispis datuma i vremena.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>

int main() {
    // Postavlja se lokalizacija
    if (setlocale(LC_ALL, "sr_RS.UTF-8") == NULL) {
        // Nemoguce postaviti lokalizacju
        return 1;
    }

    // Podaci o datumu i vremenu
    time_t tekuceVreme;
    struct tm *vremeInfo;

    time(&tekuceVreme);
    vremeInfo = localtime(&tekuceVreme);

    // Kreiranje niske u kojoj se nalayi podaci o vremenu i datumu
    char buffer[80];
    strftime(buffer, sizeof(buffer), "%A, %d %B %Y, %H:%M:%S", vremeInfo);

    // Stampanje datuma i vremena
    printf("Datum i vreme: %s\n", buffer);

    return 0;
}
```

```
Izlaz iz ovog programa:
Datum i vreme:    , 12      2023, 14:05:35
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>

int main() {
    // Postavlja se lokalizacija
    if (setlocale(LC_ALL, "en_US.UTF-8") == NULL) {
        // Nemoguce postaviti lokalizacju
        return 1;
    }
```

```
    // Podaci o datumu i vremenu
    time_t tekuceVreme;
    struct tm *vremeInfo;

    time(&tekuceVreme);
    vremeInfo = localtime(&tekuceVreme);

    // Kreiranje niske u kojoj se nalayi podaci o vremenu i datumu
    char buffer[80];
    strftime(buffer, sizeof(buffer), "%A, %d %B %Y, %H:%M:%S", vremeInfo);

    // Stampanje datuma i vremena
    printf("Datum i vreme: %s\n", buffer);

    return 0;
}
```

```
Izlaz iz ovog programa:
Datum i vreme: Sunday, 12 November 2023, 14:08:16
```

Lokali koji su podržani u okviru Unix sistema mogu se izlistati komandom `locale -a`.

5. Moguće je kreirati i niske koje sadrže karaktere koji ne pripadaju ASCII kodnoj strani.

```
#include <stdio.h>
#include <wchar.h>
#include <locale.h>

int main() {
    if (setlocale(LC_CTYPE, "en_US.UTF-8") == NULL) {
        // Nemoguce postaviti lokalizacju
        return 1;
    }

    wchar_t wideString[] = L"      ,    ! A uzmimo i primer iz grčkog: Ω";
    wprintf(L"Široka niska: %ls %zu\n", wideString, sizeof(wideString));

    return 0;
}
```

```
Izlaz iz ovog programa:
Široka niska:       ,    ! A uzmimo i primer iz grčkog: Ω
```

U prikazanom primeru je korišćena konstanta `LC_CTYPE` i time je označeno da želimo samo da utičemo na kodiranje karaktera, ali ne i na ostale parametre lokalizacije.

6. Naredni primer prikazuje kako se karakteri mogu unositi korišćenjem funkcije `wscanf`.

```
#include <stdio.h>
#include <wchar.h>
#include <locale.h>
```

```c
int main() {
    wchar_t karakter;

    if (setlocale(LC_CTYPE, "en_US.UTF-8") == NULL) {
        // Nemoguce postaviti lokalizacju
        return 1;
    }

    wprintf(L"Uneti karakter: ");
    wscanf(L"%lc", &karakter);

    wprintf(L"Uneli ste: %lc\n", karakter);
    return 0;
}
```

```
Izlaz iz ovog programa:
Uneti karakter: Č
Uneli ste: Č
```

7. Primer sa sortiranjem niski zapisanih ćirilicom.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>

// Funkcija za poredjenje koja poredi na osnovu cirilicnog rasporeda
int uporediImena(const void *a, const void *b) {
    return wcscoll(*(const wchar_t **)a, *(const wchar_t **)b);
}

int main() {
    if (setlocale(LC_ALL, "sr_RS.UTF-8") == NULL) {
        perror("Failed to set the locale");
        return 1;
    }

    // Otvaranje datoteke
    FILE *file = fopen("imena.txt", "r");
    if (file == NULL) {
        perror("Failed to open the file");
        return 1;
    }

    // Citanje imena, alokacija memorije
    wchar_t **imena = NULL;
    size_t brImena = 0;
    wchar_t line[256]; // Maksimalna duzina imena je 255 karaktera

    while (fgetws(line, sizeof(line), file) != NULL) {
        // Uklanjanje karaktera za novi red
        for (wchar_t *ptr = line; *ptr; ptr++) {
```

```
            if (*ptr == L'\n' || *ptr == L'\r') {
                *ptr = L'\0';
                break;
            }
        }

        // Alociranje prostora
        wchar_t *imeKopija = wcsdup(line);
        if (imeKopija == NULL) {
            perror("Memory allocation error");
            return 1;
        }

        // Povecavanje alociranog prostora -- napomena: prilicno neefikasno
        wchar_t **temp = realloc(imena, (brImena + 1) * sizeof(wchar_t *));
        if (temp == NULL) {
            perror("Memory allocation error");
            return 1;
        }

        imena = temp;
        imena[brImena++] = imeKopija;
    }

    // Zatvranje datoteke
    fclose(file);

    // Sortiranje koriscenjem ugradjene funkcije za sortiranje
    // ali koriscenjem specijalne funkcije za poredjenje
    qsort(imena, brImena, sizeof(wchar_t *), uporediImena);

    // Print the sorted names
    for (size_t i = 0; i < brImena; i++) {
        wprintf(L"%ls\n", imena[i]);
        free(imena[i]);
    }

    free(imena);

    return 0;
}
```

Funkcija `wcscoll` poredi dva karaktera ali uzimajući u obzir pravila koja važe u okviru zadate lokalizacije. Pogodna je za korišćenje u slučajevima kada treba da se porede (i sortiraju) niske zapisane u različitim jezicima.

Glifovi i fontovi [1]

Postoji veoma jasna razlika izmedu 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 izmedu 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.

Takode, moguće je da odredeni fontovi ne sadrže glifove za odredene 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.

Fontovi su kolekcije znakova sa doslednim dizajnom, stilom i veličinom. Font se sastoji od znakova, simbola i glifova, i definiše kako tekst izgleda na ekranu ili u štampanom obliku.

Serif i Sans-Serif fontovi

Izbor između `serif` (šerifni) i `sans-serif` (sans-Šerifni) fontova je ključno razmatranje u tipografiji koje utiče na vizuelni stil, čitljivost i ukupan ton teksta.

Serif fontovi se odlikuju prisustvom malih dekorativnih linija, poznatih kao "šerifi"na krajevima znakova. Ovi šerifi mogu imati različite oblike, uključujući šerife sa krivinama, ravne šerife blok-šerife (deblji šerifi). Primeri serif fontova uključuju Times New Roman, Georgia i Baskerville.

Sa druge strane, sans-serif fontovi ne sadrže ove dekorativne šerife. Izraz šans"je francuski za "bez"ističući odsustvo šerifa u ovim vrstama slova. Sans-serif fontovi imaju čiste, jednostavne linije i često se doživljavaju kao moderni i neposredni. Primeri sans-serif fontova uključuju Arial, Helvetica i Calibri.

Serif fontovi se obično koriste u **formalnim dokumentima** kao što su akademski radovi, pravni dokumenti i poslovni izveštaji. Njihov klasičan i tradicionalan izgled pruža autoritet i čitljivost ovim materijalima. Pružaju dobru čitljivost, pa su čest izbor u štampanim medijima.

Sans-serif fontovi se obično koriste u **digitalnim sadržajima**, uključujući veb sajtove i korisničke interfejse. Njihov čist, jednostavan izgled čini ih pogodnim za čitanje na ekranima. U savremenom dizajnu, sans-serif fontovi se favorizuju zbog svoje jednostavne i minimalističke estetike. Često se koriste u brendiranju, oglašavanju i grafičkom dizajnu.

[1] Deo preuzeto iz knige: Predrag Janičić, Filip Marić: Programiranje 1

Grupe fontova

Grupu fontova čini skup srodnih fontova koji dele zajedničko dizajnersko poreklo, ali nude varijacije u stilu, debljini i drugim karakteristikama.

Grupa fontova se sastoje od nekoliko tipografskih varijanti, svaka sa svojim jedinstvenim karakteristikama. Ove karakteristike uključuju:

1. **Dizajn**: Osnovni dizajn ili izgled znakova, koji definiše ukupni estetski izgled tipografskog stila. Grupa fontova može imati različite dizajnerske stilove, kao što su tradicionalni, moderni, dekorativni ili rukom pisani.
2. **Stil**: Stil se odnosi na varijacije unutar grupe. Na primer, grupa može uključivati regularni, italic, bold i bold italic stil.
3. **Debljina**: Debljina tipografskog stila odnosi se na debljinu ili težinu znakova. Grupa fontova može ponuditi različite debljine, od tankih i svetlih, preko regularnih, srednjih, polubold, bold, pa sve do ekstra bold varijanti.
4. **Širina**: Neke grupe pružaju varijacije u širini znakova, od kondenzovane (uske) do proširene (široke). Ove varijacije širine omogućavaju fleksibilno postavljanje teksta i prilagođavanje.
5. **Dodatne karakteristike**: Osim osnovnih karakteristika, grupe mogu sadržavati posebne karakteristike poput malih slova pisanih kao velika slova, cifara u starom stilu ili alternativnih znakova. Ove dodatne karakteristike mogu poboljšati fleksibilnost tipografskog stila.

Postoji mnogo grupa fontova koji su široko prepoznatljivi i postali su osnova kako za štampane tako i za digitalne medije. Evo nekoliko značajnih primera:

- **Arial**: Arial je veoma svestran sans-serif font poznat po svom čistom i modernom izgledu. Često se koristi za veb sadržaj i dokumente gde su jasnoća i čitljivost od suštinskog značaja.
- **Times New Roman**: Times New Roman je klasičan serif font koji odiše tradicijom i elegancijom. Često se koristi za formalne dokumente, novine i akademske publikacije.
- **Helvetica**: Helvetica je ikoničan sans-serif font poznat po svojoj neutralnosti i fleksibilnosti. Preferira se zbog svog čistog i minimalističkog dizajna i često se koristi u signalizaciji, brendiranju i štampanim medijima.
- **Verdana**: Verdana je sans-serif porodični font specijalno dizajniran za optimalnu čitljivost na ekranima. Često se koristi za veb sadržaj, posebno u malim veličinama teksta.

Izbor odgovarajućeg porodičnog fonta je ključna dizajnerska odluka koja može značajno uticati na ton i čitljivost dokumenta ili projekta. Dizajneri i tipografi pažljivo razmatraju kontekst i publiku pri izboru porodičnog fonta, osiguravajući da se usklađuje sa nameravanim porukama i vizualnom estetikom.

Veličina fonta se obično meri u dve osnovne jedinice:

1. **Tačke (pt)**: Tačka je standardna merna jedinica u tipografiji. Jedna tačka je približno jednaka 1/72 inča. U tipografiji, tačka se koristi za definisanje visine znakova u fontu. Na primer, veličina fonta od 12 tačaka znači da je visina znakova 1/6 inča.

2. **Pikseli (px)**: U digitalnoj tipografiji, posebno za veb dizajn i sadržaj na ekranu, veličina fonta može biti specificirana u pikselima. Pikseli su najmanje jedinice prikaza na ekranu, i veličina fonta u pikselima određuje tačan broj piksela dodeljenih visini znakova.

Pored tradicionalnih danas postoje i mnogi web fontovi, koji se mogu lako preuzeti i to su fontovi korišćeni na veb sajtovima kako bi se osigurala dosledna tipografija na različitim uređajima i pregledačima. Popularne usluge web fontova su Google Fonts i Adobe Typekit i nude širok izbor fontova za upotrebu. Primer upotrebe Google fonta možete videti u sledećem primeru.

```
<!DOCTYPE html>
<html>
<head>
    <!-- Link koji uključuje izabrani Google font -->
    <link href="https://fonts.googleapis.com/css2?family=Open+Sans&display=swap" rel="stylesheet">
</head>
<body>
    <!-- CSS podešavanja za font -->
    <style>
        body {
            font-family: 'Open Sans', sans-serif;
        }
    </style>
</body>
</html>
```

Fontovi se mogu kreirati koristeći specijalizovane softverske alatke poput `FontForge`, `Glyphs`, `FontLab` i `Adobe Font Folio`. Ovi alati pružaju platformu za dizajniranje i usavršavanje svakog karaktera, uključujući slova, brojeve, interpunkcijske znake i simbole.

Dizajneri fontova razmatraju različite karakteristike prilikom kreiranja fontova, uključujući visinu malih slova (x-visina), linije koje se prostiru iznad x-visine, linije koje se prostiru ispod osnovne linije i prilagođavanje razmaka između karaktera.

Mnoge organizacije i brendovi naručuju prilagođene fontove kako bi uspostavili jedinstveni vizuelni identitet. Prilagođeni fontovi su dizajnirani da odražavaju ličnost i vrednosti određenog entiteta.

Dizajneri fontova često licenciraju svoj rad kako bi zaštitili svoju intelektualnu svojinu.