

---

# Datoteke

---

nastavak

# Funkcija `fgets`

- Funkcija koja učitava podatke iz datoteke, liniju po liniju, je
  - `char *fgets(char *str, int n, FILE *fp);`
- gdje su
  - `str` – pokazivač na dio memorije (`string`) u koji će ulazna linija biti spremljena;
  - `n` – veličina memorije na koju pokazuje prvi argument;
  - `fp` – „pokazivač na datoteku” iz koje se učitava.
- Funkcija će pročitati liniju od `n-1` znakova (ili manje: sve do oznake kraja datoteke ili oznake kraja reda - `'\n'`) i na kraj će automatski dodati `'\0'`.

# Funkcije `fgets` (2)

- Ukoliko je ulazna linija dulja, ostatak će biti pročitano pri sljedećem pozivu funkcije `fgets`.
- Funkcija vraća `str` ako je sve u redu ili `NULL` u slučaju pojavljivanja pogreške ili ako se došlo do kraja datoteke.
- Funkcija
  - `char *gets(char *str);`  
čita sa standardnog ulaza.
- Kraj učitavanja – oznaka za novi red.
- Umjesto `gets(str)` može se koristiti `fgets(str, n, stdin)`.

# Funkcija `fputs`

- Funkcija za ispis podataka u datoteku, liniju po liniju, je
  - `int fputs(const char *str, FILE *fp);`
- Funkcija vraća nenegativnu vrijednost u slučaju uspjeha ili `EOF` u slučaju greške.
- `fputs` ispisuje znakovni niz na koji pokazuje `str` u datoteku na koju pokazuje `fp`. Zadnji nul znak neće biti ispisan.
- Funkcija
  - `int puts(const char *str);`  
ispisuje znakovni niz na koji pokazuje `str` na standardni izlaz. Na kraj niza funkcija dodaje znak

# Formatirani unos/ispis

za prijelaz u novi red što ju razlikuje od `fputs(str, stdout)`.

- Za formatirani upis u datoteku i ispis (čitanje) iz datoteke koristimo funkcije
  - `int fprintf(FILE *fp, const char *format, ...);`
  - `int fscanf(FILE *fp, const char *format, ...);`koje su identične s funkcijama `printf` i `scanf`, s tim da je prvi argument pokazivač na datoteku.
- `fprintf` vraća broj upisanih znakova ili negativan broj u slučaju greške.
- `fscanf` vraća broj učitanih objekata ili `EOF` ako je došlo do greške ili kraja datoteke.

# Formatirani unos/ispis (2)

- `printf(...)` je ekvivalentno s `fprintf(stdout,...)`, a `scanf(...)` je ekvivalentno s `fscanf(stdin,...)`.

## Primjer:

```
int kolicina = 100;
```

```
double cijena = 4.50;
```

```
fprintf(fp, "%d:%f\n", kolicina, cijena);
```

```
.....
```

```
int kolicina;
```

```
double cijena;
```

```
fscanf(fp, "%d:%lf\n", &kolicina, &cijena);
```

# Funkcije `sprintf` i `sscanf`

- Za formatirani upis u znakovni niz (string) i ispis iz znakovnog niza koristimo funkcije
    - `int sprintf(char *s, const char *format, ...);`
    - `int sscanf(char *s, const char *format, ...);`koje su identične s funkcijama `printf` i `scanf`, s tim da je prvi argument pokazivač na string.
  - `sprintf` vraća broj upisanih znakova bez nulznaka ili negativan broj u slučaju greške.
  - `sscanf` vraća broj učitanih objekata ili `EOF` ako je došlo do greške ili kraja datoteke.
-

# Primjer:

```
#include <stdio.h>

int main( ){
    char *podaci = "2    2.50", ispis[80];
    int kolicina; double cijena;

    sscanf(podaci, "%d %lf", &kolicina, &cijena);
    printf("%f\n", cijena);

    sprintf(ispis, "kolicina = %d\n" \
              "cijena = %f\n", kolicina, cijena);
    printf("%s\n", ispis);

    return 0;
}
```

---



# Funkcije pogreške

- Budući da funkcije vraćaju `EOF` i u slučaju kada je došlo do greške i u slučaju kada se naiđe na kraj datoteke, postoje funkcije

- `int ferror(FILE *fp);`

- `int feof(FILE *fp);`

koje razlikuju ta dva slučaja.

- `ferror` vraća broj različit od nule (istina) ako je došlo do greške, a nulu (laž) ako nije.
  - `feof` vraća broj različit od nule (istina) ako smo došli do kraja datoteke, a nulu (laž) u suprotnom.
-

# Binarno čitanje i pisanje

## Primjer:

```
do{
    c = getc(fp);
    putchar(c);
}
while(!feof(fp));
```

- Za binarno čitanje iz datoteke i pisanje u datoteku koristimo funkcije
  - `size_t fread(void *ptr, size_t size, size_t n, FILE *fp);`
  - `size_t fwrite(const void *ptr, size_t size, size_t n, FILE *fp);`

---

# Binarno čitanje i pisanje (2)

- Funkcije ne rade konverziju iz binarnog u ASCII format i obratno, tj. direktno se čita binarni zapis iz datoteke, odnosno upisuje binarni zapis u datoteku.
  - Argumenti funkcija su:
    - `ptr` – pokazivač na varijablu u koju `fread` upisuje, odnosno iz koje `fwrite` čita
    - `size` – veličina pojedinog objekta
    - `n` – broj objekata koje treba učitati/ispisati
    - `fp` – pokazivač na datoteku iz koje se čita ili u koju se piše.
-

# Binarno čitanje i pisanje (3)

## Primjer:

```
int polje[10];  
fread(polje, sizeof(int), 10, fp);  
.....  
int polje[10] = {...};  
fwrite(polje, sizeof(int), 10, fp);
```

- Prednost binarnog izlaza/ulaza je brzina i veličina zapisa.
- Nedostatak je ovisnost o arhitekturi računala i prevodiocu.

---

# Funkcije za pozicioniranje

- Funkcija

- `int fseek(FILE *fp, long offset, int origin);`

gdje su

- `fp` – pokazivač na datoteku

- `offset` – pomak u bajtovima

- `origin`:            `SEEK_SET` – od početka datoteke

- `SEEK_CUR` – od trenutne pozicije

- `SEEK_END` – od kraja datoteke

- Funkcija vraća 0 u slučaju uspjeha, a broj različit od nule (-1) u slučaju pogreške.

---

# Funkcija `fseek`

## Primjer:

```
fseek(fp, 0, SEEK_SET);    /* Pozicioniranje na */  
rewind(fp);               /* početak datoteke. */  
  
fseek(fp, 0, SEEK_END);   /* Pozicioniranje na kraj datoteke. */  
  
fseek(fp, 2L, SEEK_CUR);  
fseek(fp, -2, SEEK_END);
```

---

# Trenutna pozicija u datoteci

- Funkcija

- `int ftell(FILE *fp);`

- Funkcija vraća poziciju u datoteci ili -1 u slučaju pogreške.

Primjer:

```
fp = fopen ("dat1", "r");  
printf("%d \n", ftell(fp));  
getc(fp);  
printf("%d \n", ftell(fp));
```



# Primjer:

```
int main()
{
    FILE *fp;
    int p, c, i = 0;

    if ((fp = fopen ("dat1","r")) == NULL){ ...}

    do {
        i--;
        fseek(fp, i, SEEK_END);
        p = ftell(fp);
        c = getc(fp);
        putchar(c);
    }
    while (p != 0);    fclose(fp);    return 0; }
```

---



# Primjer:

- Sadržaj datoteke test.txt:

```
Podaci za 2 studenta/ice:
```

```
20 40
```

```
10 30
```

- Kako iščitati numerički podatak (2)?

```
FILE *fp;
```

```
if ((fp = fopen("test.txt", "r")) == NULL) ...
```

```
fscanf(fp, "%s %s %d %s \n", s1, s2, &n, s3);
```

```
fscanf(fp, "%*s %*s %d %*s", &n);
```