Ulaz i izlaz podataka

Slide 1

Funkcije `getchar` i `putchar`: (`<stdio.h>`)

```
int getchar(void);
int putchar(int);
```

Funkcija `getchar` čita jedan znak sa standardnog ulaza (tipično tipkovnice). Funkcija nema argumenata pa je sintaksa poziva:

```
c_var=getchar();
```

Funkcija `putchar` šalje jedan znak na standardni izlaz (tipično ekran). Ona uzima jedan argument (znak koji treba ispisati) i vraća cjelobrojnu vrijednost. Najčešće poziv funkcije ima oblik

```
putchar(c_var);
```

pri čemu se vraćena vrijednost ignorira.

Slide 2

Kada funkcija `getchar` naidje na kraj ulaznih podataka vraća vrijednost `EOF` (skraćeno od eng. *End of File*). `EOF` je simbolička konstanta definirana u `<stdio.h>` koja signalizira kraj datoteke i kraj ulaznih podataka (ulaz je tretiran kao datoteka).

Konstanta `EOF` mora se razlikovati od znakova iz sustava znakova koje računalo koristi. Stoga funkcija `getchar` ne vraća vrijednost tipa `char` već vrijednost tipa `int` što daje dovoljno prostora za kodiranje konstante `EOF`. Isto tako `putchar` uzima vrijednost tipa `int` i vraća vrijednost tipa `int`. Vraćena vrijednost je znak koji je ispisan ili `EOF` ako ispis znaka nije uspio.

Slide 3

Primjer: program koji kopira znak po znak ulaz na izlaz i pri tome sva slova pretvara u velika:

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{
   int c;

   while((c=getchar())!=EOF)
            putchar(toupper(c));

   return 0;
}
```

Funkcija `toupper` deklarirana je u datoteci zaglavlja `<ctype.h>`. Ona pretvara mala slova u velika, a sve druge znakove ostavlja na miru.

Slide 4

Funkcije gets i puts: (<stdio.h>)

```
char *gets(char *s);
int puts(const chat *s);
```

Funkcija `gets` učitava znakove sa standardnog ulaza sve dok ne naidje na kraj linije koji zamijenjuje nul znakom `'\0'`. Funkcija vraća pokazivač na `char` koji pokazuje na učitani znakovni niz ili `NULL` ako se došlo do kraja ulaznih podataka ili se javila greška. Simbolička konstanta `NULL` definirana je `<stdio.h>`.

Funkcija `puts` uzima kao argument znakovni niz koji će biti ispisan na standardnom izlazu. Funkcija vraća cijeli broj (`int`) veći ili jednak od nule ako je ispis uspio te `EOF` ako ispis nije uspio. Prije ispisa `puts` dodaje znak `'\n'` na kraju znakovnog niza.

Slide 5

Primjer:

```
#include <stdio.h>
/* kopiranje ulaza na izlaz */

int main(void) {
   char red[128];

   while(gets(red)!=NULL)
        puts(red);

   return 0;
}
```

Osnovni nedostatak funkcije `gets` je u tome što nije moguće odrediti maksimalni broj znakova koji će biti učitan. Ukoliko je broj znakova na ulazu veći od dimenzije argumenta funkcije `gets` doći će do greške.

Slide 6

Funkcija scanf

:

služi učitavanju podataka sa standardnog ulaza. Opća forma:

```
scanf(kontrolni_string, arg_1, arg_2, ... ,arg_n)
```

gdje je `kontrolni_string` konstantni znakovni niz koji sadrži informacije o vrijednostima koje se učitavaju u argumente `arg_1`,...,`arg_n`.

Kontrolni znakovni niz (string) je konstantan znakovni niz koji se sastoji od individualnih grupa znakova pri čemu je svakom argumentu pridružena jedna grupa. Svaka grupa znakova započinje znakom postotka (%) kojeg slijedi *znak konverzije* koji upućuje na tip podatka koji se učitava. Npr. %c ili %d itd.

Slide 7

Najčešće korišteni znakovi konverzije su sljedeći:

znak konverzije	tip podatka koji se učitava
c	jedan znak (`char`)
d	decimalni cijeli broj (`int`)
e,f,g	broj s pokretnim zarezom (`float`)
h	kratak cijeli broj (`short`)
i	decimalni, heksadecimalni ili oktalni cijeli broj (`int`
o	oktalni cijeli broj (`int`)
u	cijeli broj bez predznaka (`unsigned int`)
x	heksadecimalni cijeli broj (`int`)
s	string (`char *`)
p	pokazivač (`void *`)

Slide 8

- Unutar kontrolnog niza znakova grupe znakova mogu se nastavljati jedna na drugu bez razmaka ili mogu biti odvojene bjelinama (prazno mjesto, tabulator, prijelaz u novu liniju). Bjeline će u ulaznim podacima biti učitane i ignorirane.
- Argumenti funkcije `scanf` mogu biti samo pokazivači na varijable. Ukoliko podatak treba učitati u neku varijablu, onda `scanf` uzima kao argument adresu te varijable.
- Podaci koje `scanf` čita dolaze sa standardnog ulaza što je tipično tastatura. Ako se unosi više podataka oni moraju biti separirani bjelinama (prijelaz u novi red = bjelina). Numerički podaci na ulazu moraju imati isti oblik kao i numeričke konstante.

Slide 9

Unos cijelih brojeva

Cijeli brojevi mogu biti uneseni kao decimalni (`%d`) ili kao oktalni i heksadecimalni (`%i`). Znak konverzije (`%i`) interpretira ulazni podatak kao oktalan broj ako mu prethodi nula ili kao heksadecimalan broj ako mu prethodi `0x` ili `0X`. Npr. ako program

```
int x,y,z;
......
scanf("%i %i %i",&x,&y,&z);
```

učitava ulaznu liniju

```
 13  015    0Xd
```

onda će `scanf` u sve tri varijable (`x`, `y` i `z`) učitati vrijednost 13 (decimalno).

Slide 10

Unos cijelih brojeva

Cijeli brojevi u oktalnom i heksadecimalnom zapisu mogu se upisivati i pomoću znakova konverzije `%o` i `%x`. Ti znakovi konverzije interpretiraju ulazne podatke kao oktalne odnosno heksadecimalne i stoga ne zahtijevaju da oktalna konstanta započinje s nulom, a heksadecimalna s `0x` ili `0X`. Kôd

```
int x,y,z;
......
scanf("%d %o %x",&x,&y,&z);
```

ispravno će pročitati ulazne podatke

```
 13  15  d
```

i svim varijablama pridružiti vrijednost 13 (decimalno).

Slide 11

Unos cijelih brojeva

- Podatak učitavamo u varijablu tipa `unsigned` znakom konverzije `%u`.
- Znakovi konverzije `d`, `i`, `o`, `u`, `x` mogu dobiti prefiks `h` ako je argument pokazivač na `short` te prefiks `l` ako je argument pokazivač na `long`.

Na primjer, kôd

```
int x;
short y;
long z;
......
scanf("%d %hd %ld",&x,&y,&z);
```

učitava tri decimalna cijela broja i konvertira ih u varijable tipa `int`, `short` i `long`.

Slide 12

Unos realnih brojeva

Znakovi konverzije `e`, `f`, `g` služe za učitavnje varijable tipa `float`. Ukoliko se učitava vrijednost u varijablu tipa `double` treba koristiti prefiks `l` (`le`, `lf` ili `lg`). Na primjer,

```
float x;
double y;
long double z;
......
scanf("%f %lf %Lf",&x,&y,&z);
```

Prefiks `L` se koristi ako je argument pointer na `long double`.

Slide 13

- Funkcija `scanf` dijeli niz znakova na ulazu u polja znakova odvojena bjelinama.
- Svako polje znakova intrepretira se prema odgovarajućem znaku konverzije i upisuje u varijablu na koju pokazuje odgovarajući argument funkcije.
- Svaki znak konverzije učitava jedno ulazno polje.

U primjeru

```
scanf("%f%d",&x,&i);
```

znak konverzije `%f` učitava (i konvertira) prvo polje znakova. Pri tome se eventualne bjeline na početku preskaču. Prvo polje znakova završava bjelinom koju `%f` ne učitava. Drugi znak konverzije `%d` preskače sve bjeline koje odjeljuju prvo polje znakova od drugog polja znakova i učitava (i konvertira) drugo polje znakova.

Slide 14

Znakovi konverzije mogu biti odijeljeni bjelinama:

```
scanf("%f %d",&x,&i);
```

- Svaka bjelina u kontrolnom znakovnom nizu ima za posljedicu preskakanje svih bjelina na ulazu do početka novog ulaznog polja.

Stoga je pisanje znakova konverzije u kontrolnom znakovnom nizu razdvojeno bjelinam (kao u primjeru `"%f %d"`) ili nerazdvojeno (kao `"%f%d"`) posve ekvivalentno (to ne vrijedi za znakove konverzije `%c` i `[`).

Slide 15

U kontrolnom znakovnom nizu mogu se pojaviti i drugi znakovi osim bjelina i znakova konverzije. Njima moraju odgovarati posve isti znakovi na ulazu. Na primjer, ako realan i cijeli broj učitavamo naredbom

```
scanf("%f,%d",&x,&i);
```

onda ulazni podaci moraju biti oblika npr.

```
1.456, 8
```

bez bjeline izmedju prvog broja i zareza. Ako se želi dozvoliti bjelina prije zareza potrebno je koristiti naredbu

```
scanf("%f ,%d",&x,&i);
```

u kojoj bjelina nakon `%f` preskače sve eventualne bjeline na ulazu ispred zareza.

Slide 16

Unos stringova (%s)

Znak konverzije `s` učitava niz znakova; niz završava prvom bjelinom u ulaznom nizu znakova. Iza posljednjeg učitanog znaka automatski se dodaje nul-znak (\0). Na primjer

```
char string[128];
int x;
......
scanf("%s%d",string,&x);
```

Budući da se svako polje kao argument funkcije interpretira kao pokazivač na prvi elemet polja, ispred varijable `string` ne stavlja se adresni operator.

Znakom konverzije `%s` nije moguće učitati niz znakova koji sadrži u sebi bjeline jer bjeline služe za ograničavanje ulaznog polja.

Slide 17

Unos stringova (%[...])

- Unutar uglatih zagrada upisuje se niz znakova.
- Funkcija `scanf` će učitati u pripadni argument najveći niz znakova sa ulaza koji se sastoji od znakova navedenih unutar uglatih zagrada.
- Učitavanje završava prvi znak na ulazu koji nije naveden u uglatim zagradama i na kraj učitanog niza dodaje se nul znak.
- Vodeće bjelina se ne preskaču.

Na primjer, naredba učitava najveći niz znakova sastavljen od velikih slova i razmaka:

```
char linija[128];
........
scanf(" %[ ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);
```

Slide 18

Napomena

Uočimo da smo prije `%[` ostavili jedan razmak koji govori funkciji `scanf` da preskoči sve bjeline koje prethode znakovnom nizu. To je nužno ukoliko smo imali prethodni poziv `scanf` funkcije. Naime `scanf` uvijek ostavlja završni znak prijelaza u novi red u ulaznom nizu tako da bi naredba

```
scanf("%[ ABCDEFGHIJKLMNOPRSTUVWXYZ]", linija);
```

pročitala prethodni znak prijelaza u novi red i budući da on nije u unutar uglatih zagrada završila bi čitanje ulaznih podataka i `linija` ne bi bila učitana.

Slide 19

Unos stringova

S uglatim zagradama možemo koristiti sintaksu

```
scanf(" %[^niz znakova]", linija);
```

U `linija` će biti učitan najveći mogući niz znakova sastavljen od svih znakova osim onih koji se nalaze u uglatim zagradama.

Na primjer, učitati cijelu liniju bez znaka za novi red možemo pomoću naredbe

```
scanf(" %[^\n]", linija);
```

Na kraj učitanog niza znakova bit će dodan \0, a ispred `%[` mora biti ostavljeno prazno mjesto kako bi bili preskočene sve prethodne bjeline.

Slide 20

Unos pojedinačnih znakova (%c)

Znak konverzije `c` učitava jedan znak u varijablu bez obzira je li on bjelina ili ne.

Ako je prvi znak konverzije `c` potrebno je ispred njega staviti jednu bjelinu kako ne bi pročitao znak za prijelaz u novi red eventualne prethodne `scanf` funkcije.

Primjer:

Kontrolni niz `" %c%c%c"` čita tri znaka. Počet će s prvim znakom koji nije bjelina (zbog bjeline ispred prvog `%c` znaka) i pročitat će tri uzastopna znaka bili oni bjeline ili ne. Ako se želi čitati samo znakove bez bjelina treba koristiti `" %c %c %c"`.

Slide 21

Maksimalna širinu ulaznog polja

Uz svaki kontrolni znak moguće je zadati maksimalnu širinu ulaznog polja koje će se učitati tako da se ispred kontrolnog znaka stavi broj koji odredjuje širinu polja. Tako na primjer `%3d` učitava cijeli broj od najviše tri znamenke, a `%11s` učitava najviše 11 znakova. Ukoliko podatak sadrži manje znakova od zadane maksimalne širine polja učita se podatak samo do prve bjeline. Ako pak podatak ima više znamenaka od maksimalne širine polja višak znamenaka će biti učitan sljedećim konverzijskim znakom ili sljedećom `scanf` funkcijom.

Slide 22

Povratna vrijednost funkcije `scanf`

Funkcija `scanf` vraća broj uspješno učitanih podataka ili `EOF`.

Primjer: učitavanje brojeva većih ili jednakih od nule:

```
int n;
while(scanf("%d",&n) == 1 && n>= 0)
{
   //  radi nesto s brojem
}
```

`while` petlja se prekida ako je učitan negativan broj ili ako unos broja nije uspio.

Slide 23

Funkcija `printf`

Funkcija `printf` služi za ispis podataka na standardnom izlazu. Opća forma funkcije je

```
printf(kontrolni_string, arg_1, arg_2, ... ,arg_n)
```

gdje je `kontrolni_string` konstantan znakovni niz koji sadrži informaciju o formatiranju ispisa argumenata `arg_1`,...,`arg_n`.

Kontrolni znakovni niz ima posve istu formu i funkciju kao kod funkcije `scanf`.

Pojedine grupe znakova unutar kontrolnog znakovnog niza mogu se nastavljati jedna na drugu ili biti medjusobno razmaknute bjelinama ili nekim drugim znakovima. Svi ti znakovi bit će ispisani onako kako su uneseni.

Slide 24

Najčešće korišteni znakovi konverzije su sljedeći:

znak konverzije	tip podatka koji se ispisuje
d,i	decimalni cijeli broj (`int`)
u	cijeli broj bez predznaka (`unsigned int`)
o	oktalni cijeli broj (`int`)
x	heksadecimalni cijeli broj (`int`)
e,f,g	broj s pokretnim zarezom (`double`)
c	jedan znak (`char`)
s	string (`char *`)
p	pokazivač (`void *`)

Slide 25

Argumenti `printf` funkcije mogu biti konstante, varijable, izrazi ili polja. Na primjer, naredba

```
double x=2.0;
......
printf("x=%d, y=%f\n",x,sqrt(x));
```

će ispisati

```
x=2.000000, y=1.414214
```

Svi znakovi koji nisu znakovi konverzije ispisani su onako kako su uneseni u kontrolnom znakovnom nizu `"x=%d, y=%f\n"`. Ako treba ispisati znak `%`, onda unutar kontrolnog znakovnog niza na tom mjestu treba staviti `%%`.

Slide 26

- Pomoću znaka konverzije `%f` ispisujemo varijable tipa `float` i `double`.
- Pomoću `%d` možemo ispisati varijable tipa `int`, `char` i `short`.

Na primjer,

```
char c='w';
......
printf("c(int)=%d, c(char)=%c\n",c,c);
```

ispisat će

```
c(int)=119, c(char)=w
```

ukoliko računalo koristi ASCII skup znakova (119 je ASCII kod znaka "w").

Slide 27

Oktalni i heksadecimalni ispis

Pomoću znakova konverzije `%o` i `%x` cijeli brojevi se ispisuju u oktalnom i heksadecimalnom obliku bez vodeće nule odn. `0X`. Na primjer,

```
short i=64;
......
printf("i(okt)=%o: i(hex)=%x: i(dec)=%d\n",i,i,i);
```

ispisuje

```
i(okt)=100: i(hex)=40: i(dec)=64
```

Slide 28

Ispis brojeva tipa `long`

Izrazi tipa `long` ispisuju se pomoću prefiksa `l`. Na primjer, program

```
#include <stdio.h>
#include <limits.h>
long i=LONG_MAX;
main(){
  printf("i(okt)=%lo: i(hex)=%lx: i(dec)=%ld\n",i,i,i);
}
```

(ovisno o računalu na kojem se izvršava) može ispisati

```
i(okt)=17777777777: i(hex)=7fffffff: i(dec)=2147483647
```

Simbolička konstanta `LONG_MAX` definirana je u datoteci zaglavlje `<limits.h>` i predstavlja najveći broj tipa `long`.

Slide 29

Ispis realnih brojeva

Brojeve tipa `float` i `double` možemo ispisivati pomoću znakova konverzije `%f`, `%g` i `%e`. U konverziji tipa `f` broj se ispisuje bez eksponenta, a u konverziji tipa `e` s eksponentom. U konverziji tipa `g` način ispisa (s eksponentom ili bez) ovisi o vrijednosti koja se ispisuje. Naredba,

```
double x=12345.678;
.........
printf("x(f)=%f: x(e)=%e: x(g)=%g\n",x,x,x);
```

će ispisati

```
x(f)=12345.678000: x(e)=1.234568e+004: x(g)=12345.7
```

Slide 30

Minimalna širina ispisa

`%3d`, `%9s`, `%4d` itd. Broj označava minimalni broj znakova u ispisu.

Podatak koji sadrži manje znakova od zadane minimalne širine polja bit će dopunjen vodećim bjelinama do pune širine.

Podatak koji ima više znamenaka od minimalne širine polja bit će ispisan sa svim potrebnim znamenkama.

Sljedeća naredba

```
double x=1.2;
.........
printf("%1g\n%3g\n%5g\n",x,x,x);
```

ispisuje

```
1.2
1.2
  1.2
```

Slide 31

Preciznost ispisa realnih brojeva

`%a.bf` ili `%a.bg` ili `%a.be`

- `a` = minimalna širina ispisa,
- `b` = preciznost = broj decimala koje će biti ispisane.

Sljedeći program ispisuje broj π:

```
#include <stdio.h>
#include <math.h>
int main(void){
   double pi=4.0*atan(1.0);
   printf("%5f  %5.5f  %5.10f\n",pi,pi,pi); return 0;
}
```

Rezultat ispisa će biti

```
3.141593  3.14159  3.1415926536
```

Ispis bez specificirane preciznosti daje šet decimala.

Slide 32

Dinamičko zadavanje širine i preciznosti

- Iznos širine (preciznosti) u formatu zamijenjuje `*`.
- Iznos širine (preciznosti) odredjuje cjelobrojna varijabla na odgovarajućem mjestu.

U primjeru

```
int main(void){
    double pi=4.0*atan(1.0); int i=10;
    printf("%*f\n %*.*f\n %5.*f\n",11,pi,16,14,pi,i,pi);
    return 0;
}
```

dobivamo ispis

```
    3.141593
3.14159265358979
3.1415926536
```

Slide 33

Ispis stringova (%s)

Na primjer,

```
char naslov[]="Programski jezik C";
 ....
printf("%s\n",naslov);
```

ispisat će

```
Programski jezik C
```

i prijeći u novi red.

Preciznost = maksimalan broj znakova koji će biti ispisan.

Npr. `%5.12s` specificira da će biti prikazano minimalno 5 znakova (dopunjenih bjelinama kao treba), a maksimalno 12 znakova.

Slide 34

Primjer

```
char naslov[]="Programski jezik C";
   ....
printf("%.16s\n",naslov);
```

ispisat će

```
Programski jezik
```

Slide 35