# HINTS

* **C razlikuje velika i mala slova**

## Tipovi podataka

* int (cjelobrojni tip)
* float (realni tip)
* double (realni tip u dvostr. preciznosti)
* char (znakovni tip ili mali broj)

## Brojevni sustavi

* Binarni sustav: 0,1
* Oktalni: 0,1,2,3,4,5,6,7
* Heksadekadski: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A(10), B(11), C(12), D(13), E(14), F(15)

## Tehnika dvojnog komplementa

* Pozitivni brojevi – „normalno se pretvore“ i tako ostave
* Negativni brojevi – „normalno se pretvore“, nakon toga 0 🡪 1 i 1 🡪 0, te se doda 1

## Prikaz brojeva u registru s N bitova

* Bez predznaka: [0, 2n – 1]
* S predznakom: [-2n-1, 2n-1 – 1]

## Prefiksi za cjelobrojni tip podataka

* Short
* Long
* unsigned (samo pozitivne vrijednosti)
* signed (pozitivne i negativne vrijednosti)

## Konstante pisane u notacijama

* Dekadska notacija: 7, 34, -876, 67887654
* Oktalna notacija: 07, 034, 0666, 04567755
* Heksadekadska notacija: 0x45, 0xA34, 0x6E
* Konstantama se na kraju može dodati U ili u (za unsigned)

## Realni brojevi

### Realni brojevi jednostruke preciznosti

* Koriste 4 okteta (32 bita)
  + 1 bit predznak (1 za negativno „-„ i 0 za pozitivno „+“)
  + 8 bita za karakteristiku (K)
    - K = BE + 127, s time da je K€[0,255], BE €[-126,127]
  + 23 bita (mantisa bez skrivenog bita)
* Posebni slučajevi:
  + +0 (000000000000….0)
  + -0 (100000000000…0)
  + +∞ (011111111000000…0)
  + -∞ (111111111000000…0)
  + Denormalizirani broj: K=0 i postoje bitovi mantise !=0
  + NaN (Not a Number): K=255 i postoje bitovi mantise !=0

### Realni brojevi dvostruke preciznosti

* 8 okteta (64 bita)
  + 1 bit predznak (1 za negativno „-„ i 0 za pozitivno „+“)
  + 11 bitova za karakteristiku (K)
    - K = BE + 1023, K€ [0, 2047], BE€ [-1022, 1023]
  + mantisa (M) 52 bita
* Posebni slučajevi gotovo isti kao kod jednostruke preciznosti, samo K = 0 ili K = 2047 ….

### Realne konstante

* 2.f, 2.34F, -1.34e5f (float)
* 1., 2.34, 9e-8, 8.345e+25 (double)
* 1.L, 2.34L, -2.5e-37L (long double)

|  |  |
| --- | --- |
| Tip | Veličina |
| Char, unsigned char, signed char | 1 oktet |
| Short, unsigned short | 2 okteta |
| Int, unsigned int | 4 okteta |
| Long, unsigned long | 4 okteta |
| Float | 4 okteta |
| Double | 8 okteta |
| Long double | 8 (10, 12, 16) okteta |

|  |  |  |
| --- | --- | --- |
| Deklaracija | Broj bita | Interval brojeva |
| Int | 32 | [-2147483648, 2147483647] |
| Signed int | 32 | [-2147483648, 2147483647] |
| Short | 16 | [-32768, 32767] |
| Short int | 16 | [-32768, 32767] |
| Signed short int | 16 | [-32768, 32767] |
| Unsigned int | 32 | [0U, 4294967295U] |
| Unsigned short int | 16 | [0U, 65535U] |
| Long | 32 | [-2147483648, 2147483647] |
| Long int | 32 | [-2147483648, 2147483647] |
| Signed long int | 32 | [-2147483648, 2147483647] |
| Unsigned long int | 32 | [0U, 4294967295U] |

**Operator: Značenje:**

+ Zbrajanje

- Oduzimanje

\* Množenje

/ Dijeljenje

% Ostatak kod cjelobrojnog dijeljenja (modulo, modulus)

# PRVI CIKLUS

## INPUT - OUTPUT

* scanf(“%[tip podatka koji se ispisuje]“, &[varijabla koja se ispisuje]);
* printf(“[tekst koji se ispisuje] %[tip podatka koji se ispisuje]“, [varijabla koja se ispisuje]);
* prilagodba širine i preciznost ispisa realnog broja
  + %f – širina po potrebi, 6 decimala (default)
  + %[x].[y]f - x min. ukupna širina, y decimala
* **znanstveni zapis broja**
  + %e – širina po potrebi, 6 decimala (default)
  + %[x].[y]e – x min. ukupna širina, y decimala

|  |  |  |  |
| --- | --- | --- | --- |
| **TIPOVI PODATAKA I KONVERZIJSKI SPECIFIKATORI** | | | |
| **TIP PODATKA** | **INTEGER/FLOAT** | **OPCIONALNI KVALIFIKATORI** | **KONVERZIJSKA SPECIFIKACIJA** |
| int | integer | ∅ | %d |
| unsigned | %u |
| short | %hd |
| short unsigned | %hu |
| long | %ld |
| long unsigned | %lu |
| long long | %lld |
| long long unsigned | %llu |
| char | integer | ∅ | %c, %d |
| string | integer | ∅ | %s |
| float | float | ∅ | %f |
| double | float | ∅ | %lf |
| long | %Lf |
| znanstveni zapis\* | float | ∅ | %e |

\*naveden u tablici samo da bi se sve konverzijske specifikacije nalazile na jednom mjestu

### scanf

* scanf("%[znakovi]", niz1); - u niz1 zapisuje samo charove koji se nalaze u znakovi
* scanf("%3d", &i); - čita samo do prva tri znaka u toku

### printf

* %[znak][širina][.preciznost][modifikator]specifikator
  + 0 – dodaje vodeće nule
  + + - dodaje predznak i na pozitivnim brojevima
  + - - lijevo pozicioniranje u polju ispisa
* stringovi su zapisani u polja tipa char duljine broj\_slova+1 jer je zadnji znak u polju “\0“ koji predstavlja kraj stringa
* imena tipova podataka se mogu mijenjati naredbom typedef
  + typedef stari\_naziv novi\_naziv;
* size\_t je tip podataka u koji se može zapisati veličina varijable u bajtovima (sizeof(x)), unsigned int

## MATEMATIČKE FUNKCIJE

* Potreban “#include <math.h>“
  + sqrt(x)
  + pow(x,y)
  + sin(x)…

## GRANANJA

### IF grananja

* if (logički\_izraz){  
   blok\_naredbi;  
  }
* else if (logički\_izraz){  
   blok\_naredbi;  
  }
* else {  
   blok\_naredbi;  
  }

### SWITCH – CASE grananja

* switch (integer){  
  case konstantni\_integer1:  
   blok\_naredbi1  
  case konstantni\_integer2:  
   blok\_naredbi2....  
  case konstantni\_integer3:  
   blok\_naredbiN  
  default:  
   blok\_naredbiDefault
* treba pripaziti gdje se stavlja naredba “break;“

## PETLJE

* for (početnaVrijednost; uvijetIzvršavanja; korak){  
   blok\_naredbi;  
  }
* while (logicki\_izraz){  
   blok\_naredbi;  
  }
* do {  
  blok\_naredbi;  
  } while (logicki\_izraz);

## POLJA

* tip\_podatka ime\_polja[duljina] = {elementi\_polja};
* elemente polja nije potrebno upisivati pri inicijalizaciji
* ako se pod elemente polja napiše “{0}“, svi članovi poprimaju vrijednost 0
* indeksi elemenata od 0 do n-1
* višedimenzionalna polja
  + tip\_podatka ime\_polja[duljina1][duljina2]..[duljinaN];
  + kod **dvodinemzionalnih** polja (moguce ih je predociti kao tablice), duljina1 predstavlja broj redaka, a duljina2 broj stupaca zamišljene tablice

## STRUKTURE

* struct naziv\_strukture {  
   tip\_elementa1 ime\_elementa1;  
   tip\_elementa2 ime\_elementa2;  
   ......  
   tip\_elementaN ime\_elementaN;  
  };
  + npr.   
    struct student{  
     int JMBAG;  
     int OIB;  
    };  
    struct student ferovac;  
    ferovac.JMBAG=123456789;  
    ferovac.OIB=123456789;
* također je moguće varijable definirati uz deklaraciju strukture
  + npr.  
    struct student{  
     int JMBAG;  
     int OIB;  
    } **ferovac**;  
    ferovac.JMBAG=123456789;  
    ferovac.OIB=123456789;
* strukture ne moraju nužno imati ime u deklaraciji
  + npr.  
    struct {  
     int JMBAG;  
     int OIB;  
    } **ferovac**;  
    ferovac.JMBAG=123456789;  
    ferovac.OIB=123456789;

## FUNKCIJE

* tip\_rezultata\* naziv\_funkcije (tip\_podatka\_parametra parametar\*\*) {  
   blok\_naredbi;  
   return rezultat\*\*;  
  }  
  \*ako funkcija nema rezultat (ne vraca ga u funkciju iz koje je pozvana), tip je “void“  
  \*\*bilo koji tip podatka osim polja, ako ne povlači parametre, napisati “void“
  + npr.  
    \_Bool paran (int broj) {  
     \_Bool parnost;  
     if (broj % 2 == 0) {  
    paran = 1;  
     else  
     paran = 0;  
     return parnost;  
    }

## NAREDBE VEZANE ZA KONTROLU TOKA (PETLJE, GRANANJA I FUNKCIJE)

* break;
* continue;
* goto [labela]; – makar se ne koristi
* return;

# DRUGI CIKLUS

## POKAZIVAČI

* tip\_podatka \*ime\_pokazivaca
* ime\_pokazivaca = &ime\_varijable – postavljanje pokazivača na lokaciju varijable
* \*ime\_pokazivaca – vrijednost na lokaciji ime\_pokazivaca
* pokazivači i funkcije
  + poziv funkcije
    - naziv\_funkcije (&ime\_pokazivaca);
  + definicija/deklaracija funkcije
    - tip\_podatka naziv\_funkcije (tip\_podatka \*ime\_pokazivaca)

## MEMORIJSKI RAZREDI

* auto – default, isto kao i kada se ne navede memorijski razred pri inicijalizaciji varijable
* register – iste karakteristike kao i auto, samo što se varijabla po **preporuci** sprema tako da je sadržaju varijable najbrže pristupiti, po mogućnosti u registru procesora umjesto u RAM-u, nije primjenjiv adresni operator
* static – trajnost od početka do kraja izvršavanja programa, vidljivost ovisi o mjestu deklaracije
* extern – definirane u jednom modulu, deklarirane u više njih, trajnost od početka do kraja izvršavanja programa, vidljivost ovisi o mjestu deklaracije

## STANDARDNA BIBLIOTEKA

### <time.h>

* time\_t – tip podatka za spremanje informacija o trenutku u vremenu
* time(NULL) – vraca broj sekundi od pocetka UNIX epohe

### <stdlib.h>

* srand, rand
  + void srand(unsigned int seed);   
    int rand(void);
  + realni brojevi iz intervala [a, b]   
    (double)rand() / RAND\_MAX \* (b - a) + a
* ciejli brojevi iz intervala [a, b]   
  rand() % (b - a + 1) + a
* exit
  + exit(integer) – vrijednosti nisu standardizirane ali najčešće 0 predstavlja uspješno izvršavanje programa, dok ostale vrijednosti predstavljaju grešku

### <math.h>

* fabs() – apsolutna vrijednost
  + int abs(int n);
  + long labs(long n);
  + long long llabs(long long n); \_\_\_\_ukljucene u <stdlib.h>
  + float fabsf(float x);
  + double fabs(double x);
  + long double fabsl(long double x);
* trigonometrijske i hiperbolne funkcije su tipa double i primaju podatke tipa double, mjere kuta su izražene u radijanima
* eksponencijalne i logaritamske funkcije su tipa double i primaju podatke tipa double
* double ceil(double x) – najmanji cijeli broj veći ili jednak x
* double floor(double x) – najveći cijeli broj manji ili jednak x

### <string.h>

* char \*strcpy(char \*s1, const char \*s2); - niz s2 kopira u niz s1
* char \*strncpy(char \*s1, const char \*s2, size\_t n); - ne više od n znakova niza s2 kopira u niz s1
* char \*strcat(char \*s1, const char \*s2); - dodaje niz s2 na kraj niza s1
* char \*strncat(char \*s1, const char \*s2, size\_t n); - ne više od n znakova niza s2 dodaje na kraj niza s1
* size\_t strlen(const char \*s); - vraća duljinu niza s, ne broji \0
* int strcmp(const char \*s1, const char \*s2); - vraća 0 ako su oba stringa identični, >0 ako je ASCII vrijednost prvog različitog chara iz s1 veća od chara s istim indeksom iz s2, <0 u obrnutom slučaju
* int strncmp(const char \*s1, const char \*s2, size\_t n); - isto kao u prethodnom slučaju, samo što uspoređuje prvih n znakova
* char \*strchr(const char \*s, int c); - traži prvu pojavu znaka c u stringu s
* char \*strrchr(const char \*s, int c); - traži zadnju pojavu znaka c u stringu s
* char \*strstr(const char \*s1, const char \*s2); - traženje prve pojave podniza u s1 koji je jednak nizu s2 (znak terminatora niza s2 se ne uzima u obzir kod usporedbe)
* char \*strpbrk(const char \*s1, const char \*s2); - traženje u nizu s1 prve pojave bilo kojeg od znakova navedenih u nizu s2

### <ctype.h>

* int toupper(int c);
* int tolower(int c);

Ima još funkcija al su sve na podsjetniku..

## DATOTEKE

### <stdio.h>

* FILE \*fopen(const char \*filename, const char \*mode);
* modovi za otvaranje i pristupanje datoteci

|  |  |  |
| --- | --- | --- |
|  | **ZNAČENJE** | **ŠTO SE DEŠAVA S DATOTEKOM** |
| **w** | write | ako datoteka postoji, briše sadržaj datoteke, inače stvara i otvara novu (praznu) datoteku |
| **a** | append | ako datoteka postoji, otvara tu datoteku, inače stvara i otvara novu (praznu) datoteku |
| **r** | read | ako datoteka postoji, otvara tu datoteku, inače funkcija fopen vraća NULL |
| **r+** | read+write |
| **w+** | write+read | ako datoteka postoji, otvara tu datoteku, inače stvara i otvara novu (praznu) datoteku, dodaje na kraj datoteke |
| **a+** | read+write |

* int fclose(FILE \*stream);
* size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);
* size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);
* long ftell(FILE \*stream);
* int fseek(FILE \*stream, long offset, int whence);
* int fflush(FILE \*stream); - trajno pohranjuje sadržaj međuspremnika toka, automatski se poziva kod fclose