Programski jezik C

`http://www.math.hr/nastava/c/`

`http://www.math.hr/~jurak/predavanja.html`

Slide 1

Uvod

Programski jezici

- Strojni jezik (izvršni program, instrukcije u binarnom kôdu)
- Asemblerski jezik (izvorni program, jezik prilagođen arhitekturi računala)
- Viši programski jezici, C, FORTRAN, C++, Java, Phyton, Perl,... (izvorni program, jezik prilagođen zadaćama za koje je namijenjen)

Slide 2

Strojni jezik	Ekvivalentan	asemblerski kôd
00011110	PUSH	DS
00101011	SUB	AX,AX
11000000		
10111000	PUSH	AX
10111000	MOV	AX,MYDATA
00000001		
10001110		
11011000	MOV	DS,AX

Slide 3

Viši programski jezici

- C, Pascal, FORTRAN, BASIC, LISP, Java, C++, Python, Perl, ...
- Neovisnost o arhitekturi računala
- Veća efikasnost programiranja
- Prilagođenost pojedinim specifičinm zadaćama

Slide 4

Programski jezik C

- Autor Dennis Ritchie (Bell Telephone Laboratories)
- Razvijen sedamdesetih godina
- Standardiziran 1989. ANSI-C, (C90 standard). Novi standard izašao 1999, (C99 standard)

Literatura:

- Brian W. Kernighan i Dennis M. Ritchie, The C Programming Language, Second Edition, Prentice Hall, 1988
- S. Prata, C Primer Plus, Fourth Edition, SAMS, 2002.
- B. S. Gottfried, Theory and Problems of Programming with C, Schaum's outline series, McGraw-Hill, 2000.

Slide 5

Alati za programiranje pod UNIXom

- Editor teksta (`pico, vi, ime.c, ime.h, ...`)
- Prevodilac (compiler) (`cc, gcc`)
- man (`man cc, man vi, man scanf`)
- Linker ili punjač (`ld`)
- Debugger (`dbx, gdb`), lint (verifikacija koda)
- make (`make`, `gmake`)
- Programske biblioteke (`-l` opcija)

Slide 6

Kompilacija pod UNIXom

1 Izvorni kôd: sastoji se od više `.c` i `.h` datoteka.

2 Prevodilac prevodi svaku `.c` datoteku u `.o` datoteku (objektni kôd).

3 Linker povezuje `.o` datoteke i programske biblioteke u izvršni kôd. Prevodilac ga poziva automatski.

4 Ispravljanje grešaka (debugger, lint) GO TO 1.

Slide 7

Programiranje kroz primjer. Primjer 1

```
#include <stdio.h>
int main(void)
{
   printf("Dobar dan.\n");
   return 0;
}
```

- Program spremiti u datoteku `prvi.c`
- `cc prvi.c`
- `./a.out`

Slide 8

- C program se sastoji od funkcija i varijabli.
- Izvršavanje počinje izvođenjem `main()` funkcije.
- Tijelo funkcije se sastoji od naredbi unutar vitičastih zagrada.
- Naredbe završavaju znakom točka-zarez.
- `printf(...)` funkcija za ispis na ekranu.
- Datoteka zaglavlja `stdio.h` je učitana zbog `printf()`.

Slide 9

Funkcija `printf()` ne ispisuje automatski znak za prijelaz u novi red (\n).

```
#include <stdio.h>

int main(void)
{
   printf("Dobar ");
   printf("dan.");
   printf("\n");
   return 0;
}
```

Slide 10

Primjer 2. Program koji ispisuje prvih 10 faktorijela.

```
#include <stdio.h>   /* Primjer 2 */
int main(void) {
    int n,fakt;       /*  Deklaracija varijabli  */

    n=1;              /* Izvrsna naredbe,        */
    fakt=1;           /* pridruzivanje           */
    while(n<=10)      /* while-petlja            */
    {
        fakt=fakt*n;
        printf(" %d  %d\n",n,fakt);
        n=n+1;
    }
    return 0;
}
```

Slide 11

Tipovi varijabli

`char`	jedan znak
`short`	"kratki" cijeli broj
`int`	cijeli broj
`long`	"dugi" cijeli broj
`float`	realni broj jednostruke preciznosti
`double`	realni broj dvostruke preciznosti

Slide 12

Relacijski operatori

operator	primjer	značenje
`<=`	`a<=b`	manje ili jednako
`>=`	`a>=b`	veće ili jednako
`<`	`a<b`	strogo manje
`>`	`a>b`	strogo veće
`==`	`a==b`	jednako
`!=`	`a!=b`	nejednako

Slide 13

Aritmetički operatori

operator	primjer
`+`	`a+b`
`-`	`a-b`
`*`	`a*b`
`/`	`a/b`

Realno dijeljenje: `3.0/2.0 = 1.5`

Cjelobrojno dijeljenje: `3/2 = 1`.

Slide 14

`while` petlja

```
while(uvjet)
    naredba;
```

Dok je `uvjet` istinit izvršava se naredba. Kad uvjet više nije ispunjen prelazi se na sljedeću naredbu.

```
while(uvjet)
{
    naredbe;
    ......
}
```

Dok je uvjet ispunjen ponavlja se grupa naredbi unutar vitičastih zagrada (tijelo petlje).

Slide 15

`for` petlja

Ispis faktorijela pomocu `for` petlje.

```
#include <stdio.h>  /*  Primjer 2  */
int main(void) {
    int n,fakt;

    fakt=1;
    for(n=1;n<=10;n=n+1)
    {
        fakt=fakt*n;
        printf(" %d  %d\n",n,fakt);
    }
    return 0;
}
```

Slide 16

Usporedba `for` i `while` petlje

```
for(n=1;n<=10;n=n+1)
{
   .... // tijelo petlje
}
```

je ekvivalentno sa:

```
n=1;
while(n<=10)
{
   ....  // tijelo petlje
   n=n+1;
}
```

Slide 17

Operatori inkrementiranja (++) i dekrementiranja (--)

`n++` je ekvivalentno s `n=n+1`,

`n--` je ekvivalentno s `n=n-1`,

Umjesto `n++` može se pisati `++n`;

Umjesto `n--` može se pisati `--n`;

```
for(n=1;n<=10;++n)
{
    fakt=fakt*n;
    printf(" %d  %d\n",n,fakt);
}
```

Slide 18

Primjer 3. Napisati program računa $\sum_{k=1}^{n} \frac{1}{k(k+1)}$.

```
#include <stdio.h>
int main(void) {
    int n,k;
    double suma;

    printf("Unesite broj n: n= ");
    scanf(" %d",&n);
    suma=0.0;

    for(k=1;k<=n;k++) {
        suma=suma+1.0/(k*(k+1));
    }
    printf("Suma prvih %d clanova = %f\n",n,suma);
    return 0;
}
```

Slide 19

Učitavanja brojeva pomoću funkcije `scanf()`

```
#include <stdio.h>
    ........
    int n;
    ......
    scanf(" %d", &n);
```

znak konverzije `%d` — ispisivanje i učitavanje cijelih brojeva;

znak konverzije `%f` — ispisivanje brojeva tipa `float` i `double`;

znak konverzije `%f` — učitavanje brojeva tipa `float`;

znak konverzije `%lf` — učitavanje brojeva tipa `double`;

`&` adresni operator (daje adresu varijable). `scanf` uzima adresu varijable.

Slide 20

Primjer 4. Treba napisati program koji rješava kvadratnu jednadžbu: $ax^2 + bx + c = 0$. Rješenja su općenito kompleksna i zadana su formulom

$$z_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Koristimo oznake $z_1 = x_1 + iy_1$, $z_2 = x_2 + iy_2$.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Resavanje kvadratne jednadzbe.
      a x^2 + b x + c = 0         */
```

Slide 21

```
int main(void) {
  double a, b, c, /* Koeficijenti jednadzbe        */
         d,       /* Diskriminanta                 */
         x1, x2,  /* Realni dijelovi korijena.     */
         y1, y2;  /* Imaginarni dijelovi korijena. */

  printf("Upisite koeficijente  kvadratne jednadzbe: a, b, c: ");
  scanf ("%lf%lf%lf", &a, &b, &c);

  y1=0.0;
  y2=0.0;
```

Slide 22


```
  if(a != 0.0) {
     d=b*b-4*a*c;
     if (d > 0) {
        x1=(- b + sqrt (d))/(2 * a);
        x2=(- b - sqrt (d))/(2 * a);
     } else if (d == 0) {
        x2=x1=- b/(2 * a);
     } else{
        x1=-b/(2 * a);        x2 = x1;
        y1=sqrt(-d)/(2 * a);  y2 = - y1;
     }
  }
  else {
     printf("Jednadzba nije kvadratna.\n");
     exit(-1);
  }
```

Slide 23

```
  printf("z1=%f + i*(%f), z2=%f + i*(%f)\n",x1, y1, x2, y2);
  return 0;
}   // Kraj main funkcije
```

`stdio.h` uključujemo radi funcija `printf` i `scanf`;

`stdlib.h` uključujemo radi funkcije `exit`;

`math.h` uključujemo radi funkcije `sqrt` (`sqrt(x)`=$\sqrt{x}$).

znak konverzije `%lf` — učitavanje brojeva tipa `double`;

Slide 24

Naredba uvjetnog granjanja `if`

```
if(uvjet)
  naredba1;
```

ili

```
if(uvjet)
  naredba1;
else
  naredba2;
```

ili

```
if(uvjet1)
  naredba1;
else if(uvjet2)
  naredba2;
else
  naredba3;
```

Slide 25

Primjer 5. Treba napisati funkciju koja računa binomni koeficijent

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{1 \cdot 2 \cdots k}. \qquad (1)$$

Funkcija treba uzeti dva cjelobrojna parametra `n` i `k` te vratiti broj $\binom{n}{k}$. U glavnom programu treba zatim ispisati tzv. Pascalov trokut prikazan niže. U liniji s indeksom `n` ispisani su brojevi $\binom{n}{k}$, za sve vrijednosti `k` od 0 do `n`.

```
n = 0                    1
n = 1                  1   1
n = 2                1   2   1
n = 3              1   3   3   1
n = 4            1   4   6   4   1
n = 5          1   5   10  10   5   1
n = 6        1   6   15  20  15   6   1
n = 7      1   7   21  35  35   21  7   1
n = 8    1   8   28  56  70  56   28  8   1
```

Slide 26


```
#include <stdio.h>
      /* Funkcija binom() racuna binomni koeficijent. */
long binom(int n, int k) {
   long rezult=1;   // varijablu inicijaliziramo kod deklaracije
   int    i;

   if(n == k) return 1;
   if(k == 0) return 1;

   for(i=n;i>n-k;i--)
      rezult=rezult*i;
   for(i=1;i<=k;++i)
      rezult=rezult/i;

   return rezult;
}
```

Slide 27

Radi jednostavnosti ispisat ćemo Pacalov trokut poravnan na lijevoj strani.

```
int main(void) {  /* Glavni program */
   long bnm;
   int i, j;

   for(i=0;i<10;i++){
      for(j=0;j<=i;j++){
         bnm=binom(i,j);
         printf("%ld ",bnm);
      }
      printf("\n");
   }
   return 0;
}
```

Slide 28

Definicija funkcije

```
tip ime_funkcije(tip_1 arg_1, tip_2 arg_2, ...)
{
     deklaracije varijabli
     narebe
}
```

Primjer:

```
long binom(int n, int k) {
   long rezult=1;  // varijablu inicijaliziramo kod deklaracije
   int    i;

   if(n == k) return 1;
   ...................
   return rezult;
}
```

Slide 29

Poziv funkcije: `bnm=binom(i,j);`

- Stvarni argumenti funkcije: `i, j`;
- Formalni argumenti: `n, k`;
- Stvarni argumenti se kopiraju u formalne.
- Vraćena vrijednost (`rezult` se kopira u `bnm`)

Tko poziva funkciju `main()`? Operacijski sustav. Vraćena vrijednost je izlazni status:

- `return 0; -->` normalan završetak programa;
- `return n;` $n \neq 0$ `-->` završetak uslijed greške.

Slide 30

Prototip funkcije

Pravilo: U trenutku kada dođe do funkcijskog poziva, compiler mora znati tip funkcije te broj i tip njenih argumenata.

Mogućnosti:

1. Definirati funkciju ispred mjesta na kojem se poziva (ispred `main` funkcije);
2. Uvesti prototip funkcije.

Prototip je deklaracija funkcije. Dobije se tako da se u definiciji ispusti tijelo funkcije i zadrži samo *signatura* funkcije, završena sa znakom točka-zarez.

Slide 31

Primjer. Definiciju funkcije `binom` smještamo iz funkcije `main`.

```
#include <stdio.h>
long binom(int n, int k);  // PROTOTIP FUNKCIJE binom

int main(void) {  /* Glavni program */
    /* Sve isto kao
       i ranije */
}

long binom(int n, int k) {  // DEFINICIJA FUNKCIJE binom
   long rezult=1;
   int    i;

   /* Sve isto kao
             i ranije */
}
```

Slide 32

Primjer 6. Napisati program koja ispituje okomitost vektora u $\mathbb{R}^3$.

Matematički pojmovi vektora i matrica implementiraju se u C-u pomoću *polja*.

- Polje je niz indeksiranih varijabli istog tipa, smještenih u memoriji na uzastopnim lokacijama. Indeks polja uvijek kreće od nule.

Koristimo formule:

$$\cos\phi = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|},$$

$$\vec{a}\cdot\vec{b} = a_1b_1 + a_2b_2 + a_3b_3, \quad \|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2}.$$

Uvjet okomitosti: (zbog grešaka zaokruživanja)

$$|\cos\phi| < \varepsilon.$$

Mi uzimamo $\varepsilon = 10^{-10}$.

Slide 33

```
#include <stdio.h>
#include <math.h>

double epsilon=1.0E-10;                       // GLOBALNA VARIJABLA
double kosinus_kuta(double x[], double y[]); // PROTOTIP

int main(void) {
    double a[3], b[3];   // polja, tj. vektori a i b
    double cos_phi;
    int i;

    printf("Unesite vektor a.\n");
    for(i=0;i<3;++i){
        printf("a[%d]= ",i+1); scanf(" %lf",&a[i]);
    }
    printf("Unesite vektor b.\n");
```

Slide 34


```
    for(i=0;i<3;++i){
        printf("b[%d]= ",i+1); scanf(" %lf",&b[i]);
    }

    cos_phi= kosinus_kuta(a, b);

    if(fabs(cos_phi) < epsilon){
        printf("Vektori su okomiti.\n");
        printf("Kosinus kuta = %f\n", cos_phi);
    }
    else{
        printf("Vektori nisu okomiti.\n");
        printf("Kosinus kuta = %f\n", cos_phi);
    }
    return 0;
}
```

Slide 35

Deklaracija polja

`tip ime[MAX];`

- `ime` je ime polja;
- `tip` je tip polja (`double`, `int`,...);
- `MAX` je broj elemenata polja (konstanta).

Elementi polja : `ime[0], ime[1], ... ,ime[MAX-1]`

Primjer:

```
double x[10];
```

Elementi: `x[0], x[1], ... ,x[9]` su varijable tipa `double`.

Slide 36

Definicije funkcija: `norma`, `produkt` i `kosinus_kuta`.

```
double norma(double x[])
{
    int i;
    double suma;

    suma=0.0;
    for(i=0;i<3;++i)
        suma = suma + x[i]*x[i];

    return sqrt(suma);
}
```

Slide 37

```
double produkt(double x[], double y[]) {
    int i;
    double suma;

    suma=0.0;
    for(i=0;i<3;++i) suma = suma + x[i]*y[i];
    return suma;
}
double kosinus_kuta(double x[], double y[]) {
    double cos_phi;

    cos_phi=produkt(x,y);
    cos_phi=cos_phi/(norma(x)*norma(y));

    return cos_phi;
}
```

Slide 38

Globalne i lokalne varijable

- Svaka varijabla deklarirana unutar neke funkcije (pa i `main`) je lokalna. Ona se kreira kada započne izvršavanje funkcije i uništava kada završi. Primjer: varijable `suma` i `i` u funkciji `produkt()`.
- Varijabla deklarirana izvan svih funkcija, na početku datoteke, je globalna. Ona postoji od početka do kraja izvršavanja programa i njoj se može pristupiti u svakoj funkciji. Primjer: `epsilon`.

Posljedica: lokalnim varijablama u različitim funkcijama možemo davati ista imena.

Slide 39

Matematičke funkcije

Sve uzimaju argument tipa `double` i vraćaju vrijednost tipa `double`. Za njihovo korištenje treba uključiti datoteku `math.h`.

`sin`, `cos`, `tan` (tg),
`asin` (arcsin), `acos` (arccos),
`exp` (e^x), `log` (ln), `log10` ($\log_{10}$)
`fabs` (`fabs(x)`=|x|), `sqrt` (`sqrt(x)`=$\sqrt{\texttt{x}}$),

i mnoge druge.

Operator potenciranja ne postoji. Koristi se funkcija

`double pow(double,double)`
`pow`$(x, y) = x^y$ (dozvoljava $x < 0$ ako je $y \in \mathbb{Z}$)

Slide 40

Primjer 7. Treba ispisati adresu varijable u memoriji.

```
#include <stdio.h>
/* Adresni operator & */
 int main(void)
 {
     double x=5;

     printf("x = %f, adresa od x= %p\n",x,&x);
     /*  %p ispisuje adresu  */
     return 0;
 }
```

`&` je adresni operator. `&x` je adresa varijable `x`.

`%p` znak konverzije za ispis adrese.

Slide 41

Adresu varijable možemo zapamtiti u pokazivačkoj varijabli.

```
#include <stdio.h>
 /* Daje posve isti rezultat kao i prethodni
    program samo koristi pokazivac */

 int main(void)
 {
     double x=5;
     double *px;

     px=&x;
     printf("x = %f, adresa od x= %p\n",x,px);
     printf("x = %f, adresa od x= %p\n",*px,px);
     return 0;
 }
```

Slide 42

Deklaracija pokazivača (pointera)

```
tip *ime;
```

`ime` je varijabla tipa pokazivač na `tip`. Služi pamćenju adresa varijabli tipa `tip`. Primjeri:

```
double y,x=5.0; /* varijable tipa double */
double *px;     /* pokazivac na double */
int    i,*m; /* i = var. tipa int, m = pokazivac na int */
```

Inicijalizacija pokazivača adresom varijable:

```
px = &x;   /* &x je adresa od x */
```

Operator dereferenciranja `*` daje sadržaj lokacije na koju pokazivač pokazuje (tj. čiju adresu ima):

```
y = *px;   /* isto sto i y=x (px pokazuje na x) */
```

Slide 43

Primjer 8. Treba napisati funkciju koja uzima niz znakova i u njemu nalazi prvi samoglasnik. Funkcija treba vratiti nađeni samoglasnik i mjesto u nizu na kojem je nađen. Ukoliko u danom nizu nema samoglasnika, umjesto mjesta u nizu vraća se -1.

U ovom primjeru funkcija mora vratiti dvije vrijednosti. Kroz `return` naredbu može se vratiti samo jedna vrijednost, tako da se druga mora vratiti kroz argument funkcije. (Funkcija ne može vratiti polje.)

Signatura funkcije je sljedeća:

```
int trazi(char linija[], int n, char *psamoglasnik);
```

gdje je

- `linija[]` = niz znakova u kojem se traži samoglasnik;
- `n` = broj znakova u nizu `linija[]`;
- `psamoglasnik` = pokazivač na nađeni samoglasnik.

Slide 44

```
#include <stdio.h>
int trazi(char linija[], int n, char *psamoglasnik) {
    int i;
    char c;

    for(i=0; i<n; i++){
       c=linija[i];
       if(c == 'a' || c == 'e' || c == 'i'
                   || c == 'o' || c == 'u')
        {
            *psamoglasnik=c;
             return i;
        }
    }
    return -1;
}
```

Slide 45

Prijenos argumenata

- Argumenti se funkciji prenose "po vrijednosti" (call by value); što znači da funkcija dobiva kopije stvarnih argumenata;
- Iznimka su polja koja se ne prenose "po vrijednosti". Prilikom poziva funkcije argument tipa polje konvertira se u pokazivač na prvi element polja. Funkcija tada može dohvatiti i promijeniti svaki element polja.
- Funkcija koja treba promijeniti varijablu u glavnom programu (koja nije polje) mora dobiti pokazivač na varijablu i koristiti operator dereferenciranja.

Slide 46

Pravila:

- Deklaracija argumenta tipa polja ne treba dimenziju polja: `char linija[];`
- Znakovne konstante se dobivaju stavljanjem znaka u jednostruke navodnike: `'a', '9', '\n'` itd.
- `*psamoglasnik=c;` operator dereferenciranja može stajati na lijevoj strani izraza pridruživanja.

Logički operatori:

Operator	Značenje
`&&`	logičko I
`||`	logičko ILI
`!`	logička negacija (unarno)

Slide 47

```
int main(void)    // Glavni program
{                 // Inicijalizacija polja u viticastim zagradama
    char ime[]={'P','r','o','g','r','a','m','s','k','i',' '
        'j','e','z','i','k',' ','C','.'};
    char znak;
    int no;

    no=trazi(ime,19,&znak);
    if(no != -1){
            printf("Prvi samoglasnik = %c.\n",znak);
            printf("Nalazi se na mjestu %d.\n",no);
    }
    else    printf("Nema samoglasnika.\n");

    return 0;
}
```

Slide 48

Nizovi znakova i znakovni nizovi (stringovi)

Niz znakova je bilo koji niz znakova.

```
char poruka[128];   // poruka = niz od 128 znakova
```

Znakovni niz (li string) je niz znakova koji završava s nul-znakom '\0'. Uloga tog znaka je da označava kraj niza.

Svaki niz znakova unutar dvostrukih navodnika je konstantan znakovni niz: Na primjer,

```
"Dobar dan\n"
```

se sastoji od znakova

```
D o b a r  d a n \n \0
```

Slide 49

Primjer 9. Treba napisati program koji učitava niz znakova limitiran bjelinama, računa broj numeričkih znakova u nizu i ispisuje učitani niz u obrnutom poretku.

```
#include <stdio.h>
#include <string.h>
#define MAX 128

void inv(char []);   /* Prototip */

/* Program ucitava niz znakova limitiran bjelinama,
   racuna broj numerickih znakova i ispisuje
   ucitani niz u obrnutom poretku                */
```

Slide 50


```
int main(void)  /*   GLAVNI PROGRAM */
{
    char niz_znakova[MAX],c;
    int i,brojac=0;

    scanf("%s", niz_znakova); i=0;
    while((c=niz_znakova[i]) !='\0')
    {
       if(c>='0' && c<='9') ++brojac;
       i++;
    }
    printf("Broj numerickih znakova = %d\n",
                                        brojac);
    inv(niz_znakova);
    printf("%s\n",niz_znakova);
}
```

Slide 51

- Glavni program učitava datoteku zaglavlja `string.h` zbog funkcije `strlen()`.
- Funkcija `strlen()` daje broj znalova u stringu, ne računajući zadnji nul-znak (\0).
- Naredba

  ```
  #define MAX 128
  ```

  je preprocesorska direktiva kojom se uvodi simboličko ime `MAX`. Preprocesor svako pojavljivanje simbola `MAX` zamijenjuje sa 128.

Slide 52

Izraz pridruživanja u složenim izrazima

Kôd

```
i=0;
while((c=niz_znakova[i]) !='\0') {
    if(c>='0' && c<='9') ++brojac;
    i++;
}
```

je ekvivalentno s kodom

```
i=0; c=niz_znakova[0];
while(c !='\0') {
   if(c>='0' && c<='9') ++brojac;
   i++;
   c=niz_znakova[i];
}
```

Slide 53

```
/*  INVERTIRANJE NIZA */

void inv(char s[])
{
    int c,i,j;

    for(i=0,j=strlen(s)-1; i<j; i++,j--)
    {
       c=s[i]; s[i]=s[j]; s[j]=c;
    }
}
```

- Višestruke inicijalizacije i višestruka povećanja (smanjenja) brojača u `for` petlji, odvajaju se zarezom.

Slide 54