Osnovi programiranja

Uvod u predmet

Osnovi programiranja

- Nastavnik: doc. dr Dejan Nikolić
  dejansnikolic@gmail.com

Cilj predmeta

- Sticanje znanja o algoritamskom rešavanju problema
- Sticanje osnovnih teorijskih znanja i praktičnih znanja o programskom jeziku C
- Osposobljavanje za samostalno rešavanje programskih problema kroz praktičan rad na računaru
- Osposoboljavanje za pisanje programa na jeziku C

Ishod predmeta

- Sposobnost snalaženja i rada u različitim razvojnim okruženjima i grupni rad pri rešavanju programskih problema i projekata iz različitih oblasti računarstva.
- Posedovanje teorijskih i praktičnih znanja o programiranju u jeziku C i samostalan i grupni rad pri rešavanju programskih problema i projekata.

Raspored časova – 1. semestar 2021/2022.

Predavanja

- Termin: subota, 10:00-12:00
- Predavač: doc. dr Dejan Nikolić
- E-mail: dejansnikolic@gmail.com
- Konsultacije:

Vežbe

- Termin: subota, 12:00 –14:00

Praktična nastava

- Kroz praktičnu nastavu studenti će se bliže upoznati sa karakteristikama i korišćenjem programskog jezika C.
- Kroz samostalne zadatke dublje se prodire u određene konstrukcije i primene programskog jezika C razvijajući ujedno sposobnosti za rešavanje programskih problema i projekata iz različitih oblasti računarstva.

Ocenjivanje

- Aktivnost u toku predavanja: 10 poena
- Kolokvijum #1: 15 poena
- Kolokvijum #2: 15 poena
- Seminarski radovi: 20 poena
- Završni ispit: 40 poena
- Ukupno: 100 poena

Aktivnosti u toku predavanja – 10 poena

Uključuje:

- prisustvo na predavanjima – do 5 poena
- aktivno učešće na predavanjima
- prezentacija tema na predavanjima koje nisu uključene u teme iz domena predmeta

Kolokvijum 1 – 15 poena

- Radi se u 6. nedelji nastave
- Sastoji se od 30 teorijskih pitanja sa opcionim odgovorima iz gradiva od 1. do 5. nedelje
- Svako pitanje nosi po 0,5 poena
- Vreme izrade kolokvijuma je 30 minuta

Kolokvijum 2 – 15 poena

- Radi se u 12. nedelji nastave
- Sastoji se od 30 teorijskih pitanja sa opcionim odgovorima iz gradiva od 7. do 11. nedelje
- Svako pitanje nosi po 0,5 poena
- Vreme izrade kolokvijuma je 30 minuta

Teme seminarskih radova

Mogu se raditi do 2 seminarska rada (svaki nosi po 10 bodova).

Oblasti

- Algoritmi
- Rešavanje problema uz pomoć računara
- Razvoj programskog jezika C
- Funkcije u jeziku C
- Poređenje programskih jezika
- Razvoj aplikacija u jeziku C
- Dinamičke strukture podataka
- .....

Završni ispit – 40 poena

- Programiranje na računaru
- Sastoji se od 2 do 4 zadatka
- Svaki zadatak je konkretan problem za čije rešavanje treba napisati program
- Da bi se dobio maksimalan broj bodova potrebno je da program „proradi“ na računaru

Popravni kolokvijum

- Radi se u 15. nedelji nastave
- Ukoliko student nije zadovoljan sa brojem bodova koje je ostvario na kolokvijumu, može ga raditi ponovo i tada se računa samo rezultat sa popravnog kolovijuma
- Može se popravljati samo prvi kolokvijum, samo drugi ili oba
- Na popravni kolokvijum mogu izaći i studenti koji nisu radili redovne kolokvijume

Formiranje konačne ocene

broj poena ≤ 50	5
51 ≥ broj poena ≤ 60	6
61 ≥ broj poena ≤ 70	7
71 ≥ broj poena ≤ 80	8
81≥ broj poena ≤ 90	9
broj poena ≥ 91	10

Preduslov za uspeh na predmetu OP

- Stalno raditi, tokom celog semestra.
- Redovno pratiti nastavu, samostalno praktično vežbati na računaru.
- Postavljati pitanja o svemu što nije jasno i to odmah, ne čekati da se nejasnoće gomilaju
- Ići na konsultacije.
- Ako na početku sve zvuči poznato, ne znači da će tako ostati do kraja.

Plan predmeta

1. Uvod u predmet. Uvod u jezik C. Pojam algoritma
2. Ulaz i izlaz. Tipovi podataka. Promenljive.
3. Operatori. Naredbe grananja.
4. Naredbe ponavljanja (petlje, ciklusi).
5. Pregled pređenog gradiva i priprema za Kolokvijum #1.
6. **Kolokvijum #1.**
7. Funkcije. Rekurzija.
8. Nizovi i matrice. Pokazivači.
9. Stringovi. Strukture. Fajlovi.
10. Dinamičke strukture podataka.
11. Pregled pređenog gradiva i pripreme za Kolokvijum #2
12. **Kolokvijum #2.**
13. Standardna biblioteka.
14. Seminarski radovi. Priprema za ispit.
15. **Popravni kolokvijum.**

Osnovi programiranja

Uvod u jezik C

Programski jezici

- Postoji mnogo programskih jezika i mnogo različitih mišljenja o tome šta je „dobar programski jezik“.
  - fokus nekih jezika je brzina izvršavanja
  - fokus nekih jezika na jednostavnosti pisanja programskog kôda.
  - fokus nekih jezika je samo savršeno izvršavanje jednog jedinog zadatka

Niži programski jezici

Mašinski jezik

- Mašinski jezik je najniži nivo programskog jezika
  - jedini jezik koji računar poznaje, program napisan u tom jeziku može se bez dodatnog prevođenja izvršiti na računaru
  - program u mašinskom jeziku za jednu vrstu računara nije primenljiv na drugoj vrsti računara
  - programer, koji piše program u mašinskom jeziku, mora dobro poznavati arhitekturu jedinice za obradu računara

Niži programski jezici (2)

Asembler

- Programski jezik niskog nivoa koji mašinski jezik specifične procesorske arhitekture predstavlja u ljudima čitljivom obliku.
- Simbolički kôdovi (mnemonici) zamenjuju binarne kôdove naredbe, kao što su npr. MOV (move), STO (store).
  - ako je asemblerski program napisan za jedan računar nije primenljiv na drugom računaru

Viši programski jezici

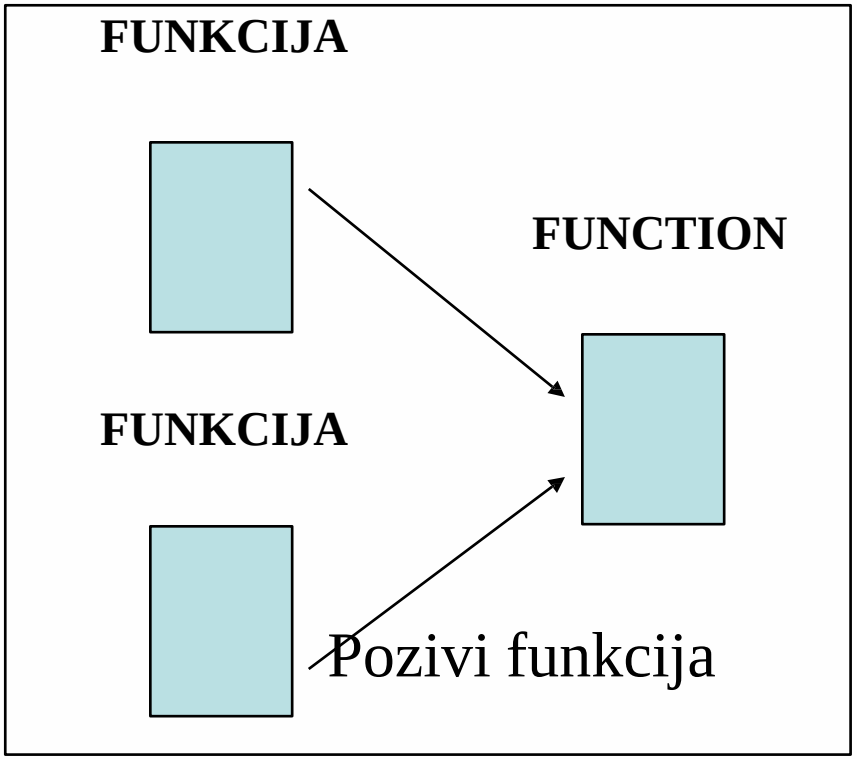
- Jezici čije se naredbe ne mogu direktno prevesti u binarne naredbe mašinskog jezika (npr. C, C++, Java).
- Više su okrenuti korisniku i problemu, a manje računaru
  - programer najčešće ne mora znati ništa o arhitekturi računara, čime je pisanje programa omogućeno
  - isti programi mogu se izvršavati na različitim vrstama računara

Tipovi programiranja

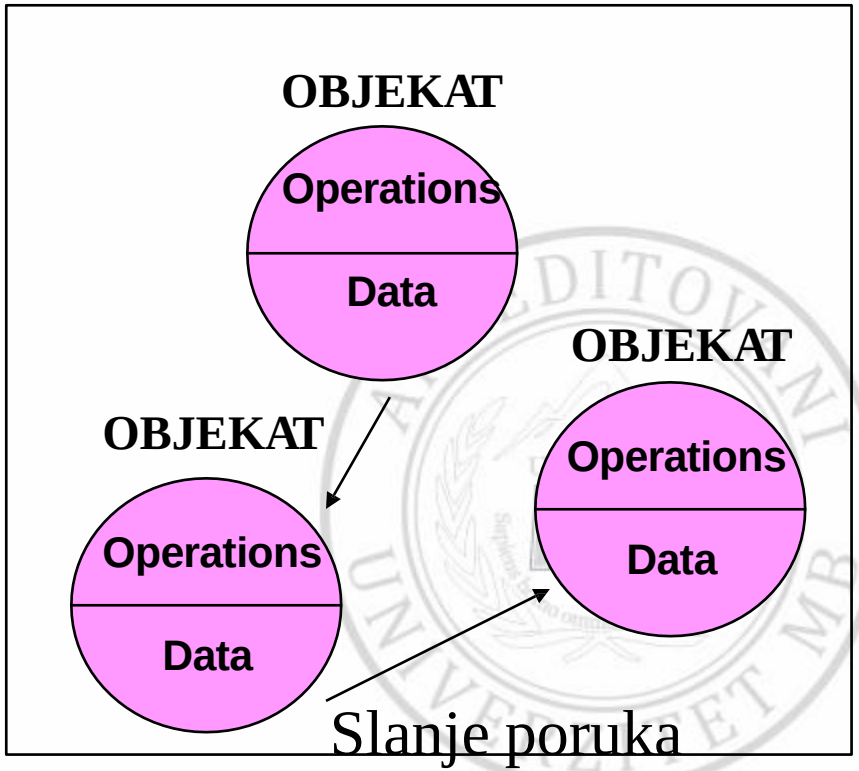
- Struktuirano programiranje (engl. structured programming)
- Objektno-orijentisano programiranje (Object-Oriented Programming - OOP)

Dva modela programiranja

Strukturni (Proceduralni) PROGRAM



Objektno-orijentisani PROGRAM



Struktuirano programiranje

- U početnoj fazi računarstva, izvršavanje operacija se odvijalo pomoću jednostavnih konstrukcija:
  - ako je uslov ispunjen, ići na zadatu lokaciju (koristiti GOTO komandu)
- Dovoljno je koristiti desetak takvih komandi da bi praćenje programa postalo zamorno.
- Umesto nepotrebnih GOTO naredbi, potrebno je izvršiti struktuiranje i programa i podataka koji se obrađuju.

Struktuirano programiranje (2)

Cilj

- Poboljšanje jasnoće, kvaliteta i vremena razvoja računarskih programa
  - intenzivno korišćenje potprograma, blok struktura, for i while petlje i slično.
  - izbegavanje korišćenja jednostavnih testova i programskih skokova, npr. izjava goto koja može da dovede do špageti koda (engl. spaghetti code) kojeg je teško i pratiti i održavati.

Struktuirano programiranje (3)

- Glavno pravilo struktuiranog programiranja je „podeli i vladaj“.
  - računarski program se može zamisliti kao skup zadataka (engl. task).
  - svaki složeni zadatak se može podeliti na skup manjih zadataka sve do trenutka kada delovi postanu dovoljno mali i sami sebi dovoljni da bi se lako mogli razumeti.

Struktuirano programiranje (4)

- Program je podeljen u manje celine koje se naknadno ugrađuju u glavni program.
  - ove celine je lakše napisati i proveriti nego kada su nedeljivi deo glavnog programa
  - skup programskih metoda koje vode ka logičkoj organizaciji i čitljivosti programa
- Logička organizacija programa olakšava pisanje, održavanje i ispravke programa.

Struktuirano programiranje (5)

- Strukturno orijentisani jezici (npr. Pascal, dBASE, C) prisiljavaju programera na strukturno programiranje, za razliku od nestrukturiranih (npr. BASIC, FORTRAN, COBOL) koji prepuštaju kreiranje programa u potpunosti programeru.

Špageti kôd (engl. spaghetti code)

- Špageti kôd je pogrdna reč za izvorni kôd koji ima složenu i zamršenu strukturu kontrole, naročito koristeći mnoge GOTO izjave, izuzetke ili druge " nestruktuirane" konstrukcije grananja.
- Ime dolazi od činjenice da program izgleda kao posuda sa špagetima.

Strukturno programiranje: Primer

Izračunavanje zarada zaposlenih u nekoj kompaniji

- Računanje zarade može biti jako složen zadatak koji se može razložiti na manje podzadatke:
  1. Koliko ima zaposlenih u kompaniji?
  2. Izračunati šta svaki zaposleni treba da dobije.
  3. Izračunati ukupan iznos za sve zarade.
  4. Podeliti ukupan iznos za sve zarade sa ukupnim brojem zaposlenih.

Biranje okruženja

- Računanje ukupnog iznosa zarada se može podeliti:
  1. Dobiti zapis (engl. record) za svakog zaposlenog.
  2. Pristupiti pojedinačnim zaradama.
  3. Dodati pojedinačnu zaradu trenutnoj ukupnoj sumi.
  4. Dobiti zapis za sledećeg zaposlenog

Karakteristike C jezika

- **C** je programski jezik opšte namene
- **C** je jezik koji kompajlira (prevodi) izvorni kôd (engl. compiled language).
- **C** je statički tipiziran jezik (engl. statically-typed language).
  - ako se želi manipulisati podacima u C jeziku, mora se unapred definisati tip podataka (npr. integer ili real)
  - to se ne može nigde promeniti u programu
- **C** je jezik visokog nivoa

Istorijat jezika C

- Godine 1972. Ken Thompson i Dennis Ritchie su kreirali jezik C.
- Jezik C je razvijen za potrebe razvoja operativnog sistema UNIX na platformi PDP7 (mini računar) u Belovim laboratorijama.
- C postao popularan jezik, jer je bio jednostavan:
  - izvršavao se prilično brzo
  - izvršavao se na svakom računaru koji je imao C prevodilac

Istorija jezika C

- Danas je C široko prihvaćen a na njegovoj sintaksi su zasnovani mnogi programski jezici: C++, C#, Java, PHP, Java Script, ...
- C je programski jezik visokog nivoa ali posjeduje i operatore bliske programskim jezicima niskog nivoa.

The ANSI Standard

- Komitet Accredited Standards Committee, koji radi po procedurama American National Standards Institute-ANSI, je kreirao međunarodni standard za C jezik.
  - poznat pod nazivom ISO (International Organization for Standardization) Standard ili NCITS (National Committee for Information Technology Standards) Standard i X3 (staro ime za NCITS) Standard i kao ANSI/ISO standard

Standardizacija jezika C

- ISO/ANSI Standard: započet 1983
- Standardizovane verzije C-a:
  - 1989. godine: ANSI-C ili C89
  - 1990. godine: C90
  - 1999. godine: C99
  - 2011. godine: C11

Kreiranje izvršne datoteke pomoću prevodioca

- C izvorni kôd (engl. source code) ima ekstenziju **.c** ili **.cpp**.
- Compiler (prevodilac) prevodi izvorni kod u mašinski jezik i smešta ga u datoteku istog imena ali s ekstenzijom **.o** (objektni kôd)
- Kôd **.o** još uvek nije izvršni, da se to desi treba pokrenuti linker koji prevodi **.o** datoteku u izvršnu datoteku **.exe**

Prvi C program

```
/* Prvi program prvi.c */
#include <stdio.h>
#include <stdlib.h> /* preprocesorske naredbe */
int main ()   /* glavna funkcija */
{
    printf („Ovo je prvi C program!\n");
    return 0;
}
```

Prepocesorske naredbe

- Jezik C ima metod da se:
  - izvorni tekstualni program, koji je napisao programer, najpre procesira posebnim programom za obradu programskog teksta
  - zatim se stvarno prevede.
- Ovaj program se zove **preprocesor** (engl. preprocessor) ili **makroprocesor**.

Preprocesorske naredbe (2)

- Naredbe koje se izvršavaju pre početka prevođenja izvornog kôda.
  - zavisno od preprocesorskih naredbi, preprocesor menja i dopunjuje izvorni kôd.
- Opšti oblik preprocesorske naredbe

#naredba parametri

  - preprocesorske naredbe započinju znakom #. Za razliku od ostalih naredbi, ne završavaju znakom tačka-zarez (;).
  - postoji više preprocesorskih naredbi, a jedna od najčešćih je naredba *include.*

C preprocesor

- C preprocesor modifikuje izvorni kôd (engl. source code) pre predaje tog koda prevodiocu.
- Najčešće, preprocesor uključuje druge datoteke direktno u program ili definiše konstante.
- Takođe će C preprocesor „ugušiti“ ulaz koji ne poštuje C leksička pravila.

Direktive

- Počinju znakom # i završavaju na kraju reda.
- Postoje 3 kategorije preprocesorskih direktiva:
  - direktive za definisanje simboličkih konstanti i makroa
  - direktive za uključivanje biblioteka
  - direktive za uslovno prevođenje programa

Direktive za definisanje simboličkih konstanti i makroa - #define

- Simbolička konstanta je identifikator čija se vrednost zadaje u fazi preprocesiranja
- Makroi su simboličke konstante sa mogućnošću zadavanja vrednosti prethodno definisanim parametrima u okviru konstante

Definisanje simboličkih konstanti

```
#define Pi  3.14159265
#define e   2.71828183
#include <stdio.h>
main ()
{
   float R1=1, R2=e, O1, O2;   /*Poluprečnik i obim kruga*/
   O1 = 2*R1*Pi;
   O2 = 2*R2*Pi;
}
```

Definisanje makroa

```
#define PRIKAZI(a)          printf („Rezultat je %d. \n“, a);
#include <stdio.h>
main ()
{
    int x=3, y=1;
    PRIKAZI(x)
    PRIKAZI(y)
}
```

Direktive za uključivanje biblioteka u program- #include

- Biblioteke su posebni fajlovi na disku koje imaju svoje ime i ekstenziju **.h**
- Nazivaju se još i *header* fajlovi (zaglavlja)
- Svrha postojanja biblioteka je da se jednom napisan programski kod koristi u više programa
- Postoje standardne i korisnički definisane biblioteke

Uključivanje biblioteka

#include <stdio.h>
Biblioteka za standardni ulaz i izlaz

#include <math.h>
Biblioteka sa dodatnim matematičkim funkcijama

#include <moje_konstante.h>
Korisnička biblioteka u kojoj su definisane simboličke konstante

Direktive za uslovno prevođenje programa

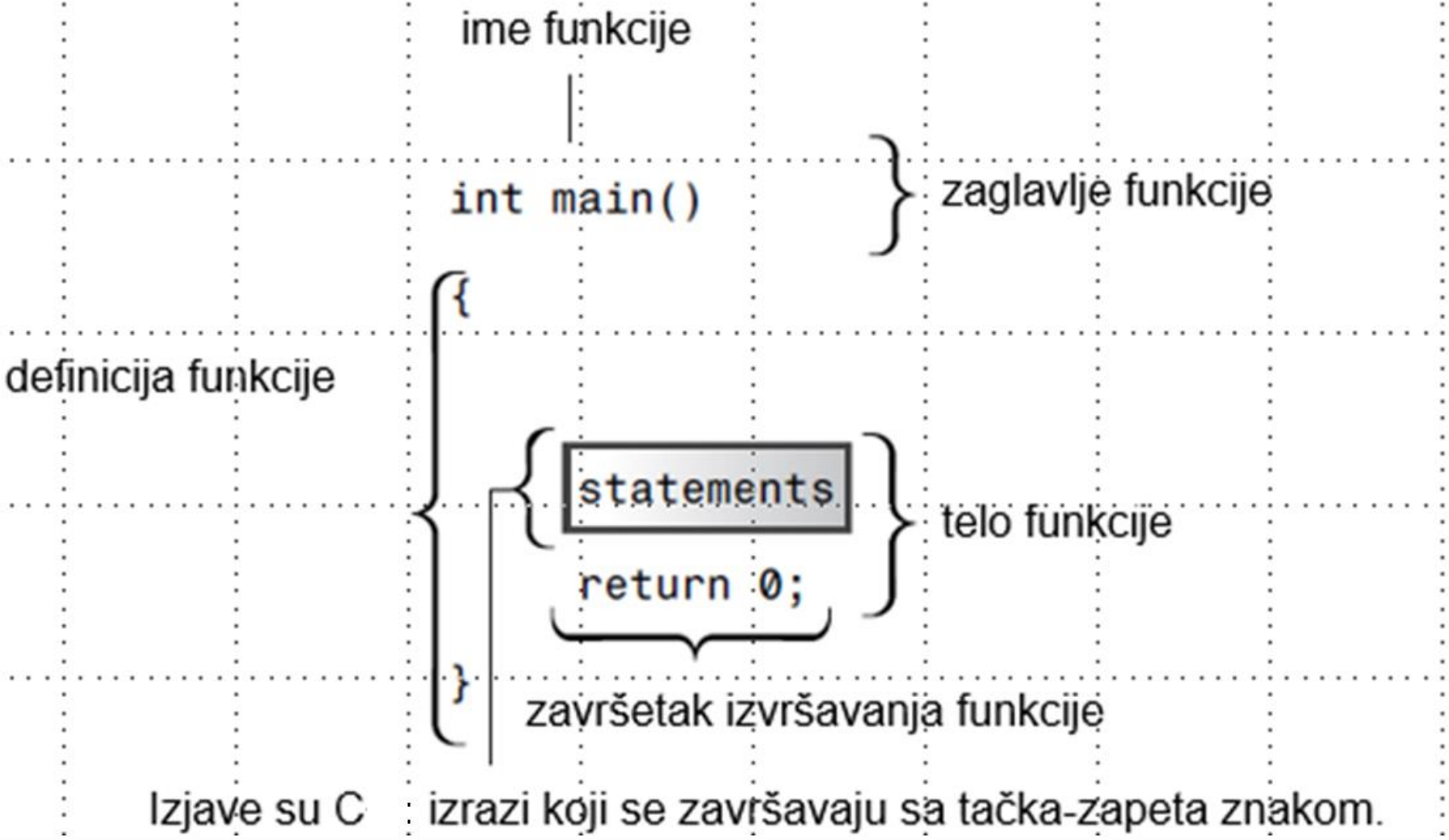
- Programer može da definiše delove programa koji će u zavisnosti od okolnosti biti uključeni u prevođenje programa ili će u potpunosti biti isključeni iz procesa prevođenja
- **#ifdef**
- **#endif**
- **#ifndef**
- **#else**
- **#elif**
- **#if**

Funkcija main ()

int main() {...}

- Definiše kôd koji treba da se izvrši kada se startuje program.
- Vitičaste zagrade {...} grupišu skup komandi u jedan blok.

Funkcija main () (2)



Prototip za funciju main ()

```
#include <stdio.h>
int main(); /* za većinu prevodilaca ova linija
               kôda nije potrebna*/
int main()
{
   printf (”Hello World!\n”);
   return 0;
}
```

Kontrolne sekvence (engl. escape sequences)

Printf („Ovo je prvi C program! **\n**");

- Primer: „\n“ označava karakter nove linije (engl. newline character.

Escape Sequence	Represented Character
\a	System bell (beep sound)
\b	Backspace
\f	Formfeed (page break)
\n	Newline (line break)
\r	“Carriage return” (returns cursor to start of line)
\t	Tab
\\	Backslash
\’	Single quote character
\"	Double quote character
some integer x	The character represented by x

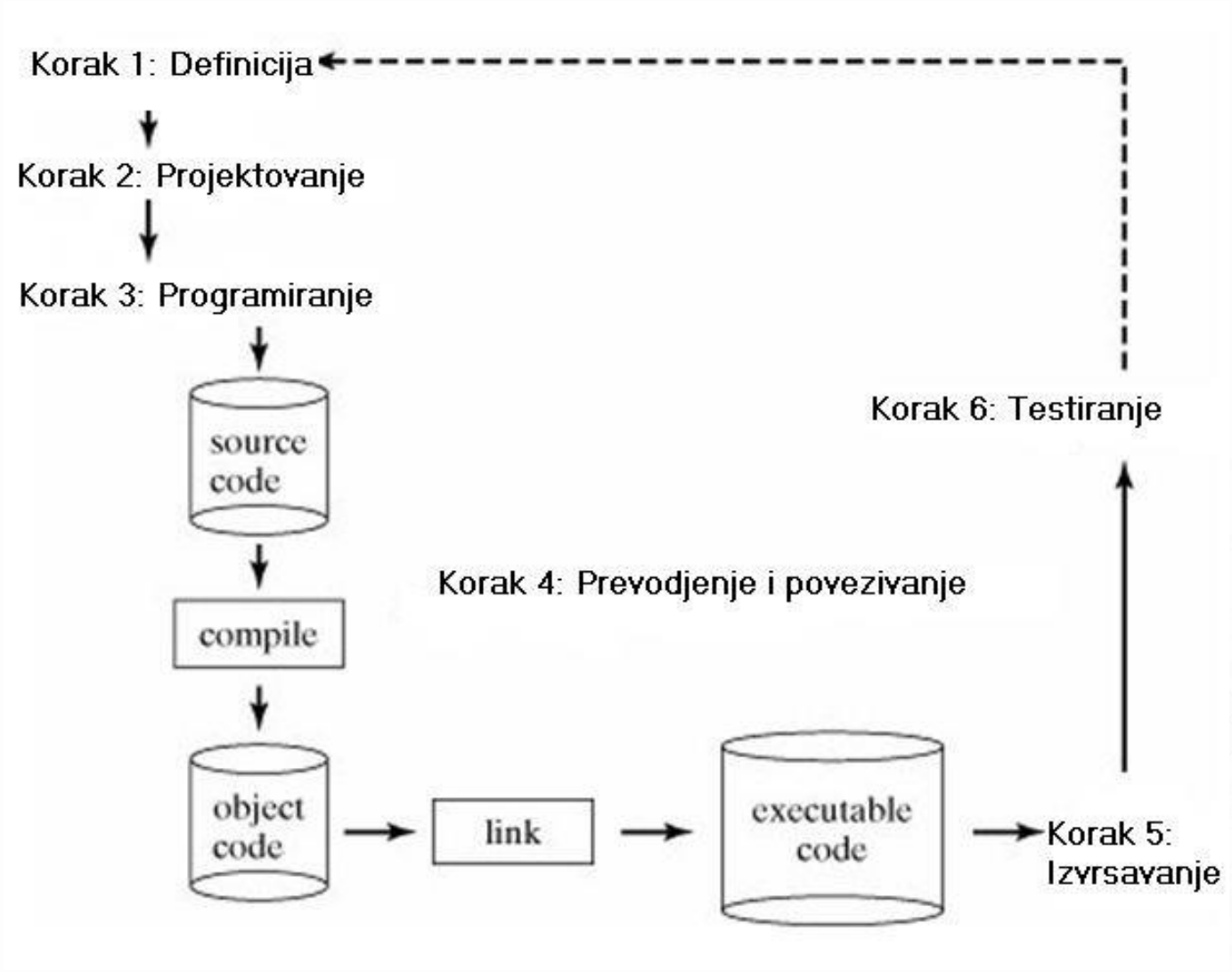
return 0

- Ova komanda kaže operativnom sistemu da se program uspešno završio – to je zadnja linija u bloku main () funkcije.

Faze C programiranja u C programskom okruženju

- **Uređivanje** (engl. edit) - u tekst editoru, snimanje datoteka
- **Preprocesiranje** - zamena teksta, ubacivanje drugih datoteka iz biblioteka u program i da se izvrši prevođenje
- **Prevođenje** (engl. compile) - u objektni kôd (engl. object code)
- **Povezivanje** (engl. link) - povezivanje objektnog kôda sa kôdom funkcija iz standardnih biblioteka koje se pozivaju u programu
- **Build** procesa - generiše se izvršni format (engl. executable image)
- **Load** – program se postavlja u memoriju
- **Execute** – izvršava se jedna po jedna instrukcija

Ciklus razvoja aplikacije



Osnovi programiranja

Pojam algoritma

Rešavanje problema na računaru

Apstrakcija problema (rešavanje složenog problema)

- Podela problema na manje celine koje se mogu nezavisno rešavati
- Zanemarivanje mnogobrojnih nevažnih detalja koji ne utiču na celokupno rešenje
- Smanjuje se složenost razvoja i održavanja velikih softverskih sistema

- Apstrakcija podataka
- Apstrakcija procedura

Rešavanje problema na računaru

Apstrakcija podataka

- Razdvajanje logičke slike podataka koji se obrađuju od njihove fizičke realizacije u računaru
- Generalizacija primitivnih tipova podataka
- Strukture podataka

Apstrakcija procedura

- Razdvajanje funkcionalnosti procedura od njihove programske realizacije u računaru
- Algoritmi

Algoritam

- Precizan postupak kojim se rešava dati problem
- Svi koraci postupka se mogu mehanički izvršiti na računaru
- Postupak je razumljiv za ljude, a ne za računare
- Algoritam ≠ program

- Tačno određen i uređen skup koraka koji vodi do rešenja nekog problema

Algoritamski problemi

- Primer: problem sortiranja
  - Dat je niz brojeva u proizvoljnom redosledu, treba ga preurediti u rastućem redosledu

Algoritamski problemi

- Primer: problem sortiranja
  - Dat je niz brojeva u proizvoljnom redosledu, treba ga preurediti u rastućem redosledu
  - **Ulaz:** niz a od n brojeva $a_1, a_2, \ldots, a_n$

    Izlaz: niz istih brojeva permutovanih u niz $a_{i1}, a_{i2}, a_{i3}, \ldots, a_{in}$ tako da je $a_{i1} \leq a_{i2} \leq a_{i3} \leq \ldots \leq a_{in}$

Algoritamski problemi

- Primer: problem sortiranja
  - Dat je niz brojeva u proizvoljnom redosledu, treba ga preurediti u rastućem redosledu
  - **Ulaz:** niz *a* od *n* brojeva $a_1, a_2, \ldots, a_n$

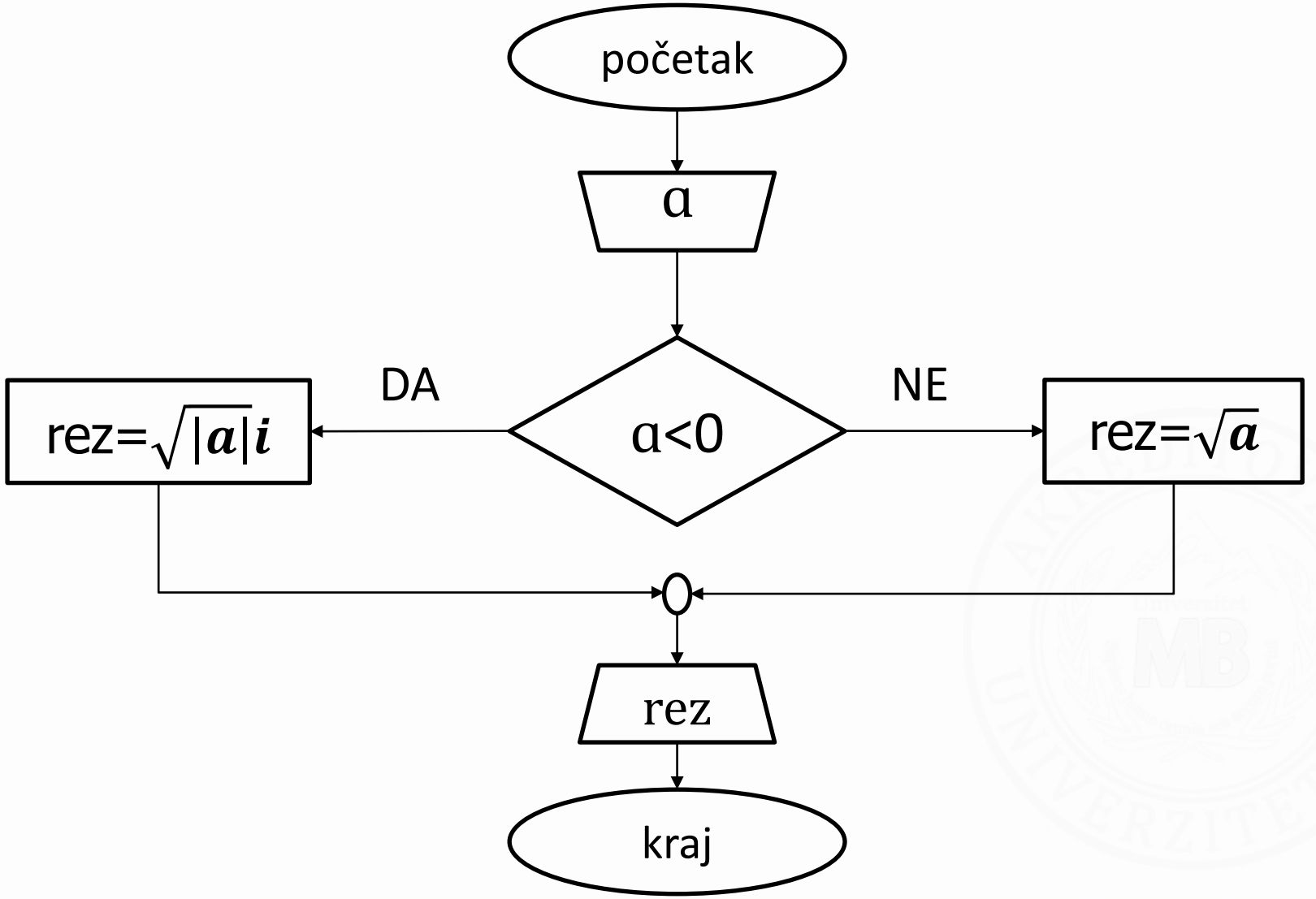
    Izlaz: niz istih brojeva permutovanih u niz $a_{i1}, a_{i2}, a_{i3}, \ldots, a_{in}$ tako da je $a_{i1} \leq a_{i2} \leq a_{i3} \leq \ldots \leq a_{in}$
  - Instanca problema sortiranja: 23, 17, 8, 15, 20, 15
    Rešenje ove instance: 8, 15, 15, 17, 20, 23

$\sqrt{a}$ – algoritam opisan rečima

Da li je $\alpha < 0$?

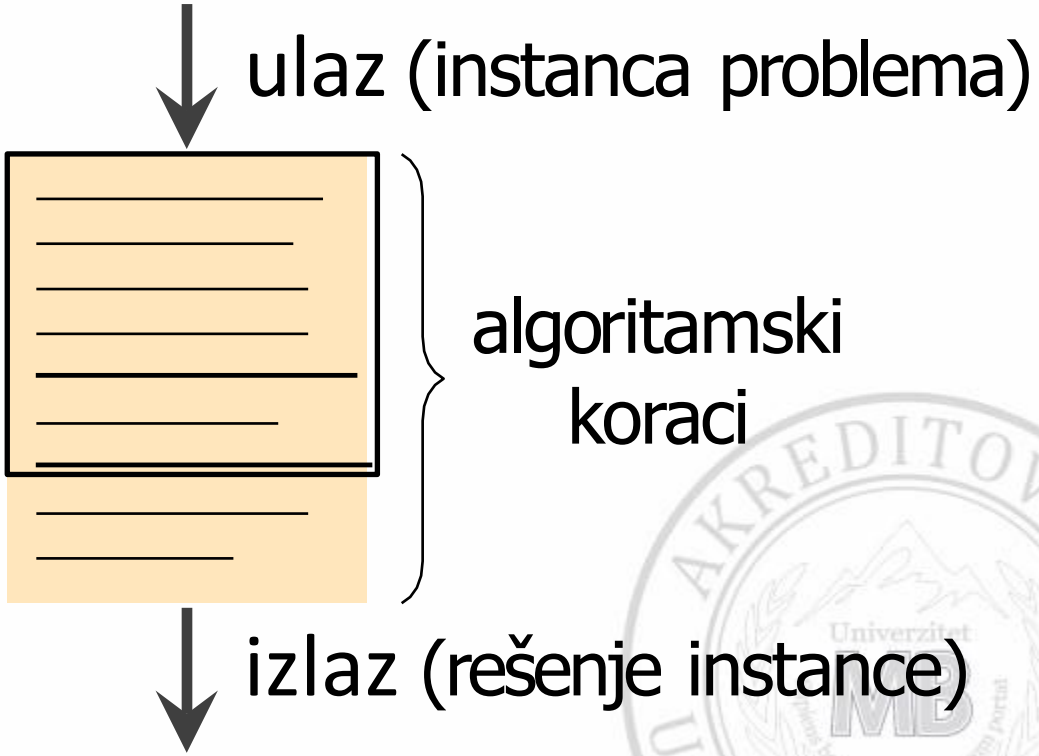
- Ako nije: rez=$\sqrt{a}$
- Ako jeste: rez=$\sqrt{|a|}i$ gdje je i - imaginarna jedinica

$\sqrt{a}$ – algoritam opisan blok dijagramom



Algoritmi

Algoritam



Algoritmi

Odlike svakog algoritma:

- Mora biti konačno mnogo koraka koji se mogu izvršiti na računaru
- Mora biti nedvosmisleno određen svaki sledeći korak za izvršavanje
- Algoritam mora biti ispravan postupak

Algoritmi

- Dizajn algoritma: pisanje niza naredbi koje sačinjavaju algoritam za rešenje datog problema
- Analiza algoritma: određivanje koliko algoritam zauzima resurse računara – Koji su to resursi?

Algoritmi

- Dizajn algoritma: pisanje niza naredbi koje sačinjavaju algoritam za rešenje datog problema
- Analiza algoritma: određivanje koliko algoritam zauzima resurse računara – Koji su to resursi? **Vreme i memorija računara**

Zapis algoritama

- Komplikovane ideje algoritma treba izraziti na što jednostavniji način
- Notacija primerena za ljude, a ne za „glupe" mašine
- Naglasak na logici rešenja, a ne na programskoj preciznosti
- Uglavnom se koriste: prirodan jezik, pseudojezik i pravi programski jezik
- Pseudo jezik: mešavina prirodnog i programskih jezika (Java, Pascal, C, Python, ...)

Zapis algoritama na pseudojeziku

- Pojam algoritma nije isto što i konkretniji pojam računarskog programa
- Računarski program se izvršava na pravom računaru i zato se piše na nekom programskom jeziku poštujući njegova striktna pravila
- Algoritmi se mogu zamisliti da se izvršavaju na idealizovanom računaru sa neograničenom memorijom
- Algoritmi su u stvari matematički objekti koji se mogu analizirati da bi se prepoznali suštinski delovi složenih problema

Primer - Zamena vrednosti dve promenljive

```
/* Ulaz: promenljive x i y sa svojim  vrednostima*/
/* Izlaz: promenljive x i y sa zamenjenim  vrednostima*/
algorithm swap(x, y)

   z = x;
   x = y;
   y = z;

   return x, y;
```

Dizajn algoritama

Dobri algoritmi

- Korektnost (tačnost) i efikasnost (vreme i memorija)
- Jednostavnost i jasnoća
- Algoritamske paradigme: rekurzivni algoritmi, pohlepni algoritmi, dinamičko programiranje, randomizirani algoritmi, …

Analiza algoritama

- Ocena efikasnosti algoritma na osnovu kriterijuma iskorišćenosti računarskih resursa
- Dragoceni računarski resursi
  - Vreme
  - Memorija
  - Mrežni saobraćaj

Zaključak

- Niži programski jezici – mašinski jezik, asembler
- Viši programski jezici – C, C++, Java
- Tipovi programiranja – struktuirano (C) i objektno orjentisano (C++)
- Struktuirano programiranje - program je podeljen u manje celine koje se naknadno ugrađuju u glavni program.

Zaključak 2

- C je programski jezik visokog nivoa, opšte namene, tipiziran, standardizovan
- Izvorni kod → compiler → objektni kod → linker → izvršni kod
- Preprocesorske direktive - #define, #include
- Funkcija main ()

Zaključak 3

- Algoritam - precizan postupak kojim se rešava dati problem
- Dizajn algoritma - pisanje niza naredbi koje sačinjavaju algoritam za rešenje datog problema
- Zapis algoritma - komplikovane ideje algoritma treba izraziti na što jednostavniji način
- Analiza algoritma - određivanje koliko algoritam zauzima resurse računara (vreme, memoriju, ...)

Kraj prezentacije

HVALA NA PAŽNJI!