Algoritmi i strukture podataka

Dr. Goran Aritonović

goran.aritonovic@bbs.edu.rs

kabinet 211

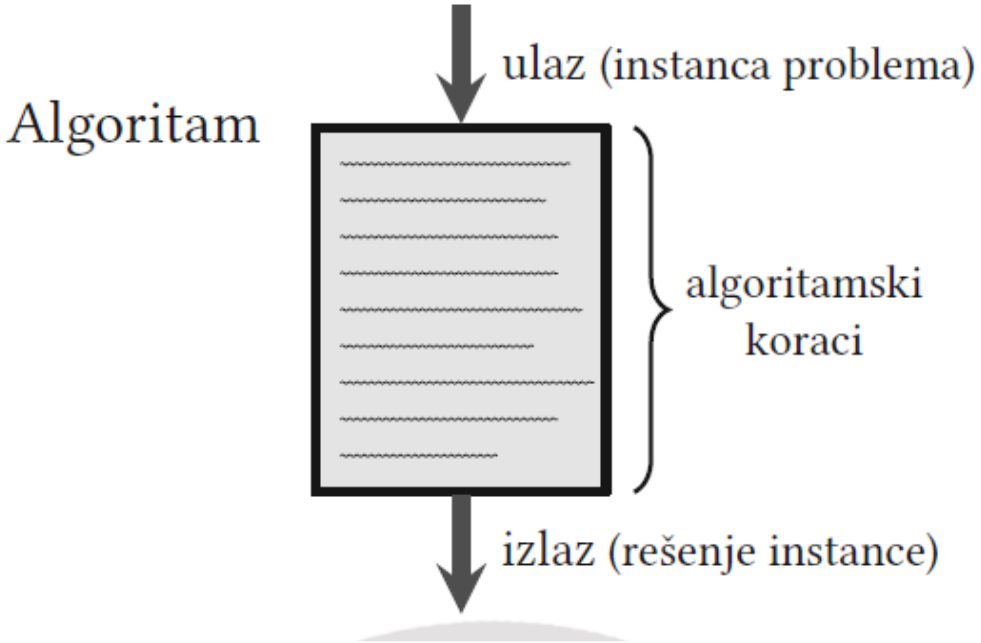
Pojam algoritama i struktura podataka

- Niklaus Wirth : Algorithms + Data Structures = Programs
  - Tvorac programskog jezika Pascal
  - 1975 napisao kljigu algoritmi + struktura podataka = programi
- Struktura podataka – opis načina organizacije podataka
- Algoritam – opis načina obrade podataka

Šta je algoritam?

- Algoritam je skup pravila za rešavanje nekog problema
- Algoritam se sastoji iz konačnog skupa algoritamskih koraka
- Mora biti nedvosmisleno određen svaki sledeći korak za izvršavanje
- Moraju biti definisani početni objekti nad kojima se obavljaju operacije
- Ishod algoritma su završni objekti ili rezultati
- Za bilo koje vrednosti ulaznih podataka algoritam mora da završi rad nakon konačnog broja koraka
- Algoritmi se kreiraju nezavisno od jezika u kome će biti implementirani

Pojednostavljena slika algoritma



Algoritamski problemi

- Primer: problem sortiranja
- Dat je niz brojeva koji imaju proizvoljni redosled, ovaj niz treba preurediti u rastući redosled
- Ulaz: niz a od n brojeva **a1, a2, … , an**
- Izlaz: niz istih brojeva permutovanih u niz **ai1, ai2, ai3, …, ain** tako da je **ai1 ≤ ai2 ≤ ai3 ≤ … ≤ ain**

Instanca problema sortiranja: 23, 17, 8, 15, 20, 15

Rešenje ove instance: 8, 15, 15, 17, 20, 23

Dizajn i analiza algoritama

- Dizajn algoritma - pisanje niza naredbi koje sačinjavaju algoritam za rešavanje datog problema
- Analiza algoritama - određivanje koliko algoritam zauzima resurse računara (vreme, memoriju, …)
- Dizajn i analiza algoritama nisu nezavisni već se prožimaju

Zapis algoritama

- Komplikovane ideje algoritma treba izraziti na što jednostavniji način
- Treba koristiti notacija primerena za ljude,a ne za mašine
- Jedan od načina zapisivanja algoritma je dijagram toka - algoritamska šema
  - zahtevaju dosta prostora
- Uglavnom se za zapisivanje algoritama koriste: prirodan jezik, pseudojezik i pravi programski jezik
- Pseudo jezik: mešavina prirodnog i programskih jezika (Java, Pascal, C, …)

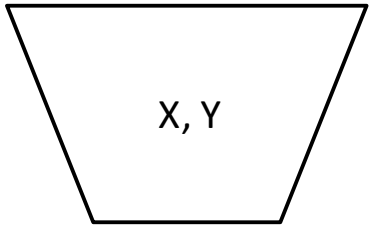
Algoritmi i programi

- Program je opis algoritma koji u nekom programskom jeziku jednoznačno određuje šta računar treba da odradi
- Programiranje – proces prevođenja opšteg rešenja problema u računarski čitljiv oblik

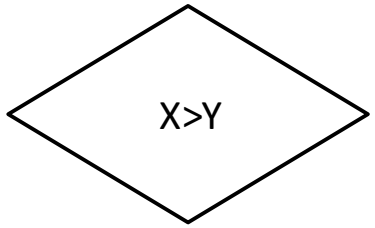
Grafički simboli u dijagramu toka -1

Definiše početak algoritma

Definiše kraj algoritma

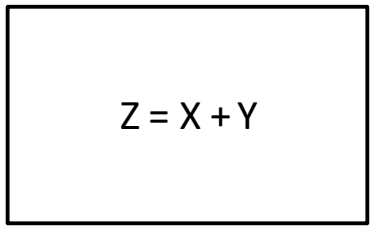


Ulazne veličine algoritma

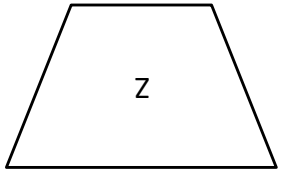


Uslovni algoritamski korak

Grafički simboli u dijagramu toka -2



Obrada podataka



Izlazne veličine algoritma

Vrste algoritamskih šema

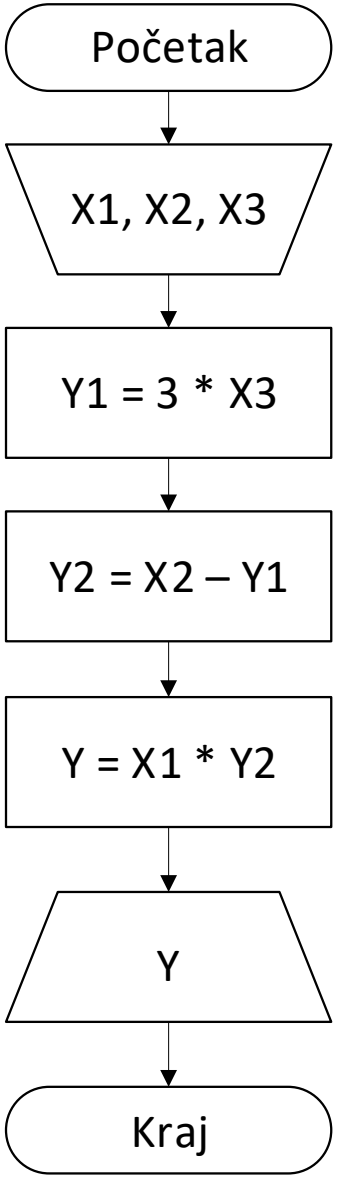
- Linijske algoritamske šeme
  - proste
  - razgranate
- Ciklične algoritamske šeme
- Složene algoritamske šeme

Proste linijske algoritamske šeme

- Svaki algoritamski korak izvršava tačno jednom u toku jednog izvršavanja algoritma
- Sastoji se od algoritamskih koraka ulaza, obrade i izlaza

Primer proste linijske šeme

Sastaviti algoritamsku šemu za izračunavanje vrednosti Y po sledećoj formuli:
Y = X1 * (X2 – 3*X3),
tako da u jednom algoritamskom koraku može biti samo jedna aritmetička operacija.



Realizacija algoritma u C# jeziku

```
static void Main(string[] args)
{
    // Y = X1 * (X2 – 3*X3),

    Console.WriteLine("Ucitaj X1");
    double x1 = double.Parse(Console.ReadLine());

    Console.WriteLine("Ucitaj X2");
    double x2 = double.Parse(Console.ReadLine());

    Console.WriteLine("Ucitaj X3");
    double x3 = double.Parse(Console.ReadLine());

    double y1 = 3 * x3;
    double y2 = x2 - y1;
    double y = x1 * y2;

    Console.WriteLine($"Rezultat je: {y}");
    Console.ReadLine();

}
```

Osnove C# jezika

Formatiranje C# koda

- Program je skup instrukcija – naredbi
- Naredba se završava oznakom ;
- Iako je moguće pisati više naredbi u istoj liniji, dobra je praksa da se u svakoj liniji piše samo po jedna naredba
- Prazan prostor u editoru koda se ignoriše od strane kompajlera
- Grupisanje naredbi (kreiranje bloka naredbi) vrši se korišćenjem vitičastih zagrada {…}

```
{
// pocetak bloka naredbi
naredba1;
naredba2;
// kraj bloka naredbi
}
```

Identifikatori

- Imena ili identifikatori se koriste za označavanje osnovnih objekta jezika: konstanti, promenljivih, funkcija i tipova podataka
- Ime može sadržati slovo, cifru i znak podvlačenja _
- Ime ne sme počinjati cifrom
- U svojstvu imena ne smeju se koristiti rezervisane reči jezika
- Velika i mala slova se razlikuju (x i X su dve različite promenljive)

Ugrađeni tipovi podataka

- Ugrađeni tipovi podataka su oni koje obezbeđuje C# i .NET framework
- Tipovi se koriste za deklarisanje promenljivih i konstanti
- Promenljive se moraju deklarisati pre nego što mogu da se koriste
- Promenljive čuvaju različite tipove podataka
- Moguće je definisati sopstvene tipove podataka

Tipovi podataka

C# tip	Veličina u bitovima	Sistemski tip
sbyte	8	System.SByte
short	16	System.Int16
int	32	System.Int32
long	64	System.Int64
byte	8	System.Byte
ushort	16	System.UInt16
uint	32	System.UInt32
ulong	64	System.UInt64
char	16	System.Char
bool	8	System.Boolean
float	32	System.Single
double	64	System.Double
decimal	128	System.Decimal
string	nije raspoloživo	System.String
object	nije raspoloživo	System.Object
dynamic	nije raspoloživo	System.Object

Celobrojni tipovi podataka

Type	Range	Size	.NET Framework type
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer	System.Int32

Type	Range	Size	.NET Framework type
short	-32,768 to 32,767	Signed 16-bit integer	System.Int16

Type	Range	Size	.NET Framework type
long	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer	System.Int64

Type	Range	Size	.NET Framework type
uint	0 to 4,294,967,295	Unsigned 32-bit integer	System.UInt32

Deklarisanje i inicijalizacija celobrojnih promenljivih

```
// deklarisanje celobrojnih promenljivih
int brojStudenata;
int brojGrupa;
```

```
// dodeljivanje vrednosti promenljivama
brojStudenata = 85;
brojGrupa = 5;
```

```
int brojStudenata = 85;
int brojGrupa = 5;
```

Realni tipovi

Type	Approximate range	Precision	.NET Framework type
float	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	7 digits	System.Single

Type	Approximate range	Precision	.NET Framework type
double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	15-16 digits	System.Double

Type	Approximate Range	Precision	.NET Framework type
decimal	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$	28-29 significant digits	System.Decimal

Primeri definisanja realnih promenljivih

```
double x = 3.14;  //3.14d 3.14D
float y = 2.51f;  //2.51F
decimal z = 4.23m;//4.23M
```

Broj sa decimalnom tačkom podrazumevano je tipa double

```
decimal stanjeNaRacunu = 3433.20; // greška !!!
```

```
decimal stanjeNaRacunu = 3433.20M;
```

Implicitna i eksplicitna konverzija

```
static void Main(string[] args)
{
    int a = 4;
    float b = a; // implicitna konverzija
    Console.WriteLine(b);
    Console.WriteLine("a={0}, b= {1}",a,b);
    Console.WriteLine($"a={a}, b= {b}");

    b = 4.5f;
    int c = (int)b; // eksplicitna konverzija, kastovanje
    Console.WriteLine($"b={b}, c={c}");

    Console.ReadLine();
}
```

Tip char

Type	Range	Size	.NET Framework type
char	U+0000 to U+ffff	Unicode 16-bit character	System.Char

```
static void Main(string[] args)
{
    char c1 = 'X';
    char c2 = '\x0058';
    char c3 = (char)88;
    char c4 = '\u0058';
    Console.WriteLine($"{c1}{c2}{c3}{c4}");
}
```

Escape karakteri

Escape sekvenca	Proizvedeni karakter	Unicode vrednost karaktera
\'	jednostruku znak navoda	0x0027
\ "	dvostruki znak navoda	0x0022
\\	Backslash (obrnuta kosa crta)	0x005C
\0	Null	0x0000
\a	zvuk zvona	0x0007
\b	Backspace	0x0008
\f	Form feed	0x000C
\n	nova linija	0x000A
\r	Carriage return	0x000D
\t	Horizontal tab	0x0009
\v	Vertical tab	0x000B

```
char jednostrukiNavodnik = '\'';
```

Deklaracija stringova

```
static void Main(string[] args)
{
    string s1 = "Pozdrav svima";
    Console.WriteLine(s1);

    string s2 = "Pozdrav\t\tsvima";
    Console.WriteLine(s2);

    string s3 = "Pozdrav\nsvima";
    Console.WriteLine(s3);

    Console.ReadLine();
}
```

Upotreba karaktera @ ispred stringa

```
static void Main(string[] args)
{
    string s1 = @"Pozdrav \ svima";
    Console.WriteLine(s1);

    string s2 = "Pozdrav \\ svima";
    Console.WriteLine(s2);

    string s3 = @"Pozdrav\t\tsvima";
    Console.WriteLine(s3);

    string s4 = @"Pozdrav\nsvima";
    Console.WriteLine(s4);

    Console.ReadLine();
}
```

Pitanje 1

Koji se od navedenih tipova podataka može koristiti isključivo za čuvanje celih brojeva:

a. float
b. double
c. int

Odgovor: c

Pitanje 2

Sistemski C# tip koji odgovara tipu podataka float je:

a. Double
b. Single
c. Decimal

Odgovor: b

Pitanje 3

Svaka naredba u C# završava se oznakom:

a. ;
b. :
c. \\

Odgovor: a

Pitanje 4

Ako želimo najveću preciznost u radu sa realnim brojevima koristićemo tip podataka:

a. float
b. double
c. decimal

Odgovor: c

Operatori

Osnovni operator dodeljivanja

- **Izraz** je sekvenca operatora i operanada
- Konstante ili promenljive koje učestvuju u izrazima nazivaju se **operandi**
- **Operator** je simbol koji precizira koju akciju treba izvršiti nad operandima
- Operacije nad jednim operandom nazivaju se unarne operacije, a operacije nad dva operanda nazivaju se binarne operacije
- Osnovni operator dodeljivanja je binarni i predstavlja se simbolom =
- Osnovni operator dodeljivanja (=) prouzrokuje da se vrednost operanda na desnoj strani dodeli operandu na levoj strani
- Operatori dodeljivanja imaju najniži prioritet

Primeri upotrebe osnovnog operatora dodeljivanja

```
x = 10;
```

Promenljivoj x se dodeljuje vrednost 10

```
y = x;
```

Promenljivoj y se dodeljuje vrednost promenljive x

```
x = x + 1;
```

Promenljivoj x se dodeljuje stara vrednost promenljive x uvećana za broj 1

Pregled operatora

- Aritmetički operatori
- Inkrementiranje i dekrementiranje
- Operatori poređenja
- Logički operatori

Aritmetički operatori

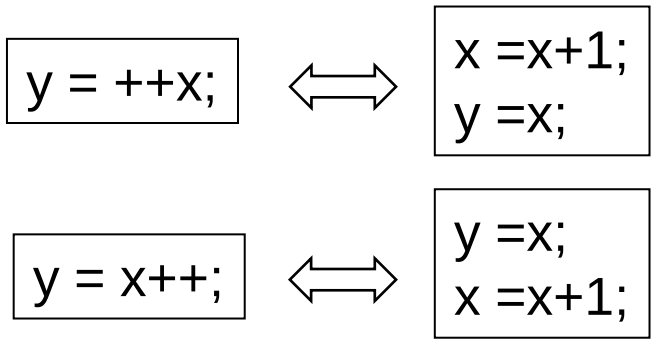
- Aritmetičke operacije su:
  - Množenje (*)
  - Deljenje (/)
  - Celobrojni ostatak (%)
  - Sabiranje i oduzimanje (+, -)
- Najveći prioritet ima unarni minus, zatim multiplikativne operacije i na kraju su aditivne operacije

Primer upotrebe aritmetičkih operatora

```
static void Main(string[] args)
{
    Console.WriteLine("Unesite ceo broj a");
    string sa = Console.ReadLine();
    int a = int.Parse(sa);

    Console.WriteLine("Unesite ceo broj b");
    string sb = Console.ReadLine();
    int b = int.Parse(sb);

    int suma = a + b;
    int razlika = a - b;

    Console.WriteLine($"{a} + {b} = {suma}, {a} - {b} = {razlika}");
    Console.ReadLine();

}
```

Inkrementiranje i dekrementiranje

- Operacija inkrementiranja (++)
- Operacija dekrementiranja (--)
- Obe operacije mogu imati prefiksni oblik tj. nalaze se ispred promenljive i sufiksni oblik tj. nalaze se iza promenljive
- Većeg su prioriteta od aritmetičkih operatora



Operatori složenog dodeljivanja

```
x += 2; // x = x+2;
x *= 2; // x = x*2;
x -= 2; // x = x-2;
x /= 2; // x = x/2;
x %= 2; // x = x%2;;
```

Operatori poređenja

- Operacije poređenja su:
  - Veće (>)
  - Veće ili jednako (>=)
  - Manje (<)
  - Manje ili jednako (<=)
  - Jednako (==)
  - Različito (!=)
- Rezultat izvršavanja operacije je logička vrednost true ako je uslov ispunjen, u suprotnom je false.
- Manjeg su prioriteta od aritmetičkih operatora

Primer upotrebe operatora poređenja

```
static void Main(string[] args)
    {
        Console.WriteLine("Unesi prvi realan broj");
        float f1 = float.Parse(Console.ReadLine());

        Console.WriteLine("Unesi drugi realan broj");
        float f2 = float.Parse(Console.ReadLine());

        bool a = f1 > f2;
        bool b = f1 < f2;
        bool c = f1 == f2;
        bool d = f1 != f2;

        Console.WriteLine($"{f1}>{f2}={a}\n" +
            $"{f1}<{f2}={b}\n" +
            $"{f1}=={f2}={c}\n" +
            $"{f1}!={f2}={d}\n");

        Console.ReadLine();
    }
}
```

Logičke operacije

- Negacija (!)
- Konjukcija – logičko I (&&)
- Disjunkcija- logičko ILI (||)
- Rezultat logičkih operacija je bool promenljiva true ili false
- Operacija negacije je unarna i daje true ako je operand false
- Operacija konjukcije je binarna i daje true ako oba operanda imaju vrednost true
- Operacija disjunkcije je binarna i daje true ako je bar jedan od operanada ima vrednost true
- Imaju niži prioritet od operacija poređenja
- Najveći prioritet ima operacija negacije, zatim konjukcija i na kraju disjunkcija

Primer upotrebe logičkih operacija

```
static void Main(string[] args)
{
    Console.WriteLine("unesi prvi realan broj");
    decimal d1 = decimal.Parse(Console.ReadLine());

    Console.WriteLine("unesi drugi realan broj");
    decimal d2 = decimal.Parse(Console.ReadLine());

    bool b1 = d1 > 5;
    bool b2 = d2 > 5;

    bool b3 = b1 && b2;
    bool b4 = b1 || b2;

    Console.WriteLine($"Oba uneta broja su veca od 5 : {b3}");
    Console.WriteLine($"Bar jedan od unetih brojeva je veci od 5 : {b4}");

    Console.ReadLine();
}
```

Pitanje 1

Promenljiva x ima vrednost 5. Koju će vrednost imati x nakon izvršavanja sledeće linije koda:
x +=2;

a. 3
b. 7
c. 10

Odgovor: b

Pitanje 2

Operator dodeljivanja se u C# programskom jeziku označava sa:

a. =
b. ==
c. +=

Odgovor: a

Pitanje 3

Rezultat izvršenja operacije poređenja dva operanda je promenljiva tipa:

a. bool
b. int
c. float

Odgovor: a

Pitanje 4

Sufiksno inkrementiranje ima veći prioritet od operatora dodeljivanja:

a. da
b. ne

Odgovor: b

Pitanje 5

Šta se dobija kao rezultat izvršavanja sledećeg koda:

```
static void Main(string[] args)
{
    int x = 5;
    int y = 3;
    int z = x / y;
    Console.WriteLine(z);
}
```

a. 1,666667
b. 1
c. 2

Odgovor: b