Predavanje 2 - Tipovi podataka

Tatjana Tomović

Institut za matematiku i informatiku
Prirodno-matematički fakultet
Univerzitet u Kragujevcu

Kragujevac, 2014.

Pregled predavanja

Tipovi podataka

- Postoje 14 osnovnih tipova podataka koji su nama od interesa.

Tipovi podataka

- Postoje 14 osnovnih tipova podataka koji su nama od interesa.
- To su: *logical*, *char*, *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64*, *uint64*, *single*, *double*, *cell*, *structure*, *function handle*.

Tipovi podataka

- Postoje 14 osnovnih tipova podataka koji su nama od interesa.
- To su: *logical*, *char*, *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64*, *uint64*, *single*, *double*, *cell*, *structure*, *function handle*.
- Numerički tip podataka čine celobrojni podaci (*int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64*, *uint64*) i tipovi podataka u pokretnom zarezu (*single*, *double*).

Tipovi podataka

- Postoje 14 osnovnih tipova podataka koji su nama od interesa.
- To su: *logical*, *char*, *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64*, *uint64*, *single*, *double*, *cell*, *structure*, *function handle*.
- Numerički tip podataka čine celobrojni podaci (*int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64*, *uint64*) i tipovi podataka u pokretnom zarezu (*single*, *double*).
- U Matlabu se tip podataka naziva klasom podataka (mi usvajamo konvenciju da koristimo ravnopravno reči klasa i tip podataka).

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.
- Označeni celobrojni tipovi uključuju (*int8*, *int16*, *int32*, *int64*).

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.
- Označeni celobrojni tipovi uključuju (*int8*, *int16*, *int32*, *int64*).
- Označeni celobrojni tip može da reprezentuje pozitivne i negativne cele brojeve.

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.
- Označeni celobrojni tipovi uključuju (*int8*, *int16*, *int32*, *int64*).
- Označeni celobrojni tip može da reprezentuje pozitivne i negativne cele brojeve.
- Neoznačeni celobrojni tipovi uključuju (*uint8*, *uint16*, *uint32*, *uint64*).

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.
- Označeni celobrojni tipovi uključuju (*int8*, *int16*, *int32*, *int64*).
- Označeni celobrojni tip može da reprezentuje pozitivne i negativne cele brojeve.
- Neoznačeni celobrojni tipovi uključuju (*uint8*, *uint16*, *uint32*, *uint64*).
- Neoznačeni tip podataka nije u stanju da reprezentuje negativne cele brojeve, ali je dobit veći opseg pozitivnih brojeva koje je moguće predstaviti u odnosu na označen tip.

Celobrojni tipovi

- Celobrojni tip podataka (*integer*) čine označeni i neoznačeni tipovi podataka.
- Označeni celobrojni tipovi uključuju (*int8*, *int16*, *int32*, *int64*).
- Označeni celobrojni tip može da reprezentuje pozitivne i negativne cele brojeve.
- Neoznačeni celobrojni tipovi uključuju (*uint8*, *uint16*, *uint32*, *uint64*).
- Neoznačeni tip podataka nije u stanju da reprezentuje negativne cele brojeve, ali je dobit veći opseg pozitivnih brojeva koje je moguće predstaviti u odnosu na označen tip.
- Brojevi u nazivu celobrojnih tipova podataka označavaju količinu bitova koja će biti upotrebljena za smeštanje u memoriju.

Celobrojni tipovi

Vrednosti maksimalnih i minimalnih brojeva koje se mogu predstaviti u pojedinim tipovima mogu se dobiti pozivanjem funkcija *intmax* i *intmin*, redom, sa imenom odgovarajućeg tipa kao argumentom.

Celobrojni tipovi

Vrednosti maksimalnih i minimalnih brojeva koje se mogu predstaviti u pojedinim tipovima mogu se dobiti pozivanjem funkcija *intmax* i *intmin*, redom, sa imenom odgovarajućeg tipa kao argumentom.

Ulaz:

```
>> intmax('int64')
>> intmin('int64')
```

Celobrojni tipovi

Ako želimo da nekoj promenljivoj pridružimo celobrojnu vrednost, moramo koristiti odgovarajuću funkciju za kreiranje tipa, jer se numerički literali po konvenciji interpretiraju kao podaci tipa *double*. Dakle, kreiranje celog broja dužine 32 bita izgleda ovako

Celobrojni tipovi

Ako želimo da nekoj promenljivoj pridružimo celobrojnu vrednost, moramo koristiti odgovarajuću funkciju za kreiranje tipa, jer se numerički literali po konvenciji interpretiraju kao podaci tipa *double*. Dakle, kreiranje celog broja dužine 32 bita izgleda ovako

Ulaz:

$$>> a = int32(32)$$

Celobrojni tipovi

Ako želimo da nekoj promenljivoj pridružimo celobrojnu vrednost, moramo koristiti odgovarajuću funkciju za kreiranje tipa, jer se numerički literali po konvenciji interpretiraju kao podaci tipa *double*. Dakle, kreiranje celog broja dužine 32 bita izgleda ovako

Ulaz:

$$>> a = int32(32)$$

Osnovne informacije o kreiranom objektu možemo dobiti upotrebom funkcije *whos*.

Celobrojni tipovi

Da bismo saznali tip matrice koja čuva smeštene podatke možemo koristiti funkciju *size*,

Celobrojni tipovi

Da bismo saznali tip matrice koja čuva smeštene podatke možemo koristiti funkciju *size*,da bismo saznali klasu podataka možemo koristiti funkciju *class*.

Celobrojni tipovi

Da bismo saznali tip matrice koja čuva smeštene podatke možemo koristiti funkciju *size*,da bismo saznali klasu podataka možemo koristiti funkciju *class*.Funkcija *isinteger* daje informaciju o tome da li je tip podataka celobrojni ili nije.

Celobrojni tipovi

Da bismo saznali tip matrice koja čuva smeštene podatke možemo koristiti funkciju *size*,da bismo saznali klasu podataka možemo koristiti funkciju *class*.Funkcija *isinteger* daje informaciju o tome da li je tip podataka celobrojni ili nije.

Ulaz:

$$>> size(a)$$
$$>> class(a)$$
$$>> isinteger(a)$$

Celobrojni tipovi

Pitanje prekoračenja opsega formata sa celobrojnim tipom podataka u Matlabu:

Celobrojni tipovi

Pitanje prekoračenja opsega formata sa celobrojnim tipom podataka u Matlabu: ako je *a* promenljiva čija je vrednost maksimalna moguća vrednost celobrojnog tipa, dodavanje promenljivoj *a* ne menja vrednost promenljive *a*. Slično, ako je vrednost promenljive *a* najmanja moguća vrednost nekog celobrojnog tipa, oduzimanje od promenljive *a* ne menja vrednost promenljive *a*.

Celobrojni tipovi

Pitanje prekoračenja opsega formata sa celobrojnim tipom podataka u Matlabu: ako je a promenljiva čija je vrednost maksimalna moguća vrednost celobrojnog tipa, dodavanje promenljivoj a ne menja vrednost promenljive a. Slično, ako je vrednost promenljive a najmanja moguća vrednost nekog celobrojnog tipa, oduzimanje od promenljive a ne menja vrednost promenljive a.

Ulaz:

$$>> a = int8(2^7 - 1)$$
$$>> a = a + 1$$
$$>> a = uint8(0)$$
$$>> a = a - 1$$

Konverzija osnove brojnog sistema

Moguće je sve celobrojne tipove prikazati u različitim osnovama.

Konverzija osnove brojnog sistema

Moguće je sve celobrojne tipove prikazati u različitim osnovama.

dec2base - daje zapis broja iz dekadne u proizvoljnu osnovu
base2dec - konvertuje zapis broja iz proizvoljne osnove u dekadnu
dec2hex - konvertuje zapis iz dekadne osnove u osnovu 16
hex2dec - konvertuje zapis broja iz heksadecimalne notacije u dekadnu
dec2bin - konvertuje zapis iz dekadne osnove u binarnu
bin2dec -konvertuje zapis broja iz binarne notacije u dekadnu

Konverzija osnove brojnog sistema

Povratni tip je string, izuzev funkcije *base2dec* čiji je povratni tip *double*.

Konverzija osnove brojnog sistema

Povratni tip je string, izuzev funkcije *base2dec* čiji je povratni tip *double*.
Funkcija *dec2base* prihvata sve numeričke tipove podataka, dok ostale funkcije prihvataju kao argument string zapisan u odgovarajućoj osnovi.

Konverzija osnove brojnog sistema

Povratni tip je string, izuzev funkcije *base2dec* čiji je povratni tip *double*.
Funkcija *dec2base* prihvata sve numeričke tipove podataka, dok ostale funkcije prihvataju kao argument string zapisan u odgovarajućoj osnovi.

Ulaz:

$$>> a = int64(56); \quad >> b = single(56); \quad >> c = 56;$$
$$>> a = dec2base(a, 7); \quad >> whos\ a$$
$$>> a = base2dec(a, 7); \quad >> whos\ a$$

Brojevi u pokretnom zarezu

- Matlab ima dva formata za predstavljanje brojeva u pokretnom zarezu: brojevi jednostruke preciznosti *single* i brojevi dvostruke preciznosti *double*.

Brojevi u pokretnom zarezu

- Matlab ima dva formata za predstavljanje brojeva u pokretnom zarezu: brojevi jednostruke preciznosti *single* i brojevi dvostruke preciznosti *double*.
- Svaki podatak tipa *double* zauzima 64 bita, ili 8 bajtova memorije. Bit na poziciji 63 nosi informaciju o znaku broja, (0 za pozitivne i 1 za negativne), bitovi 62 do 52 nose informacije o eksponentu broja, bitovi pozicije 51 do 0 služe za zapis normalizovane mantise.

Brojevi u pokretnom zarezu

- Matlab ima dva formata za predstavljanje brojeva u pokretnom zarezu: brojevi jednostruke preciznosti *single* i brojevi dvostruke preciznosti *double*.
- Svaki podatak tipa *double* zauzima 64 bita, ili 8 bajtova memorije. Bit na poziciji 63 nosi informaciju o znaku broja, (0 za pozitivne i 1 za negativne), bitovi 62 do 52 nose informacije o eksponentu broja, bitovi pozicije 51 do 0 služe za zapis normalizovane mantise.
- Maksimalna i minimalna pozitivna vrednost broja koja može biti prikazana u formatu double:

Brojevi u pokretnom zarezu

- Matlab ima dva formata za predstavljanje brojeva u pokretnom zarezu: brojevi jednostruke preciznosti *single* i brojevi dvostruke preciznosti *double*.
- Svaki podatak tipa *double* zauzima 64 bita, ili 8 bajtova memorije. Bit na poziciji 63 nosi informaciju o znaku broja, (0 za pozitivne i 1 za negativne), bitovi 62 do 52 nose informacije o eksponentu broja, bitovi pozicije 51 do 0 služe za zapis normalizovane mantise.
- Maksimalna i minimalna pozitivna vrednost broja koja može biti prikazana u formatu double:

$>>$ *realmin*, $>>$ *realmax*

Brojevi u pokretnom zarezu

Dodavanjem maksimalnom pozitivnom i oduzimanjem od minimalnog negativnog broja dobijaju se vrednosti Inf i -Inf, redom.

Brojevi u pokretnom zarezu

Dodavanjem maksimalnom pozitivnom i oduzimanjem od minimalnog negativnog broja dobijaju se vrednosti Inf i -Inf, redom.

Ulaz:

$$>> a = realmax + 1e + 308, >> a = -realmin - 1e + 308$$

Brojevi u pokretnom zarezu

Dodavanjem maksimalnom pozitivnom i oduzimanjem od minimalnog negativnog broja dobijaju se vrednosti Inf i -Inf, redom.

Ulaz:

$$>> a = realmax + 1e + 308, >> a = -realmin - 1e + 308$$

Tip *double* je vrednost koja se podrazumeva pri interpretaciji literala.

Brojevi u pokretnom zarezu

Dodavanjem maksimalnom pozitivnom i oduzimanjem od minimalnog negativnog broja dobijaju se vrednosti Inf i -Inf, redom.

Ulaz:

$$>> a = realmax + 1e + 308, >> a = -realmin - 1e + 308$$

Tip *double* je vrednost koja se podrazumeva pri interpretaciji literala.
Funkcija *isfloat* vraća informaciju da li podatak pripada tipu pokretnog zareza ili ne.

Brojevi u pokretnom zarezu

- Brojevi u formatu jednostruke preciznosti, dakle, tipa *single*. Tip podatka zauzima 32 bita, ili 4 bajta memorije.

Brojevi u pokretnom zarezu

- Brojevi u formatu jednostruke preciznosti, dakle, tipa *single*. Tip podatka zauzima 32 bita, ili 4 bajta memorije.
- Bit na poziciji 31 je bit znaka (0 za pozitivne brojeve i 1 za negativne brojeve), bitovi na pozicijama 30 do 23 služe za smeštanje eksponenta, dok bitovi 22 do 0 služe za smeštanje razlomljenog dela.

Brojevi u pokretnom zarezu

- Brojevi u formatu jednostruke preciznosti, dakle, tipa *single*. Tip podatka zauzima 32 bita, ili 4 bajta memorije.
- Bit na poziciji 31 je bit znaka (0 za pozitivne brojeve i 1 za negativne brojeve), bitovi na pozicijama 30 do 23 služe za smeštanje eksponenta, dok bitovi 22 do 0 služe za smeštanje razlomljenog dela.
- Maksimalna i minimalna pozitivna vrednost formata *single*:

$$>> realmax('single'), \quad >> realmin('single').$$

Brojevi u pokretnom zarezu

- Brojevi u formatu jednostruke preciznosti, dakle, tipa *single*. Tip podatka zauzima 32 bita, ili 4 bajta memorije.
- Bit na poziciji 31 je bit znaka (0 za pozitivne brojeve i 1 za negativne brojeve), bitovi na pozicijama 30 do 23 služe za smeštanje eksponenta, dok bitovi 22 do 0 služe za smeštanje razlomljenog dela.
- Maksimalna i minimalna pozitivna vrednost formata *single*:

$$>> realmax('single'), \quad >> realmin('single').$$

- Dodavanjem maksimalnom pozitivnom i oduzimanjem od minimalnog negativnog broja dobijaju se vrednosti *Inf* i *-Inf*, redom.

Brojevi u pokretnom zarezu

- Broj u nekom formatu ili literal može se konvertovati u broj u formatu single, koristeći funkciju za konstrukciju tipa.

Brojevi u pokretnom zarezu

- Broj u nekom formatu ili literal može se konvertovati u broj u formatu single, koristeći funkciju za konstrukciju tipa.
- Funkcija *isfloat* daje vrednost tačno i u slučaju da je argument formata *single*.

$$>> a = single(1.5), >>whos\ a, >> isfloat(a).$$

Brojevi u pokretnom zarezu

- Vrednost broja π može se dobiti pozivanjem funkcije $>>pi$.

Brojevi u pokretnom zarezu

- Vrednost broja π može se dobiti pozivanjem funkcije $>>pi$.
- Funkcija *eps* vraća vrednost broja koji treba dodati broju 1.0 da bi se dobio broj koji je različit od broja 1.0 u datom formatu.

Brojevi u pokretnom zarezu

- Vrednost broja π može se dobiti pozivanjem funkcije $>>pi$.
- Funkcija *eps* vraća vrednost broja koji treba dodati broju 1.0 da bi se dobio broj koji je različit od broja 1.0 u datom formatu.
- Kako je u Matlabu podrazumevani format double, to je podrazumevana vrednost funkcije eps broj koji treba dodati broju 1.0 da bi se dobio broj koji je veci od ovog broja a koji se može predstaviti u double formatu.

Brojevi u pokretnom zarezu

- Vrednost broja π može se dobiti pozivanjem funkcije $>>pi$.
- Funkcija *eps* vraća vrednost broja koji treba dodati broju 1.0 da bi se dobio broj koji je različit od broja 1.0 u datom formatu.
- Kako je u Matlabu podrazumevani format double, to je podrazumevana vrednost funkcije eps broj koji treba dodati broju 1.0 da bi se dobio broj koji je veci od ovog broja a koji se može predstaviti u double formatu.

$$>> eps, \quad >> eps('single').$$

Brojevi u pokretnom zarezu

- Napomenimo na kraju da se vrednosti beskonačnosti razlikuju zavisno od toga kome tipu pripadaju.

Brojevi u pokretnom zarezu

- Napomenimo na kraju da se vrednosti beskonačnosti razlikuju zavisno od toga kome tipu pripadaju.

$$>> a = Inf; \quad >> b = single(Inf); \quad >> whos.$$

Kompleksni brojevi

- Imaginarna jedinica u Matlabu je reprezentovana literalom *i* ili *j*.

Kompleksni brojevi

- Imaginarna jedinica u Matlabu je reprezentovana literalom *i* ili *j*.

```
>> a = 1 + i
>> a = complex(1,2)
>> real(a)
>> conj(a)
>> angle(a)
>>whos a
>> a = complex(1,1)
>> imag(a)
>> abs(a)
```

Kompleksni brojevi

- Imaginarna jedinica u Matlabu je reprezentovana literalom *i* ili *j*.

```
>> a = 1 + i
>> a = complex(1,2)
>> real(a)
>> conj(a)
>> angle(a)
```

```
>>whos a
>> a = complex(1,1)
>> imag(a)
>> abs(a)
```

U sadašnjoj implementaciji Matlaba, nemoguće je koristiti kompleksne brojeve sa tipom *integer*. Dakle, kompleksne brojeve je moguće kreirati samo sa tipovima podataka u pokretnom zarezu, *single* i *double*.

Kompleksni brojevi

- Kompleksni brojevi se takođe, pamte kako matrice tipa 1×1, ali odgovarajući atribut podataka ima vrednost *complex*.

Kompleksni brojevi

- Kompleksni brojevi se takođe, pamte kako matrice tipa 1×1, ali odgovarajući atribut podataka ima vrednost *complex*.
- Kompleksan broj tipa *double*, dakle, sa realnim i imaginarnim delom tipa *double*, zauzima duplo više memorije od podatka tipa *double*, jer treba smestiti realni i imaginarni deo.

```
>> a = 1 + i;                    >>whos a
>> a = single(1) + single(1) * i;    >>whos a
```

Specijalne vrednosti

- U specijalne vrednosti standarda IEEE-754 ubrajamo beskonačnost *Inf* i specijalnu oznaku *NaN* koja govori da sadržaj formata ne predstavlja broj.

Specijalne vrednosti

- U specijalne vrednosti standarda IEEE-754 ubrajamo beskonačnost *Inf* i specijalnu oznaku *NaN* koja govori da sadržaj formata ne predstavlja broj.

```
>> 1/0          >> -1/0
>> log(0)       >> isinf(log(0))
>> a = NaN;     >> isnan(a)
```

Specijalne vrednosti

- U specijalne vrednosti standarda IEEE-754 ubrajamo beskonačnost *Inf* i specijalnu oznaku *NaN* koja govori da sadržaj formata ne predstavlja broj.

```
>> 1/0          >> -1/0
>> log(0)       >> isinf(log(0))
>> a = NaN;     >> isnan(a)
```

Literal *NaN* reprezentuje vrednost formata *single* ili *double* koja nije realni broj ili beskonačnost.

Specijalne vrednosti

- U specijalne vrednosti standarda IEEE-754 ubrajamo beskonačnost *Inf* i specijalnu oznaku *NaN* koja govori da sadržaj formata ne predstavlja broj.

```
>> 1/0          >> -1/0
>> log(0)       >> isinf(log(0))
>> a = NaN;     >> isnan(a)
```

Literal *NaN* reprezentuje vrednost formata *single* ili *double* koja nije realni broj ili beskonačnost. Funkcija *isnan* daje informaciju da li je vrednost argumenta *NaN*.

Funkcije koje identifkuju numeričke tipove

Funkcije koje identifkuju numeričke tipove

whos a	ispisuje osnovne podatke promenljive *a*
class(a)	vraća tip promenljive *a*
isnumeric(a)	vraća informaciju da li je vrednost *a* numeričkog tipa
isa(a, tip)	vraća informaciju da li je vrednost *a* numeričkog tipa mogu se kao drugi argument koristiti i *'int8'*, *'uint8'*, *'int16'*, *'uint16'*, *'int32'*, *'uint32'*, *'int64'*, *'uint64'*, *'float'*, *'single'*, *'double'* sa očiglednim opisima
isreal(a)	vraća informaciju da li je vrednost *a* realan ili kompleksan broj
isnan(a)	vraća informaciju da li je vrednost *a NaN*
isinf(a)	vraća informaciju da li je vrednost *a* beskonačna
isfinite(a)	vraća informaciju da li je vrednost a konačna

Aritmetičke operacije nad skalarima

- Sa svim pobrojanim numeričkim tipovima moguće je izvršavati osnovne aritmetičke operacije, dakle, operacije +, -, *, /.

Aritmetičke operacije nad skalarima

- Sa svim pobrojanim numeričkim tipovima moguće je izvršavati osnovne aritmetičke operacije, dakle, operacije +, -, *, /.
- Sve operacije se standardno izvršavaju nad tipom *double*.

Aritmetičke operacije nad skalarima

- Sa svim pobrojanim numeričkim tipovima moguće je izvršavati osnovne aritmetičke operacije, dakle, operacije +, -, *, /.
- Sve operacije se standardno izvršavaju nad tipom *double*.
- Sabiranje dva podatka tipa *integer* koja ne pripadaju istom tipu rezultuje greškom.

Aritmetičke operacije nad skalarima

- Sa svim pobrojanim numeričkim tipovima moguće je izvršavati osnovne aritmetičke operacije, dakle, operacije +, -, *, /.
- Sve operacije se standardno izvršavaju nad tipom *double*.
- Sabiranje dva podatka tipa *integer* koja ne pripadaju istom tipu rezultuje greškom.

$$>> a = int8(2) + int16(200)$$

Aritmetičke operacije nad skalarima

- Sa svim pobrojanim numeričkim tipovima moguće je izvršavati osnovne aritmetičke operacije, dakle, operacije +, -, *, /.
- Sve operacije se standardno izvršavaju nad tipom *double*.
- Sabiranje dva podatka tipa *integer* koja ne pripadaju istom tipu rezultuje greškom.

$$>> a = int8(2) + int16(200)$$

Sa podacima tipa integer različitih tipova ne može se izvršiti ni jedna aritmetička operacija.

Aritmetičke operacije nad skalarima

- Moguće je kombinovati jedan operand tipa *integer* sa operandom tipa *double*, ali je nedozvoljeno kombinovati tip *integer* sa podatkom tipa *single*.

Aritmetičke operacije nad skalarima

- Moguće je kombinovati jedan operand tipa *integer* sa operandom tipa *double*, ali je nedozvoljeno kombinovati tip *integer* sa podatkom tipa *single*.
- Matlab kad sabira podatke tipa *integer* sa podatkom tipa *double* podatke smešta u tip integer, čak i po cenu gubitka tačne vrednosti.

Aritmetičke operacije nad skalarima

- Moguće je kombinovati jedan operand tipa *integer* sa operandom tipa *double*, ali je nedozvoljeno kombinovati tip *integer* sa podatkom tipa *single*.
- Matlab kad sabira podatke tipa *integer* sa podatkom tipa *double* podatke smešta u tip integer, čak i po cenu gubitka tačne vrednosti.

$$>> a = int8(2) + double(1e + 300) \quad >> whos\ a$$
$$>> a = int8(1) + Inf$$
$$>> a = int8(2) + single(1e + 037)$$

Aritmetičke operacije nad skalarima

- Operacije * i / sa tipom integer: rezultat se smešta u tip kome pripadaju operandi.

Aritmetičke operacije nad skalarima

- Operacije * i / sa tipom integer: rezultat se smešta u tip kome pripadaju operandi.
- Ako dođe do prekoračenja rezultat je najveća, najmanja, moguća vrednost u okviru zadatog tipa.

Aritmetičke operacije nad skalarima

- Operacije * i / sa tipom integer: rezultat se smešta u tip kome pripadaju operandi.
- Ako dođe do prekoračenja rezultat je najveća, najmanja, moguća vrednost u okviru zadatog tipa.

$$
\begin{array}{ll}
>> a = uint8(255) & >> b = uint8(255) \\
>> c = a * b & >> whos\ c \\
>> a = int8(-128) & >> b = int8(127) \\
>> c = a * b & >> whos\ c
\end{array}
$$

Aritmetičke operacije nad skalarima

```
>> a = uint8(4); b = uint8(7);
>> c = a/b                    >>whos c
>> a = int8(4); b = int8(7);
>> c = a * b                  >>whos c
>> c = a/b                    >>whos c
>> a = int8(-4); b = int8(7);
>> c = a/b                    >>whos c
>> a = int8(7); b = int8(7);
>> a = int8(-7); b = int8(7);
```

Aritmetičke operacije nad skalarima

```
>> a = uint8(4); b = uint8(7);
>> c = a/b                    >>whos c
>> a = int8(4); b = int8(7);
>> c = a * b                  >>whos c
>> c = a/b                    >>whos c
>> a = int8(-4); b = int8(7);
>> c = a/b                    >>whos c
>> a = int8(7); b = int8(7);
>> a = int8(-7); b = int8(7);
```

Deljenje celobrojnog tipa je deljenje koje rezultat operacije zaokružuje ka najbližem celom broju.

Aritmetičke operacije nad skalarima

- Deljenje celobrojnih tipova moguće je realizovati i funkcijom *idivide*.

Aritmetičke operacije nad skalarima

- Deljenje celobrojnih tipova moguće je realizovati i funkcijom *idivide*.

```
>>%zaokruzivanje prema nuli
>> idivide(int8(-2), int8(3),' fix')
>> idivide(int8(2), int8(3),' fix')
>>%zaokruzivanje prema najblizem integeru
>> idivide(int8(-2), int8(3),' round')
>> idivide(int8(2), int8(3),' round')
>>%zaokruzivanje prema -inf
>> idivide(int8(-2), int8(3),' floor')
>> idivide(int8(2), int8(3),' floor')
>>%zaokruzivanje prema +inf
>> idivide(int8(-2), int8(3),' ceil')
>> idivide(int8(2), int8(3),' ceil')
```

Aritmetičke operacije nad skalarima

- Za dobijanje ostatka deljenja dva podatka tipa integer koriste se funkcije *rem* ili *mod*.

Aritmetičke operacije nad skalarima

- Za dobijanje ostatka deljenja dva podatka tipa integer koriste se funkcije *rem* ili *mod*.

$$>> a = rem(int8(-5), int8(2))$$
$$>> a = mod(int8(-5), int8(2))$$
$$>> a = rem(int8(5), int8(2))$$
$$>> a = mod(int8(5), int8(2))$$

Aritmetičke operacije nad skalarima

- Za dobijanje ostatka deljenja dva podatka tipa integer koriste se funkcije *rem* ili *mod*.

$$>> a = rem(int8(-5), int8(2))$$
$$>> a = mod(int8(-5), int8(2))$$
$$>> a = rem(int8(5), int8(2))$$
$$>> a = mod(int8(5), int8(2))$$

Ako je vrednost drugog argumenta pozitivna, funkcije *mod* i *rem* se ponašaju na isti način pod uslovom da je prvi argument pozitivan.

Aritmetičke operacije nad skalarima

- Za dobijanje ostatka deljenja dva podatka tipa integer koriste se funkcije *rem* ili *mod*.

$$>> a = rem(int8(-5), int8(2))$$
$$>> a = mod(int8(-5), int8(2))$$
$$>> a = rem(int8(5), int8(2))$$
$$>> a = mod(int8(5), int8(2))$$

Ako je vrednost drugog argumenta pozitivna, funkcije *mod* i *rem* se ponašaju na isti način pod uslovom da je prvi argument pozitivan. Ako je drugi argument negativan funkcije ne moraju da proizvedu isti rezultat.

Aritmetičke operacije nad skalarima

- Za dobijanje ostatka deljenja dva podatka tipa integer koriste se funkcije *rem* ili *mod*.

$$>> a = rem(int8(-5), int8(2))$$
$$>> a = mod(int8(-5), int8(2))$$
$$>> a = rem(int8(5), int8(2))$$
$$>> a = mod(int8(5), int8(2))$$

Ako je vrednost drugog argumenta pozitivna, funkcije *mod* i *rem* se ponašaju na isti način pod uslovom da je prvi argument pozitivan. Ako je drugi argument negativan funkcije ne moraju da proizvedu isti rezultat. Funkcija *mod(x,y)* računa $x - floor(x/y) * y$, dok funkcija *rem* računa $x - fix(x/y) * y$.

Aritmetičke operacije nad skalarima

Kombinovanjem tipova *single* i *double* u aritmetičkim izrazima Matlab smešta rezultat u tip *single*, makar i po cenu gubitka podataka.

Aritmetičke operacije nad skalarima

Kombinovanjem tipova *single* i *double* u aritmetičkim izrazima Matlab smešta rezultat u tip *single*, makar i po cenu gubitka podataka.

```
>> a = single(1) + double(1);          >>whos a
>> a = single(1) + double(1e + 100)
```

Aritmetičke operacije nad skalarima

Kombinovanjem tipova *single* i *double* u aritmetičkim izrazima Matlab smešta rezultat u tip *single*, makar i po cenu gubitka podataka.

```
>> a = single(1) + double(1);          >>whos a
>> a = single(1) + double(1e + 100)
```

Operacije sa kompleksnim brojevima imaju slične osobine kao operacije sa tipovima kojima kompleksni brojevi pripadaju.

Aritmetičke operacije nad skalarima

Kombinovanjem tipova *single* i *double* u aritmetičkim izrazima Matlab smešta rezultat u tip *single*, makar i po cenu gubitka podataka.

```
>> a = single(1) + double(1);          >>whos a
>> a = single(1) + double(1e + 100)
```

Operacije sa kompleksnim brojevima imaju slične osobine kao operacije sa tipovima kojima kompleksni brojevi pripadaju. Međutim, postoji jedna bitna razlika. Bez obzira kog je tipa kompleksan broj nemoguće je izvršavati artimetičke operacije sa tipom *integer*.

Relacione operacije nad skalarima

Relacioni operatori dozvoljavaju bilo koju kombinaciju tipova operanada.

Relacione operacije nad skalarima

Relacioni operatori dozvoljavaju bilo koju kombinaciju tipova operanada.
Rezultat primene relacionih operatora je 1 ako je izraz tačan i 0 ako je izraz netačan.

Relacione operacije nad skalarima

Relacioni operatori dozvoljavaju bilo koju kombinaciju tipova operanada.
Rezultat primene relacionih operatora je 1 ako je izraz tačan i 0 ako je izraz netačan.
Podržani relacioni operatori su: $<, >, <=, >=, ==, \sim=$.

Relacione operacije nad skalarima

Relacioni operatori dozvoljavaju bilo koju kombinaciju tipova operanada.
Rezultat primene relacionih operatora je 1 ako je izraz tačan i 0 ako je izraz netačan.
Podržani relacioni operatori su: $<, >, <=, >=, ==, \sim=$.
Relacioni operatori nad operandima koji predstavljaju kompleksne brojeve upoređuju samo realne delove kompleksnih brojeva.

Relacione operacije nad skalarima

Relacioni operatori dozvoljavaju bilo koju kombinaciju tipova operanada.
Rezultat primene relacionih operatora je 1 ako je izraz tačan i 0 ako je izraz netačan.
Podržani relacioni operatori su: $<, >, <=, >=, ==, \sim=$.
Relacioni operatori nad operandima koji predstavljaju kompleksne brojeve upoređuju samo realne delove kompleksnih brojeva.

```
>> a = 1 + i;    >> b = 2;
>> a <= b
>> a = NaN;      >> b = NaN;
>> a == b        a ~= b
```

Operacije koje deluju nad bitovima

Funkcije koje deluju nad pojedinim bitovima neoznačenih celobrojnih tipova podataka.

Operacije koje deluju nad bitovima

Funkcije koje deluju nad pojedinim bitovima neoznačenih celobrojnih tipova podataka. Funkcije primaju argumente tipa neoznačenih celobrojnih vrednosti i vraćaju isti tip podataka.

Operacije koje deluju nad bitovima

Funkcije koje deluju nad pojedinim bitovima neoznačenih celobrojnih tipova podataka. Funkcije primaju argumente tipa neoznačenih celobrojnih vrednosti i vraćaju isti tip podataka.

bitand	vraća vrednost konjunkcije primenjene nad pojedinim bitovima argumenata
bitor	vraća vrednost disjunkcije primenjene nad pojedinim bitovima argumenata
bitcmp	vraća vrednost komplementa primenjene nad pojedinim bitovima argumenata, zadate dužine
bitxor	vraća vrednost ekskluzivne disjunkcije primenjene nad pojedinim bitovima argumenata
bitshift	vraća vrednost koja predstavlja prvi argumenta sa bitovima zarotiranim na levo ili desno

Operacije koje deluju nad bitovima

```
>> a = uint16(1234),   >> b = uint16(4321)
>> c = bitand(a, b),   >> whos c
>> dec2bin(a),   dec2bin(b),   dec2bin(c)
>> a = uint32(1234),   b = uint16(1234)
>> aC = bitcmp(a),   bC = bitcmp(b)
>> dec2bin(a),   dec2bin(aC)
>> dec2bin(b),   dec2bin(bC)
```

Logički tip podataka

Logički tip podataka je reprezentovan tipom *logical* i može imati jednu od dve vrednosti tačno ili netačno, koje su reprezentovane literalima *true* i *false*.

Logički tip podataka

Logički tip podataka je reprezentovan tipom *logical* i može imati jednu od dve vrednosti tačno ili netačno, koje su reprezentovane literalima *true* i *false*.

islogical	vraca informaciju da li je argument tipa logical
isa(*a*,′ *logical*′)	vraca informaciju da li je argument tipa logical
and	vraca vrednost konjunkcije dva argumenta
or	vraca vrednost disjunkcije dva argumenta
xor	vraca vrednost ekskluzivne disjunkcije dva argumenta
not	vraca vrednost negacije argumenta

Logički tip podataka

```
>> a = true; ,    >> b = false;
>> c = and(a, b),    >> whos c
>> a = true;    >> not(a)
>> logical(int16(3))
>> logical(single(0))
>> logical(1)
```

Logički tip podataka

```
>> a = true; ,    >> b = false;
>> c = and(a, b),    >> whos c
>> a = true;    >> not(a)
>> logical(int16(3))
>> logical(single(0))
>> logical(1)
```

Funkcija *logical* kreira logički tip podataka iz ostalih tipova podataka.

Logički tip podataka

```
>> a = true; ,    >> b = false;
>> c = and(a, b),    >> whos c
>> a = true;    >> not(a)
>> logical(int16(3))
>> logical(single(0))
>> logical(1)
```

Funkcija *logical* kreira logički tip podataka iz ostalih tipova podataka. Ako je argument funkcije *logical* vrednost jednaka nuli, bilo kojeg numeričkog tipa, onda funkcija vraća vrednost *false*, za ostale vrednosti argumenta vraćena vrednost biće *true*.

Logički operatori

Logički operatoti: &, &&, $\sim$, |, ||.

Logički operatori

Logički operatoti: &, &&, $\sim$, |, ||.
Prva oznaka & označava operator konjunkcije, dok druga oznaka && označava operator konjunkcije koji se izvršava na sledeći način: ako je na osnovu vrednosti prvog operanda moguće odrediti vrednost konjunkcije drugi operand se uopšte ne izračunava.

Logički operatori

Logički operatoti: &, &&, $\sim$, |, ||.
Prva oznaka & označava operator konjunkcije, dok druga oznaka && označava operator konjunkcije koji se izvršava na sledeći način: ako je na osnovu vrednosti prvog operanda moguće odrediti vrednost konjunkcije drugi operand se uopšte ne izračunava.
Slično je sa operacijom disjunkcije.

Logički operatori

Logički operatoti: &, &&, $\sim$, |, ||.
Prva oznaka & označava operator konjunkcije, dok druga oznaka && označava operator konjunkcije koji se izvršava na sledeći način: ako je na osnovu vrednosti prvog operanda moguće odrediti vrednost konjunkcije drugi operand se uopšte ne izračunava.
Slično je sa operacijom disjunkcije. Operatori && i || se nazivaju konjunkcija i disjunkcija u kratkom spoju (short circuit).

Logički operatori

Logički operatoti: &, &&, $\sim$, |, ||.
Prva oznaka & označava operator konjunkcije, dok druga oznaka && označava operator konjunkcije koji se izvršava na sledeći način: ako je na osnovu vrednosti prvog operanda moguće odrediti vrednost konjunkcije drugi operand se uopšte ne izračunava. Slično je sa operacijom disjunkcije. Operatori && i || se nazivaju konjunkcija i disjunkcija u kratkom spoju (short circuit).

$$>> a \&\& logical(int8(1) + int16(2))$$
$$>> a \& logical(int8(1) + int16(2))$$
$$>> a = true; \quad >> a \&\& logical(int8(1) + int16(2))$$

Aritmetičke operacije

Moguće je formirati aritmetičke izraze sa logičkim tipom kao operandom i drugim operandom koji je tipa *float.*

Aritmetičke operacije

Moguće je formirati aritmetičke izraze sa logičkim tipom kao operandom i drugim operandom koji je tipa *float.*

$$>> a = 10 + true, \quad >> whos\ a$$
$$>> aa = single(10) + false, \quad >> whos\ a$$

Relacioni operatori

Relacioni operatori, takođe, mogu imati jedan operand tipa *logical* a drugi operand numeričkog tipa.

Relacioni operatori

Relacioni operatori, takođe, mogu imati jedan operand tipa *logical* a drugi operand numeričkog tipa.
Kod relacionih operatora mogu da učestvuju svi numerički tipovi sa operandima tipa *logical*.

Relacioni operatori

Relacioni operatori, takođe, mogu imati jedan operand tipa *logical* a drugi operand numeričkog tipa.
Kod relacionih operatora mogu da učestvuju svi numerički tipovi sa operandima tipa *logical*.

$$>> a = 1; \quad >> b = (a \sim= false) \quad >> whos\ b$$
$$>> a = uint8(10); \quad >> b = (a > true) >> whos\ b$$

Relacioni operatori

Relacioni operatori, takođe, mogu imati jedan operand tipa *logical* a drugi operand numeričkog tipa.
Kod relacionih operatora mogu da učestvuju svi numerički tipovi sa operandima tipa *logical*.

$$>> a = 1; \quad >> b = (a \sim= false) \quad >> whos\ b$$
$$>> a = uint8(10); \quad >> b = (a > true) >> whos\ b$$

Povratni tip relacionih operatora je tipa *logical*.

Znakovni tip

Literali znakovnog tipa se navode unutar znakova ' i '.

Znakovni tip

Literali znakovnog tipa se navode unutar znakova ' i '.

```
>> a ='a'
>> uint16('a');
>>whos a
>> double('a')
```

Znakovni tip

Literali znakovnog tipa se navode unutar znakova ' i '.

```
>> a ='a'
>> uint16('a');
```

```
>>whos a
>> double('a')
```

Vrednost koda odgovarajućeg karaktera možemo dobiti konvertovanjem karaktera u neki od numeričkih tipova.

Aritmetičke operacije, relacioni i logički operatori

$>> a =' A'$	$>>$ *whos a*
$>> a = a + 1$	$>>$ *whos a*
$>> b = 1/'A'$	$>>$ *whos b*
$>> b = uint16(1)/a$	$>>$ *whos b*
$>> b = -a/uint16(1)$	$>>$ *whos b*

Aritmetičke operacije, relacioni i logički operatori

$>> a =' A'$	$>> whos\ a$
$>> a = a + 1$	$>> whos\ a$
$>> b = 1/'A'$	$>> whos\ b$
$>> b = uint16(1)/a$	$>> whos\ b$
$>> b = -a/uint16(1)$	$>> whos\ b$

Artimetičke operacije su moguće kad je jedan operand tipa *char* a drugi predstavlja neki numerički tip. Rezultat je istog tipa kao i operand numeričkog tipa.

Aritmetičke operacije, relacioni i logički operatori

$>> a =' A'$	$>>$ *whos a*
$>> a = a + 1$	$>>$ *whos a*
$>> b = 1/'A'$	$>>$ *whos b*
$>> b = uint16(1)/a$	$>>$ *whos b*
$>> b = -a/uint16(1)$	$>>$ *whos b*

Artimetičke operacije su moguće kad je jedan operand tipa *char* a drugi predstavlja neki numerički tip. Rezultat je istog tipa kao i operand numeričkog tipa.
Deljenje celobrojnih vrednosti zadovoljava pravilo zaokruživanja ka najbližoj celobrojnoj vrednosti u odgovarajučem tipu.

Aritmetičke operacije, relacioni i logički operatori

Relacioni operatori dozvoljavaju kombinovanje tipa *char* sa ostalim tipovima podataka. Moguća je i primena relacionih operatora u kojima su oba operanda tipa karakter.

Aritmetičke operacije, relacioni i logički operatori

Relacioni operatori dozvoljavaju kombinovanje tipa *char* sa ostalim tipovima podataka. Moguća je i primena relacionih operatora u kojima su oba operanda tipa karakter.
Moguće je primenjivati i logičke operatore sa operandima tipa *char*. U ovom slučaju moguće je kombinovati operand tipa *char* sa bilo kojim tipom podataka.

Aritmetičke operacije, relacioni i logički operatori

Relacioni operatori dozvoljavaju kombinovanje tipa *char* sa ostalim tipovima podataka. Moguća je i primena relacionih operatora u kojima su oba operanda tipa karakter.
Moguće je primenjivati i logičke operatore sa operandima tipa *char*. U ovom slučaju moguće je kombinovati operand tipa *char* sa bilo kojim tipom podataka.

$$>> a =' a'; \quad b =' b';$$
$$>> a \&\& b$$

Funkcije povezane sa znakovnim tipom

ischar	vraća vrednost tačno ako je argument tipa *char*
isa(*a*,′ *char*′)	vraća vrednost tačno ako je prvi argument tipa *char*
char	služi za kreiranje podataka tipa *char*