Tiesinės duomenų struktūros

Raimondas Čiegis

Matematinio modeliavimo katedra, e-paštas: rc@vgtu.lt

Rugsėjo 1 d., 2022

Tiesiniai sąrašai

Dažnai sprendžiame uždavinius, kuriems būdingos tokios savybės:

- didžiausias saugomų duomenų skaičius nėra iš anksto žinomas,

Tiesiniai sąrašai

Dažnai sprendžiame uždavinius, kuriems būdingos tokios savybės:

- didžiausias saugomų duomenų skaičius nėra iš anksto žinomas,
- elementų skaičius smarkiai kinta sprendžiant uždavinį,

Tiesiniai sąrašai

Dažnai sprendžiame uždavinius, kuriems būdingos tokios savybės:

- didžiausias saugomų duomenų skaičius nėra iš anksto žinomas,
- elementų skaičius smarkiai kinta sprendžiant uždavinį,
- nauji elementai gali būti įterpiami ne tik į aibės pabaigą, bet ir į bet kurią kitą vietą,

Tiesiniai sąrašai

Dažnai sprendžiame uždavinius, kuriems būdingos tokios savybės:

- didžiausias saugomų duomenų skaičius nėra iš anksto žinomas,
- elementų skaičius smarkiai kinta sprendžiant uždavinį,
- nauji elementai gali būti įterpiami ne tik į aibės pabaigą, bet ir į bet kurią kitą vietą,
- dažnai tenka šalinti elementus, esančius įvairiose aibės vietose.

Tokio uždavinio pavyzdžiu yra bibliotekoje saugomų knygų sąrašas: biblioteka gauna naujų knygų, o skaitytojai pasirenka tiek naujai išleistas, tiek senesnes knygas, kai kurios knygos nurašomos, ir biblioteka jų daugiau nesaugo. Mūsų analizė vienodai tinka tiek klasikinėms, tiek tiek ir internetinėms bibliotekoms.

Tokio uždavinio pavyzdžiu yra bibliotekoje saugomų knygų sąrašas: biblioteka gauna naujų knygų, o skaitytojai pasirenka tiek naujai išleistas, tiek senesnes knygas, kai kurios knygos nurašomos, ir biblioteka jų daugiau nesaugo. Mūsų analizė vienodai tinka tiek klasikinėms, tiek tiek ir internetinėms bibliotekoms.

Norėdami lengvai rasti reikalingas knygas, jas rūšiuojame pagal autoriaus pavardę. Tarkime, kad knygų sąrašą saugosime masyve. Tada šio masyvo ilgis turi būti gana didelis, kad galėtume ilgai naudoti šį sąrašą. Tačiau pradžioje, kol bibliotekoje yra nedaug knygų, toks didelis masyvas yra nereikalingas, todėl rezervuota kompiuterio atmintis bus naudojama neveiksmingai.

Tokio uždavinio pavyzdžiu yra bibliotekoje saugomų knygų sąrašas: biblioteka gauna naujų knygų, o skaitytojai pasirenka tiek naujai išleistas, tiek senesnes knygas, kai kurios knygos nurašomos, ir biblioteka jų daugiau nesaugo. Mūsų analizė vienodai tinka tiek klasikinėms, tiek tiek ir internetinėms bibliotekoms.

Norėdami lengvai rasti reikalingas knygas, jas rūšiuojame pagal autoriaus pavardę. Tarkime, kad knygų sąrašą saugosime masyve. Tada šio masyvo ilgis turi būti gana didelis, kad galėtume ilgai naudoti šį sąrašą. Tačiau pradžioje, kol bibliotekoje yra nedaug knygų, toks didelis masyvas yra nereikalingas, todėl rezervuota kompiuterio atmintis bus naudojama neveiksmingai.

Tarkime, kad knygų sąrašo ilgis yra 10 000 knygų, o biblioteka gavo naują Jono Avyžiaus romaną. Jį reikia įrašyti į 214 sąrašo vietą, todėl teks perstumti 99 786 įrašus, o tai tikrai ilgai trunkanti operacija. Panaši situacija susidarys ir tada, kai teks pašalinti iš sąrašo pradžios vieną iš knygų.

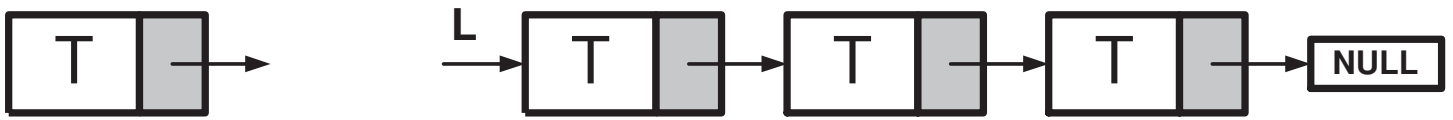
Vienakryptis tiesinis sąrašas

Beje, ši duomenų struktūrą yra pagrindinė realizuojant blokų grandines (angl. blockchain), kurios svarbios ir garantuojant kriptovaliutų technologijų efektyvumą.

Vienakryptis tiesinis sąrašas

Beje, ši duomenų struktūrą yra pagrindinė realizuojant blokų grandines (angl. blockchain), kurios svarbios ir garantuojant kriptovaliutų technologijų efektyvumą.

Vienakryptį tiesinį sąrašą vaizduoja tokia schema:

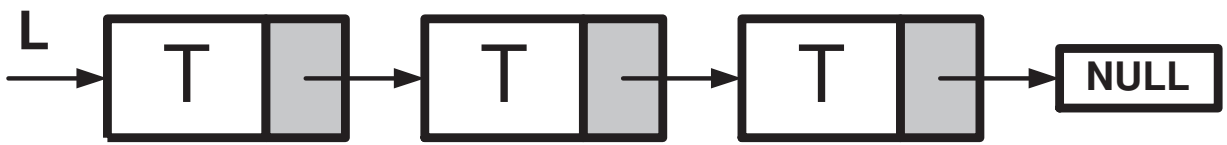


a) b)

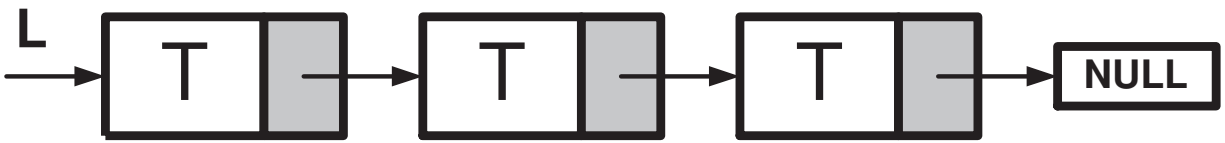
Apibrėžiame vieną atskirą *elementą*, kurį sudaro informacinė dalis data (ją realizuojame T tipo įrašu) ir rodyklė next, rodanti į kitą tokį elementą

```
struct node {
    T data;
    node * next;
}
```

Tada tiesinis vienakryptis sąrašas – tai grandinėlė, kuria sudaro pradžios rodyklė L ir rodyklėmis susietų elementų seka.



Tada tiesinis vienakryptis sąrašas – tai grandinėlė, kuria sudaro pradžios rodyklė L ir rodyklėmis susietų elementų seka.



Paskutinio elemento rodyklė rodo į tuščią elementą, tai ir yra sąrašo pabaigos požymis.

- tiesinis sąrašas yra sukuriamas apibrėžus rodyklę, kuri rodo į sąrašo pabaigą,

- tiesinis sąrašas yra sukuriamas apibrėžus rodyklę, kuri rodo į sąrašo pabaigą,
- bet kuriuo momentu jam skiriame tik tiek atminties, kiek reikia sąrašo elementams saugoti,

- tiesinis sąrašas yra sukuriamas apibrėžus rodyklę, kuri rodo į sąrašo pabaigą,
- bet kuriuo momentu jam skiriame tik tiek atminties, kiek reikia sąrašo elementams saugoti,
- kompiuterio atmintyje sąrašo elementai gali būti saugomi bet kurioje vietoje, o gretimi sąrašo elementai nebūtinai saugomi gretimose atminties ląstelėse,

- tiesinis sąrašas yra sukuriamas apibrėžus rodyklę, kuri rodo į sąrašo pabaigą,
- bet kuriuo momentu jam skiriame tik tiek atminties, kiek reikia sąrašo elementams saugoti,
- kompiuterio atmintyje sąrašo elementai gali būti saugomi bet kurioje vietoje, o gretimi sąrašo elementai nebūtinai saugomi gretimose atminties ląstelėse,
- į tiesinį sąrašą galime efektyviai įterpti naujus ir šalinti nereikalingus elementus,

- tiesinis sąrašas yra sukuriamas apibrėžus rodyklę, kuri rodo į sąrašo pabaigą,
- bet kuriuo momentu jam skiriame tik tiek atminties, kiek reikia sąrašo elementams saugoti,
- kompiuterio atmintyje sąrašo elementai gali būti saugomi bet kurioje vietoje, o gretimi sąrašo elementai nebūtinai saugomi gretimose atminties ląstelėse,
- į tiesinį sąrašą galime efektyviai įterpti naujus ir šalinti nereikalingus elementus,
- bet kuris sąrašo elementas yra pasiekiamas tik iš sąrašo pradžios, paeiliui perėjus visus prieš jį esančius elementus.

Pagrindiniai vienakrypčio sąrašo veiksmai

Sakykime, kad duomenys yra masyve A ir juos reikia perkelti į tiesinį vienakryptį sąrašą.

Pagrindiniai vienakrypčio sąrašo veiksmai

Sakykime, kad duomenys yra masyve A ir juos reikia perkelti į tiesinį vienakryptį sąrašą.

Patogiausia naują elementą įrašyti į sąrašo pradžią.

Kiekvienu ciklo žingsniu pirmiausia sukuriame naują elementą ir į jį užrašome reikalingą informaciją,

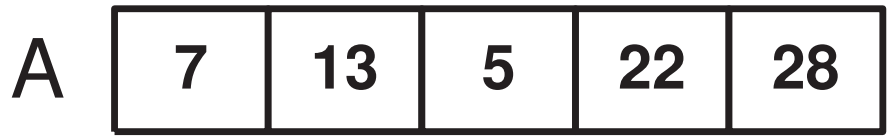
Pagrindiniai vienakrypčio sąrašo veiksmai

Sakykime, kad duomenys yra masyve A ir juos reikia perkelti į tiesinį vienakryptį sąrašą.

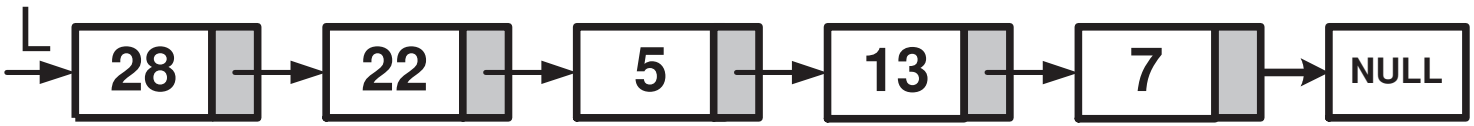
Patogiausia naują elementą įrašyti į sąrašo pradžią.

Kiekvienu ciklo žingsniu pirmiausia sukuriame naują elementą ir į jį užrašome reikalingą informaciją,

paskui šį elementą prijungiame prie sąrašo.



a)



b)

Tiesinio sąrašo L sudarymas iš masyvo A elementų.

Elemento paieška.

Reikia patikrinti, ar tiesiniame sąraše L yra saugomas įrašas, kurio raktas lygus t.

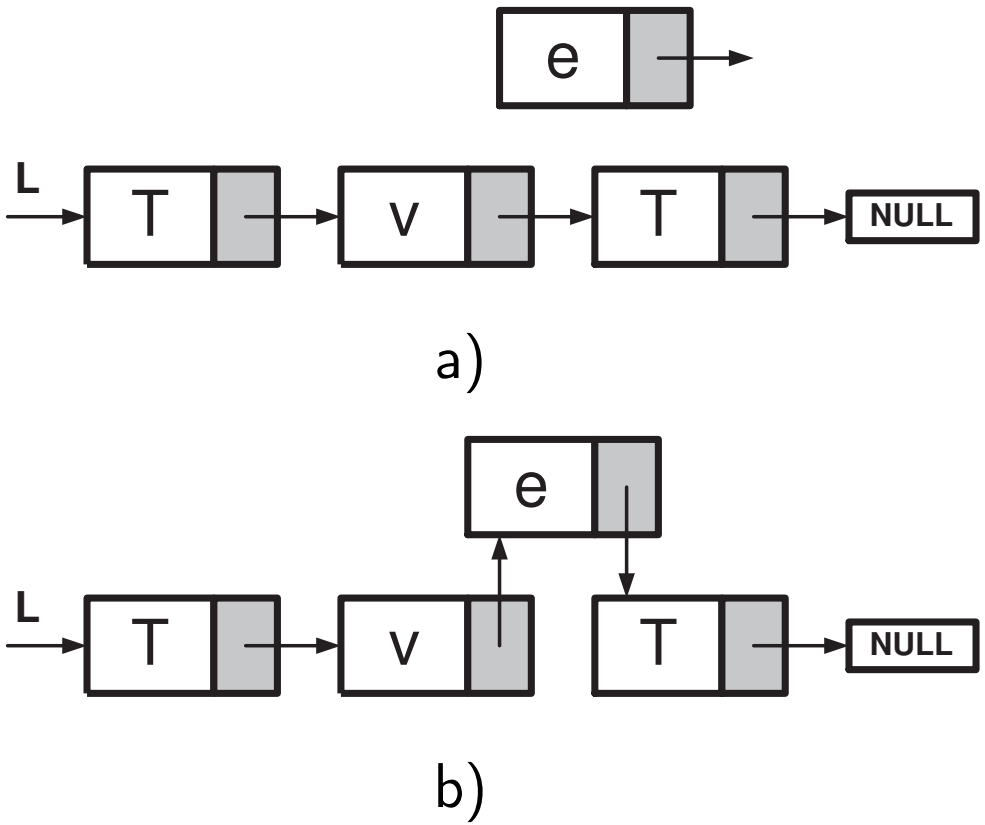
Elemento paieška.

Reikia patikrinti, ar tiesiniame sąraše L yra saugomas įrašas, kurio raktas lygus t.

Paiešką pradedame nuo sąrašo pradžios ir paeiliui tikriname visus sąrašo elementus, kol randame ieškomą įrašą arba pasiekiame sąrašo pabaigą

Naujo elemento įterpimas. Nesunku įterpti elementą e po elemento v, užtenka tik pakeisti dviejų rodyklių reikšmes.

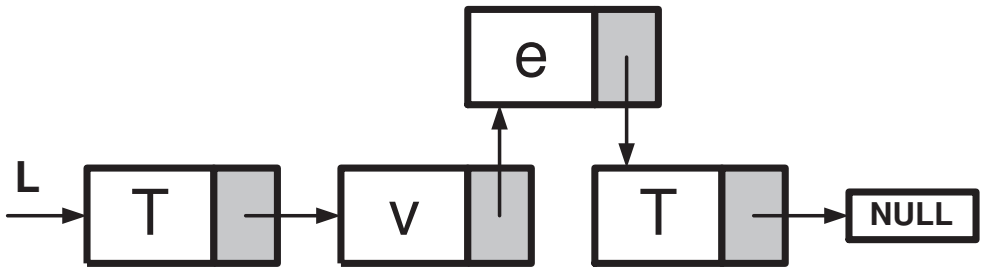
Naujo elemento įterpimas. Nesunku įterpti elementą e po elemento v, užtenka tik pakeisti dviejų rodyklių reikšmes.



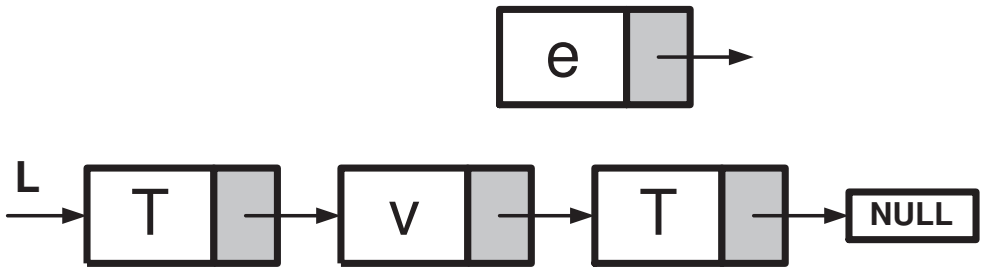
Naujo elemento įterpimas į tiesinį sąrašą: a) sąrašas prieš elemento e įterpimą, b) tiesinis sąrašas įterpus elementą.

Elemento šalinimas iš tiesinio sąrašo. Nesunku pašalinti iš sąrašo elementą, įrašytą po duotojo elemento v.

Elemento šalinimas iš tiesinio sąrašo. Nesunku pašalinti iš sąrašo elementą, įrašytą po duotojo elemento v.



a)



b)

Naujo elemento šalinimas iš tiesinio sąrašo: a) sąrašas prieš elemento e šalinimą, b) tiesinis sąrašas pašalinus elementą e.

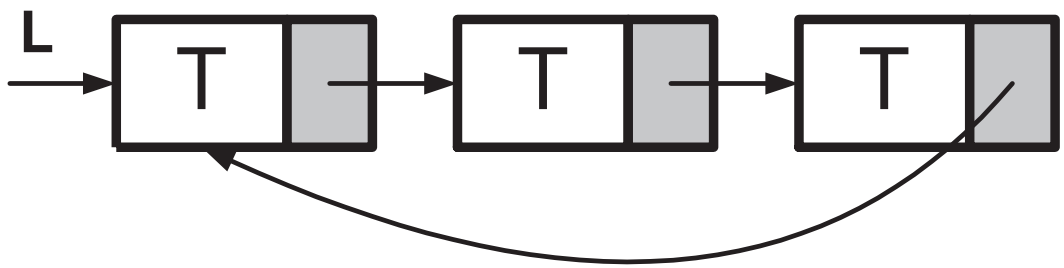
Daug sunkiau iš vienakrypčio sąrašo pašalintį patį elementą v, nes nežinome, koks elementas yra įrašytas prieš jį.

Daug sunkiau iš vienakrypčio sąrašo pašalintį patį elementą v, nes nežinome, koks elementas yra įrašytas prieš jį.

Sugalvokite, kaip tai padaryti greičiau, nei tikrinti visus sąrašo elementus nuo pradžios, kol rasime v ir pakeliui įsiminsime prieš tai stovinčio elemento adresą.

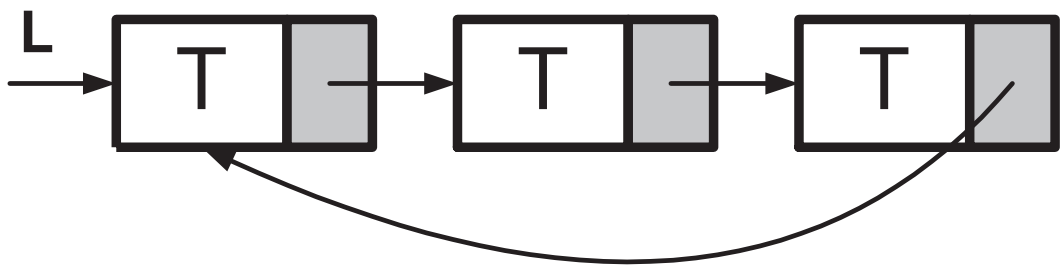
Ciklinis sąrašas

Ciklinis tiesinis sąrašas – tai vienakryptis tiesinis sąrašas, kurio paskutinis elementas yra gretimas pirmajam. Jis pavaizduotas paveiksle:



Ciklinis sąrašas

Ciklinis tiesinis sąrašas – tai vienakryptis tiesinis sąrašas, kurio paskutinis elementas yra gretimas pirmajam. Jis pavaizduotas paveiksle:



Ciklinio sąrašo paskutinio elemento next rodyklė sutampa su sąrašo pradžios rodykle, tai ir yra ciklinio sąrašo pabaigos požymis.

Paprasčiausios tiesinės duomenų struktūros

Susipažinsime su duomenų struktūromis, kurias gauname iš vienakrypčio tiesinio sąrašo apribodami jo veiksmų galimybes.

Paprasčiausios tiesinės duomenų struktūros

Susipažinsime su duomenų struktūromis, kurias gauname iš vienakrypčio tiesinio sąrašo apribodami jo veiksmų galimybes.

Atkreipsime dėmesį į tokį faktą: naujosios duomenų struktūros yra efektyvios ne todėl, kad išplečiame tiesinio sąrašo galimybes, bet atvirkščiai, pasinaudosime griežčiau apibrėžtomis jų veiksmų savybėmis.

Dėklas

Dėklas (angl. *stack*) yra labai dažnai naudojama duomenų struktūra. Ją apibūdina principas paskutinis įeina, pirmasis išeina (angl. *LIFO – Last In, First Out*).

Dėklas

Dėklas (angl. *stack*) yra labai dažnai naudojama duomenų struktūra. Ją apibūdina principas paskutinis įeina, pirmasis išeina (angl. *LIFO – Last In, First Out*).

Elementai įrašomi (operacija *push()*) ir šalinami (operacija *pop()*) tik iš sąrašo pradžios.

Dėklas

Dėklas (angl. *stack*) yra labai dažnai naudojama duomenų struktūra. Ją apibūdina principas paskutinis įeina, pirmasis išeina (angl. *LIFO – Last In, First Out*).

Elementai įrašomi (operacija *push()*) ir šalinami (operacija *pop()*) tik iš sąrašo pradžios.

Pavyzdžiui, pažvelkime į padėklų krūvą valgykloje: padėklas, kuris paskutinis padedamas ant viršaus, bus paimtas pirmas, o apatinis padėklas, kuris buvo padėtas pirmas, bus paimtas paskutinis.

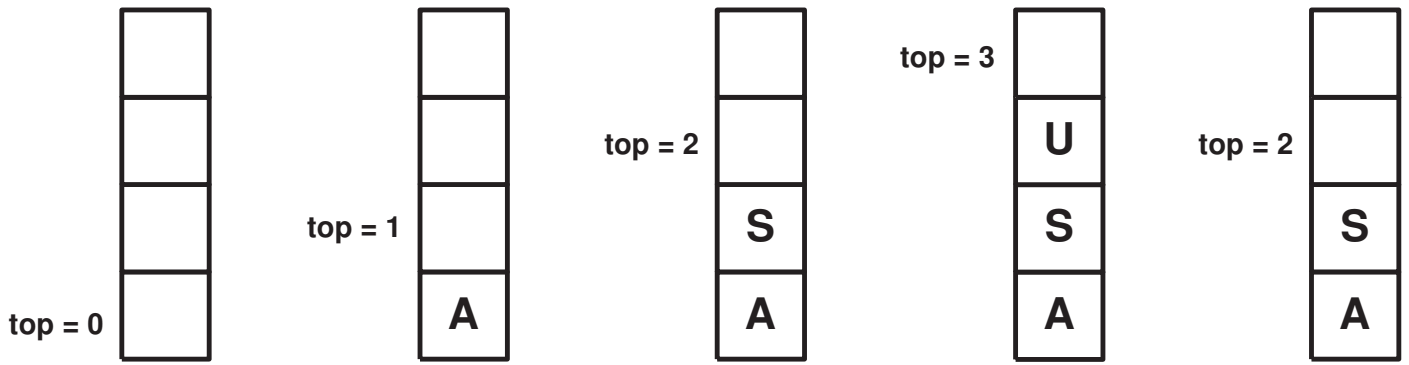
Kadangi dėklo funcionalumas yra smarkiai sumažintas lyginant su vienakrypčiu tiesiniu sąrašu, tai dažnai dėklai yra relizuojami naudojant kitas duomenų struktūras.

Kadangi dėklo funcionalumas yra smarkiai sumažintas lyginant su vienakrypčiu tiesiniu sąrašu, tai dažnai dėklai yra relizuojami naudojant kitas duomenų struktūras.

Pratybose išmoksite dėklą realizuoti naudodami masyvus ir papildomą žymeklį, nurodantį viršutinio elemento indekso numerį.

$$
\begin{aligned}
&\mathit{struct\ stack}\ \{\\
&\quad T\ \mathit{data}[N];\\
&\quad \mathit{int}\ \mathit{top} = 0;\\
&\}
\end{aligned}
$$

Paveiksle pavaizduotas dėklas, į kurį įterpiamos raidės A, S, U, o paskui raidė U pašalinama.



Dėklo realizavimas masyve: a) tuščias masyvas, b) push(A), c) push(S), d) push(U), e) pop()

Aritmetinės išraiškos vaizdavimas *postfix* forma

Dviejų skaičių a ir b sumą dažniausiai žymime $a + b$.

Aritmetinės išraiškos vaizdavimas *postfix* forma

Dviejų skaičių a ir b sumą dažniausiai žymime $a + b$.
Toks užrašas yra vadinamas aritmetinės išraiškos infix forma. Čia aritmetinis veiksmas užrašomas tarp dviejų operandų.

Aritmetinės išraiškos vaizdavimas *postfix* forma

Dviejų skaičių a ir b sumą dažniausiai žymime $a + b$.
Toks užrašas yra vadinamas aritmetinės išraiškos infix forma. Čia aritmetinis veiksmas užrašomas tarp dviejų operandų.

Kalkuliatoriuose aritmetinės išraiškos užrašomos prefix forma, kai iš pradžių rašomas aritmetinis veiksmas, o paskui operandai. Sumos *prefix* forma yra $+ab$.

Aritmetinės išraiškos vaizdavimas *postfix* forma

Dviejų skaičių a ir b sumą dažniausiai žymime $a + b$.
Toks užrašas yra vadinamas aritmetinės išraiškos infix forma. Čia aritmetinis veiksmas užrašomas tarp dviejų operandų.

Kalkuliatoriuose aritmetinės išraiškos užrašomos prefix forma, kai iš pradžių rašomas aritmetinis veiksmas, o paskui operandai. Sumos *prefix* forma yra $+ab$.

Šiuolaikiniuose kompiuteriuose aritmetinės išraiškos vaizduojamos postfix forma, kai iš pradžių rašomi operandai, o paskui aritmetinis veiksmas. Sumos *postfix* forma yra $ab+$.

Prefix ir *postfix* formos yra labai patogios, kadangi veiksmų atlikimo eiliškumą apibrėžiame nenaudodami skliaustų.

Prefix ir *postfix* formos yra labai patogios, kadangi veiksmų atlikimo eiliškumą apibrėžiame nenaudodami skliaustų.

Nagrinėkime aritmetinę išraišką $a + b * c$. Užrašysime jos *postfix* formą:

$$\begin{aligned} a + b * c &\longrightarrow a + (b * c) \longrightarrow a + (b\,c\,*) \\ &\longrightarrow a\,(b\,c\,*) + \longrightarrow a\,b\,c\,*\,+\,. \end{aligned}$$

Prefix ir *postfix* formos yra labai patogios, kadangi veiksmų atlikimo eiliškumą apibrėžiame nenaudodami skliaustų.

Nagrinėkime aritmetinę išraišką $a + b * c$. Užrašysime jos *postfix* formą:

$$a + b * c \longrightarrow a + (b * c) \longrightarrow a + (b\,c\,*)$$
$$\longrightarrow a\,(b\,c\,*) + \longrightarrow a\,b\,c\,* + .$$

Dabar nagrinėkime kitą aritmetinę išraišką $(a + b) * c$. Jos *postfix* formoje nereikia skliaustų, kurie buvo būtini *infix* formos išraiškoje:

$$(a + b) * c \longrightarrow (a\,b\,+) * c \longrightarrow (a\,b\,+)\,c\,* \longrightarrow a\,b\,+\,c\,* .$$

Priminsime aritmetinių veiksmų prioritetus:

- Aukščiausią prioritetą turi kėlimo laipsniu veiksmas, kurį žymėsime simboliu $\wedge$:

$$a^b = a \wedge b\,.$$

Šis veiksmas atliekamas iš dešinės į kairę, ir naujasis laipsnio kėlimas yra aukštesnio prioriteto veiksmas:

$$a \wedge b \wedge c = a \wedge (b \wedge c)\,.$$

Priminsime aritmetinių veiksmų prioritetus:

- Aukščiausią prioritetą turi kėlimo laipsniu veiksmas, kurį žymėsime simboliu $\wedge$:

$$a^b = a \wedge b\,.$$

Šis veiksmas atliekamas iš dešinės į kairę, ir naujasis laipsnio kėlimas yra aukštesnio prioriteto veiksmas:

$$a \wedge b \wedge c = a \wedge (b \wedge c)\,.$$

Priminsime aritmetinių veiksmų prioritetus:

- Aukščiausią prioritetą turi kėlimo laipsniu veiksmas, kurį žymėsime simboliu $\wedge$:

$$a^b = a \wedge b\,.$$

Šis veiksmas atliekamas iš dešinės į kairę, ir naujasis laipsnio kėlimas yra aukštesnio prioriteto veiksmas:

$$a \wedge b \wedge c = a \wedge (b \wedge c)\,.$$

Skaičiuojame tokį pavyzdį

$$3^{3^3} = 3^{27}.$$

Priminsime aritmetinių veiksmų prioritetus:

- Aukščiausią prioritetą turi kėlimo laipsniu veiksmas, kurį žymėsime simboliu $\wedge$:

$$a^b = a \wedge b\,.$$

Šis veiksmas atliekamas iš dešinės į kairę, ir naujasis laipsnio kėlimas yra aukštesnio prioriteto veiksmas:

$$a \wedge b \wedge c = a \wedge (b \wedge c)\,.$$

Skaičiuojame tokį pavyzdį

$$3^{3^3} = 3^{27}.$$

- Žemesnį prioritetą turi daugybos ir dalybos veiksmai. Jų eiliškumo tvarka yra iš kairės į dešinę:

$$a * b/c = (a * b)/c\,.$$

- Sumavimo ir atimties veiksmų prioritetas yra žemiausias, jų eiliškumo tvarka irgi iš kairės į dešinę:

$$a - b + c = (a - b) + c\,.$$

- Sumavimo ir atimties veiksmų prioritetas yra žemiausias, jų eiliškumo tvarka irgi iš kairės į dešinę:

$$a - b + c = (a - b) + c\,.$$

- Skliaustuose esantis reiškinys interpretuojamas kaip vienas operandas ir apskaičiuojamas prieš kitus veiksmus, t. y. skliaustų prioritetas yra aukštesnis už bet kurią aritmetinę operaciją.

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:
- Iš failo skaitome eilinį simbolį s.

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:
- Iš failo skaitome eilinį simbolį s.
- Jeigu s yra kairieji skliaustai "(" arba pradžios simbolis "[", tai s yra įterpiamas į dėklą S.

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:
- Iš failo skaitome eilinį simbolį s.
- Jeigu s yra kairieji skliaustai "(" arba pradžios simbolis "[", tai s yra įterpiamas į dėklą S.
- Jeigu s yra operandas, tai jį spausdiname (arba saugome antrajame dėkle P)

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:
- Iš failo skaitome eilinį simbolį s.
- Jeigu s yra kairieji skliaustai "(" arba pradžios simbolis "[", tai s yra įterpiamas į dėklą S.
- Jeigu s yra operandas, tai jį spausdiname (arba saugome antrajame dėkle P)
- Jeigu s yra aritmetinis veiksmas, tai tol, kol sėkmingai neužbaigsime šio etapo, iš dėklo S skaitome viršutinį elementą h (jis lieka dėkle);

Infix išraiškos užrašymas *postfix* forma

Infix išraišką papildome pradžios " [" ir pabaigos "]" simboliais.

- Kol failas su *infix* formos išraiška dar nėra perskaitytas iki pabaigos, vykdome tokį algoritmą:
- Iš failo skaitome eilinį simbolį s.
- Jeigu s yra kairieji skliaustai "(" arba pradžios simbolis "[", tai s yra įterpiamas į dėklą S.
- Jeigu s yra operandas, tai jį spausdiname (arba saugome antrajame dėkle P)
- Jeigu s yra aritmetinis veiksmas, tai tol, kol sėkmingai neužbaigsime šio etapo, iš dėklo S skaitome viršutinį elementą h (jis lieka dėkle);
- Jeigu h yra "(" arba "[", arba $h \prec s$, tai s yra įterpiamas į dėklą S. Etapas užbaigtas.

- Jeigu $s \prec= h$, tai h išimame iš dėklo S, elementą h spausdiname (arba saugome antrajame dėkle P) ir kartojame vis dar neužbaigto etapo procedūrą.

- Jeigu $s \prec = h$, tai h išimame iš dėklo S, elementą h spausdiname (arba saugome antrajame dėkle P) ir kartojame vis dar neužbaigto etapo procedūrą.
- Jeigu s yra dešinieji skliaustai ")" tai iš dėklo S paeiliui išimsime elementus h tol, kol jie nelygūs "(", ir tokius elementus spausdinsime.

- Jeigu $s \prec= h$, tai h išimame iš dėklo S, elementą h spausdiname (arba saugome antrajame dėkle P) ir kartojame vis dar neužbaigto etapo procedūrą.
- Jeigu s yra dešinieji skliaustai "$)$" tai iš dėklo S paeiliui išimsime elementus h tol, kol jie nelygūs "$($", ir tokius elementus spausdinsime.
- Jeigu s yra pabaigos simbolis "$]$" tai iš dėklo S paeiliui išimsime elementus h tol, kol jie nelygūs "$[$", ir tokius elementus spausdinsime.

Nagrinėkime *infix* aritmetinę išraišką $a + (b * c + d) \wedge f$.
Naudodami pateiktąjį algoritmą, užrašysime jos *postfix* formą.

Nagrinėkime *infix* aritmetinę išraišką $a + (b * c + d) \wedge f$.
Naudodami pateiktąjį algoritmą, užrašysime jos *postfix* formą.

Įėjimas	Dėklas	Postfix
[	[	
a	[	a
+	[+	
(	[+ (	
b	[+ (	b
*	[+ (*	
c	[+ (*	c

Nagrinėjame *infix* aritmetinę išraišką $a + (b * c + d) \wedge f$.

Įėjimas	Dėklas	Postfix
c	[	c
+	[+	
*	[+ (	
+	[+ (	*
(	[+ (+	
d	[+ (+	d
)	[+	+
^	[+ ^	
f	[+ ^	f
]	[+	^
	[	+

Taigi aritmetinės išraiškos $a + (b * c + d) \wedge f$ *postfix* forma yra

$$abc * d + f \wedge + .$$

Taigi aritmetinės išraiškos $a + (b * c + d) \wedge f$ *postfix* forma yra

$$abc * d + f \wedge + .$$

Skaičiuodami aritmetinės išraiškos, užrašytos *postfix* forma, reikšmę vėl naudojame dėklo duomenų struktūrą (pratybos).