DUOMENŲ STRUKTŪROS IR ALGORITMAI

MARIUS GŽEGOŽEVSKIS

TIESINĖS DUOMENŲ STRUKTŪROS

- ✓STEKAS.
- ✓EILĖ.
- ✓DEKAS.
- ✓TIESINIAI SĄRAŠAI.

STEKAS (angl. STACK)

Duomenų struktūra stekas - tai duomenų aibė, kuriai apibrėžtos tokios operacijos:

- ✓ inicializuoti steką (išskirti vietą stekui kompiuterio atmintyje);
- ✓ įterpti elementą x į steką (operacija **push(x)**);
- ✓ pašalinti elementą iš steko (operacija **pop**);
- ✓ skaityti steką;
- ✓ panaikinti steką (panaikinti vietą stekui kompiuterio atmintyje)

STEKAS

Duomenims steke taikomi loginiai apribojimai - operacijos pop metu iš steko bus pašalintas elementas, įterptas paskutiniu (taip vadinamas **LIFO** (Last-In-First-Out) principas).

STEKAS

Todėl stekas yra viena iš riboto išrinkimo klasės struktūrų. Lietuviškai steką siūloma vadinti dėklu, tačiau pastarasis terminas per daug netiksliai atskleidžia steko esmę.

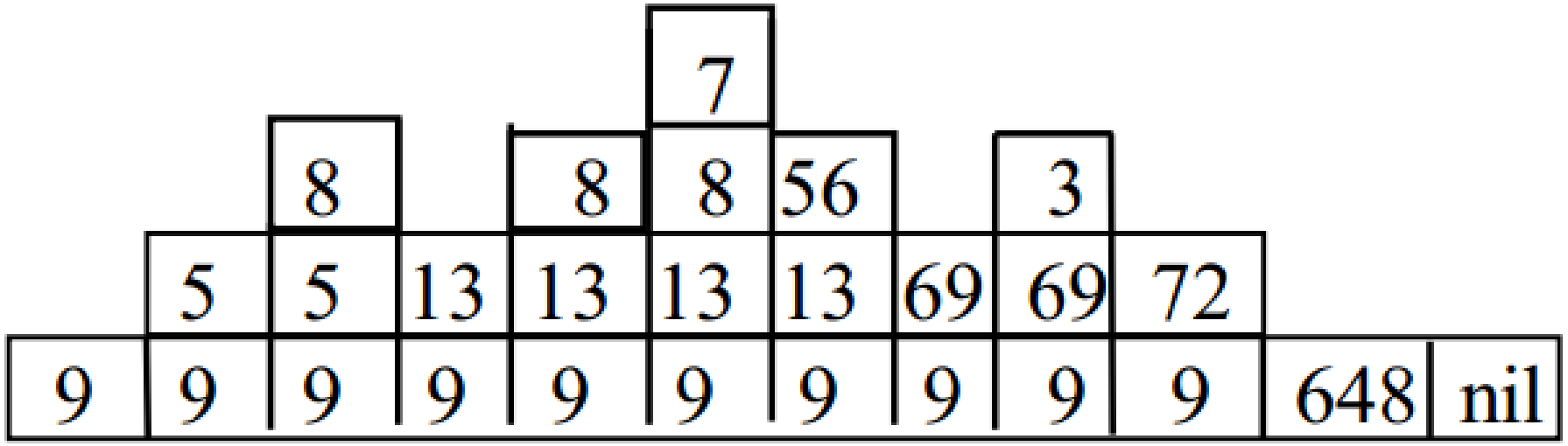
Steko struktūra naudojama įvairiausiose srityse: kompiuterio aparatūroje, programose, operacinių sistemų architektūroje, kompiliatoriuose, loginių ir simbolinių skaičiavimų algoritmuose, kitur.

STEKAS

Labai paplitęs pavyzdys, iliustruojantis steko privalumus, yra algebrinių reiškinių reikšmių skaičiavimas arba šių reiškinių teisingo užrašymo tikrinimas, pvz., ar teisingai išdėstyti skliausteliai reiškinyje. Skaičiuojant aritmetinio reiškinio **9*(((5+8)+(8*7))+3)** reikšmę, steko operacijų seka gali būti tokia:

push(9); push(5); push(8); push(pop+pop); push(8); push(7); push(pop*pop); push(pop+pop); push(3); push(pop+pop); push(pop*pop); writeln(pop).

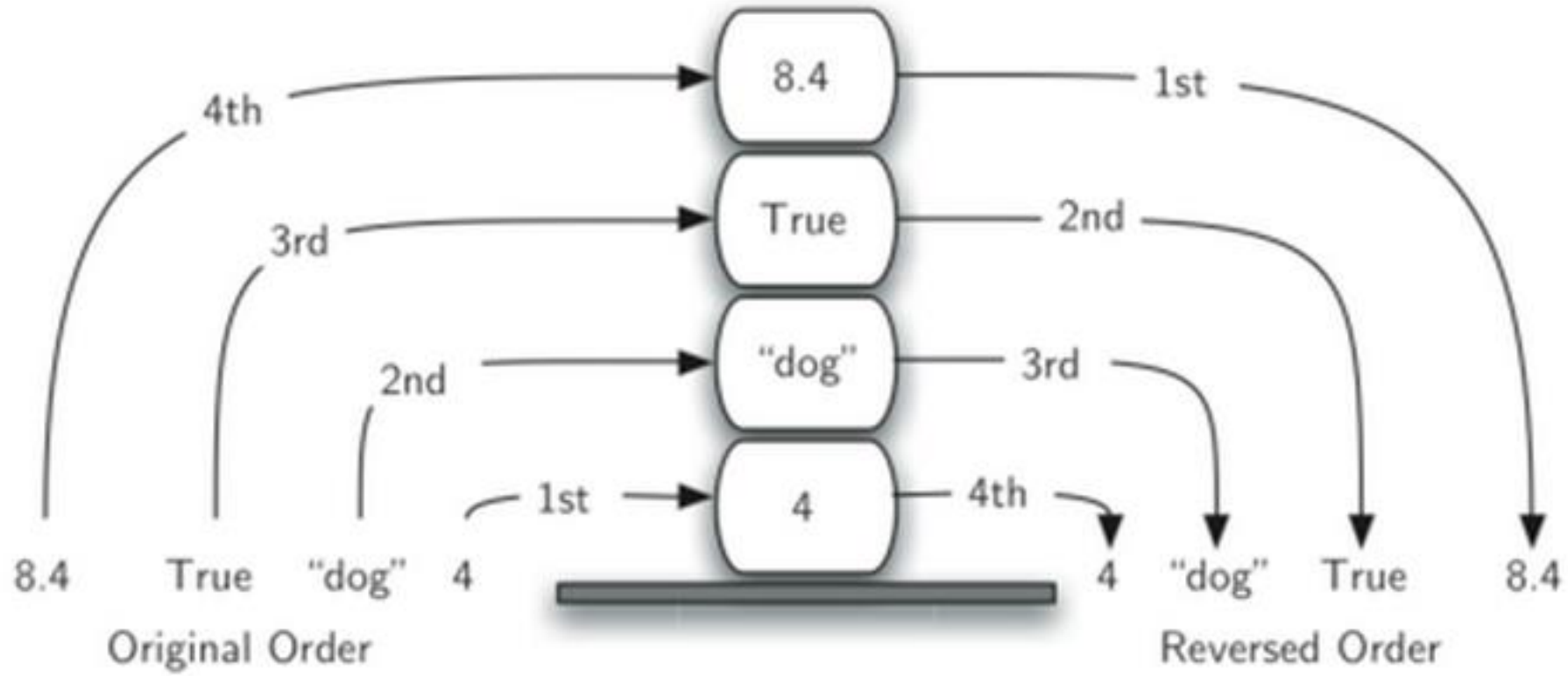
STEKO OPERACIJOS REIŠKINIUI: 9*(((5+8)+(8*7))+3)



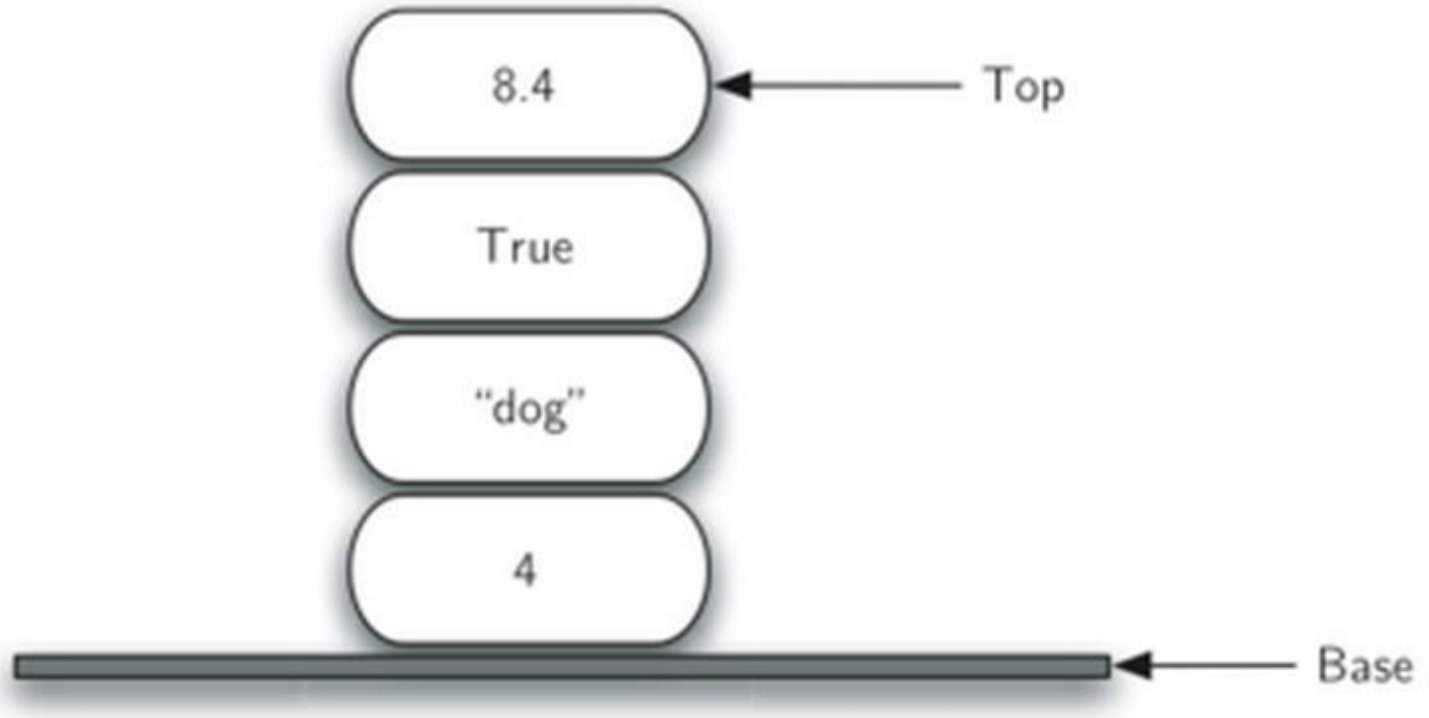
STEKAS

push(9); push(5);
push(8);
push(pop+pop);
push(8); push(7);
push(pop*pop);
push(pop+pop)
push(3);
push(pop+pop);
push(pop*pop);
writeln(pop).

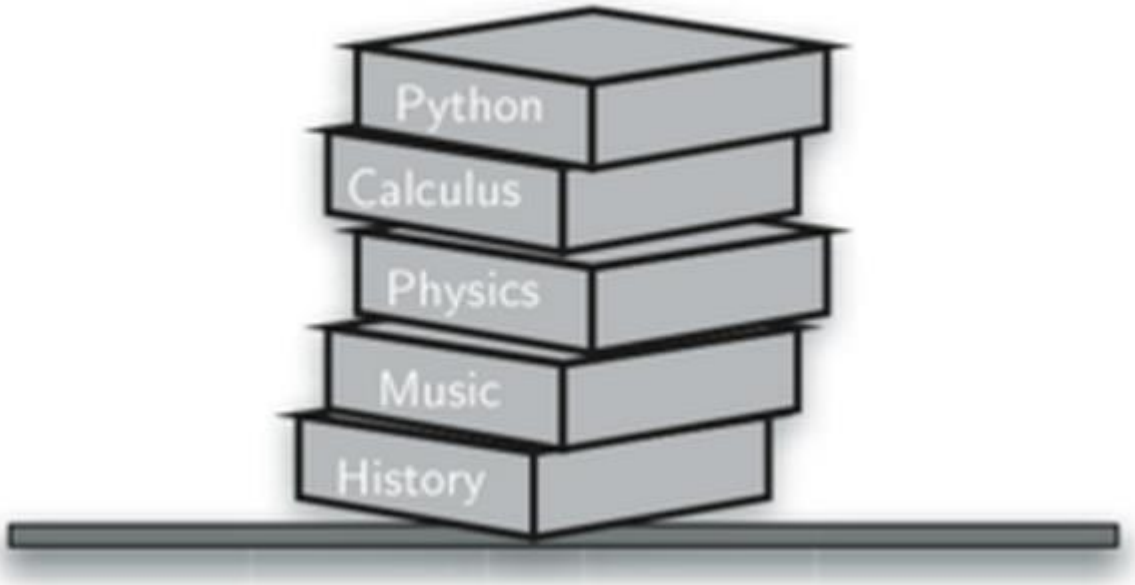
STEKO ELEMENTAI LIFO (Last in First Out: push(4), push(„dog“), push(True), push(8.4)



STEKAS



STEKAS SUDARYTAS IŠ KNYGŲ



STEKO OPERACIJOS REALIZUOJANT PYTHON PROGRAMAVIMO KALBA

Stack() - Sukuria naują steką, kuris yra tuščias. Taip pat turi būti be parametrų ir gražinti tuščią steką.
push(item) - Įdėti elementą **item** į steko viršūnę (angl. top). Parametras item skirtas nurodyti elementą, kuris bus dedamas į steką ir funkcija turi nieko negražinti.
pop() - Išmesti viršutinį elementą iš steko. Be parametrų, gražina elementą ir pakeičia steką.

STEKO OPERACIJOS REALIZUOJANT PYTHON PROGRAMAVIMO KALBA

peek() - Gražina viršutinį steko elementą, bet jo nepašalina. Funkcija be parametrų ir stekas nėra modifikuojamas.

isEmpty() – Patrikrina ar stekas yra tuščias. Funkcija gražina boolean reikšmę true arba false.

size() - Gražina skaičių esančių steke. Be parametrų ir gražina sveikojo tipo reikšmę.

Stack Operation	Stack Contents	Return Value
`s.isEmpty()`	`[]`	`True`
`s.push(4)`	`[4]`	
`s.push('dog')`	`[4,'dog']`	
`s.peek()`	`[4,'dog']`	`'dog'`
`s.push(True)`	`[4,'dog',True]`	
`s.size()`	`[4,'dog',True]`	`3`
`s.isEmpty()`	`[4,'dog',True]`	`False`
`s.push(8.4)`	`[4,'dog',True,8.4]`	
`s.pop()`	`[4,'dog',True]`	`8.4`
`s.pop()`	`[4,'dog']`	`True`
`s.size()`	`[4,'dog']`	`2`

REALIZACIJA PYTHON PROGRAMAVIMO KALBA SĄRAŠO PAGRINDU KUR TOP ELEMENTAS YRA PASKUTINIS SĄRAŠO ELEMENTAS. (Standartinės pop ir append funkcijos iš Python list efektyvumas O (1).

```
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def peek(self):
        return self.items[len(self.items)-1]

    def size(self):
        return len(self.items)
```

```
1  import Stack
2
3  s=Stack()
4
5  print(s.isEmpty())
6  s.push(4)
7  s.push('dog')
8  print(s.peek())
9  s.push(True)
10 print(s.size())
11 print(s.isEmpty())
12 s.push(8.4)
13 print(s.pop())
14 print(s.pop())
15 print(s.size())
```

Išvedimas:

```
True
dog
3
False
8.4
True
2
```

REALIZACIJA PYTHON PROGRAMAVIMO KALBA SĄRAŠO PAGRINDU KUR TOP ELEMENTAS YRA PIRMAS SĄRAŠO ELEMENTAS. (Standartinės pop ir insert funkcijos iš Python list efektyvumas O (n).

```
1  class Stack:
2      def __init__(self):
3          self.items = []
4
5      def isEmpty(self):
6          return self.items == []
7
8      def push(self, item):
9          self.items.insert(0,item)
10
11     def pop(self):
12         return self.items.pop(0)
13
14     def peek(self):
15         return self.items[0]
```

```
17     def size(self):
18         return len(self.items)
19
20 s = Stack()
21 s.push('hello')
22 s.push('true')
23 print(s.pop())
```

Reikėtų atkreipti dėmesį į algoritmo efektyvumą, kadangi galimų realizacijų yra keletas.

DUOMENŲ STRUKTŪRA STEKAS SKLIAUSTŲ SUBALANSAVIMO PROBLEMOS SPRENDIMUI

Naudojant steką yra išsprendžiamos realios kompiuterių mokslo problemos.

(5+6)∗(7+8)/(4+3) – tokia išraiška neabejotinai skliaustai nurodo operacijų seką.

```
(defun square(n)
    (* n n))
```

Programavimo kalboje Lisp funkcija yra apibrėžiama taip: Šios funkcijos pavadinimas yra square ir ji apibrėžia skaičių n pakeltą kvadratu.

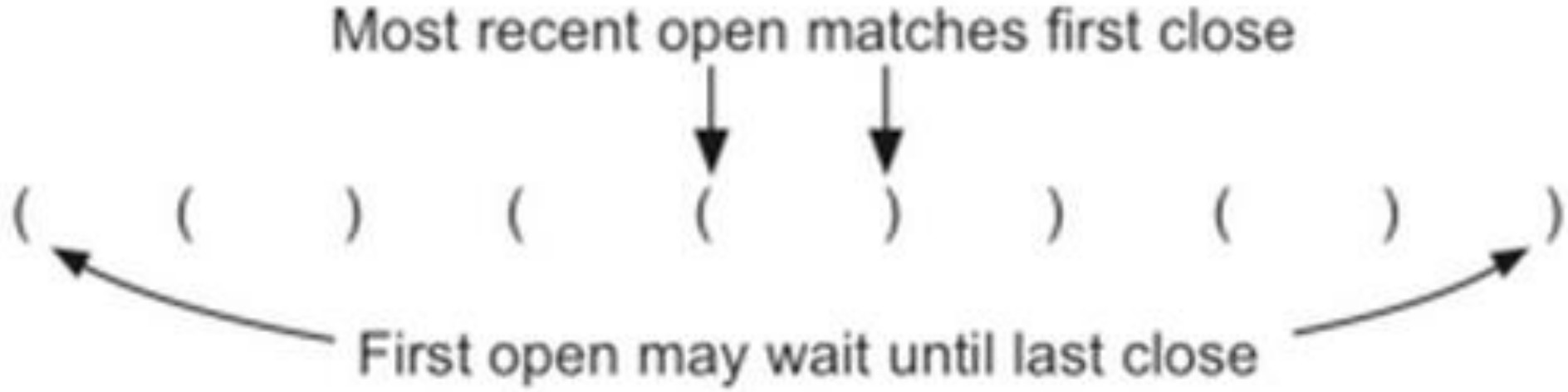
DUOMENŲ STRUKTŪRA STEKAS SKLIAUSTŲ SUBALANSAVIMO PROBLEMOS SPRENDIMUI

Abu pavyzdžiai turi griežtą skliaustų išdėstymą, kadangi tai gali nulemti nekorektišką veikimą. Skliaustai turi būti subalansuoti, pvz: Atidarytas (ir) uždarytas. Lisp programavimo kalboje yra apstu skliaustų:

```
(()()()())

((()()))

(()((())()))
```

```
(((((((())   A

())          B

(()()(()     C
```

Kuris Nesubalansuotas?


Most recent open matches first close
(( ) (( )) ( ))
First open may wait until last close

```
1  from pythonds.basic.stack import Stack
2
3  def parChecker(symbolString):
4      s = Stack()
5      balanced = True
6      index = 0
7      while index < len(symbolString) and balanced:
8          symbol = symbolString[index]
9          if symbol == "(":
10             s.push(symbol)
11         else:
12             if s.isEmpty():
13                 balanced = False
14             else:
15                 s.pop()
16
17         index = index + 1
18
19     if balanced and s.isEmpty():
20         return True
21     else:
22         return False
23
24 print(parChecker('((()))'))
25 print(parChecker('(()'))
```

```
True
False
```

PYTHON SKLIAUSTAI

```
{ { ( [ ] [ ] ) } ( ) }

[ [ { { ( ( ) ) } } ] ]

[ ] [ ] [ ] ( ) { }
```

```
( [ ) ]

( ( ( ) ] ) )

[ { ( ) ]
```

```
from pythonds.basic.stack import Stack

def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol in "([{":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                top = s.pop()
                if not matches(top,symbol):
                        balanced = False
        index = index + 1
    if balanced and s.isEmpty():
        return True
    else:
        return False

def matches(open,close):
    opens = "([{"
    closers = ")]}"
    return opens.index(open) == closers.index(close)


print(parChecker('{{([][])}()}'))
print(parChecker('[{()]'))
```

```
True
False
```

EILĖS (angl. QUENES)

Duomenų struktūra eilė - tai duomenų aibė, kuriai apibrėžtos tokios operacijos:

1. inicializuoti eilę (išskirti vietą eilei kompiuterio atmintyje);
2. įterpti tam tikrą elementą x į eilę (operacija put(x));
3. pašalinti elementą iš eilės (operacija get);
4. skaityti eilę;
5. panaikinti eilę (panaikinti vietą eilei kompiuterio atmintyje)

EILĖS

Duomenims eilėje taikomi loginiai apribojimai - operacijos get metu iš eilės bus pašalintas elementas, įterptas pirmasis (vadinamasis **FIFO** (First-In-First-Out) principas). Todėl eilė, kaip ir stekas, yra viena iš riboto išrinkimo klasės struktūrų.

EILĖS

Eilės struktūra taikoma įvairiose srityse: algoritmams realizuoti, programų loginėms schemoms, operacinių ir taikomųjų sistemų architektūroje, kompiliatoriams, loginių ir simbolinių skaičiavimų algoritmams, matematinio modeliavimo uždaviniams ir pan.

DEKAS (angl. DEQUEUE)

Duomenų struktūra dekas - tai duomenų aibė, kuriai apibrėžtos tokios operacijos:

- ✓ inicializuoti deką (išskirti vietą dekui kompiuterio atmintyje);
- ✓ įterpti tam tikrą elementą x į deko pradžią;
- ✓ įterpti tam tikrą elementą x į deko pabaigą;
- ✓ pašalinti elementą iš deko pradžios;

DEKAS (angl. DEQUEUE)

- ✓ pašalinti elementą iš deko pabaigos;
- ✓ skaityti deko pradžią;
- ✓ skaityti deko pabaigą;
- ✓ panaikinti deką (panaikinti vietą dekui kompiuterio atmintyje).

DEKAS (angl. DEQUEUE)

Duomenims **deke** taikomi loginiai apribojimai - operacijos pašalinimo iš deko pradžios ar pabaigos metu bus išmestas elementas, įterptas vėliausiai atitinkamai į pradžią ar į pabaigą. Todėl dekas, kaip ir eilė ar stekas, yra viena iš riboto išrinkimo klasės struktūrų.

DEKAS (angl. DEQUEUE)

Dekas dar vadinamas abipusiu steku (ar abipuse eile). Jo programavimas naudojant bazinį masyvo ar rodyklės tipą panašus į steko ar eilės programavimą, tik reikalauja sudėtingesnės realizacijos.

TIESINIAI SĄRAŠAI (angl. Linked lists)

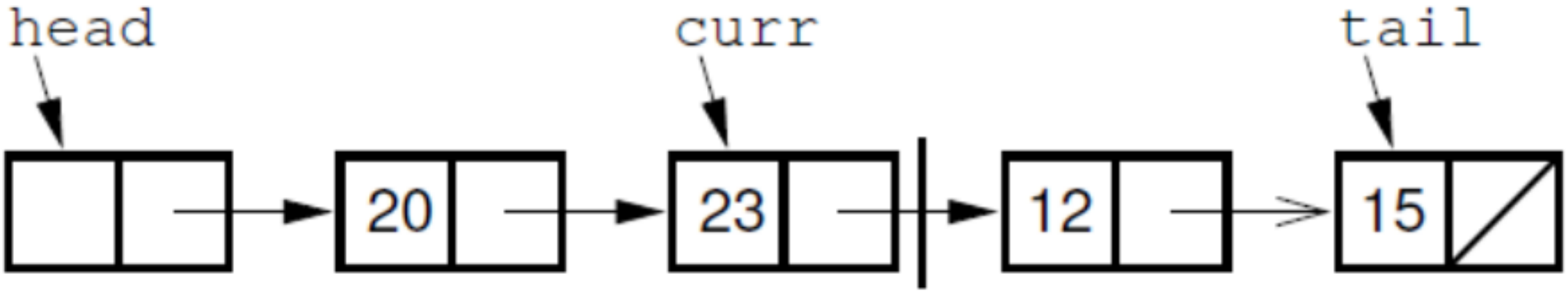
Duomenų struktūra tiesinis sąrašas - tai duomenų aibė, kurios kiekvienas elementas susideda iš dviejų dalių - informacinės dalies ir rodyklės - kuriai taikomos tokios operacijos:

- ✓ inicializuoti sąrašą; įterpti elementą x į sąrašo tam tikrą vietą;
- ✓ įterpti elementą x į sąrašą, tenkinant nurodytas sąlygas (pvz., didėjimo tvarka);
- ✓ išmesti iš sąrašo elementą, esantį tam tikroje vietoje;
- ✓ išmesti elementą x iš sąrašo, tenkinant nurodytas sąlygas;
- ✓ panaikinti sąrašą.

TIESINIAI SĄRAŠAI

Surūšiuotas sąrašas panašus į studentų sąrašą išdėstytą abėcėlės tvarka. Pridedant naują įrašą, jis talpinamas į reikiamą vietą, todėl sąrašas visada išlieka išrūšiuotas. Sąrašą sudaro elementai. Elementai susideda iš duomenų ir rodyklės. Sąrašo pradžios rodyklė turi pirmojo sąrašo elemento adresą, kuriuo pradedant galima nuosekliai pereiti per visus sąrašo elementus.

TIESINIAI SĄRAŠAI



SĄRAŠO TIPAI

Tiesinio dinaminio sąrašo tipai:

- ✓Vienkryptis tiesinis sąrašas;
- ✓Dvikryptis tiesinis sąrašas;
- ✓Vienkryptis ciklinis sąrašas;
- ✓Dvikryptis ciklinis sąrašas;

SĄRAŠO SAVYBĖS

Tiesinio dinaminio sąrašo savybės:

Visi elementai yra to paties duomenų tipo;

Kiekvienas elementas turi rodyklę, rodančią į vieną kaimyninį elementą;

Paskutinio sąrašo elemento rodyklė yra tuščia. Tai sąrašo pabaigos požymis.

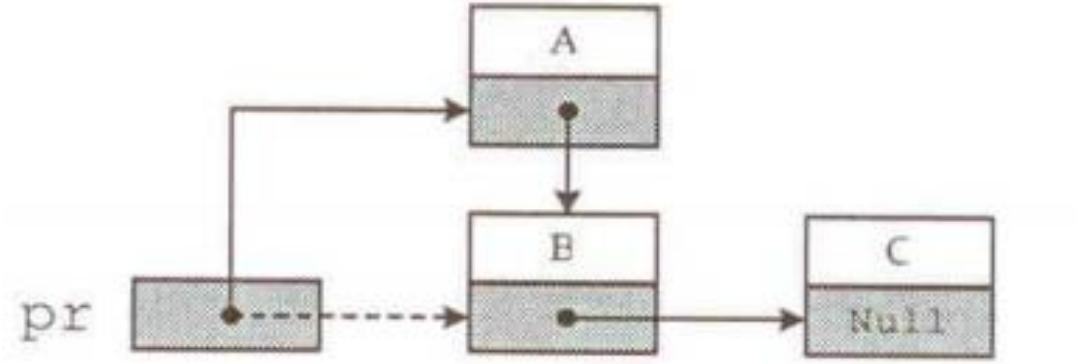
Elementai saugomi bet kurioje atminties vietoje;

Sąrašas yra laikomas tuščias, kai neturi nei vieno elemento.

Sąrašo ilgis – tai elementų skaičius sąraše.

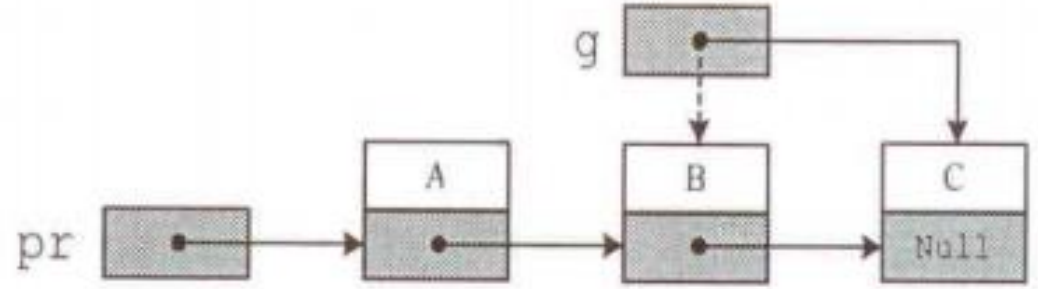
NAUJO SĄRAŠO SUDARYMAS

Naujas sąrašas gali būti sudaromas naudojant tiesioginės ir atvirkštinės kelties algoritmus.



Atvirkštinė keltis (naujas elementas į pabaigą)

pr – sąrašo pradžios rodyklė **g** – sąrašo galo rodyklė.



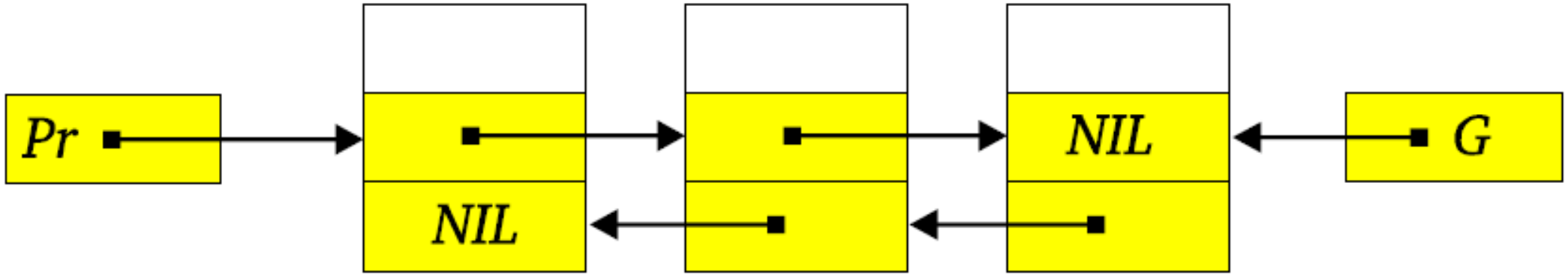
Tiesioginė keltis
(naujas elementas įrašomas į pradžią)

DVIKRYPTIS TIESINIS SĄRAŠAS

Dvikryptis dinaminis sąrašas nuo paprasto skiriasi tuo, kad kiekvienas elementas turi antrą rodyklę, rodančią prieš jį esantį elementą.

Į dvikryptį sąrašą galima žiūrėti kaip į du paprastus, priešingomis kryptimis „sujungtus“ sąrašus:

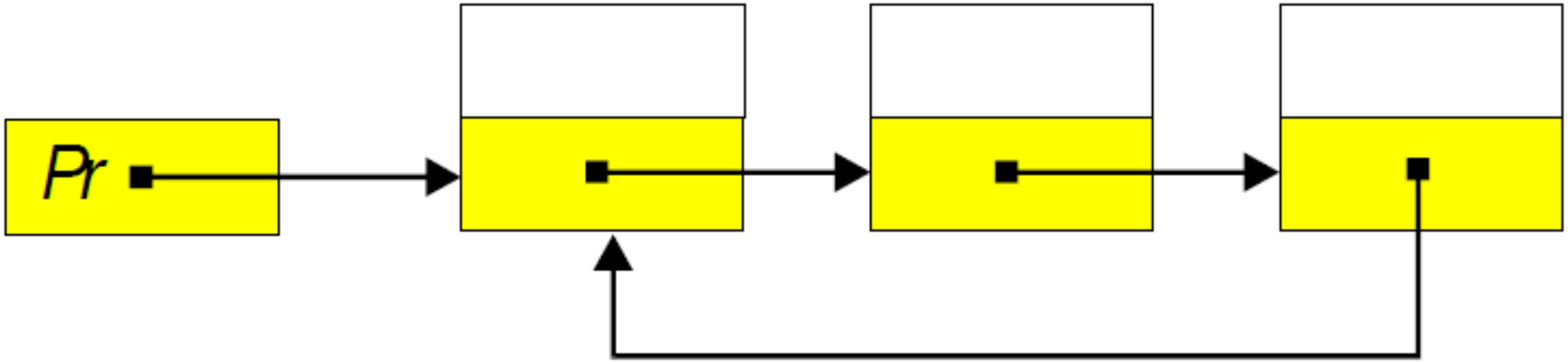
DVIKRYPTIS TIESINIS SĄRAŠAS



CIKLINIS SĄRAŠAS

Tai uždaro kelio ciklinį sąrašas, kuriame paskutinis elementas yra gretimas pirmajam. Šio sąrašo elemente nėra sąrašo galo požymio. Todėl skaičiavimų pabaiga nustatoma lyginant tarpinę rodyklę P su sąrašo pradžios rodykle Pr.

CIKLINIS SĄRAŠAS



DVIKRYTIS CIKLINIS SĄRAŠAS

Kiekvienas dvikrypčio sąrašo elementas turi dvi rodykles: kitas ir ankstesnis.

Dvikrypčio ciklinio sąrašo paskutinio elemento rodyklė **kitas** rodo pirmąjį, o pirmojo rodyklė **ankstesnis** – paskutinį elementą.

DVIKRYTIS CIKLINIS SĄRAŠAS

Veiksmai su dvikrypčio ciklinio sąrašo elementais su paprastėja panaudojus fiktyvų pradžios elementą.

Tuo atveju sąrašo pradžią nusako jo rodyklė kitas, o galą – rodyklė ankstesnis.

Fiktyvus elementas leidžia supaprastinti įrašymo į sąrašą ir šalinimo iš jo operacijas.

DVIKRYTIS CIKLINIS SĄRAŠAS