
Структури от данни и програмиране

Лекция 5
Втора част

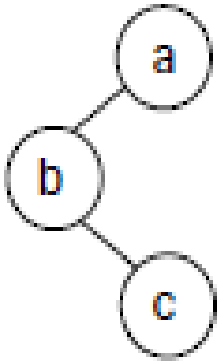
# Двоично дърво

# Дефиниция на двоично дърво

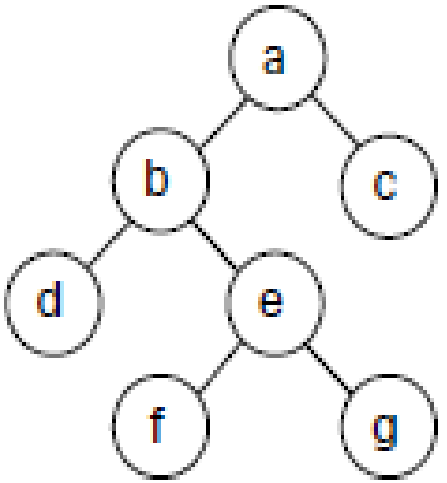
- Празното двоично дърво е двоично дърво
- Ако L и R са двоични дървета, а X е данна, то (X, L, R) е двоично дърво с
  - корен X
  - ляво поддърво L
  - дясно поддърво R

# Примери

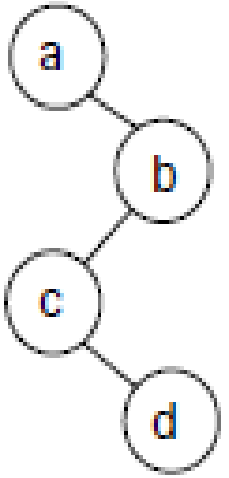
а)




б)

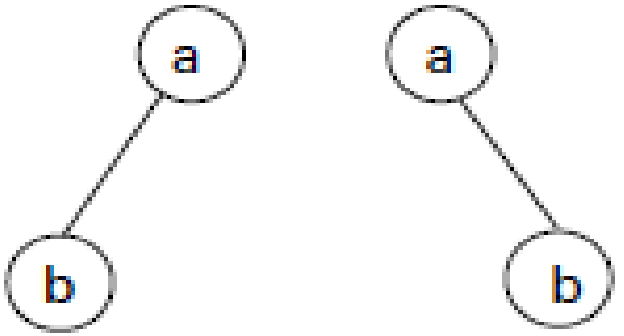



в)



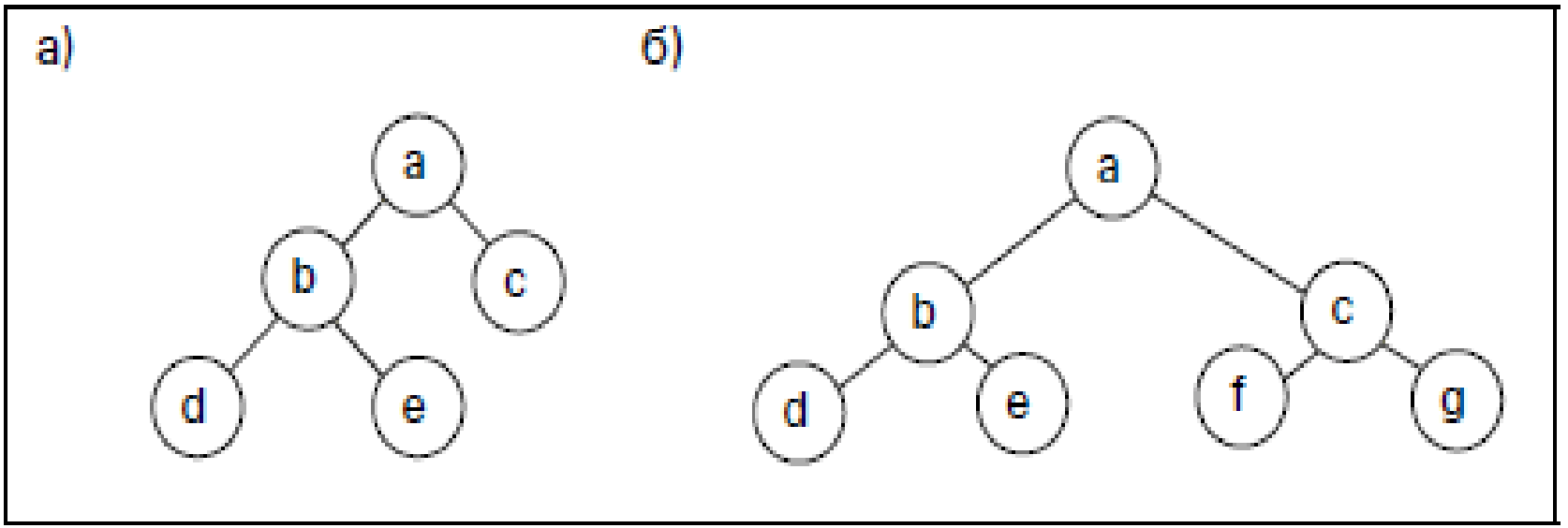
- Може ли двоичното дърво винаги да се разглежда като частен случай на дървото от предишната лекция?

- Не. Следните две двоични дървета са различни:



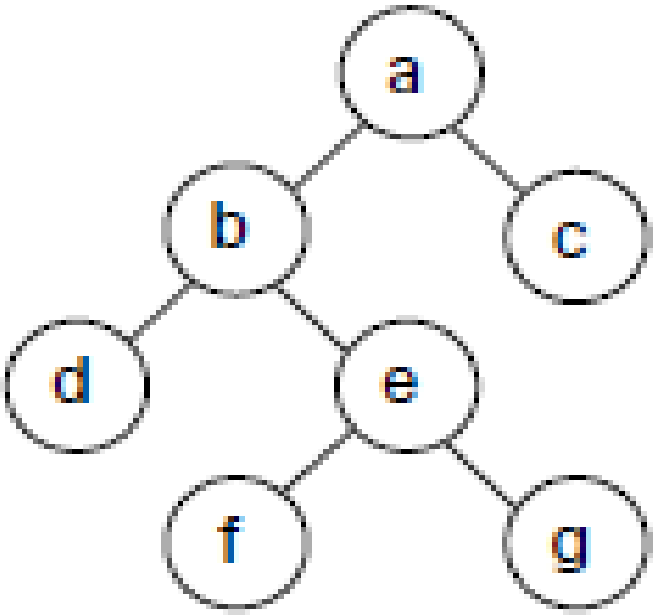
# Още дефиниции

- Двоично дърво, в което всеки възел, който не е листо, има точно два наследника, се нарича *строго*
- Ако всички листа на строго двоично дърво са на едно и също ниво, двоичното дърво се нарича *пълно*
  - Колко са възлите, ако височината е *h*?



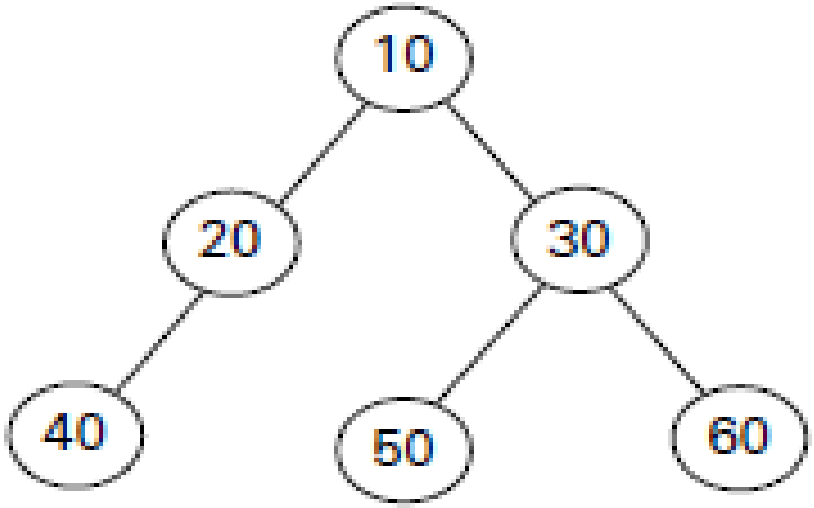
# Обхождане на двоично дърво

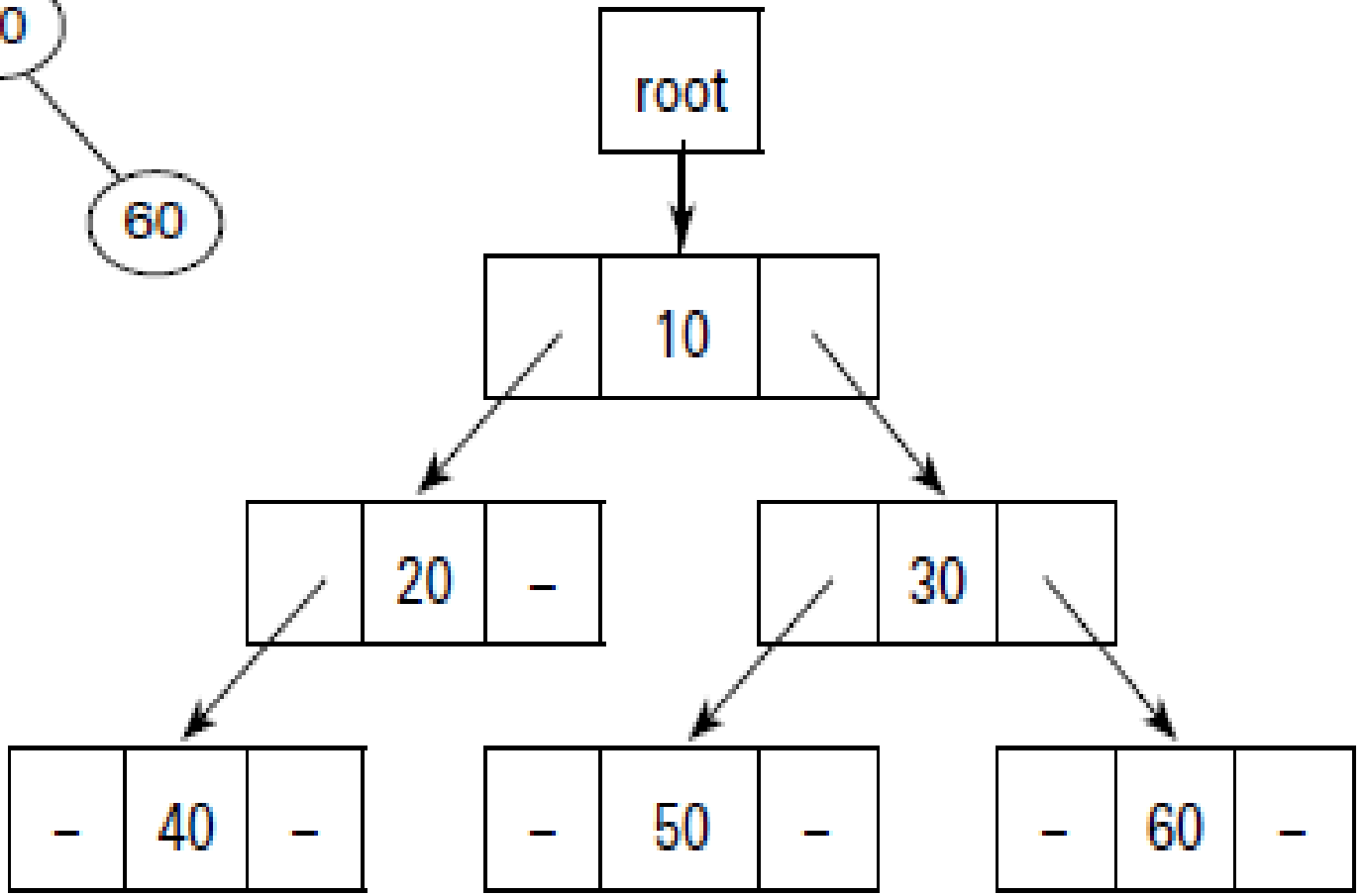
- ЛКД (ляво-корен-дясно)
- КЛД (корен-ляво-дясно)
- ЛДК (ляво-дясно-корен)
- ДКЛ
- КДЛ
- ДЛК



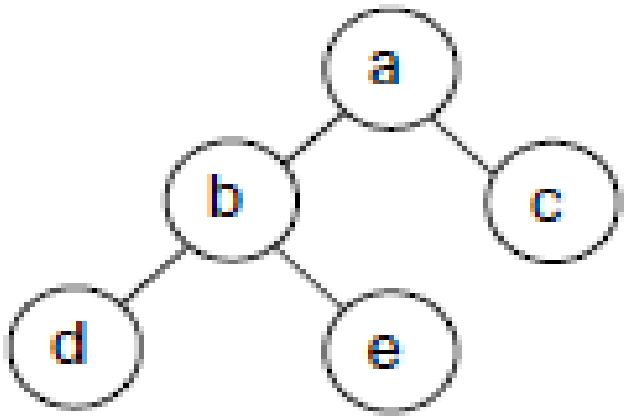

| | |
|---|---|
| ЛКД: dbfegac | ДКЛ: cagefbd |
| КЛД: abdefgc | КДЛ: acbegfd |
| ЛДК: dfgebca | ДЛК: cgfedba |

# Свързано представяне



# Последователно представяне




| | върхове | | ЛПД | | ДПД |
|---|---|---|---|---|---|
| 0 | d | 0 | -1 | 0 | -1 |
| 1 | b | 1 | 0 | 1 | 4 |
| 2 | c | 2 | -1 | 2 | -1 |
| 3 | a | 3 | 1 | 3 | 2 |
| 4 | e | 4 | -1 | 4 | -1 |
| 5 | ... | 5 | ... | 5 | ... |

# Реализация на свързаното представяне

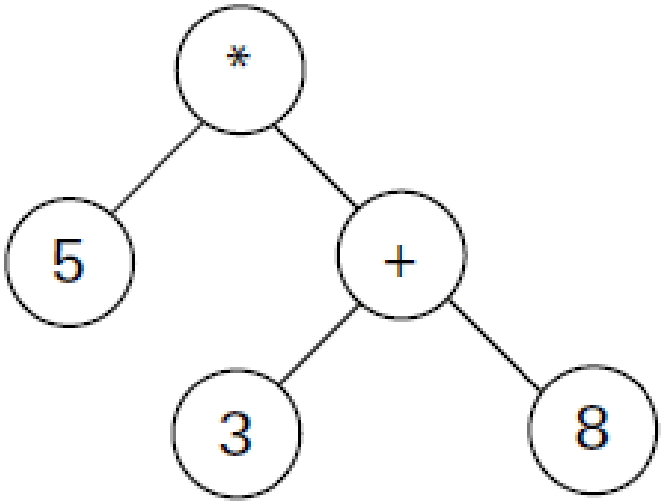
- `tree.h` – абстрактен базов клас на дърво и абстрактен базов клас на итератор върху дърво
- `tree.cpp` – конкретни наследници на двата класа

# Задачи върху двоично дърво

- Дълбочина (височина)
  - 1 + по-голямата от височините на ЛПД и ДПД
- Равенство на две двоични дървета

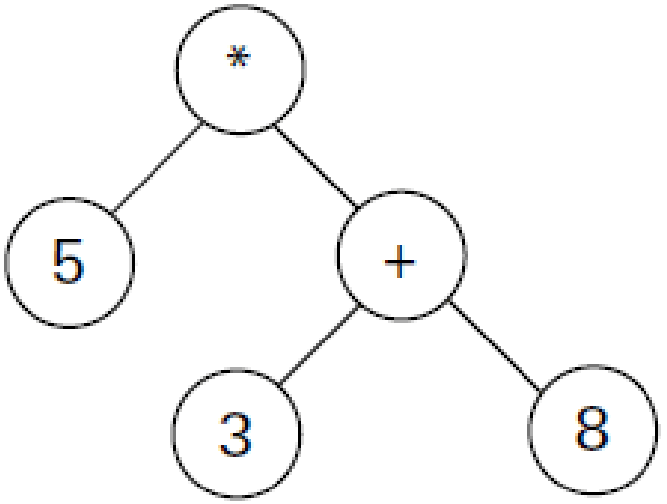
# Приложение: expression trees

- Числата и променливите (терминалите) се представят с листа
- Пример: 5 * (3 + 8) →



# Генериране на различните видове запис на израз

- Ляво-корен-дясно (ЛКД) – инфиксен запис
  (5 * (3 + 8))
- Ляво-дясно-корен (ЛДК) – обратен полски запис
  538+*
- Корен-ляво-дясно (КЛД) – прав полски запис
  *5+38



# Основни задачи: генериране и изчисляване

- За изрази, в които няма липсващи скоби, т.е.

```
<expr> ::= 0-9 | (<expr><op><expr>)
<op>   ::= + | - | * | /
```

  – във файла с примери

- Идеи за развитие:
  - За изрази, в които операциите имат различен приоритет, може да се използва алгоритъмът за преобразуване на израз в обратен полски запис (от лекцията за стек)
  - Поддръжка на многоцифрени числа и променливи – дървото няма да е от тип `char`

# Обобщение