Управляващи оператори в C++

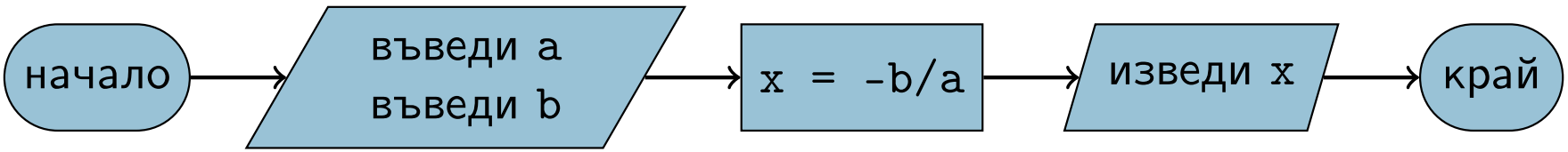
Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток, 2018/19 г.

18 октомври 2018 г.

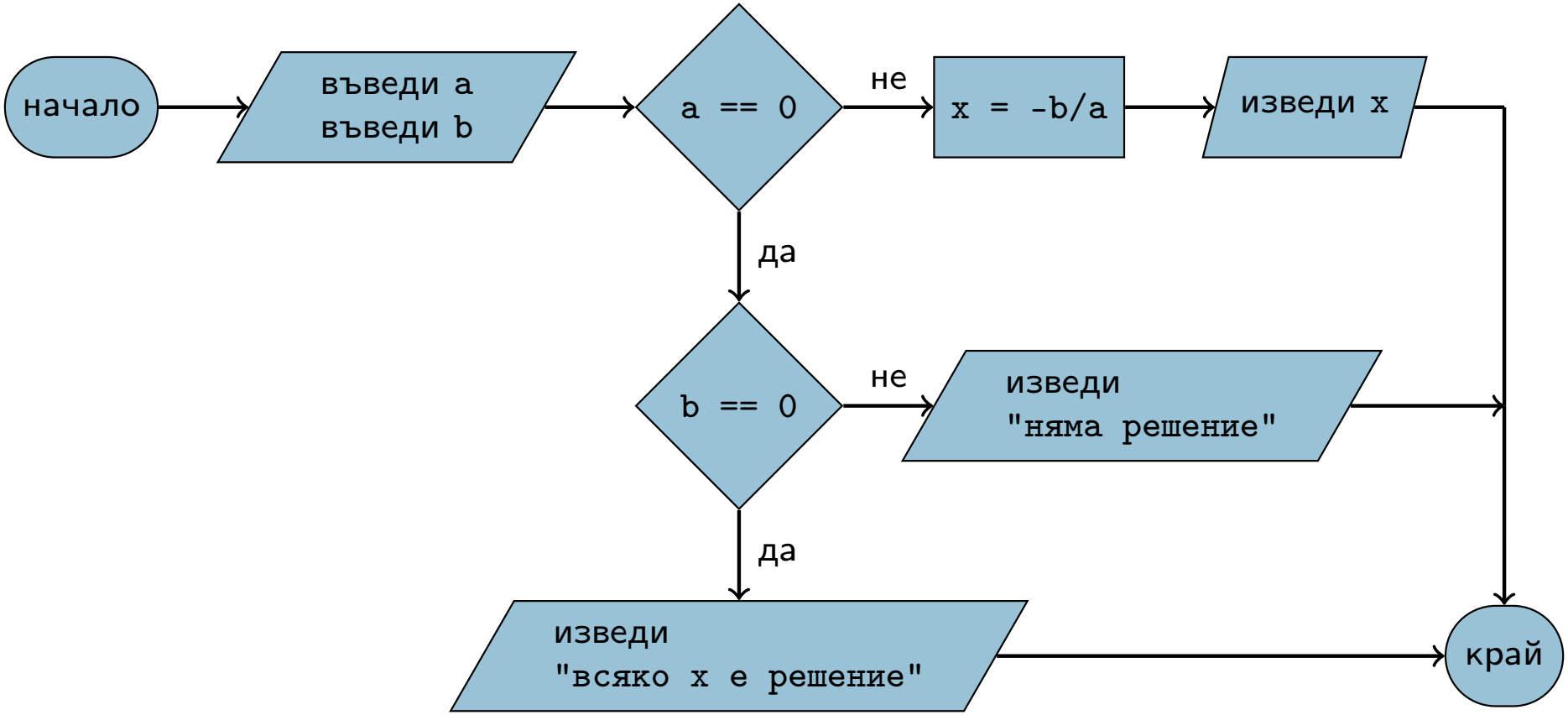
Изчислителни процеси

- Алгоритъм: последователност от стъпки за извършване на пресмятане
- Блок-схема

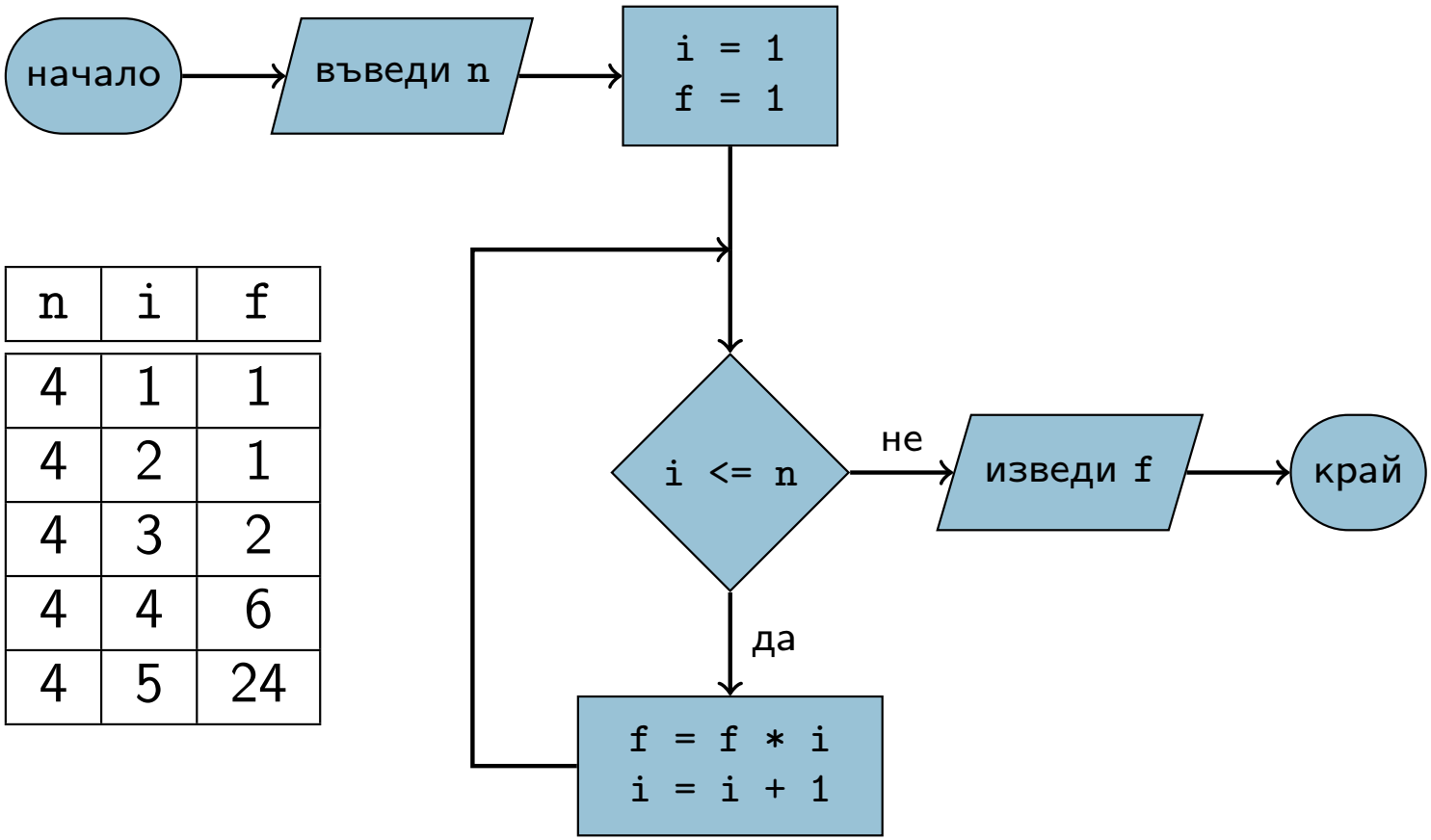


Пример за линеен процес

Разклоняващи се процеси

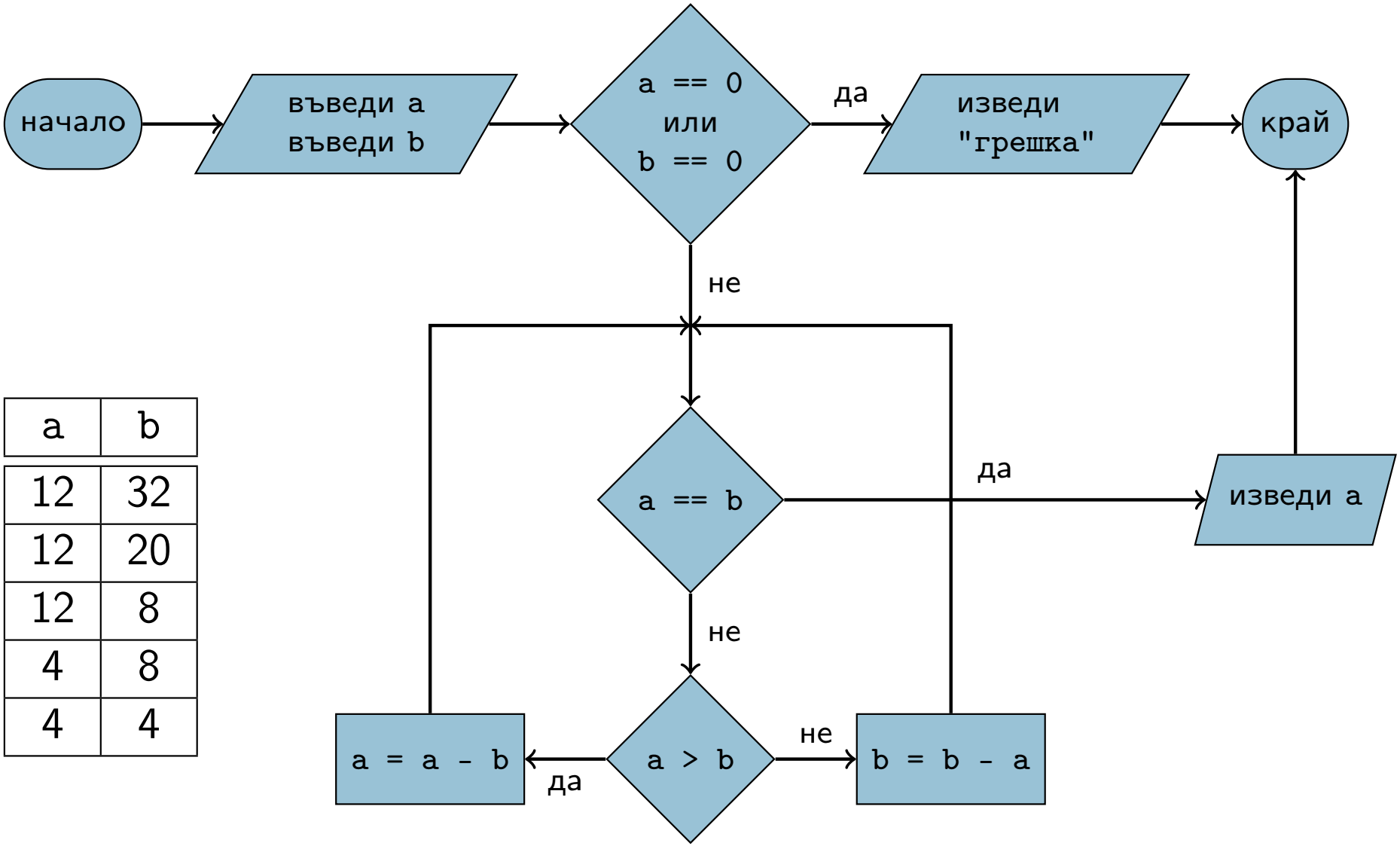


Индуктивни циклични процеси



n	i	f
4	1	1
4	2	1
4	3	2
4	4	6
4	5	24

Итеративни циклични процеси



a	b
12	32
12	20
12	8
4	8
4	4

Структурни езици — разклонение

1. Въведи `a`, `b`
2. Ако `a == 0`, към 5
3. `x = -b / a`
4. Премини към 9
5. Ако `b = 0`, към 8
6. "Няма решения"
7. Премини към 9
8. "Всяко x е решение"
9. Край

- Въведи `a`, `b`
- Ако `a == 0`
  - Ако `b == 0`
    - "Всяко x е решение"
  - Иначе
    - "Няма решения"
- Иначе
  - `x = -b / a`

Структурни езици — индуктивен цикъл

1. Въведи `n`
2. `i = 1`
3. `f = 1`
4. Ако `i > n`, към 8
5. `f = f * i`
6. `i = i + 1`
7. Премини към 4
8. Изведи `f`
9. Край

- Въведи `n`
- `i = 1`
- `f = 1`
- Повтаряй `n` пъти
  - `f = f * i`
  - `i = i + 1`
- Изведи `f`

Структурни езици — итеративен цикъл

1. Въведи a, b
2. Ако a == b, към 6.
3. Ако a > b, към 5.
4. b = b - a; към 2.
5. a = a - b; към 2.
6. Изведи a
7. Край

- Въведи a, b
- Докато a != b
  - Ако a > b
    - a = a - b
  - В противен случай
    - b = b - a
- Изведи a

Основни понятия

- **Операция** (operator)
- **Израз** (expression)
- **Оператор/команда** (statement)
- <израз> ::= <константа> | <променлива> |
  <едноместна_операция> <израз> |
  <израз> <двуместна_операция> <израз>
- <оператор> ::= <израз>;

Оператор за присвояване

- <променлива> = <израз>;
- <lvalue> = <rvalue>;
- <lvalue> — място в паметта със стойност, която може да се променя
  - **Пример:** променлива
- <rvalue> — временна стойност, без специално място в паметта
  - **Пример:** константа, литерал, резултат от пресмятане
- стандартно преобразуване на типовете:
  <rvalue> се преобразува до типа на <lvalue>

Присвояването като операция

- дясноасоциативна операция
- `a = (b = (c = 2));`
- ~~`(((a = b) = c) = 2);`~~
- **Пример:** `cout << x + (b = 2);`
- **Пример:** `(a = b) = a + 3;`

Операция за изброяване

- <израз1>, <израз2>
- оценява и двата израза, но крайният резултат е оценката на втория израз
- `a, b, c, d` $\Leftrightarrow$ `(a, (b, (c, d)))`
- дясноасоциативна
- използва се рядко
- **Пример:** `a = (cout << x, x);`

Съкратени оператори за присвояване

- `a = a + 2` ⇔ `a += 2`
- `-=`, `*=`, `/=`, `%=`
- `a = a + 1` ⇔ `++a`
- `a = a - 1` ⇔ `--a`
- `a++` увеличава `a` с 1, но връща предишната стойност на `a`
  - `a++` ⇔ `(a = (tmp = a) + 1, tmp)`
- `a--` действа аналогично
- `++a` връща `a`, което е <lvalue>
  - **Пример:** `++a += 5;`
- `a++` връща предишната стойност на `a`, което е <rvalue>
  - **Пример:** `x = a++ * b;` ~~`a++ += 5;`~~

Оператор за блок

- { { <оператор> } }
- { <оператор$_1$> <оператор$_2$> . . . <оператор$_n$> }
- Вложени блокове

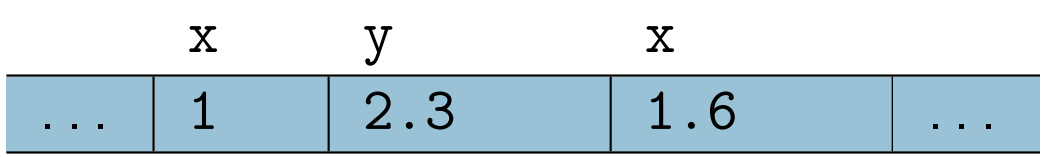
```
{
  int x = 2;
  {
    x += 2;
    cout << x;
  }
}
```

Област на действие (scope)

- областта на действие се простира от дефиницията на променливата до края на блока, в който е дефинирана
- дефиниция на променлива със същото име в същия блок е забранена
- дефиниция на променлива във вложен блок покрива всички външни дефиниции със същото име

Област на действие (scope) — пример

```
int x = 0;
{
  x++;
  double y = 2.3;
  {
    double x = 1.6;
    y = x * x;
  }
  double y = 2.4;
  x += 3;
}
x += 4;
y /= 2.1;
```



Празен оператор

- ;
- ; $\Leftrightarrow$ {}
- няма никакъв ефект

Условен оператор

- `if (`<израз>`)` <оператор> [`else` <оператор>]
- Съкратената форма $\Leftrightarrow$ пълна форма с празен оператор
  - `if (A) X;` $\Leftrightarrow$ `if (A) X; else;`
- **Пример:** `if ( x < 2 ) y = 2;`
- **Пример:** `if ( x > 5 ) y = 5; else y = 3;`

Вложени условни оператори

Какво имаме предвид, когато пишем:

```
if (a > 0) if (b > 0) cout << 1; else cout << 3;
```

```
if (a > 0) {
  if (b > 0)
    // a > 0 && b > 0
    cout << 1;
  else
    // a > 0 && b ≤ 0
    cout << 3;
}
```

или

```
if (a > 0) {
  if (b > 0)
    // a > 0 && b > 0
    cout << 1;
}
else
  // a ≤ 0
  cout << 3;
```

Съкратено оценяване на логически операции

Представяне на логически операции с вложени условни оператори:

```
if (!A) X;
else    Y;
```
⇔
```
if (A) Y;
else   X;
```

```
if (A && B) X;
else        Y;
```
⇔
```
if (A)
  if (B) X;
  else   Y;
else Y;
```

```
if (A || B) X;
else        Y;
```
⇔
```
if (A) X;
else
  if (B) X;
  else   Y;
```

Пример: `if (x > 0 && log(x) < 5) ...`
Пример: `if (x == 0 || y / x == 1) ...`