Algoritmizace diskrétních simulačních modelů

Úvodní poznámky

- Při programování simulačních modelů lze hlavní dílčí problémy shrnout do následujících bodů:
  1) Zachycení statických vlastností (struktury modelu).
  2) **Zachycení dynamických vlastností modelu**.
  3) Problematika generování náhodných proměnných vyskytujících se v modelu.

Úvodní poznámky

4) Realizace sběru potřebných údajů (výstupů) z chodu simulačního modelu.
5) Návrh a vyhodnocení experimentů s modelem.
6) Zpracování a výstup simulačních výsledků.
7) Kontrolní, monitorovací funkce.

- V této prezentaci se budeme zabývat problematikou zachycení dynamických vlastností.

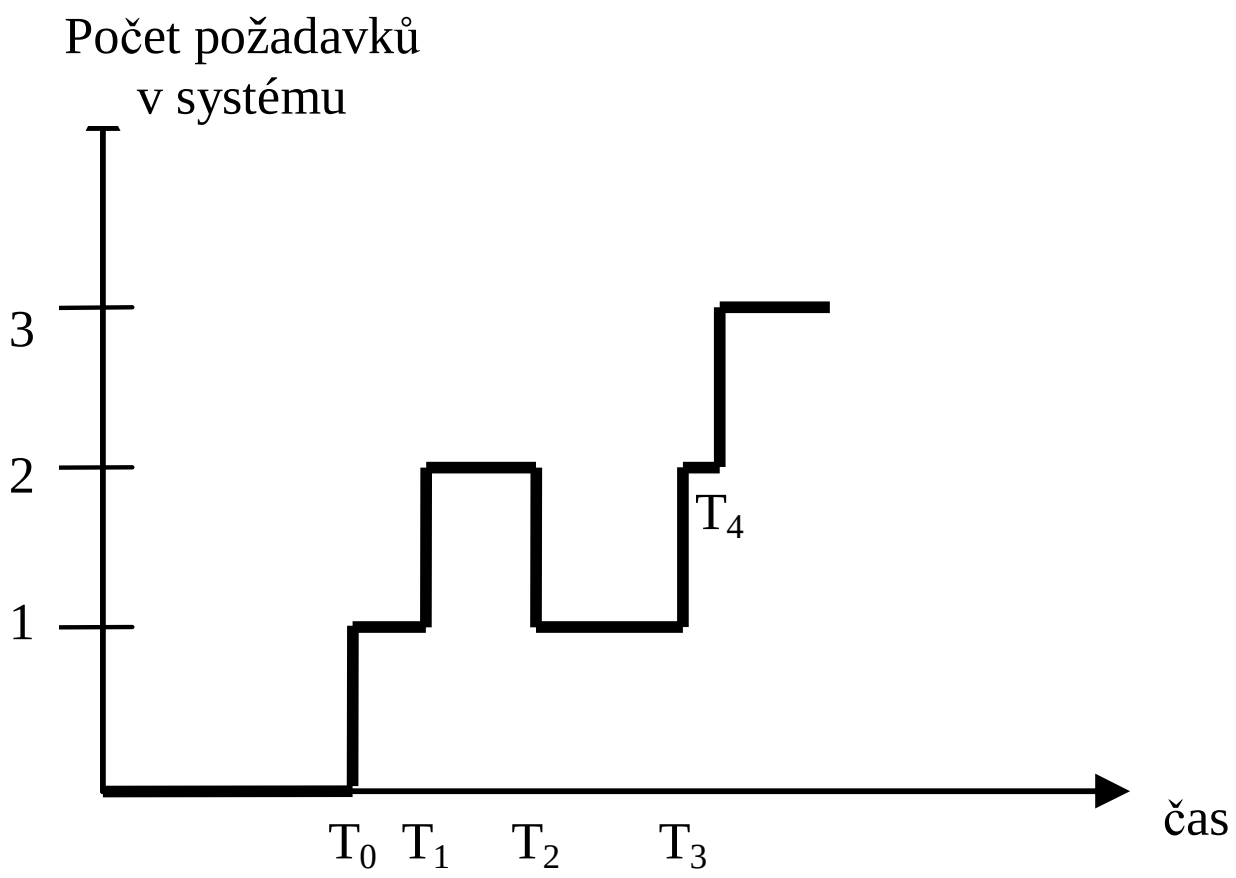
Úvodní poznámky

- Při simulaci je třeba rozlišovat:
  - **Reálný čas** – čas, ve kterém probíhá skutečný děj v reálném systému.
  - **Simulární čas** – modelový (vnitřní) čas simulačního modelu. Tento čas může běžet mnohonásobně rychleji (nebo i pomaleji) než reálný čas. Jedná se tedy o fiktivní čas, který nemusí běžet plynule a rovnoměrně.
  - **Strojový čas** – čas, který je spotřebovaný na výpočet programu, závisí na složitosti modelu a nezávisí se simulárním časem.

Úvodní poznámky

- O **diskrétní simulaci** hovoříme tehdy, mění-li se stavové proměnné diskrétně (skokově) v určitých okamžicích simulárního času – v okamžicích výskytů **událostí**. Tyto okamžiky někdy nazýváme jako **přechodové okamžiky**. Stav systému se mezi dvěma přechodovými okamžiky nemění.

Úvodní poznámky

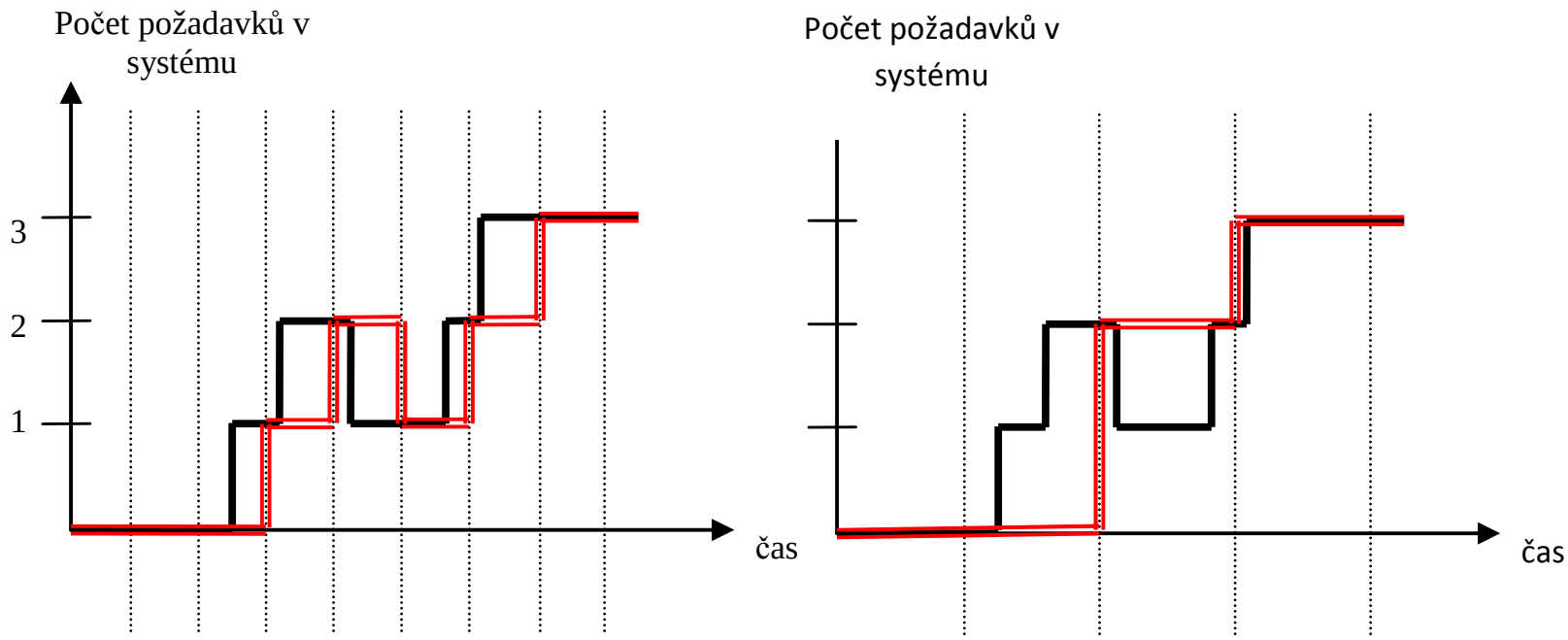


Úvodní poznámky

- Problémem je, jak zachytit dynamické změny v modelu systému. Obecně máme dvě možnosti a to:
  1) **Pevný (konstantní) časový krok** – Δt = konst.
  2) **Proměnlivý časový krok** – $\Delta t \neq$ konst.

Pevný časový krok

- Pevný časový krok je nejjednodušším způsobem organizace časoví struktury simulačního modelu.



Pevný časový krok

- Problémem je však stanovit, jak velký časový krok Δt stanovit.
  - Bude-li příliš malý, zvýší se zbytečně objem výpočetních operací.
  - Nevhodně velký časový krok zase způsobuje vznik více událostí v jednom časovém okamžiku (shlukování), příp. nezaregistrování některých událostí.
- Proto je vhodnější použít proměnlivý časový krok.

Proměnlivý časový krok

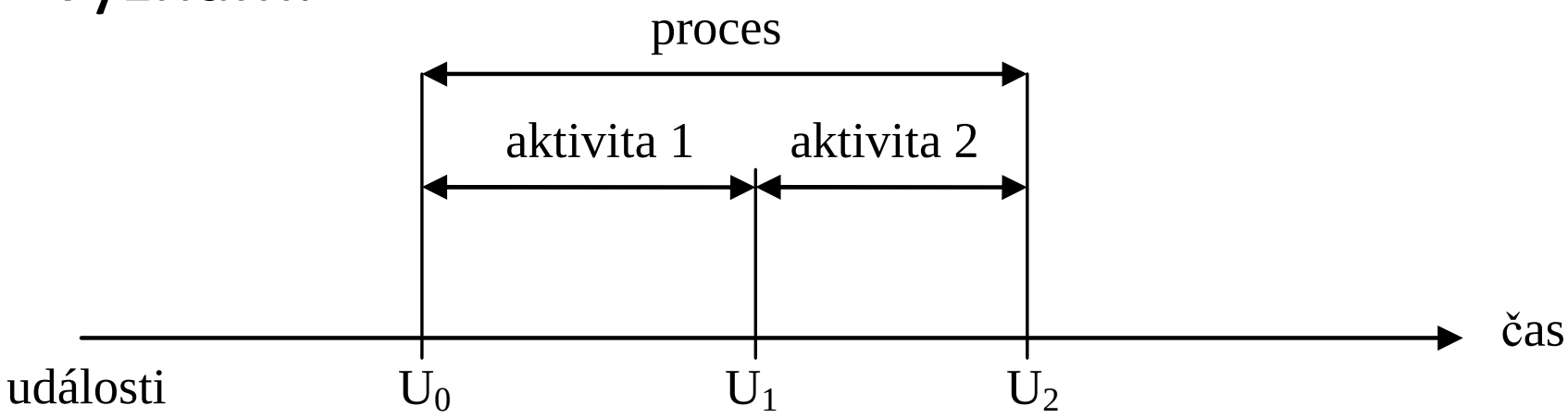
- Zaveďme si nejdříve pojmy, které budeme používat:
  - **Okamžik (přechodový okamžik)** – okamžikem rozumíme takovou hodnotu simularního času, v níž dochází ke změně alespoň jedné stavové proměnné simulačního modelu (dochází tedy ke změně stavu modelu).
  - **Interval** – Intervalem označujeme časový úsek mezi dvěma po sobě jdoucími okamžiky.

Proměnlivý časový krok

- **Časové rozpětí** představuje několik na sebe navazujících intervalů.
- **Událost** – změna stavu modelu, která nastává v okamžiku, ve kterém začíná aktivita.
- **Aktivitou** označujeme stav modelu mezi dvěma událostmi popisujícími po sobě jdoucí změny stavu modelu.
- **Proces** je posloupnost stavů modelu v časovém rozpětí.

Proměnlivý časový krok

- Uvažujme jednolinkový systém hromadné obsluhy bez odmítání. Výše uvedené pojmy mohou mít v tomto případě následující význam.



Proměnlivý časový krok

- Událost U_0 – příchod *k*-tého požadavku do systému.
- Událost U_1 – zahájení obsluhy *k*-tého požadavku.
- Událost U_2 – ukončení obsluhy *k*-tého požadavku.
- Aktivita 1 – čekání *k*-tého požadavku ve frontě.
- Aktivita 2 – obsluha *k*-tého požadavku obslužnou linkou.
- Proces popisuje, co se s *k*-tým požadavkem dělo od okamžiku jeho příchodu do systému do okamžiku jeho odchodu (ukončení obsluhy) ze systému.

Proměnlivý časový krok

- V případě proměnlivého časové kroku rozeznáváme:
  1) **Algoritmy orientované na události**.
  2) **Algoritmy orientované na aktivity**.
  3) **Algoritmy orientované na procesy**.

- Simulační software Witness používá první přístup, tedy orientaci na události.

Algoritmy orientované na události

- Zaveďme množinu U jako množinu všech tříd U_i událostí, jejichž výskyt má za následek změnu stavu modelu, tedy:

  $$U = \{U_1; U_2; \ldots; U_n\}.$$

  Výskyt libovolné události z jedné třídy má za následek stejný charakter změny stavu modelu (lze jej tedy popsat stejným úsekem programu).

Algoritmy orientované na události

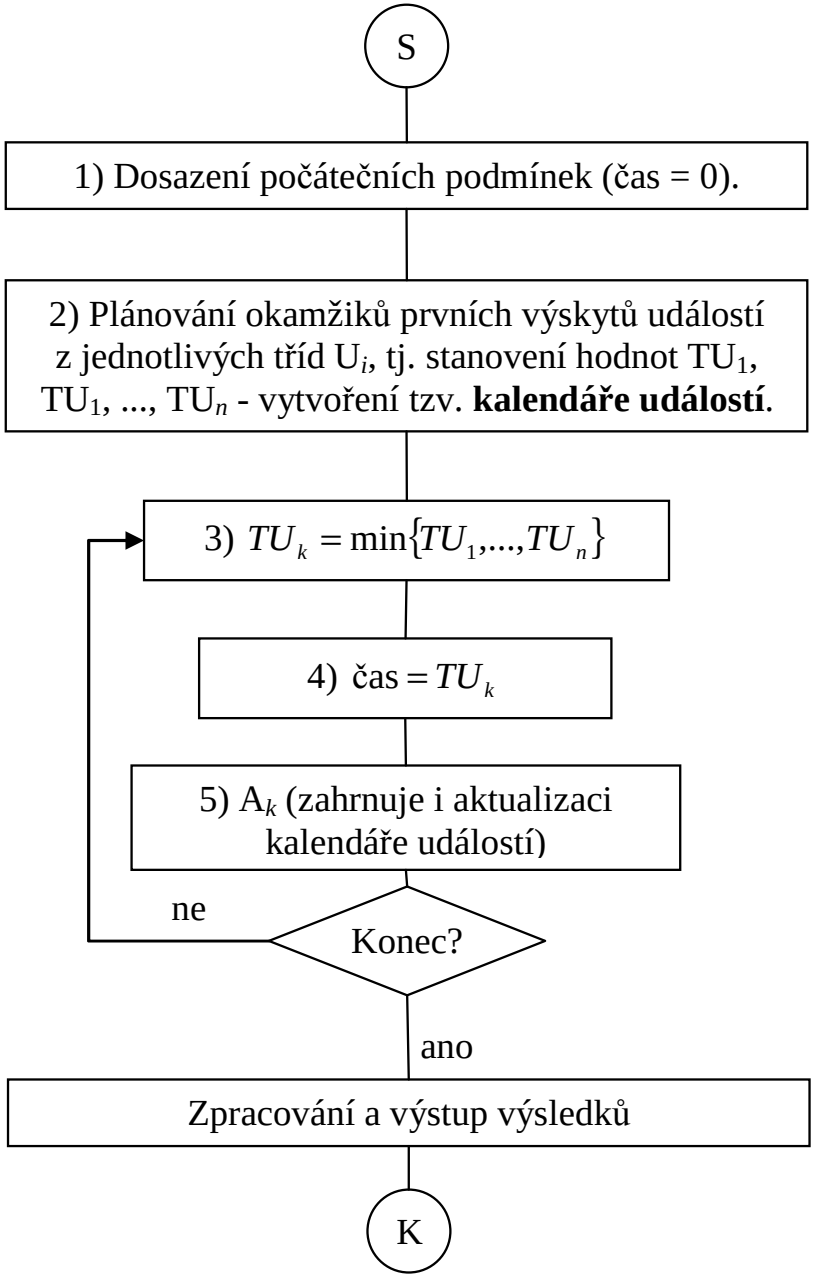
- Např. v jednolinkém systému hromadné obsluhy bez odmítání můžeme příchody jednotlivých požadavků zařadit do jedné třídy událostí. Obdobně ukončení obsluhy jednotlivých požadavků mohou být zařazena do jedné třídy událostí.
- Označme TU_i okamžik příštího (vzhledem k aktuální hodnotě simulárního času) výskytu události z třídy U_i.

Algoritmy orientované na události

- Symbolem M budeme označovat hodnotu, která je větší než hodnota simulárního času odpovídajícího výskytu libovolné události. Nebudeme-li schopni určit okamžik příštího výskytu události z třídy U_i,, potom položíme TU_i = M (tzn. Odložíme výskyt události této třídy na neurčito).

Algoritmy orientované na události

- Uvažujme, že důsledky výskytu libovolné události z třídy U_i můžeme popsat algoritmem A_i, kde $i = 1, 2, \ldots, n$.
- Označením čas budeme rozumět stavovou proměnnou modelu uchovávající aktuální hodnotu simulárního času.
- Algoritmus orientovaný na události můžeme popsat následujícím vývojovým diagramem.


S
1) Dosazení počátečních podmínek (čas = 0).
2) Plánování okamžiků prvních výskytů událostí z jednotlivých tříd U_i, tj. stanovení hodnot TU_1, TU_1, ..., TU_n - vytvoření tzv. **kalendáře událostí**.
3) $TU_k = \min\{TU_1,...,TU_n\}$
4) čas $= TU_k$
5) A_k (zahrnuje i aktualizaci kalendáře událostí)
ne
Konec?
ano
Zpracování a výstup výsledků
K

Algoritmy orientované na události

- ad 1) Dosazení počátečních podmínek ve výchozím okamžiku simulárního času.
- ad 2) Plánování okamžiků, kdy má dojít k prvním výskytům událostí z jednotlivých tříd. V případě, že nejsme schopni určit okamžik výskytu události z některé třídy (výskyt události je např. vázán na výskyt jiné události, která ještě nenastala), potom bude tato hodnota rovna M.

Algoritmy orientované na události

- ad 3) Výběr okamžiku výskytu nejbližší události. V případě, že může dojít k výskytu více událostí pro danou hodnotu simulárního času, vybíráme-li libovolnou z nich (nejsou-li stanoveny priority událostí).
- ad 4) Skoková změna simulárního času na hodnotu TU_k.
- ad 5) Realizace změn v modelu, které jsou důsledkem nastátí události z třídy U_k. Součástí tohoto kroku je i aktualizace kalendáře událostí.

Algoritmy orientované na události

- **Př. 1:** Uvažujme jednolinkový systém hromadné obsluhy s řádným frontovým režimem a nekonečnou délkou fronty. Nechť je doba mezi příchody po sobě jdoucích požadavků náhodná proměnná X, doba obsluhy požadavku náhodná proměnná Y. Nakreslete vývojový diagram zachycující modelující algoritmus tohoto systému orientovaný na události.

Algoritmy orientované na události

- Zaveďme si vnitřní (stavové) proměnné modelu:
  - Proměnná F bude vyjadřovat aktuální počet požadavků ve frontě.
  - Proměnná S bude popisovat stav linky (0 – linka nepracuje, 1 – linka provádí obsluhu požadavku).
- V tomto systému hromadné obsluhy lze rozlišit tři třídy událostí:
  1) Příchod požadavku do systému.
  2) Zahájení obsluhy požadavku.
  3) Ukončení obsluhy požadavku.

Algoritmy orientované na události

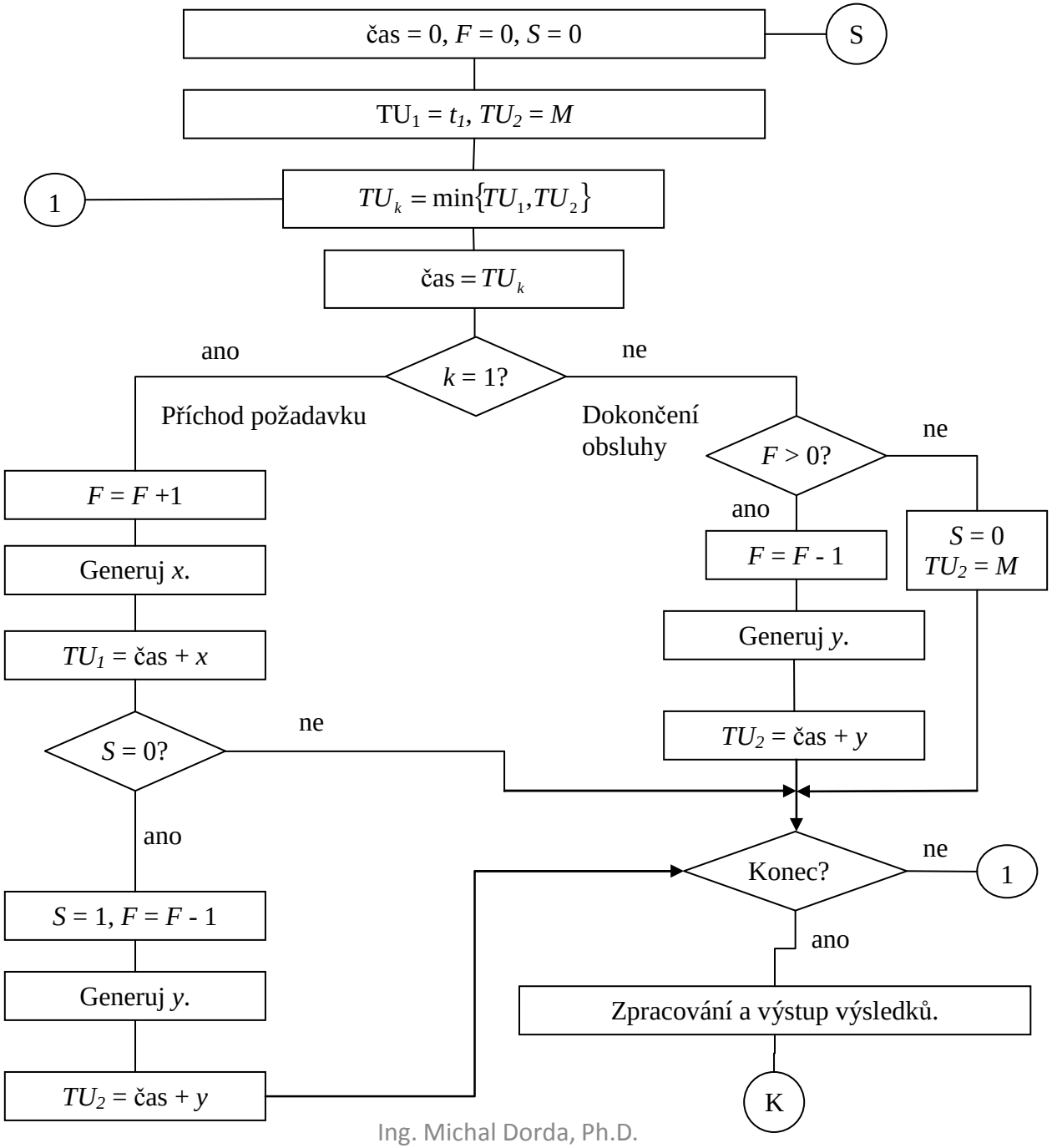
- Jelikož k zahájení obsluhy dojde tehdy, je-li ve frontě alespoň jeden požadavek (tedy $F \geq 1$) a je-li linka volná (tedy $S = 0$), je tato událost podmíněna stavem systému , nemusíme tedy tuto třídu událostí uvažovat.
- Budeme tedy pracovat pouze se dvěma třídami událostí – U_1 – příchod požadavku a U_2 – ukončení obsluhy požadavku.

Algoritmy orientované na události

- Linka může v každém okamžiku obsluhovat nejvýše jeden požadavek.
- Obsluha požadavku může začít tehdy, je-li splněna podmínka $(F > 0) \wedge (S = 0)$.
- Dále je třeba definovat počáteční podmínky, nechť je systém v okamžiku čas = 0 prázdný, tedy F = 0 a S =0.

Algoritmy orientované na události

- První požadavek přijde v okamžiku t_1, takže okamžik prvního výskytu události z třídy událostí U_1 bude $TU_1 = t_1$.
- Jelikož v okamžiku čas = 0 žádná obsluha neprobíhá, nejsme tedy schopni naplánovat první okamžik výskytu události z třídy událostí U_2, tudíž $TU_2 = M$.
- Vývojový diagram je nakreslen na následujícím obrázku.


čas = 0, F = 0, S = 0
S
TU1 = t1, TU2 = M
1
TUk = min{TU1,TU2}
čas = TUk
ano
k = 1?
ne
Příchod požadavku
Dokončení obsluhy
F > 0?
ne
F = F +1
ano
F = F - 1
S = 0
TU2 = M
Generuj x.
Generuj y.
TU1 = čas + x
TU2 = čas + y
S = 0?
ne
ano
Konec?
ne
1
S = 1, F = F - 1
ano
Generuj y.
Zpracování a výstup výsledků.
TU2 = čas + y
K

Algoritmy orientované na události

- Zkusme si nyní napodobit funkci algoritmu bez použití počítače. Mějme vygenerovány hodnoty náhodné proměnné X a Y.

x	2	3	6	8	2	4
y	6	2	3	3	10	2

- Uvažujme dále, že první požadavek přijde v čase $t_1 = 2$.

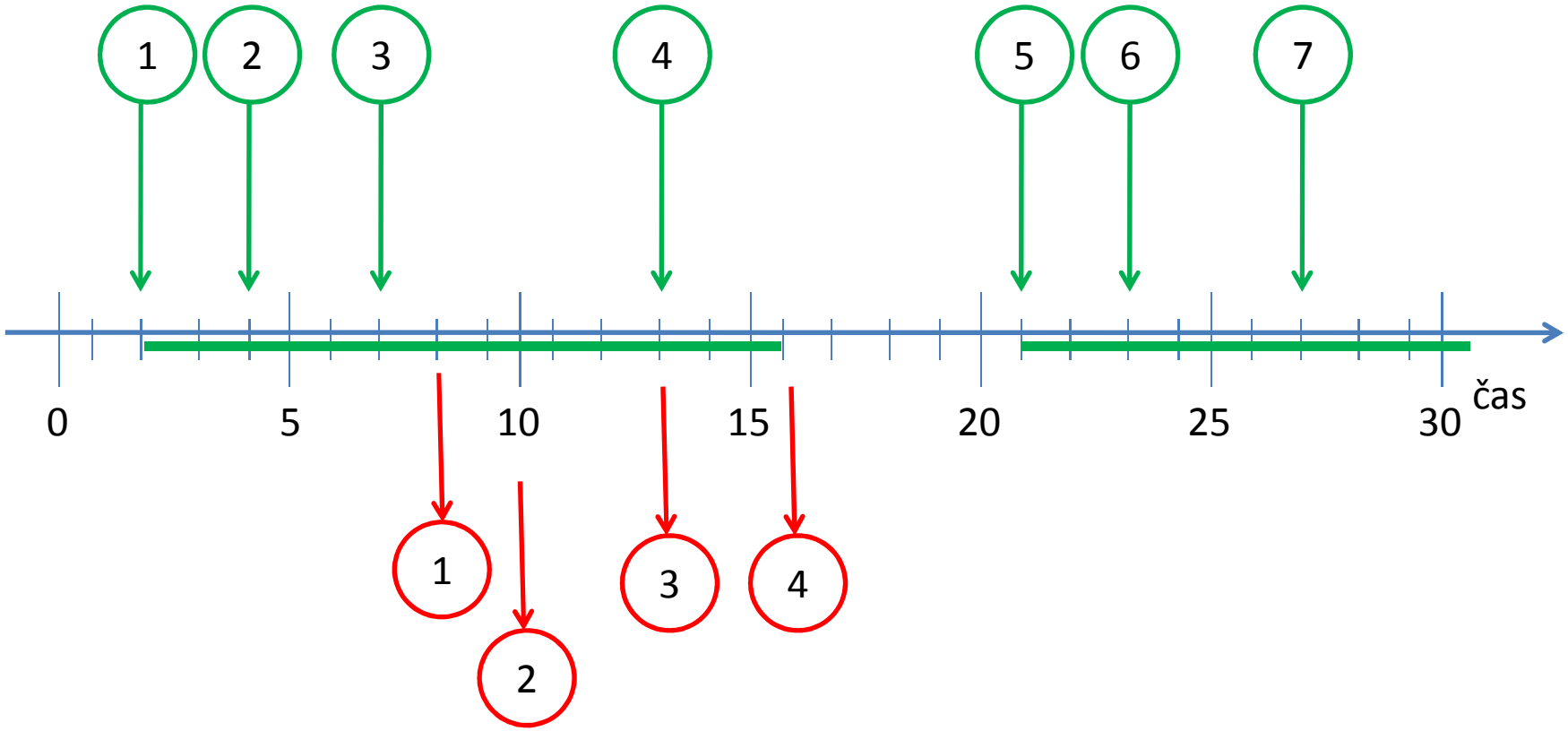
Algoritmy orientované na události

Čas	TU_1	TU_2	F	S
0	2	M	0	0
2	4	8	0	1
4	7	8	1	1
7	13	8	2	1
8	13	10	1	1
10	13	13	0	1
13	21	13	1	1
13	21	16	0	1
16	21	M	0	0
21	23	31	0	1
23	27	31	1	1
27	-	-	-	-

V tabulce jsou uvedeny jak hodnoty výskytů nejbližších událostí z obou tříd, tak hodnoty stavových proměnných modelu.

Algoritmy orientované na události

Na tomto obrázku je znázorněna činnost modelovaného systému graficky.

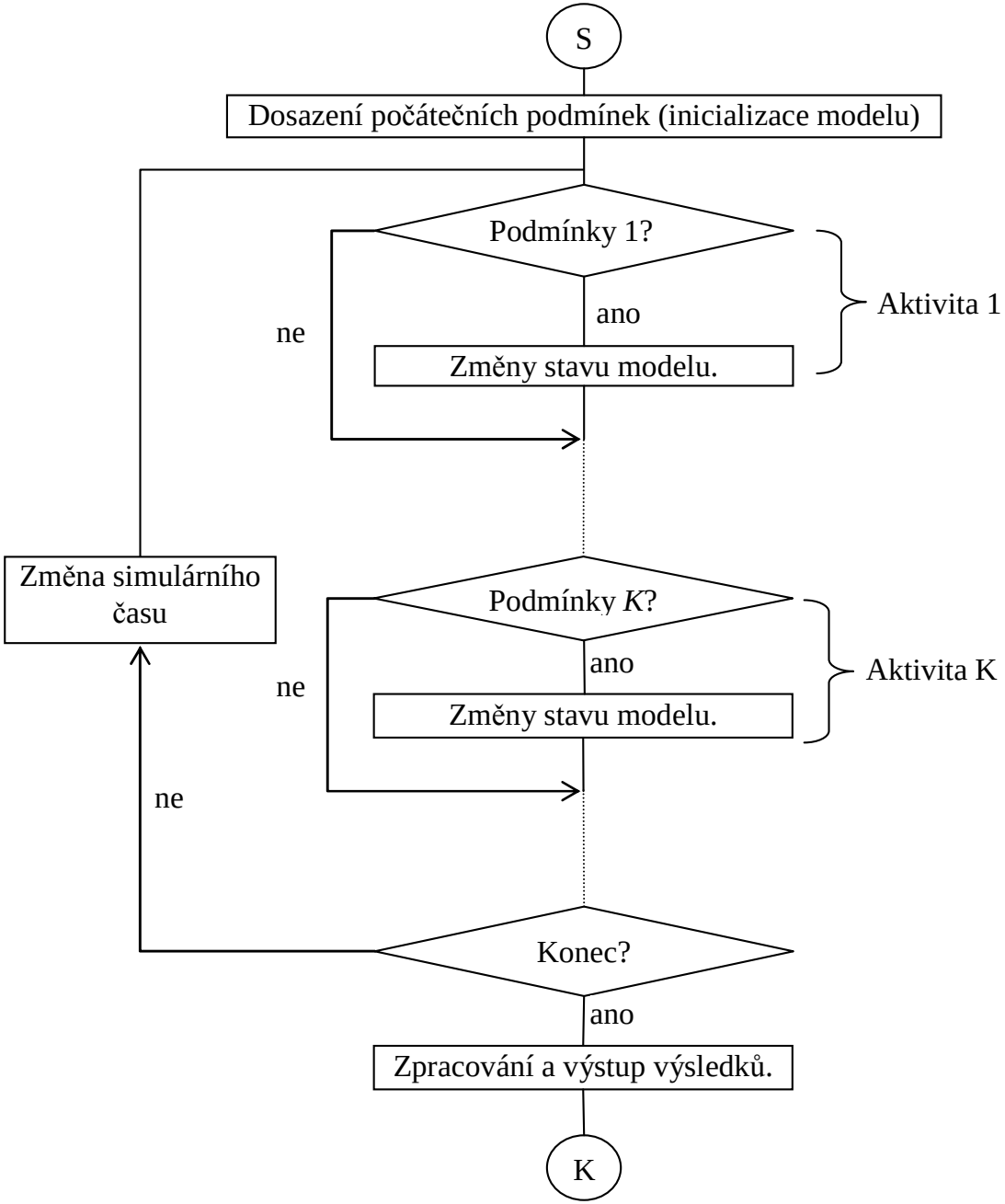


Algoritmy orientované na aktivity

- Pro zachycení dynamických vlastností modelovaného systému se využívají aktivity. Je nutno stanovit:
  - **Soubor aktivit**.
  - **Podmínky**, které musí být splněny, aby mohla daná aktivita nastat.
  - **Změny**, ke kterým dochází působením aktivit.

Algoritmy orientované na aktivity

- Každá aktivita začíná a končí událostí. Výskyty těchto událostí jsou zadány podmínkami, které musí být splněny. Jsou-li podmínky splněny, mohou se realizovat příslušné změny stavu modelu.
- Princip algoritmu orientovaného na aktivity je znázorněn na následujícím obrázku.


S
Dosazení počátečních podmínek (inicializace modelu)
Podmínky 1?
ano
ne
Změny stavu modelu.
Aktivita 1
Změna simulárního času
Podmínky K?
ano
ne
Změny stavu modelu.
Aktivita K
ne
Konec?
ano
Zpracování a výstup výsledků.
K

Algoritmy orientované na aktivity

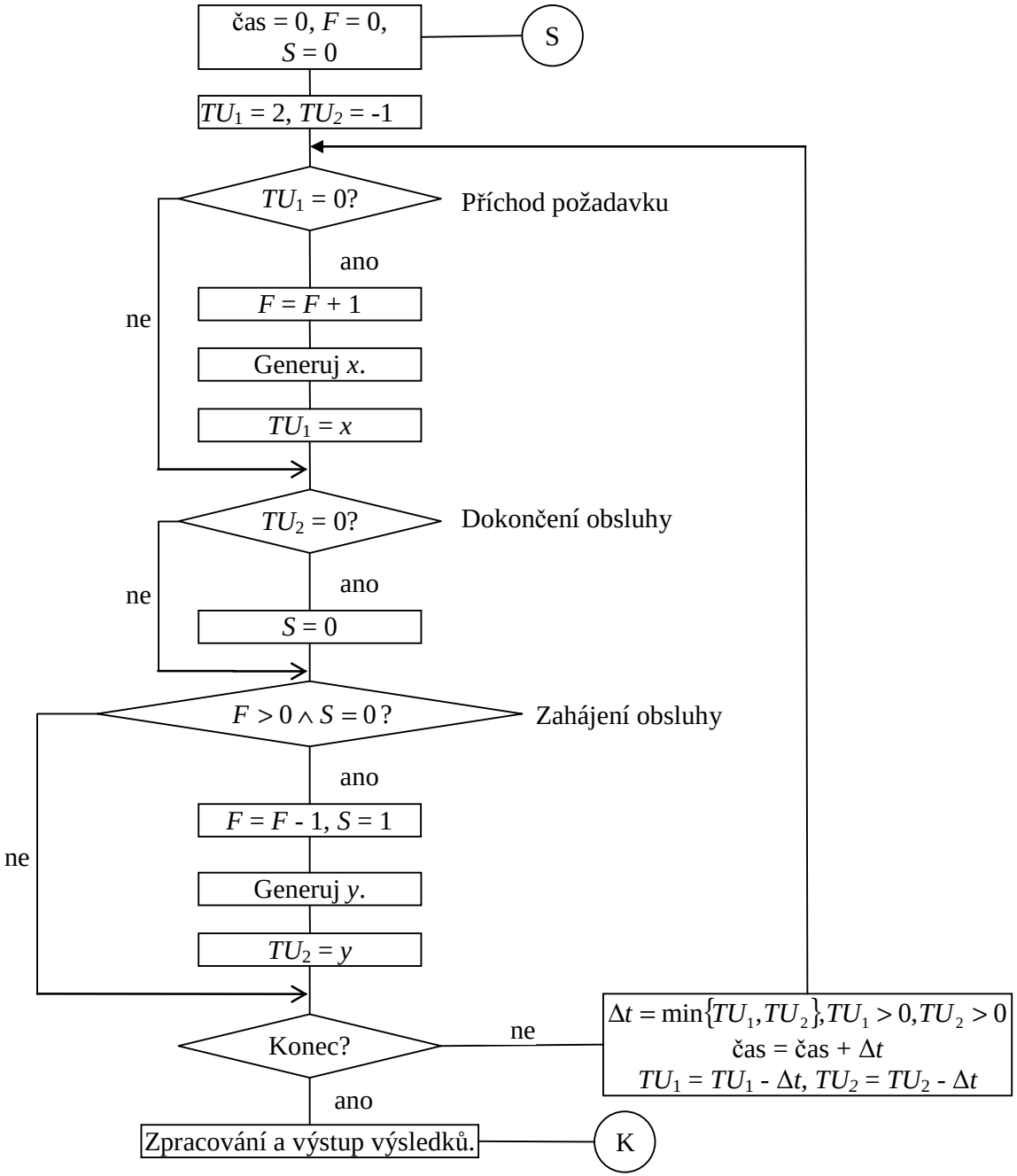
- Jelikož je při každé změně simulárního času testovat podmínky pro všechny aktivity, je z hlediska rychlosti výpočtu tento přístup zpravidla méně efektivní než přístup orientovaný na události.
- Tento přístup je vhodnější tehdy, je-li výskyt událostí spíše vázán na splnění určitých podmínek než na dosažení určité hodnoty simulárního času.

Algoritmy orientované na aktivity

- **Př. 2:** Nakreslete vývojový diagram zachycující modelující algoritmus orientovaný na aktivity systému hromadné obsluhy z předchozího příkladu.
- Definujme si opět stavové proměnné F a S, jež budou mít stejný význam.
- Proměnné TU_1 a TU_2 budou vyjadřovat, za jak dlouho (vzhledem k aktuální hodnotě simulárního času) dojde k dalšímu příchodu, resp. odchodu požadavku.

Algoritmy orientované na aktivity

- Bude-li některá proměnná TU_1 a TU_2 záporná, potom bude vyjadřovat dobu, jež uplynula od posledního výskytu příslušné aktivity.
- Při určování okamžiku příští změny stavu modelu se vyhledává minimum z kladných hodnot TU_1 a TU_2. Vybraná hodnota se přičte k hodnotě aktuálního simulárního času a současně se odečte od všech časových proměnných.


čas = 0, F = 0,
S = 0
S
TU1 = 2, TU2 = -1
TU1 = 0?
Příchod požadavku
ano
ne
F = F + 1
Generuj x.
TU1 = x
TU2 = 0?
Dokončení obsluhy
ano
ne
S = 0
F > 0 ∧ S = 0 ?
Zahájení obsluhy
ano
ne
F = F - 1, S = 1
Generuj y.
TU2 = y
Konec?
ne
Δt = min{TU1,TU2},TU1 > 0,TU2 > 0
čas = čas + Δt
TU1 = TU1 - Δt, TU2 = TU2 - Δt
ano
Zpracování a výstup výsledků.
K

Algoritmy orientované na procesy

- Tento přístup je kombinací obou předchozích přístupů.