Uvod v računalništvo (UvR)

Načrtovanje algoritmov

Danijel Skočaj
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Literatura: Invitation to Computer Science, poglavje 2

v1.0
Št. leto 2013/14

Cilji predavanja

- Pseudokoda in načrtovanje algoritmov

- Razumeti prednosti pseudokode
- Predstaviti algoritme s pseudokodo
- Identificirati zaporedne, pogojne in iterativne stavke
- Abstrahirati problem in narediti dekompozicijo problema od zgoraj navzdol
- Ilustrirati delovanje vzorčnih algoritmov

Predstavitev algoritmov

- Kako predstaviti algoritem?

- Naravni jezik
  - zelo ekspresiven, bogat, enostaven za uporabo, pogosto uporabljan
  - redundanten, nestrukturiran, dvoumen, kontekstno odvisen
- Programski jezik
  - strukturiran, načrtovan za računalnike, neredundanten
  - gramatično tečen, preveč precizen, kriptičen, veliko tehničnih podrobnosti

- Pseudokoda je nekje vmes
  - „programski jezik brez podrobnosti“
  - enostavna, berljiva, strukturirana, a brez strogih pravil
  - enostavna za vizualizacijo
  - enostavna za prevod v programske jezike

Primer v pseudokodi

FIGURE 1.2

Given: $m \geq 1$ and two positive numbers each containing m digits, $a_{m-1}\, a_{m-2} \ldots a_0$ and $b_{m-1}\, b_{m-2} \ldots b_0$

Wanted: $c_m\, c_{m-1}\, c_{m-2} \ldots c_0$, where $c_m c_{m-1}\, c_{m-2} \ldots c_0 = (a_{m-1}\, a_{m-2} \ldots a_0) + (b_{m-1}\, b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of *carry* to 0

Step 2 Set the value of i to 0

Step 3 While the value of i is less than or equal to $m - 1$, repeat the instructions in Steps 4 through 6

Step 4 Add the two digits a_i and b_i to the current value of *carry* to get c_i

Step 5 If $c_i \geq 10$, then reset c_i to $(c_i - 10)$ and reset the value of *carry* to 1; otherwise, set the new value of *carry* to 0

Step 6 Add 1 to i, effectively moving one column to the left

Step 7 Set c_m to the value of *carry*

Step 8 Print out the final answer, $c_m\, c_{m-1}\, c_{m-2} \ldots c_0$

Step 9 Stop

Algorithm for adding two *m*-digit numbers

Primer v naravnem jeziku

FIGURE 2.1

Initially, set the value of the variable *carry* to 0 and the value of the variable *i* to 0. When these initializations have been completed, begin looping as long as the value of the variable *i* is less than or equal to $(m - 1)$. First, add together the values of the two digits a_i and b_i and the current value of the carry digit to get the result called c_i. Now check the value of c_i to see whether it is greater than or equal to 10. If c_i is greater than or equal to 10, then reset the value of *carry* to 1 and reduce the value of c_i by 10; otherwise, set the value of *carry* to 0. When you are finished with that operation, add 1 to *i* and begin the loop all over again. When the loop has completed execution, set the leftmost digit of the result c_m to the value of *carry* and print out the final result, which consists of the digits $c_m\ c_{m-1} \ldots c_0$. After printing the result, the algorithm is finished, and it terminates.

The addition algorithm of Figure 1.2 expressed in natural language

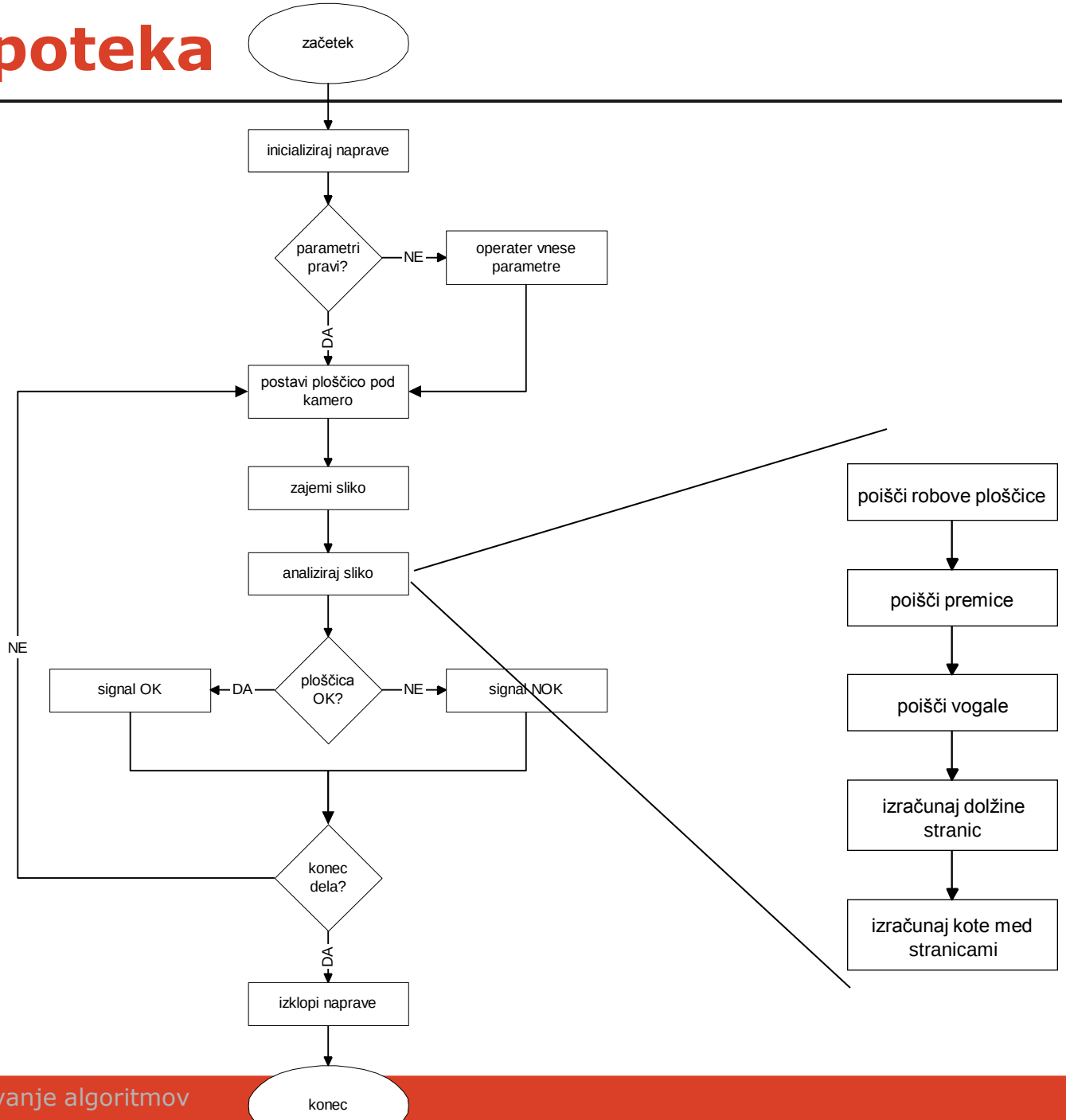
Primer v programskem jeziku

FIGURE 2.2

```
{
Scanner inp = new Scanner(System.in);
int i, m, carry;
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];
m = inp.nextInt();
for (int j = 0;j <= m-1;j++) {
       a[j] = inp.nextInt();
       b[j] = inp.nextInt();
}
carry = 0;
i = 0;
while (i < m) {
    c[i] = a[i] + b[i] + carry;
    if (c[i] >= 10)
              .
              .
              .
```

The beginning of the addition algorithm of Figure 1.2 expressed in a high-level programming language

Diagram poteka



Zaporedne operacije

- Tri osnovne zaporedne operacije:
  - računske: posamezen numerični izračun
  - vhod: dobi podatke v algoritem
  - izhod: posreduje podatke izven algoritma

- Spremenljivke: poimenovana lokacija za hranjenje vrednosti

- Zaporedni algoritem
  - je sestavljen samo iz zaporednih operacij
  - „premočrtni“ algoritem

Primer zaporednega algoritma

- Računanje povprečne porabe avtomobila

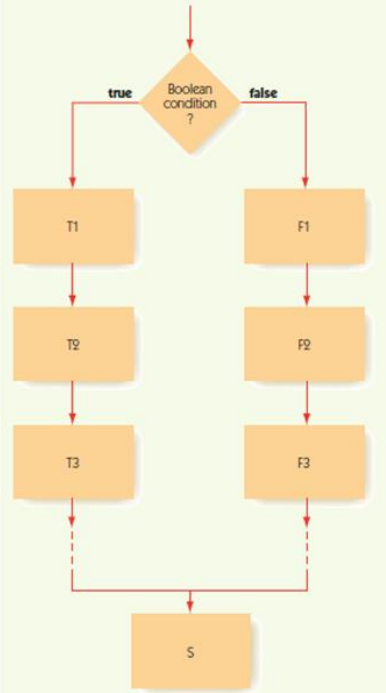
FIGURE 2.3

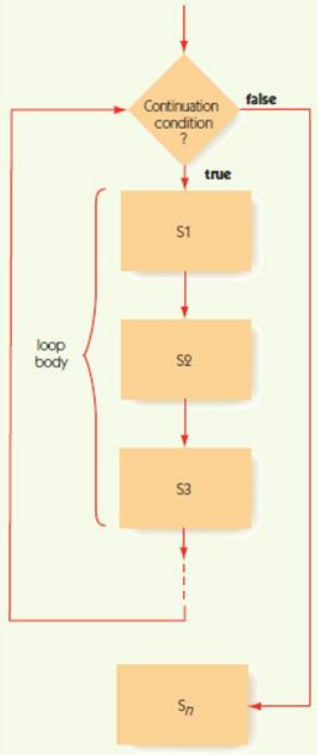
	Step	Operation
vhod	1	Get values for *gallons used*, *starting mileage*, *ending mileage*
računske	2	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
	3	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
izhod	4	Print the value of *average miles per gallon*
	5	Stop

Algorithm for computing average miles per gallon (version 1)

Nadzorne operacije

- Operacije za nadzor poteka izvajanja algoritma
  - Pogojni stavki (vejitve)
  - Iterativni stavki (iteracije)





Pogojni stavek

- Če-potem-sicer stavki (if-then-else)
- Kaj, če…

```
IF pogoj THEN
        stavki1
ELSE
        stavki2
```

1. Oceni pogoj
2. Če pogoj drži, se izvedejo stavki1
3. Če pogoj ne drži, se izvedejo stavki2

Vejitev

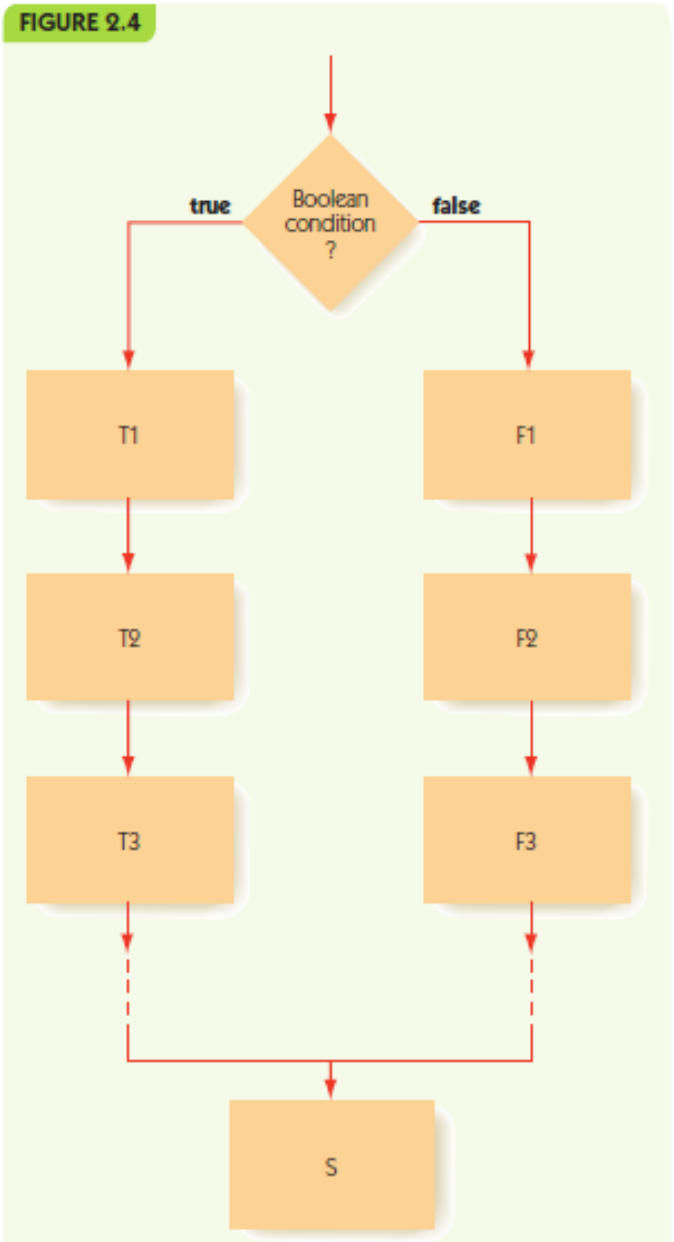


FIGURE 2.4
Boolean condition ?
true
false
T1
T2
T3
F1
F2
F3
S
The if/then/else pseudocode statement

Primer pogojnega stavka

FIGURE 2.5

Step	Operation
1	Get values for *gallons used*, *starting mileage*, *ending mileage*
2	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
4	Print the value of *average miles per gallon*
pogoj 5	If *average miles per gallon* is greater than 25.0 then
T 6	Print the message 'You are getting good gas mileage'
	Else
F 7	Print the message 'You are NOT getting good gas mileage'
8	Stop

Second version of the average miles per gallon algorithm

Iteracije

- While stavek (delaj dokler)
- Blok operacij (telo zanke) se iterativno izvaja v zanki dokler je izpolnjen pogoj za nadaljevanje
- Testiranje pogoja na začetku zanke (pretest loop):

```
WHILE pogoj DO
    stavki
    stavki
KONEC ZANKE
```

- Testiranje pogoja na koncu zanke (posttest loop):

```
DO
    stavki
    stavki
WHILE pogoj
```

- Vedno moramo poskrbeti za izhodni pogoj
  - izogibati se neskončni zanki

Iteracije

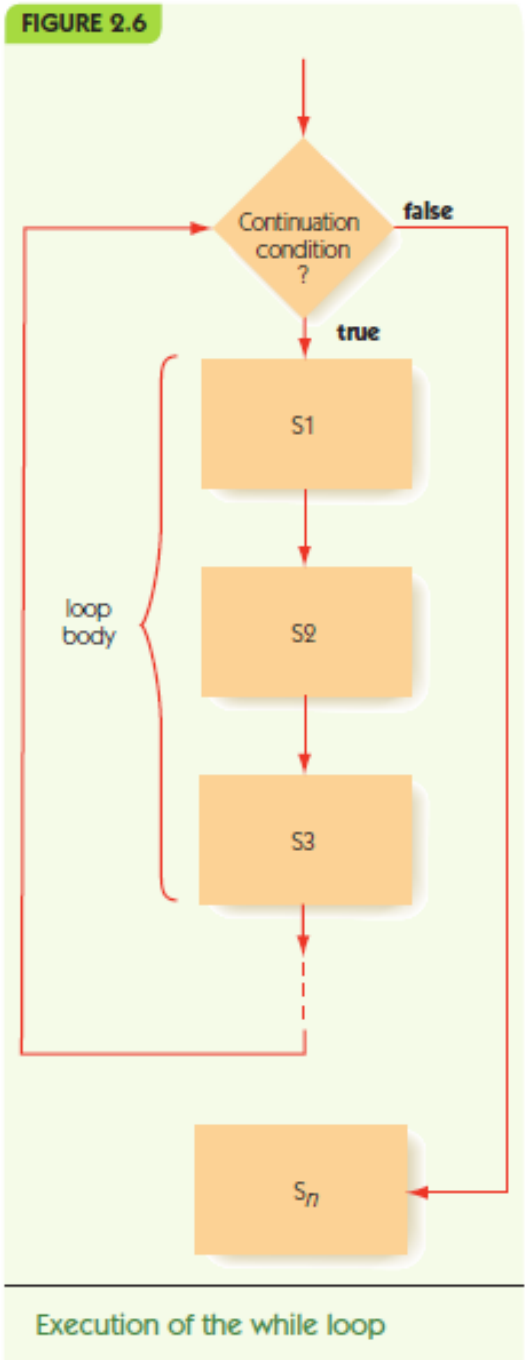


FIGURE 2.6
Continuation condition ?
false
true
S1
S2
S3
loop body
Sn
Execution of the while loop

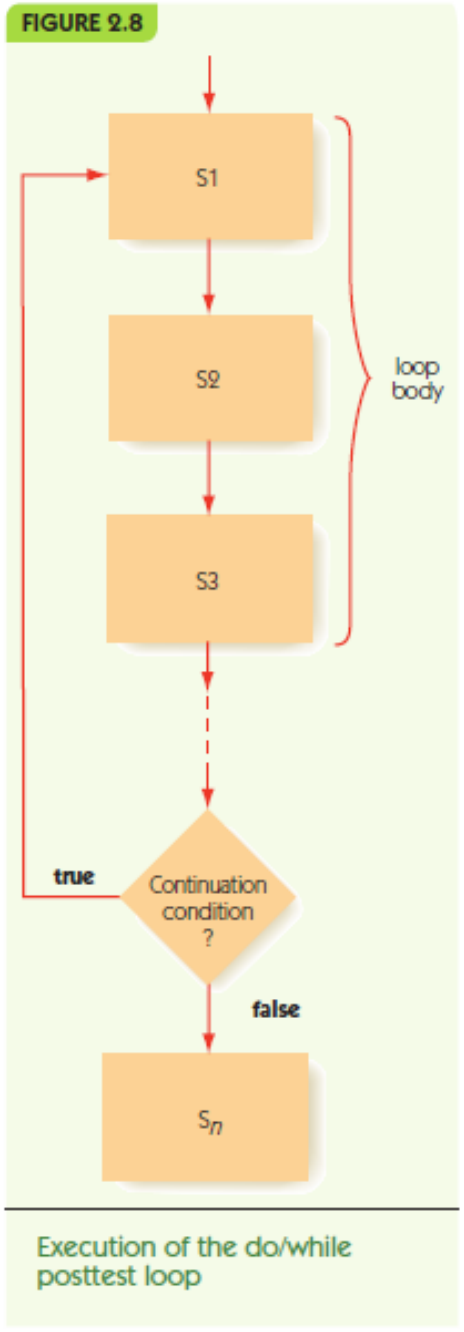


FIGURE 2.8
S1
S2
S3
loop body
true
Continuation condition ?
false
Sn
Execution of the do/while posttest loop

Primer iterativne zanke

FIGURE 2.7

Step	Operation
1	*response* = Yes
2	While (*response* = Yes) do Steps 3 through 11
3	Get values for *gallons used*, *starting mileage*, *ending mileage*
4	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
5	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
6	Print the value of *average miles per gallon*
7	If average miles per gallon > 25.0 then
8	Print the message 'You are getting good gas mileage'
	Else
9	Print the message 'You are NOT getting good gas mileage'
10	Print the message 'Do you want to do this again? Enter Yes or No'
11	Get a new value for *response* from the user
12	Stop

Third version of the average miles per gallon algorithm

Primitivne operacije

- Računske
- Vhod/izhod
- Pogojni stavek
- Iteracija

- Z njimi lahko predstavimo katerikoli algoritem!

FIGURE 2.9

Computation:

Set the value of "variable" to "arithmetic expression"

Input/Output:

Get a value for "variable", "variable"…
Print the value of "variable", "variable", …
Print the message 'message'

Conditional:

If "a true/false condition" is true then
    first set of algorithmic operations
Else
    second set of algorithmic operations

Iterative:

While ("a true/false condition") do Step *i* through Step *j*
    Step *i*: operation
    .
    .
    .
    Step *j*: operation

While ("a true/false condition") do
    operation
    .
    .
    .
    operation
End of the loop

Do
    operation
    operation
    .
    .
    .
While ("a true/false condition")

Summary of pseudocode language instructions

Primer 1: Množenje s seštevanjem

- Zmnoži števili a in b tako, da b-krat prišteješ a:

> *"Given two nonnegative integer values, a ≥ 0, b ≥ 0, compute and output the product (a × b) using the technique of repeated addition. That is, determine the value of the sum a + a + a + . . . + a (b times)."*

- Dobi vhodne vrednosti
  - dobi vrednosti a in b
- Izračunaj rezultat
  - v zanki v b-tih iteracijah, vsakokrat prištevaj a
- Vrni rezultat
  - izpiši končno vrednost

Primer 1: Množenje s seštevanjem

FIGURE 2.10

Get values for *a* and *b*
If (either *a* = 0 or *b* = 0) then
    Set the value of *product* to 0
Else
    Set the value of *count* to 0
    Set the value of *product* to 0
    While (*count* < *b*) do
        Set the value of *product* to (*product* + *a*)
        Set the value of *count* to (*count* + 1)
    End of loop
Print the value of *product*
Stop

Algorithm for multiplication of nonnegative values via repeated addition

Primer 2: Zaporedno iskanje

- Iskanje po telefonskem imeniku, ki ni urejen po abecedi:

> *"Assume that we have a list of 10,000 names that we define as N1, N2, N3, . . . , N10,000, along with the 10,000 telephone numbers of those individuals, denoted as T1, T2, T3, . . . , T10,000. To simplify the problem, we initially assume that all names in the book are unique and that the names need not be in alphabetical order."*

- Odkrivanje algoritmov!
- Tri verzije:
  - zaporedni algoritem (brez zank in pogojev)
  - nepopolni iterativni algoritem
  - pravilni algoritem

Primer 2: Zaporedno iskanje

FIGURE 2.11

Step	Operation
1	Get values for *NAME*, $N_1, \ldots, N_{10,000}$, and $T_1, \ldots, T_{10,000}$
2	If *NAME* = N_1 then print the value of T_1
3	If *NAME* = N_2 then print the value of T_2
4	If *NAME* = N_3 then print the value of T_3
.	.
.	.
.	.
10,000	If *NAME* = $N_{9,999}$ then print the value of $T_{9,999}$
10,001	If *NAME* = $N_{10,000}$ then print the value of $T_{10,000}$
10,002	Stop

First attempt at designing a sequential search algorithm

Primer 2: Zaporedno iskanje

FIGURE 2.12

Step	Operation
1	Get values for *NAME*, $N_1, \ldots, N_{10,000}$, and $T_1, \ldots, T_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While (*Found* = NO) do Steps 4 through 7
4	If *NAME* is equal to the *i*th name on the list N_i then
5	Print the telephone number of that person, T_i
6	Set the value of *Found* to YES
	Else (*NAME* is not equal to N_i)
7	Add 1 to the value of *i*
8	Stop

Second attempt at designing a sequential search algorithm

Primer 2: Zaporedno iskanje

FIGURE 2.13

Step	Operation
1	Get values for *NAME*, N_1, … , $N_{10,000}$, and T_1, … , $T_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While both (*Found* = NO) and ($i \leq 10{,}000$) do Steps 4 through 7
4	If *NAME* is equal to the *i*th name on the list N_i then
5	Print the telephone number of that person, T_i
6	Set the value of *Found* to YES
	Else (*NAME* is not equal to N_i)
7	Add 1 to the value of *i*
8	If (*Found* = NO) then
9	Print the message 'Sorry, this name is not in the directory'
10	Stop

The sequential search algorithm

Primer 2: Zaporedno iskanje

- Še vedno zelo potraten algoritem
- Problem: neustrezna organizacija podatkov!

- Izbira algoritma za reševanje danega problema je zelo odvisna od načina kako so organizirani vhodni podatki

- Rešitev: seznam urejen po abecednem vrstnem redu

Primer 3: Iskanje največjega števila

- Iskanje največjega števila v (neurejenem) seznamu števil (ki se ne ponavljajo)
  - vrni največje število in njegovo lokacijo

> *"Given a value n ≥ 1 and a list containing exactly n unique numbers called A1, A2, . . . , An, find and print out both the largest value in the list and the position in the list where that largest value occurred."*

- Pogost problem v drugih algoritmih
- Algoritem je pogosto osnovni gradnik drugih algoritmov
- Knjižnica algoritmov: zbirka koristnih algoritmov

Primer 3: Iskanje največjega števila

FIGURE 2.14

Get a value for n, the size of the list
Get values for $A_1, A_2, \ldots, A_n$, the list to be searched
Set the value of *largest so far* to A_1
Set the value of *location* to 1
Set the value of i to 2
While ($i \leq n$) do
    If $A_i >$ *largest so far* then
        Set *largest so far* to A_i
        Set *location* to i
    Add 1 to the value of i
End of the loop
Print out the values of *largest so far* and *location*
Stop

Algorithm to find the largest value in a list

Primer 4: Razpoznavanje vzorcev

- V seznamu zaporednih podatkov poišči del, ki se ujema z vzorcem, ki je predmet povpraševanja

> *"You will be given some text composed of n characters that will be referred to as T1 T2 . . . Tn. You will also be given a pattern of m characters, m ≤ n, that will be represented as P1 P2 . . . Pm. The algorithm must locate every occurrence of the given pattern within the text. The output of the algorithm is the location in the text where each match occurred."*

- Zelo pogost problem:
  - iskanje po besedilu v urejevalniku besedil
  - iskanje po spletu
  - razpoznavanje vzorcev na slikah
  - iskanje po človeškem genomu

Primer 4: Razpoznavanje vzorcev

- Algoritem ima dva dela:
  1. Pomikaj vzorec vzdolž besedila in ga po vrsti poravnaj z vsako pozicijo
  2. Za vsako posamezno pozicijo, ugotovi, če se vzorec ujema s trenutnim delom besedila

- Reši oba dela ločeno, uporabljaj:
  - abstrakcijo
    - osredotoči se na visokonivojske koncepte, ne na podrobnosti
  - načrtovanje od zgoraj navzdol
    - začni z veliko sliko
    - postopoma razdelaj posamezne dele

Primer 4: Razpoznavanje vzorcev

FIGURE 2.15

Get values for n and m, the size of the text and the pattern, respectively
Get values for both the text $T_1\ T_2 \ldots T_n$ and the pattern $P_1\ P_2 \ldots P_m$
Set k, the starting location for the attempted match, to 1
Keep going until we have fallen off the end of the text
    Attempt to match every character in the pattern beginning at
        position k of the text (this is Step 1 from the previous page)
    If there was a match then
        Print the value of k, the starting location of the match
    Add 1 to k, which slides the pattern forward one position (this is Step 2)
End of the loop
Stop

First draft of the pattern-matching algorithm

Primer 4: Razpoznavanje vzorcev

FIGURE 2.16

Get values for n and m, the size of the text and the pattern, respectively
Get values for both the text $T_1\ T_2 \ldots T_n$ and the pattern $P_1\ P_2 \ldots P_m$
Set k, the starting location for the attempted match, to 1
While $(k \leq (n - m + 1))$ do
    Set the value of i to 1
    Set the value of *Mismatch* to NO
    While both $(i \leq m)$ and $(Mismatch = NO)$ do
        If $P_i \neq T_{k+(i-1)}$ then
            Set *Mismatch* to YES
        Else
            Increment i by 1 (to move to the next character)
    End of the loop
    If *Mismatch* = NO then
        Print the message 'There is a match at position'
        Print the value of k
    Increment k by 1
End of the loop
Stop, we are finished

Final draft of the pattern-matching algorithm

Povzetek

- Pseudokoda
- Operacije:
  - zaporedne
  - pogojne
  - iterativne
- Algoritmično reševanje problemov:
  - postopno reševanje algoritmov po delih
  - uporaba abstrakcije
  - reševanje problemov od zgoraj navzdol

- Pravilnost algoritmov
- Učinkovitost algoritmov!