Algoritmus a jeho vlastnosti

Algoritmus – všeobecný zápis návodu, postup na mechanické vyriešenie úlohy
Proces – dej, ktorý prebieha počas vykonávania algoritmu
Procesor – vykonávateľ algoritmu (človek, počítač)

Vstupná podmienka – podmienka, ktorú musia spĺňať údaje na vstupe algoritmu, pre ktoré, pri správnom algoritme, dostaneme správny výsledok.
Napr.: A, B sú celé čísla;
Výstupná podmienka – podmienka, ktorú musia spĺňať všetky údaje na výstupe algoritmu, ak použité vstupné údaje spĺňajú vstupnú podmienku a algoritmus je správny.
Napr.: NSD je celé číslo, pre ktoré platí: NSD|A a NSD|B a ∄ NSD1>NSD: NSD1|A a NSD1|B
Špecifikácia algoritmickej úlohy – určenie vstupnej a výstupnej podmienky.
Údaje, ktoré spĺňajú vstupnú podmienku nazývame **vstupné údaje** (**vstupné hodnoty**).
Údaje, ktoré spĺňajú výstupnú podmienku nazývame **výstupné údaje** (**výstupné hodnoty**).

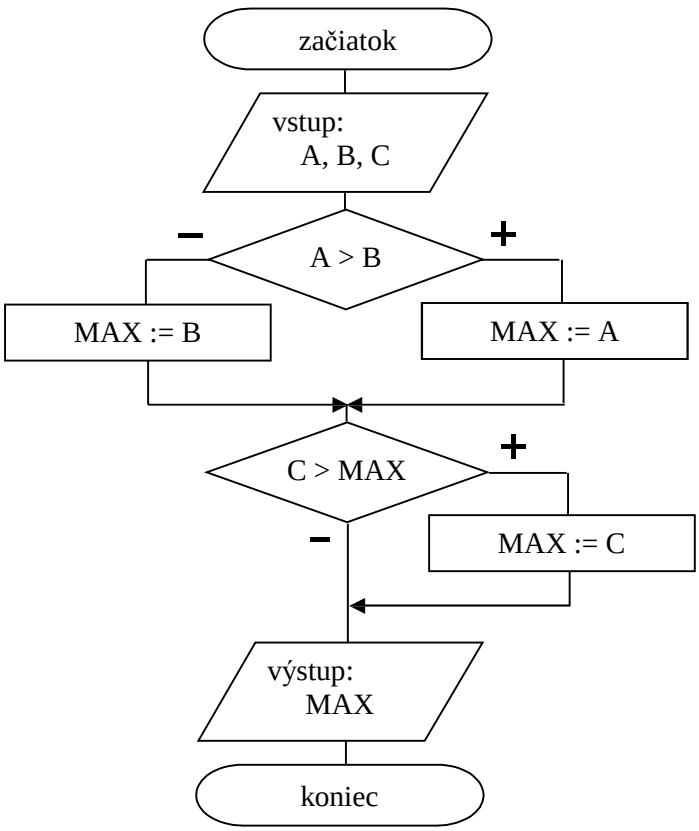
Algoritmus je konečná postupnosť elementárnych príkazov, vykonanie ktorých, pre prípustné vstupné údaje, spĺňajúce vstupné podmienky, po konečnom počte krokov vedie mechanicky k výstupným údajom spĺňajúcim výstupné podmienky.

Vlastnosti algoritmu:
- hromadnosť – algoritmus slúži na riešenie celej skupiny úloh rovnakého typu. Algoritmus sa „dozvie“ až po zadaní konkrétnych vstupných údajov, ktorú „konkrétnu“ úlohu rieši.
- jednoznačnosť (deterministickosť) – po každom kroku (akcii) je jednoznačne určený ďalší krok. Táto vlastnosť zabezpečuje, že pre tie isté vstupné údaje dostane vždy rovnaké výsledky.

Ďalšie vlastnosti sú konečnosť, rezultatívnosť, elementárnosť atď.

Formy zápisu algoritmu:

- **zápis v jazyku vývojových diagramov** (JVD) používa štátnou normou stanovené značky, prehľadne (graficky) vyjadruje tok riadenia a dát; ide o staršiu formu zápisu algoritmu.
  Príklad: Algoritmus na nájdenie najväčšieho z troch čísel vo forme vývojového diagramu.

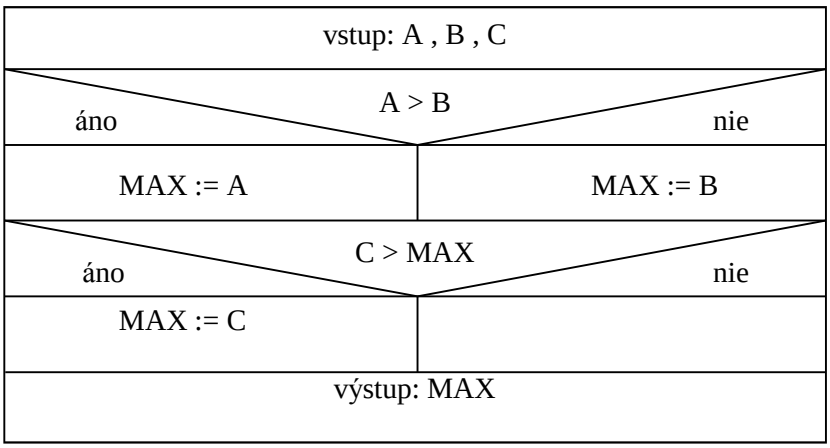


Algoritmus sa vykonáva zhora nadol. Vetvou „+“ sa pôjde, keď bude podmienka splnená a vetvou „ -“, keď podmienka nebude splnená. Symbol „:=“ čítame „priraď“.

- **slovný zápis** používa tzv. vyhradené slová, napr. ak – tak – inak, čítaj, začiatok a pod.
  Príklad: Slovný zápis algoritmu na nájdenie najväčšieho z troch čísel.

```
začiatok
čítaj ( A , B , C );
ak A > B
    tak   MAX := A
    inak MAX := B;
ak C > MAX
    tak   MAX :=C;
píš ( MAX );
koniec.
```

- **zápis štruktúrogramom** – najnovšia grafická forma zápisu algoritmu.
  Príklad: Algoritmus na nájdenie najväčšieho z troch čísel zapísaný N-S diagramom.

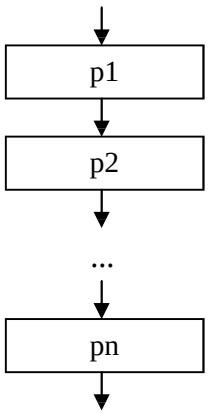


Algoritmické konštrukcie

Ak je úloha algoritmicky riešiteľná, jej algoritmus možno vytvoriť kombináciou len troch algoritmických konštrukcií. Algoritmické konštrukcie sú: **sekvencia, vetvenie a cyklus**.

SEKVENCIA

Sekvencia je najjednoduchšou algoritmickou konštrukciou. Použijeme ju, ak sa majú príkazy vykonať za sebou, v poradí, ako sú zapísané.
Má tvar:



v slovnom zápise a v pascale:

p1;
p2;
...
pn;

kde p1, p2, ... , pn sú príkazy

Príkazy od seba oddeľujeme bodkočiarkou!

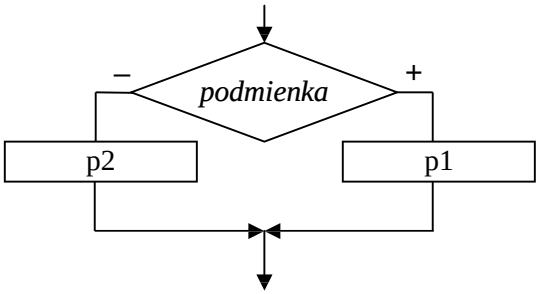
Vykonanie sekvencie: Príkazy p1, p2 až pn sa vykonajú za sebou v poradí, ako sú zapísané (ak neobsahujú riadiaci príkaz).
Riadiace príkazy môžu zmeniť poradie vykonávania príkazov, môžu zabezpečiť vykonanie skupiny príkazov, len ak je splnené určitá podmienka (vetvenie) alebo môžu zabezpečiť opakované vykonávanie skupiny príkazov (cyklus).

VETVENIE, ROZHODOVANIE

Vetvenie použijeme, ak vykonanie príkazu (alebo skupiny príkazov) je podmienené splnením určitej podmienky. Nesplnenie danej podmienky môže viesť k vykonaniu inej skupiny príkazov.
Vetvenie poznáme **binárne** (dve možnosti) a **n-árne** (n možností, n≥2).

Úplné binárne vetvenie, podmienený príkaz if

Má tvar:



ak ***podmienka***
tak p1
inak p2
kde p1 a p2 sú príkazy

Vykonanie: Ak je podmienka splnená, vykoná sa príkaz p1, ak nie je splnená, vykoná sa príkaz p2.

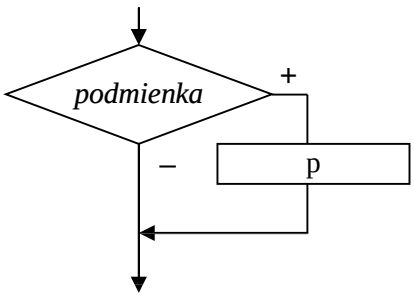
V pascale úplnému binárnemu vetveniu zodpovedá úplný príkaz if, ktorý má tvar:

if b **then** p1
else p2 kde b je výraz typu boolean, p1 a p2 sú príkazy

Vykonanie úplného príkazu if: Ak výraz b nadobudne hodnotu true, vykoná sa príkaz p1, ak nadobudne hodnotu false, vykoná sa príkaz p2.

Neúplné binárne vetvenie, neúplný príkaz if

Má tvar:



ak ***podmienka***
tak p
kde p je príkaz

Vykonanie: Ak je podmienka splnená, vykoná sa príkaz p, inak je vetvenie bez účinku.

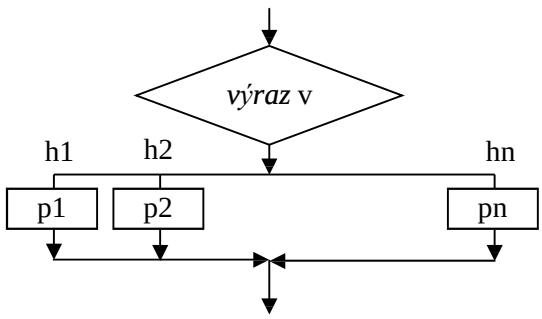
V TP neúplnému binárnemu vetveniu zodpovedá neúplný podmienený príkaz if, ktorý má tvar:

if b **then** p kde b je výraz typu boolean a p je príkaz

Vykonanie neúplného príkazu if: Ak výraz b nadobudne hodnotu true, vykoná sa príkaz p, ak nadobudne hodnotu false, príkaz if je bez účinku.

N-árne vetvenie, podmienený príkaz case

Má tvar:



h1, h2 až hn sú hodnoty, ktoré môže nadobudnúť výraz v

v slovnom zápise nemá ekvivalent, realizuje sa zápisom:

ak *podm1* tak p1
inak ak *podm2* tak p2
inak ak *podm3* tak p3
inak ...
inak ak *podmn* tak pn

kde p1 až pn sú príkazy

V pascale sa n-árne vetvenie realizuje podmieneným príkazom case.

Má tvar:	**case v of**	kde v je tzv. výberový výraz ordinálneho typu,
	h1 : p1;	h1, h2 až hn sú hodnoty rovnakého typu ako výberový výraz
	h2 : p2;	a p, p1, p2 až pn sú príkazy
	...	
	hn : pn	
	[else p]	do hranatých zátvoriek sa umiestňuje nepovinná časť
	end	príkaz case končí vyhradeným slovom end

Vykonanie: Vyhodnotí sa výberový výraz a vykoná príkaz, predznačený hodnotou, ktorú nadobudol výberový výraz. Ak výraz v nenadobudne ani jednu z hodnôt h1 až hn a príkaz case obsahuje časť else, vykoná sa príkaz p; ak príkaz case neobsahuje časť else, príkaz case je bez účinku.

CYKLUS

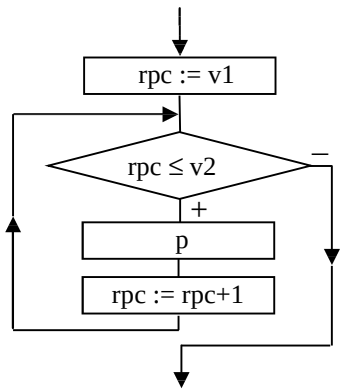
Použijeme, ak nejaký príkaz alebo skupina príkazov sa má opakovane vykonávať, pokiaľ je splnená (prípadne nesplnená) určitá podmienka.

Cykly rozdeľujeme:

1. podľa umiestnenia podmienky na:
   a) cyklus s podmienkou na začiatku
   b) cyklus s podmienkou na konci
   c) úplný cyklus (s podmienkou v strede)
2. podľa toho, či je známy alebo neznámy počet opakovaní v cykle na:
   a) cyklus s explicitne (zvonka) daným počtom opakovaní
   b) cyklus s implicitne (zvnútra) daným počtom opakovaní.

Cyklus s pevným počtom opakovaní, príkaz for

Cyklus s pevným počtom opakovaní použijeme pri známom počte opakovaní príkazov v cykle. V algoritmizácii nemá tento cyklus jednoznačnú formu zobrazenia, my sme sa rozhodli pre tvar:



v slovnom zápise:

pre *rpc* := *v1* až po (naspäť po) *v2* opakuj
p

kde rpc je tzv. riadiaca premenná cyklu (riadi počet prechodov cyklom), v1 a v2 sú výrazy a p je príkaz

V pascale cyklu s pevným počtom opakovaní zodpovedá príkaz for.

Má tvar: **for *rpc* := *v1* to (downto) *v2* do *p***

kde rpc je tzv. riadiaca premenná cyklu ordinálneho typu, v1 a v2 sú výrazy rovnakého typu ako rpc a p je príkaz; príkaz for má dva varianty, buď s „to“ alebo s „downto“

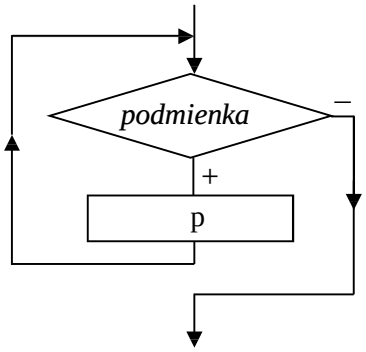
Vykonanie príkazu for (v zátvorkách pre downto):

1. vyhodnotia sa výrazy v1 a v2; hodnota výrazu v1 sa priradí ako začiatočná hodnota rpc, hodnota výrazu v2 ako koncová hodnota rpc,
2. pokiaľ je hodnota rpc menšia (väčšia) alebo rovná koncovej hodnote, opakovane sa vykonáva príkaz p a rpc nadobúda hodnoty nasledovníka rpc (predchodcu rpc).

Cyklus s podmienkou na začiatku, príkaz while

O typickom použití cyklu s podmienkou na začiatku si povieme v časti „Kedy ktorý cyklus“.

Má tvar:



v slovnom zápise:

pokiaľ *podmienka* opakuj *p*

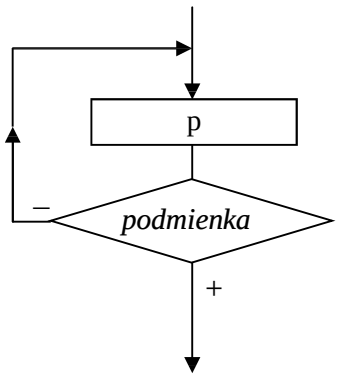
kde p je príkaz

V pascale má tvar: **while *b* do *p*** kde b je výraz typu boolean a p príkaz

Vykonanie: Pokiaľ výraz b nadobúda hodnotu true, opakovane sa vykonáva príkaz p, ak výraz b nadobudne hodnotu false, cyklus sa ukončí.

Cyklus s podmienkou na konci, príkaz repeat

Má tvar:



v slovnom zápise:

opakuj
p1;
p2;
...
pn
pokiaľ nebude *podmienka* (splnená)

kde p1, p2 až pn sú príkazy

V pascale má tvar:

repeat
p1;
p2;
...
pn
until *b* kde b je výraz typu boolean a p1 až pn sú príkazy

Vykonanie: Vykonajú sa príkazy p1, p2 až pn a opakovane sa budú vykonávať, pokiaľ výraz b bude nadobúdať hodnotu false. Ak výraz b nadobudne hodnotu true, cyklus sa ukončí.

Kedy ktorý cyklus

Keď už máme základnú predstavu o fungovaní jednotlivých cyklov a im zodpovedajúcich príkazov, môžeme si poznatky trochu zosystematizovať:

- ak vieme počet opakovaní príkazov v cykle a premenná, ktorá riadi počet prechodov cyklom, sa môže meniť na nasledovníka alebo predchodcu (krok +/- 1 alebo postupnosť za sebou idúcich znakov prípadne vymenovaných hodnôt), použijeme príkaz for
- ak nevieme počet opakovaní príkazov v cykle alebo premenná, ktorá riadi počet prechodov cyklom, nenadobúda „pekné“ hodnoty (hodnoty nasledovníkov alebo predchodcov), použijeme príkazy while alebo repeat, pričom:

- príkaz while, t.j. s podmienkou na začiatku použijeme, ak môže nastať situácii, že príkaz v cykle sa nemá vykonať ani raz
- príkaz repeat, t.j. s podmienkou na konci použijeme, ak sa príkazy v cykle majú vykonať aspoň raz.

Uvedomte si, že v každom cykle musí byť premenná, ktorá riadi počet prechodov daným cyklom! Kontrolujte, či sa jej hodnoty zväčšujú alebo zmenšujú a či sú ohraničené podmienkou ukončenia cyklu, ináč cyklus nikdy neskončí.

Literatúra:

Základy programovania – Turbo Pascal (www.gymparnr.edu.sk/informatika)
Informatika pre stredné školy (učebnica pre 1.ročník)