Algoritmus a algoritmizácia

Algoritmom rozumieme návod na systematický matematický postup, ktorý v konečnom počte krokov dáva odpoveď na určitú otázku alebo dáva riešenie určitého problému.
(napr. návod na nájdene najväčšieho spoločného deliteľa, sčítanie a odčítanie veľkých čísel....)

Zjednodušene:
Algoritmus je postupnosť krokov, pomocou ktorého môžeme vyriešiť zadaný problém.

Algoritmus by mal spĺňať určité vlastnosti a to:

- **Konečnosť** (algoritmus vždy skončí)[1]
- **Determinovanosť** (v každom kroku je jednoznačne určené, čo treba urobiť ďalej – teda výsledok každého kroku je jednoznačne určený výsledkami predchádzajúcich krokov)
- **Hromadnosť**(algoritmus je použiteľný v nekonečnom množstve prípadov – teda vstupné údaje môžu byť vyberané z potenciálne nekonečnej množiny)
- **Diskrétnosť** (je to postup prebiehajúci v diskrétnom čase tak, že na začiatku je daný vstupný údaj a v každý nasledujúci moment dostávame nový údaj)

Algoritmizácia je utváranie, utvorenie algoritmov; zostavovanie, zostavenie algoritmov. Ak sa podarí algoritmizovať materiálne alebo duševné procesy, je to v mnohých ohľadoch významné. Algoritmizácia duševných procesov (matematických operácií logických dedukcií atď.) je predpokladom toho, že tieto operácie môžu prevziať počítače.

Etapy riešenia problému

Spracovanie informácií predstavuje proces, v ktorom sú konkrétne vstupné údaje pretvárané do výsledkov, ktoré možno použiť na riadenie a rozhodovanie. Ak chceme riešiť akúkoľvek úlohu na počítači, treba ju rozdeliť na celý rad prípravných prác – etáp. Algoritmizácia úloh má tri základné etapy: a) **formulácia úlohy** b) **analýza úlohy** c) **zostavenie riešiaceho algoritmu**

Formulácia úlohy

Prvým predpokladom, aby sme danú úlohu mohli riešiť na počítači, je jej jasná a jednoznačná formulácia a identifikácia, ako aj ujasnenie cieľa, ktorý sledujeme riešením príslušnej úlohy. Za tým nasleduje tzv. formulácia problému, napr. matematickými prostriedkami (modelom), čiže problém musíme formalizovať pomocou nejakej sústavy vzťahov medzi premennými a konštantami. Formalizovanie konkrétnej úlohy si spravidla vyžaduje individuálny prístup, adaptáciu štandardných postupov, príp. nový typ modelu. Na formalizáciu možno použiť aj iný spôsob ako matematický, môže to byť napr. grafický model. Pre riešenie úloh na počítači je však matematická formulácia najvhodnejšia.

[1] Podľa L. Bukovského (a nielen neho ☺) to s konečnosťou tak nie je – odpoveď na to, či algoritmus pri danom vstupe skončí je dokonca netriviálna – o tom hovorí tzv. Halting problem – tzn. je nedokázateľné, že algoritmus pri danom vstupe skončí)

Analýza úlohy

V tejto etape je potrebné nájsť algoritmus riešenia úlohy. Zisťuje sa, či úloha je riešiteľná, či má jedno alebo viac riešení, načrtávajú sa možnosti riešenia a rozhoduje sa o druhu metód. Vytypovaná metóda riešenia musí zabezpečovať dosiahnutie požadovaných výsledkov (výstupné informácie), ale zároveň musí presne určiť, ktoré vstupné údaje budú potrebné. Úloha sa zovšeobecňuje a uskutočňuje sa prvá predstava o algoritmickej riešiteľnosti.

Zostavenie riešiaceho algoritmu

Po správnej formulácii a analýze úlohy nasleduje etapa syntetickej činnosti, v ktorej sa popíše logika a postup riešenia úlohy. Výsledkom tejto etapy je riešiaci algoritmus. Do tejto etapy môžeme zahrnúť aj programovanie úlohy. Pod pojmom programovanie rozumieme činnosť, pomocou ktorej sa uskutočňuje prevod úlohy z ľudského vedomia do formy vhodnej pre spracovanie na počítači. Výsledkom tejto činnosti je program. Program je algoritmus v takej forme, ktorej rozumie počítač, t. j. program je zápis algoritmu v niektorom programovacom jazyku.

Do etáp riešenia problému by sme mohli zahrnúť aj ladenie programu:

Ladenie – odstraňovanie chýb (debuging) a testovanie (je dobré vyskúšať hraničné situácie).

Chyby v programe

1. Syntaktické (syntax errors)
- spôsobené porušením pravidiel programovacieho jazyka
- prekladač programovacieho jazyka, potom nerozumie zapísanému algoritmu
- Napr.:

```
writeline;
a = a + 1;
readln(a)
```

2. Logické (logical errors)
- spôsobené tým, že program robí niečo iné ako má
- chyby tohto typu sa zisťujú ťažšie.
- pre hľadanie týchto chýb slúži v programovacom prostredí Pascal – možnosť - Debuger
- Napr.:

Za predpokladu, že chceme zväčšovať hodnotu premennej X o jednotku a zapíšeme to príkazom:

```
inc(x+1);
```

vzniká logická chyba – správny príkaz

```
inc(x); alebo x:=x+1;
```

3. Chyby pri behu programu (run-time errors)
- sú chyby spôsobené tým, že program pracuje s nesprávnymi hodnotami

JEDNODUCHÉ TYPY	ORDIÁLNE	celočíselné	Byte shortint integer word longint
		Boolean char vymenovaný interval	
	NEORDINÁLNE	Real single double extended comp	
ŠTRUKTUROVANÉ TYPY	Pole	viacrozmerná skupina dát rovnakého typu	
	Záznam	pevne daná skupina dát rôzneho typu	
	Množina	pevne daná skupina dát rovnakého typu (bez usporiadania)	
	Súbor	premenná štruktúra prvkov rovnakého typu, z ktorých je prístupný vždy len jeden	
UKAZOVATEĽ			

Typ	Rozsah	Nároky na pamäť
byte	0..255	1 byte
shortint	-128..127	1 byte
word	0..65535	2 byty
integer	-32 768..32 767	2 byty
longint	-2 147 483 648 .. 2 147 483 647	4 byty