Petr Drlík

TURBO PASCAL I.

Nitra 1998

1. PROBLÉM A RIEŠENIE PROBLÉMU (POSTUP – ALGORITMUS – PROGRAM)

AKO ZAČAŤ?

Chceme sa naučiť riešiť problémy pomocou počítača[1], vedieť s ním komunikovať nielen ako bežní používatelia hotových produktov, ale aj ako tvorcovia (autori) "nových" riešení. Je zrejmé, že je málo toho, čo by už nebolo povedané či napísané. Ale nie je isté, či to bolo povedané a napísané tak, ako to dokážeme my. Do programov – zjednodušene povedané predpisov toho, čo má vykonávať počítač – vkladá každý programátor kúsok seba, spolu s klasickým postupom riešenia dodáva k nemu aj svoj estetický a podľa neho efektívny pohľad na spracovávanú problematiku. O tom, či je to pre iného používateľa estetické a efektívne, by sa dalo meditovať. Ale základom programovania je práve možnosť realizovať riešenie problému podľa svojich predstáv, dať riešeniu svoj vlastný "image" (čítaj "imidž" - preložiť zrejmé pojmy je obtiažne). Je to podobné umeniu i skutočnému životu (nie škole), kde platí, že keď dvaja robia to isté, nikdy to nie je to isté.

Programovanie má oproti mnohým iným činnostiam, ku ktorým nás "dobrovoľne-povinne" vedú v škole, obrovskú výhodu. Často nás núti poriadne naplno roztočiť svoje mozgové závity, rozmýšľať o tom, čo je podstatné a čo nie, čo má byť viditeľné a čo skryté, realizovať svoju vlastnú predstavu použitím známych prostriedkov. Vždy, keď sa človek do niečoho pustí, musí mať správnu motiváciu. Tou môže byť aj skutočnosť, že s počítačmi a informačnými technológiami sa už dnes stretnete prakticky všade. Nechcete dokázať prinútiť stroj – počítač, aby robil presne to, čo mu zadáte? Je to výzva, ktorej sa ťažko odoláva, ak chcete naozaj niečo dosiahnuť. Skúste to!

Prvé, čo potrebujeme, je zaviesť základné pojmy, vzťahy medzi nimi a metódy – t. j. spôsoby práce s nimi, skrátka a múdro povedané - *terminológiu*. Mali sme tu pojmy "problém", "vyriešenie problému", "informatika", "programovanie", "počítač", určite toho však budeme musieť zvládnuť viac.

---

[1] *"Osobné počítače, ktoré sa dostali aj do domácností majú oproti ostatným členom domácnosti, kde dnes často hrajú významnú úlohu aj domáce zvieratá, isté prednosti: a) nehryzú, neškriabu, neštekajú, b) nerobia kôpky a mláčky, nepotrebujú venčiť, c) s kýmkoľvek sa ochotne hrajú najrôznejšie hry, d) narozdiel od ostatných členov domácnosti sa dajú hocikedy vypnúť, e) môžeme ich chovať bez akýchkoľvek problémov a sporov so susedmi aj v špecifických podmienkach panelákov, f) v prenosnom vydaní sú vhodné aj na rande do parku, či na dlhé cesty (ak nemáte na lietadlo) za poznaním."* (I. Kopeček, J. Kučera: *Programátorské poklesky. Mladá fronta, Praha 1989.*)

1. 1. TERMINOLÓGIA

Každý vedný odbor má svoje základné pojmy a metódy ich spracovania, hlavne však špecifikáciu toho, čo je predmetom jeho záujmu. Matematika sa zaoberá predovšetkým kvantitatívnymi a priestorovými vzťahmi, fyzika skúmaním podstaty a spôsobov zmien energie, biológia existenciou a fungovaním biologických systémov a procesov... Informatika je v porovnaní s týmito klasickými prírodovednými disciplínami dieťaťom v plienkach. Kým matematika, fyzika, biológia... sa vyučujú stovky až tisíce rokov, informatika nemá ako vedná disciplína ani päťdesiat (a ako predmet vyučovania ešte menej). Prečo sa potom dostala do takej pozornosti spoločnosti, prečo by mal jej základy ovládať každý (stredoškolsky) vzdelaný človek?

Všimnime si, čo je dnes dôležité pre každého z nás. Okrem klasických poznatkov, zručností a návykov sa musíme dokázať orientovať v množstve najrôznejších informácií (aj dezinformácií), efektívne využívať rôzne databázy údajov, komunikovať pomocou najrozličnejších technických prostriedkov nielen v regionálnom, ale aj v celosvetovom rámci. Dnešná spoločnosť prežíva informačnú revolúciu; bez potrebných znalostí a prostriedkov nie je možné udržať krok so svetom nielen vo vede, ale ani v obchode, priemysle, politike... Kľúčom k úspechu by mohla byť práve znalosť informatiky zaoberajúcej sa spracovaním informácií z rôznych zorných uhlov.

Pri chápaní informatiky, jej poslania a cieľov dochádza niekedy k deformácii. Väčšina laikov redukuje informatiku a jej vyučovanie len na znalosť a zručnosť práce s počítačom, prípadne výpočtovou technikou. Je to veľmi podobné situácii, keď by sme matematiku a jej vyučovanie "povýšili" iba na počítanie. "Veď z toho, čo sme sa učili v matematike v škole, dnes potrebujem iba sčítať, odčítať, násobiť, niekedy deliť, najviac desatinné čísla a vedieť si vypočítať nejaké percentá," - tvrdí veľa ľudí. Potom by ale stačilo, keby sme sa na matematike učili iba prácu s kalkulačkou?! (*Úloha*: Zistite, kedy sa objavili prvé kalkulačky! Pozn. Vieme isto, že Robinson Crusoe ich ešte nemal k dispozícii.[2])

Vyučovanie (učenie sa) matematiky však sleduje iný cieľ, ktorý má podstatne väčší význam pre rozvoj osobnosti: učí človeka koncepčne a logicky rozmýšľať, dávať si do súvislostí logické vzťahy medzi presne definovanými objektmi, dokázať riešiť úlohy teoretického, ale aj praktického charakteru. Podobné je to aj s vyučovaním (učením sa) informatiky. Výpočtová technika je iba prostriedkom, ktorý umožňuje efektívne realizovať informatické poznatky. Informatika aj výpočtová technika sa veľmi úzko ovplyvňujú, spolu vytvárajú jeden celok a vzájomne sa rýchlo posúvajú dopredu. Informatika okrem poznatkov z hardware a software výpočtovej techniky dáva teoretické a praktické poznatky o tom, ako spracová-

---

[2] *Keby ste náhodou nenašli príslušný zdroj informácií: Kalkulačka v dnešnom ponímaní sa objavila okolo r. 1970, kedy sa začali vyrábať LCD-displeje - zobrazovače, ktoré umožnili vybaviť kalkulačky menej energeticky náročným realizátorom a zobrazovačom činnosti.*

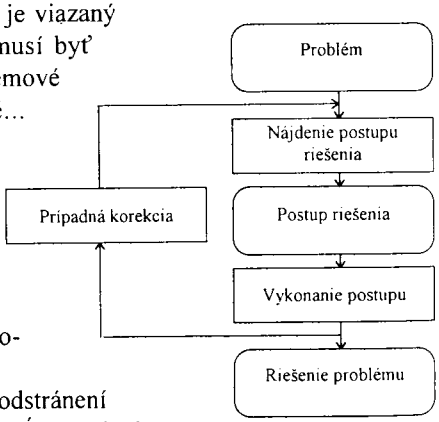
vať, uchovávať, organizovať, prenášať a vyhľadávať informácie ľubovoľného druhu, rozvíja tiež schopnosti efektívne využívať rôzne technické prostriedky pre spracovanie informácií. Navyše umožňuje rozvíjať systematické a koncepčné myslenie, využívať obmedzenú sadu nástrojov pre riešenie daného problému – teda riešiť problémy. A to je jeden z najdôležitejších výsledkov, ktoré nám vyučovanie (učenie sa) informatiky môže priniesť.

Dnes sa často objavuje pojem informačné technológie. Zjednodušene si ich môžeme predstaviť ako spôsoby, metódy a prostriedky spracovania informácií najrôznejšieho druhu s využitím modernej techniky a spôsoby prenosu týchto informácií na veľké vzdialenosti. Informačné technológie sú istým spôsobom akýmsi praktickým zastrešením čiastkových poznatkov informatiky a výpočtovej techniky. Vzhľadom na iné ciele sa im však v tejto knihe venovať nebudeme.

1. 2. MÁM PROBLÉM, ČO S NÍM?

Kto dnes nemá problémy? Bolo by však dobré, keď sa chceme zaoberať riešením problémov, presne si používané pojmy charakterizovať. Pokúsme sa o to. Pripomínam však, že charakteristiky pojmov sú iba opisné, ich presné "definovanie" je dosť problematické.

Problém je stav, v ktorom jestvuje rozdiel medzi tým, čo v danom momente poznáme (vieme, máme), a tým, čo potrebujeme. Inak povedané: disproporcia medzi možnosťami a cieľom. Problém je viazaný na jeho "majiteľa" (pre iného to nemusí byť problém, ale nezmysel) a na isté problémové prostredie (citové, finančné, školské... problémy). V našom prípade sa zameriame na hľadanie riešení problémov z oblasti spracovania informácií.



Riešenie problému chápeme vo význame splnenia cieľa, t. j. odstránenia disproporcie. Potom môžeme schematicky znázorniť postup vyriešenia problému ako na obrázku.

V oválach sú uvedené stavy, ktoré pri odstránení problému potrebujeme absolvovať, v obdĺžnikoch činnosti, ktoré musíme pre zmeny od problému k riešeniu vykonať. Šípky naznačujú postupnosť zmien stavov a činností.

Všimnime si činnosti *Nájdenie postupu* a *Vykonanie postupu*.

Činnosť *Nájdenie postupu* je činnosť tvorivá – je potrebné aktívne premyslieť jej podrobnosti a postupné kroky. Vymyslením postupu riešenia je možné poveriť iba človeka, zatiaľ (pre človeka našťastie) nejestvuje také technické zariadenie, ktoré by bolo schopné tvorivo myslieť.

Inak je to s činnosťou *Vykonanie postupu*: Táto je rutinnou činnosťou v prípade, že postup je už daný alebo známy. Jej vykonaním môžeme poveriť niekoho, kto je schopný daný postup realizovať – či už človeka, alebo stroj. Nemusí rozmýšľať, stačí, ak bude daný postup presne realizovať. Takémuto zariadeniu sa hovorí *procesor*. (Pozor! Procesor je v tomto význame oveľa širší pojem ako zaužívané označenie pre technickú jednotku – súčasť hardware počítačov.) Ak uvažujeme, že budeme riešiť problémy z oblasti spracovania informácií, potom procesorom spôsobilým vykonávať nami vytvorené postupy môže byť buď človek, alebo počítač.

1. 3. ALGORITMUS

Je samozrejmé, že postup musí byť určený pre nemysliace zariadenie, ktoré vôbec nevie, čo má byť výsledkom jeho realizácie. Preto máme isté obmedzenia, na ktoré musíme pri formulácii postupu myslieť. Postupu určenému pre nemysliace zariadenie (procesor) hovoríme **algoritmus**.

Algoritmus[3] je elementárnym pojmom informatiky. Elementárnym v tom zmysle, že nie je možné ho opísať pomocou ešte elementárnejších pojmov, podobne ako v geometrii bod. (Poznámka: Nie je to až tak úplne pravda, ale pre naše potreby toto priblíženie postačuje.)

Preto algoritmus definujeme iba opisne. Tu je jeden z možných opisov:

Algoritmus je postup, ktorého realizáciou získame zo zadaných vstupných údajov po konečnom počte činností v konečnom čase správne výsledky.

Pre upresnenie toho, či je postup algoritmom, sa používajú doplňujúce vlastnosti, ktoré znovu môžeme uviesť iba opisne:

Vlastnosti algoritmu

P1. Elementárnosť – Postup je zložený z činností, ktoré sú pre realizátora elementárne, zrozumiteľné.

P2. Determinovanosť – Postup je zostavený tak, že je v každom momente jeho vykonávania jednoznačne určené, aká činnosť má nasledovať, alebo či sa už postup skončil.

P3. Rezultatívnosť – Postup dáva pre rovnaké vstupné údaje vždy rovnaké výsledky (ak skončí).

P4. Konečnosť – Postup skončí vždy v konečnom čase a po vykonaní konečného počtu činností.

---

[3] *Meno algoritmus má zaujímavý pôvod: Vzniklo latinským prepisom mena arabského matematika Abú Jáfar Mohamed Ibn Músa al-Chworezmiho, ktorý okolo r. 825 napísal knihu o riešení rovníc v desiatkovej číselnej sústave s názvom "Kitab al jabr w'al-mugabala" (podčiarknuté vám iste, a nie náhodou, pripomína slovo "algebra"). Kniha v latinčine začínala vetou "Algorithmi dici... ", t. j. "Al-Chworezmi hovorí... " (Pri preklade z arabských "klikihákov" nemusí meno byť vždy rovnaké.)*

P5. Hromadnosť – Postup je aplikovateľný na celú triedu prípustných vstupných údajov.

P6. Efektívnosť – Postup sa uskutočňuje v čo najkratšom čase a s využitím čo najmenšieho počtu prostriedkov.

Týchto "šesť pé" sa nemusí na prvé prečítanie zdať dôležitými, ale vzťahujú sa predovšetkým na to, aby postup mohlo realizovať nemysliace zariadenie. Toto si nevie uvedomiť, že postup sa vykonáva podozrivo dlho, nevie experimentovať, nemá žiadne skúsenosti, neučí sa z chýb (cudzích ani vlastných). Vlastnosť *P6 (Efektívnosť)* je viac-menej relatívna. Najlepšie sa o tom môžeme presvedčiť pri rôznych algoritmoch triedenia. Hľadanie efektívnych algoritmov je jednou z najdôležitejších oblastí informatiky.

Venujme sa teraz podrobnejšiemu opisu vlastností spolu s uvedením príkladov postupov, ktoré sú nám známe zo života a nie sú (v používanom tvare) algoritmami.

P1. ELEMENTÁRNOSŤ

Jeden postup môže byť zapísaný rôznymi spôsobmi, a tak (ne-)zrozumiteľný pre prípadných realizátorov. Čo je elementárnou činnosťou pre jedného, nemusí byť pre iného.

Príklad 1.1. Už deti na prvom stupni základnej školy vedia násobiť. Ale skúste im povedať: "Zistite šiestu mocninu čísla 2!" Asi nebudú vedieť reagovať. Keď im zadáme úlohu v tvare: "Zistite výsledok súčinu 2.2.2.2.2.2!", bude to pre nich hračkou. Zistenie mocniny pre nich totiž nie je elementárnou činnosťou.

Príklad 1.2. Pre murárskeho učňa je iste elementárnou činnosťou pri murovaní "zarobenie kalfasu dobre mastnej malty". Obávam sa, že pre väčšinu z vás (vrátane mňa) to však bude značný problém.

Príklad 1.3. Mám veľmi rád kuchárske recepty, hlavne ich produkty. Realizácia je však pre mňa často problémom. V jednom z receptov je napríklad elementárnou činnosťou "Zarob bešamel", v inom "Meľ dva dni staré rožky", v ďalšom "Priprav marinádu"!? A čo tak "Pridaj štipku soli" alebo "Pridaj do toho dve celé vajcia"? Samé problémy s elementárnosťou, často však subjektívne.

Človek má jednu výhodu – dokáže sa učiť a vytvárať si stále zložitejšie a zložitejšie elementárne činnosti, ktoré už potom kombinuje do ešte zložitejších postupov. Takúto vlastnosť však nemajú počítače, majú iba obmedzenú sadu elementárnych činností z oblasti spracovania informácií.

P2. DETERMINOVANOSŤ

Zdanlivo nepochopiteľná vlastnosť – presné určovanie poradia činností, pomáhať si umelými odkazmi, čím pokračovať ako ďalšou činnosťou (príklad 2.1). Ľudia

podvedome či cieľavedome dokážu chápať postupy, v ktorých nie je zvýraznená riadiaca zložka, teda poradie vykonávaných činností, prípadne, čo a ako dlho opakovať. Pre nemysliace zariadenie je však uvdenie poradia nevyhnutnosťou.

Príklad 2.1. Popíšme si postup pri prechode cez ulicu. Nie je to veľmi jednoduchá úloha v prípade, že by sme chceli uvažovať všetky možnosti. Neuvažujme teraz svetelné križovatky, a pre istotu ani fakt, že by vodiči mali dať chodcovi na "zebre" prednosť. (Naivne veriť predpisom sa v tomto prípade dá iba raz.)

Tu je jeden pokus o zápis postupu:

1. Pozri doľava.
2. Ak ide auto, opakuj krok 1.
3. Prejdi do polovice ulice.
4. Pozri doprava.
5. Ak ide auto, opakuj krok 4.
6. Prejdi na druhú stranu ulice.

Postup by bolo možné zapísať aj ináč, ale je tu zvýraznené použitie špeciálnych príkazov na riadenie postupu "opakuj krok...", ktorým sa v programovaní ťažko vyhneme. Navyše postup určite nespĺňa všetky podmienky zrozumiteľnosti, ba aj bezpečnosti. Čo ak sa blíži autobus? Skúste napísať lepší postup na bezpečný prechod cez ulicu.

P3. REZULTATÍVNOSŤ

Znovu zdanlivo nezmyselná vlastnosť - ako môže byť výsledkom viacerých realizácií postupu s rovnakými vstupnými údajmi rôzny výsledok? Ale pozor! Nie je to v bežnom živote až tak nezvyčajné! Veď nie nadarmo sa používa výrok "Keď robia dvaja to isté, nemusí to byť to isté." Ešte neveríte?

Príklad 3.1. Skúste si vybrať ľubovoľný kuchársky recept a urobiť jedlo presne tej istej chute. Vždy keď sa v ňom objavia nejaké približné údaje, napr. ako je potrebné jedlo variť či piecť a pod., nebude to to isté. Aj najlepšej gazdinej sa z času na čas niečo pripáli, niekedy je slané, inokedy plané... Ovplyvňujú to, samozrejme, tiež subjektívne okolnosti, ale ani ten plyn, na ktorom sa jedlo pripravuje, nemusí byť vždy rovnako silný.

Príklad 3.2. Častá situácia v škole: Začiatok prestávky po písomke z matematiky, a tým aj začiatok zisťovania, komu ako ktorý príklad vyšiel. Mali by predsa vyjsť rovnako! Ale je vopred isté, že to tak nebude, často nie sú ani dve riešenia v celej triede (ak sa nedá opisovať) rovnaké. Podobne je to aj v živote. Aj keď ste si istí, že sa správate rovnako ako minule, efekt (často aj afekt) môže byť úplne iný.

Vo všetkých týchto prípadoch je realizácia postupu závislá na subjektívnych aj objektívnych podmienkach, ktoré nedokážeme ovplyvniť. Vieme však zo skúsenosti, že s tým musíme počítať. O to je život zaujímavejší. [4]

P4. KONEČNOSŤ

Skúsenosti nám umožňujú prerušiť nejaký postup v momente, kedy vidíme, že nevedie k požadovaným výsledkom. Skúsenosti však od nemysliaceho zariadenia nevyžadujeme, preto zostáva na nás formulovať postup riešenia problému tak, aby určite skončil. Inak by mohli nastať situácie podobné uvedeným príkladom:

Príklad 4.1. Časť zo zaručeného návodu na nájdenie pokladu: "Poklad nájdeš, keď pôjdeš... Kop na tom mieste, kým nenarazíš na poklad."

Máte smolu, ak poklad už niekto vybral alebo tam nikdy nebol. Nemysliace zariadenie by sa tým asi zamestnávalo do konca svojej kariéry.

Príklad 4.2. Návod bez zmyslu, ale môže byť podobný niektorým vami vytvoreným programom:

1. Mysli si číslo.
2. Pokiaľ sa nebude rovnať 1, odčítaj od neho 2.

Netreba ani zvýrazňovať, že v prípade zadania desatinného, záporného alebo párneho čísla bude nemysliace zariadenie opakovane odčítať jednotku "pomerne dlho".

Je aj iný druh "nekonečnosti". Niektoré metódy, ktoré sú teoreticky konečné a algoritmicky správne, môžu trvať tak dlho, že sú vlastne prakticky nerealizovateľné:

Príklad 4.3. Majme zistiť počet zrniek piesku, ktoré sú na pláži.

Môžeme použiť najmenej dva postupy:

I. Urobiť si čisté miesto a zrnko po zrnku za neustáleho počítania tam prenášať.

II. Spočítať približný počet zrniek na lopate a prehádzať ňou celú pláž, počet lopát si pamätať.

Iste by ste našli aj lepšie riešenia. Väčšinou je potrebné vhodne zjednodušiť problém a riešiť ho iba približne - tak, že uvažujeme s istou chybou. Podobných úloh je možné nájsť viac.

P5. HROMADNOSŤ

Táto vlastnosť už patrí skôr k užitočným ako nevyhnutným. Existujú aj jednoúčelové postupy, ktoré nemajú premenlivé vstupné údaje, a pritom sú

---

[4] *Ako by vám vyhovovalo, keby platila epizóda zo Židovských anekdot Karla Poláčka: Pán Kohn stretne na ulici neznámeho pána, ktorý ho uctivo pozdraví: "Dobrý deň, pán Kohn!" "Prepáčte, ale ja vás nepoznám. Odkiaľ ma vy poznáte?" - čuduje sa pán Kohn. "Ja som si vás vypočítal!"*

algoritmami. Takýmito príkladmi môžu byť nastaviteľné, ale po voľbe už pevné, programy pre automatickú práčku, výrobu súčiastok na NCR strojoch a pod.

Príklad 5.1. Naučiť niekoho násobiť dve prirodzené čísla je niečo iné ako naučiť ho, koľko je 2x2. Cieľom je naučiť postup, ako medzi sebou násobiť ľubovoľné dve prirodzené čísla. A tento postup môžeme zovšeobecniť na násobenie dvoch ľubovoľných celých, racionálnych, ba dokonca až reálnych čísiel.

Hromadnosť postupu je teda skrytá v tom, že postup pripúšťa premenlivé vstupné údaje a umožňuje nám riešiť úlohy podobného typu.

P6. EFEKTÍVNOSŤ

Efektívnosť algoritmu je viac-menej doplnková, niekedy však veľmi potrebná vlastnosť. Zvlášť pri veľmi veľkom počte spracovávaných údajov, ako aj tam, kde potrebujeme veľký počet zmien spracovávaných informácií. Často je cieľom najskôr zostaviť hocijaký, ale funkčný algoritmus, potom ho v prípade potreby a s lepšou znalosťou problému vylepšovať. Ako kritériá efektívnosti slúžia časová a pamäťová zložitosť algoritmu, ktorým sa ešte budeme venovať. Uveďme si dva príklady pokusu o efektívnosť algoritmu (postupu).

Príklad 6.1. Prechod cez ulicu. Túto úlohu sme už mali. Ale máme aj jedno pre zápis určite efektívne riešenie, ktoré vytvoril žiak 5. triedy základnej školy (dnes prezident slušne prosperujúcej počítačovej firmy):

1. Choď cez ulicu tak, aby ťa nič nezrazilo.

Je jasné, že za toto riešenie získal plný počet bodov, aj keď je určené skôr pre človeka, ktorý má už svoje skúsenosti s dopravnou situáciou. Efektivita zápisu a skoro geniálny nápad sú však zrejmé.

Príklad 6.2. Na vyučovaní matematiky dostali asi 10-roční žiaci úlohu spočítať prvých 50 prirodzených čísiel. (Učiteľ si chcel v kľude prečítať noviny, a keby nestihol, zvýši počet na 100.)

Žiaci postupovali väčšinou podľa jeho predpokladov: sčítali postupne 1+2+3+4+5+... a trvalo im to úmerne ich "sčítacím" schopnostiam.

Našiel sa však žiak, nazvime ho malý Gauss[5], ktorý prevrátil plány učiteľa naruby. Nekonal hneď, ale rozmýšľal: 1+50=51, 2+49=51, 3+48=51,... , 25+26=51 a takýchto dvojíc je polovica z počtu čísiel, teda 25. Vynásobenie 25x51=1275 bolo otázkou okamihu.

---

[5] *Karl Friedrich Gauss (1777-1855) - jeden z najvýznamnejších prírodovedcov 19. storočia, nemecký matematik, fyzik, geofyzik a astronóm, univerzitný profesor a riaditeľ hvezdárne v Göttingene.*

Prichádza hneď do úvahy zovšeobecnenie riešenia tak, aby bol postup hromadný, teda riešil problém pre ľubovoľný počet prvých prirodzených čísel N. Tu je:

```
SÚČET = N/2 x (N+1)
```

Jeho efektívnosť je optimálna – nech je počet prirodzených čísiel ľubovoľne veľký, pre zistenie súčtu stačí jedno sčítanie, jedno násobenie a jedno delenie. Nie všetky problémy, dokonca ani matematické, sa však dajú previesť do tvaru vzorcov.

Vidíme, že je nevyhnutné, aby sme si presnejšie určili, čo má byť vstupom a čo výsledkom algoritmu. Preto sa dohodnime, že zadanie algoritmu budeme zapisovať takto:

```
{VST: vstupné podmienky}
                ?
{VÝS: výstupné podmienky}
```

kde na začiatok doplníme vstupné podmienky - vzťahy, ktoré platia na začiatku realizácie algoritmu, a na konci určíme výstupné podmienky - čo má byť výsledkom realizácie algoritmu.

Na záver tejto kapitoly si bez veľkého vysvetľovania uveďme ešte niektoré pojmy, s ktorými budeme v ďalšom texte pracovať.

ALGORITMIZÁCIA

Algoritmizáciou rozumieme schopnosť aktívne vytvárať algoritmy určené pre nemysliace zariadenie. Je nevyhnutnou súčasťou schopnosti programovať na počítačoch. Tak ako iba teoretická znalosť jazyka nestačí, aby človek mohol komunikovať, tak ani zvládnutie iba prostriedkov algoritmického či programovacieho jazyka "od A po Z" nestačí na to, aby človek vedel vytvárať efektívne a správne algoritmy. Je na to potrebná aj dôkladná znalosť problémového prostredia, skúsenosť s formulovaním algoritmov a schopnosť využiť obmedzené prostriedky konkrétneho jazyka, resp. techniky.

Čo je to správny algoritmus? Môžeme sa dohodnúť na týchto termínoch:

- Algoritmus nazývame *čiastočne správny*, ak v prípade, že skončí, dáva vždy správne výsledky.
- Algoritmus nazývame *konečný*, ak skončí v konečnom čase pre ľubovoľné vstupné údaje.
- Algoritmus nazývame *správny*, ak je čiastočne správny a konečný.

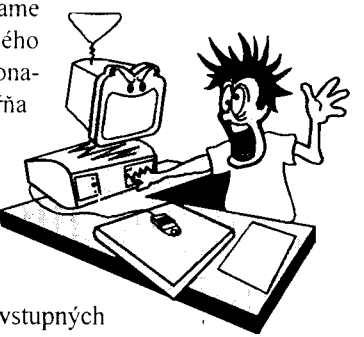
ALGORITMUS A PROGRAM

Aký je vzťah medzi algoritmom a programom? Pod *programom* chápeme algoritmus napísaný v programovacom jazyku. Oproti algoritmu obsahuje navyše

ďalšie inštrukcie pre počítač, predovšetkým vzťahujúce sa k určeniu typov spracovávaných údajov, využívaniu hardware, prípadne ďalšiemu softwaru počítača. Veľa programov spolupracuje s rôznymi doplnkovými súbormi, ktoré obsahujú pomocné údaje pre činnosť programu, napr. obrázky, hudbu, tabuľky výsledkov... Preto je pri tvorbe zložitejších celkov vhodnejšie používať pojem *programový produkt*.

PROGRAMOVANIE

Programovanie je konštruktívna myšlienková, ale aj praktická činnosť, kedy vytvárame nové programové produkty realizovateľné na počítači. Programovaním sa učíme predovšetkým myslieť, organizovať svoje myšlienky a dokázať ich realizáciou poveriť počítač. A nielen to. Poznávame tým lepšie aj samého seba ako tvora nedokonalého a s radosťou rozvíjajúceho svoje chyby až k dokonalosti. Reálne programovanie na počítačoch zahŕňa v sebe viacero rovnocenných súčastí, bez vhodného skĺbenia ktorých nebýva výsledok oslňujúci.



Vytvorenie programu pozostáva z týchto činností:

- Algoritmizácia daného problému – určenie vstupných a výstupných podmienok.
- Vytvorenie programu (programového produktu) a vhodnej programovej dokumentácie.
- Zapísanie a odladenie programu priamo na počítači.

V mnohých prípadoch navonok skúsení programátori postupujú pri programovaní systémom "sadnem a píšem". Neberte si z nich príklad, pretože skôr či neskôr sa stratíte v záplave činností, ktoré je potrebné mať jednak dobre premyslené, jednak rutinne zvládnuté. A pri tvorbe zložitejších produktov je nevyhnutná dôkladná algoritmická a systémová príprava, ktorá jediná môže minimalizovať čas strávený pri počítači a vyhnúť sa nekonečnému prepracovávaniu zdanlivo už hotových častí.

2. JAZYK, ALGORITMICKÝ JAZYK, PROGRAMOVACÍ JAZYK

Ak máme s niekým komunikovať, potrebujeme zodpovedajúci prostriedok dorozumievania – jazyk. Jazyk je podľa J. Mistríka (Jazyk a reč, Mladé letá, Bratislava 1984) "súhrn pravidiel, na základe ktorých vzniká reč". Pôvodne bol jazyk iba prostriedkom komunikácie medzi ľuďmi[6], dnes už je aj prostriedkom komunikácie medzi človekom a strojom. Pozrime sa na jazyk tohto typu podrobnejšie.

Potrebujeme niekoho naučiť postupy – algoritmy. Pretože algoritmy sú postupy so špecifickými vlastnosťami, bolo by potrebné použiť aj špecifický jazyk. Jazyk určený pre zápis algoritmov nazývame algoritmický jazyk. Jazyky používané pri komunikácii medzi ľuďmi nevyhovujú z viacerých dôvodov:

a) Počet slov je neúmerne vysoký (napr. slovenčina pozná vyše 110.000 slov, angličtina takmer 800.000). Navyše sú tieto jazyky v neustálom vývoji, ročne pribúdajú desiatky nových slov.

b) V ľudských jazykoch existuje veľa výnimiek spôsobených historickým vývojom, zdanlivo nezmyselné príslovia (napr. "Lož má krátke nohy.", "Kúpil mačku vo vreci."), prirovnania, zaužívané slovné spojenia (napr. "to je babylon", "kocky sú hodené", "medvedia služba").

c) Existencia homoným (slov, ktoré majú viacero významov, napr. "strana", "koruna", "list", "dospievať", "mať"...) a synoným (rôznych slov, ktoré majú rovnaký význam, napr. "najmä" - "hlavne" - "predovšetkým" - "najskôr") môže spôsobovať nejasnosť vysvetlenia. (Obľúbené sú slovné hračky a tzv. dvoj- či en-zmysly.)

d) Niekedy nie je možné ani z kontextu odhadnúť, čo dané slovo vyjadruje. Napr. úryvok z náhodne vypočutého rozhovoru: "Jano išiel do mesta. Mesiac sa neukázal." (cit. z výbornej knižky Hvorecký, Kelemen: Algoritmizácia) Je slovo "Mesiac" príslovkovým určením času alebo podmetom druhej vety? Môžeme iba odhadovať podľa znalosti veci (Jana) alebo hlasovať. Podobné je to s obľúbenou otázkou majora Teraskyho: "Čím je vojak?" (Vyberte si z možností "Obrancom vlasti" či "Lyžicou".)

e) Prirodzený jazyk obsahuje veľa prvkov a konštrukcií, ktoré sú pri formulovaní postupov zbytočné, napr. nič nehovoriace debaty a zdvorilostné otázky ("Už ani to počasie nie je, čo bývalo." alebo "Ako sa máš?" – aj keď mi je to úplne ukradnuté, čo sa však nehovorí, a pod.)

---

[6] *Na svete existuje 6528 známych, aj keď niekedy už nepoužívaných jazykov. Najviac ľudí hovorí čínsky – 726 miliónov, ďalej nasledujú angličtina, španielčina, hinština, arabčina, portugalčina,... , ruština je ôsma, nemčina desiata, francúzština jedenásta. Svetovým dorozumievacím jazykom je angličtina, ktorou sa bude vedieť dohovoriť okolo roku 2000 každý štvrtý človek na svete. (údaje z r. 1997)*

Tieto dôvody viedli k potrebe vytvoriť formálne jazyky – umelo vytvorené a špeciálne určené pre zápis algoritmov. Podobne ako sa nepodarilo ľudstvu dohovoriť sa na jednotnom jazyku, ani pri algoritmických jazykoch nedošlo k dohode a vo všeobecnosti sa používa viacero z nich. Najčastejšie sú:

A. *Grafické* algoritmické jazyky – napr. vývojové diagramy, rôzne typy štruktúrogramov,

B. *Lineárne* algoritmické jazyky – napr. slovný zápis v národnom jazyku, programovací jazyk.

Algoritmický jazyk by mal svojou konštrukciou napomáhať splneniu vlastností algoritmu. Preto v jestvujúcich algoritmických jazykoch sú dve zvýraznené zložky – *operačná a riadiaca.*

Operačná zložka

Obsahuje sadu prostriedkov, ktoré umožňujú spracovávať údaje – elementárne činnosti, ktoré dokáže procesor vykonávať. Pretože naším cieľom je prechod k programovaniu, ako základ budú pre nás slúžiť elementárne činnosti, ktoré sa používajú pri programovaní. Základnými činnosťami sú *príkazy* a *podmienky*.

Príkazy sú vety jazyka, ktoré prikazujú procesoru vykonať isté, presne stanovené činnosti. Pre začiatok vystačíme s príkazmi vstupu, výstupu a priradenia. Tieto príkazy musia spracúvať nejaké objekty. V programovaní sú nimi premenné, konštanty a výrazy.

Premenná je objekt, ktorý obsahuje počas realizácie algoritmu konkrétnu hodnotu[7] presne stanoveného typu (napr. celé číslo, reálne číslo, reťazec znakov...).

Konštanta je objekt, ktorý nadobúda počas celej realizácie algoritmu jedinú konkrétnu hodnotu príslušného typu. Je to obdoba konštánt známych z matematiky, napr. π, e, ale aj z fyziky: g, c, κ, ε, μ.

Výraz je predpis, ktorý obsahuje konštanty, premenné a spôsob ich spracovania pomocou operácií a funkcií podobných tým, ktoré poznáme z matematiky. Jeho výsledkom je hodnota príslušného typu, ktorá vznikne po vykonaní vo výraze naznačeného spracovania.

Príkaz priradenia má tvar:

```
p := v
```

kde *p* je meno premennej, *v* je výraz. Vykonaním príkazu nadobudne hodnota premennej *p* hodnotu výrazu umiestneného na pravej strane priradenia. (Často sa tento príkaz pletie s rovnosťou známou z matematiky, tá sa v programovaní využíva v podmienkach.)

[7] *Premenná v informatike a v matematike majú odlišný význam. V matematike je premenná chápaná ako symbol, t. j. zástupca celej triedy hodnôt určitého typu, napr. celých čísiel, nemusí to byť žiadna konkrétna hodnota. V informatike (presnejšie v programovaní) je premenná pamäťové miesto príslušnej veľkosti, ktoré vždy obsahuje istú hodnotu z určeného typu údajov. Je to teda vždy momentálna hodnota, ktorá je pod menom premennej uložená.*

Meno premennej budeme zapisovať tak, že musí začínať písmenom (anglickej abecedy "A" až "Z", resp. "a" až "z" - bez "príkras" – dĺžňov, mäkčeňov...) a môže obsahovať ľubovoľný počet písmen a číslic. Je vhodné voliť také mená premenných, ktoré napovedia čitateľovi, aký má premenná význam. Malé a veľké písmena nerozlišujeme, ale je vhodné, ak si "dôležité" premenné budeme označovať veľkými, pomocné malými písmenami. Takže menami premenných môžu byť A, B, C, podiel, b1, pomoc, alfa... , ale nie 1A, cS, žvacheľ a pod.

Riadiaca zložka

V algoritmickom jazyku musí byť presne stanovené poradie vykonávania jednotlivých činností. Je to potrebné preto, aby realizátor (procesor) nemusel uvažovať, čo má kedy vykonať. Oproti prirodzenému jazyku sa preto do algoritmu vkladajú riadiace príkazy a činnosti, ktoré určujú presnú postupnosť vykonávania jednotlivých činností. Časom sa vyvinuli najefektívnejšie prostriedky pre organizovanie postupnosti vykonávania činností známe ako základné algoritmické konštrukcie. Preto sa im venujme podrobnejšie spolu s uvedením príkladu konkrétneho algoritmického jazyka.