Základné pojmy a princípy programovania

Program je postupnosť príkazov popisujúcich nejakú činnosť. Každý program vyžaduje autora, ktorý ho píše, a *procesor*, ktorý ho zrealizuje. Program preto musí byť napísaný v jazyku, ktorému rozumie ako autor, tak aj procesor.

Realizácia programu sa skladá z realizácie jednotlivých príkazov a nazýva sa *proces*. Ak je realizácia príkazov sekvenčná, t.j. ak sa začne každý príkaz realizovať až po realizácii predchádzajúceho príkazu, hovoríme o *sekvenčnom procese*. Okrem sekvenčných procesov existujú aj *paralelné procesy*, v ktorých realizácia príkazov môže byť súbežná.

Každý proces realizuje operácie nad nejakými objektmi a má určitý efekt, spočíva vo vopred požadovanej transformácii objektov. Objekty, s ktorými pracujú procesy vo výpočtovej technike, sa nazývajú *údaje*. Vlastnosti údajov, napr. ich označenie, štruktúra, množina možných hodnôt, prípustné operácie apod., sa v programe špecifikujú pomocou *deklarácií*.

Okrem deklarácií a *príkazov* pre spracovanie údajov sa v programe vyskytujú tiež príkazy, ktorými sa riadi nadväznosť akcií pri realizácii programu v závislosti od údajov. Pomocou týchto príkazov špecifikujeme vetvenie (t.j. determinovaný výber jednej alternatívy postupu z niekoľkých možných) a cykly (t.j. opakované realizovanie určitých príkazov).

Zatiaľ čo *program* sám je *statický*, *proces*, ktorý vzniká jeho realizáciou, je *dynamický*.

Programom je napr. aj kuchársky recept, podľa ktorého postupuje kuchár pri varení (tu kuchár = procesor, varenie = proces, jednotlivé ingrediencie = objekty a napr. kurča na paprike = žiadaný efekt).

Etapy tvorby programu:

1. Definícia problému: 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).

2. Riešenie problému chápeme vo význame splnenia cieľa, t.j. odstránenia disproporcie:

a) Nájdenie postupu riešenia 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.

b) Vykonanie postupu 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*.

Algoritmus

Postupu určenému pre nemysliace zariadenie (procesor) hovoríme algoritmus. Algoritmus je elementárnym pojmom informatiky. 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.*

Vlastnosti algoritmu

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

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

Skúste deťom na prvom stupni základnej školy 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.

Alebo elementárne činnosti z kuchárskych kníh: "Meľ dva dni staré rožky.", "Pridaj štipku soli.", alebo "Pridaj do toho dve celé vajcia.".

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).

Ako môže byť výsledkom viacerých realizácií postupu s rovnakými vstupnými údajmi rôzny výsledok? Nuž, nie nadarmo sa používa výrok "Keď robia dvaja to isté, nemusí to byť to isté."

Častá situácia v škole: Začiatok prestávky po písomke z matematiky, a tým aj začiatok zisťovaní, 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.

Realizácia postupu je teda závislá na subjektívnych a objektívnych podmienkach, ktoré nedokážeme ovplyvniť. Vieme však zo skúsenosti, že s tým musíme počítať. O to je život zaujímavejší. Ako by vám vyhovovalo, keby platila epizóda zo Židovských anekdot Karla Poláčka: Pán Kohn stretne na ulici neznámeho pána, ktorý ho úctivo pozdraví: "Dobrý deň, pán Kohn!" "Prepáčte, ale ja vás nepoznám. Odkiaľ ma vy poznáte?" - čuduje sa pán Kohn. "Ja som si vás vypočítal!"

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

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.

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

Táto vlastnosť už patrí skôr k užitočným ako nevyhnutným. Existujú aj jednoúčelové postupy, ktoré nemajú premenlivé vstupné údaje, a pritom sú algoritmami. Takýmito príkladmi môžu byť nastaviteľné, ale po voľbe už pevné, programy pre automatickú práčku, výrobu súčiastok na NCR strojoch a pod.

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

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

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.

Algoritmizácia

- schopnosť aktívne vytvárať algoritmy určené pre nemysliace zariadenie. Je nevyhnutnou súčasťou schopnosti programovať na počítačoch.

Čo je to správny algoritmus?

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.

Pod programom chápeme algoritmus napísaný v programovacom jazyku. Oproti algoritmu program obsahuje navyše ďalšie inštrukcie pre počítač, predovšetkým vzťahujúce sa k určeniu typov spracovávaných údajov, využívaniu hardvéru, prípadne ďalšiemu softvéru počítača.

Programovanie

- je konštruktívna myšlienková činnosť, 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.

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.

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, dnes už je aj prostriedkom komunikácie medzi človekom a strojom.

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 110000 slov, angličtina takmer 800000). 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 homonym (slov, ktoré majú viacero významov, napr. "strana", "koruna", "list", "dospievať", "mať",...) a synonym

(rôznych slov, ktoré majú rovnaký význam, napr. "najmä" - "hlavne" - "predovšetkým" - "najskôr") môže spôsobovať nejasnosť vysvetlenia.

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." Je slovo "Mesiac" príslovkovým určením času alebo podmetom druhej vety? Podobne 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?", a pod.)

Tieto dôvody viedli k potrebe vytvoriť formálne jazyky - umelo vytvorené a špeciálne určené pre zápis algoritmov. Najčastejšie sú:

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

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

V jestvujúcich algoritmických jazykoch sú dve výrazné zložky:

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. Tieto príkazy musia spracúvať nejaké objekty. V programovaní sú nimi premenné, konštanty a výrazy.

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

Konštanta je objekt, ktorý nadobúda počas celej realizácie algoritmu jedinú konkrétnu hodnotu príslušného typu.

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.

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.

Základné algoritmické konštrukcie

Sekvencia - postupnosť príkazov sa vyplní v poradí, v akom sú príkazy pod sebou zapísané.

Vetvenie (alternatíva) - v závislosti od splnenia podmienky sa postup vetví na rôzne prípady.

Cyklus - pri opakovaní nejakej činnosti musíme mať vyjasnené dve veci: čo sa má opakovať a dokedy sa to má opakovať. Činnosť, ktorá sa má opakovať, nazývame telom cyklu, podmienku, ktorá určuje dokedy sa bude telo cyklu opakovať, nazývame podmienka cyklu.

Vzťah medzi telom a podmienkou cyklu môže byť rôzny - podmienka môže predchádzať telu, cyklus sa môže vykonávať dovtedy, kým (ne-)bude splnená podmienka a pod. Najčastejšie sa používajú tieto typy cyklov:

1. Cyklus so známym počtom opakovaní: Telo cyklu sa opakuje vopred známy počet krát. Pre zisťovanie počtu už vykonaných opakovaní cyklu sa zavádza tzv. riadiaca premenná cyklu, ktorá nadobúda hodnoty od dolnej hranice po hornú hranicu (po "jednej").

2. Cyklus s podmienkou na začiatku: Najčastejšie sa volí cyklus (nazvime ho "opatrný"), kedy sa telo cyklu opakuje, pokiaľ platí podmienka cyklu. Podmienkou cyklu je tvrdenie, o ktorom dokáže procesor (ten, kto má algoritmus vykonávať) rozhodnúť, či je alebo nie je pravdivé. Tvar cyklu naznačuje, že najskôr sa kontroluje splnenie podmienky. Ak je splnená, vykoná sa telo cyklu, a potom sa znovu kontroluje splnenie podmienky. V momente, keď prvýkrát podmienka cyklu neplatí, telo cyklu sa vynechá a pokračuje sa v plnení príkazov nasledujúcich za cyklom.

3. Cyklus s podmienkou na konci: Je prakticky opačný ako cyklus s podmienkou na začiatku - najskôr sa vykoná telo cyklu. Potom sa zisťuje splnenie podmienky cyklu. Ak je splnená, vykonávanie cyklu sa ukončí, v opačnom prípade sa riadenie procesu znovu vráti na vykonávanie tela cyklu. (Tento cyklus môžeme označiť ako "neopatrný": najskôr sa niečo vykoná, potom sa rozhoduje, či to bolo dobre.)

Elementárne činnosti algoritmického jazyka:

Príkaz vstupu: Umožňuje zadať procesoru konkrétne hodnotu údajov, ktoré má spracovávať. Tieto hodnoty sa uložia do premenných (môžeme si ich predstaviť ako priehradky (skrine)) s

pevne stanovenou veľkosťou. V každom momente vykonávania algoritmu by mala byť v priehradke - premennej nejaká konkrétna hodnota.

Príkaz výstupu: Umožňuje získať od procesora výsledky algoritmu alebo iné oznamy (napr. oznam o tom, že sme zadali nesprávne vstupné údaje). Súhrnne tieto rôzne druhy výstupných informácií nazývame položky.

Priraďovací príkaz: Zmena hodnôt premenných je počas vykonávania algoritmu možná dvomi spôsobmi: príkazom vstupu alebo priradením novej hodnoty. Priraďovací príkaz nariaďuje procesoru, aby vykonal na jeho pravej strane naznačené operácie alebo funkcie a výsledok uložil do premennej, meno ktorej je na ľavej strane.

Podmienka: je v programovacích jazykoch chápaná ako logický výraz, t.j. zistenie vzťahov (relácie) medzi výrazmi, prípadne zviazané logickými operáciami (and="a súčasne", or="alebo" a not="neplatí, že").

Trochu histórie

Programovací jazyk pascal (nazvaný podľa Blaise Pascala (1623-1662) - významný francúzsky matematik, fyzik a filozof. Okrem iného sa zaoberal binomickými koeficientmi (Pascalov trojuholník), teóriou pravdepodobnosti, kužeľosečkami. R. 1642 zostavil počítací stroj na aritmetické operácie, skúmal hydrostatický tlak, formuloval Pascalov zákon.) vytvoril Niklaus Wirth (nar. 1934 - profesor informatiky na Vysokej škole technickej a Univerzite v Zürichu

(Švajčiarsko), popredný informatik, autor publikácií, ktoré sa stali klasickou výbavou pre programátorov) s cieľom zabezpečiť vyučovanie systematického programovania. Rozšíril tak "babylon" programovacích jazykov, ktorých v tej dobe vznikalo neuveriteľné množstvo. Ideou bolo vytvoriť programovací jazyk, ktorý by bol kompromisom medzi abstraktnými štruktúrami algoritmov a konkrétnou reprezentáciou spracovávaných údajov v počítači a nutnosťou poznať technické parametre. "Je jednoduchšie vytvoriť program, v ktorom sa manipuluje s pojmami, ako sú čísla, množiny, postupnosti alebo cykly, ako taký, v ktorom sa používajú pojmy bity, slová alebo skoky," znelo jeho motto.

Významnou črtou pascalu je jeho štruktúrovanosť, ktorá neumožňuje pri správnom programovaní vytvárať krkolomné monštrá príkazov, v ktorých sa nevyzná od istého momentu ani sám autor programu.

Tabuľka naznačuje krátky prierez vývojom programovacích jazykov, pričom obsahuje len tie známejšie:

obdobie	programovanie
40-te roky 20. storočia	programovanie v strojovom jazyku počítača
50-te roky 20. storočia	programovanie v jazyku symbolických adries
1956	programovací jazyk fortran (FORmula TRANslation)

1958	programovací jazyk algol (ALGOrithmic Language)
1961	programovací jazyk basic (Beginners All-Purpose Symblic Instruction Code)
okolo 1970	programovací jazyk pascal (systematické, štruktúrované programovanie)
okolo 1980	programovací jazyk C (prechod k objektovo orientovanému programovaniu)
90-te roky 20. storočia	vývoj komplexnejších verzií jazykov s cieľom využitia nových možností predovšetkým osobných počítačov (grafika, zvuk, multimédiá), programovanie riadené udalosťami (Visual Basic)
súčasnosť	programovacie jazyky pre tvorbu aplikácií v globálnych sieťach (Java)
budúcnosť	nechajme sa prekvapiť

Najstaršími a v hierarchii programovacích jazykov najnižšími sú strojové jazyky a ich symbolické verzie - jazyky symbolických inštrukcií (nazývané tiež jazyky symbolických adries alebo asemblery). Súhrne tieto jazyky nazývame strojovo orientované jazyky. Program zapísaný v strojovo orientovanom jazyku je

postupnosť elementárnych príkazov, ktoré nazývame inštrukcie a ktoré môže počítač priamo realizovať (každá inštrukcia zodpovedá jednej operácii procesora). Aj dosť dlhý program v tomto jazyku je teda dobre zvládnuteľný počítačom, ktorý mechanicky realizuje jednotlivé inštrukcie bez ohľadu na ich význam a dôsledky. Omnoho ťažšia je však práca programátora, ktorý sa musí zaoberať významom programu a stretáva sa teda s ťažkým problémom porozumenia dlhej postupnosti elementárnych inštrukcií.

Jednou z hlavných príčin nezrozumiteľnosti programov napísaných v strojovo orientovanom jazyku je ich neštruktúrovanosť. Tento nedostatok sa snažia postupne odstraňovať vyššie programovacie jazyky. Pre ich históriu je charakteristické hľadanie a zavádzanie vhodných štruktúr (operačných, riadiacich a údajových), ktoré zvyšujú zrozumiteľnosť programu a tým uľahčia programovanie. Súčasne s programovacími jazykmi sa však vyvíjali aj programy, nazývané prekladače, ktoré umožňujú prevod programov zapísaných v týchto jazykoch.

Typy údajov

Medzi bežné typy údajov patria napr. typ celých čísel, typ reálnych čísel a typ logických hodnôt. Pre hodnoty týchto typov je príznačné, že z hľadiska operácií prípustných vo vyššom programovacom jazyku sa javia ako ďalej nedeliteľné elementy. Preto sa tieto typy tiež nazývajú jednoduché typy údajov a premenné týchto typov jednoduché premenné. Pre špecifikáciu vlastností štruktúrovaných údajových objektov, akými sú napr. polia, sa v moderných programovacích jazykoch zavádzajú

štruktúrované typy. Typ údajov tvorí základ koncepcie údajov v jazyku Pascal.

Konštanty

Konštanta je údajový objekt, ktorého hodnota sa v priebehu výpočtu nemení. Podľa spôsobu identifikácie (označovania) konštanty v programu rozlišujeme dva druhy konštánt, a to priame konštanty alebo literály (25 3,14159 "Ahoj") a pomenované konštanty.

Premenné

Premenná je údajový objekt, ktorého hodnota sa môže v priebehu výpočtu meniť. Premenné sa najčastejšie identifikujú pomocou mena, t.j. pomocou identifikátora premenných. Pomenované premenné sa v programe zavádzajú deklaráciami. Deklaráciou premennej je pritom určený jej identifikátor, jej typ, ktorým je vymedzená množina prípustných hodnôt premennej. Prečo sa obory hodnôt premenných špecifikujú?

a) Znalosť oboru hodnôt premenných je dôležitá pre pochopenie programu.

b) Veľkosť pamäťového miesta, v ktorom bude uložená hodnota premennej, závisí od oboru hodnôt tejto premennej.: ak premenná môže nadobúdať celkovo n rôznych hodnôt, potom ich reprezentácia v pamäti musí obsahovať aspoň $\log_2(n)$ bitov. Znalosť oboru hodnôt premenných je teda dôležitá aj pre prekladač, ktorý premenným vymedzuje pamäťové miesta.

c) Operácie používané vo vyšších programovacích jazykoch sú obvykle definované iba pre určité typy hodnôt.

Deklarácia

Deklarácia slúži k zavedeniu a pomenovaniu určitých súčastí programu, napr. premenných, procedúr, funkcií apod. Dôležitou charakteristikou deklarácie je rozsah programu, na ktorý sa vzťahuje účinok deklarácie a v ktorom je možné deklarovanú súčasť použiť. Z tohoto hľadiska rozlišujeme globálne deklarácie, ktoré sa vzťahujú na celý program, a lokálne deklarácie, ktoré sa vzťahujú iba na časť programu.

Výrazy

Výraz je operačná štruktúra, pomocou ktorej sa vo vyšších programovacích jazykoch popisuje výpočet hodnoty. Výraz, na rozdiel od príkazu, je teda súčasne popisom akcie i označením hodnoty, ktorá je výsledkom tejto akcie. Výrazy v programovacom jazyku majú podobný tvar ako matematické výrazy: skladajú sa z operandov, operátorov, zátvoriek. Operandom môže byť konštanta, premenná alebo výsledok aplikácie funkcie. Operátory sa obvykle delia na unárne a binárne (podľa počtu operandov) a sú pre ne stanovené prípustné typy operandov, typy výsledných hodnôt a ďalej tzv. priority, podľa ktorých sa určuje poradie pri realizácii operácií vo výraze, v ktorom sa vyskytuje viac než jeden operátor. Medzi operátory patria aj relačné operátory, pomocou ktorých sa tvoria relácie. Relácia je výraz, ktorý nadobúda jednu z dvoch logických hodnôt podľa toho, či hodnoty operandov v

ňom uvedených sú či nie sú vo vzťahu, ktorý udáva relačný operátor.

Príkazy

Jednotlivé výpočtové akcie a ich náväznosti sa popisujú pomocou príkazov. Medzi jednoduché príkazy patrí predovšetkým priraďovací príkaz, popisujúci akciu priradenia hodnoty premennej. K štruktúrovaným príkazom patria zložený príkaz, podmienené príkazy a príkazy cyklu. Pre štruktúrované príkazy je charakteristické to, že sa skladajú zo základných príkazov a obsahujú tiež výrazy, udávajúce podmienky pre vetvenie či opakovanie.

Podprogramy

Systematický prístup k algoritmizácii zložitejších problémov spočíva v rozklade problému na niekoľko podproblémov a v samostatnej algoritmizácii týchto jednoduchších problémov. Vyššie programovacie jazyky umožňujú aplikovať tento prístup i pri písaní programu jeho rozkladom na podprogramy. Podprogram je popisom čiastkového algoritmu, v ktorom konkrétne údaje môžu byť zastúpené formálnymi parametrami. Vyvolanie podprogramu z nejakého miesta v programe sa realizuje odkazom na daný podprogram (tento odkaz môže mať formu príkazu alebo výrazu podľa toho, o aký druh podprogramu ide), pričom sa

súčasne stanovujú skutočné parametre, ktorými majú byť v podprograme nahradené formálne parametre.

Podprogramy sa delia na procedúry a funkcie. Procedúra je popisom algoritmu, ktorý nejakým spôsobom transformuje údaje. Funkcia popisuje algoritmus výpočtu jednej hodnoty, ktorá môže byť použitá ako operand vo výraze.