Čo by ste mali vedieť z druhého ročníka

Algoritmus je návod na riešenie problému. Môže byť zapísaný aj v slovenskom jazyku. Jednou zo základných vlastností algoritmu je hromadnosť, t.j. algoritmus musí byť návodom na riešenie celej skupiny úloh určitého typu. Všeobecnosť zápisu algoritmu sa dosahuje použitím premenných. Vykonávanie algoritmu sa realizuje vždy s konkrétnymi hodnotami premenných. Vykonávateľ algoritmu sa nazýva procesor.

Napríklad
```
čítaj(a,b,c)
V ← a*b*c
S ← 2*(a*b + a*c + b*c)
píš(V,S)
```
je algoritmus na výpočet objemu a povrchu kvádra za predpokladu, že sa za a, b a c dosadia kladné čísla (vstupná podmienka). Zrejme po konečnom počte krokov hodnoty v premenných V a S nadobudnú hodnoty objemu a povrchu kvádra so stranami a, b, c (výstupná podmienka).

Presnejšie: **Algoritmus** je postupnosť elementárnych príkazov, vykonanie ktorých pre prípustné vstupné hodnoty vedie mechanicky po konečnom počte krokov k výstupným údajom spĺňajúcim výstupné podmienky.

Program je algoritmus prepísaný do programovacieho jazyka a vykonateľný počítačom. Každý „poriadny“ program má vstup, výpočet a výstup. Vstup obsahuje zadanie vstupných hodnôt pre ktoré má prebehnúť výpočet. Vstupné údaje môžu byť na výzvu vložené užívateľom z klávesnice, zadané priradením hodnôt v programe alebo načítané zo súboru na disku. Výpočtom sa získajú nové údaje a výstup nám ich sprístupní.

Všetky názvy, ktoré si volí autor programu, sú identifikátory. **Identifikátor** je postupnosť písmen anglickej abecedy alebo číslic 0 až 9 začínajúca písmenom. Medzi písmená je dodefinovaný aj podčiarnik.

Najpoužívanejším príkazom je **príkaz priradenia**. Slúži na priradenie hodnoty premennej. Má tvar:

v algoritmizácii	Delphi	Python, C, Java	
premenná ← výraz	premenná := výraz	premenná = výraz	kde premenná je identifikátor

Šípka „←“ (aj „=“) sa číta „priraď“ a zdôrazňuje, že priradenie sa realizuje sprava doľava(!), ako o tom vypovedá aj popis vykonania príkazu priradenia: Vyhodnotí sa výraz na pravej strane a jeho hodnota sa priradí ako nová hodnota premennej na ľavej strane príkazu priradenia.

Príklad: Nech a, b a c majú hodnoty 1, 2 a 3, čo znamená, že v pamäťových miestach označených a, b a c sú uložené hodnoty 1, 2 a 3. Príkaz V ← a*b*c alebo V = a*b*c znamená, že sa vyberú aktuálne hodnoty z a, b a c, t.j. 1, 2 a 3, vynásobia a získaná hodnota 6 sa uloží do pamäťového miesta označeného V (predchádzajúca hodnota vo V bude nenávratne prepísaná).

Úloha: Napíšte príkazy priradenia, ktoré navzájom vymenia hodnoty dvoch premenných. Napr. nech v A je hodnota 7 a v B hodnota 5. Po vykonaní príkazov má byť v A hodnota 5 a v B hodnota 7.

Hodnotami premenných môžu byť reťazce, čísla alebo hodnoty False alebo True. **Reťazec** je postupnosť znakov uzavretá v úvodzovkách alebo apostrofoch. Napríklad "Python", 'Súčet čísel je', "Priemer = ". Prázdny reťazec neobsahuje znaky, jeho dĺžka (= počet znakov; označuje sa length) je nula a zapisuje sa "". Ak je hodnotou premennej číslo, môže to byť celé číslo označované integer alebo reálne číslo označované real alebo float. False a True sú hodnoty logického typu označovaného boolean.

Algoritmus aj program na výpočet objemu a povrchu kvádra obsahujú **príkazy, ktoré sa vykonajú za sebou v poradí, ako sú zapísané**. Je to typické pre jednoduché výpočty pomocou vzorcov (napr.: $c = \sqrt{(a^2 + b^2)}$; $s = ½.a.t^2$; $O = 2.\pi.r$ atď.) Takejto algoritmickej konštrukcii hovoríme **sekvencia** (postupnosť príkazov). Ak by sme však chceli aby sa príkazy pre výpočet objemu a povrchu kvádra vykonali len ak boli zadané kladné čísla, so sekvenciou nevystačíme. Musíme použiť **podmienený príkaz**, ktorý má vetvu „tak“, ktorej príkazy sa vykonajú, len ak je podmienka splnená a vetvu „inak“, kde sa uvádzajú príkazy, ktoré sa majú vykonať, ak podmienka nie je splnená.

Teda po príkaze vstup(a,b,c) by nasledoval príkaz
```
ak a>0 a b>0 a c>0
  tak začiatok
        V ← a*b*c
        S ← 2*(a*b + a*c + b*c)
```

```
        píš(V,S)
    koniec (vetvy „tak“)
  inak píš("Hodnoty a, b, c musia byť kladné čísla!")
```

Hovoríme, že algoritmus sa rozvetvuje a túto konštrukciu voláme **vetvenie**. S použitím podmieneného príkazu dokážeme vyriešiť mnoho ďalších úloh, napríklad po zadaní pH dokáže program vypísať, či je pre zadané pH prostredie kyslé, neutrálne alebo zásadité; vie tiež rozhodnúť, ktoré z troch zadaných čísel je najväčšie, vie ich zoradiť atď. Ale nevieme napríklad napísať algoritmus/program na výpočet mocniny x^n alebo na uhádnutie čísla pri zadávaní tipov od hráča. Obe úlohy vyžadujú použitie novej algoritmickej konštrukcie a to **cyklu** (požaduje sa opakovanie príkazov). Cyklus môže byť realizovaný **cyklom s pevným počtom opakovaní** alebo **cyklom s podmienkou**. Cyklus s pevným počtom opakovaní sa v programe realizuje príkazom for alebo **for-cyklom**. Cyklus s podmienkou sa realizuje príkazom while alebo **while-cyklom**. Pri výpočte mocniny x^n (x reálne, n prirodzené číslo) je počet opakovaní násobenia vopred pevne daný (napr. 3^5 sa vypočíta ako 3*3*3*3*3), čo sa najjednoduchšie realizuje príkazom for:

```
read(x,n)
sucin = 1
for i = 1 to n do sucin = sucin*x
write(sucin)
```

```
čítaj(x,n)
sucin ← 1
pre i od 1 až po n opakuj sucin ← sucin*x
píš(sucin)
```

Opäť by bolo viac ako žiaduce ošetriť vstup, lebo 0^0 nie je definované! Teda aspoň:

```
read(x,n)
if x=0 and n=0
then write("Nula na nultú nedefinované!")
else begin
        sucin = 1
        for i = 1 to n do sucin = sucin*x
        write(sucin)
    end (vetvy „else“)
```

S naprogramovaním hry Hádaj je to trochu inak. Nech si počítač vymyslí (vygeneruje) celé číslo napríklad od 1 po 100. Úlohou hráča je uhádnuť vygenerované číslo vkladaním tipov, pričom počítač na každý tip vypíše „Veľa, uber!“ alebo „Málo, pridaj!“ prípadne „Uhádol si!“. Opäť treba použiť cyklus (hráč opakovane zadáva tipy, kým neuhádne), ale nevieme, po koľkom opakovaní sa skončí. Vieme len, že cyklus skončí, keď hráč uhádne, t.j. bude splnená podmienka tip = uhadnut (toto neznamená tip priraď uhadnut!; problém znaku „=“ riešia rôzne programovacie jazyky rôzne, v Pythone sa „rovná sa“ zapisuje „==“). For-cyklus nemôžeme použiť, musíme použiť cyklus s podmienkou, t.j. while-cyklus.

```
uhadnut ← random(100)+1
čítaj(tip)
pokiaľ je tip <> uhadnut opakuj
začiatok
        ak tip<uhadnut
        tak píš("Málo, pridaj!")
        inak píš("Veľa, uber!")
        čítaj(tip)
koniec
píš("Uhádol si!")
```

```
uhadnut = random(100)+1
read(tip)
while tip <> uhadnut do
begin
        if tip<uhadnut
        then write("Málo, pridaj!")
        else write("Veľa, uber!")
        read(tip)
end
write("Uhádol si!")
```

Poznať by ste mali funkciu random(celé_číslo), ktorá vráti náhodné celé číslo od 0 po celé_číslo-1. Random() vráti reálne číslo z polouzavretého intervalu <0,1). Často sa pri práci s celými číslami využívajú funkcie div a mod. Funkcia div vracia celočíselný podiel a funkcia mod zvyšok po celočíselnom delení, t.j. nech a div b = c a a mod b = d, b>d>=0 potom a = b.c + d. Napríklad 10 div 7 = 1 a 10 mod 7 = 3 a 10 = 7.1+3; alebo 13 div 3 = 4 a 13 mod 3 = 1; alebo 5 div 9 = 0 a 5 mod 9 = 5 a 5 = 9.0+5; alebo 15 div 3 = 5 a 15 mod 3 = 0 a 15 = 3.5+0.