štruktúrované príkazy – CYKLY

Čo je to vlastne cyklus? Jednoducho povedané je to opakovanie určitej postupnosti príkazov.
Ak vieme dopredu povedať koľkokrát máme zopakovať určitú postupnosť príkazov – hovoríme o cykle pevným počtom opakovaní alebo tiež o cykle typu **`for`**.
Pri riešení úloh nemusíme vždy dopredu vedieť, koľkokrát sa majú príkazy zopakovať. Opakovanie môže byť určené platnosťou nejakej podmienky – teda logického výrazu.
Platnosť podmienky môžeme zisťovať:

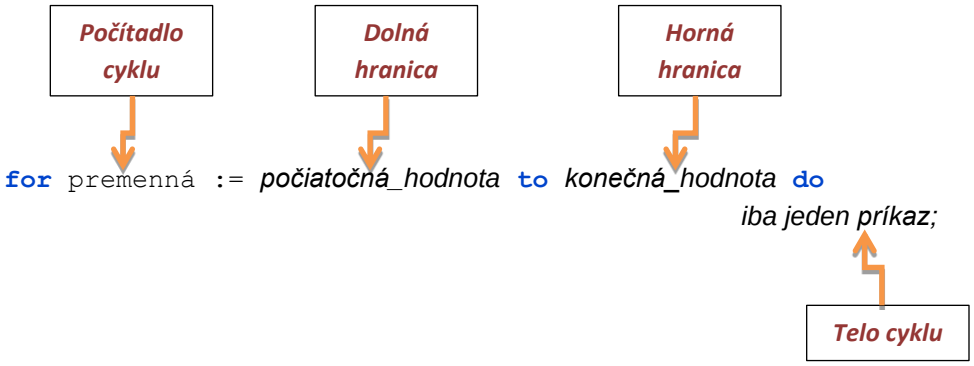
- na začiatku cyklu – hovoríme o ***cykle s podmienkou na začiatku*** alebo tiež o cykle typu **`while`**
- na konci cyklu – hovoríme o ***cykle s podmienkou na konci*** alebo tiež o cykle typu **`repeat`**

Cyklus FOR

Pozrite sa na časť programu, v ktorom sme nejaký príkaz zapísali úplne rovnako viackrát: vypisujeme 5 – krát slovo Ahoj.

```
begin
  Writeln('Ahoj');
  Writeln('Ahoj');
  Writeln('Ahoj');
  Writeln('Ahoj');
  Writeln('Ahoj');
end;
```

Vo väčšine programovacích jazykoch existujú konštrukcie - hovoríme im ***cyklus***, pomocou ktorých sa jednoducho zapíše opakovanie nejakých príkazov. Začneme cyklom ***s pevným počtom opakovaní***. Jeho zápis je:



alebo

```
for premenná := počiatočná_hodnota to konečná_hodnota do begin
                                                        príkaz1;
                                                        príkaz2;
                                                        .
                                                        .
                                                    end ;
```

Zložený príkaz

Predchádzajúci príklad zapíšeme pomocou tejto konštrukcie:

```
var
  I: Integer;
begin
  for I := 1 to 5 do
   Writeln('Ahoj');
end;
```

Inak povedané: ***Opakuj 5-krát.***

Cyklus FOR:

- Používa počítadlo cyklu (riadiacu premennú), v ktorej si program postupne počíta prechody cyklu
- Na začiatku sa do riadiacej premennej priradí počiatočná hodnota
- Hodnota tejto premennej sa bude zvyšovať o jedna zakaždým, keď sa vykoná príkaz tela cyklu
- Zvyšuje sa dovtedy, pokiaľ nenadobudne koncovú hodnotu
- Dá sa povedať, že príkaz sa vykoná raz pre každú hodnotu počítadla od počiatočnej po konečnú hodnotu

Pre cyklus **`for`** platí, ak koncová hodnota je menšia než štartová, telo cyklu sa nevykoná ani raz. Zrejme, ak sú tieto dve hodnoty rovnaké, telo cyklu sa vykoná len raz.

Premenná cyklu musí byť ***zadeklarovaná***, v našich príkladoch ju deklarujeme ako celé číslo. Táto premenná ešte pred príkazom cyklu je "obyčajnou" premennou a môžeme ju používať na ľubovoľné účely. Počas cyklu sa stáva počítadlom a nesmie sa nijakým spôsobom do nej priraďovať. Po skončení cyklu, až po záverečný end programu, je to opäť obyčajná premenná.

- Ak chceme v cykle **`for`** vykonať viac ako jeden príkaz, použijeme zápis s **`begin`** a **`end`**. Ukážeme si to v nasledujúcom príklade. Program načítava postupne desať čísel, spočíta ich a výsledok vypíše. Doplňte doňho všetko potrebné, aby pracoval správne.

```
var
  I, A, S: Integer;
Begin
  ?
  for I := ? to ? do
  begin
    writeln('Zadaj cislo');
    readln(A);
    ?
  end;
Writeln('Sucet cisel je ', ?);
end;
```

- Prekontrolujte, koľkokrát sa vykoná príkaz v cykle:
  a) ak dolná hranica > horná hranica
  b) ak dolná hranica = horná hranica
  c) ak dolná hranica je -10 a horná hranica je 1, `(for I := -10 to 1 do)`
  d) ak dolná hranica je -1 a horná hranica je -10, `(for I := -1 to -10 do)`
  e) ak dolná hranica je 3 a horná hranica je 8, `(for I := 3 to 8 do)`.

Počítadlo cyklu je premenná, ktorú môžeme v cykle veľmi užitočne používať. Môže byť v nejakom výraze, môže byť parametrom nejakého príkazu, môžeme ju priradiť inej premennej, ale nesmieme zmeniť jej hodnotu.

- Ukážeme si to v nasledujúcom programe, ktorý vypíše pod seba čísla aritmetickej postupnosti $3n+1\ _{n=1}^{20}$ a potom vypíše aj ich súčet:

```
Var S,I,A:integer;
begin
S:=0;
for I := 1 to 20 do
 Begin
  A:= 3*I+1;
  Writeln(A);
  S:=S+A;
 End;
Writeln('Sucet clenov postupnosti je ', S)
End;
```

Prevrátený cyklus

Niekedy potrebujeme (alebo je pre nás výhodnejšie), aby sa premenná v cykle znižovala, namiesto zvyšovala – použijeme prevrátený cyklus

`for` premenná:= *horná_hranica* **`downto`** *dolná_hranica* **`do`** príkaz;

- Prevrátený cyklus ukážeme na programe, ktorý vypíše pod seba druhé a tretie mocniny čísel od 20 do 10

```
var
 I: Integer;
begin
 for I := 20 downto 10 do
  begin
   Write(I, ' ');
   Write(I*I, ' ');
   Writeln(I*I*I, ' ');
  end;
```

Úlohy:

1. Zostavte program, ktorému používateľ zadá, koľko rožkov je schopný zjesť na posedenie. Program bude postupne vypisovať: „Jem 1. rozok, jem 2. rozok, ..., jem x. rozok.“ a nakoniec „Uz mam dost“.
2. Zostavte program, ktorý vypíše všetky párne čísla počínajúc 2 do 100.
3. Zostavte program, ktorý pre zadané číslo vypíše všetky jeho násobky (od 1 do 10 násobok) v tvare 1xčíslo=číslo.
4. Dané sú dve prirodzené čísla A, B. Zostavte program, ktorý vypočíta súčet čísel z intervalu <A,B>, ktoré sú deliteľné číslom 3.
5. Zostavte program, ktorý vypíše faktoriál pre zadané N.
6. Zostavte program, ktorý vypočíta 2^N pre zadané N.
7. Zostavte program, ktorý vypíše K-tu mocninu čísla N. Čísla K a N zadávame.
8. Syn dostával od otca vlani pevné vreckové N eur každý týždeň. Od Nového roka mu otec zvyšuje vreckové každý týždeň o 2 eurá. Zostavte program, ktorý vypočíta, koľko otec vyplatí synovi na vreckovom do 31. decembra.
9. Zostavte program, ktorý zo zadaného šesťmiestneho čísla N vypíše jednotlivé cifry pod seba.
10. Človek oslobodil džina z fľaše. Džin mu ponúkol dve možnosti odmeny:
    a) na každé zo 16 políčok šachovnice položí po 2000 zlatiek a daruje ich človeku;
    b) na prvé zo šestnástich políčok šachovnice položí 1 zlatku a na každé ďalšie 2 – krát viac zlatiek a tiež mu ich daruje.

    Zostavte program, ktorý zistí, ktorá možnosť je pre človeka výhodnejšia.
11. Zahrajte sa hru: hráč zvolí číslo od 2 po 9. Postupne každý z hráčov hovorí čísla počnúc 1 až po 100. Ak číslo, ktoré má hráč povedať, obsahuje zvolené číslo, alebo je ním deliteľné, povie len: „bum“. Vyhráva ten, kto sa bez chyby dostane vo vymenovávaní čísel najďalej. Zostavte program, ktorý načíta číslo a pre vedúceho hry vytvorí tabuľku, v ktorej budú vypísané všetky čísla 1..100, ale namiesto čísel, pre ktoré sa má povedať „bum“ bude hviezdička.
12. Zostavte program na výpočet súčtu radu
$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots\ldots\ldots+\frac{1}{9999}-\frac{1}{10000}$$
13. Číslo N nazveme dokonalým, keď sa jeho hodnota rovná súčtu všetkých jeho deliteľov, okrem neho samého. Dve najmenšie dokonalé čísla sú 6, 28, lebo
    6=1+2+3
    28=1+2+4+7+14
    Zostavte program, ktorý vypíše všetky dokonalé čísla z intervalu <1,9000>..
14. Zostavte program, ktorý vypočíta 2. mocninu zadaného prirodzeného čísla N, ako súčet prvých N nepárnych čísel. Napr.
    2^2=1+3
    5^2=1+3+5+7+9
15. Fibonačiho čísla.
    Za jeden deň mladý zajac zostarne a starý porodí jedného mladého. Zostavte program, ktorý vypíše, koľko bude zajacov na N – tý deň, keď prvý deň bol jeden zajac.
16. Trezor má kód zložený z troch čísel idúcich po sebe z cifier 0..9. Napríklad 4, 5, 6. Zostavte program, ktorý vypíše všetky možnosti kódu.
17. Zostavte programy , ktoré vykreslia podľa zadaného čísla N útvary z hviezdičiek. Napríklad pre N=4 vyzerajú útvary takto:

a)
```
****
****
****
****
```
b)
```
*
**
***
****
```
c)
```
****
***
**
*
```
d)
```
*******
 *****
  ***
   *
```
e)
```
****
*  *
*  *
****
```
f)
```
****
 ***
  **
   *
```
g)
```
*
**
***
****
***
**
*
```
h)
```
*
 *
  *
   *
  *
 *
*
```

Generátor náhodných čísel

- `Random` je špeciálna funkcia, ktorá počíta náhodné hodnoty
- Má jeden parameter – celé číslo, podľa ktorého vyberá z množiny prvkov
- Každé zavolanie funkcie `Random(N)` vyberie náhodne číslo od 0 po N-1, t.j. jednu z N rôznych hodnôt
- Volanie Random(6) vyberie jednu z hodnôt zo 6 prvkovej množiny {0, 1, 2, 3, 4, 5}
- Keď túto funkciu zavoláme viackrát, tak počítač môže dávať rôzne výsledky

- Použitie funkcie ukážeme na programe, ktorý vypíše desať náhodných čísel od 0 po 20.

```
var
  I: Integer;
begin
  for I := 1 to 10 do
    Writeln(random(21));
end;
```

- Pomocou funkcie `Random` môžeme generovať nielen celé čísla z intervalu <0, N-1>, ale ak k výsledku niečo pripočítame, alebo ho vynásobíme, môžeme dostať aj iné náhodné množiny. Napr.

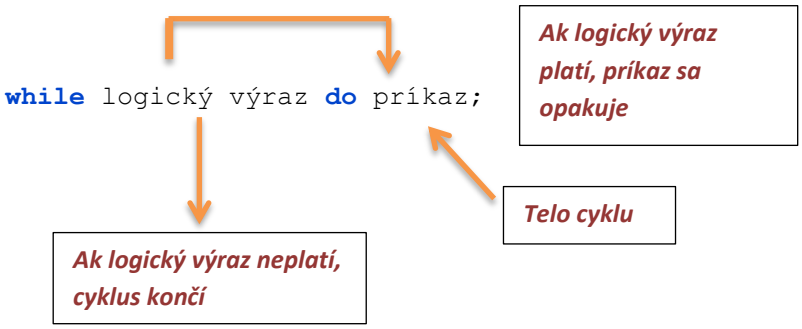
```
Random(2) {0, 1}
```

```
Random(3)-1 {-1, 0, 1}
Random(6)+1 {1, 2, 3, 4, 5, 6}
2*Random(2)-1 {-1, 1}
2*Random(4)+2 {?, ?, ?, ?}
Random(21)-10 <-10, 10>
Random(?)-? <-90, 90>
Random(?)+A interval <A, B>
Random(2)*(B-A)+A dvojprvková množina {?, ?}
```

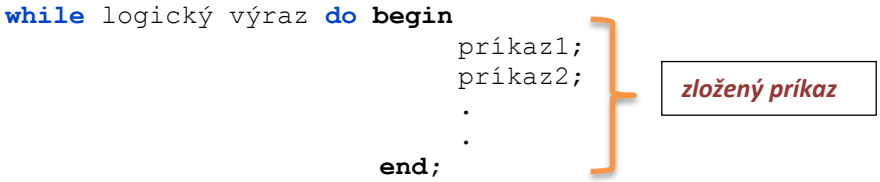
18. Zostavte program, ktorý namiesto načítania, bude generovať 100 náhodných čísel z intervalu <1,1000>. Zistite, ktoré číslo bolo vygenerované ako najväčšie.
19. Pre losovanie lotérie potrebujeme vygenerovať náhodné 6-ciferné číslo, ktoré nebude obsahovať 0 (generujte po cifrách).
20. Zostavte program, ktorý preskúša žiaka zo sčítania, odčítania a násobenia. Žiak bude dostávať 10 náhodných úloh zo zadaného intervalu a náhodnú operáciu. Program ho ohodnotí známkou:
    10,9 správnych - 1,
    8,7 správnych – 2,
    6,5 správnych – 3,
    4,3 správnych – 4,
    menej správnych – 5.
21. Zostavte program s ktorým si zahráte desať krát s počítačom hru kameň, papier, nožnice. Po hre program zobrazí štatistiku výhier počítač a hráč.
22. Zostavte program, pomocou ktorého budete hádať číslo zo zadaného intervalu. Číslo náhodne vyberie počítač a hádať môžete 4 – krát. Po každom type vám poradí: je to viac alebo je to menej.
23. Čo ste zistili pri viacnásobnom spúšťaní predošlých programov? Skúste doplniť na začiatok programu príkaz `Randomize.` Zmenilo sa niečo?

Cyklus `WHILE`

Je to cyklus s neznámym počtom opakovaní. Cyklus je ukončený podmienkou – logickým výrazom, ktorý je na začiatku cyklu. Uveďme si syntax príkazu **`WHILE:`**



V tele cyklu je – príkaz - jediný príkaz. Ak chceme v tele cyklu viac príkazov, musíme použiť zložený príkaz nasledovne:



Inak povedané: ***Pokiaľ platí podmienka rob príkazy***

Cyklus WHILE sa realizuje takto:

- Najskôr sa vyhodnotí podmienka - logický výraz
- Ak je podmienka splnená, tak sa vykoná príkaz a znovu sa vyhodnocuje podmienka
- Cyklus sa ukončí, ak je prvýkrát nesplnená podmienka – logický výraz
- **`While`** cyklus sa nemusí vykonať ani raz, ak je podmienka nepravdivá ,lebo test je hneď pri vstupe do cyklu

POZOR!
Pozor na ***nekonečný cyklus***! Aby cyklus skončil, musí nastať situácia , kedy podmienka – logický výraz bude mať hodnotu false a vtedy cyklus skončí.

Ukážme si malý príklad použitia:

- Príklad cyklu zo života:

```
Choď_do_postele;
while si_unavený do
                    zdriemni_si;
vylez_z_postele;
```

- Príklad cyklu zo života s vetvením v cykle:

```
choď_do_školy;
while hodina<=7 do
  if pozera_na_teba_ucitel then predstieraj_ze_pracujes
                           else lahni_si_na_lavicu;
chod_domov;
```

- Príklad cyklu zo života s počítadlom:

```
 chod_do_krcmy;
pocet_piv:=10;
while pocet_piv>0 do
          begin
            vypi_pivo;
            pocet_piv:=pocet_piv-1;
          end;
 chod_domov;
```

- Zabezpečme načítavanie iba kladných celých čísel od 1 do 100 z klávesnice.

```
writeln('Zadaj cislo');
readln(cislo);
while (    ?    ) ? (    ?    ) do begin
                                        writeln('Zadaj cislo');
                                        readln(cislo);
                                end;
```

- Zostavte program, ktorý nájde najväčší spoločný deliteľ dvoch prirodzených čísel pomocou Euklidovho algoritmu **.** Ukážka:

x	42	18	18	6
y	24	24	6	6

Pokiaľ sa čísla x a y nerovnajú, tak od väčšieho čísla odrátame menšie a menšie ponecháme. Doplňte nasledovný program.

```
Program NSD;
var x,y:integer;
begin
write('Zadaj dve cisla: ');
readln(x,y);
while   ?   do   if  ? then   ?

                          else   ? ;
writeln('NSD tychto cisel je ',?);
readln;
End.
```

Úlohy:

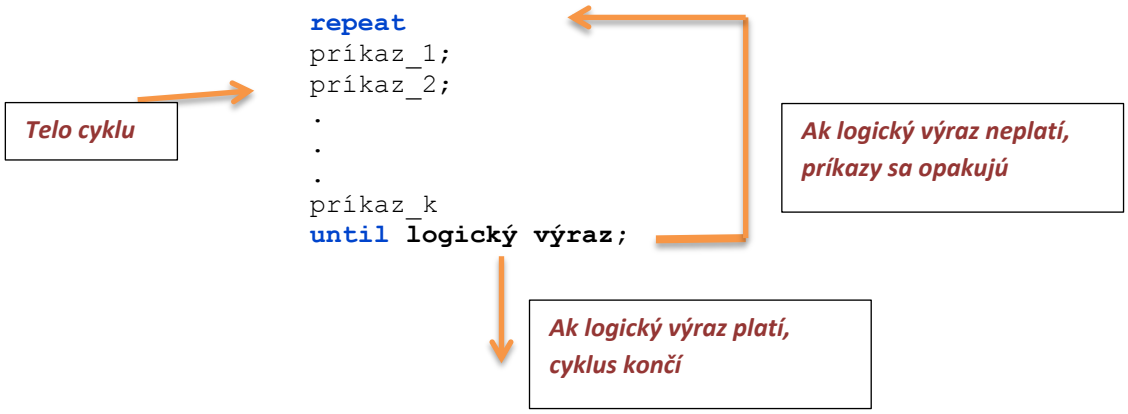
1. Najväčší spoločný deliteľ vieme nájsť i veľmi jednoduchým spôsobom. Zadáme dve celé nezáporné čísla X a Y. Zistíme menšie z nich. Pokiaľ menšie nebude deliteľom oboch čísel X a Y, budeme to menšie zmenšovať o 1, až kým nenájdeme spoločného deliteľa. Naprogramujte uvedený algoritmus.
2. Viete zefektívniť algoritmus z úlohy 1?
3. Nosnosť výťahu je 600 kg. Do výťahu nakladáme postupne balíky s rôznou hmotnosťou H. Ak celková hmotnosť nákladu prekročí nosnosť výťahu, musíme posledný balík vybrať z výťahu. Zostavte program, ktorým napodobníte činnosť kontrolného programu.
4. Predavač zmrzliny má ráno nachystaných 100 kopčekov zmrzliny. Zákazníci si od neho kupujú (náhodne) jeden, dva alebo tri kopčeky. Zostavte program, ktorým napodobníte prácu zmrzlinára. Program vypíše počet kupujúcich a či posledný zákazník dostal toľko zmrzliny koľko chcel alebo nie.
5. Predpokladajme, že v programovacom jazyku nie je operácia div (celočíselné delenie). Zostavte program, ktorý pomocou odčitovania bude dávať výsledok operácie A div B.
6. Zostavte program, ktorý zistí, či zadané číslo N je prvočíslo.
7. Zostavte program, ktorý bude načítavať postupnosť kladných čísel K, ukončených 0. Vypíšte aritmetický priemer zadaných čísel, ktoré číslo bolo najväčšie, koľko bolo párnych a nepárnych čísel.
8. Lenivý Jim má z domu 100 m. Každé ráno, keď sa na pole vyberie, uprostred cesty sa spamätá a rozhodne: „Na pole pôjdem, ak dostanem vyššie znamenie.“ Vyšším znamením je takáto hra náhody:
   Jim hodí mincou. Ak padne líce, pokračuje smerom na pole, ak padne rub, obráti sa naspäť domov. S riešením nie je spokojný a hádzanie mincou opakuje každých 10 m. Keď padne líce pokračuje smerom ako išiel doteraz, keď padne rub, otočí sa. Zostavte program, ktorý pomocou generátora náhodných čísel bude simulovať hádzanie mincou a určite, či sa Jim dostane na pole alebo príde domov.
9. Hodnota sin x, kde x je uhol v oblúkovej miere, sa počíta podľa vzorca

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

   pre $x \leq \pi$. Vzorec sa používa tak, že na pravej strane sa sčíta toľko členov, koľko treba na dosiahnutie presnosti p>0.

Cyklus REPEAT

Je to cyklus s neznámym počtom opakovaní. Cyklus je ukončený podmienkou, ktorá je na konci cyklu.
Uveďme si syntax príkazu **REPEAT**:



Inak povedané: ***Opakuj príkazy pokiaľ nie je splnená podmienka***

Cyklus REPEAT sa realizuje takto:

- Najskôr sa vykoná postupnosť príkazov medzi **`repeat a until`**
- Potom sa vyhodnotí logický výraz za **`until.`**
- Ak jeho hodnota je pravdivá, cyklus sa ukončí. Ak je hodnota nepravdivá, tak sa proces opakuje. Cyklus sa ukončí, ak je prvýkrát splnená podmienka – logický výraz.
- Na rozdiel od **`WHILE`** cyklus **`REPEAT`** prebehne aspoň raz, lebo test je až po prvom vykonaní postupnosti príkazov.

POZOR!
V tele cyklu sa musí niečo meniť tak, aby od určitého okamihu podmienka prestala platiť, lebo by nastal prípad nekonečného cyklu, a program by nikdy neskončil.
Príkazy **`repeat a until`** nahrádzajú slová **`begin a end`**.

Ukážme si malý príklad použitia:

- Príklad cyklu zo života:

```
repeat
daj_si _koláč
until máš_dosť;
```

- Zabezpečme načítavanie iba kladných celých čísel od 1 do 100 z klávesnice. Všimnime si rozdiel, túto úlohu sme riešili už pomocou cyklu `While`

```
repeat
writeln('Zadaj cislo');
readln(cislo);
until (   ?   ) ? (   ?   );
```

- Zistite, čo bude výsledkom nasledovného programu

```
Program TEST;
  Var N: integer;
  Begin
     Writeln (´Zadaj cislo N>1´);
     Read (N);
     K:=1;
      Repeat
             K:=K+1;
      until (N mod K=0);
     Write (N);
     If N = K then Writeln (´nie je  ????????´);
               else Writeln (´je ????????´);
    Readln;
 End.
```

- Zistite, čo bude výsledkom nasledovného programu

```
repeat
       a:=random(6);
until a=6;
writeln(a);
```

Úlohy:

1. Vyberte správne:

   Príkaz `for i:= 0 to 9 do write(i);` vypíše tie isté hodnoty ako príkaz

   a) `for i:= 9 downto 0 do write(10 – i);`
   b) `i:= 0; while i = 9 do begin write(i); i:= i + 1; end;`
   c) `for i:= 0 to 9 do begin write(i); i:= i + 1; end;`
   d) `i:= -1; repeat i:= i+1; write(i) until i = 9;`

2. Zostavte program, ktorý napodobňuje začiatok hry „Človeče, nehnevaj sa“, v ktorom hráč hádže kockou, kým mu nepadne šestka, až potom môže postaviť svojho panáčika na hracie pole. Nech program vypíše, koľko kôl hráč čakal, kým mu padla šestka.
3. Zostavte program na výpočet ciferného súčtu (súčet cifier) zadaného celého kladného čísla X.
4. Zostavte program na zistenie, koľkokrát sa istá zadaná cifra vyskytuje v zápise zadaného celého čísla X.
5. Trieda má záväzok odovzdať 1000 kg papiera. Zostavte program, ktorý sčítavať hodnoty odovzdaného papiera P dovtedy, kým prvý krát nebude prekročená hodnota 1000.
6. Zostavte program, ktorý prevedie zadané prirodzené číslo X do dvojkovej sústavy.
7. Zostavte program, ktorý prevedie zadané prirodzené číslo X do ľubovoľnej zadanej sústavy.
8. Zostavte program, ktorý prevedie zadané číslo X z dvojkovej sústavy do desiatkovej.
9. Zostavte program na nájdenie Armstrongových čísel (od 1 do 10000). Otestujte teda každé číslo z uvedeného intervalu, či spĺňa podmienku na Amstrongovo číslo (súčet tretích mocnín cifier čísla sa musí rovnať danému číslu). Vypíšte len Armstrongove čísla.
10. Zostavte program, ktorý zistí počet mrazivých (pod nulou) a nemrazivých dní (nad nulou). Zabezpečte zadávanie postupnosti kladných a záporných teplôt T, ktoré sú ukončené nulou. Vypíšte, ktorých dní bolo viac – mrazivých či nemrazivých.
11. Zostavte program na zistenie, či načítané číslo X, môže byť číslom zapísaným v sústave Z, ktorú tiež zadávame zo vstupu. Zabezpečte, aby pre z platiloť 0<Z<=10.
12. Na Valentína som dostala dve čokolády, každá z nich váži 100 gramov. Budem ich jesť tak, že každý deň zjem polovicu toho čo mám. Zostavte program, ktorý zistí, ako dlho budem jesť čokoládu kým mi ostane kúsok vážiaci menej ako 1 gram.
13. Zostavte program, ktorý bude načítavať prvky postupnosti a vypíše o nej jednu z týchto správ: UTRIEDENÉ, NEUTRIEDENÉ, ROVNAKÉ.
14. Predpokladajme, že počítače vedia násobiť a deliť len dvomi. (V podstate to nie je ďaleko od pravdy). Súčin dvoch zadaných čísel M a N, sa môže získať “ruskou” metódou násobenia. Pozostáva z následného delenia menšieho čísla dvomi (zvyšok sa ignoruje) a následného násobenia väčšej hodnoty dvomi, kým z menšieho čísla nezostane 1. Počas procesu sa pripočítava k súčtu iba výsledok tých násobkov väčšieho čísla, pre ktoré delenie menšieho predstavuje nepárne číslo.
    Napr. 37 x 65
    37 18 9 4 2 1
    65 130 260 520 1040 2080 = 2405

Upozornenie na záver cyklov:

Ak ste nepochopili algoritmus s cyklom, ***opakujte preštudovanie materiálu.***

Cyklus umožňuje elegantne zovšeobecniť riešenie problému, umožňuje realizovať hromadné spracovanie údajov, elegancia znamená menej príkazov, čo najmenšiu spotrebu pamäti počítače a čo najkratšiu dobu výpočtu.

.