BALTÍK II – modul 8

V tomto module sa si preberieme jednu zo základných programátorských konštrukcií – cyklus s pevným počtom opakovaním – tzv. cyklus for.

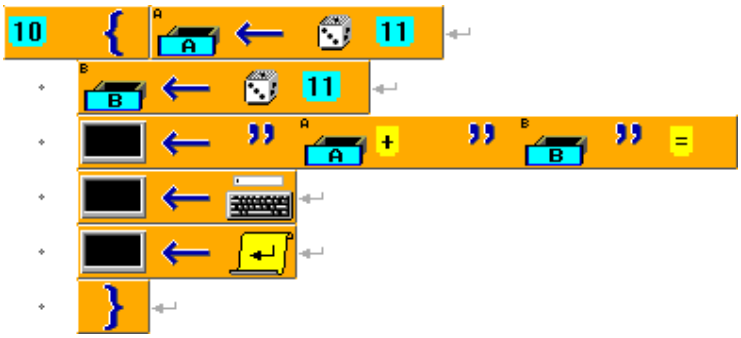
CYKLUS S PEVNÝM POČTOM OPAKOVANÍ

V programovaní používame dva druhy cyklov:

- cyklus s podmienkou na začiatku (prípadne na konci),
- cyklus s pevným počtom opakovaní.

U cyklov s podmienkou (cyklus **while**, resp. **do-while** – tým sme sa nezaoberali) nevieme dopredu, koľkokrát prebehne, závisí to od toho, ako dlho bude platiť podmienka.
U cyklu s pevným počtom opakovaní je na jeho začiatku jednoznačne dané, koľkokrát prebehne.
Cyklus s pevným počtom opakovaní nám nie je v Baltíkovi až tak neznámy. V určitej forme sme ho už používali.

V tomto prípade Baltík 14-krát vyčaruje kvietok a posunie sa dopredu.

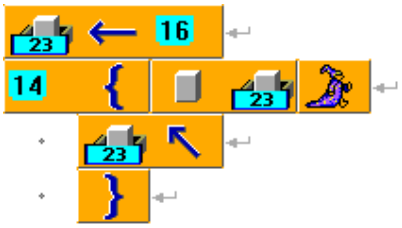


A v tomto programe sa 10-krát vypíše na obrazovku príklad na sčítanie do desať, ku ktorému bude možné dopísať výsledok.

V oboch prípadoch jasne vidíme, že cyklus sa prevedie pevný počet krát, v prvom prípade 14-krát, v druhom 10-krát, t.j. môžeme hovoriť o cykle s pevným počtom opakovaní. Oba prípady majú ale spoločnú ešte jednu vec – **v cykle sa počas jeho opakovania nič nemení v závislosti na tom, koľký raz cyklus prebieha** .

Čo v prípade, ak by sme ale chceli dosiahnuť, aby Baltík nečaroval 14-krát kvietok, ale postupne predmety číslo 16 – 29?

S našimi doterajšími vedomosťami by naše riešenie vyzeralo asi takto:



V iných programovacích jazykoch ale takáto forma zápisu cyklu s pevným počtom opakovaní neexistuje a ani v Baltíkovi nebudeme toto riešenie uprednostňovať. Aby sme si to dopredu roztriedili a ujasnili:

- tam, kde sa cyklus s pevným počtom opakovaní nemení v závislosti od toho, koľký raz prebieha, budeme používať v Baltíkovi doterajšiu formu zápisu,
- tam, kde sa cyklus s pevným počtom opakovaní mení v závislosti od toho, koľký raz prebieha, budeme používať novú formu zápisu, ktorú si teraz ukážeme.

CYKLUS FOR

Cyklus **for** („for“ po slovensky znamená „pre“) určuje svoj počet opakovaní pomocou premennej – budeme jej hovoriť **riadiaca premenná**. V hlavičke cyklu vždy uvedieme názov premennej, jej počiatočnú hodnotu, koncovú a o koľko má pri každom ďalšom zopakovaní cyklu zmeniť svoju hodnotu. Následne sa môžeme na túto premennú v cykle odvolávať, hneď si ukážeme, ako.

Pre vytvorenie hlavičky cyklu **for** budeme potrebovať nasledujúce prvky:

 - prvok pre kľúčové slovo **for**

 - logické zátvorky (v zásade by sa dalo bez nich obísť, ale vďaka ním bude hlavička cyklu podstatne prehľadnejšia)

 - čiarka oddeľujúca jednotlivé parametre v hlavičke

Pozrime sa, ako bude vyzerať teraz celý cyklus. Ako príklad nám poslúži práve program, v ktorom má Baltík vyčarovať v dolnej časti plochy predmety 16 – 29:

Prvý riadok je hlavička cyklu, v druhom riadku vidíme, čo sa má v cykle daný počet ráz urobiť. Pozrime sa, čo znamenajú jednotlivé parametre v hlavičke cyklu.

Hlavička cyklu musí vždy ako prvé obsahovať kľúčové slovo **for**. V zátvorke za ním nasledujú vždy 4 parametre:

1. riadiaca premenná, v našom prípade je jej názov **Predmet**,
2. jej počiatočná hodnota (t.j. hodnota, ktorú bude mať, keď cyklus prebehne prvýkrát) – v našom prípade je to **16**,
3. jej koncová hodnota (t.j. hodnota, ktorú bude mať, keď cyklus prebehne naposledy) – v našom prípade je to **29**,
4. krok (t.j. hodnota, o ktorú sa má premenná zmeniť pri každom zopakovaní cyklu) – v našom prípade sa bude **zväčšovať o 1**.

Všimnime si, že v nasledujúcom bloku príkazov sa riadiaca premenná hneď využíva. Pri každom opakovaní cyklu sa vyčaruje predmet s takým číslom, aká je práve hodnota riadiacej premennej, takže cyklus bude prebiehať asi takto:

1-krát (**Premenná = 16**): vyčaruje sa predmet č. 16; Baltík sa posunie

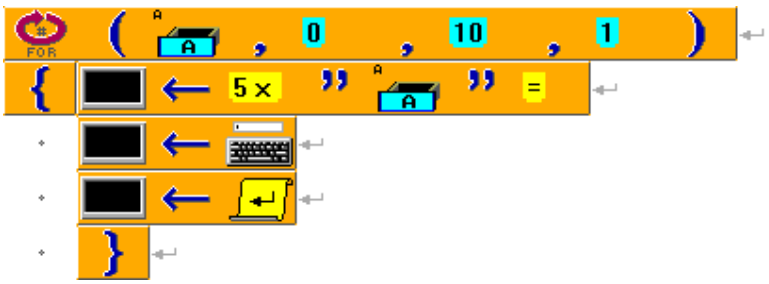
2-krát (**Premenná = 17**): vyčaruje sa predmet č. 17; Baltík sa posunie

3-krát (**Premenná = 18**): vyčaruje sa predmet č. 18; Baltík sa posunie

....

14-krát (**Premenná = 29**): vyčaruje sa predmet č. 29; Baltík sa posunie

Poďme sa pozrieť na druhý program, ktorý sme si na začiatku ukazovali. Skúsme ho zmeniť tak, aby sa v danom cykle vypísali príklady na násobenie čísla 5 s číslami o 0 po 10.



Riadiaca premenná je v tomto prípade **A** a cyklus bude prebiehať takto:

1-krát (**A=0**): vypíše sa príklad 5 x 0=; program si vyžiada výsledok

2-krát (**A=1**): vypíše sa príklad 5 x 1=; program si vyžiada výsledok

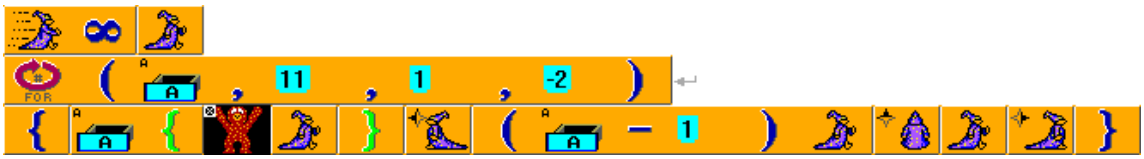
2-krát (**A=2**): vypíše sa príklad 5 x 2=; program si vyžiada výsledok

....

2-krát (**A=10**): vypíše sa príklad 5 x 10=; program si vyžiada výsledok

Riadiaca premenná nemusí zákonite v cykle stúpať, ale môže aj klesať. Pozrime sa napríklad, ako by mohol vyzerať program, v ktorom by Baltík postavil pyramídu z opíc.
Skúsme si najprv pripraviť riešenie bez cyklu **for**. Mohlo by vyzerať takto:

Vidíme, že medzi riadkami platí určitá zákonitosť v klesaní niektorých hodnôt. A to je ideálna príležitosť pre skrátenie programu pomocou cyklu **for**:

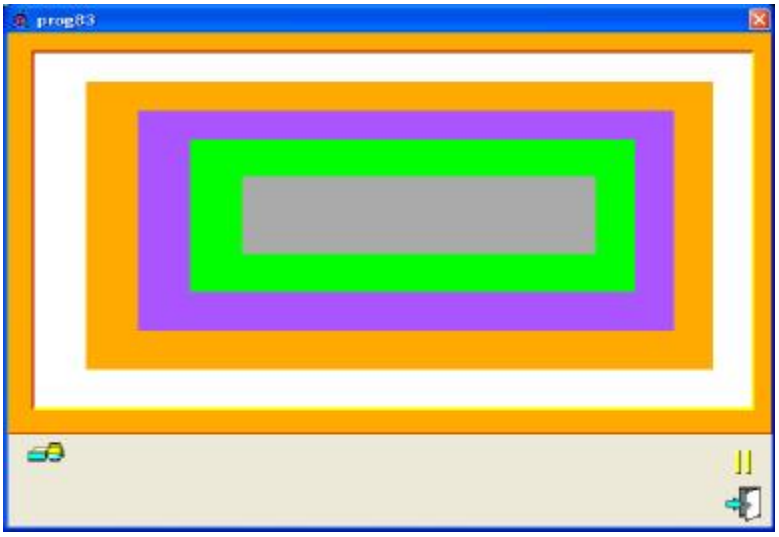


1-krát bude mať riadiaca premenná hodnotu 11, čo spôsobí, že sa v bloku príkazov udeje to isté, čo v prvom riadku opíc v predošlom riešení.
2-krát bude mať riadiaca premenná hodnotu 9, to zodpovedá druhému riadku opíc atď.

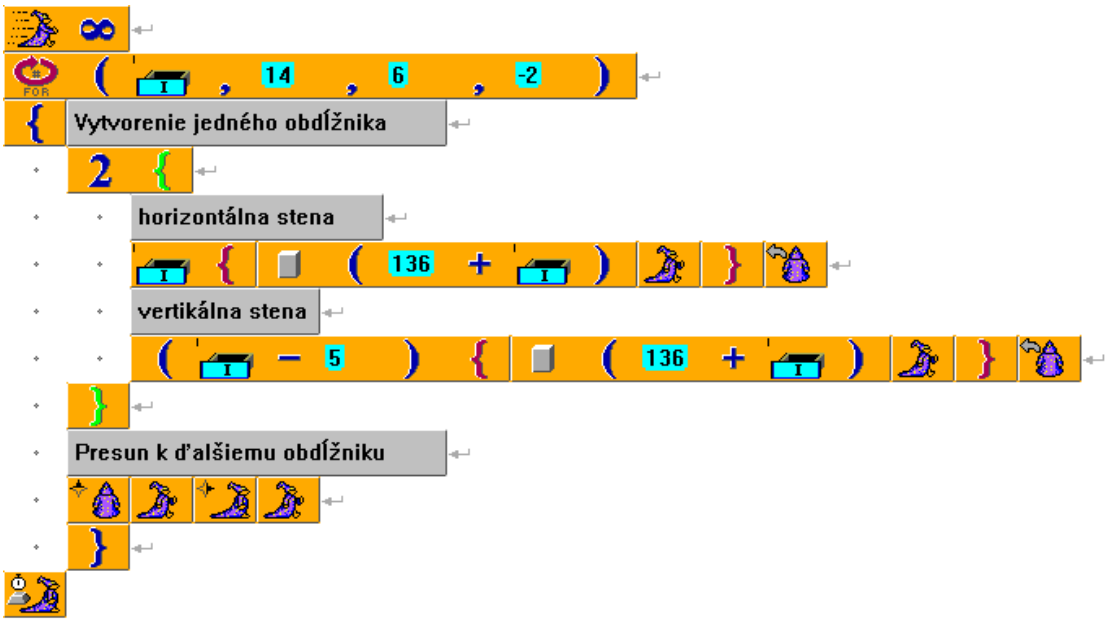
V tomto programe vidíme, že počiatočná hodnota riadiacej premennej je vyššia, ako koncová a jej hodnota sa musí zákonite meniť o záporné číslo (viď 4. parameter hlavičky cyklu).
Vyskúšajme si ešte niekoľko programov a pokúsme sa rozobrať v nich fungovanie cyklu **for**.

Príklad č. 1:

Program, v ktorom sa vytvoria do seba vložené obdĺžniky niekoľkých farieb (viď obr.).

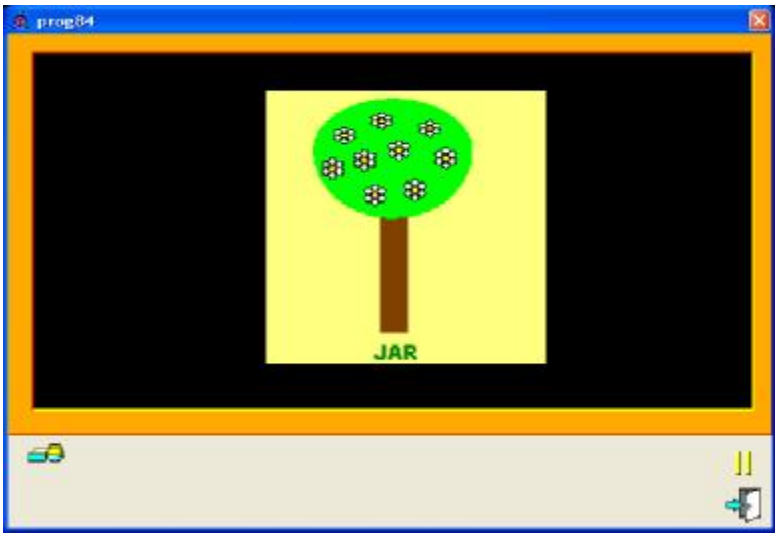


Možné riešenie:

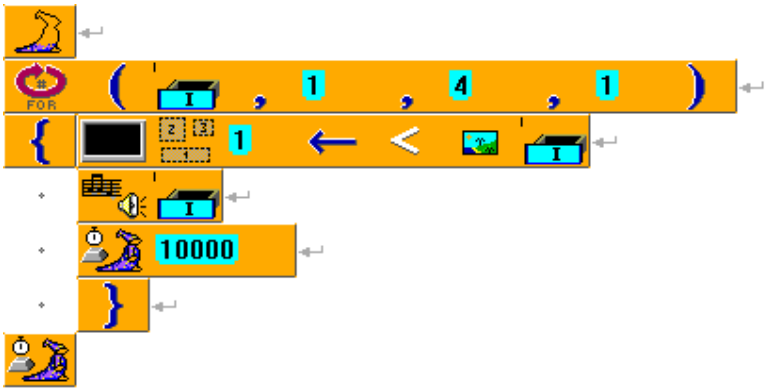


Príklad č. 2:

Program, v ktorom sa postupne na 10 sekúnd zobrazia obrázky symbolizujúce jar, leto, jeseň a zimu. Ku každému ročnému obdobiu bude zároveň hrať aj iná skladba.



Možné riešenie:



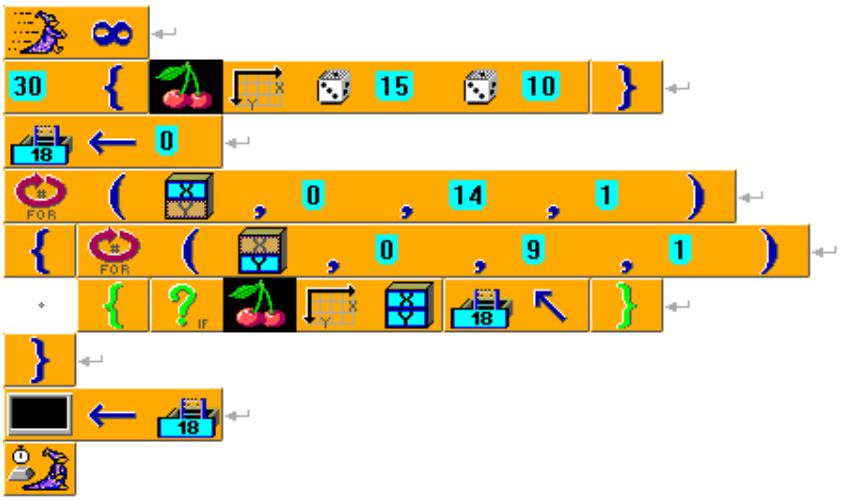
Ako obrázky môžeme použiť tie, ktoré sme vytvárali pre modul č. 6 (zaoberali sme sa v ňom oblasťami).

Podobne by sme mohli za sebou načítavať aj scény, oblasti apod. Ako vidíme, stačí využiť, že sú vždy jednoducho očíslované. Toto číslo len nahradíme riadiacou premennou, alebo prepočtom z nej a hneď máme v programe ušetrených niekoľko riadkov.

No a na záver ešte niečo náročnejšie. Cyklus **for** môže v sebe pokojne obsahovať ďalší cyklus **for** a niekedy je takéto riešenie veľmi užitočné.

Klasickým príkladom je program, v ktorom sa vyčaruje napr. 30-krát na Baltíkovu plochu čerešnička. Keďže v skutočnosti sa pravdepodobne niektoré čerešničky prekryli, po skončení tejto časti programu nemôžeme presne vedieť, koľko sa ich na ploche nachádza. Ak by sme chceli napr. vytvoriť program, v ktorom bude Baltík pomocou šípok chodiť a zbierať čerešničky a tento program by mal skončiť vo chvíli, keď ich všetky pozbiera, musíme zákonite najprv zistiť, koľko ich tam skutočne máme.

Riešenie by mohlo vyzerať takto:



Riadiace premenné sú v tomto prípade **X** a **Y** a cyklus bude prebiehať takto:

1. **X=0:**
    1.1 **Y=0:** ak je na súradniciach **0, 0** čerešňa, **Počítadlo** zväčší o 1
    1.2 **Y=1:** ak je na súradniciach **0, 1** čerešňa, **Počítadlo** zväčší o 1
    1.3 **Y=2:** ak je na súradniciach **0, 2** čerešňa, **Počítadlo** zväčší o 1
    ....
    1.10 **Y=9:** ak je na súradniciach **0, 9** čerešňa, **Počítadlo** zväčší o 1
2. **X=1:**
    2.1 **Y=0:** ak je na súradniciach **1, 0** čerešňa, **Počítadlo** zväčší o 1
    2.2 **Y=1:** ak je na súradniciach **1, 1** čerešňa, **Počítadlo** zväčší o 1
    2.3 **Y=2:** ak je na súradniciach **1, 2** čerešňa, **Počítadlo** zväčší o 1
    ....
    2.10 **Y=9:** ak je na súradniciach **1, 9** čerešňa, **Počítadlo** zväčší o 1

....

15. **X=14**
    15.1 **Y=0:** ak je na súradniciach **14, 0** čerešňa, **Počítadlo** zväčší o 1

15.2 **Y=1:** ak je na súradniciach **14, 1** čerešňa, **Počítadlo** zväčší o 1

15.3 **Y=2:** ak je na súradniciach **14, 2** čerešňa, **Počítadlo** zväčší o 1

....

15.10 **Y=9:** ak je na súradniciach **14, 9** čerešňa, **Počítadlo** zväčší o 1

Vnorenie týchto dvoch cyklov spôsobuje, že sa postupne prezrú všetky stĺpce pracovnej plochy. Najprv stĺpec s *X*-ovou súradnicou 0, v ktorom sa prejdú všetky políčka postupne podľa ich *Y*-ových súradníc, potom stĺpec s *X*-ovou súradnicou 1, atď. až po posledný stĺpec.

Vnorenie dvoch cyklov do seba je na predstavu dosť náročné, ale ako vidíte na zdrojovom kóde, pomôže nám vytvoriť veľmi krátke a efektívne riešenia.

ZADANIA – VIĎ PRÍLOHA