evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Karel použití podmínek

Číslo projektu	CZ.1.07/1.5.00/34.0950
Kódování materiálu	vy_32_INOVACE_inf3_prg03
Označení materiálu	prg03_karel_podminky
Název školy	Gymnázium Kladno
Autor	Mgr. Pelikánová Lucie
Anotace	Výukový materiál Algoritmizace a programování, materiál je určen pro LMS systém, lze jej využít také pro samostatnou práci žáků. Tato část kurzu seznamuje s použitím podmiňovacího příkazu a různými druhy podmínek.
Předmět	Informatika a výpočetní technika
Tematická oblast	Algoritmizace a programování
Téma	Karel použití podmínek
Očekávané výstupy	Žák se naučí používat podmiňovací příkaz, úplný i neúplný. Seznámí se s druhy podmínek v prostředí jazyka Karel. Vytvoří několik nových jednoduchých příkazů, které využívají podmínky.
Klíčová slova	větvení programu, podmiňovací příkaz, úplný a neúplný podmiňovací příkaz, spojování podmínek, if – else – end, and – or – not
Druh učebního materiálu	výukový kurz (lekce)
Ročník	3
Cílová skupina	vyšší stupeň osmiletého gymnázia, čtyřleté gymnázium
Ověřeno	14. 5. 2013, O7
Pokud není uvedeno jinak, uvedený materiál je z vlastních zdrojů autora.	

Metodický pokyn

Vstupní soubory a složky:

prg03_karel_podminky.pdf	výukový text
prg03_karel_podminky.krl	praktické ukázky v prostředí Karel

Výstupy (řešení):

Tato ukázková řešení nejsou určena pro žáky, ale jsou pro učitele.

prg03_karel_podminky_reseni.pdf	řešení zadaných úloh
reseni/behej_dole.krl	řešený závěrečný příklad

Téma: Karel použití podmínek

Větvení programu

Větvení v programu používáme tehdy, pokud některé příkaz chceme provádět pouze za předpokladu platnosti nějaké podmínky. V takovém případě se nám program rozdělí do několika částí (rozvětví se). Větvení programu umožňuje konstrukce, kterou nazýváme podmíněný příkaz.

Podmínkou je výraz, který vrací tzv. logickou hodnotu. Odpovídá reprezentaci **PRAVDA / NEPRAVDA** (TRUE/FALSE). V Karlovi nalezneme podmínky speciální, ale i podmínky numerické.

Jednoduchý podmíněný příkaz

Zápis podmínky:

```
...
If podmínka
     příkazy //podmínka je splněna
else
     příkazy //podmínka není splněna
end
```

Jestliže je podmínka splněna, pak se provedou příkazy následující za příkazem `if`. Pokud podmínka není splněna, pak se vykonají příkazy, které následují po příkazu `else`. Pokud mezi `else` a `end` nejsou žádné příkazy, tak lze příkaz `else` vynechat. Jedná se o neúplný podmíněný příkaz.

Příklad 1 OPATRNÝ KROK

Karel pomocí speciální podmínky zjistí, zda se nachází těsně před zdí. Pokud ano, nebude dělat nic (zůstane stát), pokud před ním není zeď, udělá krok. Tento nový příkaz nazveme `step?`. Otazník v názvu nám naznačuje, že Karel bude dělat KROK pouze tehdy, pokud není před ním zeď. Často se tento speciální krok nazývá **OPRATRNÝ KROK.**
Použijeme podmínku **wall**. Tato podmínka vrací hodnotu true, pokud stojí Karel těsně před zdí.

```
procedure step?
  if wall
  else
    step
  end
end
```

Tento zápis je trošku nešikovný. Zbytečně používáme úplný podmiňovací příkaz, i když při splnění podmínky Karel nic nedělá. Je možné použít podmínku v tzv. negaci ve významu, **není zeď**.
K vytvoření negace použijeme speciální **logický (boolovský)** operátor **not.**

Tedy **not wall** představuje podmínku, ve významu není zeď.

Upravený program

```
procedure step?
  if not wall
    step
  end
end
```

Zde je použit neúplný podmíněný příkaz. Větev pro část, kdy podmínka není splněna, byla prázdná a bylo možné ji vynechat.

Praktické cvičení č. 1

1. Dojděte s Karlem opakovaným ručním pouštěním příkazu `step?` ke zdi.
2. Napište příkaz `ke_zdi`, který pomocí příkazu na opakování `repeat`, několikrát (stačí 30x pokud používáte základní místnost) použije příkaz `step?`. Kde se Karel zastaví?

Příkaz `ke_zdi` najdete na konci textu vyřešený.

Podmínky Karel

NORTH, SOUTH, EAST, WEST - tyto podmínky zjišťují, do jakého směru je právě robot Karel natočen.
WALL - tato podmínka je splněna, když je před Karlem zeď.
MARK - tato podmínka je splněna, když se pod Karlem nachází alespoň jedna značka
SPACE - tato podmínka je splněna, když lze položit značku.
TRUE - tato podmínka je vždy splněna
FALSE - tato podmínka není nikdy splněna

Spojování podmínek

Podmínky lze spojovat logickými spojkami AND, OR, negace pomocí NOT.
Lze také používat relační operátory =, <>, >, <, >=, <=.

Příklad 2 OPATRNÉ SBÍRÁNÍ

Naprogramujte příkaz `pick?` Karel sebere značku pouze tehdy, pokud ji má k dispozici.

```
procedure pick?
  if mark
    pick
  end
end
```

Příklad 3 Otočení Karla určeným směrem

Naprogramujte proceduru `na_jih`, která otočí Karla z jakéhokoli směru do směru jižního (dolů). Tento příkaz budeme řešit později efektivněji, tentokrát ho řešíme pomocí podmíněného příkazu.

Je nutné použít několik podmíněných příkazů v sobě.

```
procedure na_jih
  if north
    left
    left
  else
    if east
      right
    else
      if west
        left
      end
    end
  end
end
```

Praktické cvičení č. 2

Naprogramujte příkaz `put?` (OPATRNÉ POLOŽENÍ CIHLIČKY). Položí cihlu pouze tehdy, pokud je místo (podmínka `space`). Vyzkoušejte.

Větší příklad

Příklad: behej_dole

Naučte Karla pomocí repeat cyklu a podmínky „běhat“ po dolní řádce, u zdi se vždy otočí a jde zpátky.
Nápověda: Použijte repeat tak, že neuvedete počet cyklů, vytvoříte tak **nekonečný cyklus**.
Nekonečný cyklus ukončete stiskem červeného tlačítka stop.

Např.

```
procedure jdi
     repeat
          step? //opatrný krok
     end
end
```

Co tento příkaz udělá?

Pro použití v tomto úkolu, upravte `step?` takto: pokud je před Karlem zeď, otočí se, pokud není zeď udělá krok.

Řešení tohoto úkolu uložte jako behej_dole.krl.

Karel – podmínky řešení

(1)

```
procedure step?
  if not wall
    step
  end
end

procedure ke_zdi
  repeat 30
    step?
  end
end
```

(2)

```
procedure put?
  if space
    put
  end
end
```

Příklad behej_dole

```
procedure otoc
//Karel se otočí o 180 stupňů
  left
  left
end

procedure step?
//Opetrný krok, který u zdi Karla otočí
  if wall
    otoc
  else
    step
  end
end

procedure behej_dole
//Nekonečný cyklus Karel běhá po dolní řádce
  home
  repeat
    step?
  end
end
```