Příloha[1]

1. Jazyk Coach

1.1 Úvod

Součástí integrovaného prostředí Coach jsou programy Modelování a Řídicí prostředí, ve kterých je možno navrhovat, zapisovat, ladit a provádět modelové výpočty a řídicí programy. Instrukce programu musí být formulovány a pospojovány podle pravidel jazyka Coach, tj. způsobem, který je srozumitelný kompilátoru programů v prostředí Coach.

Před provedením modelového nebo řídicího programu se musí jeho text přeložit do strojového kódu (tzv. kompilace). Kompilátor také ověřuje, zda byly jednotlivé příkazy správně použity. Jestliže najde chybu, vypíše na obrazovku chybové hlášení.

V průběhu výpočtů se mohou vyskytnout chyby jiného druhu, např. hodnota argumentu funkce může překročit definiční interval této funkce nebo jmenovatel ve zlomku nabude nulové hodnoty. V takovém případě se programový výpočet zastaví a opět se vypíše chybové hlášení (viz §6, odst. Chybová hlášení).

1.2 Srovnání jazyka Coach s jinými programovacími jazyky

Jazyk Coach je relativně jednoduchý programovací jazyk odvozený od jazyka Pascal. Máte-li již nějaké zkušenosti s programováním v jazycích jako BASIC nebo Pascal, bude vám připadat většina výrazů a příkazů povědomá a srozumitelná. Abyste mohli začít psát programy v jazyce Coach, musíte se naučit pravidla pro zápis slova symbolů ve výrazech a příkazech (**syntaxe**) a samozřejmě také smysl těchto výrazů a příkazů (**sémantika**).

Na rozdíl od jiných univerzálnějších programovacích jazyků není v *jazyce Coach* možno definovat "pole" a "typy proměnných" nebo zadávat v průběhu výpočtu hodnoty z klávesnice a vkládat soubory z disku.

1.3 Použití jazyka Coach

Jazyk Coach umožňuje v programu Modelování zapsat příkazové řádky programu pro výpočet *modelu* spolu s tabulkou *počátečních hodnot;* takto je možno řešit i úlohy vedoucí k řešení diferenciálních rovnic. Programy se skládají z jednoho nebo více příkazů (programových řádek), které jsou sestaveny ze slov, čísel a znaků, resp. symbolů.

[1] (Zpracováno podle Přílohy č. 2 manuálu k systému IP Coach.)

2. Identifikátory, čísla a symboly

2.1 Identifikátory

Při zápisu vzorce nebo programu se pro označení konstant, proměnných, procedur a funkcí *používají jména (identifikátory),* která mohou být až na několik málo omezení libovolná. Je vhodné, když mají v některém živém jazyce smysl (např. proměnná *cas* nebo *time* je lepší než proměnná *XDR2W)*.

Při zápisu jmen **jsou povoleny** následující znaky z následujících sad:
– velká písmena {A, B, C, ...Z},
– malá písmena {a,b, c, ...z},
– číslice {1 2 3 … **O**},
– znaky {£ _ & ~ ! I { } [ ] };
– znaky ASCII kódy s výjimkou znaků π, ÷, Σ, √.

Při zápisu jmen se **nesmí** použít tyto znaky:
– znaky s ASCII kódy {0 ... 31},
– znaky označující aritmetické operace { ( ) - + * / + ^ = <>},
– vyhrazené znaky {' . , 7. " : ; $ # @} včetně mezery.

Poznámka č. 1

Není dovoleno používat:
– číslici jako první znak jména (např. 1strom),
– jména identicky shodná s klíčovými slovy, např. slovo 'sin'.

Poznámka č. 2

Omezení uvedená v poznámce č. 1 neplatí v tom případě, že jsou jména zapsána v hranatých závorkách. Je-li první znak jména "[" a poslední "]", je např. [2πr] povolené jméno proměnné.

Velká a malá písmena

Kompilátor nerozlišuje mezi malými a velkými písmeny. Například není možno současně použít znak R pro odpor žárovky a znak r pro výkon.

2.2 Čísla a číslice

Při zápisu čísel je povoleno použít znaky z následující sady:

{0 1 2 3 4 5 6 7 8 9 + – . e E}

Maximální počet platných míst je jedenáct: 1,0e–36 je nejmenší a 1,0e+36 je největší kladné číslo, která je možné zadat. Při překročení tohoto rozsahu je vypisováno chybové hlášení.

Při označení mocnin 10 se používá písmeno e nebo E (tzv. semilogaritmický zápis), např.: 0,5e-3 = 0,5E-3 = 0,0005 = 0.5*10^-3.

UPOZORNĚNÍ
Jazyk Coach používá jako oddělovače desetinných míst desetinnou tečku (.).
V textu této příručky je však používána jako oddělovač desetinných míst (ve shodě s národními zvyklostmi) desetinná čárka. Použití desetinné čárky jako oddělovače kompilátor *jazyka Coach* nepřipouští a vypisuje chybové hlášení.
Číslo nesmí začínat desetinnou tečkou a kompilátor vyžaduje zápis "0.1", nikoliv ".1".

2.3 Znaky a klíčová slova

Znaky

Některé *znaky* mají zvláštní význam a nemohou být použity ve jménech konstant, proměnných, funkcí a procedur:

{ + – * / ÷ ^ √ = > < . , ( ) : ; π ' }

Také některé *dvojice znaků* mají zvláštní význam:

{<= >= := <>}

Klíčová slova

Zvláštní význam mají i některá *slova,* která se nesmí používat pro označení konstant, proměnných, funkcí nebo procedur. Tato slova jsou jmenovitě vypsána v následující tabulce. Klíčová slova On (**Zap**), Off (**Vyp**) a Pi (π) označují konstanty, které mají hodnotu 255, 0, resp. 3,141592654 a které **nesmí být měněny.**

Abs	Fac	ResetAbsolute
And	Function	Round
ArcCos	If	RunningTime
ArcSin	Interval	SaveData
ArcTan	Level	Set
Becomes	Ln	SetAbsolute
Bit	Log	Sign
Column	Max	Sin
Cos	Min	Sound

Count	Not	Sqr
Counter	Off	Sqrt
Do	On	Stop
Else	Or	Stopwatch
EndDo	Pi	Tan
EndFunction	Print	Then
EndIf	Procedure	Until
EndProcedure	Rand	Wait
EndRedo	Redo	Latched
Entier	Repeat	Unlatched
Exp	Reset	While

Obr. 2.2: Přehled klíčových slov jazyka Coach

3. Výrazy

Výraz použitý v programu má číselnou nebo logickou hodnotu. Výrazy mohou být jednoduché nebo složené, např:

– samostatná čísla jako 6.13 nebo 105 a dále výsledky základních aritmetických operací s nimi, např. "6.13 +105", - konstanty (Pi, On, Off),
– proměnné, viz §3.1,
– volané funkce, viz §5.

3.1 Proměnné

Jméno (identifikátor) proměnné může být považováno za "úkryt" číselné hodnoty. Byla-li jménu proměnné přiřazena hodnota *aritmetického výrazu,* znamená to, že se tato hodnota uchovává v paměti počítače na místě označeném jménem proměnné, např. "Součet := 5 + 6" znamená, že v paměti počítače má proměnná "Součet" hodnotu "11". Pro identifikátory proměnných platí omezení podle §2.

Výsledkem *logického výrazu* je rovněž hodnota uchovávaná v paměti. Proměnná je pak označována jako *logická proměnná* (nebo booleovská proměnná).

Je-li výsledkem logického výrazu hodnota **On** (Zap, Ano, True, Pravda, logická jednička), zapisuje se do paměti číselná hodnota 255. Je-li výsledkem **Off** (Vyp, Ne, False, nepravda, logická nula), zapisuje se číselná hodnota 0.

Proměnné používané v programech mají obvykle *globální* význam, což znamená, že jsou dostupné v celém zbytku programu stejně jako v procedurách a funkcích, viz §5. *Lokální* proměnné se používají jen v definicích procedur a funkcí a ve zbytku programu je nelze přímo použít.

3.2 Aritmetické a logické operace

Výrazy se tvoří z proměnných (příp. konstant nebo funkcí) a operačních znamének (operátorů), které vyjadřují matematické operace mezi nimi. Z hodnot jedné či více proměnných obsažených ve výrazu se tak vypočítává jediná hodnota výrazu.

Operace je tedy předpis, podle kterého se z hodnot proměnných vypočítává jiná hodnota výrazu. Ve výrazech s více operacemi je pořadí výpočtu určeno tzv. prioritou jednotlivých operací, např.:

3*(V+1) - operace násobení "*" působí mezi hodnotami "3" a "(V+1)".

Jestliže operační znaménko působí na jedinou hodnotu, musí být tato hodnota zapsána za znaménkem. Jestliže operace probíhá mezi dvěma hodnotami, pak se znaménko zapisuje mezi nimi. Nejdříve se vykonávají operace s prioritou jedna, pak s prioritou dvě atd.

Aritmetické operace

Operace	Popis činnosti	Počet hodnot	Priorita
–	opačná hodnota	1	1
^	mocnina	2	1
*	násobení	2	2
/	dělení	2	2
+	sčítání	2	3
–	odčítání	2	3

Obr. 3.1: Aritmetické operace a pořadí jejich priority

V aritmetice mají operace 'sčítání' a 'odčítání' stejnou prioritu. Obdobně je tomu s operacemi 'násobení' a 'dělení'. Jsou-li ve výrazu použity operace stejné priority, výraz se vyhodnocuje zleva doprava.

Relační operátory

Při vzájemném porovnávání dvou hodnot se používají relační operátory. výsledkem porovnávání je logická hodnota Ano nebo Ne (v *jazyce Coach* odpovídají hodnotám On nebo Off, viz §3.1). Všechny relační operátory mají stejnou prioritu.

Pomocí logických operátoru mohou být vytvářeny složitější výrazy s relačními operátory pro porovnávání více hodnot (relační výrazy). V takových výrazech musí být vzájemně porovnávané dvojice hodnot vždy umístěny do kulatých závorek.

Operátor	Význam	Počet hodnot	Priorita
=	je rovno	2	4
<>	není rovno	2	4
<	menší než	2	4
>	větší než	2	4
<=	menší nebo rovno	2	4
>=	větší nebo rovno	2	4

Obr. 3.4: Relační operátory

3.3 Syntaxe výrazů

Použití mezer a klávesy <Enter>

V logických výrazech musí být identifikátory odděleny od operátoru *mezerami*. V aritmetických nebo relačních výrazech není použiti mezer jako oddělovačů povinné. výrazy se nemusí zapisovat na samostatnou řádku. Jestliže je výraz složitější a obsahuje více operátoru, bývá vhodné jej pro větší srozumitelnost rozepsat na více řádek.

Výrazy musí být vzájemně odděleny mezerou nebo znakem <Enter>, který se do textu programu vloží po stisknutí klávesy <Enter>. Jsou-li dva nebo více výrazU odděleny mezerou, mohou být zapsány na stejnou řádku. Při použití oddělovače <Enter> se každý výraz zapíše na samostatnou řádku.

***Použití závorek* ( )**

Kulaté závorky se použijí tehdy, je-li zapotřebí změnit prioritu operací. Mohou být použity také pro zpřehlednění zápisu výrazu.

4. Příkazy

Příkaz je takový prvek programu, který může být proveden samostatně. Nadále budeme rozlišovat mezi jednoduchými a složenými příkazy.

4.1 Jednoduché příkazy

Přiřazení

Příkazem *přiřazení* se původní hodnota proměnné nahradí výsledkem vypočteného výrazu. Symbolem přiřazovacího operátoru je ' := ' *('dvojtečka rovnítko')*.

V *jazyce Coach* je rovněž povoleno použití samostatného symbol ' = ' (*'rovnítko')* nebo slova ' becomes ' (stává se, nabývá hodnoty). Oddělování operátorů pomocí mezer není nezbytné:

```
proměnná := výraz
```

Příklady:
```
X:= Y + Z
Mokro becomes (déšť) and (not(deštník))
```

Volání procedury vzniklé spojením několika příkazů

Procedura se skládá z několika příkazů a je označena identifikátorem (jménem). Příkazy obsažené v proceduře se vykonávají na tom místě programu, na kterém je zapsáno její jméno (tzv. volání procedury). Definice procedury často obsahuje několik parametrů, viz §5.1. Používáním procedur se programy zpřehledňují a zlepšuje se jejich srozumitelnost.

4.2 Podmíněné příkazy typu 'If … Then ... ' (Jestliže ... Pak ...)

Podmíněné příkazy 'If ... Then .. .' (Jestliže ... Pak ...) mohou být dvojího druhu:

```
If Výraz                    Then Jestliže Výraz Pak
  Příkazy          tj.        Příkazy
EndIf                       KonecJestliže
```

Příkazy uvedené mezi klíčovými slovy Then a EndIf se provedou jen tehdy, má-li výraz následující za klíčovým slovem If logickou hodnotu On (Ano).

If Výraz Then
  Příkazy
Else
  Příkazy
Endlf
	tj	Jestliže Výraz Pak
		  Příkazy
		Nebo
		  Příkazy
		KonecJestliže

Výpočet výrazu zapsaného mezi klíčovými slovy If a Then musí dávat logickou hodnotu On (Ano) nebo Off (Ne). Jestliže má výraz logickou hodnotu On, pak se vykonají příkazy zapsané mezi klíčovými slovy Then a Else. Je-li naopak logická hodnota tohoto výrazu Off, vykonají se příkazy zapsané mezi klíčovými slovy Else a EndIf.

Několik příkladů výše uvedených podmíněných příkazů je uvedeno v následující tabulce na obr. 4.1.

IfX> 0 Then Y:= sqrt(X) Endlf	Je-li splněna podmínka (X > 0), přiřadí se proměnné Y hodnota funkce sqrt(X).

If (a < 2) and (b > 5) Then
  a := a + 1 b:= b - 5
Else
  a := a - 1 b:= b + 3
Endlf	

If a> b Then
  Maximum = a
Else
  Maximum = b
Endlf	

If a = 1 Then b := c Else
  If a = 2 Then b := -c Else
  b:= O
  Endlf
Endlf
	Hodnota přiřazená proměnné b závisí na hodnotě proměnné a.
	
	Pozor! Dva podmíněné příkazy jsou vnořeny do sebe a každý z nich musí být ukončen klíčovým slovem Endlf!

Obr. 4.1 Příklady použití podmíněného příkazu *'If ... Then ... '*

Podmíněný příkaz 'Repeat ... Until ... ' (Opakuj ... Dokud ...)

Příkazy zapsané mezi klíčovými slovy Repeat a Until jsou vykonávány tak dlouho, dokud není splněna podmínka uvedená za klíčovým slovem Until. Příkaz má syntaxi:

Repeat Příkazy Until Výraz tj. Opakuj Příkazy Dokud Výraz

Příkazy a výraz musí být odděleny od klíčových slov Repeat a Until mezerami. Příkazy se nejprve jedenkrát provedou a teprve pak se vyhodnocuje, zda je logická hodnota výrazu On (Ano) nebo Off (Ne). Jinak řečeno, příkazy se poprvé provedou bez ohledu na to, jaký je výsledek vyhodnocení výrazu.

Výsledkem výrazu musí být logická hodnota. Je-li výsledkem výrazu logická nula Off (Ne), provedení příkazů se opakuje. Je-li naopak výsledkem logická jednička On (Ano), pak se smyčka příkazů 'Repeat ... Until ...' ukončí a program pokračuje dále následujícím příkazem.

n:= 1 Repeat n:= n*2 Until n > 1000	Zdvojnásobování hodnoty n se opakuje tak dlouho, dokud hodnota n nepřesáhne 1 000 (tj. poslední a tím výsledná hodnota n bude 1024).
Repeat Záblesk Until Level(1) >= 4	Provádění procedury záblesk se opakuje tak dlouho, dokud hodnota level(1) (úroveň napětí v 1. kanálu karty AUR) je větší nebo rovna 4 (V).

Obr. 4.2 Příklady použití podmíněného příkazu *'Repeat ... Until ...' (Opakuj ... Dokud ...)*

Podmíněný příkaz 'ReDo ... EndRedo' (Opakuj ... KonecOpakování)

Tento podmíněný příkaz provádí pevně stanovený počet opakování skupiny příkazů a jeho syntaxe je následující:

Redo Výraz		Opakuj Výraz
Příkazy	tj.	Příkazy
EndRedo		KonecOpakování

Výraz a příkazy musí být odděleny mezerou jak navzájem, tak od klíčových slov ReDo a EndRedo. Výsledkem výrazu musí být číselná hodnota, která se chápe jako hodnota čítače a která se snižuje o jedničku po každém opakování skupiny příkazů (smyčky). Opakování příkazů pokračuje tak dlouho, dokud hodnota v čítači nedosáhne nuly, resp. hodnoty menší než nula. Několik příkladů použití je uvedeno na obr. 4.3.

n := 5 y:= n x:= 1 Redo n-1 x:= x*y y:= y-1 EndRedo	Výpočet faktoriálu n!
n:= 2.001 Redo n x:= sqr(x) EndRedo	Tento příkaz se třikrát opakuje.

Obr. 4.3 Příklady použití podmíněného příkazu *'ReDo ... EndRedo' (Opakuj ... KonecOpakování)*

Podmíněný příkaz 'While ... Do ... ' (Pokud ... Dělej ...)

Provádění skupiny příkazů se opakuje tak dlouho, dokud je logická hodnota podmiňujícího výrazu **On** (Ano). Příkaz má následující syntaxi:

While Výraz Do
Příkazy
EndDo
	tj.	Pokud Výraz Dělej
		Příkazy
		KonecDělej

Výraz a příkazy musí být odděleny mezerou jak navzájem, tak od klíčových slov While, Do a EndDo. Jestliže výraz nabude logické hodnoty Off (Ne), opakování příkazů se ukončí.

sum:= 0 While n > 0 Do sum:= sum + n n:= n-1 EndDo	Výpočet hodnoty součtu řady n + (n-1) + (n-2) .. + (n-i) pokračuje, pokud (n-1) je větší než 0, tj. dokud (n-i) <= 0

Obr. 4.4 Příklad použití podmíněného příkazu *'While...Do...' (Pokud…Dělej...*)

5. Procedury a funkce

Procedury a funkce zlepšují srozumitelnost programu. Obsahují skupinu několika příkazů, které se provádějí společně po vyvolání jména funkce, resp. procedury. Výsledkem provedení procedury je vykonání příkazů, které procedura obsahuje. Výsledkem provedení funkce je vypočítaná hodnota funkce.

V *jazyce Coach* je možno používat velkého počtu standardních (tzv. knihovních) funkcí a procedur, nebo zadefinovat v programech funkce a procedury nové.

5.1 Definování funkce

V *jazyce Coach* je syntaxe definované funkce následující:

```
Function Jméno_funkce(p1;p2;p3; ... )
    Příkazy
Jméno_funkce := výraz
EndFunction
```

Příkazové řádky funkce se zapisují mezi klíčová slova Function a EndFunction.

– Nejprve zapište *Jméno_funkce* včetně *parametrů* pl, p2, p3, atd., které jsou v příkazech a výrazech považovány za lokální proměnné. Identifikátor *Jméno_funkce* musí být oddělen od klíčového slova Function mezerou.
– Pak zapište potřebné *příkazy,* které musí být vzájemně odděleny mezerami nebo znakem <Enter>.
– Definice funkce musí být ukončena *přiřazovacím příkazem,* kterým se přiřadí identifikátoru *Jméno_funkce* hodnota výrazu vyjadřujícího hodnotu funkce. Znak přiřazení nemusí být oddělen mezerami.

Při definování funkce nezapomeňte, že:

– jméno funkce se **nesmí** shodovat s klíčovým (vyhrazeným) slovem, jménem proměnné, procedury nebo již dříve definované funkce,
– parametry definované v zadání funkce mají význam lokálních proměnných, identifikátory (jména) těchto parametrů musí být odlišné od ostatních a stejná jména nesmí být použita jinde v programu,
– parametry musí být vzájemně odděleny středníkem (;).

Je však dovoleno použít:

– globální proměnné v příkazech a měnit zde jejich hodnotu,
– funkci pro definování sama sebe (rekurzivní použití),
– při definování funkce jiné funkce, které již byly definovány.

5.2 Volání funkce

V programu se funkce používá tak, že se vhodné proměnné přiřadí hodnota funkce vypočtená podle funkčního předpisu definovaného výše popsaným způsobem. Příkaz k tomuto výpočtu a přiřazení (tzv. volání funkce) musí mít následující syntaxi:

```
x:= Jméno_funkce(a1;a2;a3; .. )
```

Počet parametrů použitých při volání funkce musí být shodný s počtem parametrů použitých v definici funkce (v hlavičce na řádce Function *Jméno_funkce(pl; p2; p3;...*). Jednotlivé parametry musí být odděleny středníkem (;).

Hodnoty parametrů zadané při volání funkce se po řadě jednoznačně přiřadí hodnotám lokálních proměnných použitých v definičním předpisu funkce. Při výpočtu funkční hodnoty pak program pracuje s těmito lokálními proměnnými a po ukončení výpočtu jsou všechny lokální proměnné z paměti vymazány.

Function Pythagoras(a;b) Pythagoras := √(a*a + b*b) EndFunction x:=3 y:=4 z := Pythagoras(x;y) k := Pythagoras(3*5;max(x;y))	Definice funkce Volání funkce Stejný význam má k := Pythagoras(15;4).

Function Factoria/(a)
  If a> 1 Then
  Factorial := a*Factorial(a-1)
  Else
  F actorial := 1
  Endlf
EndFunction
x:= Factorial(7)
	Rekurentní předpis pro výpočet faktoriálu, který se ukončí po dosažení hodnoty a=1.
	
	Výpočet hodnoty 7!

Obr. 5.1 Příklady správně definovaných a použitých funkcí

5.3 Definování procedur

Postup při definování procedury je velmi podobný výše popsanému definování funkce. Rozdíl spočívá v tom, že definice funkce musí končit přiřazovacím příkazem *Jméno_funkce := výraz,* jehož obdobu definice procedury nemusí obsahovat. Syntaxe definice procedury je následující:

```
Procedure Jméno_procedury(p1;p2;p3; .. )
  Příkazy
EndProcedure
```

Příkazy vykonávané po vyvolání procedury se zapisují mezi klíčová slova Procedure a Endprocedure.

– Nejprve zapište *Jméno_procedury* včetně *parametrů* p1, p2, p3, atd., které jsou v příkazech považovány za lokální proměnné. Identifikátor *Jméno_procedury* musí být oddělen od klíčového slova Procedure mezerou.
– Pak zapište potřebné *příkazy,* které musí být vzájemně odděleny mezerami nebo znakem <Enter>.

Při definování procedury nezapomeňte, že:
– jméno procedury se **nesmí** shodovat s klíčovým (vyhrazeným) slovem, jménem proměnné, funkce nebo již dříve definované procedury,
– parametry definované v zadání procedury mají význam lokálních proměnných, identifikátory (jména) těchto parametrů musí být odlišné od ostatních a stejná jména nesmí být použita jinde v programu,
– parametry musí být vzájemně odděleny středníkem (;).

Je však dovoleno použít:
– globální proměnné v příkazech a měnit zde jejich hodnotu,
– volání již definované funkce v definici procedury.

5.4 Volání procedur

Příkaz pro volání procedury musí mít následující syntaxi:

```
Jméno_procedury ( a1;a2;a3;...)
```

Počet parametrů použitých při volání procedury musí být roven počtu parametrů použitých v definici procedury (v hlavičce na řádce Procedure *Jméno_procedury(pl;p2;p3;...)*). Jednotlivé parametry musí být odděleny středníkem (;).

Hodnoty parametrů zadané při volání procedury se po řadě jednoznačně přiřadí hodnotám lokálních proměnných použitých v definici procedury. Při provádění procedury pak program pracuje s těmito lokálními proměnnými.

Procedure Bliknutí(t1;t2;n) Redo n SetAbsolute(1;3;5;7) Wait(t1) ResetAbsolute(1;3;5;7) Wait(t2) EndRedo EndProcedure Bliknutí(2;3;15)	Procedura Bliknutí n-krát provede následující smyčku: Na dobu t1 [s] nastaví bity č. 1, 3, 5 a 7 výstupního portu na úroveň log. 1 a ostatní bity na úroveň log. 0. Pak na dobu 12 [s] nastaví výše vyjmenované bity výstupního portu na úroveň log. 0 a ostatní bity na úroveň log. 1.

Obr. 5.2 Příklad správně definované procedury a jejího volání

5.5 Standardní funkce

Jazyk Coach umožňuje využívat přibližně 20 standardních matematických funkcí, které jsou uvedeny v následující tabulce. Syntaxe příkazu pro volání těchto funkcí je tato:

```
y := f(x1;x2;...)
```

Věnujte pozornost správnému použití závorek při zápisu složitějších funkcí, např.:

$\sin^2(x)$ musí být zapsáno ve tvaru sin(x)^2 nebo (sin(x))^2 ,

$\sin(x^2)$ musí být zapsáno ve tvaru sin(x^2), zápis sin^2(x) způsobí chybu a výpis chybového hlášenÍ.

Funkce je možno kombinovat a vytvářet tzv. složené funkce.

Identifikátor	**Význam**	**Parametry** *
sin (x)	sinus argumentu x	x se zadává v radiánech
cos(x)	kosinus argumentu x	x se zadává v radiánech
tan(x)	tangens argumentu x	x se zadává v radiánech $x \neq (1/2+n)*\pi$, $n \in Z$
arcsin(x)	arkus sinus (výsledek v rad)	$x \in (-1;1)$

arccos(x)	arkus kosinus (výsledek v rad)	x ∈ (-1;1)
arctan(x)	arkus tangens (výsledek v rad)	
sqr(x)	druhá mocnina x	x ∈ (-√(max); √(max»
sqrt(x)	druhá odmocnina x	x ∈ (0; max)
√(x)	druhá odmocnina x	x ∈ (0; max)
ln(x)	přirozený logaritmus (základ e)	x ∈ (0; max)
log (x)	dekadický logaritmus (základ 10)	x ∈ (0; max)
exp(x)	x-tá mocnina čísla e, tj. (e^x)	x ∈ (-max; 81)
entier(x)	zaokrouhluje číslo dolů: entier(2,8)=2; entier(-3,2)=-4	
round(x)	běžné zaokrouhlování: round(2,8) = 3; round(-3,2) = -3	
abs(x)	absolutní hodnota x	
rand	generátor náhodného čísla z intervalu (0; 1)	
max(x1;x2; ..)	výběr největšího z i parametrů x1, x2, ... ,xi	jeden nebo více parametrů
min(x1;x2; ..)	výběr nejmenšího z i parametrů x1, x2, ... ,xi	jeden nebo více parametrů
sign(x)	funkce signum má hodnotu: -1 pro x < 0, +1 pro x > 0 a 0 pro x = 0	
fac(x)	výpočet hodnoty (round(x))! (factorial)	x E (0;+33,5)

*max $\approx 10^{15}$

Obr. 5.3 Standardní funkce jazyka Coach

6. Chybová hlášení

V programu se mohou vyskytnout chyby dvou různých druhů:
– *syntaktické chyby*, které jsou zjišťovány při překladu (kompilaci) modelu nebo řídícího programu; syntaktická správnost se ověřuje před výpočtem programu nebo při změně škálování os, resp. proměnných vynášených na jednotlivé osy grafu,
– *chyby vznikající při běhu programu*, které jsou průběžně zjišťovány při programovaných výpočtech

6.1 Chybová hlášení vypisovaná při překladu

Je-li při překladu (kompilaci) zjištěná syntaktická chyba, zobrazí se text programu na displeji a na stavovou řádku se vypíše chybové hlášení. Kurzor bliká za pozicí, na které byla chyba zjištěna, a tato chyba může být okamžitě opravena.

```
t := t + dt
If t > 13 Then a := F/m
Else
a:= F/(2*m)
v:= v + a*dt
x:= x + v*dt
```

Obr. 6.1 Model se syntaktickou chybou

Kurzor nemusí vždy blikat právě na pozici skutečné chyby. V modelu uvedeném na obr. 6.1 chybí ukončení podmíněného příkazu klíčovým slovem 'EndIf'. Kompilátor však tuto chybu nezjistí, dokud neprojde všemi příkazy za klíčovým slovem Else.

V tomto případě se vypíše chybové hlášení Kompilátor očekává: "EndIf" a blikající kurzor se objeví až za posledním znakem posledního příkazu, nikoliv za přiřazovacím příkazem "a := F/(2*m)"

Chybové hlášení	***Příčina***
Znak není očekáván	Znak se nachází na pozici, na které by neměl být.
Toto není číslo	Kompilátoru se nepodařilo přečíst číslo.
Toto jméno již má jiný význam	Chyba způsobená použitím stejného identifikátoru (jména) pro více různých proměnných, funkci nebo procedur.

Různé typy proměnných	Chyba způsobena pokusem o přirazení nějaké hodnoty identifikátoru procedury, resp. výsledku funkce mimo její definici.
Kompilátor očekává: „...“	Kompilátor očekává jméno, znak nebo symbol.
Kompilátor nechce „...“	Daný znak nebo symbol kompilátor nepřijal (příčina této chyby není většinou okamžitě zřejmá).
Příliš mnoho proměnných	Paměť vyhrazená pro jména proměnných byla vyčerpána; použijte kratší identifikátory.
Funkční hodnota není přiřazena	Definice funkce neobsahuje přiřazení *Jméno_funkce := výraz* (pro výpočet hodnoty funkce)

Obr. 6.2 Chybová hlášení, která se mohou při kompilaci vyskytnout

6.2 Chybová hlášení při běhu programu

I když byl program úspěšně přeložen, není ještě zaručeno, že se po jeho spuštění neobjeví další chyby. Jestliže se tak stane, programový výpočet se ukončí a na obrazovku se vypíše chybové hlášení

Chybové hlášení	***Příčina***
Dělení nulou.	Proměnná ve jmenovateli zlomku nabyla nulové hodnoty.
Hodnota mimo povolený rozsah.	Číslo je příliš velké nebo příliš malé.
Hodnota nepatří do def. oboru funkce.	Hodnota argumentu standardní funkce není prvkem jejího definičního oboru (intervalu).

Obr. 6.3 Nejčastější chyby vznikající při programovém výpočtu a odpovídající chybová hlášení