**function**

*Задание пользовательской функции.*

**Синтаксис:**

**function**  <Имя функции> (<имя входной переменной 1> {:<тип входной переменной 1>} {=<инициирующее выражение 1>} {,**out** <имя выходной переменной> {:<тип выходной переменной>} {=<инициирующее выражение>}}){:<тип выходного значения>}

<операции выполняемые при вызове функции>;

**end**;

**Описание:**

Задание новой пользовательской функции. Значение функции может быть не присвоено. Выходные переменные декларируются словом **out** – в такую переменную функция записывает значение. Если тип значения функции не указан, то он определяется автоматически по типу выражения, где присваивается выходное значение функции. Вместо **function** можно использовать **procedure**. Если типы параметров функции не указаны, то при создании кода функции типы принимаются как у операндов при соответствующем вызове.

**Пример 1:**

|  |  |
| --- | --- |
|  | **function** PolyF(A,B)  PolyF = A\*A + B\*B;  **end**; |

В примере 1, если при вызове операнды A и B будут действительными числами, то результат тоже будет действительным числом. Если A и B будут векторами, то результат будет вектором.

В тексте программы примера 2, одна и та же функция может быть по-разному определена несколько раз, а также переопределена под другим именем в соответствии со следующим форматом:

**function** <имя функции>=<имя ранее определённой функции>

**Пример 2:**

|  |  |
| --- | --- |
|  | **function** SumQ(A,**out** R:integer):complex  *//Присваивание значений функции*  SumQ = sqr(A);  *//Присваивание выходной переменной*  R = round(real(SumQ));  **end**;  *//Переопределение SumQ как FSum*  **function** Fsum=SumQ; |

Функции и процедуры допускают также рекурсивный вызов, пример которого представлен в скрипте примера 3.

Пример 3

|  |  |
| --- | --- |
|  | **procedure** ScanCurrentContainer(contid: integer)  **for**(i=1,**getobjcount**(contid)) **begin**  **var** id,imagecontid,submodelid: integer;  *//Получаем идентификатор*  id= **getobj**(contid,i);  *//Получаем выводим сообщение*  **seterrorstr**(**getobjname**(id)+":"+ **getobjclassname**(id),0);  *//Получаем идентификатор контейнера изображения*  imagecontid = **getobjcontainer**(id);  *//Рекурсивный вызов функции, если у неё есть вложенный контейнер*  **if** imagecontid<>0 **then ScanCurrentContainer**(imagecontid);  *//Обход содержимого субмоделей*  submodelid = **getsubmodelid**(id);  if submodelid <> 0 then **ScanCurrentContainer**(submodelid);  **end;**  **end;**  **initialization**  **ScanCurrentContainer(getmainpageid);**  **end;** |

После запуска скрипта примера 3, в окне сообщений выводится информация, содержащая имена и типы всех блоков проекта. При этом обход начинается с главной страницы и если блок содержит вложенные контейнеры, то процедура вызывает сама себя, находясь во вложенных контейнерах.

Язык программирования позволяет создавать также библиотеки функций в виде текстовых файлов. Это может быть сделано при помощи ключевого слова **include**. Оно позволяет загрузить фрагмент программы из текстового файла.