**Процедуры и функции**

Функция – подпрограмма, в которой выполняется некоторый набор действий, в результате чего имя функции получает некоторое единственное значение, передаваемое в вызываемую подпрограмму, при этом управление при выходе из функции возвращается в точку ее вызова.

Процедура – подпрограмма, в которой производится некоторый набор действий, при этом имени процедуры ничего не присваивается: процедура не имеет возвращаемого значения. Результаты в вызывающую программу передаются через механизм параметров или глобальные переменные. Управление возвращается за точку вызова процедуры.

*Локальные const, var, type, label – описательная часть.*

**Сравнение процедур и функций.**

- Функция обязательно имеет тип возвращаемого значения. Процедура возвращаемого значения не имеет.

- Внутри тела функции – обязательно присвоение имени значения.

- Процедура вызывается отдельно с помощью оператора вызова с передачей параметров, функция – вызывается в операторе присваивания.

- Функция всегда возвращает хотя бы 1 результат; процедура не возвращает ничего.

*Модель дальнего вызова подпрограмм.*

Спецификатор far, объявленный около имени процедуры, включает директиву дальнего вызова. В данном режиме компилятор определяет модель памяти, для которой любые указатели воспринимаются как дальние, то есть, данные могут располагаться в сегменте, отличном от сегмента программы.

*Предварительное описание подпрограмм.*

В ряде случаев необходимо выполнять предварительные описания процедур или функций. Например, пусть существует процедура a, которая в ходе своей работы выполняет обращение к процедуре b. Следовательно, процедура b должна быть объявлена и известна компилятору до момента вызова (т.е. выше процедуры a).

*Внешнее описание подпрограмм.*

В процессе накопления опыта программирования может быть создан набор подпрограмм, удобных для дальнейшего использования. Обычно такие функции объединяют в библиотеки. Библиотека представляет собой откомпилированный файл, не являющийся программой.

В таком случае объектный код содержится в некотором \*.obj-файле, а само подключение производится директивой {$L имя файла}. После описания заголовка подпрограммы из \*.obj-файла необходимо задать служебное слово external. Это укажет компилятору, что вызываемую подпрограмму с данным именем следует искать в одном из подключенных объектных файлов.

*Механизм параметров. Формальные и фактические параметры. Параметры значения, переменные, константы, нетипизированные параметры***.**

Организация обмена данными происходит с помощью механизма параметров.

Все данные можно разбить на 4 основных класса:

- значения

- ссылки

- константы

- нетипизированные параметры

При организации вызова подпрограмм в оперативной памяти в специальной области, называемой областью рекурсивного стека, создается копия рабочих полей подпрограммы, содержащая всю локальную для этой подпрограммы информацию, необходимую для ее выполнения. Рекурсивный стек состоит из страниц и заполняется по мере вызова подпрограмм.

1 – указатель связи (адрес вызываемой подпрограммы)

2 – локальные переменные, объявленные в описательной части подпрограммы

3 – адрес возврата (куда будет передано управление)

4 – параметры из заголовка вызываемой подпрограммы, при этом список записывается в страницу стека в прямом порядке

5 – возвращаемое значение, если таковое имеется.

После завершения подпрограммы страница стека полностью удаляется.

*Формальные и фактические параметры.*

Обмен информацией между вызываемой и вызывающей подпрограммой осуществляется с помощью механизма передачи параметров. Переменные, указанные в заголовке подпрограммы называют формальными параметрами. Эти переменные используются только внутри подпрограммы.

Переменные, указанные в строке вызова подпрограммы называются фактическими параметрами. Их значение зависит от механизма передачи параметра. Для вызываемой подпрограммы данные переменные являются глобальными.

Между фактическими и формальными параметрами устанавливается взаимооднозначное соответствие. Количество, типы и порядок следования параметров должны совпадать.

Параметры значения: используются для передачи данных в подпрограмму, при этом создается копия передаваемого значения. Изменения таких параметров не видны на внешнем уровне. Тип формального и фактического параметра должен быть совместим по присваиванию.

Если в вызывающей программе описаны переменные X: Integer, C: Byte; Y: Longint;, то синтаксически правильными вызовами P будут P(X), P(C), P(Y), P(200).

Основной недостаток – временные затраты, необходимые для создания копии передаваемого параметра, затрата памяти в стеке и его переполнение.

Параметры ссылки (параметры-переменные): используются для организации передачи результатов работы подпрограммы через параметры, при этом передается адрес параметра без создания его копии.

Все изменения, производимые с параметром, передаваемым по ссылке, видны после вызова подпрограммы.

Параметры-константытак же, как и параметры-переменные, позволяют передавать в подпрограмму данные по ссылке, но в отличие от параметров-переменных модификация переданных значений внутри подпрограммы невозможна и эта ситуация будет отслеживаться компилятором. Параметры-константы используются для передачи в подпрограмму больших объемов данных с целью экономии динамической памяти.

Такие параметры передаются с указанием ключевого слова const и приводят в текущую область стека адрес передаваемого значения.

Нетипизированные параметры: В ряде случаев необходимо передавать параметры, тип которых в момент описания функциинеизвестен. Такие параметры называют нетипизированными, при этом тип формального параметра в заголовке функции не указывается, а производится подстановка типа фактического параметра.

Замечание: при работе с нетипизированными параметрами следует учитывать, что в качестве нетипизированных параметров могут использоваться только параметры-переменные. Для использования нетипизированных параметров необходимо выполнять приведение типов. Данное приведение должно быть явным. Явное приведение - указание имени типа перед именем переменной.

*Процедурные типы. Параметры-функции и параметры-процедуры.*

Передаваемым в подпрограмму параметром может также быть процедура или функция (указатель на нее), т.е. параметр процедурного типа.

Для параметров-функций и параметров-процедур существуют следующие правила: подпрограммы должны иметь ключевое слово far; они не должны являться стандартными подпрограммами; не должны объявляться внутри других подпрограмм; не должны иметь директив inline или interrupt.

Назначение процедурных типов состоит в том, чтобы дать программисту гибкое средство для передачи процедур или функций в качестве фактических параметров. Для объявления процедурного типа используется заголовок процедуры или функции без указания её имени с определением набора параметров, а для функции - возвращаемого значения.

Пример:

Type

F1 = procedure;

F2 = function(X: Integer; var S: String): Boolean;

function F2(X: Integer; var S: String): Boolean; far;

Begin

end;

procedure F(Func: F2);

Begin

end;

*Литература*

[*https://infopedia.su/11x1664.html*](https://infopedia.su/11x1664.html)