**16. Подпрограммы. Формальные и фактические параметры.**

Формальные параметры указывают, с какими параметрами следует обращаться к этой процедуре (функции), т.е. их количество, последовательность, тип. Они задаются в заголовке процедуры (функции) в виде списка параметров, разбитого на группы (секции) однотипных параметров, разделенных ‘;’. Порядок следования групп (секций) произвольный. Количество параметров м.б. любым.  
        Для каждого формального параметра следует указать имя и тип. Имена параметров могут быть любыми, они могут совпадать с глобальными именами программы. Необходимо лишь помнить, что в этом случае параметр основной программы с таким именем становится недоступным для непосредственного использования подпрограммой. Но формальные имена не могут совпадать с локальными переменными!!!!, т.к. они располагаются в одном стековом сегменте памяти. Тип формального параметра может быть практически любым, однако в заголовке процедуры (функции) нельзя вводить новый тип.

**Фактическими параметрами** называются параметры, которые задаются при вызове процедуры (функции).

**Процедура вызывается по имени:**

|  |
| --- |
| <имя процедуры> {(фактические параметры)}; |

**Правила задания фактических параметров:**

1. Фактические параметры должны совпадать с формальными по количеству, по порядку следования, по типу.

2. Если формальный параметр – параметр-переменная, то фактический параметр при вызове процедуры задается только идентификатором (вызов по ссылке).

3. Если формальный параметр – параметр- значение, то фактический параметр при вызове процедуры задается:

* идентификатором
* константой
* выражением

4.  Для параметров-переменных (вызов по ссылке) должно быть точное совпадение по типу фактических и формальных параметров, а для параметров-значений (вызов по значению) допустима совместимость.

**17. Подпрограммы. Механизм передачи параметров. Параметры – значения. Параметры – переменные.**

**Параметрами - значениями** называются переменные, которые могут изменяться внутри процедуры (функции), но не передают свои значения в программу.

**Параметрами - переменными** называются переменные, которые изменяются в процедуре (функции) и передают свое значение в программу.

Перед параметрами-переменными ставится служебное слово **Var.**        Список формальных параметров обеспечивает связь процедуры (функции) с вызывающей программой. Через него в процедуру (функцию) передаются исходные данные и возвращается результат.

        При вызове процедуры (функции) из программы указываются фактические параметры.

**18. Рекурсия и рекурсивные алгоритмы. Рекурсивная триада. Рекурсивный спуск и рекурсивный подъем.**

**Рекурсия** – это определение объекта через обращение к самому себе.

**Рекурсивный алгоритм** – это алгоритм, в описании которого прямо или косвенно содержится обращение к самому себе. В технике процедурного программирования данное понятие распространяется на функцию, которая реализует решение отдельного блока задачи посредством вызова из своего тела других функций, в том числе и себя самой. Если при этом на очередном этапе работы функция организует обращение к самой себе, то такая функция является *рекурсивной.*

**рекурсивная триада** – этапы моделирования задачи, на которых определяется набор параметров и соотношений между ними. Рекурсивную триаду составляют параметризация, выделение базы и декомпозиция.

На этапе параметризации из постановки задачи выделяются параметры, которые описывают исходные данные. При этом некоторые дальнейшие разработки решения могут требовать введения дополнительных параметров, которые не оговорены в условии, но используются при составлении зависимостей. Необходимость в дополнительных параметрах часто возникает также при решении задач оптимизации рекурсивных алгоритмов, в ходе которых сокращается их временная сложность.

Выделение базы рекурсии предполагает нахождение в решаемой задаче тривиальных случаев, результат для которых очевиден и не требует проведения расчетов. Верно найденная база рекурсии обеспечивает завершенность рекурсивных обращений, которые в конечном итоге сводятся к базовому случаю. Переопределение базы или ее динамическое расширение в ходе решения задачи часто позволяют оптимизировать рекурсивный алгоритм за счет достижения базового случая за более короткий путь обращений.

Декомпозиция представляет собой сведение общего случая к более простым подзадачам, которые отличаются от исходной задачи набором входных данных. Декомпозиционные зависимости описывают не только связь между задачей и подзадачами, но и характер изменения значений параметров на очередном шаге. От выбранных отношений зависит трудоемкость алгоритма, так как для одной и той же задачи могут быть составлены различные зависимости. Пересмотр отношений декомпозиции целесообразно проводить комплексно, то есть параллельно с корректировкой параметров и анализом базовых случаев.

**Рекурсивный спуск** – движение вычислительного процесса в направлении последовательных рекурсивных вызовов.

**Рекурсивный подъем** – движение вычислительного процесса в направлении возврата из ранее вызванных копий рекурсивной процедуры.

**19. Работа в графическом режиме. Настройка палитры. Функции Pascal для работы с графическими примитивами.**

**Графические примитивы.**

SetPixel(x,y,color) – закрашивает один пиксел с координатами (x,y) цветом color;  
GetPixel(x,y) – возвращает текущее значение цвета для пиксела с координатами (x,y);  
Line(x1,y1,x2,y2) – рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2);  
Rectangle(x1,y1,x2,y2) – рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2);  
Ellipse(x1,y1,x2,y2) – рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1,y1) и (x2,y2);  
RoundRect(x1,y1,x2,y2,w,h) – рисует прямоугольник со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев;  
Circle(x,y,r) – рисует окружность с центром в точке (x,y) и радиусом r;  
Arc(x,y,r,a1,a2) – рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки);  
Pie(x,y,r,a1,a2) – рисует сектор окружности, ограниченный дугой (параметры процедуры имеют тот же смысл, что и в процедуре Arc);  
Chord(x,y,r,a1,a2) – рисует фигуру, ограниченную дугой окружности и отрезком, соединяющим ее концы (параметры процедуры имеют тот же смысл, что и в процедуре Arc);  
MoveTo(x,y) – передвигает невидимое перо к точке с координатами (x,y); эта функция работает в паре с функцией LineTo(x,y);  
LineTo(x,y) – рисует отрезок от текущего положения пера до точки (x,y); координаты пера при этом также становятся равными (x,y);  
TextOut(x,y,s) – выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s);  
FloodFill(x,y,c) – заливает область одного цвета цветом c, начиная с точки (x,y);  
FillRect(x1,y1,x2,y2) – заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти;  
Polygon(a,n) – строит ломаную по n точкам, координаты которых заданы в массиве a элементов типа TPoint ;  
Polyline(a,n) – строит замкнутую ломаную по n точкам, координаты которых заданы в массиве a элементов типа TPoint .

**Функции для работы с цветом.**

RGB(r,g,b) – возвращает целое значение, являющееся кодом цвета, который содержит красную, зеленую и синюю составляющие с интенсивностями r, g и b соответственно (r, g и b – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной). GetRed(color)– выделяет красную составляющую из цвета color (целое в диапазоне от 0 до 255);  
GetGreen(color)– выделяет зеленую составляющую из цвета color (целое в диапазоне от 0 до 255);  
GetBlue(color) – выделяет синюю составляющую из цвета color (целое в диапазоне от 0 до 255).

**Стандартные цвета**

|  |  |  |
| --- | --- | --- |
| clBlack – черный clPurple – фиолетовый clWhite – белый clMaroon – темно-красный clRed – красный clNavy – темно-синий clGreen – зеленый clBrown – коричневый clBlue – синий clSkyBlue – голубой clYellow – желтый |  | clCream – кремовый clAqua – бирюзовый clOlive – оливковый clFuchsia – сиреневый clTeal – сине-зеленый clGray – серый clLime – ярко-зеленый clLightGray – светло-серый clMoneyGreen – цвет зеленых денег clDarkGray – темно-серый |

**20. Работа в графическом режиме. Экранная и декартова системы координат.**

Обычное соглашение для экранных координат - установить верхний левый угол в качестве начала координат, пусть ось x проходит по ширине экрана, а ось y проходит по высоте экрана (см. Ниже). Это означает, что экранные координаты являются типом декартовых координат.

Однако обычно ось y направлена "вверх" при использовании декартовых координат - противоположно оси "экранных координат".

Декартова система координат - это система координат, которая позволяет вам находить местоположение точки относительно другой фиксированной точки, начала координат.