Процеду́рное программи́рование — это парадигма программирования, основанная на концепции *вызова процедуры*. Процедуры, также известны как подпрограммы, методы или функции (это не математические функции, но функции, подобные тем, которые используются в функциональном программировании). Процедуры просто содержат последовательность шагов для выполнения. В ходе выполнения программы любая процедура может быть вызвана из любой точки, включая саму данную процедуру.

Процедурное программирование — это лучший выбор, чем просто последовательное или неструктурированное программирование во многих ситуациях, которые вызываются умеренной сложностью, или тех, которые требуют значительного упрощения поддержки. Возможные выгоды:

> Возможность повторного использования одного и того же кода из нескольких мест программы без его копирования.
>
> Легче отследить поток выполнения программы, чем в случае использования инструкций `GOTO` или `JUMP`, которые могут сделать из большой, сложной программы так называемый «спагетти-код».
>
> Возможность поддержки модульности и структурности.

Процедуры и функции

В языках программирования высокого уровня используется два типа подпрограмм: процедуры и функции.

> Функция — это подпрограмма специального вида, которая, кроме получения параметров, выполнения действий и передачи результатов работы через параметры имеет ещё одну возможность — она может *возвращать результат*. Вызов функции является, с точки зрения языка программирования, выражением, он может использоваться в других выражениях или в качестве правой части присваивания. Подробнее см. в статье Функция (программирование).
>
> Процедура — это любая подпрограмма, которая не является функцией.

Побочные эффекты

Побо́чный эффе́кт функции — возможность в процессе выполнения своих вычислений читать и модифицировать значения глобальных переменных, осуществлять операции ввода/вывода, реагировать на исключительные ситуации, вызывать их обработчики. Если вызвать функцию с побочным эффектом дважды с одним и тем же набором значений входных аргументов, может случиться так, что в качестве результата вычисляются разные значения.

Такие функции называются недетерминированными функциями с побочными эффектами.

Вложенные функции и окружения

Подпрограмма (англ. *subprogram*) — поименованная или иным образом идентифицированная часть компьютерной программы, содержащая описание определённого набора действий. Подпрограмма может быть многократно *вызвана* из разных частей программы.

Некоторые языки программирования (например, Паскаль, Ада, Модула-2) допускают описание вложенных подпрограмм, то есть помещение подпрограмм внутрь других подпрограмм. Такие вложенные подпрограммы могут использоваться только в той подпрограмме, в которой они описаны. В иных случаях (например, в языке Си) вложение подпрограмм не допускается. Никаких принципиальных преимуществ вложение подпрограмм не даёт, но может быть удобно для более логичной структуризации программы (если какая-то подпрограмма используется только в некоторой другой подпрограмме, логично поместить первую во вторую).

Функции окружения – надо полагать, обычные глобальные функции, доступные в пределах модуля и всей программы в целом.

Рекурсия

В программировании рекурсия — вызов функции (процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная рекурсия), например, функция A вызывает функцию B, а функция B — функцию A. Количество вложенных вызовов функции или процедуры называется глубиной рекурсии.

Мощь рекурсивного определения объекта в том, что такое конечное определение способно описывать бесконечно большое число объектов. С помощью рекурсивной программы же возможно описать бесконечное вычисление, причём без явных повторений частей программы.

Имеется специальный тип рекурсии, называемый «хвостовой рекурсией». Интерпретаторы и компиляторы функциональных языков программирования, поддерживающие оптимизацию кода (исходного и/или исполняемого), выполняют хвостовую рекурсию в ограниченном объёме памяти при помощи итераций.

Следует избегать избыточной глубины рекурсии, так как это может вызвать переполнение стека вызовов.