**Процеду́рное программи́рование** — это [парадигма программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на концепции *вызова процедуры*. Процедуры, также известны как [подпрограммы](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), методы или функции (это не математические функции, но функции, подобные тем, которые используются в [функциональном программировании](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). 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](http://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) во многих ситуациях, которые вызываются умеренной сложностью, или тех, которые требуют значительного упрощения поддержки. Возможные выгоды:

* Возможность [повторного использования](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0) 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»](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B0%D0%B3%D0%B5%D1%82%D1%82%D0%B8-%D0%BA%D0%BE%D0%B4).
* Возможность поддержки модульности и [структурности](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

# Процедуры и функции

В языках программирования высокого уровня используется два типа подпрограмм: [процедуры](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) и [функции](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29).

* 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)](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29).
* Процедура — это любая подпрограмма, которая не является функцией.

# Побочные эффекты

**Побо́чный эффе́кт** функции — возможность в процессе выполнения своих [вычислений](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) читать и модифицировать значения [глобальных переменных](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F), осуществлять операции [ввода/вывода](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B2%D0%BE%D0%B4/%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4&action=edit&redlink=1), реагировать на [исключительные ситуации](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), 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.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *subprogram*) — поименованная или иным образом идентифицированная часть [компьютерной программы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), 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](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) рекурсия — вызов [функции](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) ([процедуры](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0)) 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](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирования, поддерживающие оптимизацию кода (исходного и/или исполняемого), выполняют хвостовую рекурсию в ограниченном объёме памяти при помощи [итераций](http://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F).

Следует избегать избыточной глубины рекурсии, так как это может вызвать переполнение [стека вызовов](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D0%BE%D0%B2).