4.2 Процедурное программирование. Процедуры и функции. Побочные эффекты. Вложенные функции и окружения. Непосредственная и косвенная рекурсия.

Процеду́рное программи́рование

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.

Процедурные языки программирования

- Ада (язык общего назначения)
- Бейсик (версии начиная с Quick Basic до появления Visual Basic)
- Си
- КОБОЛ
- Фортран
- Модула-2
- Глагол (русский аналог Оберона)
- Паскаль
- ПЛ/1
- Рапира
- REXX

Подпрограмма

Подпрограмма — поименованная или иным образом идентифицированная часть компьютерной программы, содержащая описание определённого набора действий. Подпрограмма может быть многократно вызвана из разных частей программы. В языках программирования для оформления и использования подпрограмм существуют специальные синтаксические средства.

Подпрограммы изначально появились как средство оптимизации программ по объёму занимаемой памяти. К настоящему времени данная функция подпрограмм стала

вспомогательной, главное их назначение — структуризация программы с целью удобства её понимания и сопровождения.

Выделение набора действий в подпрограмму и вызов её по мере необходимости позволяет логически выделить целостную подзадачу, имеющую типовое решение. Такое действие имеет ещё одно (помимо экономии памяти) преимущество перед повторением однотипных действий: любое изменение (исправление ошибки, оптимизация, расширение функциональности), сделанное в подпрограмме, автоматически отражается на всех её вызовах, в то время как при дублировании каждое изменение необходимо вносить в каждое вхождение изменяемого кода.

Даже в тех случаях, когда в подпрограмму выделяется однократно производимый набор действий, это оправдано, так как позволяет сократить размеры целостных блоков кода, составляющих программу, то есть сделать программу более понятной и обозримой.

Способ передачи параметров в подпрограмму

- Передача параметров по значению
- Передача параметров по ссылке
- Передача параметров по имени. В формальный параметр может быть помещено произвольное выражение. При этом вычисление этого выражения произойдёт внутри подпрограммы в тот момент, когда потребуется его значение. Если это значение фигурирует несколько раз, то и вычисляться оно будет тоже несколько раз. Параметры, передаваемые по имени, дают возможность писать довольно универсальные подпрограммы. Такой способ передачи параметров используется, к примеру в языках Алгол или Алгол 68.
- Передача параметров через стек. Это фактически разновидность передачи параметра по значению «с ручным приводом», в данном случае отсутствует понятие формальных и фактических параметров. Все параметры лежат на стеке, причём их типы, количество и порядок не контролируются компилятором. Данный подход реализован в языке Форт.

Виды подпрограмм

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.

Процедура — это любая подпрограмма, которая не является функцией.

Побочные эффекты

Побочным эффектом подпрограммы называется любое изменение функцией состояния программной среды, кроме возврата результата (изменение значений глобальных переменных, выделение и освобождение памяти, ввод-вывод и так далее). Теоретически наиболее правильным является использование функций, не имеющих побочного эффекта (то есть таких, в результате вызова которых возвращается вычисленное значение, и только), хотя на практике приходится использовать функции с побочным эффектом, хотя бы для обеспечения ввода-вывода и отображения результатов работы программы. Существует специфическая парадигма программирования — функциональное программирование, в которой любая программа представляет собой набор вложенных вызовов функций, не вызывающих побочных эффектов. Наиболее известный язык программирования, реализующий эту парадигму — Лисп.

4.2

Вложенные процедуры

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 (непосредственная) или через другие функции (косвенная), 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.