# Выделение памяти

***Автоматическое хранилище***

*Обычные переменные, объявленные внутри функции, используют автоматическое хранилище и называются втоматическими переменными. Этот термин означает, что они создаются автоматически при вызове содержащей их функции и уничтожаются при ее завершении. Например, массив temp в листинге 4.22 существует только во время работы функции getname (). Когда управление программой возвращается main (), то память, используемая temp, освобождается автоматически. Если бы getname () возвращала указатель на temp, то указатель name в main () остался бы установленным на адрес памяти, которая скоро может быть использована повторно. Вот почему внутри getname () необходимо вызывать new. На самом деле автоматические значения являются локальными по отношению к блоку, в котором они объявлены. Блок — это раздел кода, ограниченный фигурными скобками. До сих пор все наши блоки были целыми функциями. Но как будет показано в следующей главе, можно иметь блоки и внутри функций. Если вы объявите переменную внутри одного из таких блоков, она будет существовать только в то время, когда программа выполняет операторы, содержащиеся внутри этого блока. Автоматические переменные обычно хранятся в стеке. Это значит, что когда выполнение программы входит в блок кода, его переменные последовательно добавляются к стеку в памяти и затем освобождаются в обратном порядке, когда выполнение покидает данный блок. (Этот процесс называется UFO (last-in, first-out — "последним пришел — первым ушел".) Таким образом, по мере продвижения выполнения стек растет и уменьшается.*

***Статическое хранилище***

*Статическое хранилище — это хранилище, которое существует в течение всего времени выполнения программы. Доступны два способа для того, чтобы сделать переменные статическими. Один заключается в объявлении их вне функций. Другой предполагает использование при объявлении переменной ключевого слова static:*

*static double fee = 56.50;*

*Согласно правилам языка С стандарта K&R, вы можете инициализировать только статические массивы и структуры, в то время как C++ Release 2.0 (и более поздние), а также ANSI С позволяют также инициализировать автоматические массивы и структуры. Однако, как вы, возможно, обнаружите, в некоторых реализациях C++ до сих пор не реализована инициализация таких массивов и структур. В главе 9 статическое хранилище обсуждается более подробно. Главный момент, который вы должны запомнить сейчас относительно автоматического и статического хранилищ — то, что эти методы строго определяют время жизни переменных. Переменные могут либо существовать на протяжении всего выполнения программы (статические переменные), либо только в период выполнения функции или блока (автоматические переменные).*

***Динамическое хранилище***

*Операции new и delete предлагают более гибкий подход, нежели использование автоматических и статических переменных. Они управляют пулом памяти, который в C++ называется свободным хранилищем или кучей. Этот пул отделен от области памяти, используемой статическими и автоматическими переменными. Как было показано в листинге 4.22, операции new и delete позволяют выделять память в одной функции и освобождать в другой. Таким образом, время жизни данных при этом не привязывается жестко к времени жизни программы или функции. Совместное применение new и delete предоставляет возможность более тонко управлять использованием памяти, чем в случае обычных переменных. Однако*

*управление памятью становится более сложным. В стеке механизм автоматического добавления и удаления приводит к тому, что части памяти всегда являются смежными. Тем не менее, чередование операций new и delete может привести к появлению промежутков в свободном хранилище, усложняя отслеживание места, где будут распределяться новые запросы памяти.*