В программировании **сборка мусора** (устоявшийся термин,[1] с точки зрения русского языка правильнее «сбор мусора»,[2] англ. *garbagecollection*, *GC*) — одна из форм автоматического управления памятью. Специальный процесс, называемый ***сборщиком мусора*** (англ. *garbagecollector*), периодически освобождает память, удаляя объекты, которые уже не будут востребованы приложениями — то есть производит «сбор мусора».

Если бы память компьютера была бесконечной, можно было бы просто оставлять ненужные объекты в памяти. Сбор мусора — эмуляция такого бесконечного компьютера на конечной памяти.[3] Многие ограничения сборщиков мусора (нет гарантии, что финализатор выполнится; управляет только памятью, но не другими ресурсами) вытекают из этой метафоры.

Сборка мусора — технология, позволяющая, с одной стороны, упростить программирование, избавив программиста от необходимости вручную удалять объекты, созданные в динамической памяти, с другой — устранить ошибки, вызванные неправильным ручным управлением памятью.

В системе со сборкой мусора обязанность освобождения памяти от объектов, которые больше не используются, возлагается на среду исполнения программы. Программист лишь создаёт динамические объекты и пользуется ими, он может не заботиться об удалении объектов, поскольку это делает за него среда. Для осуществления сборки мусора в состав среды исполнения включается специальный программный модуль, называемый «сборщиком мусора». Этот модуль периодически запускается, определяет, какие из созданных в динамической памяти объектов более не используются, и освобождает занимаемую ими память.

Периодичность запуска сборщика мусора определяется особенностями системы. Сборщик может работать в фоновом режиме, запускаясь при неактивности программы (например, когда программа простаивает, ожидая ввода данных пользователем). Сборщик мусора запускается безусловно, прерывая на время своей работы выполнение программы, когда очередную операцию выделения памяти оказывается невозможно выполнить из-за того, что вся доступная память исчерпана. После освобождения памяти прерванная операция выделения памяти возобновляется и программа продолжает исполняться дальше. Если же оказывается, что освободить память невозможно, среда исполнения останавливает программу с сообщением об ошибке «Недостаточно памяти».

Достоинства и недостатки

По сравнению с ручным управлением памятью сборка мусора безопаснее, поскольку она предотвращает утечки памяти и возникновение висячих ссылок из-за несвоевременного удаления объектов. Другой положительный момент — упрощение самого процесса программирования. С другой стороны, наличие сборки мусора может вызывать у неопытного разработчика чувство ложной безопасности, базирующееся на представлении, что вопросам выделения и освобождения памяти вообще не надо уделять внимания, поскольку они решаются сборщиком мусора. Например, объект никогда не будет удалён, если на него остался хотя бы один необнулённый указатель в глобальной области видимости, и поиск такой псевдоутечки в языках со сборщиком мусора особенно сложен. Программист не может полностью игнорировать вопрос управления памятью при наличии сборщика мусора, хотя затраты ручного труда на управление памятью в этом случае всё-таки существенно меньше по сравнению с языками с полностью ручным управлением (без

сборщика и автодеструкторов). Зачастую критически важной является не только гарантия освобождения ресурса, но и гарантия того, что он освободится до вызова какой-то другой процедуры — например, открытые файлы, входы в критические секции. Отдавать управление этими ресурсами сборщику мусора нельзя, поэтому приходится убирать их вручную. Впрочем, в последнее время даже в языках со сборщиком мусора вводят возможность создавать классы с детерминированным вызовом специального метода-«деструктора» (Dispose) при выходе из зоны видимости.

Во многих случаях системы со сборкой мусора демонстрируют меньшую эффективность, как по скорости, так и по объёму используемой памяти (что неизбежно, так как сборщик мусора сам потребляет ресурсы и нуждается в некотором избытке свободной памяти для нормальной работы). Кроме того, в системах со сборкой мусора сложнее реализуются низкоуровневые алгоритмы, требующие прямого доступа к оперативной памяти компьютера, поскольку свободное использование указателей невозможно и прямой доступ к памяти требует наличия специальных интерфейсов, написанных на низкоуровневых языках. С другой стороны, в современных системах со сборкой мусора операция выделения памяти сведена к элементарному добавлению блока в конец кучи, причём куча время от времени уплотняется, уменьшая фрагментацию данных.