Сборщик мусора (СМ). GC.

Занимается дефрагментацией памяти (удаление пустых мест из кучи). У крупных объектов область памяти называется Large Object Heap (объекты > 85 000 байт).

На время работы СМ работа приложения приостанавливается.

Всего в кучи выделяют три поколения: 0,1,2. К 0-му относятся объекты, в которых еще не разу не выполнялась сборка мусора. К 1-му относятся объекты, которые однажды пережили сборку мусора. К 3-му относятся объекты, которые пережили больше одной сборки. Сначала идет удаление объектов 0-го уровня, потом 1-го, потом 2-го.

Поскольку объекты 0-го поколения являются более молодыми, они нередко находятся в адресном пространстве близко друг от друга. Их удаление проходит с наименьшими издержками.

System GC. Класс упр. сборщиком мусора.

AddMemoryPressure() – информирует среду CLR о выделении большого объема неуправляемой памяти, которую надо учесть при планировании сборки мусора.

В связке с этим методом используется метод RemoveMemoryPressure(), который указывает среде исполнения, что ранее зарезервированный участок памяти больше не нужен.

Collect() – приводит в действие механизм сборки мусора. GCollectMode – Default; Forced – немедленный запуск GC; Optimized – позволяет GC определить, является ли объект пригодным для зачистки.

GetGeneration(Object) – узнать поколение объекта.

GetTotalMemory() – возвращает объем памяти в байтах управляемый кучей.

WaitForPendingFinalizers() – приостанавливает работу текущего потока до освобождения всех объектов, для которых проводится сборка мусора.

Финализаторы – методы, вызываемые в момент прекращения работы с объектом.

Если объект имеет финализатор, то указатель на него сохраняется в очереди финализатора, и тот момент, когда происходит сборка мусора, данный объект копируется в еще одну таблицу и окончательно уничтожается лишь при следующем проходе сборщика мусора.

Способы работы с финализаторами: деструкторы, интерфейс IDisposoble.

Интерфейс IDisposoble вызывает метод Dispose(). В этом методе должно происходить удаление неуправляемых ресурсов.

**Советы по применению.**

+ Dispose позволяет в любой момент времени вызвать освобождение ресурсов.

- можно забыть вызвать dispose.

При вызове деструктора в Dispose передается false, чтобы избежать очистки управляемых ресурсов.

GC.SuppressFinalize(Object) - блокирует вызов finalize для объекта.

1) Деструктор следует реализовывать только в тех классах, где он реально необходим, так как вызов finalize сильно снижает производительность.

2) После вызова Dispose(), если он перегружен, необходимо блокировать вызов Finalize().

3) При создании производных классов от классов, реализующих IDisposible, следует вызывать метод Dispose() базового класса.

4) Если класс работает с неуправляемыми ресурсами, рекомендуется реализовывать в нем комбинированный подход к освобождению ресурсов.

**Указатели.**

Указатель позволяет получить доступ к определенной ячейке памяти, произвести манипуляции с данными внутри этой ячейки. Указатель – переменная, которая содержит в себе адрес другой переменной. Работая с указателями, за состоянием объектов приходится следить самому.

unsafe private void Func() { }

unsafe { }

unsafe struct MySt(){ }

\*- получение значения по указателю

&- получение адреса переменной

При работе в 32 битной системе, размер – 4 байта.

При работе в 64 битной системе, размер – 8 байт.

int\*, sbyte\*, short\*, ushort\*, uint\*, char\*, float\*, double\*, decimal\*, bool\*, void\*, enum\*, structure\* и на другие указатели.

unsafe {

int\* x;

int y = 10;

x = &y; // принимает адрес y

y = y+20;

\*x = 50;

}

**Указатели на массив. Оператор Stackalloc.**

С использованием ключевого слова Stackalloc можно выделить память под массив в стеке.

unsafe {

int\* array = stackalloc int [100]; // занимать будет 400 байт

int\* p = array;

\*(p++) = 1; // первая ячейка = 1. Передвинулись к следующей ячейке.

\*(--p); // переход к предыдущей ячейке

}

**Оператор fixed.**

Позволяет фиксировать на время работы указатель на объекты классов. Объект становится неперемещаемым в памяти.

fixed (int\* p = &person age) {

if(\*p < 30) {

\*p = 30;

}

}

Marshall класс