Указатели заводятся: gc\_ptr< Typename>

А память выделяется : gc\_new <Typename>(size\_t size)

Для корректой работы GC необходимо заменить ВСЕ указатели на объекты класса gc\_ptr.  
ex:(необходимые изменение кода для использования GC)  
1. int\* a = new int[5];  
 →  
 gc\_ptr<int\*> a = gc\_new<int\*>(5);

2. struct List {  
 ...  
 List\* next;  
 }  
 →  
 struct List {  
 ...  
 gc\_ptr<List\*> next;  
 };

Механизм поиска рутов:

1. Внутри gc\_new(ф-ия отведения памяти под gc\_ptr) выставляется флаг new\_active – декларирует место (куча или стек) создания объекта класса gc\_ptr.
2. При вызове конструктора gc\_ptr для стекового объекта в список all\_ptr(список всех существующих стековых объектов класса gc\_ptr) добавляется указатель на this. В случае, когда new\_active неактивен, т.е. создается стековый объект, внутри объекта класса gc\_ptr заводится указатель(me) на соответствующий ему элемент в списке all\_ptr, иначе me == 0.
3. В деструкторе gc\_ptr происходит удаление себя из all\_ptr, т.к. мы знаем me (он же указатель на себя в списке), удаление происходит за О(1).

Механизм создания объектов в куче:

1. Все происходит внутри функции gc\_new.
2. Объем выделяемой памяти первосходит память, занимаемую объектом, на размер слова(указатель на начало метаинформации) и размер метаинформации(4 машинных слова).
3. В начале блока памяти кладется метаинформация, затем указатель на начало метаинформации, за ней сам объект. Объект конструируется с использованием placement new.
4. Пользователю возвращается указатель на начало объекта.
5. Указатель на начало объекта кладется в вектор ptr\_in\_heap(нужно для того, чтобы отслеживать все объекты в куче).

Создание и хранение метаинформации:

1. В метаинформации хранится ссылка на обертку(box) и mark\_bit(для сборки мусора).
2. Обертка создается в единственном экземпляре для каждого типа.
3. В обертке хранятся смещения указателей вложенных объектов типа gc\_ptr от носительно указателя на объект (нужно для того, чтобы уметь ходить по всем достижимым объектам).
4. В функции gc\_new все смещения записываются в вектор offsets. Offsets получается следующим образом:

А) внутри конструктора gc\_ptr, при активном флаге new\_active, указатель на this кладется в глобальный список ptr\_in\_heap(все gc\_ptr кучи).

Б) смещения получаются взятием разности между значением в глобальном массиве и указателем на начало объекта.

Сборка мусора:

1. Функция сборки мусора: mark\_and\_sweep – вызывается внутри gc\_new каждые К(некоторая константа) байт выделений памяти.
2. Mark\_and\_Sweep делает следующее:

А) С помощью функции go обходит и помечает(устанавливает mbit = 1) все достижимые из рутов объекты.

Б) Удаляет отовсюду недостижимые объекты.

В) Обновляет ptr\_in\_heap

Г) У всех выживших объектов устанавливает mbit = 0.