پیاده سازی کلاس stack (فقط int ) :

در هنگام پیاده سازی کلاس باید به فکر آزاد کردن حافظه گرفته شده از heap باشید

این کار معمولا در distructor انجام می شود

اسم distructor هر کلاس یک "~" و سپس اسم کلاس است

دلایل استفاده از حافظه heap :

کنترل حافظه دست خودمان است و هر موقع که بخواهیم آبجکنی را delete می کنیم

ممکن است در موقع کامپایل حجم آبجکت را ندانیم پس در runtime از heap استفاده می کنیم

نکته ای که باید در مورد by value passing آبجکت های که به طور پویا تخصیص حافظه دارند این است که ,اگر copy semantics به طور صحیح برای آن تعریف نشده باشد می تواند باعث runtime Error شود به این دلیل که هر دو آبجت در پوینتر داخل خود به یک حافظه new شده اشاره می کنند و در distructor خود آن را delete می کنند که این عمل برای اولین آبجکت با موفقیت انجام می شود ولی برای دومین آبجکت با خطای حافظه پاک شده مواجه می شود (shallow copy )

راه حل این مشکل این است که در copy semantics علاوه بر لایه اول (pointer ها) لایه دوم ( حافظه اشغال شده در heap ) رو هم کپی کنیم (deep copy)

Copy constructor هم مانند constructor آبجکت میسازد اما از روی آبجکتی دیگر

و همیشه آبجکت دیگر را به صورت const و با reference به عنوان آرگومان به آن می دهیم

علاوه بر copy constructo , operator= هم باید برای جلوگیری از مشکل مدیریت حافظه overload شود

و نکته آخر برای جلوگیری از مشکل مدیریت حافظه این است که باید جلوی self assignment گرفته شود

(if(this == &other) return ; )

چه در copy constructor و چه در move constructor

وقتی کلاسی خودش new می کند حتما باید copy constructor , copy assignment operator و distructor برای آن تعریف شود

مستحب موکد آن است که constructor بدون آرگومان هم برایش تعریف شود (به علت توانایی ذخیره سازی در وکتور و container های دیگر)

به طور کلی وقتی کلاسی تعریف می کنیم باید مسئولیت پاک عناصر داخلش را به عهده بگیریم

C++ برای سرعت garbage collection ندارد