Destructor

- when any object of class type is created, the compiler generates a call to a constructor of that type
- when any object's lifetime ends, the compiler generates a call to the destructor of that type

Copy constructor

The destructor is responsible for freeing resources tot avoid ***leaks***. The copy constructor is responsible for duplicating resources to avoid ***double frees***

The Rule of Five

- If your class directly manages some kind of resource (such as a new'ed pointer), then you almost certainly need to hand-write three special member functions:
-
    - A ***destructor*** to free the resource
    - A ***copy constructor*** to copy the resource
    - A ***move constructor*** to transfer ownership of the resource
    - A ***copy assignment operator*** to free the left-hand resource and copy the right-hand one
    - A ***move assignment operator*** to free the left-hand resource and transfer ownership of the right-hand one
- Use the copy-and-swap idiom to implement assignment.

RAII and exception safety

"Resource Acquisition Is Initialization."Resource Freeing Is Destruction
Destructors help us write code that is robust against exceptions

- C++ supports try/catch and throw
- When an exception is thrown, the runtime looks "up the call stack" until it finds a suitable catch handler for the type of the exception being thrown. Assuming it finds one...
- The runtime performs ***stack unwinding***. For every local scope between the throw and the catch handler, the runtime invokes the destructors of all local variables in that scope.
- To avoid leaks, place all your ***cleanup*** code in ***destructors***.

The Rule of Zero

- If your class does not directly manage any resource, but merely uses library components such as vector and string, then you should strive to write ***no*** special member functions.

  Default them all!

    - Let the compiler implicitly generate a defaulted destructor
    - Let the compiler generate the copy constructor

- Let the compiler generate the copy assignment operator
- Prefer Rule of Zero when possible

類別	正負號	實際指數	有偏移指數	指數域	尾數域	數值
零	0	-127	0	0000 0000	000 0000 0000 0000 0000 0000	0.0
負零	1	-127	0	0000 0000	000 0000 0000 0000 0000 0000	−0.0
1	0	0	127	0111 1111	000 0000 0000 0000 0000 0000	1.0
-1	1	0	127	0111 1111	000 0000 0000 0000 0000 0000	−1.0
最小的非正規數	*	-126	0	0000 0000	000 0000 0000 0000 0000 0001	$\pm 2^{-23} \times 2^{-126} = \pm 2^{-149} \approx \pm 1.4 \times 10^{-45}$
中間大小的非正規數	*	-126	0	0000 0000	100 0000 0000 0000 0000 0000	$\pm 2^{-1} \times 2^{-126} = \pm 2^{-127} \approx \pm 5.88 \times 10^{-39}$
最大的非正規數	*	-126	0	0000 0000	111 1111 1111 1111 1111 1111	$\pm(1-2^{-23}) \times 2^{-126} \approx \pm 1.18 \times 10^{-38}$
最小的正規數	*	-126	1	0000 0001	000 0000 0000 0000 0000 0000	$\pm 2^{-126} \approx \pm 1.18 \times 10^{-38}$
最大的正規數	*	127	254	1111 1110	111 1111 1111 1111 1111 1111	$\pm(2-2^{-23}) \times 2^{127} \approx \pm 3.4 \times 10^{38}$
正無窮	0	128	255	1111 1111	000 0000 0000 0000 0000 0000	+∞
負無窮	1	128	255	1111 1111	000 0000 0000 0000 0000 0000	−∞
NaN	*	128	255	1111 1111	非全0	NaN
* 符號位可以為0或1 .						