|  |  |
| --- | --- |
|  | Put yourself in the compiler's position: when you forward declare a type, all the compiler knows is that this type exists; it knows nothing about its size, members, or methods. This is why it's called an*incomplete type*. Therefore, you cannot use the type to declare a member, or a base class, since the compiler would need to know the layout of the type.  Assuming the following forward declaration.  class X;  Here's what you can and cannot do.  **What you can do with an incomplete type:**   * Declare a member to be a pointer or a reference to the incomplete type: * class Foo { * X \*pt; * X &pt;   };   * *Declare* functions or methods which accept/return incomplete types: * void f1(X);   X f2();   * *Define* functions or methods which accept/return pointers/references to the incomplete type (but without using its members): * void f3(X\*, X&) {} * X& f4() {}   X\* f5() {}  **What you cannot do with an incomplete type:**   * Use it as a base class   class Foo : X {} // compiler error!   * Use it to declare a member: * class Foo { * X m; // compiler error!   };   * *Define* functions or methods using this type * void f1(X x) {} // compiler error!   X f2() {} // compiler error!   * Use its methods or fields, in fact trying to dereference a variable with incomplete type * class Foo { * X \*m; * void method() * { * m->someMethod(); // compiler error! * int i = m->someField; // compiler error! * }   };  When it comes to templates, there is no absolute rule: whether you can use an incomplete type as a template parameter is dependent on the way the type is used in the template.  For instance, std::vector<T> requires its parameter to be a complete type, while boost::container::vector<T> does not. Sometimes, a complete type is required only if you use certain member functions; [this is the case for std::unique\_ptr<T>](http://stackoverflow.com/questions/6012157/is-stdunique-ptrt-required-to-know-the-full-definition-of-t), for example.  A well-documented template should indicate in its documentation all the requirements of its parameters, including whether they need to be complete types or not. |