23.Перевантаження унарних операцій, зокрема \* («розіменування») та -> («розумний вказівник»).

Перегрузка этих операторов может быть оправдана для классов умных указателей. Этот оператор обязательно определяется как функция класса, причём на него накладываются некоторые ограничения: он должен возвращать либо объект (или ссылку), либо указатель, позволяющий обратиться к объекту.

Перегрузка оператора разыменования ( \* ) работает так же, как и любая другая перегрузка оператора. Если вы хотите иметь возможность изменять разыменованное значение, вам нужно вернуть неконстантную ссылку. Таким образом,  \*sp = value фактически изменит значение, на которое указывает  sp.pData , а не временное значение, сгенерированное компилятором.

Перегрузка оператора разыменования структуры (->)является частным случаем перегрузки оператора. Оператор фактически вызывается в цикле до тех пор, пока не будет возвращен реальный указатель, а затем этот реальный указатель разыменовывается. Предположим, у вас были следующие классы:

struct A {

int foo, bar;

};

struct B {

A a;

A \*operator->() { return &a; }

};

struct C {

B b;

B operator->() { return b; }

};

struct D {

C c;

C operator->() { return c; }

};

Если бы у вас был объект  d типа  D, вызов  d->bar сначала вызвал бы  D::operator->() , затем  C::operator->() , а затем  B::operator->() , который , наконец, возвращает реальный указатель на структуру  A, и его член  bar разыменовывается обычным способом. Обратите внимание, что в следующем:

struct E1 {

int foo, bar;

E1 operator->() { return \*this; }

};

Вызов  e->bar, где  e имеет тип  E1, приводит к бесконечному циклу. Если бы вы действительно хотели разыменовать  e.bar, вам нужно было бы сделать это:

struct E2 {

int foo, bar;

E2 \*operator->() { return this; }

};

Подводя итог:

1. При перегрузке оператора разыменования тип должен быть  T& , поскольку это необходимо для изменения значения, на которое указывает  pData .
2. При перегрузке разыменования структуры тип должен быть  T\* , потому что этот оператор является частным случаем, и именно так он работает.