The code to convert a formula-based linear list to a chain is

```
template<class T>
void ArrayToChain(const LinearList<T>& L, Chain<T>& C)
{// Convert L to a chain C.
 // Do not catch exceptions that might be thrown.
   T x;
   C.Erase(); // empty the chain
   for (int i = L.Length(); i > 0; i--) {
      // copy i'th element of L
      L.Find(i,x);
      C.Insert(0,x);
      }
}
```

`Erase` is a member function of the class `Chain` and is defined in the file `echain.h`. The complexity of `Find` and `Insert(0,x)` is $\Theta(1)$. So the actual conversion takes $\Theta($`length`$)$ time where `length` is the length of the list. In addition, we need to erase the initial chain `C`. This takes time linear in the length of this initial chain. We could have set `C` to the empty chain by using the function `Zero`. While this has complexity $\Theta(1)$, it does not delete the nodes that were initially on the chain and this space is therefore lost to the program.

The test code and output are in the files `tochain.cpp` and `tochain.out`.