chúng sẽ được kiểm tra tự động. Đi kèm với Spec# là một bộ kiểm tra chương trình tự động, có tên là Boogie [18]. Nó cho phép kiểm tra các đặc tả theo kỹ thuật phân tích tĩnh [19].

Hình 3 minh họa về đặc tả của Spec# cho lớp *ArrayList*. Trong đó, phương thức *Insert* được đặc tả với các tiền điều kiện (***requires***), hậu điều kiện (***ensures***). Đặc biệt, Spec# còn cho phép các đặc tả có thể thừa kế lẫn nhau. Điều này giúp cho các đặc tả của Spec# rõ ràng, đáng tin cậy và dễ đọc hơn so với các ngôn ngữ đặc tả khác. Chi tiết về cách thức sử dụng Spec# chúng ta có thể tham khảo trong [17].

```
class ArrayList {
  public void Insert
        (int index, object
value)
  requires 0 <= index &&
                 index <= Count
;
  requires !IsReadOnly &&

!IsFixedSize;
  ensures Count == old(Count)+1;
  ensures value == this[index];
  ensures foral{int i in
0:index;

old(this[i])==this[i]};
  ensures foral{int i in
              index:old(Count);
       old(this[i])==this[i
+1]};
  { /*...*/ }
  //… …
}
```

Hình 3: Minh họa về đặc tả Spec#

2.2.3 Frama-C và ngôn ngữ ACSL

Frama-C [20] là một nền tảng dành riêng cho việc phân tích tĩnh mã nguồn chương trình được viết bằng ngôn ngữ C. Frama-C tập hợp một số kỹ thuật phân tích tĩnh [19] vào trong một khung thức duy nhất. Để được phân tích bởi Frama-C, các chương trình C cần được đặc tả bởi ngôn ngữ ACSL (ANSI/ISO C Specification Language) [21]. Đây là ngôn ngữ đặc tả hình thức cho phép chúng ta đặc tả các thuộc tính của một chương trình C. ACSL cũng được "lấy cảm hứng" từ ngôn ngữ JML.

Khái niệm quan trọng nhất trong ACSL đó là *thỏa thuận* hàm. Một thỏa thuận hàm cho một hàm *f* trong ngôn ngữ C là một tập các yêu cầu trên các đối số của *f* và một tập các thuộc tính cần được đảm bảo sau khi kết thúc việc thực hiện *f*. Chúng ta hãy xem xét ví dụ về hàm *max* (Hình 4). Hàm này không cần tiền điều kiện, không cần bất kỳ sự ràng buộc nào trên các đối số truyền vào. Hậu điều kiện của chúng mô tả rằng kết quả trả về của hàm *max* luôn lớn hơn hoặc bằng giá trị của *x*, *y* và sẽ bằng một trong hai tham số *x* hoặc *y* truyền vào.

```
/*@ ensures \result >= x &&
                      \result >=
y;
    ensures \result == x ||
                   \result == y;
*/
int max (int x, int y)
{ return (x > y) ? x : y; }
```

Hình 4: Minh họa về đặc tả ACSL

Ngoài tiền và hậu điều kiện, ACSL còn hỗ trợ đặc tả bất biến cho các đối tượng và vòng lặp. Tuy nhiên, ACSL lại không có cơ chế xử lý ngoại lệ như JML. Chi tiết về ACSL chúng ta có thể tham khảo trong [21].

2.2.4 Jahob

Jahob [22] là một hệ thống kiểm tra dành cho các chương trình viết bằng ngôn ngữ Java (tập con của ngôn ngữ Java). Sử dụng Jahob, các nhà phát triển có thể chứng minh theo phương pháp phân tích tĩnh rằng các phương thức hiện thực có phù hợp với các "thỏa thuận" đặc tả của chúng hay không. Jahob còn cho phép đặc tả các bất biến về các cấu trúc dữ liệu quan trọng cũng như các ràng buộc thiết kế. Ý tưởng cơ bản của Jahob là mô hình hóa các trạng thái của chương trình và các cấu trúc dữ liệu mà nó sử dụng thành các tập trừu tượng các đối tượng và các mối quan hệ giữa các đối tượng. Jahob sẽ hỗ trợ phát biểu các ràng buộc trên tập các đối tượng này. Chúng ta xét ví dụ về đặc tả của Jahob cho lớp *List* như được mô tả trong Hình 5.

Trong ví dụ trên, đặc tả của lớp *List* sử dụng biến đặc tả *content* để chỉ tập các đối tượng thể hiện trong danh sách. Tập này không tồn tại khi chương trình chạy – nó đơn giản là một sự trừu tượng. Chương trình Jahob chỉ sử dụng nó cho mục đích đặc tả và bộ kiểm tra Jahob sử dụng nó để kiểm tra chương trình. Chi tiết về cấu trúc và cú pháp của ngôn ngữ đặc tả cho hệ thống Jahob chúng ta có thể tham khảo trong [22].