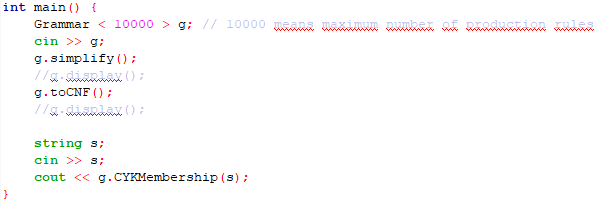
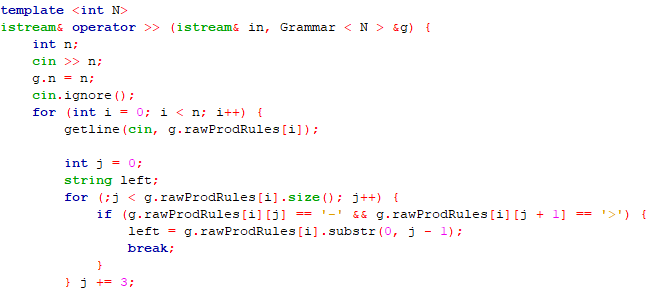
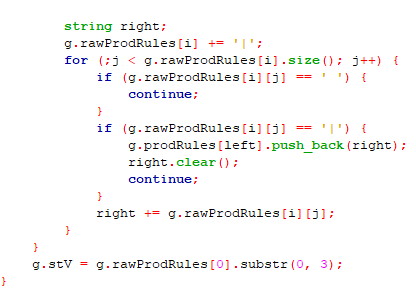
**Grammar .1**

روند کلی

روند کلی برنامه به این صورت است که ابتدا شی گرامر از ورودی خوانده می‌شود سپس ساده‌سازی می‌شود، به فرم چامسکی تبدیل می‌شود و در نهایت الگوریتم CYK با توجه به رشته‌ی ورودی روی آن اجرا می‌شود.

ورودی گرفتن گرامر

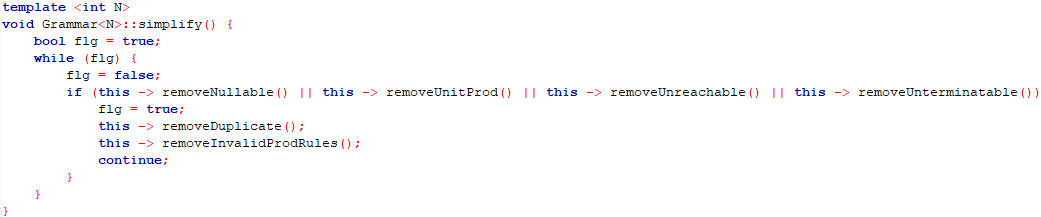
ابتدا همه‌ی خطوط گرامر از ورودی خوانده می‌شوند و سمت چپ production rule از سمت راست آن جدا می‌شود.



در نهایت به کمک داده‌ساختار map سمت چپ هر production rule را به سمت راست آن متناظر می‌کنیم و در شی ذخیره می‌کنیم.

متغیر شروع (Start Variable) را سمت چپ اولین production rule در نظر گرفته و به عنوان یکی از ویژگی‌های شی گرامر ذخیره می‌کنیم.

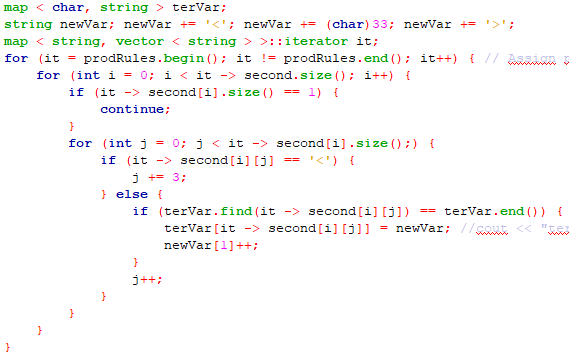
ساده‌سازی گرامر

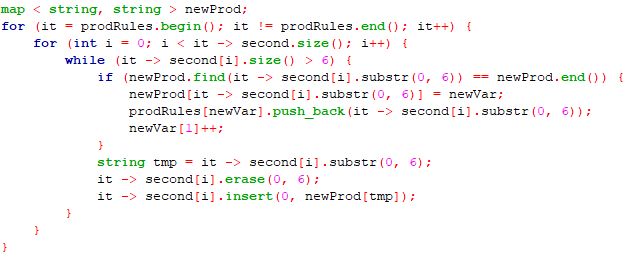


تا زمانی که یک Nullable یا Unit یا Useless بتوانیم حذف کنیم اینکار را می‌کنیم و ترتیب انجام این‌کار هم همانطور که از قبل می‌دانستیم Nullable،Unit ،Useless انجام می‌دهیم. (که البته برای Useless دو متود جداگانه برای Unreachableها و آن‌هایی که به terminal ختم نمی‌شوند نوشته شده است) هر کدام از متودهای گفته شده یک مقدار boolean خروجی می‌دهند که نشان می‌دهد آیا آن متود توانسته است چیزی را ساده کند یا نه که برای فهمیدن اینکه کی ساده‌سازی به پایان می‌رسد از این خروجی‌ها استفاده می‌کنیم.

بعد از هر عملیات حذف (ساده‌سازی) یکبار بررسی می‌کنیم اگر بین production rule ها تکراری داشتیم آن را حذف می‌کنیم (removeDuplicate) و اگر در production ruleای یک متغیر نامعتبر وجود داشت (متغیری که برایش production ruleای نباشد) آن را هم حذف می‌کنیم (removeInvalidProdRules)

تبدیل به فرم نرمال چامسکی

ابتدا هر terminalای که در بین production ruleها پیدا می‌کنیم را با variable جایگزینی که برایش می‌سازیم جایگزین می‌کنیم. variableهای جایگزین را با استفاده از کاراکترهای خاص ! و $ و... می‌سازیم تا variable تکراری تولید نشود که برای اینکار از عدد اسکی ۳۳ به بعد را استفاده کردیم.

سپس هر production rule را تا جایی که متغیرهای سمت راست آن به دوتا برسد کاهش می‌دهیم و به ازای هر کاهش یا متغیر جدیدی برای آن می‌سازیم یا اگر قبلا production ruleای برای آن وجود داشته از همان production rule قدیمی استفاده می‌کنیم.

الگوریتم CYK Membership