دانشکده مهندسی کامپیوتر

پروژه پایانی نظریه زبان‌ها و ماشین‌ها

حسین بابازاده - محمدجواد جلیلوند

بهار ۱۴۰۴

تاریخ انتشار: ۱ اردیبهشت ۱۴۰۴
تاریخ تحویل: ۵ خرداد ۱۴۰۴

فهرست مطالب

اول مقدمه ۱

دوم مروری بر دانش نظری مربوط به پروژه ۲

۱ پایپ‌لاین کامپایلر و نقش درخت تجزیه ۲

۲ گرامرهای قابل تجزیه و انتخاب گرامر LL1 ۲

۳ جدول تجزیه ۳

۴ ساخت DPDA از روی جدول تجزیه ۳

سوم مراحل پیاده‌سازی ۴

۱ ورودی گرفتن و ذخیره گرامر ۴

۲ نگهداری و اجرای پردازش با DPDA ۴

۳ تبدیل گرامر LL1 به DPDA ۴

۴ تولید و نمایش درخت تجزیه ۴

۵ تغییر نام نماد در متن ۴

بخش اول

مقدمه

سلام به تمام نظریه‌دان‌های زبان‌ها و ماشین‌ها

چرا پروژه؟

نظریه زبان‌ها و ماشین‌ها همان‌طور که از اسمش مشخصه خیلی تئوری هست حتی تمرین‌ها هم فقط جنبه تئوری دارن و شما مثل هر مهندس کامپیوتر دیگه ای دوست دارید دست به کد بشوید و ببینید این درس کجای این دنیای کامپیوتر قرار می‌گیرد.

پروژه چیه؟

اپلیکیشن‌های زیادی حول این درس وجود داره. یکی از این موارد کامپایلر هست. در واقع ساختار زبان‌های برنامه نویسی و فرایند تبدیل شدن کد به زبان ماشین ارتباط تنگاتنگی با نظریه زبان‌ها و ماشین‌ها داره. البته اپلیکیشن‌های پیشرفته تری مثل Learning Automaton یا Quantum finite automaton هم هستن.

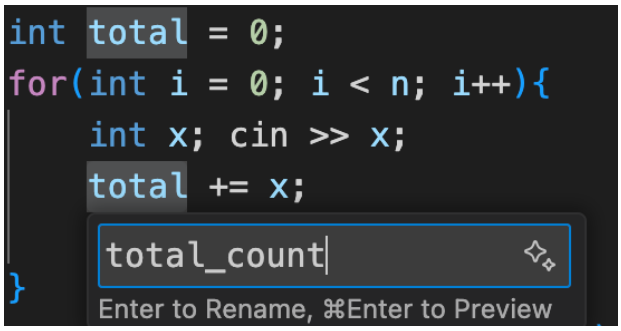
این درس پیش نیاز اصلی درس کامپایلره که ترم‌های بعد قرار هست پاس کنید (شاید هم همین الان در حال پاس کردن این درس باشید D:). خلاصه که یکی از کاربردهای مهمش در کامپایلر هست.

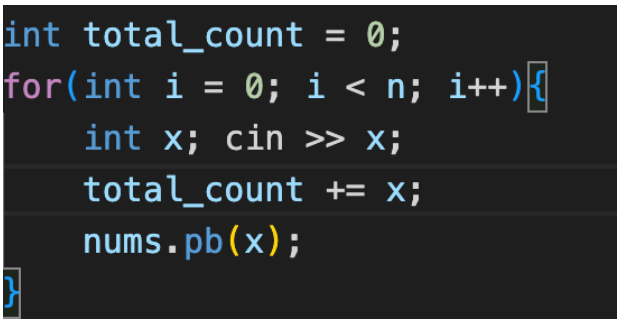
ما قصد داریم تا یه کامپایلر کوچولو بنویسیم.

حالا چجوری؟

ما برای سادگی پروژه رو فازبندی کردیم و مراحل پیاده سازی رو به ترتیب و با جزئیات توضیح دادیم. سعی بر این بوده مطالب خارج از درس کلاس کم باشه ولی صفر نیست. به همین خاطر هر جا نیاز بود در ادامه توضیح داده شده یا رفرنس داده شده.

در نهایت هم اکشن Rename Symbol مانند تصویر زیر را سعی می‌کنیم پیاده سازی کنیم.





بخش دوم

مروری بر دانش نظری مربوط به پروژه

در این بخش، مفاهیم پایه‌ای مرتبط با پروژه شامل ساختار کامپایلر، گرامرهای قابل تجزیه، گرامر LL1، جدول تجزیه و تبدیل آن به DPDA شرح داده می‌شود.

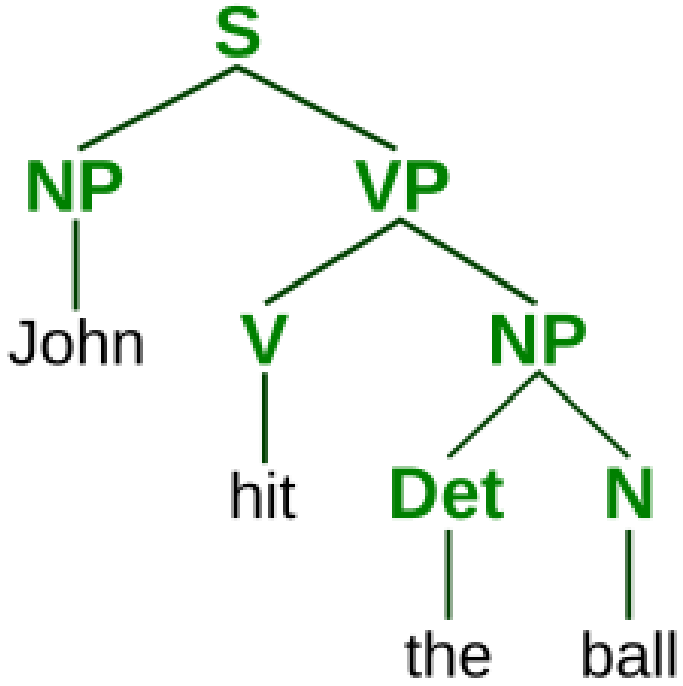
۱ پایپ‌لاین کامپایلر و نقش درخت تجزیه

کامپایلر به‌عنوان ابزاری برای ترجمه کد منبع به کد ماشین، دارای مراحل مختلفی است.

ابتدا تجزیه لغوی (Lexical Analysis) است که طی آن کد ورودی به توکن‌ها (قطعات کوچک‌تر، مانند کلمات کلیدی، نام چیزها، اعداد، رشته‌ها و...) شکسته می‌شود. سپس تجزیه نحوی (Syntax Analysis) است که با استفاده از قواعد گرامری، درخت تجزیه (Parse Tree) ساخته می‌شود.

درخت تجزیه نمایی سلسله‌مراتبی از ساختار متن ورودی بر اساس گرامر است. در ادامه، کامپایلر برای کارهایی مانند تحلیل معنایی، بررسی نوع داده‌ها، بررسی استفاده صحیح از متغیرها و در نهایت تولید کد از این درخت در کنار ساختمان‌های داده‌ای دیگر استفاده می‌شود.

یکی از این عملیات در پروژه شما تغییر نام نمادها در متن تجزیه‌شده است.



نمونه درخت تجزیه یک جمله انگلیسی

۲ گرامرهای قابل تجزیه و انتخاب گرامر LL1

همه گرامرهای مستقل از متن (CFG) قابلیت تجزیه دارند، اما برخی از آنها پیچیدگی‌هایی مانند ابهام یا بازگشت چپ دارند که مانع استفاده مستقیم برای تجزیه می‌شوند. گرامر LL(k) یک فرم از گرامر (شبیه فرم نرمال گریباخ)

است که تضمین می‌کند برای هر عنصر غیرپایانی با دانستن k عنصر پایانی بعدی حداکثر یک قانون تولید وجود دارد. برخلاف فرم نرمال چامسکی و گریباخ، همه گرامرها لزوماً به یک فرم LL(k) تبدیل نمی‌شوند.

برای تجریه متن و ساخت کامپایلر، معمولا ابتدا گرامرها به فرم LL(k) تبدیل می‌شوند. در این پروژه برای سادگی یک گرامر LL1 برای یک زبان ساده انتخاب شده است که امکان تجزیه قطعی بدون بازگشت به عقب را فراهم می‌کند.

۳ جدول تجزیه

گرامر LL1 ساده‌تر است، چرا که تصمیم‌گیری برای انتخاب قاعده تولید، تنها با نگاه به تنها یک نماد ورودی انجام می‌شود. این خاصیت، حالت‌های انتخاب را محدود می‌کند و می‌توانیم همه آنها را در یک جدول نمایش دهیم. جدول تجزیه LL1 ابزار اصلی برای این نوع گرامرهاست که شامل غیرپایانه‌ها در سطرها، پایانه‌ها در ستون‌ها، و قواعد تولید مربوطه در سطر و ستون تقاطع آنها است.

مثلا گرامر زیر را درنظر بگیرید:

- $E \to TE'$
- $E' \to +TE' \mid \epsilon$
- $T \to FT'$
- $T' \to *FT' \mid \epsilon$
- $F \to \text{id} \mid (E)$

برای این گرامر، جدول زیر تولید می‌شود:

Non-Terminal	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to \text{id}$			$F \to (E)$		

۴ ساخت DPDA از روی جدول تجزیه

هر گرامر مستقل از متن قابلیت تبدیل به اتوماتای پشته‌ای (PDA) را دارد. اما این اتوماتا لزوما قطعی نیست که پیاده‌سازی و استفاده از آن را دشوار و هزینه‌بر می‌کند. با استفاده از جدول تجزیه LL1، می‌توان یک DPDA (اتوماتای پشته‌ای قطعی) ساخت که قواعد تولید را بر اساس نماد پیش‌بینی اعمال کند. این DPDA امکان پردازش با پیچیدگی زمانی قابل قبول را فراهم کرده و برای اجرای عملیات مختلف روی متن مناسب است.

بخش سوم

مراحل پیاده‌سازی

پروژه درس شامل چندین مرحله پیاده‌سازی برای دستیابی به عملکرد مورد نظر است. توجه کنید که هر مرحله لزوما به همه مراحل قبلی نیاز ندارد.

۱ ورودی گرفتن و ذخیره گرامر

کلاسی طراحی کنید که یک گرامر مستقل از متن را نگهداری کند. می‌توانید این موضوع را درنظر بگیرید که گرامر ورودی LL1 است. همچنین باید گرامر را از فایل ورودی خوانده و آن را در یک نمونه از این کلاس ذخیره کنید.

۲ نگهداری و اجرای پردازش با DPDA

کلاسی طراحی کنید که یک اتوماتای پشته‌ای قطعی (DPDA) را نگهداری کند. همچنین کدی بنویسید که با دریافت یک DPDA و یک رشته ورودی، آن رشته را به ماشین ورودی داده و نتیجه پردازش (پذیرفته بودن یا نبودن رشته) را مشخص کند.

۳ تبدیل گرامر LL1 به DPDA

کدی بنویسید که یک گرامر LL1 را به یک DPDA تبدیل کند. ورودی این قسمت یک نمونه گرامر LL1 و خروجی آن یک نمونه DPDA (از کلاس‌های مراحل قبل) است. برای این کار باید ابتدا از روی گرامر، جدول تجزیه بسازید که نشان می‌دهد به ازای هر ترکیب یک حرف از الفبای ورودی (non-terminal) و یک حرف از الفبای استک (terminal)، در ادامه متن چه چیزی باید ببینیم. سپس با اضافه کردن قوانین متناظر هر خانه از جدول، DPDA را کامل کنید.

۴ تولید و نمایش درخت تجزیه

در این مرحله کدی بنویسید که درخت تجزیه را بر اساس پردازش یک متن داده شده بر روی یک DPDA تولید کند. درخت تجزیه ساختار متن ورودی را بر اساس قوانین تعریف شده نشان می‌دهد. این مرحله برای تحلیل و درک ساختار متن، بررسی وجود خطا یا تغییر دادن آن ضروری است.

۵ تغییر نام نماد در متن

در نهایت باید کدی بنویسید که همه تکرارهای یک نماد را در یک متن تجزیه‌شده تغییر نام می‌دهد. ورودی‌ها برای این فرآیند درخت تجزیه، یک گره خاص در درخت و نام جدید برای نماد هستند. خروجی متن تغییر یافته با همه موارد نماد با نام جدید مشخص شده خواهد بود.

توجه کنید که فقط جاهایی که به همان متغیر ارجاع می‌دهند تغییر کنند. اگر متغیری با همان نام در اسکوپ دیگری وجود داشته باشد، باید بدون تغییر بماند. برای پیدا کردن دقیق موارد مربوطه، ایده‌های مختلفی با استفاده از درخت تجزیه قابل پیاده‌سازی است. خلاقیت شما در این مرحله بخش مهمی از پروژه است.

با آرزوی موفقیت و کامیابی