تعریف فاز اول پروژه کامپایلر
نیمسال اول 87-86 – دکتر ابوالحسنی
موعد تحویل: 86/8/29

پروژه‌ی این درس، یک کامپایلر کامل از زبان MiniJava به زبان ماشین است. طی انجام آن شما با مراحل طراحی و پیاده‌سازی یک کامپایلر تا حدی آشنا خواهیدشد. این پروژه در 3 یا 4 فاز تعریف خواهدشد. برنامه را باید به زبان Java بنویسید. حجم کار زیاد است و شما باید در گروه‌های یک یا دو نفره آن‌را انجام دهید؛ گروه‌ها پس از انجام فاز اول قابل تغییر نخواهند بود. در هر فاز به همراه خود برنامه، باید documentation نیز ارسال کنید، شامل توضیحاتی جامع درباره‌ی این که چه کار کرده‌اید (لازم نیست خیلی مفصل باشد، حدود 2 صفحه)، و ایده‌هایی که به ذهنتان رسیده و اعمال کرده‌اید.

در فاز اول باید برنامه‌ای بنویسید که اعمال lexical analysis و parsing و ساخت abstract syntax tree را انجام دهد. برای دو عمل اول برنامه‌هایی وجود دارند که با گرفتن ساختار زبان، قسمت سنگین کار را انجام می‌دهند؛ ولی برای عمل سوم شما باید حجم اصلی کد را بنویسید. ترتیب زیر برای انجام این کار پیشنهاد می‌شود:

1. با زبان MiniJava آشنا شوید! در [1] می‌توانید گرامر این زبان را ببینید. این زبان بسیار شبیه Java است ولی از آن ساده‌تر است. وجه تمایزهای آن با Java را یاد بگیرید. به‌علاوه دقت کنید که گرامر داده‌شده مبهم (ambiguous) است. با توجه به اولویت اپراتورها، که همان اولویت معمولی زبان Java است، گرامر غیرمبهمی برای آن به دست‌آورید.
2. با نرم‌افزار JLex آشنا شوید! در [2] می‌توانید اطلاعات مربوطه را پیدا کنید و برنامه را دریافت کنید. این برنامه شبیه Lex یک lexical analyzer generator است ولی خروجی آن یک برنامه‌ی Java است. طرز کار با آن را یاد بگیرید، سپس مشخصات MiniJava را به آن بدهید و یک lexical analyzer برای MiniJava بسازید.
3. با نرم‌افزار CUP آشنا شوید! در [3] می‌توانید اطلاعات مربوطه را پیدا کنید و برنامه را دریافت کنید. این برنامه شبیه Yacc یک parser generator است و ورودی/خروجی آن به Java است. CUP یک LALR parser تولید می‌کند. طرز کار با آن را یاد بگیرید. سپس به کمک مشخصات MiniJava (گرامر غیرمبهمی که ساخته‌اید) و خروجی JLex، یک parser برای MiniJava بسازید.
4. ساختار abstract syntax tree (AST) خود را معلوم کنید، یعنی تعیین کنید که nodeهای AST شما از چه نوعی باید باشد، و چگونه باید با هم ارتباط داشته‌باشند. برای تعریف AST به بخش 2.8.2 کتاب مراجعه

کنید. مباحث کامل‌تر در این مورد در فصل 5 آمده‌است. طراحی شما باید به گونه‌ای باشد که، اولاً در هنگام parse بتوان این درخت را بدون دردسر زیاد ساخت، ثانیاً بتوان (بعداً) راحت این درخت را طی کرد و عملیاتی مثل type-checking و تولید کد میانی را روی آن انجام داد. به خصوص به nodeهایی توجه کنید که تعداد فرزندانشان متغیر است، مثلاً function callها. شاید [4] بتواند به شما برای یک طراحی خوب ایده بدهد.

5. برنامه‌ی تولیدی توسط CUP، فقط parse می‌کند ولی parse tree به شما نمی‌دهد. در مقابل این امکان را به شما می‌دهد که در هنگام parse کردن، کدهای دلخواهی را اجرا کنید. با توجه به طراحی خود در مرحله‌ی قبل، کدهای مربوطه را بنویسید تا به برنامه‌ای برسید که عمل ساخت AST را نیز انجام می‌دهد. این قسمت خسته‌کننده‌ترین و طولانی‌ترین قسمت کار است.

نکات:

- ✓ این پروژه وقت‌گیر است و مهلت کار به همین دلیل زیاد درنظر گرفته شده. هر چه سریع‌تر کار را شروع کنید تا وقت کم نیاورید. مهلت تعیین شده تمدید نخواهدشد، و دیرکرد جریمه خواهدداشت. پس از یک هفته پس از مهلت تعیین شده، نمره‌ای نخواهیدگرفت. به‌علاوه، در فازهای بعدی از خروجی این فاز خود استفاده خواهیدکرد پس سعی کنید حتماً تا زمان تعیین شده برنامه را تکمیل کنید.
- ✓ پروژه در گروه‌های حداکثر دو نفره باید انجام شود و تا فاز اول همگروهی‌های خود را باید شناخته باشید، هر فردی در فاز اول همگروهی نداشته باشد باید تنهایی پروژه را تا فاز نهایی تحویل دهد.
- ✓ نمره‌های همگروهی‌ها یکسان نخواهد بود و بر اساس میزان مشارکت در پروژه خواهد بود.
- ✓ برنامه نهایی خود + فایل ورودی که به JLex داده‌اید + فایل ورودی که به CUP داده‌اید + documentation را به صورت proj1-sn1-sn2.zip درآورده (اگر تنها هستید به صورت proj1-sn.zip) و به ce.compiler.course@gmail.com ارسال کنید. Subject نیز باید مشابه اسم فایل باشد، بدون پسوند. پروژه تحویل حضوری نیز خواهد داشت، که اعلام خواهدشد.
- ✓ اگر سوالی داشتید در Discussion Area مطرح کنید یا به abbas.mehrabian@gmail.com بفرستید.
- ✓ پروژه را باید خودتان انجام دهید، با تقلب به شدت برخورد خواهدشد.

مراجع:

[1] Minijava Grammar,
http://www.cambridge.org/us/features/052182060X/grammar.html

[2] JLex Homepage,
http://www.cs.princeton.edu/~appel/modern/java/JLex/

[3] CUP Homepage,
http://www.cs.princeton.edu/~appel/modern/java/CUP/

[4] JTB Auto-generated Classes,
http://compilers.cs.ucla.edu/jtb/jtb-2003/docs.html#auto