# پروژه درس اصول طراحی کامپایلر

## جدول نشانه‌ها و تحلیلگر معنایی

ترم اول سال
تحصیلی ۹۸ – ۹۷

هدف از این مرحله از پروژه تکمیل پیاده‌سازی زبان LULU نسخه 2.1 در ادامه مرحله اول است. به این منظور دانشجویان می‌توانند همچنان از ابزار ANTLR[1] استفاده کرده و کنش‌های معنایی مناسب را به قواعد تولید گرامر که در مرحله قبلی پیاده‌سازی شده است اضافه کنند، یا مستقیماً کدهای تولید شده در مرحله قبلی پروژه توسط ANTLR در زبان هدف را با پویش درخت تجزیه حاصل از برنامه تکمیل کنند. برای آشنایی بیشتر و توضیحات تکمیلی در مورد بکارگیری هر یک از این دو روش می‌توانید به راهنمای اینترنتی ANTLR[2] یا کتاب مرجع کامل ابزار در گروه درس (در LMS) مراجعه کنید.

**در این مرحله از پروژه دانشجویان باید با ایجاد و تکمیل جدول نشانه‌ها (symbol table) برای برنامه ورودی در حین (یا پس از) تجزیه، تحلیلگر معنایی (semantic analyzer) کامپایلر زبان LULU2.1 را برای بررسی سازگاری در برنامه ورودی پیاده‌سازی کنند. خروجی این مرحله از پروژه علاوه بر جدول(های) نشانه‌های ایجاد شده باید شامل پیام‌هایی برای اعلام هر نوع خطای معنایی شناسایی شده در صورت وجود و یا پیام عدم وجود خطای معنایی در برنامه باشد.**

# ۱ – توضیحات تکمیلی در مورد زبان LULU نسخه 2.1

در این قسمت ویژگی‌های بیشتری از زبان LULU که باید در این مرحله مدنظر قرار گیرد معرفی خواهد شد.

## ۱ – ۱ – نوع‌های کاربری

نوع‌های کاربری (user-defined types) می‌توانند شامل تعریف متغیرهای عضو (همان فیلد – field) و توابع باشند. همچنین یک نوع کاربری می‌تواند متغیرهای عضو و توابع خود را از نوع کاربری دیگر به ارث ببرد (inheritance). اعضای یک نوع کاربری می‌توانند دارای سطوح دسترسی private، protected و public باشند که مانند سایر زبان‌های شیءگرا به ترتیب دسترسی فقط در نوع کاربری تعریف شده، در زیرنوع‌های به ارث برده شده و در خارج از نوع کاربری را فراهم می‌کنند. نوع‌های کاربری تو در تو (nested) و واسطها (interfaces) در این زبان تعریف نشده‌اند. برای ایجاد یک نمونه (instance) از یک نوع کاربری، از کلیدواژه allocate استفاده شده و برای حذف آن از کلیدواژه destruct استفاده می‌شود. نوع‌های کاربری می‌توانند قبل از تعریف در ابتدای برنامه در ساختار declare (به گرامر زبان مراجعه کنید) اعلام شوند تا بتوان پیش از تعریف آنها در برنامه، متغیرهایی از

[1] www.antlr.org
[2] https://github.com/antlr/antlr4/blob/master/doc/index.md

نوع آنها تعریف کرد، اما هر نوع کاربری اعلام شده باید حتماً در برنامه تعریف شود. دقت کنید هر نوع کاربری قبل از استفاده برای تعریف متغیرها، پارامترهای توابع یا وراثت، باید حتماً تعریف یا اعلام شده باشد و نام آن در کل برنامه یکتا باشد. عرض نوع‌های کاربری بر اساس عرض فیلدهای آن تعیین می‌شود. هر فیلد و تابع که به عنوان عضو یک نوع کاربری بکار گرفته می‌شود باید در آن نوع کاربری یا ابرنوع‌های (supertypes) آن تعریف شده و همچنین با توجه به سطح دسترسی تعیین شده برای آن، در حوزه فعلی قابل دسترسی باشد.

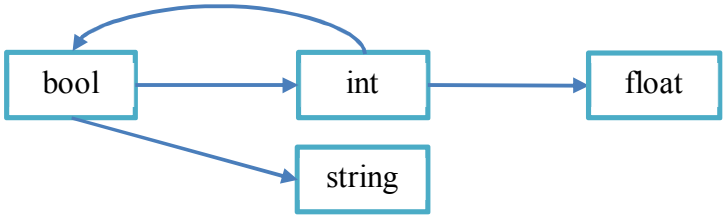
## ۱ – ۲ – متغیرها

متغیرها در زبان LULU می‌توانند دارای نوع پایه‌ای (primitive) یا نوع کاربری بوده و به صورت ساده یا به صورت آرایه یک یا چند بعدی تعریف می‌شوند. هر متغیر باید قبل از استفاده در دستورات و عبارت‌ها تعریف شده باشد. نوع‌های داده‌ای پایه در نظر گرفته شده در این زبان و اندازه‌های متناظر با هر یک عبارتند از:

| نوع | bool | int | float | string |
|---|---|---|---|---|
| اندازه | 1 | 4 | 8 | 2 + تعداد نویسه‌ها × 2 |

تمام متغیرهای تعریف شده با نوع‌های کاربری از نوع ارجاعی (reference) بوده و حاوی آدرس یک نمونه از آن نوع کاربری یا آدرس تهی (nil) هستند. بنابراین عرض یک متغیر از نوع کاربری برای تمام انواع کاربری ثابت و برابر با 4 واحد حافظه است.

نوع‌های پایه با توجه به شکل ۱ به صورت ضمنی توسط کامپایلر قابل تبدیل به یکدیگر هستند. برای تبدیل نوع int به bool هر مقدار غیر صفر برابر با true و مقدار صفر برابر با false در نظر گرفته می‌شود. برای تبدیل نوع bool به string مقدار false برابر با رشته ‘\X0’ (null) و مقدار true برابر با رشته ‘\X01’ است.




*شکل ۱: تبدیل نوع‌های پایه‌ای مجاز در زبان LULU*

متغیرهای آرایه‌ای برای ذخیره مقادیر همنوع در یک حافظه پیوسته و پشت سر هم مورد استفاده قرار می‌گیرد. برای تعریف یک متغیر به صورت آرایه در مقابل نوع آن از نویسه‌های ‘[’ و ‘]’ استفاده شود که در بین این دو نویسه اندازه آرایه قرار می‌گیرد. برای دسترسی به خانه‌های آرایه، اندیس خانه مربوطه در بین نویسه‌های ‘[’ و ‘]’ در مقابل نام متغیر آرایه‌ای آورده می‌شود (مثلاً a[3]). نوع عبارت مورد استفاده در اندیس‌دهی به خانه‌های یک آرایه باید int یا قابل تبدیل به int باشد. با بکارگیری بیش از یک جفت از نویسه‌های ‘[’ و ‘]’ می‌توان آرایه‌های چند بعدی تعریف و به خانه‌های آن دسترسی داشت. اندازه آرایه‌ها بر اساس تعداد درایه‌ها و اندازه هر درایه آنها مشخص می‌شود. فقط متغیرهای آرایه‌ای قابل اندیس‌دهی هستند.

متغیرهای سراسری فقط در ابتدای برنامه در ساختار declare (به گرامر زبان مراجعه کنید) تعریف می‌شوند و امکان تعیین سطح دسترسی را ندارند. متغیرهای تعریف شده به صورت ثابت (با کلیدواژه const) باید در زمان تعریف مقداردهی شده و پس از آن

دیگر امکان مقداردهی ندارند. هر متغیری حوزه (scope) تعریف مخصوص به خود را دارد. نام هر متغیر در هر حوزه باید یکتا باشد (امکان بازتعریف متغیر در حوزه‌ای که قبلاً در آن تعریف شده است وجود ندارد). متغیرهای سراسری از هر حوزه‌ای از برنامه قابل دسترسی هستند مگر آنکه یک متغیر محلی همنام با آنها در حوزه فعلی تعریف شده باشد. متغیرهای محلی نمی‌توانند همنام نوع‌های کاربری یا سایر توابع قابل دسترسی باشند. متغیرهای محلی فقط در داخل حوزه تعریف خود قابل دسترسی بوده و با اتمام این حوزه دیگر قابل دسترسی نیستند. فیلدهای تعریف شده در یک نوع کاربری نمی‌توانند همنام توابع در همان نوع کاربری یا ابرنوع‌های آن باشند و به صورت پیش‌فرض دارای سطح دسترسی private هستند به این معنی که در تمام توابع عضو آن نوع کاربری قابل دسترسی هستند (فقط با بکارگیری کلیدواژه this). متغیرها با سطح دسترسی protected در تمام توابع عضو نوع کاربری و تمام توابع عضو نوع‌های کاربری به ارث برده شده قابل دسترسی هستند (فقط با بکارگیری کلیدواژه super). متغیرها با سطح دسترسی public علاوه بر توابع ذکر شده قبلی در تمام توابع دیگری که یک متغیر از این نوع کاربری را تعریف کرده باشند قابل دسترسی هستند. یک فیلد اگر به صورت private در یک نوع کاربری تعریف شود می‌تواند در زیرنوع‌های (subtypes) آن نوع بازتعریف شود.

## ۱ – ۳ – عملگرها

همانطور که قبلاً توضیح داده شد، عملگرهای موجود در زبان می‌توانند بر اساس تعداد عملوندها، تکی (unary) یا دوتایی (binary) باشند. نوع عملوندهایی که در یک عملگر ریاضی یا مقایسه‌ای دوتایی استفاده می‌شود باید جزء نوع‌های پایه‌ای و همچنین یکسان یا قابل تبدیل به یکدیگر باشند. در صورتی که نوع عملوندها غیریکسان ولی قابل تبدیل به یکدیگر باشند، نوع کل عبارت ریاضی برابر با نوع بزرگتر (با تعداد واحدهای حافظه بیشتر) است. نوع عبارت مقایسه‌ای از نوع bool است. نوع عملوندهایی که در یک عملگر منطقی استفاده می‌شود باید bool یا قابل تبدیل به bool باشد. نوع کل عبارت منطقی bool است. نوع عملوندهای مورد استفاده در عملگرهای بیتی باید int یا قابل تبدیل به int باشد. نوع کل عبارت بیتی int است. نوع مبدأ و مقصد دستور انتساب باید یکسان یا نوع مبدأ باید قابل تبدیل به نوع مقصد باشد. برای نوع‌های کاربری اگر نوع مبدأ و مقصد یکسان نباشد، نوع مقصد باید ابرنوعی از نوع مبدأ باشد.

## ۱ – ۴ – توابع

توابع (functions) می‌توانند چندین پارامتر ورودی و چندین نتیجه خروجی داشته باشند. پارامترهای تابع به صورت ارجاعی به آن ارسال شده (call-by-reference) و نتایج خروجی به صورت مقداری (call-by-value) از آن برگردانده می‌شود. آرایه‌ها در این زمینه استثناء هستند و آرایه‌هایی که پارامترهای خروجی تابع باشند نیز به صورت ارجاعی برگردانده می‌شوند. در بدنه توابع امکان تعریف متغیرهای محلی و بکارگیری انواع دستورات تعریف شده در زبان وجود دارد. تابع با رسیدن به دستور return پایان می‌یابد که باید قبل از آن تمام پارامترهای خروجی تابع مقداردهی شده باشند. قبل از تعریف توابع سراسری می‌توان آنها را در ابتدای برنامه در ساختار declare (به گرامر زبان مراجعه کنید) اعلام کرد تا بتوانند در سایر توابع و نوع‌های کاربری مورد استفاده قرار گیرند، اما هر تابع اعلام شده باید حتماً در برنامه تعریف شود. توابعی که در نوع‌های کاربری تعریف می‌شوند دارای یکی از سطوح دسترسی private، protected و public هستند (سطح دسترسی پیش فرض private است) که مشابه متغیرهای یک نوع

کاربری می‌توانند مورد دسترسی قرار گیرند. در صورتی که در یک حوزه بیش از یک تابع با نام یکسان اما با امضاهای (signatures) متفاوت تعریف شوند، این تابع بارگذاری مضاعف (overload) شده است. دو تابع همنام با امضای یکسان نمی‌توانند در یک حوزه تعریف شوند.

هر تابع باید قبل از فراخوانی، تعریف یا اعلام شده باشد و در هنگام فراخوانی تعداد، نوع و ترتیب آرگومان‌های ورودی و خروجی آن با نحوه تعریف یا اعلام تابع سازگاری داشته باشد. برای پارامترهای ورودی و خروجی تابع که به صورت آرایه تعریف شده‌اند نوع پایه و تعداد ابعاد آرگومان مربوطه باید برابر با نوع و تعداد ابعاد تعریف شده برای تابع باشد.

در این زبان دو تابع ویژه وجود دارد که می‌توانند در تمام برنامه‌ها بدون اعلام یا تعریف مورد استفاده قرار گیرند. تابع read برای خواندن مقدار از ورودی (stdin) بر حسب نوع پارامتر آن است و نوع خروجی آن برابر با نوع پارامتر ورودی است. تابع write مقدار پارامتر خود را در خروجی (stdout) می‌نویسد. نوع خروجی این توابع int است. همچنین هر برنامه باید دقیقاً یک تابع سراسری (خارج از یک نوع کاربری) با نام start داشته باشد که هیچ پارامتر ورودی ندارد و نوع خروجی آن int است. این تابع نقطه شروع برنامه است و عدم وجود آن در برنامه خطاست.

### ۱ – ۵ – دستورات

مجموعه متنوعی از دستورات در زبان LULU در نظر گرفته شده است که در مرحله قبلی پروژه به آنها اشاره شد. قوانین خاص ناظر به دستورات زبان به شرح زیر هستند:

نوع عبارت استفاده شده در دستورات حلقه (for و while) و شرط (if) باید bool یا قابل تبدیل به bool باشد. نوع متغیر مورد استفاده در دستور switch باید int یا قابل تبدیل به int باشد. دستورات break و continue باید حتماً در محدوده بلوک مرتبط با یک دستور حلقه قرار داشته باشند. متغیر مورد استفاده در دستور destruct باید از نوع کاربری باشد یا به صورت آرایه تعریف شده باشد که در این صورت تعداد "[]" بعد از دستور destruct باید برابر با تعداد ابعاد این متغیر باشد.

## ۲ – جدول نشانه‌ها

جدول نشانه‌ها شامل لیستی از نام‌های قابل استفاده در هر حوزه (scope) از برنامه است و محیط (environment) مورد نیاز برای بررسی صحت عبارت‌های مختلف برنامه را تعیین می‌کند. با هر دستور تعریف (definition) یا اعلام (declaration) اطلاعات جدیدی به جدول نشانه‌ها اضافه می‌شود. جدول نشانه‌ها می‌تواند در حین تجزیه یا پس از آن با پویش درخت تجزیه (یا AST) ساخته و تکمیل گردد. به این منظور با رؤیت نشانه‌های غیرپایانی مرتبط با تعریف‌ها و اعلام‌ها (یا گره‌های مربوط به آنها از درخت تجزیه) اطلاعات مورد نیاز به جدول نشانه‌ها افزوده می‌شود. جدول نشانه‌ها باید بگونه‌ای پیاده‌سازی شود که امکان پشتیبانی از حوزه‌های مختلف موجود در برنامه‌های زبان LULU را داشته باشد. حداقل اطلاعات لازم در جدول نشانه‌ها که باید در هنگام نمایش آن در خروجی به کاربر نشان داده شود عبارتند از:

- نام‌ها (شناسه‌ها): شامل توابع، نوع‌های کاربری، متغیرهای سراسری، فیلدهای موجود در نوع‌های کاربری، پارامترهای ورودی و خروجی توابع و متغیرهای محلی تعریف شده در توابع هستند که هر یک باید در حوزه مرتبط با خود در نظر گرفته شوند.
- نوع هر یک از نام‌ها

- عرض (width) هر یک از نام‌ها: تعداد واحدهای حافظه مورد نیاز برای هر نام
  - عرض متغیری از نوع‌های پایه طبق مشخصات زبان در بخش ۱ – ۲ تعیین می‌شود.
  - عرض یک متغیر از نوع کاربری برای تمام انواع کاربری ثابت و برابر با 4 واحد حافظه است.
  - عرض نوع‌های کاربری بر اساس نوع و تعداد فیلدها به اضافه مقدار ثابت 10 واحد حافظه در نظر گرفته می‌شود.
  - عرض تابع (به عنوان یک حوزه) بر اساس پارامترهای ورودی، خروجی و نیز متغیرهای محلی آن تعیین می‌شود.
  - عرض آرایه بر اساس تعداد و نوع درایه‌های آن تعیین می‌شود.
    - عرض آرایه‌ای که به عنوان پارامتر ورودی یا خروجی یک تابع مشخص شده است (یک یا چندبعدی) برابر با 4 واحد حافظه است.
- آدرس نسبی نام متغیرها، فیلدها و پارامترها نسبت به ابتدای حوزه‌ای که در آن قرار گرفته‌اند
- مجموع واحدهای حافظه‌ای که در هر یک از حوزه‌ها برای نام‌ها باید در نظر گرفته شود (مثلاً در حوزه سراسری، در حوزه هر یک از نوع‌های کاربری، در حوزه هر یک از توابع: همان عرض تابع).

دانشجویان می‌توانند در صورت نیاز هر گونه اطلاعات دیگری را برای انجام تحلیل معنایی به جدول اضافه کنند.

# ۳ – کتاب مرجع

جهت اطلاعات بیشتر در مورد کدهای تولید شده توسط ANTLR و نحوه انجام عملیات تکمیلی در حین یا پس از تجزیه به کتاب مرجع ANTLR بخصوص بخش‌های زیر مراجعه کنید.

- 2.5: Parse tree listeners and visitors
- 3.1: The ANTLR Tool, Runtime and Generated Code
- 4.2: Building a Calculator Using a Visitor
- 4.3: Building a Translator with a Listener
- 4.4: Making Things Happen During the Parse
- 7: Decoupling Grammars from Application-Specific Code
- 8.4: Validating Program Symbol Usage
- 10: Attributes and Actions
- 13: Exploring the Runtime API
- 15.4: Attributes and Actions Reference

# ۴ – خروجی‌ها

پس از انجام تحلیل معنایی، خروجی تولید شده توسط کامپایلر باید شامل موارد زیر باشد:

- حوزه‌های تشخیص داده شده در برنامه که هر یک باید شامل اطلاعات زیر باشد:
  - نام‌های تعریف شده در هر حوزه به همراه نوع و عرض آنها
    - آدرس نسبی برای متغیرها، فیلدها و پارامترهای توابع
  - فضای حافظه مورد نیاز برای هر حوزه

- خطاهای معنایی شناسایی شده در برنامه ورودی بر اساس قوانین معنایی مشخص شده با پس‌زمینه زرد
  - برای هر خطا باید مکان آن در برنامه ورودی تعیین شود.

نحوه نمایش خروجی به دلخواه دانشجویان است و می‌تواند در یک واسط گرافیکی، فایل و حتی کنسول (console) باشد.

## ۵ – نکات قابل توجه

- این مرحله از پروژه باید در قالب همان گروه‌های تعیین شده قبلی انجام شود و گروه‌ها قابل تغییر نیست.
- تحویل این مرحله از پروژه به صورت حضوری و توسط تمام اعضای گروه خواهد بود. در زمان تحویل از تمام اعضای گروه سوال خواهد شد و همه اعضاء باید از نحوه پیاده‌سازی تمام بخش‌های پروژه اطلاعات کافی داشته باشند. نمره هر فرد از گروه بر اساس ارزیابی صورت گرفته در زمان تحویل پروژه و توانایی او در پاسخ به سوالات تعیین خواهد شد. بسته به کیفیت کار گروهی صورت گرفته در پیاده‌سازی پروژه، نمره گروه‌های چند نفره تا ۱٫۲ برابر در نظر گرفته می‌شود.
- زمان (احتمالاً اوایل بهمن ماه) و مکان تحویل پروژه متعاقباً به اطلاع دانشجویان خواهد رسید (از طریق LMS).
- هر گروه باید قبل از تحویل پروژه به صورت حضوری، فایلهای زیر را به صورت فشرده در قالب یک فایل zip به ایمیل professor.karshenas@gmail.com ارسال کند. از گروه‌هایی که فایل‌های خود را ارسال نکرده باشند پروژه تحویل گرفته نمی‌شود.
  - فایل ورودی برنامه ANTLR
  - سایر فایل‌های کد برای پیاده‌سازی این مرحله از پروژه
  - یک فایل توضیحات با نام ReadMe و به شکل یک فایل pdf حاوی اطلاعات زیر:
    - نام اعضای گروه همراه با شماره دانشجویی آنها
    - توضیح قابل قبول از نقش هر عضو در پیاده‌سازی این مرحله از پروژه و نحوه تقسیم کار
    - متن تعهدنامه اخلاقی زیر با درج نام دانشجویان

**ما (نام دانشجویان عضو گروه) تعهد می‌نماییم که تمام مراحل پروژه تحویل داده شده نتیجه کار گروهی ما بوده و در هیچ یک از بخشهای انجام شده از کار دیگران کپی برداری نشده است. در صورتی که مشخص شود که پروژه تحویل داده شده کار این گروه نبوده است، طبق ضوابط آموزشی با ما برخورد شده و حق اعتراض نخواهیم داشت.**

    - توضیح مختصر در مورد هر یک از فایل‌های کد تحویل داده شده
    - توضیح در مورد نحوه اجرای کامپایلر پیاده‌سازی شده
    - هرگونه توضیحات اضافی درباره پیاده‌سازی این مرحله از پروژه

- یادآوری می‌شود هر گروه برای نامگذاری عنوان ایمیل جهت ارسال پروژه باید از فرمت زیر استفاده کند در غیر این صورت به ایمیل ارسالی ترتیب اثر داده نمی‌شود.

Compiler97-phase(phase#)-group(group#)

مثلاً compiler97-phase2-group1، که در آن شماره گروه (#group) همان شماره اختصاص داده شده به هر گروه در لیست گروه‌های پروژه اعلام شده در گروه درس (در LMS) است.

- برای آگاهی از اطلاعیه‌های تکمیلی در مورد پروژه به صورت مداوم به گروه درس (در LMS) مراجعه کنید.

موفق باشید

کارشناس