



درس: طراحی کامپایلر

عنوان: پروژه تحلیل گر لغوی

عضو اول: علیرضا کریمی

شماره دانشجویی عضو اول: ۹۹۳۶۲۳۰۳۵

عضو دوم: علی پورقیصری

شماره دانشجویی عضو دوم: ۹۹۳۶۱۳۰۱۴

استاد: دکتر آرش شفیعی

فهرست مطالب

۱- مقدمه	۳
۲- مجموعه متغیر های سراسری	۳
۳- کلاس Token	۳
۴- کلاس Error	۳
۵- توابع مورد استفاده در طول کد	۳
۴-۱- تابع keyword	۳
۴-۲- تابع identifier	۴
۴-۳- تابع operator	۵
۴-۴- تابع comment	۵
۴-۵- تابع separator	۶
۴-۶- تابع value	۶
۴-۶- تابع analyze_file	۷
۴-۷- تابع main	۷
مراجع	۷

۱- مقدمه

در فاز اول پروژه، قرار بر این است که یک تحلیلگر لغوی را از صفر با استفاده از زبان پایتون پیاده سازی کنیم. تحلیلگر لغوی در این کد با استفاده از ترکیبی از دیکشنری ها برای تعریف کلمات کلیدی، عملگرها، جداکننده ها، علائم و کاراکترهای فضای خالی پیاده سازی شده است. کلاس تعریف شده در این کد ویژگی های هر توکن را در بر می گیرد و روش های درون آن بر اساس قوانین از پیش تعریف شده، نوع توکن را تعیین می کنند. تحلیلگر، خطوط ورودی را به توابع مختلفی که در ادامه توضیح می دهیم، پاس می دهد و کاراکترهای خطوط تشخیص داده شده و از هم متمایز می شوند سپس به لیست توکن ها اضافه می کند. به طور کلی تحلیلگر لغوی هر عنصر را به انواع نشانه های خاص دسته بندی می کند و نمایشی ساختاریافته از کد را برای پردازش یا تحلیل بیشتر ارائه می دهد.

۲- مجموعه متغیرهای سراسری

این متغیرهای سراسری که در سرتاسر کد برای دسته بندی نشانه ها در طول تحلیل لغوی استفاده می شوند، از داخل فایل constants اضافه شده اند. با استفاده از آنها منطق متمرکز برای تعریف و مدیریت انواع توکن ها، افزایش خوانایی، قابلیت نگهداری و انعطاف پذیری کد را میتوان فراهم نمود. این کاراکترها شامل انواع علائم منطقی، نشانه گذاری، بلوک بندی و ... می باشد که معمولاً در سایر زبان ها نیز وجود دارند.

۳- کلاس Token

کلاس Token در این کد به عنوان ساختاری برای نمایش توکن های فردی در کد استفاده شده است. این کلاس شامل ویژگی هایی برای ذخیره واژگان، نوع آن، نام نشانه یا شناسه، شماره کاراکتری که بر روی آن خطا افتاده، و موقعیت آن در کد (ردیف و ستون) است. در متد __str__ این کلاس، فرمت خروجی مد نظر ما هنگام چاپ کردن مشخص شده است.

۴- کلاس Error

کلاس Error در این کد به عنوان ساختاری برای نمایش خطا هایی که ممکن است در طول کامپایل اتفاق بیوفتد، پیاده سازی شده است. این کلاس شامل ویژگی هایی برای ذخیره واژگان، سطر، ستون، پیغام خطا و شماره کاراکتری که بر روی آن خطا اتفاق افتاده را از اول فایل نشان می دهد. در متد __str__ همانند کلاس قبل، خروجی به فرمت مد نظر چاپ می شود.

۵- توابع مورد استفاده در طول کد

در طول برنامه یکسری توابع استفاده شده است که پروسه تحلیل لغوی از طریق این توابع انجام می گیرد. این توابع به شرح زیر می باشند:

۴-۱ keyword(line, col, row, file_index): این تابع که برای هر کلمه موجود در خط صدا زده می شود، دنبال کلمات کلیدی می گردد به این صورت که ابتدا حرف اول کلمه را می بیند، سپس پردازش می کند که این حرف، ابتدای چه کلماتی هستند. سپس با پیمایش بر روی حروف بعدی، کلمه کلیدی مورد نظر پیدا می شود.

عبارت منظم:

$keyword \rightarrow [bool|break|char|continue|else|false|for|if|int|print|return|true]$

$bool \rightarrow bool$

$break \rightarrow break$

$char \rightarrow char$

$continue \rightarrow continue$

$else \rightarrow else$

$false \rightarrow false$

$for \rightarrow for$

$if \rightarrow if$

$int \rightarrow int$

$print \rightarrow print$

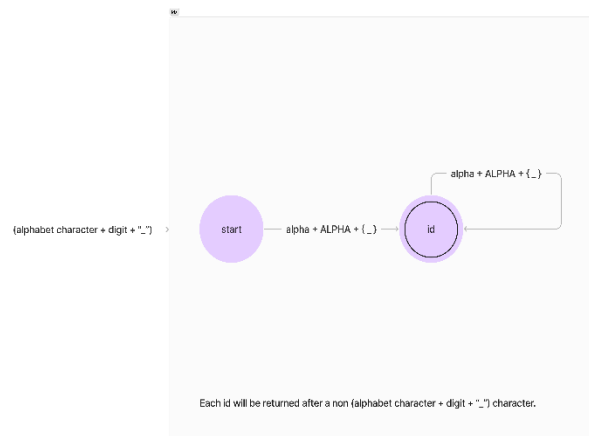
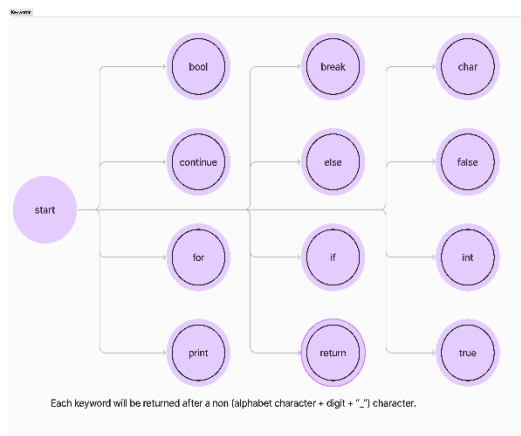
$return \rightarrow return$

$true \rightarrow true$

۴-۲- $identifier(line, col, row, file_index)$: این تابع اسامی متغیر ها و توابعی که در طول برنامه نوشته شده

است را مشخص و ذخیره می کند.

نمودار گذار:

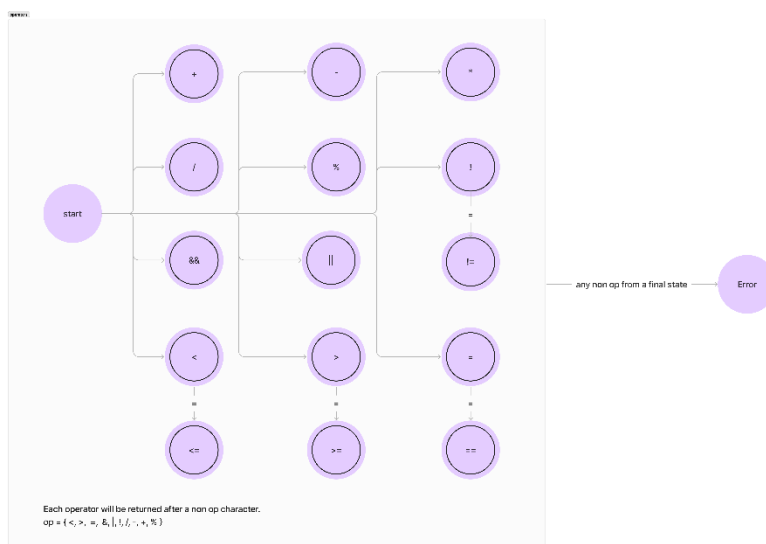


عبارت منظم:

$id \rightarrow [a-z|A-Z|underline]^+[0-9|a-z|A-Z|underline]^+$

$underline \rightarrow _$

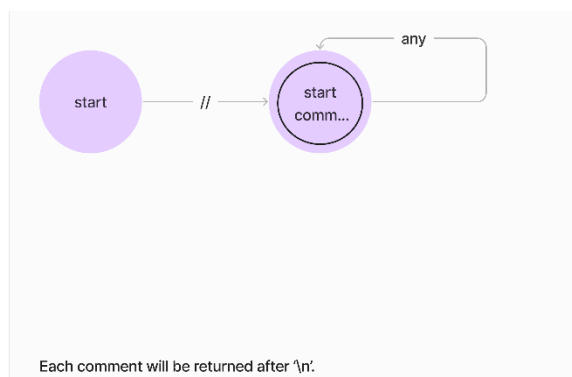
۳-۴- $operator(line, col, row, file_index)$: این تابع که برای هر عملگر موجود در خط صدا زده می‌شود، دنبال به دنبال عملگر هایی که مورد قبول است می‌گردد به این صورت که عملگر های تک کاراکتری را مشخص می‌کند، و اگر عملگر دو کاراکتری بود، در مرحله بعد تشخیص داده می‌شود.
نمودار گذار:



عبارت منظم:

$operator \rightarrow [+ | - | * | / | \% | = | == | >= | <= | > | < | ! | \&\& | || | !=]$

۴-۴- $comment(line, col, row, file_index)$: این تابع، زیر رشته هایی که با // شروع یا ادامه داده شوند را به عنوان کامنت در نظر می‌گیرد.
نمودار گذار:



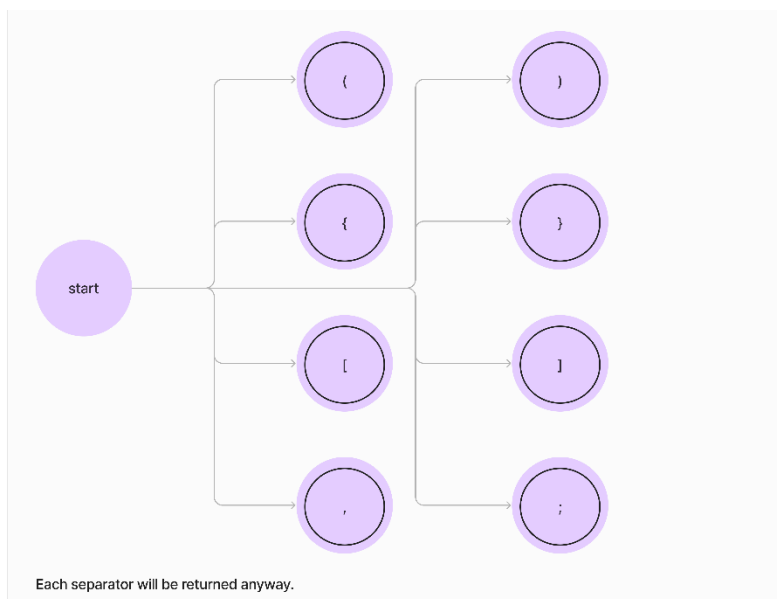
عبارت منظم:

$comment \rightarrow // [any - \{\backslash n\}]^*$

۴-۵) `separator(line, col, row, file_index)`: این تابع، تمام پرانتزها، کروشه‌ها، براکت‌ها، کاما‌ها و سمیکالن

ه‌ا‌را را تحلیل و ذخیره می‌کند.

نمودار گذار:



عبارت منظم:

$$separator \rightarrow [(|) | [|] | \{| \} | , | ;]$$

۴-۶- $value(line, col, row, file_index)$: این تابع، تمام مقادیر متعلق به متغیر ها، از جمله اعداد، کاراکتر ها و

رشته ها را تحلیل و شناسایی می کند.

نمودار گذار:



عبارت منظم:

$value \rightarrow [string \mid char \mid dec \mid hex]$
 $string \rightarrow " (any - \{ \} + \{ \backslash \}) * "$
 $char \rightarrow ' (any - \{ \} + \{ \backslash \}) '$
 $any \rightarrow any \text{ ascii character}$
 $dec \rightarrow digit^+$
 $hex \rightarrow 0xhex_digit^+$
 $digit \rightarrow [0 - 9]$
 $hex_digit \rightarrow [0 - 9 \mid a - f \mid A - F]$

۴-۶- `analyze_file(file_path)`: این تابع یک فایل را خط به خط، نویسه به نویسه می‌خواند، و اطلاعات این نویسه‌ها را به توابع توضیح داده شده می‌دهد تا تحلیل را بر روی آنها انجام دهند.

۴-۷- `main()`: این تابع جایست که `analyze_file(file_path)` فراخوانی می‌شود.

- chat.openai.com
- [Introduction of Lexical Analysis - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-of-lexical-analysis/)
- https://www.youtube.com/watch?v=O4Bt_CyZWbl
- <https://medium.com/@pythonmembers.club/building-a-lexer-in-python-a-tutorial-3b6de161fe84>