

دانشکده مهندسی کامپیوتر

**پروژه درس اصول طراحی کامپایلر مهندسي کامپیوتر**

**گزارش پروژه اصول طراحی کامپایلر**

**پریا خان‌جان 40117733**

**مهتا رنجبر دامغانی 40118813**

**حنانه حکاک 40117573**

**استاد راهنما:**

دکتر محمد‌ هادی علائیان

**10/11/1403**



###### چکيده

در این پروژه، در فاز اول قصد طراحی تحلیل‌گر لغوی1 و در فاز دوم طراحی تجزیه‌گر2 را داریم. این مراحل از فازهای اولیه کامپایل3 کردن در کامپایلر4 هستند که در تشخیص لغات و فرم ظاهری گرامر بسیار مهم هستند. در صورت نبود این دو مرحله عملیات کامپایل کردن انجام نمی‌شود. خروجی این مراحل در کامپایلر، به تحلیل‌گر مفهومی داده می‌شود. کامپایلر دارای دو بخش تجزیه و ترکیب است که این دو مرحله در بخش تجزیه قرار دارند. در بخش تجزیه، برنامه مبدا به اجزاء تشکیل‌دهنده آن تبدیل می‌شود و جزئیات کاملی از آن استخراج می‌شود. در ادامه کد میانی را تولید می‌کند و براساس آن، برنامه مقصد را در بخش ترکیب تولید می‌کند.

**کلید واژه:** تحلیل‌گر لغوی و تجزیه‌گر

[[1]](#footnote-1)

###### فهرست مطالب

عنوان صفحه

[فهرست شکل‏ها ه](#_Toc531381832)

[فصل 1- مقدمه... 7](#_Toc531381835)

[1-1- تحلیل‌گر لغوی 7](#_Toc531381836)

[1-2- تجزیه‌گر 8](#_Toc531381836)

[فصل 2- گزارش فاز اول پروژه 9](#_Toc531381838)

[2-1- مقدمه فاز اول 9](#_Toc531381839)

[2-2- ساختار کلی گرامر 9](#_Toc531381840)

[2-3- ساختار تجزیه‌گر لغوی 10](#_Toc531381841)

[2-4- نمونه خروجی 10](#_Toc531381840)

[فصل 3- گزارش فاز دوم پروژه 12](#_Toc531381842)

[3-1- مقدمه فاز دوم..... 12](#_Toc531381843)

[3-1- تعریف گرامر..... 12](#_Toc531381843)

[2-3- رفع ابهام گرامر 13](#_Toc531381841)

[2-4- نمونه‌ای از ورودی معتبر 13](#_Toc531381840)

[فصل 4- نتیجه‌گیری 15](#_Toc531381868)

[4-1- هدف پروژه 15](#_Toc531381869)

[پیوست ‌ه- واژه‏نامه فارسي-انگليسي 16](#_Toc531381874)

[پیوست ‌و- واژه‏نامه انگليسي-فارسي 18](#_Toc531381875)

[فهرست مرجع‏ها 20](#_Toc531381877)

د

فهرست شکل‏ها

ه

عنوان صفحه

[شکل1‑2- نمونه ورودی تحلیل‌گر لغوی 11](#_Toc530263777)

[شکل 2-2- نمونه خروجی تحلیل‌گر لغوی. 11](#_Toc530263778)

[شکل1‑3- نمونه ورودی تجزیه‌گر 14](#_Toc530263777)

# **فصل 1- مقدمه**

7

## **تحلیل‌گر لغوی**

تحلیل‌گر لغوی تابعی است که فاز اول کامپایل کردن یک برنامه است، در ورودی خود برنامه مورد کامپایل را بعنوان یک فایل متن باز نموده، کاراکتر به کاراکتر می‌خواند، با رسیدن به هر کاراکتر جداکننده یک لغت بعدی را تشخیص و تفکیک می‌کند. جداکننده‌ها مانند Tab, Blank یا New Line فقط به منظور جداسازی لغات بکار می‌روند. دسته دیگر از جداکننده‌ها مانند > و + و غیره، ارزش لغوی دارند. تحلیل‌گر لغوی در خروجی خود اطلاعات مربوط به لغت را در یک رکورد یا ساختار بنام بسته لغت1 قرار می‌دهد.

عمل تحلیل لغوی می‌تواند به دو مرحله تقسیم شود: اسکن، که رشته‌ی ورودی را به گروه‌هایی تقسیم‌بندی می‌کند و آن‌ها را به کلاس‌های بسته لغت دسته‌بندی می‌کند؛ و ارزیابی، که کاراکترهای ورودی خام را به مقادیر پردازش شده تبدیل می‌کند.

بسته لغت شامل اطلاعاتی در مورد لغت تشخیص داده شده توسط تحلیل‌گر لغوی مانند سطر، ستون و نوع لغت است. همچنین شامل واژگان کلیدی2، عملگرها3، جداکننده‌ها4، ثابت‌ها5 و شناسه‌ها6 هستند.

1Token

2Keyword

3Operator

4Delimiter

5Literal

6Identifier

## **تجزیه‌گر**

8

تجزیه‌گر فاز دوم عمل کامپایل است. گرامر مورد استفاده در این مرحله گرامر مستقل از متن1 است. در حین این مرحله از کامپایل است که خطاهای نحوی تشخیص داده می‌شوند زیرا صحت فرم ظاهری دستورالعمل‌های برنامه‌سازی در این مرحله تشخیص داده می‌شود. در مرحله تحلیل نحوی برنامه ورودی از نظر دستوری بررسی می‌شود. تحلیل‌گر نحوی یا تجزیه‌گر، برنامه ورودی را كه به صورت دنباله‌ای از بسته‌های لغت است، از تحلیل‌گر لغوی گرفته و تعین می‌کند که آیا این جمله‌بندی می‌تواند به وسیله گرامر زبان مورد نظر تولید شود یا خیر.

تجزیه‌گر در هر مرحله پیش‌بینی می‌کند که ترم بعدی چه باید باشد. عمل خواندن لغات از چپ به راست انجام می‌شود. با در دست داشتن یک یا بیشتر، از ترم‌های پیش‌بینی، تجزیه‌گر می‌تواند تصمیم‌گیری کند که کدام گسترش از گسترش‌های متفاوت ترم مورد انتظار را باید انتخاب نماید.

1Context Free

# **فصل 2- گزارش فاز اول پروژه**

9

## **2-1- مقدمه فاز اول**

در این پروژه، زبان طراحی شده و تجزیه‌گر لغوی پیاده‌سازی شده به طور کامل توضیح داده شده است. هدف از این گزارش ارائه توضیحات جامعی در مورد قوانین زبانی، کلمات کلیدی، نمادها، ساختارها و نحوه مدیریت خطاها در این زبان می‌باشد. این زبان برای پروژه تعریف شده طراحی و تجزیه‌گر آن به کمک انتلر1 پیاده‌سازی شده است.

**2-2- ساختار کلی گرامر**

این زبان طراحی شده شامل مجموعه‌ای از کلمات کلیدی، عملگرها، و ساختارهای کنترلی است که برای نوشتن کدهای قابل اجرا در این زبان تعریف شده اند.

زبان از ویژگی‌های زیر بهره‌مند است:

• پشتیبانی از انواع داده‌ها مانند عدد صحیح2، عدد اعشاری3 و متغیر بول4

• دستورات کنترلی مانند else-if ، while و for

• تعریف توابع با کلمه‌ی کلیدی fun

• عملیات منطقی (not, and, or) و محاسباتی (+، -، \*، %، /)

• مدیریت خطاهای لغوی برای شناسایی ورودی‌های غیرمجاز

1Antlr

2Int

3Float

4Bool

**2****-3- ساختار تجزیه‌گر لغوی**

10

تجزیه‌گر لغوی از قوانین زیر پیروی می‌کند:

1 - شروع پردازش:

پردازش از قاعده‌ی start شروع می‌شود که شامل تعریف مجموع‌های از دستورات است.

2 - دستورات:

هر دستور1 می‌تواند شامل کلمات کلیدی، عملگرها، یا مقادیر باشد.

3 - مدیریت خطاها:

خطاهای مربوط به فرم اعداد (صحیح، اعشاری) ، مربوط به شناسه2 ، مربوط به عملگرها و خطاهای تعریف‌نشده شناسایی می‌شوند.

• مدیریت اعداد صحیح: مدیریت این موضوع در قسمت ERROR\_INT است که برای حالاتی که بیش از نه رقم داشته باشد خطا گرفته شود و صفرهای قبل از عدد را در نظر نگیرید.

• مدیریت اعداد اعشاری: مدیریت این موضوع در قسمت ERROR\_FLOAT است که برای حالاتی که بیش از نه رقم قبل اعشار داشته باشد خطا گرفته شود، صفرهای قبل از عدد و انتهای بخش اعشاری را در نظر نگیرید و همچنین برای حالاتی که اعشار (.) بیش از یکی باشد با در جای اشتباه باشد خطا بگیرید.

• مدیریت عملگرها: در قسمت ERROR\_OP، اگر بیش از یک عملگر اصلی کنار هم بیاید، خطا می‌دهد.

• مدیریت شناسه‌ها: در قسمت ERROR\_ID، اگر شناسه با عدد یا حرف بزرگ شروع شود خطا می‌گیرید.

• مدیریت توکن‌های شناسایی نشده: در قسمت OTHER\_ERRORS در انتهای کد، به ازای مقادیری که توکن نیستند، خطا می دهد.

**2-4- نمونه خروجی**

تجزیه‌گر لغوی از انتلر برای پردازش متن استفاده می‌کند. تنظیمات شامل:

• فعال‌سازی backtrack برای تحلیل دقیق‌تر.

• تابع کمکی print برای نمایش نوع کلمات شناسایی شده.

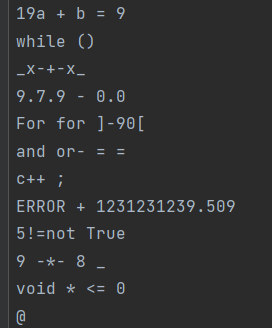
در زمان اجرا، تجزیه‌گر لغوی متن ورودی را پردازش کرده و نوع هر کلمه را شناسایی و چاپ می‌کند.

نمونه ورودی در شکل1‑2- خروجی شکل 2‑2- را نشان می‌دهد.

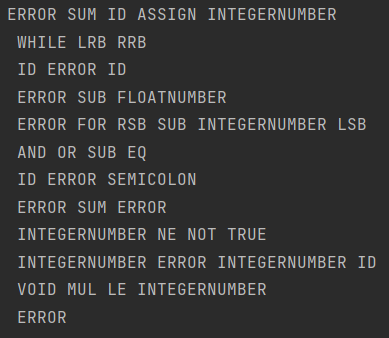
1statement

2Id

11



شکل1‑2- نمونه ورودی تحلیل‌گر لغوی



شکل2‑2- نمونه خروجی تحلیل‌گر لغوی

# **فصل3- گزارش فاز دوم پروژه**

12

## **3-1- مقدمه فاز دوم**

در این پروژه، هدف اصلی طراحی و پیاده‌سازی یک تجزیه‌گر برای زبان برنامه‌نویسی ساده‌ای بود که گرامر آن در بخش قبل تعریف شده است. این تجزیه‌گر با استفاده از ابزار انتلر ایجاد شده و توانایی تحلیل و تجزیه کدهای نوشته شده مطابق با این گرامر را دارد. در فاز اول، یک تحلیل‌گر لغوی طراحی شد که ورودی برنامه را به بسته‌های لغت تبدیل می‌کند. در فاز دوم، تجزیه‌گر طراحی و پیاده‌سازی گردید که از این بسته‌های لغت برای تحلیل و اعتبارسنجی نحوی کد استفاده می‌کند. بسته‌های لغت تعریف شده در فاز اول را هم در انتهای گرامر جدید ضمیمه شده است. این گزارش شامل جزئیات طراحی گرامر، پیاده‌سازی تجزیه‌گر، و ارزیابی عملکرد آن می‌باشد.

## **3-2- تعریف گرامر**

گرامر مورد استفاده در این پروژه بر اساس نیازمندی‌های تعریف شده در مستندات پروژه و با در نظر گرفتن اصول زیر طراحی شده است:

1 - پشتیبانی از ساختارهای برنامه نویسی پایه مانند تعریف متغیرها، توابع، شرط‌ها، حلقه‌ها، و دستورات بازگشتی.

2 - رفع ابهام‌های1 گرامری از طریق اولویت‌بندی عملگرها و تغییرات جزئی در ساختار گرامر، بدون تغییر زبان پذیرنده.

3 - تعریف یک زبان ساده و قابل فهم که از توابع بازگشتی و داده‌های اولیه مانند اعداد صحیح، اعداد اعشاری، و متغیرهای بولی پشتیبانی کند.

1Ambiguity

**3-3- رفع ابهام**

13

• فرم B -> A I B A I ε که در قواعد declist ، stmtlist ، cases و elseiflist دیده می‌شود و خود بازگشتی چپ دارند را به صورت B -> A\* نوشته شده است.

• در قواعدی که مجموعه سرآغار1 تکراری داشتند، عمل فاکتورگیری انجام داده شده است. مثلا قواعد iddec ، funcdec ، paramdec، lvalue و بخش‌های for و if در قسمت stmt .

• فرم B -> A I B, A در قواعد explist ، paramdecslist وidlist دیده می‌شود و باعث خود بازگشتی چپ می‌شود را به صورت B -> A (, A)\* نوشته شده است.

• در exp، به خاطر فراخوانی relopexp، خودبازگشتی چپ دارد. برای رفع ابهام این موضوع، کل قاعده relopexp را درexp جایگزاری شده است و چون این قاعده استفاده‌ای در جای دیگری از گرامر نداشت، آن گرامر حذف شده است.

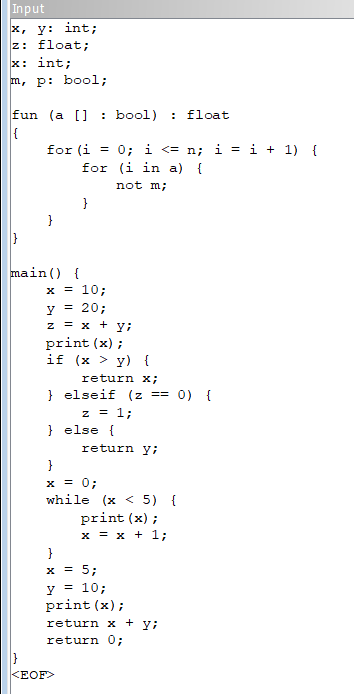
• از آن جایی که exp در قسمت‌های زیادی از گرامر کاربرد دارد، برای از بین بردن ابهام آن و رفع خودبازگشتی چپ، عملوندها از آن جدا شده‌اند و در قاعده simpleExp قرار داده شده است. برای تکرار سمت راست عملیات‌ها از \* استفاده شده است. چون عملوندها را از عملگرها جدا شده، در سمت چپ قاعده simpleExp تکرار می‌شود که برای رفع ابهام این موضوع، عمل فاکتورگیری انجام شده است.

## **3-4- نمونه‌ای از ورودی معتبر**

در خروجی در antlr workspace ، درخت خلاصه نحوی ورودی نشان داده می‌شود.

نمونه ورودی‌ای به تجزیه‌گر که در شکل1‑3- نشان‌داده شده است، داده می‌شود که در خروجی درخت آن داده می‌شود که نشان‌دهنده معتبر بودن ورودی است.

1First



14

شکل1‑3- نمونه ورودی تجزیه‌گر

# **فصل4- نتیجه‌گیری**

15

**4-1- هدف پروژه**

این پروژه به طراحی و پیاده‌سازی یک کامپایلر ساده برای زبان برنامه‌نویسی اختصاص دارد. پروژه در دو فاز اصلی تقسیم‌بندی شده است. فاز اول طراحی تحلیل‌گر لغوی که به شناسایی و تفکیک بسته‌های لغت از ورودی پرداخته و اطلاعات مربوط به هر بسته‌ لغت را در ساختار خاصی ذخیره می‌کند و این مرحله شامل مدیریت خطاهای مربوط به ورودی‌های نامعتبر نیز می‌باشد، فاز دوم طراحی تجزیه‌گر است که وظیفه بررسی ساختار نحوی برنامه را بر عهده دارد و اطمینان حاصل می‌کند که ورودی‌ها با گرامر زبان مطابقت دارند.

16

پیوست ه- واژه‏نامه فارسي-انگليسي

| Equivalent English  17 | واژه فارسی |
| --- | --- |
| Ambiguity | ابهام |
| Antlr | انتلر |
| Token | بسته لغت |
| Parser | تجزیه‌گر |
| Lexical analyzer | تحلیل‌گر لغوی |
| Literal | ثابت |
| Delimiter | جداکننده |
| Statement | دستور |
| Id | شناسه |
| Identifier | شناسه |
| Float | عدد اعشاری |
| Int | عدد صحیح |
| Operator | عملگر |
| Compile | کامپایل |
| Compiler | کامپایلر |
| Bool | متغیر بول |
| First | مجموعه سرآغاز |
| Context Free | مستقل از متن |
| Keyword | واژگان کلیدی |

18

پیوست و - واژه‏نامه انگليسي-فارسي

| Equivalent English  19 | واژه فارسی |
| --- | --- |
| Ambiguity | ابهام |
| Antlr | انتلر |
| Bool | متغیر بول |
| Compile | کامپایل |
| Compiler | کامپایلر |
| Context Free | مستقل از متن |
| Delimiter | جداکننده |
| First | مجموعه سرآغاز |
| Float | عدد اعشاری |
| Id | شناسه |
| Identifier | شناسه |
| Int | عدد صحیح |
| Keyword | واژگان کلیدی |
| Lexical analyzer | تحلیل‌گر لغوی |
| Literal | ثابت |
| Operator | عملگر |
| Parser | تجزیه‌گر |
| Statement | دستور |
| Token | بسته لغت |

فهرست مرجع‏ها

20

|  |  |  |
| --- | --- | --- |
| [1] |  | A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, “Compilers Principles, Techniques & Tools”, Prentice Hall, 2006. |
| [2] |  | T. Parr, ""Language Implementation Patterns, Pragmatic Bookshelf *,* 2010. |
|  |  |  |



21

**K. N. Toosi University of Technology**

**Faculty of Computer Engineering**

**A Report Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science (B.Sc.)**

**in Computer Engineering -** Choose an item.

**Principles of compiler design Project**

**By:**

Hannaneh Hakkak 40117573

Paria Khanjan 40117733

Mahta Ranjbar Damghani 40118813

**Advisor:**

Dr. Alaeiyan

1/29/2025

1. Lexical analyzer

   2 Parser

   3 Compile

   4 Compiler [↑](#footnote-ref-1)