دانشکده مهندسی کامپیوتر

پروژه درس اصول طراحی کامپایلر مهندسی کامپیوتر

گزارش پروژه اصول طراحی کامپایلر

پریا خان‌جان ۴۰۱۱۷۷۳۳

مهتا رنجبر دامغانی ۴۰۱۱۸۸۱۳

حنانه حکاک ۴۰۱۱۷۵۷۳

استاد راهنما:

دکتر محمد هادی علائیان

۱۴۰۳/۱۱/۱۰


بسم الله الرحمن الرحیم

چکیده

در این پروژه، در فاز اول قصد طراحی تحلیل‌گر لغوی[۱] و در فاز دوم طراحی تجزیه‌گر[۲] را داریم. این مراحل از فازهای اولیه کامپایل[۳] کردن در کامپایلر[۴] 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

---

[۱] Lexical analyzer
[۲] Parser
[۳] Compile
[۴] Compiler

فهرست مطالب

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

فصل ۱- مقدمه

۱-۱- 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 لغت[۱] 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 کلیدی[۲]، عملگرها[۳]، جداکننده‌ها[۴]، ثابت‌ها[۵] و شناسه‌ها[۶] هستند.

[۱] Token
[۲] Keyword
[۳] Operator
[۴] Delimiter
[۵] Literal
[۶] Identifier

۱-۲- تجزیه‌گر

تجزیه‌گر فاز دوم عمل کامپایل است. گرامر مورد استفاده در این مرحله گرامر مستقل از متن[۱] 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 که 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.

[۱] Context Free

فصل ۲- گزارش فاز اول پروژه

۲-۱- 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 انتلر[۱] پیاده‌سازی شده است.

۲-۲- 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 صحیح[۲]، عدد اعشاری[۳] و متغیر بول[۴]
- 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

[۱] Antlr
[۲] Int
[۳] Float
[۴] Bool

۲-۳- ساختار تجزیه‌گر لغوی

تجزیه‌گر لغوی از قوانین زیر پیروی می‌کند:

۱ - شروع پردازش:

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

۲ - دستورات:

هر دستور[۱] می‌تواند شامل کلمات کلیدی، عملگرها، یا مقادیر باشد.

۳ - مدیریت خطاها:

خطاهای مربوط به فرم اعداد (صحیح، اعشاری) ، مربوط به شناسه[۲] ، 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 یا 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.

۲-۴- 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 شکل۱-۲- خروجی شکل ۲-۲- را نشان می‌دهد.

[۱] statement
[۲] Id

```
19a + b = 9
while ()
_x-+-x_
9.7.9 - 0.0
For for ]-90[
and or- = =
c++ ;
ERROR + 1231231239.509
5!=not True
9 -*- 8 _
void * <= 0
@
```

شکل۱-۲- نمونه ورودی تحلیل‌گر لغوی

```
ERROR SUM ID ASSIGN INTEGERNUMBER
 WHILE LRB RRB
 ID ERROR ID
 ERROR SUB FLOATNUMBER
 ERROR FOR RSB SUB INTEGERNUMBER LSB
 AND OR SUB EQ
 ID ERROR SEMICOLON
 ERROR SUM ERROR
 INTEGERNUMBER NE NOT TRUE
 INTEGERNUMBER ERROR INTEGERNUMBER ID
 VOID MUL LE INTEGERNUMBER
 ERROR
```

شکل۲-۲- نمونه خروجی تحلیل‌گر لغوی

فصل۳- گزارش فاز دوم پروژه

۳-۱- 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.

۳-۲- 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:

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

۲ - رفع ابهام‌های[۱] گرامری از طریق اولویت‌بندی عملگرها و تغییرات جزئی در ساختار گرامر، بدون تغییر زبان پذیرنده.

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

[۱] Ambiguity

۳-۳- رفع ابهام

• 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 سرآغار[۱] 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.

۴-۳- نمونه‌ای از ورودی معتبر

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

نمونه ورودی‌ای به تجزیه‌گر که در شکل۱-۳- نشان‌داده شده است، داده می‌شود که در خروجی درخت آن داده می‌شود که نشان‌دهنده معتبر بودن ورودی است.

[۱] First

Input

```
x, y: int;
z: float;
x: int;
m, p: bool;

fun (a [] : bool) : float
{
    for(i = 0; i <= n; i = i + 1) {
        for (i in a) {
            not m;
        }
    }
}

main() {
    x = 10;
    y = 20;
    z = x + y;
    print(x);
    if (x > y) {
        return x;
    } elseif (z == 0) {
        z = 1;
    } else {
        return y;
    }
    x = 0;
    while (x < 5) {
        print(x);
        x = x + 1;
    }
    x = 5;
    y = 10;
    print(x);
    return x + y;
    return 0;
}
<EOF>
```

شکل۱-۳- نمونه ورودی تجزیه‌گر

فصل۴- نتیجه‌گیری

۴-۱- 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.

پیوست ه- واژه‌نامه فارسی-انگلیسی

Equivalent English	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

پیوست و - واژه‌نامه انگلیسی-فارسی

Equivalent English	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

[۱] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, “Compilers Principles, Techniques & Tools”, Prentice Hall, ۲۰۰۶.

[۲] T. Parr, ""Language Implementation Patterns, Pragmatic Bookshelf , ۲۰۱۰.

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 ٤٠١١٧٥٧٣

Paria Khanjan ٤٠١١٧٧٣٣

Mahta Ranjbar Damghani ٤٠١١٨٨١٣

Advisor:

Dr. Alaeiyan

١/٢٩/٢٠٢٥