پروژه درسی (تعریف)

نکات:

۱- عنوان پروژه :

طراحی و پیاده سازی Lexical Analyzer و Syntax Analyzer برای زبان برنامه نویسی مایک

۲- فایل‌های آموزشی لازم برای پروژه، در اختیار دانشجویان قرار خواهد گرفت.

۳- جهت انجام پروژه مذکور لازم است که دانشجویان در گروه های ۲ نفره سازمان دهی شوند . در صورت تمایل افراد، تشکیل گروه با تعداد نفرات بیشتر نیز امکان پذیر است.

۴- تحویل پروژه به صورت آنلاین می باشد (زمان تحویل از طریق گروه تلگرامی درس به اطلاع دانشجویان خواهد رسید).

۵- در زمان تحویل آنلاین پروژه، تمام افراد گروه باید ضمن حضور در جلسه تحویل، به تمام ابعاد پروژه تسلط کامل داشته باشند. در غیر اینصورت، هیچ نمره ای به ایشان تعلق نخواهد گرفت.

۶- تشابه بیش ازحد کدهای دو گروه، به منزله‌ی تخلف بوده و نمره ای برای گروه های متخلف در نظر گرفته نمی شود.

۶- در صورت وجود ابهام یا سوال درباره پروژه، می توانید از طریق تلگرام (@fadaeii1) با طراح پروژه در ارتباط باشید.

۱- مقدمه

در این پروژه قصد داریم با کمک ابزار های Flex و Bison کامپایلری برای یک زبان برنامه نویسی ساده، طراحی کنیم. در این کامپایلر فقط ماژول های Lexical Analyzer و Syntax Analyzer پیاده سازی خواهند شد.

۲- معرفی ابزار Flex

برای ساخت Lexical Analyzer باید مراحل زیر را به ترتیب پیاده سازی کنیم:

۱- Lexical Specification

۲- Regular Expression

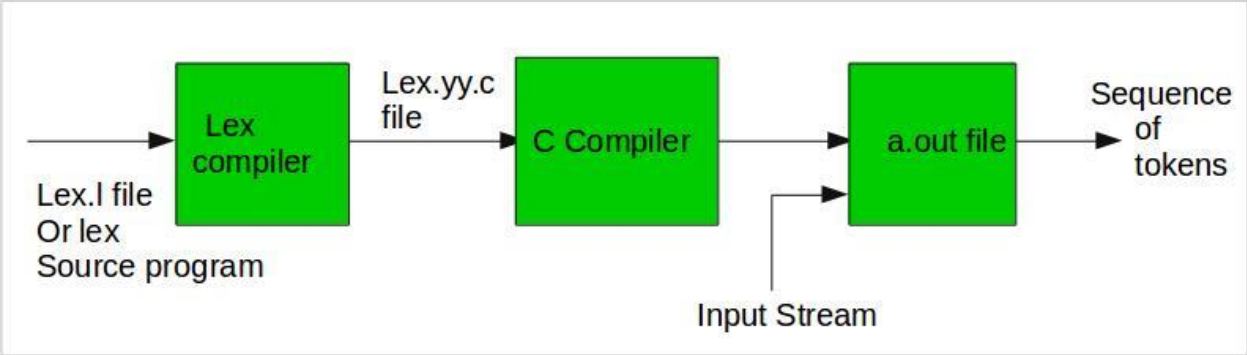
۳- NFA

۴- DFA

۵- Table implementation of DFA

ابزار Flex می تواند مراحل Regular Expression به NFA و NFA به DFA را به صورت خودکار برای ما پیاده سازی کند.

مراحل استفاده از ابزار Flex در شکل (۱) مشخص شده است.



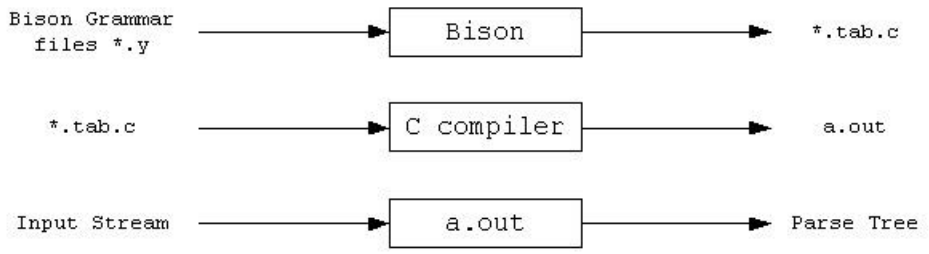
شکل ۱- مراحل استفاده از ابزار Flex

۳- معرفی ابزار Bison

ابزار Bison با استفاده از context-free grammar که برای زبان برنامه نویسی مورد نظر طراحی شده است، یک bottom-up parser پیاده سازی می کند (Syntax Analyzer).

در ادامه، می توان برای هر قانون از قوانین این grammar یک Semantic action با زبان برنامه نویسی C پیاده سازی کرد.

مراحل استفاده از ابزار Bison در شکل (۲) مشخص شده است.



شکل ۲ - مراحل استفاده از ابزار Bison

۴- زبان برنامه نویسی مورد نظر

زبان برنامه نویسی مورد نظر در این پروژه، شباهت زیادی به زبان برنامه نویسی C دارد ولی برای سادگی بخش هایی از آن حذف شده اند.

۴-۲- Token های زبان برنامه نویسی مایک

۴-۲-۱- کلمات کلیدی زبان

کلماتی هستند که در یک زبان برنامه نویسی نمی توان از آن ها به عنوان مفهوم دیگری مانند نام متغیر ها استفاده کرد. کلمات کلیدی مورد استفاده در این پروژه برای زبان مورد نظر ما شامل موارد زیر می باشند:

[int, char, if, else, elseif, while, for, return, void, main, continue, break]

۴-۲-۲- نام متغیر ها و توابع

باید ترکیبی از حروف انگلیسی بزرگ یا کوچک، اعداد و ’_’ باشد که نمیتواند با عدد شروع شود

۴-۲-۳- مقادیر ثابت

می‌توانیم از اعداد صحیح (مثبت یا منفی) برای متغیرهای int و از کاراکترها برای char استفاده کنیم (مقدارهای ثابت برای کاراکتر ها بین " " قرار می‌گیرند.). همچنین بزرگ‌ترین و کوچک‌ترین مقدار برای اعداد صحیح متناسب با یک سیستم ۱۶ بیتی می باشد.

۴-۲-۴- Comment های تک خطی و چند خطی

Comment های تک خطی با # در ابتدا خط کد و Comment های چند خطی با !# شروع و با #! پایان می یابند.

۴-۲-۵- عملگر ها

در این زبان، عملگرهای شرطی، منطقی یا محاسباتی داریم:

```
[<    <=    ==    !=    >     >=
 |    &     ||    &&    ^     !
 +    -     *     / ]
```

۴-۲-۶- سایر Token ها

Token هایی که در هیچ‌یک از گروه های پنج گانه فوق قرار نمی گیرند

```
(     )
[     ]
{     }
,     .
```

نکته: فاصله های بین Token ها بی تاثیر بوده و باید نادیده گرفته شوند.

۴-۳- قواعد زبان برنامه نویسی مایک

۴-۳-۱- برنامه شامل یک تابع **main** است.

این تابع کاربرد مشابهی با تابع main زبان برنامه نویسی C دارد.

۴-۳-۲- ساختار IF

```
if (condition)
{
    #code here
}
elseif(condition)
{
    #code here
}
else
{
    #code here
}
```

۴-۳-۳- ساختار While

```
While(condition)
{
    #code here
}
```

۴-۳-۴- ساختار For

```
for(variable definition, condition, step)
{
    #code here
}
```

۴-۳-۵- تعریف متغیر

```
int variable.
char string = "H".
```

۴-۳-۶- توابع

توابع موجود در برنامه prototype نداشته و حتما قبل از تابع main تعریف می شوند. توابع می توانند حداقل یک و حداکثر سه آرگومان ورودی داشته باشند.

نوع خروجی توابع نیز می توانند به صورت void, int یا char باشد.

۴-۳-۷- نکات

- تمامی دستورات به جز دستورات شرطی و حلقه ها با کاراکتر . (dot) پایان می‌پذیرند.
- هر بلاک کد بین {} قرار می گیرد.

۵- خطا های کامپایلر

- برنامه شما باید بتواند هرگونه خطا در فایل ورودی را تشخیص داده و آن را با پیام مناسب چاپ کند.
- در صورت وجود خطا در کد ورودی، عمل کامپایل بدون تولید کدی پایان پذیرد.

۶- لینک های مفید

- https://medium.com/@ilyarudyak/flex-tutorial-9ed34fd1ff28
- http://alumni.cs.ucr.edu/~lgao/teaching/bison.html
- https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/
- http://www.cs.ecu.edu/karl/4627/spr18/Notes/Bison/run.html
- https://aquamentus.com/flex_bison.html
- https://www.gnu.org/software/bison/manual/html_node/index.html#SEC_Contents