پروژه‌ی درس کامپایلر

دانشگاه صنعتی اصفهان

استاد درس: زینب زالی

بهار ۹۹

فهرست مطالب

تعریف پروژه ۳

هدف ۳

کلمات کلیدی: ۳

متغیرها: ۳

کامنتها: ۴

توکنها: ۴

مقادیر ثابت: ۴

عملگرها و توکنهای خاص: ۵

اسامی توکنها: ۵

تعریف پروژه

در این پروژه قصد داریم که یک کامپایلر برای یک زبان ساده‌ی خود تعریف، طراحی کنیم. پیاده سازی این کامپایلر باید با استفاده از ابزار بایسون و فلکس انجام شود.

کامپایلر هدف باید بتواند یک فایل حاوی کد ورودی را دریافت کند و با در نظر گرفتن semantic action و دیگر مفایم لازم که در درس کامپایلر خوانده‌اید یک کد خروجی به زبان MIPS تولید کند.

هدف از طراحی این پروژه این است که کامپایلر مذکور را قدم به قدم و همگام با مطالبی که در درس می‌آموزید پیاده کنیم تا با جنبه‌های عملی نوشتن یک کامپایلر ابتدائی، آشنا شوید.

بنابراین در این بخش سعی داریم که به طراحی قسمت اول یک کامپایلر، یعنی تحلیل‌گر لغوی، بپردازیم.

هدف

شما باید یک کد به زبان flex بنویسید و آن را در سامانه آپلود کنید.

یک برنامه به زبانی که توصیف خواهد شد به کد شما داده می‌شود، در صورتی که آن برنامه قواعد لغوی زبان برنامه نویسی را رعایت کرده باشد، شما باید در خروجی، توکن‌های آن برنامه را چاپ کنید و در غیر این صورت بدون تولید هر گونه توکنی، باید خطای مناسب را چاپ نمایید.

کلمات کلیدی:

هر زبان برنامه نویسی دارای یک سری کلمات کلیدی است که نمی‌توان از آنها به عنوان مفهوم دیگری همانند اسم متغیرها استفاده کرد؛ در این پروژه نیز زبان ورودی دارای تعدادی کلمات کلیدی است که لیست آن‌ها به شکل زیر است:

void	int	foreach	return
if	else	main	float
double	string	char	break
continue			

متغیرها:

در زبان ورودی، متغیرها ترکیبی از حروف، اعداد انگلیسی و خط تیره هستند که حتما باید با یک حرف ورودی شروع شوند.

زبان ورودی حساس به بزرگ(کوچک) بودن حروف است، بنابراین می‌توانیم دو متغیر به فرم pro و Pro داشته باشیم.

تعداد حروف هر متغیر نباید از ۳۱ عدد عبور کند.

کامنت‌ها[1]:

کامنت‌ها در زبان ورودی دقیقا همانند کامنت ها در زبان سی هستند:

۱. کامنت‌های تک خطی که با علامت // شروع می‌شوند.
۲. کامنت‌های چند خطی که با علامت */ شروع می‌شوند و با علامت /* پایان می‌یابند.

توکن‌ها:

توکن ها از طریق فاصله(منظور هر نوع فاصله‌ای، whitespace، است، همانند tab و space و ...) و یا از طریق توکن‌های خاص از هم جدا می‌شوند.

هر تعداد فاصله که بین دو توکن وارد شود بی‌تاثیر است و باید نادیده گرفته شوند.

مقادیر ثابت:

در زبان ورودی، با متغیرهای integer، float ، double ، string ، char و سر و کار داریم پس مقادیر ثابت که داریم شامل اعداد صحیح و اعشاری، یک کاراکتر و یک رشته از کاراکترها خواهد بود.

دقت شود که در این پروژه، اعداد صحیح (بدون هر گونه اعشار) علامت دار هستند که بزرگترین مقدار آن باید 2^{31} (متناسب با یکی سیستم ۳۲ بیتی) باشد.

مقادیر اعشاری نیز می‌توانند هم به فرم ساده(گسترده) و هم به فرم نماد علمی نوشته شوند[2].

همانند زبان C، مقادیر کاراکتر با علامت ' و مقادیر string با " مشخص می‌شوند که این علامت‌ها هم عضو مقادیر محسوب می‌شوند(یعنی توکن جداگانه‌ای تشکیل نمی دهند).

---

[1] اصولا کامنت‌ها به وسیله‌ی preprocessor پردازش می‌شوند و کامپایلر وظیفه‌ی پردازش آنها را ندارد، اما چون در این پروژه، preprocessor وجود ندارد باید به وسیله‌ی تحلیل گر لغوی پردازش شوند.

[2] برای اعداد اعشاری، نیازی به چک کردن بازه ی اعداد نیست. فرض کنید که اعداد داده شده حتما در متغیر مورد نظر جای می‌گیرد.

عملگرها و توکن‌های خاص:

عملگرهایی که در زبان ورودی مجاز هستند شامل عملگرهای محاسباتی، منطقی و شرطی می‌شوند که لیست آن‌ها در زیر آورده شده است:

+ - * / < <= == != > >= | & ||

&& ^ !

توکن‌های خاص به توکن‌هایی گفته می‌شود که نه متغیر هستند و نه کلمه‌ی کلیدی و نه عملگر که لیست آن‌ها در زیر آمده است:

( ) { } ; , [ ] ..

اسامی توکن‌ها:

همان‌گونه که در بالا گفته شد، چنانچه که برنامه‌ی درستی به تحلیل‌گر شما داده شود باید بتواند که توکن‌های آن را استخراج کند.

به منظور استخراج توکن‌ها از برنامه‌ی ورودی، تنها نام آن توکن و مقدار آن را در خروجی بنویسید(ابتدا نام توکن و سپس مقدار آن).

برای مثال:

```
Input Code:
    int x;
analyzer output:
    TOKEN_INTTYPE    int
    TOKEN_WHITESPACE
    TOKEN_ID         x
    TOKEN_SEMICOLON  ;
```

در ادامه لیست نام توکن‌ها آورده شده است:

نام توکن	توکن
TOKEN_VOIDTYPE	void
TOKEN_INTTYPE	int

foreach	TOKEN_LOOP
return	TOKEN_RETURN
if	TOKEN_IFCONDITION
Variable(متغير)	TOKEN_ID
+	TOKEN_ARITHMATICOP
-	TOKEN_ARITHMATICOP
*	TOKEN_ARITHMATICOP
/	TOKNE_ARITHMATICOP
&&	TOKEN_LOGICOP
&	TOKEN_BITWISEOP
|	TOKEN_BITWISEOP
||	TOKEN_LOGICOP
<=	TOKEN_RELATIONOP
<	TOKEN_RELATIONOP
>	TOKEN_RELATIONOP
=	TOKEN_ASSIGNOP
>=	TOKEN_RELATIONOP
==	TOKEN_RELATIONOP
!=	TOKEN_RELATIONOP
^	TOKEN_ARITHMATICOP
!	TOKEN_LOGICOP
(	TOKEN_LEFTPAREN
)	TOKEN_RIGHTPAREN
{	TOKEN_LCB
}	TOKEN_RCB
;	TOKEN_SEMICOLON
,	TOKEN_COMMA
..	TOKEN_UNTIL
[	TOKEN_LB
]	TOKEN_RB
\n [newLine]	TOKEN_WHITESPACE
\t [tab]	TOKEN_WHITESPACE
[space]	TOKEN_WHITESPACE
//[some string until \n]	TOKEN_COMMENT
/*[some string]*/	TOKEN_COMMENT
double	TOKEN_DOUBLETYPE

float	TOKEN_FLOATTYPE
char	TOKEN_CHARTYPE
string	TOKEN_STRINGTYPE
main	TOKEN_MAINFUNC
break	TOKEN_BREAKSTMT
continue	TOKEN_CONTINUESTMT
3 [or other integers]	TOKEN_INTCONST
3.14 [or other floating point numbers]	TOKEN_FLOATCONST
“[some strings]”	TOKEN_STRINGCONST
‘a’ [or other characters]	TOKEN_CHARCONST

موفق باشید.