پروژه اصول طراحی کامپایلر فاز اول

هومان هنرور 993613063

مروارید رهبر 993623020

در این فاز از پروژه برای بررسی لغوی زبان PL از عبارت های منظم استفاده شده است و برای پیاده سازی این عبارت های منظم نیز از ماشین های متناهی استفاده کردیم و در خروجی دو دیاگرام گذار که تقریبا یک مطلب را میرساند رسم کرده ایم. دیاگرام اول برای هر قسمت و هر کلمه از زبان دیاگرام هارا جدا جدا کرده است و در دیاگرام دوم تمام قوانین در یک دیاگرام رسم شده است.

عبارت منظمی که برای این زبان در نظر گرفته شده است، به صورت زیر است:

993613063
993623020

Letter_ → [A-Z-a-z] | _

digit → [0-9] start_digit → [1-9] digits → start_digit digit*

T_decimal → ((- | +)? digits T_Hexadecimal → 0x Hexadecimal_digits

Hexadecimal_digit → [0-9] | [A-Fa-f]

Hexadecimal_digits → Hexadecimal_digit Hexadecimal digit*

T_id → letter_ (letter_ | digit)*

T_whitespace → (blank | tab | newline)*

T_String → " (\")* (anycharachter)* (\")* (anycharachter)* "

T_char → ' anycharachter '

T_Comment → // (anycharacter)* \n

T_Int → int T_Bool → bool T_char → char T_For → for T_If → if

T_Else → else T_Break → break T_Continue → Continue T_True → true

T_False → false T_print → print T_Return → return

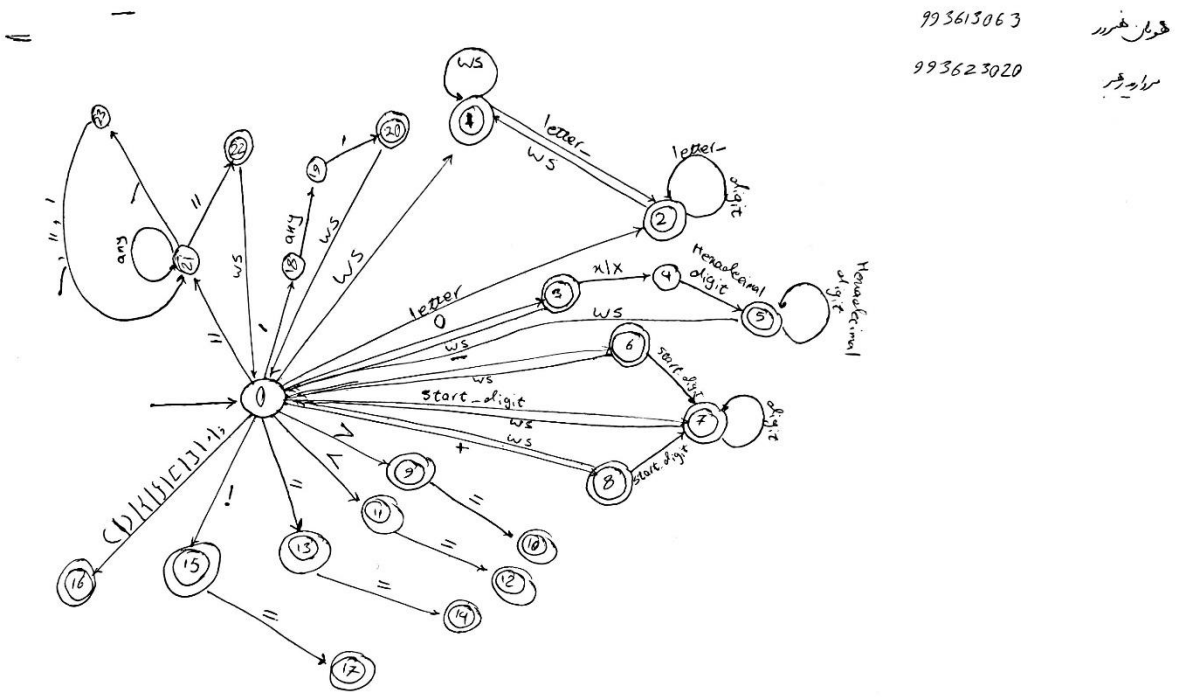
relop → > | < | = | >= | <= | == | != Lop → && | || | !

T_Lp → (T_Rp →) T_LC → { T_RC → } T_LB → [

T_RB →] T_comma → , T_semicolon → ;

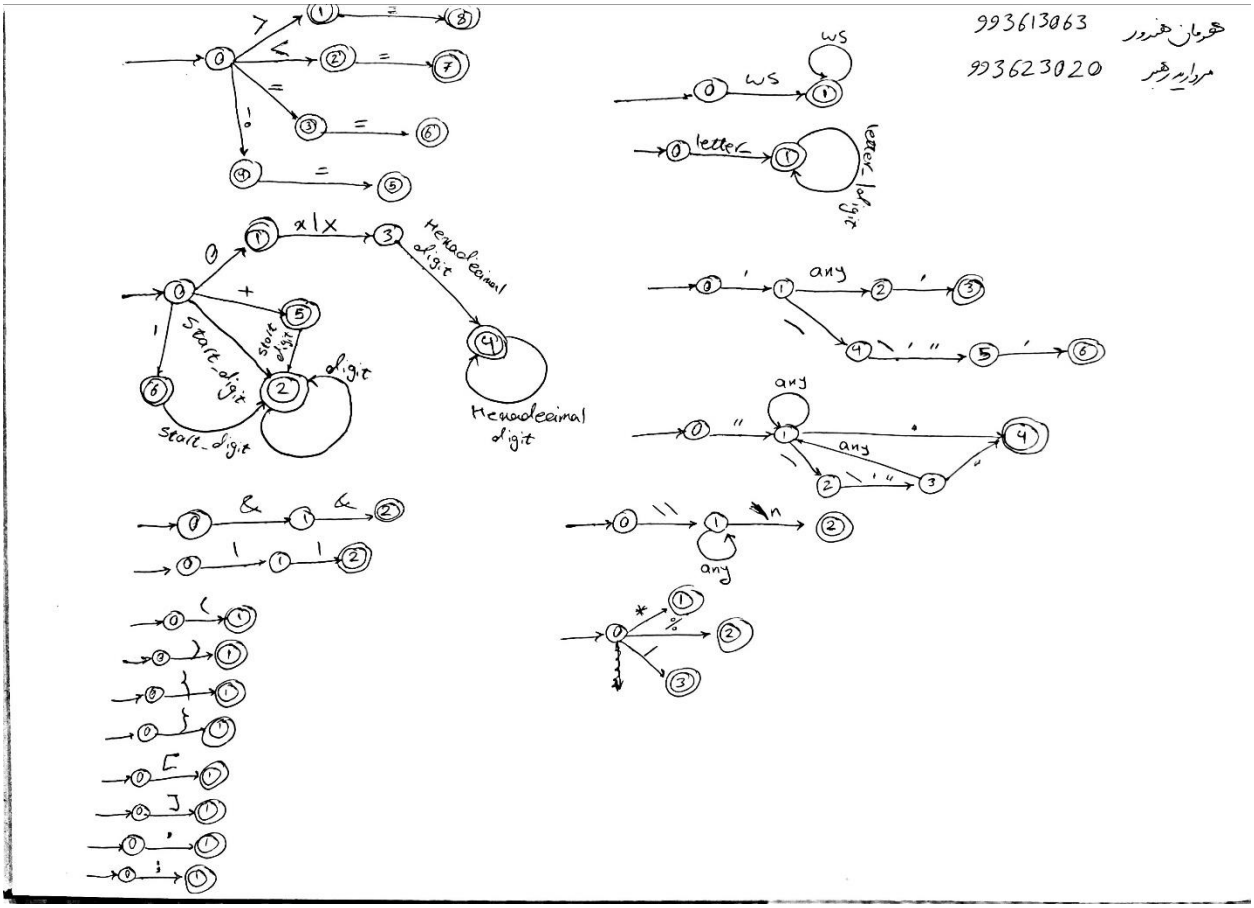
Math_op → + | - | * | % | /

برای این عبارت های منظم دیاگرام 1 به صورت زیر رسم شده است که کل قوانین رو در یک شکل رسم نکرده است:



این دیاگرام فقط یک مشکل دارد آن هم این است که برای جلوگیری از پیچیدگی از تصویر و پیچیدگی بصری از هر حالت به حالت دیگر با کاراکتر خاص نشان داده نشده است و فرض بر این است که بتوان از همه حالت ها با اپسیلون یا لاندا به حالت 0 رفت یا اینکه برای هر حالت به حالت دیگر با تمام کاراکتر های ممکن خط مورد نظر را رسم کرد.

دیاگرام 2 قوانین را جداسازی کرده است و کامل است و قابل استفاده برای پیاده سازی در کد :



توضیح کد:

با توجه به اینکه کامنت گذاری در کد انجام شده است اما بهتر است توضیحاتی در رابطه با کد داده شود. این توضیحات به صورت زیر است:

- main_symbol_table : این یک جدول علائم است که قرار است در فاز بعدی به تحلیل گر نحوی فرستاده شود. البته این جدول علائم تمامی متغییر های حوضه های متفاوت و توابع متفاوت را در یک جدول رسم کرده است که در فاز سوم قرار است که از همدیگر تفکیک شوند. 12 سطر اول این جدول را کلید واژه های اصلی زبان پر کرده اند که با برخورد به متغییر جدید اول 12 سطر اول جدول بررسی میشود و در صورت عدم تطابق با هرکدام از این کلید واژه ها به عنوان یک سطر جدید به جدول اضافه میشد. این جدول حاوی 4 ستون است . word که دقیقا همان چیزی است که در کد بیانگر آن متغییر است، name نامی است که برای توکن آن متغییر در نظر گرفته شده است،

تایپ نوع آن متغییر که میتواند int , bool, char , string باشد و Line شماره خط قرار گرفتن متغییر است.

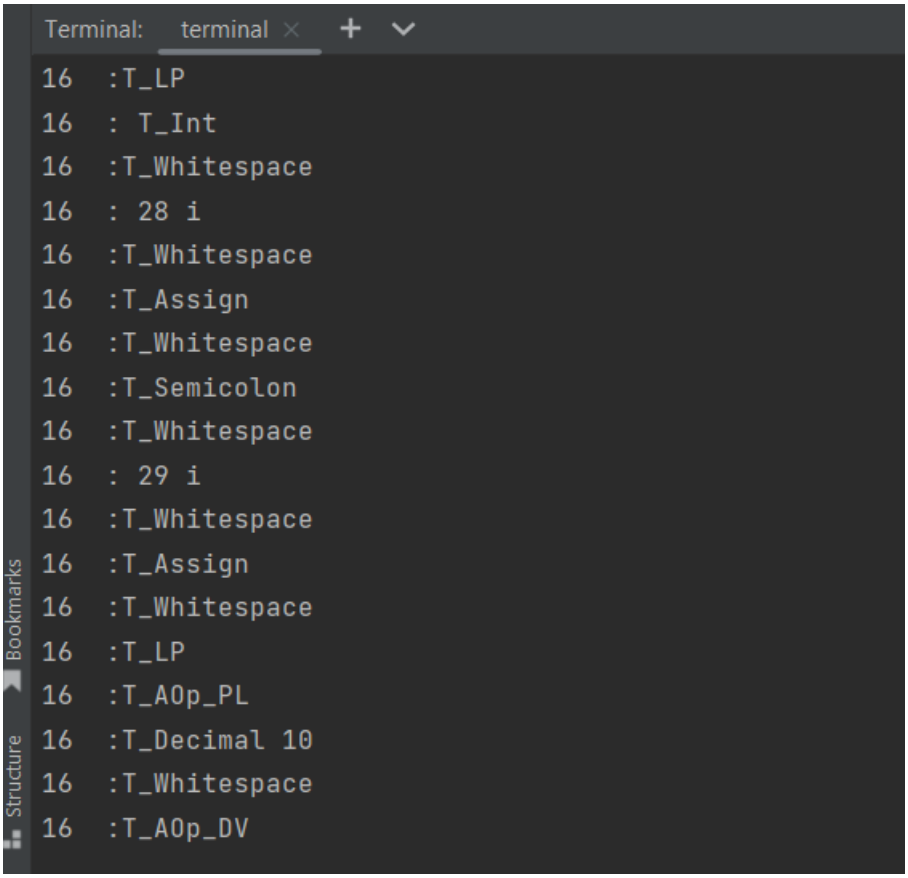
- Buffer : با خواندن خط به خط از ورودی تمامی کارکتر ها در buffer ذخیره میشود و یک buffer_index برای یادآوری شماره کاراکتر این بافر در نظر گرفته میشود. با جلو رفتن کاراکتر به کاراکتر از بافر حالت های دیاگرام گذار نیز طی میشود.
- Switch case : برای پیاده سازی دیاگرام گذار دو راه حل وجود دارد :1- فراخوانی هر یک از توابع در نظر گرفته شده، برای هر کلمه تشخیص داده شده.2- یک دیاگرام کلی و یک الگوریتم یک دست و یکنواخت. در این برنامه از یک الگوریتم کلی و یکنواخت برای تشخیص توکن ها استفاده میشود که با switch case پیاده سازی شده است و متغییر state بیانگر آن حالت های استخراج کننده توکن های متفاوت هستند و state =0 نیز مثل یک لابی عمل میکند و با تشخیص کاراکتر خوانده شده از بافر تشخیص میدهد که باید به کدام حالت فرستاده شود و با استخراج سازی توکن مربوطه از بافر دوباره به لابی برمیگردد که کاراکتر جدید را شناسایی و به حالت مربوطه راهنمایی کند.
- T_id : این یک متغییر موقت است که در حالت هایی مثل رشته و ای دی و اعداد و ... که نیاز است کاراکتر های قبلی در آن ذخیره شود در این متغییر ذخیره شود و با اتمام الگوریتم و شناسایی توکن این متغییر مجدد خالی میشود.

عملکرد:

برای استفاده از این برنامه باید در ترمینال و در پوشه ای که فایل main.py وجود دارد دستور main.py main.c را وارد کنید که main.c نام همان برنامه ای است که قرار است به تحلیل گر لغوی ارسال شود.

خروجی :

خروجی این برنامه به صورت یک لیست از توکن ها است که هرکدام با یک کاراکتر \n جداسازی شده اند و اول شماره خط آنها سپس نام توکن آنها امده است. اگر توکن خروجی جز id یا string باشد که در جدول علائم ذخیره میشود شماره سطری که آنها در جدول علائم ذخیره شده اند را نیز قبل از نام توکن چاپ میکند.



پس از چاپ تمامی توکن ها end را چاپ میکند و سپس جدول علائم را چاپ میکند که به صورت زیر است:

```
43  :T_RC
end
```

	word	name	type	line
0	bool	T_Bool	keyword	0
1	int	T_Int	keyword	0
2	char	T_Char	keyword	0
3	false	T_False	keyword	0
4	true	T_True	keyword	0
5	break	T_Break	keyword	0
6	else	T_Else	keyword	0
7	if	T_If	keyword	0
8	print	T_Print	keyword	0
9	return	T_Return	keyword	0
10	contintue	T_Contintue	keyword	0
11	for	T_For	keyword	0
12	test_function	T_Id	id	1
13	a	T_Id	id	1
14	b	T_Id	id	1
15	c	T_Id	id	1
16	c	T_Id	id	3
17	a	T_Id	id	4
18	b	T_Id	id	4
19	a	T_Id	id	7
20	b	T_Id	id	7