**به نام خدا**

**فاز پایانی پروژه درس کامپایلر**

**استاد : سرکار خانم مهندس راضیه اسکندری**

**تیم همکاری :**

**سجاد مؤمنی**

**حمید نصر اصفهانی**

**عرفان ذکری اصفهانی**

**نیمسال تحصیلی : بهمن 96**

**صورت مسأله**

گرامری برای ما در فایل های دراپ باکس کامپایلر تعریف شده بود که دارای ابهام و برخورد های reduce / reduce و همچنین shift / reduce بود که به همکاری تیم توسعه نرم افزار و با تغییر گرامر یا تغییر اولویت های گرامر تعریف شده برطرف شود و در yacc پیاده سازی شود که اگر جمله ای توی این گرامر تعریف نشده بود پیغام خطای مناسب را بدهد و در غیر این صورت تمام کاهش را نمایش دهد.

**نحوه نصب کردن نرم افزار yacc**

برنامه yacc را با دستورات زیر در linux نصب میکنیم :

sudo apt-get install bison

sudo apt-get install byacc

sudo apt-get install flex

**نحوه کار با yacc**

در اینجا باید یک فایل با فرمت .y ایجاد کنیم که ساختار اطلاعات ورودی ما به شکل زیر میباشد :

%{

اعلان ثابت ها

اعلان متغیر ها

%}

اعلان توکن های موجود در گرامر

%%

قواعد گرامر

%%

C روال های پشتیبان به زبان

فایل yacc از سه قسمت تشکیل شده که عبارتست از

بخش اول :

اعلان متغیر ها و ثابت ها و همچنین توکن ها

بخش دوم :

تعریف قواعد گرامر ها

بخش سوم :

روال های پشتیبانی به زبان C

اگر قانونی به شکل زیر داشته باشیم :

پایانه 3 | پایانه 2 | پایانه 1 <--- غیر پایانه

آن را به صورت زیر پیاده سازی میکنیم :

عملیاتی که باید انجام شود پایانه 3 : غیر پایانه

پایانه 2 | عملیاتی که باید انجام شود

پایانه 1 | عملیاتی که باید انجام شود

;

اگر گرامر به صورت A---> €بود باید به شکل زیر بنویسیم :

A :

اول قوانین گرامر رو به صورت بالا تعریف میکنیم ، سپس برای هر کدام از غیر پایانه ها یک توکن تعریف میکنیم . این توکن ارتباط بین yacc و lex را برقرار میکند . چون زمانی که lex پایانه رو پیدا کند نام مربوط به آن را به yacc بازگشت میدهد .

در کد زیر لیستی از توکن های تعریف شده که به برنامه yacc بازگشت میدهد را مشاهده میکنید :

"print" { return PRINT; }

"in" { return IN; }

"false" { return FALSE; }

"true" { return TRUE; }

"end" { return END; }

"case" { return CASE; }

"write" { return WRITE; }

"read" { return READ; }

"return" { return RETURN; }

"continue" { return CONTINUE; }

"until" { return UNTIL; }

"repeat" { return REPEAT; }

"break" { return BREAK; }

"of" { return OF; }

"switch" { return SWITCH; }

"else" { return ELSE; }

"then" { return THEN; }

"if" { return IF; }

"while" { return WHILE; }

"do" { return DO; }

"down" { return DOWN; }

"to" { return TO; }

"for" { return FOR; }

"int" { return INTEGER; }

"double" { return DOUBLE; }

"char" { return CHARACTER; }

"bool" { return BOOLEAN; }

"constant" { return CONSTANT; }

["].["] { return STRING; }

['].['] { return CHAR; }

"program" { return program; }

در ادامه با استفاده از دستورات زیر از فایل .y که داریم استفاده کرده و دو فایل y.tab.c و y.tab.h را میسازیم :

مرحله اول :

این دستور دو فایل y.tab را میسازد و فایل yacc را اگر خطایی داشته باشد به ما میدهد :

yacc –d yacc.y

مرحله دوم :

این دستور فایل lex را بررسی میکند و اگر دارای اروری باشد آنرا به ما اطلاع میدهد و فایل lex.yy.c رو میسازد :

lex lex.l

مرحله سوم :

با اجرای این دستور با استفاده از دو فایل lex.yy.c و y.tab.c فایل اجرایی برنامه yacc را درست میکنیم که ما آنرا با نام compiler در نظر گرفتیم :

gcc –g lex.yy.c y.tab.v –o compiler

فایل y.tab.h رو به عنوان کتابخانه به برنامه lex اضافه میکنیم ، چون شامل توکن های گرامر و اطلاعات دیگر هست ، در فایل lex هم با استفاده از عبارات منظم چیز هایی رو که میخواهیم تشخیص میدهیم و به اسم همان توکن موجود در فایل yacc بازگشت میدهیم تا بتوانیم آنجا از آن توکن ها استفاده کنیم.

حال فایل اجرایی برنامه رو با دستور زیر اجرا میکنیم ، رشته ورودی از ما دریافت میکند . اگر رشته ورودی قابل پذیرش توسط گرامر تعریف شده باشد تمام کاهش های آن را چاپ میکند و در غیر اینصورت پیغام syntax error میدهد :

./compiler

**توضیح کد**

کدی که در زیر مشاهده میکنید قسمتی است که متغیر هایی که میخواهیم با آنها در قسمت توکن و اکشن ها استفاده کنیم را تعریف میکنیم :

%union {

int num;

char \*id;

char \*str;

double doub;

}

با کد زیر غیر پایانه ای که گرامر ما با آن شروع میشود را معرفی میکنیم :

%start Program

اگر lex یک IntNumber را تشخیص دهد با استفاده از خط کد زیر آن را به متغیر تعریف شده به صورت توکن در yacc ارسال میکند :

Lex.l : yylval.num = atoi(yytext); return IntNumber;

Yacc.y : %token <num> IntNumber

با دستور زیر نیز مابقی توکن ها را تعریف میکنیم ، برای مثال :

%token FOR THEN DOWN TO DO WHILE SWITCH CASE OF