الجلسة التاسعة

التعامل مع أنواع مختلفة للمتحول yylval (بداية التحليل المعنوي)

Dealing with multiple types of yylval

الهدف من الجلسة

- التعرف على كيفية التعامل مع أنواع مختلفة للمتحول yylval

مستلزمات الجلسة

- حاسب بمواصفات دنيا RAM: 1 GB, CPU: 1.6 GHz, Windows 10 OS 64 bit
- Turbo c++/ DevC++
- LEX & BISON tools

خطوات العمل

- بناء ملفي وصف ماسح ومعرب وتعلم كيفية تعديل بنية الملفات للتعامل مع أنواع مختلفة للمتحول yylval

الخلاصة والنتائج:

يفترض عند نهاية الجلسة:

- تمكن الطالب من معرفة كتابة ملف وصف الماسح وملف وصف لبرنامج مصدري مؤلف من قسمين تصريح وإسناد مع إضافة للأنواع المختلفة للمتحول yylval.

1.1 كيفية التعامل مع أنواع مختلفة للمتغير yylval

بفرض أننا نريد بناء مترجم للعبارة التالية المؤلف من قسمين تصريح وإسناد:

```
int x;
char c;
chain h;
real r;
c=”a”;
h=”computer engineer”
r=0.05;
```

نلاحظ أن هناك أنواعاً محتلفة من المتغيرات int,char,string ، النوع الصحيح int هو النوع الذي تعلمنا كيفية التعامل معه أما الأرقام من النوع real أو المحارف أو السلاسل فهي شيء جديد.

يجب أن يكون المتغير yylval متكيفاً بحسب نوع القيمة التي يعيدها والتي يمكن أن تكون قيمة عددية (صحيحة أو حقيقية)، أو قيمة حرفية، وهنا لا بد من إعادة التصريح عن المتغير yylval بشكل آخر لذا فإننا بحاجة إلى ما يسمى بـ union، لن نغير نوع yylval من int إلى char أو غيره بل سنمرر لـ YACC عدة أنواع. إذاً الحل هو باستخدام union وهي شبيهة بالسجلات في لغة الـ C وَ تشغل مساحة محددة من الذاكرة على سبيل المثال:

```
typedef union{
char Tstr[20];
int Tint;
float Treal;
} new_type;
new_type t;
```

هنا يتم تعريف union اسمه new_type ويحتوي سلسلة من النوع char حجمها 20 محرف، وحقل من النوع الصحيح int، وحقل من النوع الحقيقي float، ثم يتم اشتقاق متغير t من النوع new_type الذي عرفناه وهذا يعني أن المتغير t هو بمثابة سجل له ثلاثة حقول أحدها صحيح والثاني حقيقي والثالث سلسلة. مثلاً: للوصول إلى الحقل ذي النوع float من المتغير t نكتب : t.Treal.

ملاحظة
إذا قمنا باسناد قيمة لـ t.Tstr ثم أسندنا قيمة لـ t.Tint فإن القيمة الأولى ستضيع لأن المتغيرات الثلاثة تتشارك في الذاكرة المحجوزة لها وهي الذاكرة اللازمة لتخزين أكبر متغير وهنا تساوي 128 بايت (وهذا يختلف عن السجلات struct).

لكن كيف سنعبر عن ذلك في ملف المعرب:

فيما يلي ملفا وصف الـ Scanner وَ الـ Parser.

1.1.1 1-3-7 ملف وصف المعرب Parser:

لجعل المتغير yylval ينتمي للنوعين (الرقم والحرف) نعرف union يتضمن حقلاً خاصاً بالأرقام وآخر للأحرف والثالث لأرقام الفاصلة العائمة

وبهذا نكون قد غيرنا نوع yylval إلى النوع الذي نحتاجه وبقي لدينا أن نمرر اسم المتغير إلى YACC كلما وجدناه وهو على مستوى ملف المعرب Parser.

```
%{
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include "decl.c"
int errors=0;
%}
%union{
char Tstr[10];
int Tint;
float Treal;
}
%token DEC_INT DEC_CHAR DEC_REAL DEC_CHAIN EQ SEMICOLON
%token ID
%token <Tint> NUM
%token < Treal > REAL
%token <Tstr> CHAR
%token <Tstr> CHAIN
%right '='
%start st
%%
st:declaration expression;
declaration:
|decl_type ident SEMICOLON declaration;
decl_type:DEC_INT
```

```
|DEC_REAL
|DEC_CHAR
|DEC_CHAIN
|error;
ident:ID| ID ident |error;
expression:
|ID EQ ID SEMICOLON  expression
|ID EQ CHAIN SEMICOLON expression
|ID EQ CHAR SEMICOLON expression
|ID EQ NUM SEMICOLON expression
|ID EQ REAL SEMICOLON expression
|error;
%%
int yyerror (char *s)
{
errors++;
printf("Error:: %d %s at line:: %d in statement:: %s\n",errors,s,line,yytext);}
int yywrap(){
return 1;
}
main()
{
clrscr();
if((yyin=fopen("input.txt","r"))==NULL)
{
printf("input.txt not found !\n");
return;
}
yyparse();
if(!errors) printf("Ok");
return;
}
```

1.1.2 2-3-7 ملف وصف الـ Scanner:

حتى يتمكن الـ Scanner من تمييز الأحرف الكبيرة والصغيرة واعتبارها نفس المحرف، يجب تعريف قوالب الأحرف بحيث تأخذ بعين الاعتبار الأحرف الكبيرة والصغيرة. مثلاً المحرف a يمكن أن يكون حرفاً صغيراً a أو كبيراً A. كمافي القاعدة التالية:

a [aA]

b [bB]

والسؤال الذي يطرح نفسه الآن كيف يمكن أن يميز الماسح بين الأنواع المختلفة لرموز الدخل بحيث يستجيب للرموز ذات النوع الصحيح بشكل مختلف عن الأخرى ذات النوع الحقيقي مثلاً.

والجواب:

عند التعرف على رقم digit يتم إعادة الـ token المسمى NUM كما يتم إعادة قيمة هذا الرقم في الحقل Tint في السجل yylval كما في التعليمات التالية:

```
{digit}+ {
yylval.Tint=atoi(yytext);
return(NUM);
}
```

لاحظ أننا استخدمنا الحقل Tint فقط من السجل yylval وبالاعتماد على الدالة atoi قمنا بأخذ القيمة الصحيحة للرقم digit.

وعند التعرف على رقم rdigit وهو عدد حقيقي، يتم إعادة الـ token المسمى REAL كما يتم إعادة قيمة هذا الرقم في الحقل Treal في السجل yylval . نستخدم التابع atof من المكتبة stdlib.h لتحويل السلسة المخزنة في yytext إلى رقم حقيقي.

```
{rdigit} {
yylval.Treal=atof(yytext);
return(REAL);
}
```

لاحظ أننا استخدمنا الحقل Treal وذلك لأن الرمز الذي تم التعرف عليه هو من النوع الحقيقي وليس من النوع الصحيح.

أما عند التعرف على رقم معرف ident فيتم إعادة الـ token المسمى ID كما يتم إعادة السلسلة الحرفية التي تمثل هذا المعرف في الحقل Tstr في السجل yylval

التابع strcpy ينسخ السلسلة الموجودة في المصفوفة yytext إلى الحقل Tstrهذا التابع يحتاج لتضمين المكتبة string.h.

```
{ident} {
strcpy(yylval.Tstr,yytext);
```

```
return(ID);
}
```

والآن وبعد معرفة كل الحالات الجديدة في ملف الماسح، لنكتب ملف وصف الماسح:

```
%{
#include<stdlib.h>
#include<string.h>
#include "dec.h"
int line=1;
%}
alpha [a-zA-Z]
digit [0-9]+
blanc [ \t]+
rdigit {digit}\.{digit}
ident {alpha}({alpha}|{digit})*
a [aA]
b [bB]
c [cC]
d [dD]
e [eE]
f [fF]
g [gG]
h [hH]
i [iI]
j [jJ]
k [kK]
l [lL]
m [mM]
n [nN]
o [oO]
p [pP]
q [qQ]
r [rR]
```

```
s [sS]
t [tT]
u [uU]
v [vV]
w [wW]
x [xX]
y [yY]
z [zZ]
%%
{blanc}
\n { line++; }
{i}{n}{t} return(DEC_INT);
{r}{e}{a}{l} return(DEC_REAL);
{c}{h}{a}{r} return(DEC_CHAR);
{c}{h}{a}{i}{n} return(DEC_CHAIN);
"=" return EQ;
"\"."\"" return(CHAR);
"\""(.)+"\"" return(CHAIN);
";" return(SEMICOLON);
{digit}+ {
yylval.Tint=atoi(yytext);
return(NUM);
}
{rdigit} {
yylval.Treal=atof(yytext);
return(REAL);
}
{ident} {
strcpy(yylval.Tstr,yytext);
return(ID);
}
%%
```

بالتالي عندما يصادف الـ Scanner رقماً أو سلسلة حرفية فهو بكلتا الحالتين قادر على إعادة قيم الرقم ومحارف السلسلة إلى المعرب الذي سيحتاجها لاحقاً في مرحلة بناء جدول الرموز والتحليل المعنوي وهو موضوع المحاضرة القادمة.

ملاحظة:

العبارة "\".\"" تعني محرفاً واحداً حيث إن النقطة "." تشير إلى أي محرف من لوحة المفاتيح والتوجيه "\ يعني وجود إشارتي تنصيص من اليسار أما التوجيه "\ فيعني وجود إشارتي تنصيص من اليمين، مثلاً "a". أما العبارة: "\"+(.)\"" فهي تعني محرفاً أو أكثر أي تدل على سلسلة كاملة. لاحظ وجود إشارة + التي تعني نسخة أو أكثر مما ضمن قوسين أي محرف أو أكثر.

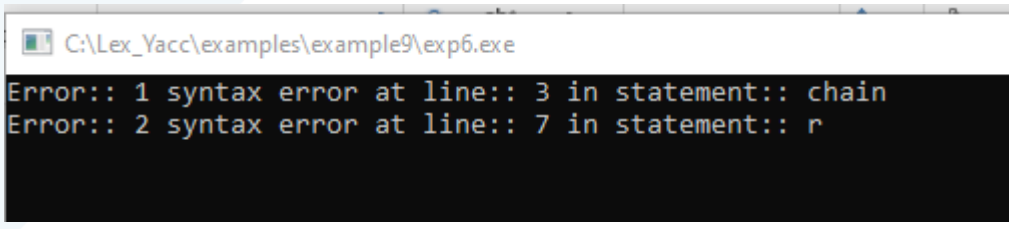
والآن نختبر المترجم على العبارة:

```
real x;
char c;
chain h;
x=22.22;
c="a";
h="computer engineer";
```

والنتيجة هي أنه سيطبع لنا عبارة Ok أي أنه لا توجد أخطاء.

لنجري بعض الأخطاء ونختبر قدرة المترجم على اكتشافها:

```
int x;
char c
chain h;
real r;
c="a";
h="computer engineer"
r=0.05;
```



مثال آخر:

int x

char c;

chain h;

r real;

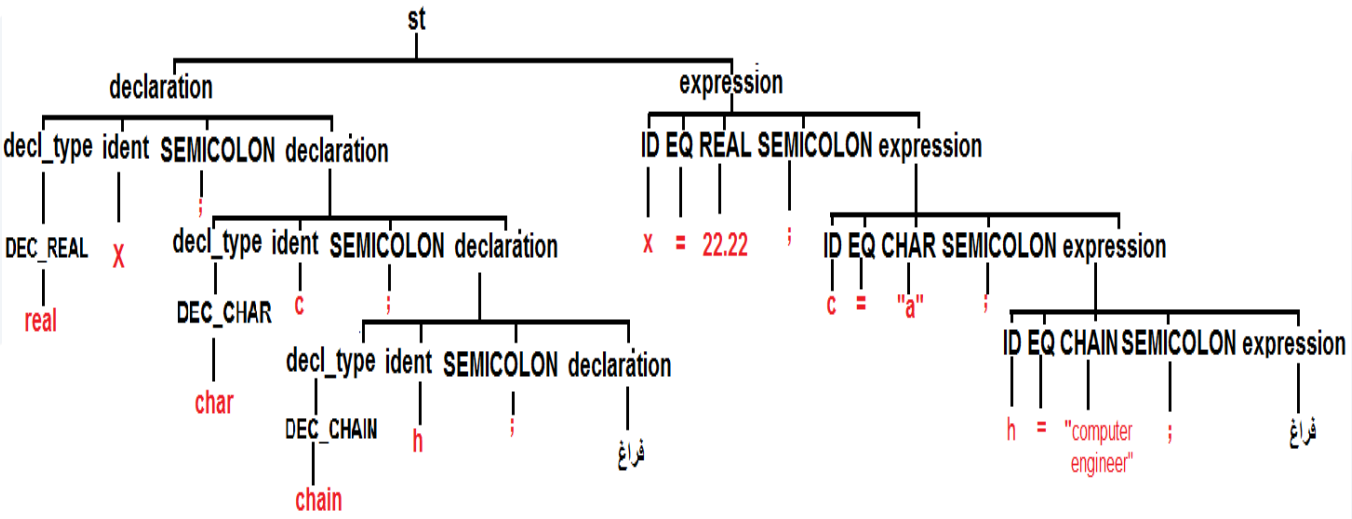
c="a";

h="computer engineer";

r=0.05;

```
C:\Lex_Yacc\examples\example9\exp6.exe
Error:: 1 syntax error at line:: 2 in statement:: char
Error:: 2 syntax error at line:: 4 in statement:: real
```

لنرسم شجرة الإعراب للعبارة السابقة:



الشكل (3-7) شجرة الإعراب للعبارة المصدرية السابقة

انتهت الجلسة – د. علي ميا ، م. رشا شباني