# الجلسة الخامسة

## بناء مترجم لعبارات الشرط

## Compiler of If-then-else

### الهدف من الجلسة

- التعرف على كيفية بناء مترجم متكامل لعبارات الشرط.

### 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 مترجم خاص بالشرط IF THEN ELSE
- بناء ملف وصف الماسح لعبارات الشرط
- كتابة ملف وصف المعرب لعبارات الشرط
- قواعد اللغة لعبارات الشرط
- كود ملف وصف المعرب لعبارات الشرط

### الخلاصة والنتائج:

يفترض عند نهاية الجلسة:

- تمكن الطالب من معرفة كتابة ملف وصف الماسح وملف وصف المعرب لمترجم يترجم عبارات الشرط والتفرع.
- تطبيق مثال عملي على الأجهزة.

### 1.1 1-5 بناء مترجم خاص بالشرط IF THEN ELSE:

بفرض أننا نريد كتابة مترجم خاص بتعليمة الشرط IF THEN ELSE، بحيث نقوم ببناء مترجم لعبارة من الشكل:

if (a==1) then b=1; else b=2;

أي سيقوم المترجم وفقاً لبرنامجنا من التحقق من العبارة المدخلة ليتأكد إن كانت صحيحة لفظياً وقواعدياً، وسيقوم بطباعة العبارة التالية:

Input accepted.

أما إن كانت عبارة الدخل غير متطابقة مع النحو المستخدم من قبل المترجم، سيطبع لنا المترجم على الخرج العبارة التالية:

Parse error.

### 1.2 2-5 بناء ملف وصف الماسح:

بداية وكما تعلمنا في المحاضرات السابقة، سنقوم بكتابة الملف الذي يمثل وصف الـ Scanner وهنا نحدد القوالب التي سنستخدمها للتعرف على الـ tokens. والقوالب التي سنستخدمها هي:

blank [\t\n]+ قالب الفراغات والأسطر.

alpha [A–Za–z] حرف أبجدي (كبير أو صغير).

digit [0–9] رقم من 0 إلى 9.

وفي قسم القواعد التي سيعمل على أساسها برنامج الـ Scanner بمجرد تعرفه على الـ Tokens نكتب مجموعة من القواعد:

حيث تمثل blank الفراغات البيضاء أو whitespace والتي سيتجاهلها الـ Scanner وفقاً للقاعدة الأولى، سيقوم الـ Scanner كذلك بإعادة الـ token المسمى IF لدى مصادفته للعبارة if في الدخل وفقاً للقاعدة if return IF، كما سيقوم الـ Scanner كذلك بإعادة الـ token المسمى THEN لدى مصادفته للعبارة then في الدخل وفقاً للقاعدة then return THEN ، كما سيقوم الـ Scanner كذلك بإعادة الـ token المسمى ELSE لدى مصادفته للعبارة else وفقاً للقاعدة else return ELSE. أما وفقاً للقاعدة {digit}+ return NUM فإن الـ Scanner سيعيد الـ token المسماة NUM ليشير لوجود عدد مكون من رقم أو أكثر في تسلسل الدخل. بينما سيعيد الـ Scanner الـ token المسماة ID عند مصادفته معرف مثل A أو a أو A1 أو a1 أو A12 وهكذا.

وفيمايلي ملف الـ LEX الذي يتضمن وصف الـ Scanner كمايلي:

```
%{

#include <stdlib.h>
```

```
#include"y.tab.h"
%}
blank [ \t]+
alpha [A-Za-z]
digit [0-9]
%%
{blank}
if    return IF;
then    return THEN;
else    return ELSE;
{digit}+   {
yylval=atoi(yytext);
 return NUM;}
{alpha}({alpha}|{digit})* return ID;
"("    return LPAR;
")"    return RPAR;
";"    return SEMI;
"<"    return LT;
">"    return GT;
"<="    return LE;
">="    return GE;
"=="    return EQ;
```

تضمين الملف الرأسي الذي يتضمن تعريفات التوكين والتصريح الكامل للمتحول yylval

قالب الفراغات blank

قالب الأحرف وقالب الأرقام

الاستجابة للقوالب

الاستجابة لمصادفة الماسح لرقم في العبارة المصدرية

1- يسحب قيمة الرقم ويضعها في المتحول yylval

2- يعيد التوكين NUM

الاستجابة لقالب المعرفات (المتحولات)

```
"!="    return NE;

"||"    return OR;

"&&"    return AND;

"+"   return PLUS;

"*"   return MULT;

"–"    return MINUS;

"/"    return DIVS;

"="   return EQUAL ;

%%
```

## 1.3 3-5 كتابة ملف وصف المعرب:

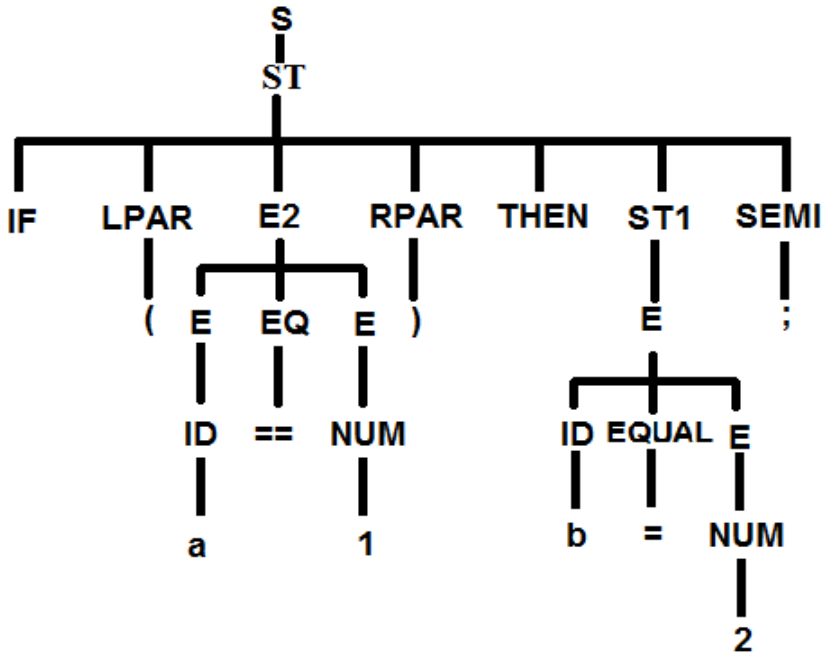
### 1.3.1 1-3-5 قواعد اللغة:

فيمايلي القواعد اللازمة لترجمة عبارة مثل if (a==1) then b=1; else b=2;

```
S      : ST
ST     : IF LPAR E2 RPAR THEN ST1 SEMI ELSE ST1 SEMI
       | IF LPAR E2 RPAR THEN ST1 SEMI;
ST1   : ST| E;
E     : ID EQUAL E | E PLUS E | E MINUS E | MINUS E | E MULT E | E DIVS E | E LT E |
E GT E | E LE E | E GE E | E EQ E | E NE E | E OR E | E AND E | ID | NUM;
E2   : E LT E | E GT E | E LE E | E GE E | E EQ E | E NE E | E OR E | E AND E | ID |
NUM;
```

باستخدام القواعد السابقة سنقوم ببناء شجرة الإعراب للعبارة:

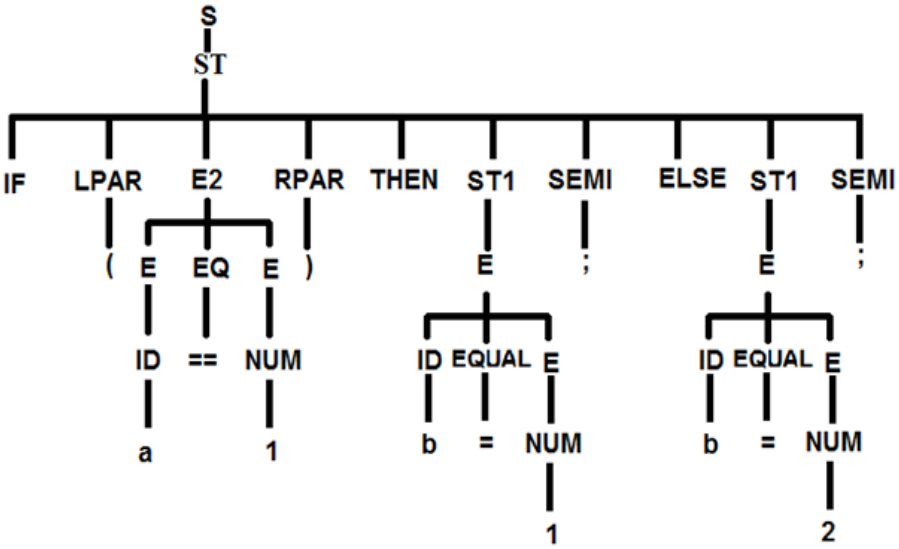
if (a==1) then b=1;




**الشكل (5-1) شجرة الإعراب للعبارة if (a==1) then b=1;**

من جديد، باستخدام القواعد السابقة سنقوم ببناء شجرة الإعراب للعبارة:

if (a==1) then b=1; else b=2;




**الشكل (5-2) شجرة الإعراب للعبارة if (a==1) then b=1;**

### 1.3.2 2-3-5 كود ملف وصف المعرب:

```
%{
#include <stdio.h>
#include<conio.h>
#include <stdlib.h>
#include"lex.yy.c"
%}
%token ID NUM IF THEN LE GE EQ NE OR AND ELSE EQUAL LT GT PLUS MINUS MULT
DIVS LPAR RPAR SEMI
%right EQUAL
%left AND OR
%left LT GT LE GE EQ NE
%left PLUS MINUS
%left MULT DIVS
%%
S      : ST {printf("Input accepted.\n");};
ST     : IF LPAR E2 RPAR THEN ST1 SEMI ELSE ST1 SEMI
       | IF LPAR E2 RPAR THEN ST1 SEMI
       ;
ST1    : ST
       | E
       ;
E      : ID EQUAL E
       | E PLUS E
       | E MINUS E
       | MINUS E
       | E MULT E
       | E DIVS E
       | E LT E
       | E GT E
```

تضمين ملف الماسح ضمن ملف وصف المعرب

التصريح عن قائمة التوكين

التصريح عن أولويات تنفيذ العمليات المعنوية في شجرة الإعراب

الجزء الخاص بكتابة قواعد

عند العودة لرمز بداية الإعراب يتم طباعة رسالة تم قبول الدخل

ننهي كل قاعدة بفاصلة منقوطة

```
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
E2  : E LT E
      | E GT E
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
%%
int yywrap()
{
return 1;
}
int yyerror (char *s)
{
printf(s);
}
main()
```

قسم التوابع

```
{
if((yyin=fopen("input.txt","r"))==NULL)
{
printf("input.txt not found !\n");
return;
}
  yyparse();
  getchar();
}
```

في هذا البرنامج سيتم قراءة البرنامج المصدري من ملف نصي اسمه input.txt

في حال كان الملف غير موجود أو تعذر فتحه يتم طباعة رسالة تفيد بأن الملف غير موجود

في حال العثور على الملف يتم بدء الإعراب باستدعاء المعرب yyparse والي يستدعي الماسح yylex ضمنياً كون أننا ضمنا

بعد التصريحات سنقوم بتحديد الأولويات باستخدام %left وَ%right، حيث يمثل السطر الأول (من قسم الأولويات) العمليات الأقل أولوية والأخير الأولوية الأعلى. مثلاً:

'=' %right تعني أن الأولوية الأدنى هي للإسناد ويتم حسابها من اليمين (إسناد الطرف الأيمن إلى الأيسر).

ثم يأتي قسم القواعد، نضع قواعد النحو التي ناقشناها بداية الفقرة. لا ننسى أن نضع جميع الأفعال المعنوية semantic actions التي نرغب أن يقوم المترجم بإنجازها عند تحقق كل قاعدة. في هذا المثال نحن بحاجة لعملية واحدة فقط هي:

```
{printf("Input accepted.\n");exit(0);}
```

وهذه التعليمة سنضعها بعد القاعدة S: ST حيث تقوم بطباعة العبارة .Input accepted إذا تمت ترجمة تسلسل الدخل بنجاح. حيث إنه عندما يصل المترجم لهذه القاعدة سيكون قد وصل من أطراف شجرة الإعراب إلى رمز الدخل S وهذا يعني أن العبارة المصدرية قد تم ترجمتها بنجاح.

**ملاحظة هامة:**

**يتم تضمين الملف الرأسي stdio.h من أجل التابع ()printf، بينما يتم تضمين الملف الرأسي coino.h من أجل التابع () getchar ، و الملف الرأسي stdlib.h من أجل التابع fopen.**

يعيد التابع ;()yyparse القيمة 0 إذا تم إعراب الدخل بنجاح أي إذا كانت سلسلة الدخل تنطبق مع القواعد المستخدمة من قبل المعرب parser ويعيد 1 إذا فشلت عملية الإعراب.

والآن نتبع خطوات المحاضرة السابقة نفسها للحصول على الملف التنفيذي النهائي بعد ترجمة ملفي الـ Scanner والـ Parser باستخدام مترجم GC++ للحصول على الملف ify.exe. يمكننا الآن اختبار المترجم بالنقر المزدوج على هذا الملف وكتابة العبارة:

```
if (a==1)  then b=1; else b=2;
```

في ملف المدخلات input.txt وسيطبع لنا المترجم العبارة التالية:

.Input accepted

**التنفيذ العملي:**

قم بتكرار تنفيذ نفس خطوات بناء المترجم في المحاضرة السابقة.

قم بإنشاء ملف input.txt في نفس مسار الجلسة

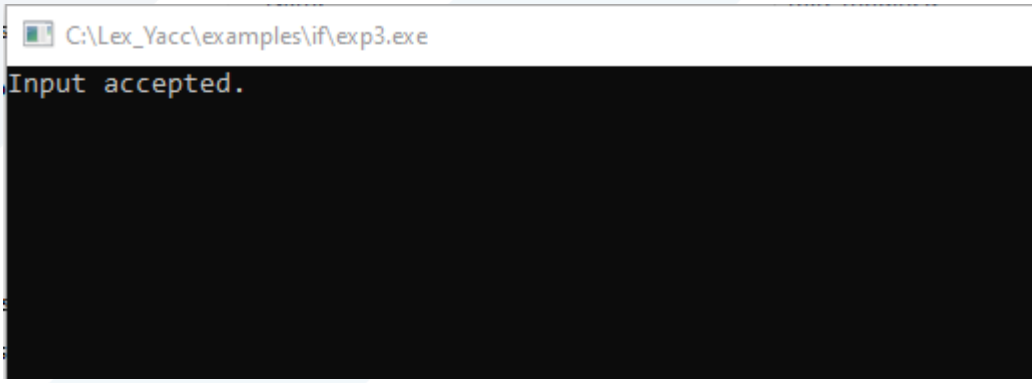
اكتب فيه تعليمات البرنامج المصدري مثلاً:

```
if (a==1)  then b=1; else b=2;
```

والآن قم بتنفيذ المترجم بالضغط على الملف التنفيذي double click

ستحصل على الرسالة

Input accepted




الآن عدل ملف input.txt ليصبح:

```
if(a<5))

then b=1;

else b=*2;
```

وأعد تنفيذ المترجم:

C:\Lex_Yacc\examples\if\exp3.exe

```
syntax error
```

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