Sesión 05: **Instalación y reconocimiento del entorno Bison**

I. OBJETIVOS

- Instalar el software bison
- Utilizar las funcionalidades básicas de bison
- Crear una aplicación simple

II. TEMAS A TRATAR

- Bison

III. MARCO TEORICO

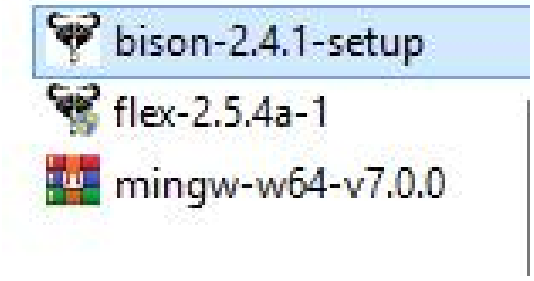
Bison

Bison es un generador de analizador de propósito general que convierte una gramática libre de contexto anotada en un analizador determinista LR o LR generalizado (GLR) que emplea tablas de analizador LALR (1). Como característica experimental, Bison también puede generar tablas analizadoras IELR (1) o LR (1) canónicas. Una vez que domine Bison, puede usarlo para desarrollar una amplia gama de analizadores de idiomas, desde los utilizados en calculadoras de escritorio simples hasta lenguajes de programación complejos.

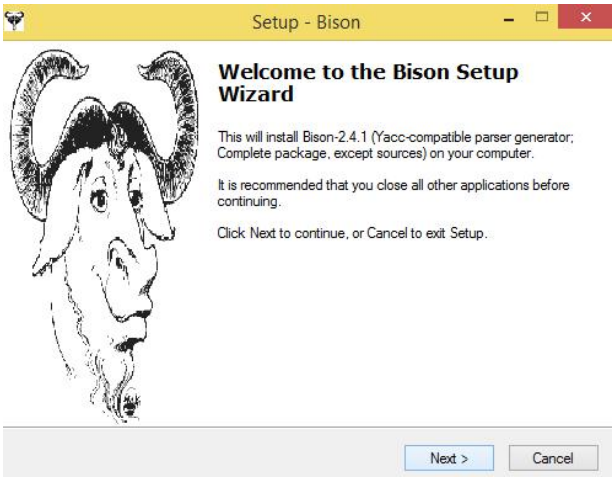
Bison es compatible con Yacc: todas las gramáticas de Yacc correctamente escritas deberían funcionar con Bison sin cambios. Cualquier persona familiarizada con Yacc debería poder usar Bison con pocos problemas. Debes tener fluidez en la programación en C o C ++ para usar Bison. Java también es compatible como una característica experimental.

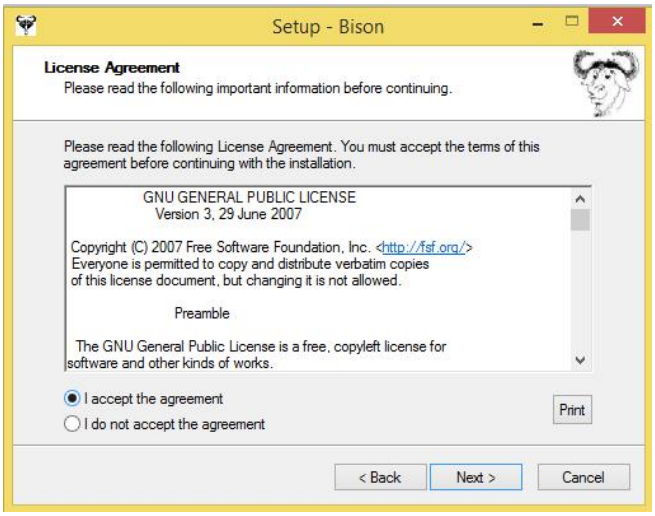
IV. ACTIVIDADES (La práctica tiene una duración de 2 horas)

1. Bison

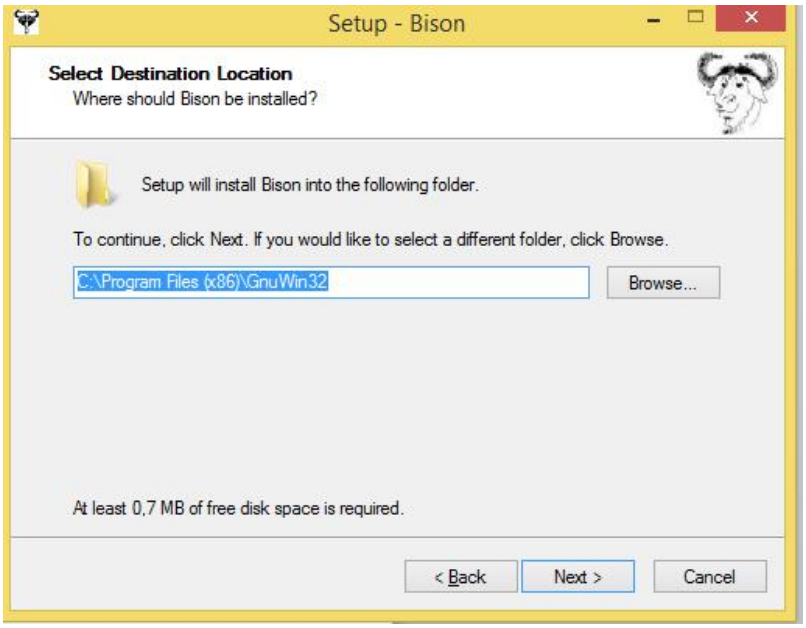


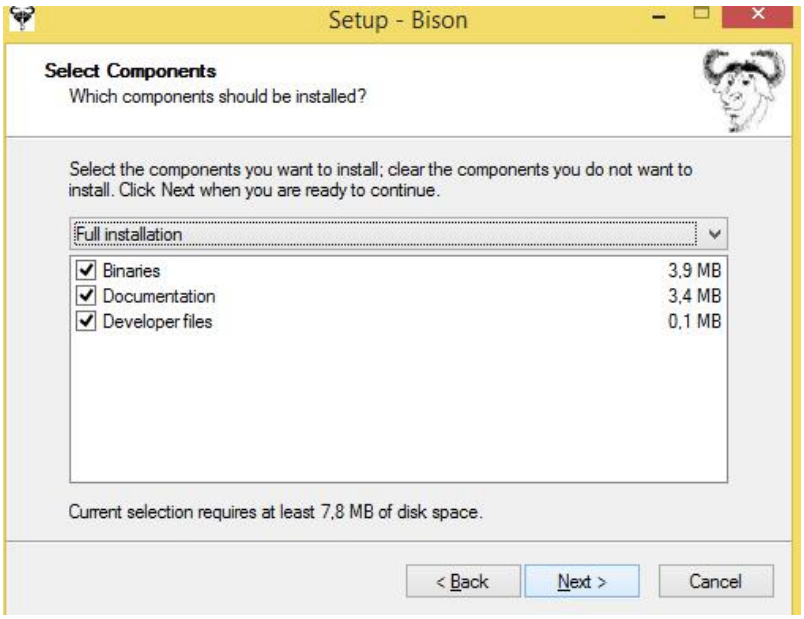
Instalar Bison:

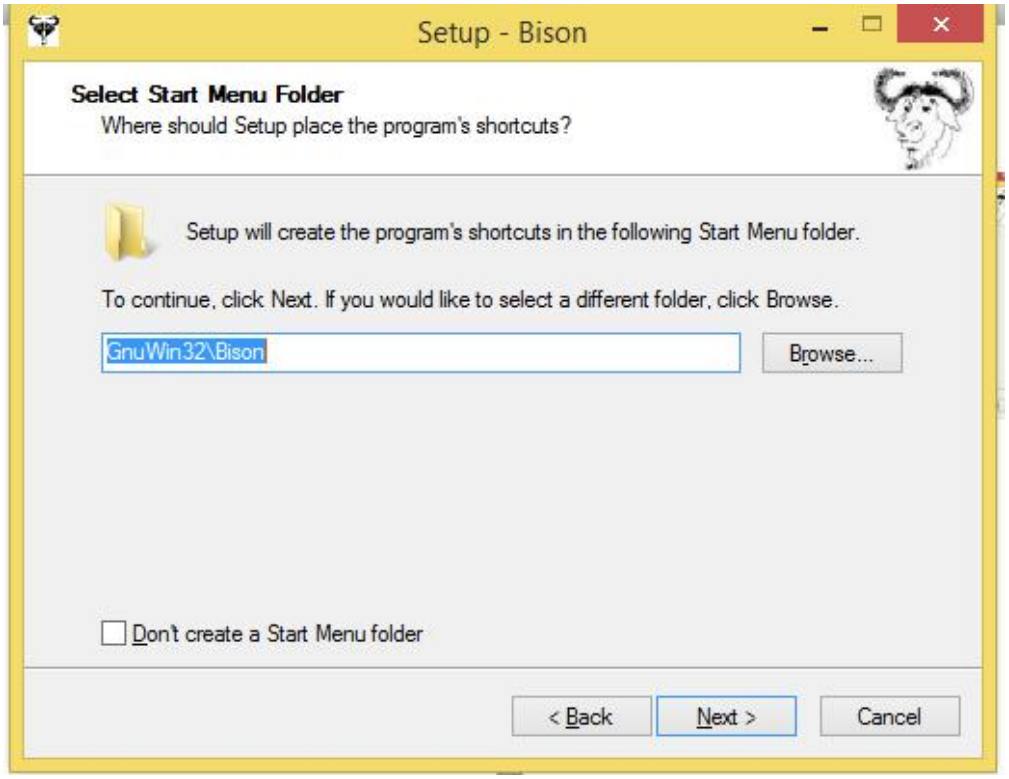




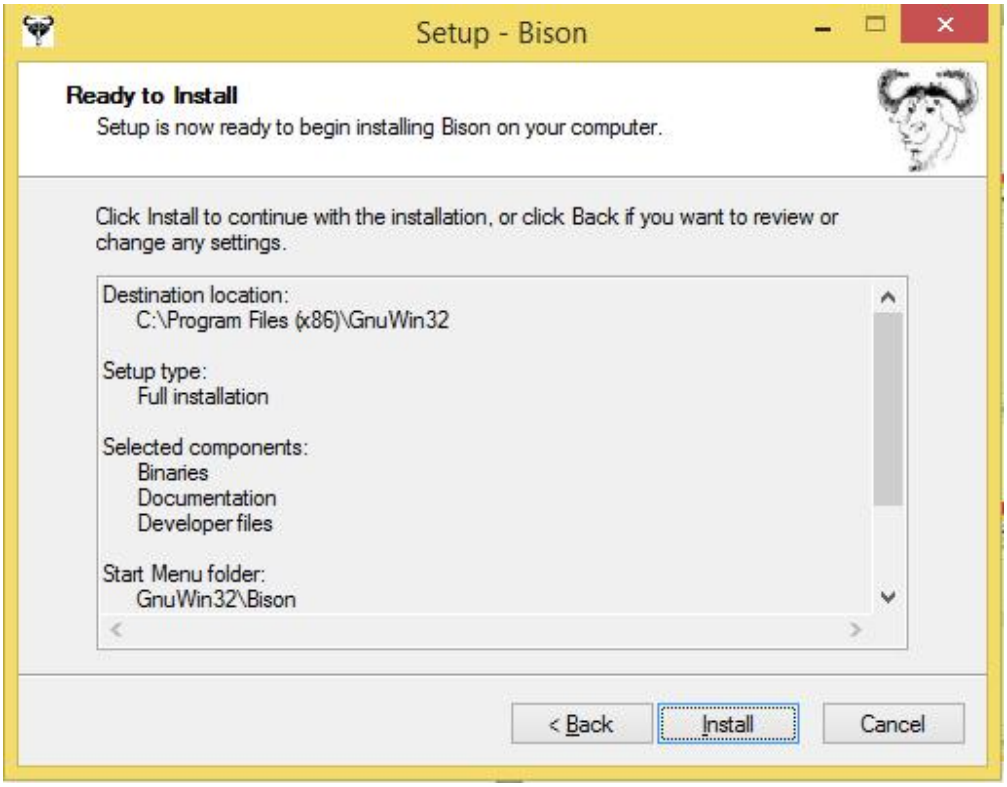
Copiar la dirección donde instalamos: C:\Program Files (x86)\GnuWin32 Esta dirección la agregaremos al PATH luego de la instalación





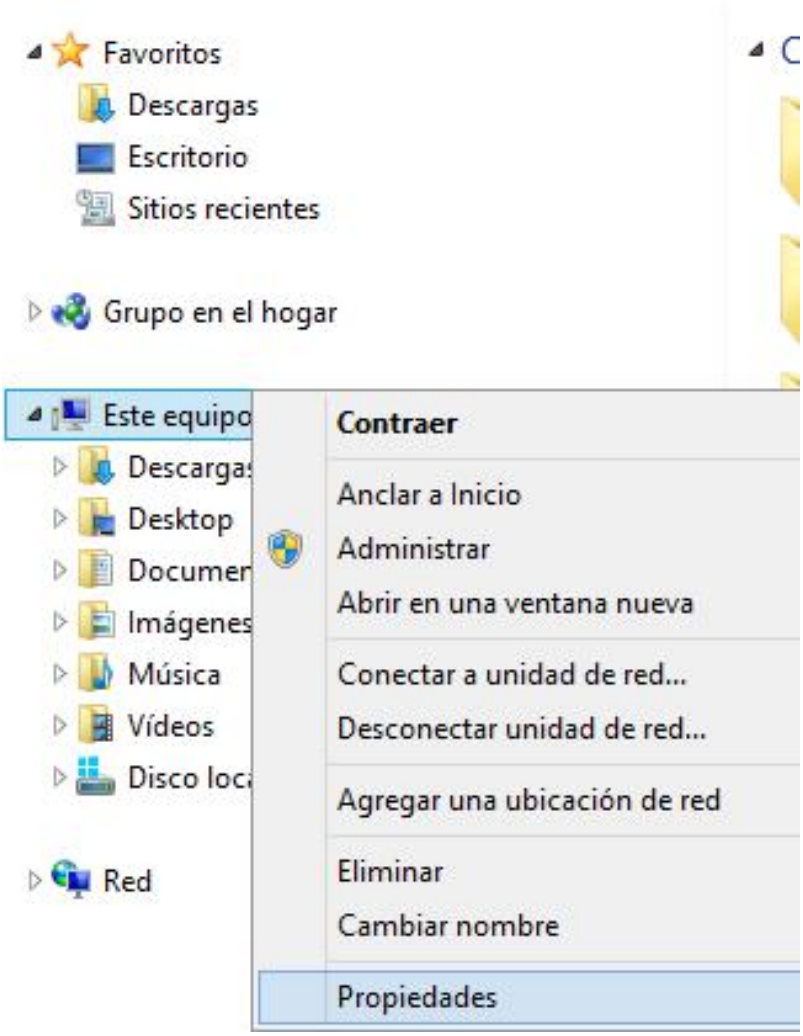


Hacer click en Next y finalmente en Install

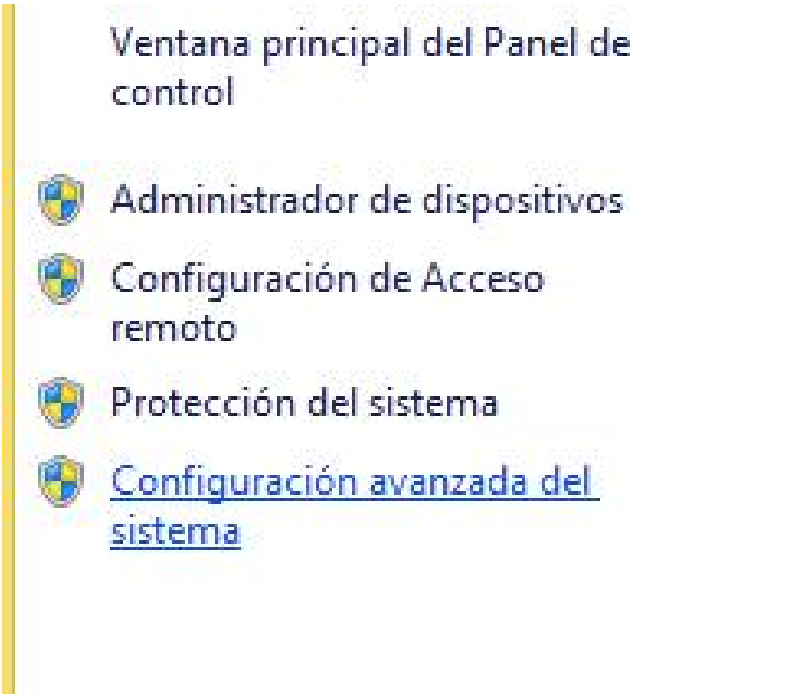


2. Configurar Bison

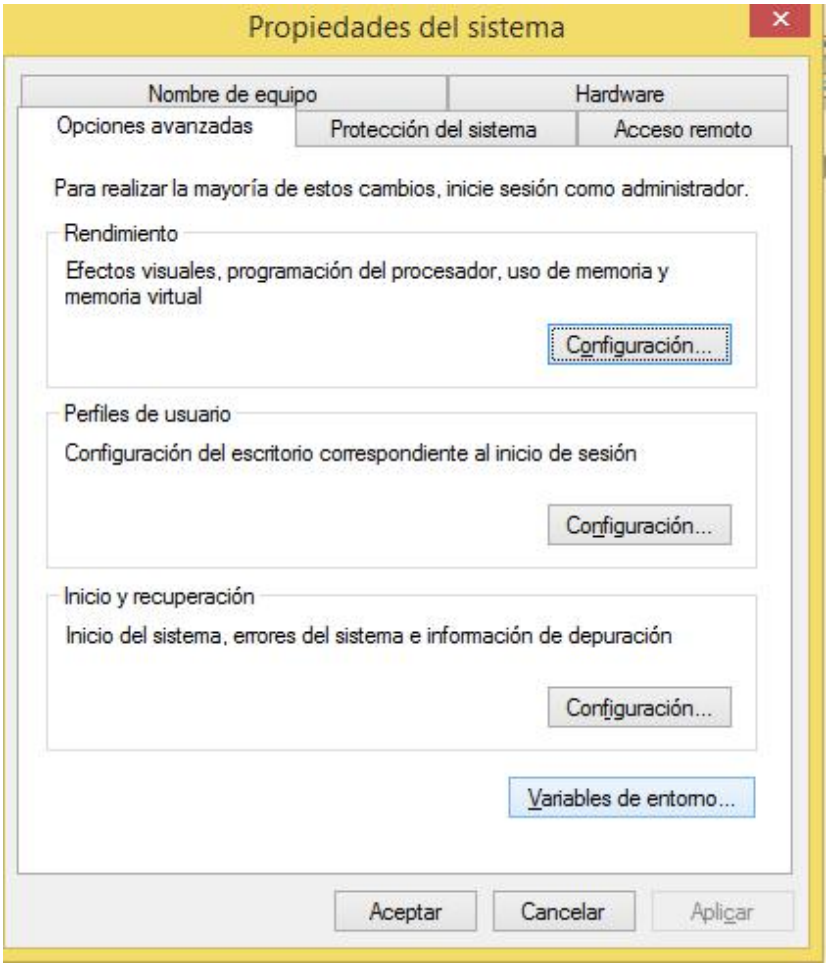
Ahora vamos a configurar las variables del entorno. Para eso vamos a las propiedades del equipo:



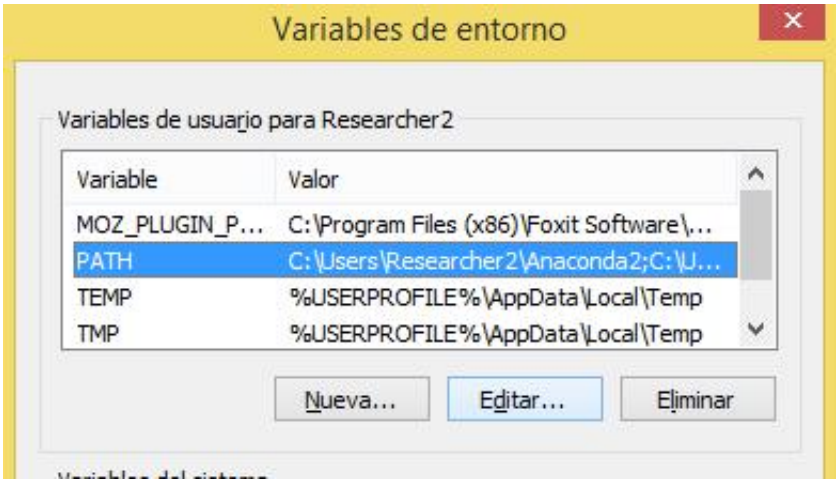
Configuración avanzada del sistema



Variables de entorno



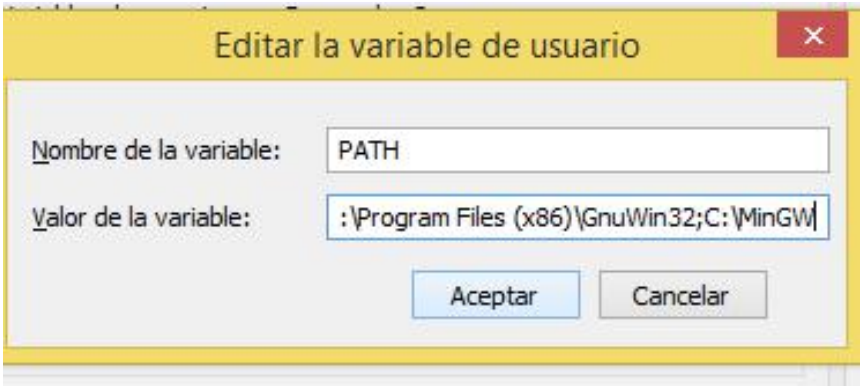
Hacemos click en PATH y Editar



Si el campo no está vacío, ir al final, escribir ";" y agregar las rutas de Bison y MinGx con un \bin al final, separados por un ";"
variables_anteriores**;C:\Program Files (x86)\GnuWin32\bin; C:\MinGW\bin**

Si el campo está vacío, escribiremos:

C:\Program Files (x86)\GnuWin32\bin; C:\MinGW\bin



Aceptamos y aceptamos

Formato de fichero de especifiación de Bisón I

Un fichero de especifiación de Bison tiene tres secciones separadas por lineas que contienen el separador %%

Sección de definiciones

%{

** delimitadores de código C **

%}

%%

Sección de reglas

%%

Sección de funciones de usuario

Se declaran los símbolos terminales %token, se pueden declarar otros símbolos, los símbolos no terminales van en Mayúsculas

Es necesario utilizar el flex en conjunto con el Bison, para tener un programa en bison, es neceario seguir los siguientes comandos

flex archivo.l

bison -d archivo.y

1. Crear una simple interacción entre flex y bison

Primero el archivo lexico.l

```
%{
   #include "sintactico.tab.h"
```

```
    int yyparse();
%}

%%

El {return ARTICULO;}


%%
int yywrap(){ return 0;}

void main(){
      yyparse();
   yylex();
}
```

Segundo el archivo sintactico.y

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

2. Utilizar dos articulos como alternativa para el analisis sintactico.

primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}
```

```
%%

El|Ella {return ARTICULO;}


%%
int yywrap(){ return 0;}

void main(){
     yyparse();
    yylex();
}
```

segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

3.Agregar el articulo Ellos para que sea reconocido por el analizador sintactico

Primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}
```

```
%%

El|Ella|Ellos {return ARTICULO;}


%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

4.Escribir el analizador léxico para crear un analizador sintáctico que reconozca los artículos “El”, “Ella” y “Ellos”,

Primero el léxico

```
%{
   #include "sintactico1.tab.h"
   int yyparse();
%}
```

```
%%

El|Ella|Ellos {return ARTICULO;}


%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico, asegurarse de guardarlos como sintactico1.y

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

5.Porner los articulos en minusculas y ver si operan con dichos articulos

primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%
```

```
el|ella|ellos {return ARTICULO;}


%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}

Segundo el sintactico
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

6.Utilizar los articulos lo, la y el para realizar un analizador sintáctico.

```
Primero el léxico
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

lo|la|el {return ARTICULO;}
```

```
%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

7.Crear un SUJETO en el analizador léxico

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

lo|la|el {return ARTICULO;}

perro {return SUJETO;}
```

```
%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */

%token ARTICULO

%token SUJETO

%%

cadena: ARTICULO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

8.Agregar un SUJETO al archivo sintactico

primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

lo|la|el {return ARTICULO;}
```

```
perro {return SUJETO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%%

cadena: ARTICULO SUJETO{printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

9.Agregar perra al analizador léxico

Primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%
```

```
lo|la|el {return ARTICULO;}

perro|perra {return SUJETO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%%

cadena: ARTICULO SUJETO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

10.Agregar el nombre carros a los sujetos aceptados por la gramatica

Primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}
```

```
%%

lo|la|el|los {return ARTICULO;}

perro|perra|carros {return SUJETO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%%

cadena: ARTICULO SUJETO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

11. Agregar el token verbo y el verbo “come”

Primero el léxico

```
%{
   #include "sintactico.tab.h"
```

```
   int yyparse();
%}

%%

lo|la|el|los {return ARTICULO;}

perro|perra|carros {return SUJETO;}

come {return VERBO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%token VERBO

%%

cadena: ARTICULO SUJETO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

12. Agregar VERBO a la cadena en el archivo sintactico

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

lo|la|el|los {return ARTICULO;}

perro|perra|carros {return SUJETO;}

come {return VERBO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%token VERBO

%%

cadena: ARTICULO SUJETO {printf("Se imprimio una cadena \n");}
;

%%
```

```
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

13.Agregar los siguientes verbos a la parte léxica transitar y muerde

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

lo|la|el|los {return ARTICULO;}

perro|perra|carros {return SUJETO;}

come|transitar|morder {return VERBO;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token ARTICULO

%token SUJETO

%token VERBO
```

```
%%

cadena: ARTICULO SUJETO {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

14.Crear un analizador sintactico que reconozca solo un digito

primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9] {return OPERADOR;}



%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR

%%
```

```
cadena: OPERADOR {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

15.Crear un analizador sintactico que reconozca solo dos digitos

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9] {return OPERADOR;}



%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR
```

```
%%

cadena: OPERADOR {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

16.Crear un anlizador sintactico que reconozca todos los número enteros

Primero el lexico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9]* {return OPERADOR;}



%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintactico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
```

```
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR

%%

cadena: OPERADOR {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

17. Crear un analizador sintáctico que reconozca la suma de dos números

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9]* {return OPERADOR1;}

\+ {return OPERACION;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR1
```

```
%token OPERACION

%%

cadena: OPERADOR1 OPERACION OPERADOR1 {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

18.Crear un analizador sintactico que reconozca la resta de dos números

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9]* {return OPERADOR1;}

- {return OPERACION;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
```

```
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR1

%token OPERACION

%%

cadena: OPERADOR1 OPERACION OPERADOR1 {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

19.Crear un analizador sintáctico que reconzca la multiplicación de dos números

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9]* {return OPERADOR1;}

\* {return OPERACION;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}
```

```
/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR1

%token OPERACION

%%

cadena: OPERADOR1 OPERACION OPERADOR1 {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

20.Crear un analizador sintáctico que reconozca la división enttre dos números

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[0-9][0-9]* {return OPERADOR1;}

\/ {return OPERACION;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);
```

```
%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token OPERADOR1

%token OPERACION

%%

cadena: OPERADOR1 OPERACION OPERADOR1 {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

21.Crear un analizador sintáctico que permita la operación de asignación de valor a una variable

Primero el léxico

```
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[a-z][0-9][0-9]* {return VARIABLE;}

= {return IGUALDAD;}

[0-9][0-9]* {return OPERADOR;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
```

```
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token VARIABLE

%token IGUALDAD

%token OPERADOR

%%

cadena: VARIABLE IGUALDAD OPERADOR {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

22. Crear un analizador sintactico que permita la finalización de cualquier palabra con “;”

Primero el léxico

```
%{
   #include "sintactico.tab.h"
   int yyparse();
%}

%%

[a-zA-Z][a-z]* {return PALABRA;}

; {return PUNTOYCOMA;}

%%
int yywrap(){ return 0;}

void main(){
yyparse();
   yylex();
}
```

Segundo el sintáctico

```
%{

#include <stdio.h>
int yylex();
int yyerror (char *s);


%}

/* Declaraciones de BISON */
/* se imprimio error porque sujeto no esta en cadena*/
%token PALABRA

%token PUNTOYCOMA

%token OPERADOR

%%

cadena: PALABRA PUNTOYCOMA {printf("Se imprimio una cadena \n");}
;

%%
int yyerror (char *s)
{

 printf ("%s\n", s);
 return 0;
}
```

VII. Ejercicios

1. Hacer un manual de instalación de bison para el sistema operativo linux
2. Crear un programa en bison y flex que reconozca la apertura de paréntesis y el cierre de paréntesis.
3. Crear un programa en bison y flex que reconozca cinco combinaciones de notas musicales
4. Crear un programa en bison y flex que reconozca los dos puntos ":" como final de una línea
5. Redactar un informe con los ejercicios propuestos

VII. Bibliografía y referencias

1. https://www.gnu.org/software/bison/