```
%{
  #include<stdio.h>
  #include<string.h>
  char lexema[255];
  void yyerror(char *);
%}

// Especificamos los tokens
%token CORREO

// Especificamos la gramatica
%%
instruccion: CORREO;
instruccion: ;
%%

void yyerror(char *msg) {
  printf("error: %s", msg);
}

// Especificamos las reglas de los tokens
int yylex() {
  char c;
  while(1) {
    c = getchar();
    if(c == '\n') continue;
    if(c == ' ') continue;

    // Token CORREO
    if(isalpha(c)) {
      int i = 0;
      do {
        lexema[i++] = c;
        c = getchar();
      } while(isalpha(c));
      char dominio[] = "@uni.edu.pe";
      int j = 0;

      if(c == dominio[j]) {
        do {
          lexema[i++] = c;
          c = getchar();
          j++;
        } while(c == dominio[j] && j < 11);
        if (j == 11) {
          ungetc(c, stdin);
```

```
          lexema[i] == 0;
          return CORREO;
        }
      }
    }
    return c;
  }
}

int main() {
  if(!yyparse()) printf("cadena valida\n");
  else printf("cadena invalida\n");
  return 0;
}
```

Código valido solo para token CORREO (letras seguido de @uni.edu.pe)

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
 8s216ms >_ .\a.exe
rgebr@uni.edu.pee
error: syntax errorcadena invalida

[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
 12s820ms >_ .\a.exe
b324123@uni.edu.pe
error: syntax errorcadena invalida
```

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
 7s220ms >_ .\a.exe
jAlvarez@uni.edu.pe
^Z
cadena valida

[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
 10s823ms >_ .\a.exe
rgherhehEGT@uni.edu.pe
^Z
cadena valida
```

```
%{
  #include<stdio.h>
  #include<string.h>
  char lexema[255];
  void yyerror(char *);
%}

// Especificamos los tokens
%token NUMERO

// Especificamos la gramatica
%%
instruccion: NUMERO;
instruccion: ;
%%

void yyerror(char *msg) {
  printf("error: %s", msg);
}

// Especificamos las reglas de los tokens
int yylex() {
  char c;
  while(1) {
    c = getchar();
    if(c == '\n') continue;
    if(c == ' ') continue;

    // Token NUMERO
    if(isdigit(c) && c != '0') {
      int i = 0;
      lexema[i++] = c;
      if (c == '1') {
        c = getchar();
        if(c == '0') {
          lexema[i++] = c;
          c = getchar();
          if(c == '.') {
            lexema[i++] = c;
            c = getchar();
            // Despues del . continuan varios 0s
            if(c == '0') {
              do {
                lexema[i++] = c;
                c = getchar();
              } while(c == '0');
```

```
            // Despues una E
            if(c == 'E') {
              // Despues varios digitos
              do {
                lexema[i++] = c;
                c = getchar();
              } while(isdigit(c));
              ungetc(c, stdin);
              lexema[i] == 0;
              return NUMERO;
            }
          }
        }
      }
      if(c == '.') {
        lexema[i++] = c;
        c = getchar();
        // Despues del . continuan varios digitos
        if(isdigit(c)) {
          do {
            lexema[i++] = c;
            c = getchar();
          } while(isdigit(c));
          // Despues una E
          if(c == 'E') {
            // Despues varios digitos
            do {
              lexema[i++] = c;
              c = getchar();
            } while(isdigit(c));
            ungetc(c, stdin);
            lexema[i] == 0;
            return NUMERO;
          }
        }
      }
    } else {
      // Continua un . para que numero real
      // este entre 1 y 10
      if(c == '.') {
        lexema[i++] = c;
        c = getchar();
        // Despues del . continuan varios digitos
        if(isdigit(c)) {
          do {
            lexema[i++] = c;
```

```
            c = getchar();
          } while(isdigit(c));
          // Despues una E
          if(c == 'E') {
            // Despues varios digitos
            do {
              lexema[i++] = c;
              c = getchar();
            } while(isdigit(c));
            ungetc(c, stdin);
            lexema[i] == 0;
            return NUMERO;
          }
        }
      }
    }
    return c;
  }
}

int main() {
  if(!yyparse()) printf("cadena valida\n");
  else printf("cadena invalida\n");
  return 0;
}
```

Código valido solo para token NUMERO (Real entre 1 y 10 seguido de E seguido de numero)

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]

10s268ms >_ .\a.exe
1.63456E3245346
^Z
cadena valida

[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]

14s335ms >_ .\a.exe
9.56546E456546
^Z
cadena valida
```

```
┌─【USUARIO@LAPTOP-1D2RDUOO 🏠 \Desktop\Compiladores\semana4】
└─ ⏱ 9s657ms >_ .\a.exe
41234.32535E342534
error: syntax errorcadena invalida

┌─【USUARIO@LAPTOP-1D2RDUOO 🏠 \Desktop\Compiladores\semana4】
└─ ⏱ 6s432ms >_ .\a.exe
1.32432E213423
^Z
cadena valida

┌─【USUARIO@LAPTOP-1D2RDUOO 🏠 \Desktop\Compiladores\semana4】
└─ ⏱ 10s529ms >_ .\a.exe
10.12432E324532
error: syntax errorcadena invalida
```

```
┌─【USUARIO@LAPTOP-1D2RDUOO 🏠 \Desktop\Compiladores\semana4】
└─ ⏱ 8s312ms >_ .\a.exe
10.0E3453425
^Z
cadena valida
```

```
┌─【USUARIO@LAPTOP-1D2RDUOO 🏠 \Desktop\Compiladores\semana4】
└─ ⏱ 13s231ms >_ .\a.exe
10.0000E2134
^Z
cadena valida
```

Código valido para ambos tokens

```
%{
  #include<stdio.h>
  #include<string.h>
  char lexema[255];
  void yyerror(char *);
%}

// Especificamos los tokens
%token CORREO NUMERO

// Especificamos la gramatica
%%
instruccion: expr instruccion;
instruccion: ;
expr: CORREO | NUMERO;
%%

void yyerror(char *msg) {
  printf("error: %s", msg);
}

// Especificamos las reglas de los tokens
int yylex() {
  char c;
  while(1) {
    c = getchar();
    if(c == '\n') continue;
    if(c == ' ') continue;

    // Token CORREO
    if(isalpha(c)) {
      int i = 0;
      do {
        lexema[i++] = c;
        c = getchar();
      } while(isalpha(c));
      char dominio[] = "@uni.edu.pe";
      int j = 0;

      if(c == dominio[j]) {
        do {
          lexema[i++] = c;
          c = getchar();
          j++;
        } while(c == dominio[j] && j < 11);
```

```
      if (j == 11) {
        ungetc(c, stdin);
        lexema[i] == 0;
        return CORREO;
      }
    }
  }

  // Token NUMERO
  if(isdigit(c) && c != '0') {
    int i = 0;
    lexema[i++] = c;
    if (c == '1') {
      c = getchar();
      if(c == '0') {
        lexema[i++] = c;
        c = getchar();
        if(c == '.') {
          lexema[i++] = c;
          c = getchar();
          // Despues del . continuan varios 0s
          if(c == '0') {
            do {
              lexema[i++] = c;
              c = getchar();
            } while(c == '0');
            // Despues una E
            if(c == 'E') {
              // Despues varios digitos
              do {
                lexema[i++] = c;
                c = getchar();
              } while(isdigit(c));
              ungetc(c, stdin);
              lexema[i] == 0;
              return NUMERO;
            }
          }
        }
      }
      if(c == '.') {
        lexema[i++] = c;
        c = getchar();
        // Despues del . continuan varios digitos
        if(isdigit(c)) {
          do {
```

```
            lexema[i++] = c;
            c = getchar();
          } while(isdigit(c));
          // Despues una E
          if(c == 'E') {
            // Despues varios digitos
            do {
              lexema[i++] = c;
              c = getchar();
            } while(isdigit(c));
            ungetc(c, stdin);
            lexema[i] == 0;
            return NUMERO;
          }
        }
      }
    } else {
      // Continua un . para que numero real
      // este entre 1 y 10
      if(c == '.') {
        lexema[i++] = c;
        c = getchar();
        // Despues del . continuan varios digitos
        if(isdigit(c)) {
          do {
            lexema[i++] = c;
            c = getchar();
          } while(isdigit(c));
          // Despues una E
          if(c == 'E') {
            // Despues varios digitos
            do {
              lexema[i++] = c;
              c = getchar();
            } while(isdigit(c));
            ungetc(c, stdin);
            lexema[i] == 0;
            return NUMERO;
          }
        }
      }
    }
  }

  return c;
}
```

```c
}

int main() {
  if(!yyparse()) printf("cadena valida\n");
  else printf("cadena invalida\n");
  return 0;
}
```

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
>_ .\a.exe
reg@uni
error: syntax errorcadena invalida
```

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
>_ .\a.exe
rgreh@uni.edu.pe
^Z
cadena valida
```

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
6s957ms >_ .\a.exe
jAlvarez@uni.edu.pee
error: syntax errorcadena invalida
```

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
10s641ms >_ .\a.exe
13.35345E43534
error: syntax errorcadena invalida

[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
7s752ms >_ .\a.exe
1.1234E13245
^Z
cadena valida
```

Considerando que después del punto debe continuar al menos un numero

```
[USUARIO@LAPTOP-1D2RDUOO \Desktop\Compiladores\semana4]
7s594ms >_ .\a.exe
1.E3532
error: syntax errorcadena invalida
```

Penadillo Lazares Wenses Johan