

Centro Universitario de Ciencias Exactas e Ingenierías

Programación de sistemas embebidos

Hands-on 1: Implementación de Autómatas

Carrera: Ingeniería en computación.

Alumno: Fabián Joheshua Escalante Fernández

Materia: Compiladores

Calendario: 2025A

Fecha: 15/03/2024

Hands-on 1: Implementación de Autómatas

Índice

Cadenas alfabéticas

1.1 Código en C

1.2 Código en Python

Números reales

2.1 Código en C

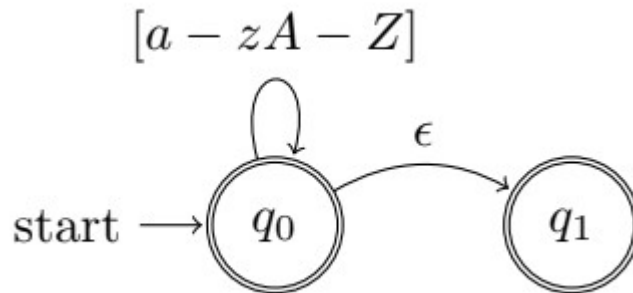
2.2 Código en Python

Sentencias selectivas

3.1 Código en C

3.2 Código en Python

Cadenas alfabéticas



Código en C

Figura 1

El código realizado en C usa una variable de tipo char, después en el proceso principal realiza la operación de la función validate_alpha, esta función toma un char y lo convierte a string, este valor lo toma y con <ctype.h> usa isalpha(*str), se pone a iterar con un while cada elemento de la cadena hasta encontrar un valor que no es una letra ni mayúscula ni minúscula, si encuentra un elemento así entonces regresa el valor False, si todas son letras entonces regresa True. Estos booleanos se pasan a la operación principal y se usan para usar una selectiva if else para imprimir un mensaje según el caso.

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int validate_alpha(const char *str) {
5      while (*str) {
6          if (!isalpha(*str)) return 0;
7          str++;
8      }
9      return 1;
10 }
11
12
13 int main() {
14     const char *input = "FabianJoheshuaEscalanteFernandez";
15     if (validate_alpha(input)) {
16         printf("Cadena válida.\n");
17     } else {
18         printf("Cadena inválida.\n");
19     }
20     return 0;
21 }
```

Filter Code

[Running] cd "c:\Users\Venus\Desktop\test of mine\" && gcc try.c -o try &&
"c:\Users\Venus\Desktop\test of mine\"try
Cadena válida.

Figura 2

Hands-on 1: Implementación de Autómatas

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int validate_alpha(const char *str) {
5      while (*str) {
6          if (!isalpha(*str)) return 0;
7          str++;
8      }
9      return 1;
10 }
11
12
13 int main() {
14     const char *input = "123Letra";
15     if (validate_alpha(input)) {
16         printf("Cadena válida.\n");
17     } else {
18         printf("Cadena inválida.\n");
19     }
20     return 0;
21 }
```

... Filter Code

[Done] exited with CODE=0 in 0.04 seconds

[Running] cd "c:\Users\Venus\Desktop\test of mine\" && gcc try.c -o try &&
"c:\Users\Venus\Desktop\test of mine\"try
Cadena inválida.

Figura 3

Código en Python

Este código es muy similar al anterior, aquí la diferencia es que python tiene un método integrado que se llama `isalpha()` que checa todos los elementos de un string para ver si son letras.

```
Validación_de_Cadenas_Alfabéticas.py > ...
1  def validate_alpha(s):
2      return s.isalpha()
3
4  input_str = "123Letra"
5  if validate_alpha(input_str):
6      print("Cadena válida.")
7  else:
8      print("Cadena inválida.")
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER powershell +

PS C:\Users\Venus\Desktop\test of mine> .\Validación_de_Cadenas_Alfabéticas.py
Cadena válida.

Figura 4

Compiladores

```
Validación_de_Cadenas_Alfabéticas.py > ...
1 def validate_alpha(s):
2     return s.isalpha()
3
4 input_str = "FabianJoheshuaEscalanteFernandez"
5 if validate_alpha(input_str):
6     print("Cadena válida.")
7 else:
8     print("Cadena inválida.")
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

PS C:\Users\Venus\Desktop\test of mine> .\Validación_de_Cadenas_Alfabéticas.py
Cadena válida.

Figura 5

Números reales

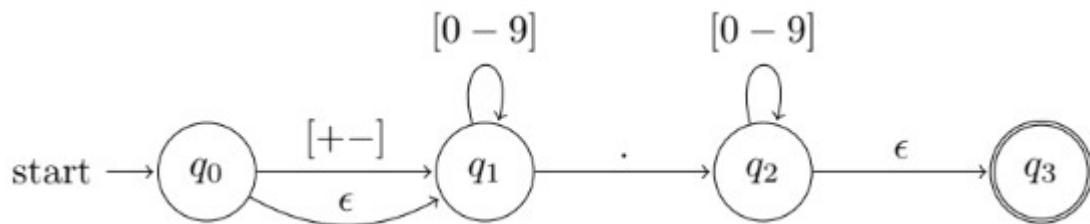


Figura 6

Código en C

Este código en C utiliza una variable tipo char y se lo pasa a la función validate real, esta convierte el char en string valida y que esta tenga un signo + o - y que contenga un punto, si contiene un punto entonces retorna un valor True. Si no tiene signos usa con <ctype.h> el método isdigit para checar si cada elemento del string es un número, si no lo es entonces retorna False. En la operación principal según si es False o True retornamos un mensaje diferente.

Hands-on 1: Implementación de Autómatas

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int validate_real(const char *str) {
5      int has_dot = 0;
6      if (*str == '+' || *str == '-') str++;
7      while (*str) {
8          if (*str == '.') {
9              if (has_dot) return 0;
10             has_dot = 1;
11         }
12         else if (!isdigit(*str)) return 0;
13         str++;
14     }
15     return has_dot;
16 }
17
18 int main() {
19     const char *input = "-123.456";
20     if (validate_real(input)) {
21         printf("Número válido.\n");
22     } else {
23         printf("Número inválido.\n");
24     }
25     return 0;
26 }
```

... Filter Code ▾
Número válido.

Figura 7

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int validate_real(const char *str) {
5      int has_dot = 0;
6      if (*str == '+' || *str == '-') str++;
7      while (*str) {
8          if (*str == '.') {
9              if (has_dot) return 0;
10             has_dot = 1;
11         }
12         else if (!isdigit(*str)) return 0;
13         str++;
14     }
15     return has_dot;
16 }
17
18 int main() {
19     const char *input = "-Noteriet.456";
20     if (validate_real(input)) {
21         printf("Número válido.\n");
22     } else {
23         printf("Número inválido.\n");
24     }
25     return 0;
26 }
```

... Filter Cc
Número inválido.

Figura 8

Compiladores

Código en Python

Igualmente, es parecido al ejemplo en C pero en este caso se intenta convertir en la función `validate_real` al string a un tipo de dato `float`, entonces esta operación tiene que ser exitosa si es que tiene números porque solo estamos cambiando de string a float, pero va a fallar si tiene algún tipo de letra porque no se puede convertir una letra a número de esta forma.

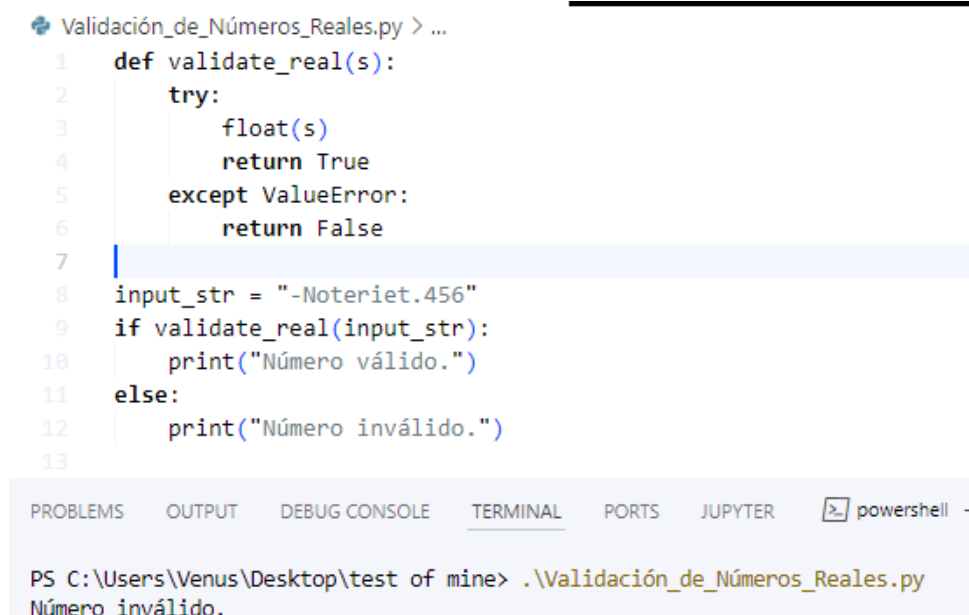


```
Validación_de_Números_Reales.py > ...
1 def validate_real(s):
2     try:
3         float(s)
4         return True
5     except ValueError:
6         return False
7
8 input_str = "-123.456"
9 if validate_real(input_str):
10     print("Número válido.")
11 else:
12     print("Número inválido.")
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER powershell + v

PS C:\Users\Venus\Desktop\test of mine> .\Validación_de_Números_Reales.py
Número válido.

Figura 9



```
Validación_de_Números_Reales.py > ...
1 def validate_real(s):
2     try:
3         float(s)
4         return True
5     except ValueError:
6         return False
7
8 input_str = "-Noteriet.456"
9 if validate_real(input_str):
10     print("Número válido.")
11 else:
12     print("Número inválido.")
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER powershell -

PS C:\Users\Venus\Desktop\test of mine> .\Validación_de_Números_Reales.py
Número inválido.

Figura 10

Hands-on 1: Implementación de Autómatas

Sentencias selectivas

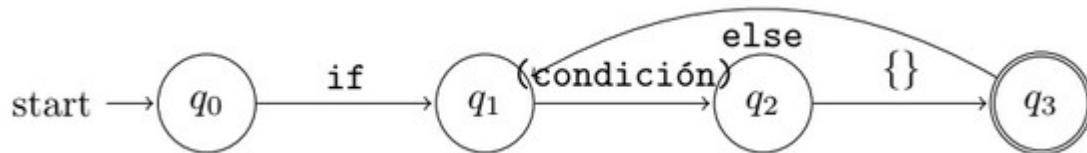


Figura 11
Código en C

Este código toma una variable tipo char y utiliza la función `devalidate_if_else` convirtiendo este char a un string, este valor es tomado y con `<string.h>` usa `strstr()` que busca un string dentro de otro, lo que hace es retornar `False` o `True`, dependiendo si el string tiene `else` y `if`, en dado caso retorna `True`. En la operación principal según el booleano se imprime un mensaje.

```
1  #include <stdio.h>
2  #include <string.h>
3  int validate_if_else(const char *str) {
4      return strstr(str, "if") != NULL && strstr(str, "else") != NULL;
5  }
6
7  int main() {
8      const char *input = "if (x > 0) { } else { }";
9      if (validate_if_else(input)) {
10         printf("Sentencia válida.\n");
11     } else {
12         printf("Sentencia inválida.\n");
13     }
14     return 0;
15 }
```

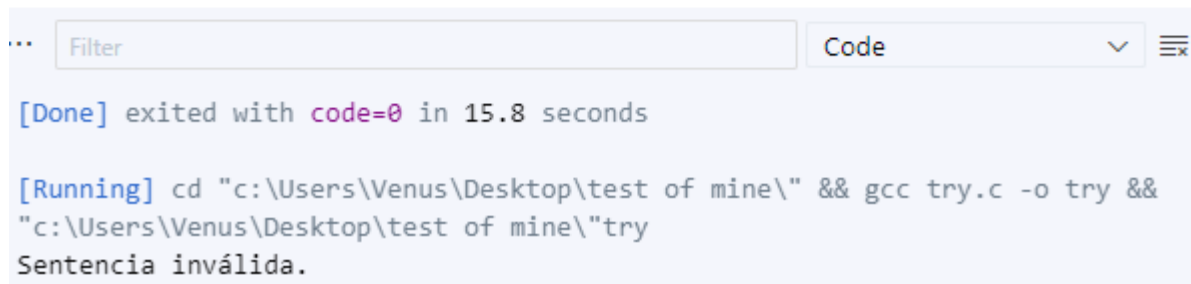
... Filter Code

```
[Running] cd "c:\Users\Venus\Desktop\test of mine\" && gcc try.c -o try &&
"c:\Users\Venus\Desktop\test of mine\"try
Sentencia válida.
```

Figura 12

Compiladores

```
1  #include <stdio.h>
2  #include <string.h>
3  int validate_if_else(const char *str) {
4      return strstr(str, "if") != NULL && strstr(str, "else") != NULL;
5  }
6
7  int main() {
8      const char *input = "ola (x > 0) { } else { }";
9      if (validate_if_else(input)) {
10         printf("Sentencia válida.\n");
11     } else {
12         printf("Sentencia inválida.\n");
13     }
14     return 0;
15 }
```



```
... Filter Code v ≡
[Done] exited with code=0 in 15.8 seconds
[Running] cd "c:\Users\Venus\Desktop\test of mine\" && gcc try.c -o try &&
"c:\Users\Venus\Desktop\test of mine\"try
Sentencia inválida.
```

Figura 13

Código en Python

Este código es parecido al de C, en este caso solo tomamos el string, lo guardamos en una variable y lo pasamos a una función que checa con in, haciendo la comparación si if y else para retornar un booleano e imprimir un mensaje según el resultado.

Hands-on 1: Implementación de Autómatas

```
Validación_de_Sentencias_Selectivas.py > ...
1  def validate_if_else(s):
2      return "if" in s and "else" in s
3
4  input_str = "if x > 0: pass else: pass"
5  if validate_if_else(input_str):
6      print("Sentencia válida.")
7  else:
8      print("Sentencia inválida.")
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

Sentencia válida.

Figura 14

```
Validación_de_Sentencias_Selectivas.py > ...
1  def validate_if_else(s):
2      return "if" in s and "else" in s
3
4  input_str = "ola x > 0: pass else: pass"
5  if validate_if_else(input_str):
6      print("Sentencia válida.")
7  else:
8      print("Sentencia inválida.")
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

Sentencia inválida.

Figura 15