# Capítulo 6

# Caracteres y Cadenas

# Objetivos

- Representar caracteres usando el tipo char.
- Codificar caracteres utilizando ASCII y Unicode.
- Pasar un valor numérico a un carácter y pasar un carácter a un número entero.
- Comparar y probar caracteres usando los métodos estáticos en la clase Caracter.
- Introducir objetos y métodos de instancia.
- Representar cadenas utilizando los objetos String.
- Devolver la longitud de la cadena utilizando el método length().
- Devolver un carácter en la cadena usando el método charAt(i).
- Usar el operador + para concatenar cadenas.
- Leer cadenas desde la consola.
- Leer un carácter desde la consola.
- Casting entre caracteres y números.
- Comparar cadenas utilizando el método equals y los métodos compareTo..
- Obtener subcadenas.
- Encontrar un carácter o una subcadena en una cadena usando el método indexOf.
- Formatear la salida utilizando el método System.out.printf.

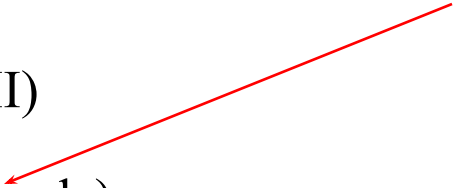
# Tipo de datos de carácter

char letter = 'A'; (ASCII)

char numChar = '4'; (ASCII)

char letter = '\u0041'; (Unicode)

char numChar = '\u0034'; (Unicode)



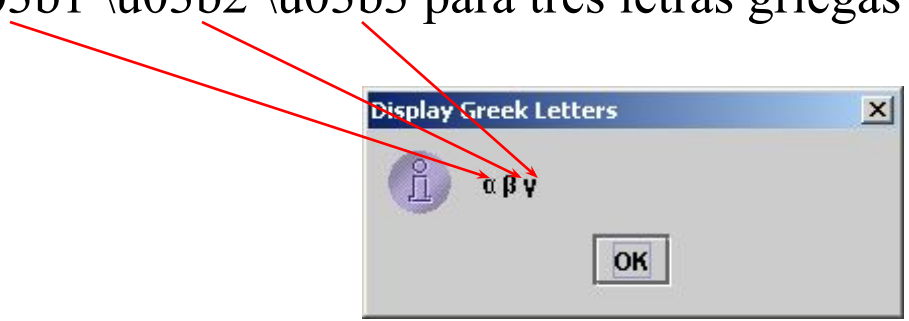

NOTA: Los operadores de incremento y decremento también se pueden usar en variables char para obtener el carácter Unicode siguiente o anterior. Por ejemplo, las siguientes instrucciones muestran el carácter b.

```
char ch = 'a';
System.out.println(++ch);
```

# Formato Unicode

Los caracteres Java utilizan Unicode, un esquema de codificación de 16 bits establecido por el Consorcio Unicode para apoyar el intercambio, procesamiento y visualización de textos escritos en los diversos idiomas del mundo. Unicode toma dos bytes, precedidos por \u, expresados en cuatro números hexadecimales que van desde '\u0000' a '\uFFFF'. Por lo tanto, Unicode puede representar 65.535 + 1 caracteres.

Unicode \u03b1 \u03b2 \u03b3 para tres letras griegas



# Código ASCII para caracteres

Clasificación de caracteres

- Dígitos: formado por los carateres '0', '1',...,'9'.
- Letras: formado por todos los elementos del alfabeto, tanto en minúscula ('a', 'b',...) como en mayúscula ('A', 'B',...) .
- Caracteres blancos: como el espacio o el tabulador,... (no tienen representación al mostrarse o imprimirse.
- Otros caracteres: signos de puntuación, matemáticos, etc.

| Characters | Code Value in Decimal | Unicode Value |
|---|---|---|
| '0' to '9' | 48 to 57 | \u0030 to \u0039 |
| 'A' to 'Z' | 65 to 90 | \u0041 to \u005A |
| 'a' to 'z' | 97 to 122 | \u0061 to \u007A |

# Las secuencias para caracteres especiales

| Secuencia de escape | Nombre | Codigo Unicode | Valor decimal |
|---|---|---|---|
| \b | Retroceso | \u0008 | 8 |
| \t | Tabulación | \u0009 | 9 |
| \n | Avance de línea | \u000A | 10 |
| \f | Salto de página | \u000C | 12 |
| \r | Retorno de carro | \u000D | 13 |
| \\ | Barra invertida | \u005C | 92 |
| \" | Comillas dobles | \u0022 | 34 |

# Casting entre caracteres y números

```
int i = 'a'; // Igual que int i = (int)'a';

char c = 97; // Igual que char c = (char)97;
```

## Aritmética de Caracteres

```
char ch = 'e' - 2;  //ch vale 'c'
char ch = 'e' + 2;  //ch vale 'g'
```

## Comparar y comprobar Caracteres

```
if (ch >= 'A' && ch <= 'Z')
  System.out.println(ch + " es una letra mayúscula ");
else if (ch >= 'a' && ch <= 'z')
  System.out.println(ch + " es una letra minúscula ");
else if (ch >= '0' && ch <= '9')
  System.out.println(ch + " es un carácter numérico ");
```

# Métodos de la clase Character

La clase Character amplía la funcionalidad del tipo char.

Los métodos de Character para consultar si un carácter pertenece a alguno de estos grupos devuelve un booleano, true en caso de que pertenezca o false en caso contrario.

| Method | Description |
|---|---|
| `isDigit(ch)` | Devuelve true si el carácter especificado es un dígito.. |
| `isLetter(ch)` | Devuelve true si el carácter especificado es una letra.. |
| `isLetterOfDigit(ch)` | Devuelve true si el carácter especificado es una letra o un dígito. |
| `isLowerCase(ch)` | Devuelve true si el carácter especificado es una letra minúscula. |
| `isUpperCase(ch)` | Devuelve true si el carácter especificado es una letra mayúscula. |
| `isSpaceChar(ch)` | Devuelve true si el carácter especificado es un espacio(' '). |
| `isWhiteSpace(ch)` | Devuelve true si el carácter especificado es algún tipo de blanco (' ', '\r', '\n', '\t',...). |

# Conversión de caracteres

Los métodos de conversión devuelven el carácter en otro caracter o en un valor de un tipo distinto. También realizan la operación inversa, convierten un valor de otro tipo en un carácter.

**char toLowerCase(char c):** si el carácter especificado es una letra lo devuelve convertido a minúscula. En otro caso, devuelve el mismo.

```
char c1 = 'A', c2;
c2 = Character.toLowerCase(c1); //la variable c2 toma el valor 'a'
c2 = Character.toLowerCase('3'); //al no ser una letra, devuelve el
                                 //mismo valor pasado: '3'
```

**char toUpperCase(char c):** si el carácter especificado es una letra lo devuelve convertido a mayúscula. En otro caso, devuelve el mismo.

```
char c1 = 'g', c2;
c2 = Character.toUpperCase(c1); //a c2 se le asigna el valor 'G'
```

# Conversión de caracteres

**int digit(char c, int base):** devuelve el valor numérico de un carácter c en la base indicada. Si no es posible realizar la conversión devuelve -1.

```
int num = Character.digit('3', 10); //num vale 3
```

**String toString(char c):** devuelve una cadena de longitud uno que contiene a c. Con la cadena que obtenemos se podrá trabajar con todas las funciones de la clase String.

```
String cad;
cad = Character.toString('b'); //cad toma el valor de la cadena "b"
```

**char forDigit(int digito, int base):** devuelve el carácter que representa a dígito en la base indicada. Si no es posible realizar la conversión devuelve el caracter nulo.

```
Character.forDigit(3, 10) //devuelve el carácter '3'
Character.forDigit(12, 16) //devuelve el carácter 'c', que representa
                           //el 12 en hexadecimal
```

# Ejercicios

1. Escribe un programa que lea un carácter y determine si es una letra, un dígito o algún otro símbolo.
2. Escribe un programa que lea un carácter e indique si es un espacio en blanco usando el método adecuado de la clase Character.
3. Solicita al usuario un carácter y convierte las letras minúsculas a mayúsculas y las mayúsculas a minúsculas. Si el carácter ingresado no es una letra, indica que es inválido.
4. Escribe un programa que lea una letra y un carácter específico ('M' para mayúscula, 'm' para minúscula) e indique si el carácter ingresado coincide con el tipo especificado.
5. Escribe un programa que tome un carácter como entrada y verifique si es una vocal (tanto mayúscula como minúscula).
6. Escribe un programa que reciba dos caracteres como entrada y verifique si son iguales, ignorando si son mayúsculas o minúsculas.

# Práctica: Generar caracteres aleatorios

Todos los operadores numéricos se pueden aplicar a los operandos de char. El operando char se convierte en un número si el otro operando es un número o un carácter. Entonces, la expresión anterior se puede simplificar de la siguiente manera :

```
'a' + Math.random() * ('z' - 'a' + 1)
```

Entonces, una letra minúscula al azar es

```
(char)('a' + Math.random() * ('z' - 'a' + 1))
```

Para generalizar, se puede generar un carácter aleatorio entre dos caracteres ch1 y ch2 siendo ch1 < ch2 de la siguiente manera :

```
(char)(ch1 + Math.random() * (ch2 – ch1 + 1))
```

# La Clase RandomCharacter

```
// RandomCharacter.java: Genera caracteres aleatorios
public class RandomCharacter {
  /** Genera un carácter aleatorio entre ch1 y ch2 */
  public static char getRandomCharacter(char ch1, char ch2) {
    return (char)(ch1 + Math.random() * (ch2 - ch1 + 1));
  }
   /** Genera una letra minúscula al azar */
  public static char getRandomLowerCaseLetter() {
    return getRandomCharacter('a', 'z');
  }
   /** Genera una letra mayúscula al azar */
  public static char getRandomUpperCaseLetter() {
    return getRandomCharacter('A', 'Z');
  }
   /** Genera un carácter aleatorio numérico */
  public static char getRandomDigitCharacter() {
    return getRandomCharacter('0', '9');
  }
   /** Genera un carácter aleatorio */
  public static char getRandomCharacter() {
    return getRandomCharacter('\u0000', '\uFFFF');
  }
}
```

# Ejercicios

1. Realiza un programa que muestre al azar el nombre de una carta de la baraja francesa. Esta baraja está dividida en cuatro palos: picas, corazones, diamantes y tréboles. Cada palo está formado por 13 cartas, de las cuales 9 cartas son numerales y 4 literales: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K y A (que sería el 1).
2. Realiza un programa para rellenar una quiniela al azar, el programa muestra un 1, una X o un 2 pero, haciendo que la probabilidad de que salga un 1 sea de 1/2, la probabilidad de que salga X sea de 1/3 y la probabilidad de que salga 2 sea de 1/6.
   Pista: 1/2 = 3/6 y 1/3 = 2/6.
3. Realiza un programa que escriba por pantalla una palabra formada por 5 caracteres alfabéticos aleatorios en mayúsculas y esos mismos caracteres en minúsculas.
4. Realiza un programa que pinte por pantalla una línea con 5 caracteres. El carácter con el que se pinta esa línea se elige de forma aleatoria entre uno de los siguientes: #,$,€,&
5. Realiza el programa anterior con los siguientes caracteres:*, -, =, ., |, @.

# El tipo de cadena - String

El tipo char sólo representa un carácter. Para representar una cadena de caracteres, utilice el tipo de datos denominado String. Por ejemplo,

```
String message = "Welcome to Java";
```

String es en realidad una clase predefinida en la biblioteca Java al igual que la clase System y la clase Scanner. El tipo String no es un tipo primitivo. Es conocido como un tipo de referencia. Cualquier clase Java puede usarse como un tipo de referencia para una variable. Los tipos de datos de referencia se analizarán a fondo en el Capítulo: "Objetos y clases".

Por el momento, sólo necesita saber:

- cómo declarar una variable String,
- cómo asignar una cadena a la variable,
- cómo concatenar y realizar operaciones simples con cadenas.

# Métodos simples para objetos String

Un método estático se puede invocar sin utilizar un objeto. Todos los métodos definidos en la clase Math son métodos estáticos. No están vinculados a una instancia de objeto específica.

En las cadenas existe un método estático que permite representar un tipo primitivo en forma de cadena:
**static String value0f (tipo valor)**
es un método sobrecargado, en función del parámetro que se le pase.

```
String cad;
cad = String.valueOf(1234); //cad = "1234"
cad = String.valueOf(-12.34); //cad = "-12.34"
cad = String.valueOf('C'); //cad = "C"
cad = String.valueOf(false); //cad = "false"
```

# Métodos simples para objetos String

Las cadenas son objetos en Java. Los métodos que veremos a continuación sólo se pueden invocar desde una instancia de cadena específica. Por esta razón, estos métodos se llaman métodos de instancia.

La sintaxis para invocar un método de instancia es

**variableReferenciada.nombreMetodo(argumentos)**.

**Comparar y comprobar Cadenas**

Puesto que String es una clase no podemos comparar con (==).
**boolean equals (String otraCadena)**
compara el contenido de la cadena que invoca el método y con el del parámetro otraCadena. El resultado de la comparación será true o false, según sean iguales o distintas.

# Métodos simples para objetos **String**

```
String cad1 = "Hola mundo";
String cad2 = "Hola mundo";
String cad3 = "Hola, buenos días"
boolean iguales;
iguales = cad1.equals(cad2); //iguales vale true
iguales = cad1.equals(cad3); //iguales vale false
```

Puede que queramos comparar sin tener en cuenta las mayúsculas y minúsculas. Para ello existe el método,

**boolean equalsIgnoreCase (String otraCadena)**

compara el contenido de la cadena que invoca el método y con el del parámetro otraCadena, pero sin distinguir mayúsculas de minúsculas.

```
String cad1 = "Hola mundo";
String cad2 = "HOLA Mundo";
boolean iguales;
iguales = cad1.equals(cad2); //false, no son iguales
iguales = cad1.equalsIgnoreCase(cad2); //true, sin atender a
                                       //mayúsculas/minúsculas son iguales
```

# Métodos simples para objetos **String**

Puede que queramos comparar alfabéticamente cadenas.
**int compareTo (String Cadena)**
devuelve **0** si las cadenas son exactamente iguales.
**negativo** si la cadena invocante es menor alfabéticamente que la cadena pasada como parámetro.
**positivo** si la cadena invocante es mayor alfabéticamente que la cadena pasada como parámetro.

```
String cad1 = "Alondra";
String cad2 = "Nutria";
String cad3 = "Zorro";
System.out.println(cad2.compareTo(cad1)) //valor mayor que 0
//"Nutria" está depués que "Alondra" alfabéticamente
System.out.println(cad2.compareTo(cad3)) //valor menor que 0
//"Nutria" está antes que "Zorro" alfabéticamente
```

**int compareToIgnoreCase (String Cadena)** comparar alfabéticamente cadenas sin distinguir entre letras mayúsculas y minúsculas.

# Métodos simples para objetos String

| Method | Description |
|---|---|
| `length()` | Devuelve el número de caracteres de esta cadena. |
| `charAt(index)` | Devuelve el carácter en el índice especificado de esta cadena. |
| `concat(s1)` | Devuelve una nueva cadena que concatena esta cadena con la cadena s1. |
| `toUpperCase()` | Devuelve una nueva cadena con todas las letras en mayúsculas. |
| `toLowerCase()` | Devuelve una nueva cadena con todas las letras en minúsculas. |
| `trim()` | Devuelve una nueva cadena con espacios en blanco recortados en ambos lados. |

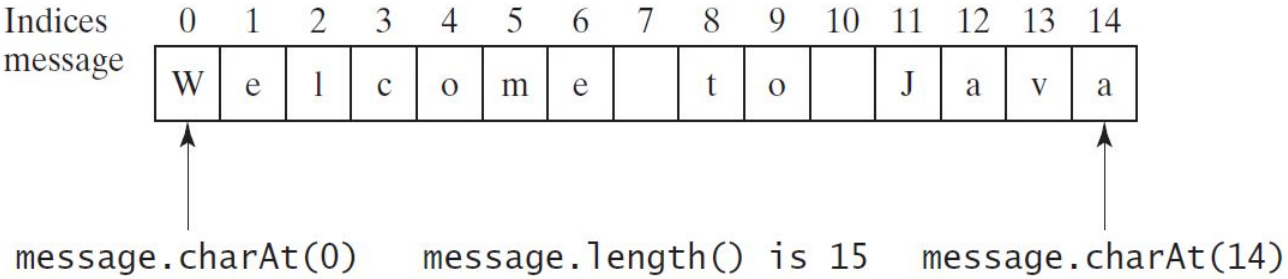
Ejemplo Getting String Length:

String mensaje = **“Bienvenido a Java"**;

System.out.println(**" La longitud de "** + mensaje + **" es "** + mensaje.length());

```
int longitud;
String cad1 = "Hola" , cad2 = "";
longitud = cad1.length(); //devuelve 4
longitud = cad1.length(); //devuelve 0
```

# Cómo obtener caracteres de una cadena




String message = "Welcome to Java";

System.out.println("The first character in message is " + message.charAt(0));

# Conversión de cadenas

"Welcome".toLowerCase() devuelve un nuevo string, welcome.

"Welcome".toUpperCase() devuelve un nuevo string, WELCOME.

"  Welcome  ".trim() devuelve un nuevo string, Welcome.

```
String cad1 = "Mi PeRrO: sE lLaMa PeRiCo23.";
String cad2;
cad2 = cad.toLowerCase(); //"mi perro: se llama perico23."
cad2 = cad.toUpperCase(); //"MI PERRO: SE LLAMA PERICO23."
```

# Concatenación de String

```
String s3 = s1.concat(s2); or String s3 = s1 + s2;

// Concatenar tres cadenas
String message = "Welcome " + "to " + "Java";

// El string Chapter se concatena con el número 2
String s = "Chapter" + 2;
      // s se convierte en Chapter2

// El string Supplement se concatena con el caracter B
String s1 = "Supplement" + 'B';
      // s1 se convierte en SupplementB
```

# Lectura de una cadena desde la consola

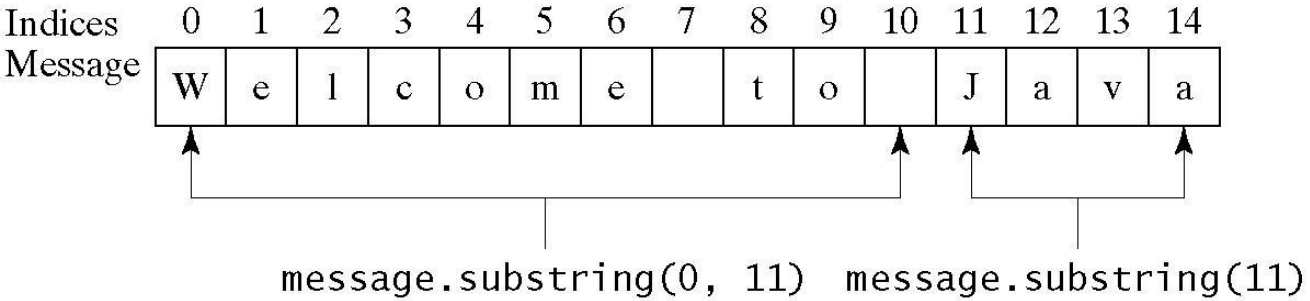
```
Scanner input = new Scanner(System.in);
System.out.print("Ingrese tres palabras separadas por espacios: ");
String s1 = input.next();
String s2 = input.next();
String s3 = input.next();
System.out.println("s1 is " + s1);
System.out.println("s2 is " + s2);
System.out.println("s3 is " + s3);
```

# Leyendo un carácter desde la consola

```
Scanner input = new Scanner(System.in);
System.out.print(“Escriba un caracter: ");
String s = input.nextLine();
char ch = s.charAt(0);
System.out.println(“El caracter introducido es" + ch);
```

# Obtención de subcadenas

| Method | Description |
|---|---|
| `substring(beginIndex)` | Devuelve la subcadena de esta cadena que comienza con el *beginIndex* y se extiende hasta el final de la cadena, como se muestra en la Figura 4.2. |
| `substring(beginIndex, endIndex)` | Devuelve la subcadena de esta cadena que comienza en el especificado `beginIndex` y se extiende hasta el carácter en el índice `endIndex - 1` como se muestra en la Figura 9.6 Tenga en cuenta que el carácter `endIndex` no forma parte de la subcadena. |

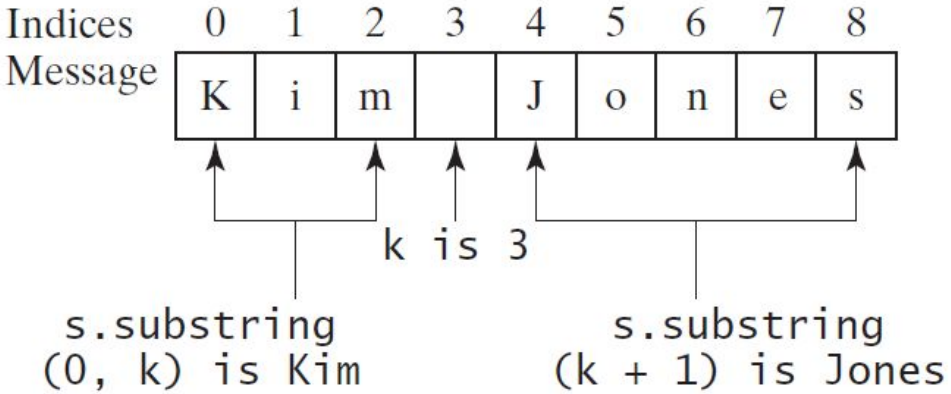


# Encontrar carácter o subcadena en una cadena

| Method | Description |
|---|---|
| `indexOf(ch)` | Devuelve el índice de la primera aparición de **ch** en la cadena. Devuelve **-1** si no coincide. |
| `indexOf(ch, fromIndex)` | Devuelve el índice de la primera aparición de **ch** después de **fromIndex** en la cadena. Devuelve **-1** si no coincide. |
| `indexOf(s)` | Devuelve el índice de la primera aparición de la cadena ***s*** en esta cadena. Devuelve **-1** si no coincide. |
| `indexOf(s, fromIndex)` | Devuelve el índice de la primera aparición de la cadena ***s*** en esta cadena después de Index. Devuelve **-1** si no coincide. |
| `lastIndexOf(ch)` | Devuelve el índice de la última aparición de **ch** en la cadena. Devuelve **-1** si no coincide. |
| `lastIndexOf(ch, fromIndex)` | Devuelve el índice de la última aparición de **ch** antes de **fromIndex** en esta cadena. Devuelve **-1** si no coincide. |
| `lastIndexOf(s)` | Devuelve el índice de la última aparición de cadena ***s***. Devuelve **-1** si no coincide. |
| `lastIndexOf(s, fromIndex)` | Devuelve el índice de la última aparición de cadena ***s*** antes de Índice. Devuelve **-1** si no coincide. |

# Encontrar carácter o subcadena en una cadena

```
int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
```



# Ejercicios

1. Escribe un programa que lea una cadena y dos números enteros (inicio y fin) y luego extraiga la subcadena desde la posición inicio hasta fin (no inclusivo).
2. Escribe un programa que lea dos cadenas e indique si la segunda cadena se encuentra dentro de la primera.
3. Escribe un programa que lea una cadena y la convierta a mayúsculas y minúsculas.
4. Escribe un programa que lea una cadena y un carácter a reemplazar y el nuevo carácter.
5. Escribe un programa que lea una cadena y un carácter, e indique la primera y última posición de dicho carácter en la cadena.
6. Escribe un programa que lea una cadena y un carácter, y cuente cuántas veces aparece el carácter en la cadena.

# Conversión entre cadenas y números

```
int intValue = Integer.parseInt(intString);
double doubleValue = Double.parseDouble(doubleString);

String s = number + "";
```

# Ejercicios

1. Escribe un programa que permita partir un número introducido por teclado en dos partes. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Suponemos que el usuario introduce correctamente los datos, es decir, el número introducido tiene dos dígitos como mínimo y la posición en la que se parte el número está entre 2 y la longitud del número.

   Ejemplo: *Por favor, introduzca un número entero positivo: 406783*
   *Introduzca la posición a partir de la cual quiere partir el número: 5*
   *Los números partidos son el 4067 y el 83*

2. Escribe un programa que sea capaz de insertar un dígito dentro de un número indicando la posición. El nuevo dígito se colocará en la posición indicada y el resto de dígitos se desplazará hacia la derecha. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Suponemos que el usuario introduce correctamente los datos. Se recomienda usar long en lugar de int ya que el primero admite números más largos.

   Ejemplo: *Por favor, introduzca un número entero positivo: 406783*
   *Introduzca la posición donde quiere insertar: 3*
   *Introduzca el dígito que quiere insertar: 5*
   *El número resultante es 4056783.*

3. Programa que inserte "<>" justo en la mitad de una cadena de caracteres desde teclado.
4. Programa que divida una palabra desde la primera A que encuentre.