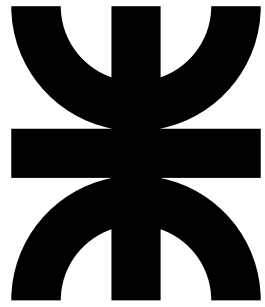
Informática II
Caracteres y cadenas

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2019 –

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

```
char cadena1[] = {67, 97, 100, 49, 0};
printf("%s", cadera1); // ¿Qué se imprime?
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

```
char cadena1[] = {67, 97, 100, 49, 0};
printf("%s", cadera1); // Imprime "Cad1"
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'};
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
char cadena4[] = {67, 97, 100, 52, 0}; // ASCII
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
char cadena4[] = {67, 97, 100, 52, 0}; // ASCII
char cadena5[] = "Cad5";
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
char cadena4[] = {67, 97, 100, 52, 0}; // ASCII
char cadena5[] = "Cad5";
char *cadena6 = "Cad6";
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
char cadena4[] = {67, 97, 100, 52, 0}; // ASCII
char cadena5[] = "Cad5";
char *cadena6 = "Cad6";
```

Imprimir

```
printf("La␣cadena␣es:␣%s\n", cadena6);
```

Cadenas

- ¿Existe el tipo de datos *string* (cadena) en C?
- ¿Cómo se almacenan/tratan las cadenas en C?

En lenguaje C las *cadenas* son arreglos de caracteres (ASCII)

Ejemplos:

```
char cadena1[] = {'C', 'a', 'd', '1', '\0'};
char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
char cadena3[4] = {'C', 'a', 'd', '3', '\0'}; // ERROR
char cadena4[] = {67, 97, 100, 52, 0}; // ASCII
char cadena5[] = "Cad5";
char *cadena6 = "Cad6";
```

Imprimir

```
printf("La␣cadena␣es:␣%s\n", cadena6);
```

- ASCII: American Standard Code for Information Interchange
- Constante de caracter: `'a'`, `'\0'`, `'\n'`, etc.

Arreglo de caracteres

Archivo `cadenas.c`

```c
#include <stdio.h>

int main(void)
{
  char cadena1[] = {'C', 'a', 'd', '1', '\0'};
  char cadena2[5] = {'C', 'a', 'd', '2', '\0'};
/* char cadena3[4] = {'C', 'a', 'd', '3', '\0'};*/
  char cadena4[] = {67, 97, 100, 52, 0};
  char cadena5[] = "Cad5";
  char *cadena6 = "Cad6";

  printf("Cadena␣1:␣%s\n", cadena1);
  printf("Cadena␣2:␣%s\n", cadena2);
/* printf("Cadena 3: %s\n", cadena3);*/
  printf("Cadena␣4:␣%s\n", cadena4);
  printf("Cadena␣5:␣%s\n", cadena5);
  printf("Cadena␣6:␣%s\n", cadena6);

  return 0;
}
```

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`
- Puede incluir letras, dígitos y caracteres especiales como `+`, `-`, `*`, `/`, `$`, etc.

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`
- Puede incluir letras, dígitos y caracteres especiales como `+`, `-`, `*`, `/`, `$`, etc.
- En C, las *literales* o *constantes* de cadenas se escriben en comillas dobles.

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`
- Puede incluir letras, dígitos y caracteres especiales como `+`, `-`, `*`, `/`, `$`, etc.
- En C, las *literales* o *constantes* de cadenas se escriben en comillas dobles.
- Se tiene acceso mediante el puntero al primer caracter.

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`
- Puede incluir letras, dígitos y caracteres especiales como `+`, `-`, `*`, `/`, `$`, etc.
- En C, las *literales* o *constantes* de cadenas se escriben en comillas dobles.
- Se tiene acceso mediante el puntero al primer caracter.

```
char color = "blue";
char *colorPtr = "blue";
```

- Un arreglo de caracteres debe tener el tamaño adecuado para contener la cadena mas el caracter de terminación

Cadenas en lenguaje C

Serie de caracteres tratados como una única unidad

- En C, una cadena es un arreglo de caracteres terminado con el caracter nulo `'\0'`
- Puede incluir letras, dígitos y caracteres especiales como `+`, `-`, `*`, `/`, `$`, etc.
- En C, las *literales* o *constantes* de cadenas se escriben en comillas dobles.
- Se tiene acceso mediante el puntero al primer caracter.

```
char color = "blue";
char *colorPtr = "blue";
```

- Un arreglo de caracteres debe tener el tamaño adecuado para contener la cadena mas el caracter de terminación

```
char cadena[20];
scanf("%s", cadena);
```

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`
- `int isxdigit(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`
- `int isxdigit(int c)`
- `int islower(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`
- `int isxdigit(int c)`
- `int islower(int c)`
- `int isupper(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`
- `int isxdigit(int c)`
- `int islower(int c)`
- `int isupper(int c)`
- `int tolower(int c)`

Biblioteca para el manejo de caracteres

- Archivo de cabecera `<ctype.h>`
- Incluye funciones para realizar pruebas o verificación y para la manipulación de datos tipo caracteres.

Algunas funciones son:

- `int isdigit(int c)`
- `int isalpha(int c)`
- `int isalnum(int c)`
- `int isxdigit(int c)`
- `int islower(int c)`
- `int isupper(int c)`
- `int tolower(int c)`
- `int toupper(int c)`

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Algunas funciones son:

- `double atof(const char *nptr)`

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Algunas funciones son:

- `double atof(const char *nptr)`
- `int atoi(const char *nptr)`

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Algunas funciones son:

- `double atof(const char *nptr)`
- `int atoi(const char *nptr)`
- `long atol(const char *nptr)`

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Algunas funciones son:

- `double atof(const char *nptr)`
- `int atoi(const char *nptr)`
- `long atol(const char *nptr)`
- `float strtof(const char *nptr, char **endptr)`

Funciones de conversión de cadenas

- Archivo de cabecera `<stdlib.h>` (biblioteca general de utilería).
- Convierte cadenas de dígitos a enteros y valores en punto flotante.

Algunas funciones son:

- `double atof(const char *nptr)`
- `int atoi(const char *nptr)`
- `long atol(const char *nptr)`
- `float strtof(const char *nptr, char **endptr)`
- `double strtod(const char *nptr, char **endptr)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strcat(char *dest, const char *src)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strcat(char *dest, const char *src)`
- `char *strncat(char *dest, const char *src, size_t n)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strcat(char *dest, const char *src)`
- `char *strncat(char *dest, const char *src, size_t n)`
- `char *strcmp(const char *s1, const char *s2)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strcat(char *dest, const char *src)`
- `char *strncat(char *dest, const char *src, size_t n)`
- `char *strcmp(const char *s1, const char *s2)`
- `char *strncmp(const char *s1, const char *s2, size_t n)`

Manipulación y comparación de cadenas

- Archivo de cabecera `<string.h>`

Algunas funcione son:

- `char *strcpy(char *dest, const char *src)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strcat(char *dest, const char *src)`
- `char *strncat(char *dest, const char *src, size_t n)`
- `char *strcmp(const char *s1, const char *s2)`
- `char *strncmp(const char *s1, const char *s2, size_t n)`
- `size_t strlen(const char *s)`

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros
- Uso común: arreglos de cadenas

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros
- Uso común: arreglos de cadenas
  - Cada entrada del arreglo es una cadena

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros
- Uso común: arreglos de cadenas
  - Cada entrada del arreglo es una cadena
  - Cada entrada es un puntero al primer caracter de la cadena

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros
- Uso común: arreglos de cadenas
  - Cada entrada del arreglo es una cadena
  - Cada entrada es un puntero al primer caracter de la cadena

Ejemplo

```
char *key[4] = {"Top", "Down", "Left", "Right"};
```

Arreglos de punteros – Arreglos de cadenas

- Los arreglos pueden contener punteros
- Uso común: arreglos de cadenas
  - Cada entrada del arreglo es una cadena
  - Cada entrada es un puntero al primer caracter de la cadena

Ejemplo

```
char *key[4] = {"Top", "Down", "Left", "Right"};
```

¿Qué pasa si se almacenan las cadenas en un arreglo doble?

Ejemplo – Argumentos a la función `main`

```
int main(void)
{
  /* Programa */
  . . .
  return 0;
}
```

Ejemplo – Argumentos a la función `main`

```c
int main(void)
{
  /* Programa */
  . . .
  return 0;
}
```

```c
int main(int argc, char *argv[])
{
  /* Programa */
  . . .
  return 0;
}
```

Ejemplo – Argumentos a la función `main`

```c
int main(void)
{
  /* Programa */
  . . .
  return 0;
}
```

```c
int main(int argc, char *argv[])
{
  /* Programa */
  . . .
  return 0;
}
```

- `argc`: cantidad de argumentos en la línea de comandos al ejecutar el programa
- `argv`: puntero a un vector de cadenas que contiene los argumentos (`argv[argc]` es un puntero `NULL`)

Actividad práctica

Actividad práctica

1. Escribir un programa para evaluar las funciones de manejo de caracteres (archivo de cabecera `<ctype.h>`). La interacción con el usuario es la siguiente:
   - El programa debe solicitarle al usuario ingresar un caracter de forma continuada hasta precionar Ctrl-C (combinación de teclas Ctrl y C),
   - e imprimir una nueva línea por cada función a evaluar, p.e.
     `isdigit: SI`
     `isnum: NO`
     Etc.

Actividad práctica

1. Escribir un programa para evaluar las funciones de manejo de caracteres (archivo de cabecera `<ctype.h>`). La interacción con el usuario es la siguiente:
   - El programa debe solicitarle al usuario ingresar un caracter de forma continuada hasta precionar Ctrl-C (combinación de teclas Ctrl y C),
   - e imprimir una nueva línea por cada función a evaluar, p.e.
     `isdigit: SI`
     `isnum: NO`
     Etc.

2. Escribir un programa que imprima mediante un bucle `for` las cadenas pasadas a la función `main` y la cantidad de caracteres (largo) de cada cadena.

Actividad práctica

1. Escribir un programa para evaluar las funciones de manejo de caracteres (archivo de cabecera `<ctype.h>`). La interacción con el usuario es la siguiente:
   - El programa debe solicitarle al usuario ingresar un caracter de forma continuada hasta precionar Ctrl-C (combinación de teclas Ctrl y C),
   - e imprimir una nueva línea por cada función a evaluar, p.e.
     ```
     isdigit: SI
     isnum: NO
     ```
     Etc.

2. Escribir un programa que imprima mediante un bucle `for` las cadenas pasadas a la función `main` y la cantidad de caracteres (largo) de cada cadena.

3. Modificar el programa del ejercicio 1 para que el programa reciba por línea de comandos el caracter a evaluar, p.e.
   ```
   > ./a.out a
   ```
   Además, el programa debe verificar que se recibe un único caracter y en caso contrario indicar error.

Actividad práctica

4. Escribir un programa que realice la suma de dos enteros pasados desde la línea de comandos, y tenga la siguiente salida

```
> ./suma 10 12
10 + 12 = 22
> ./suma
ERROR. Debe ejecutarse. > ./suma <entero1> <entero2>
```

Actividad práctica

4. Escribir un programa que realice la suma de dos enteros pasados desde la línea de comandos, y tenga la siguiente salida

```
> ./suma 10 12
10 + 12 = 22
> ./suma
ERROR. Debe ejecutarse. > ./suma <entero1> <entero2>
```

5. Escribir un programa que reciba dos cadenas por línea de comandos y compare las cadenas utilizando la función `strcmp`, e indicar si la primer cadena es menor, igual o mayor que la segunda.

Actividad práctica

4. Escribir un programa que realice la suma de dos enteros pasados desde la línea de comandos, y tenga la siguiente salida

```
> ./suma 10 12
10 + 12 = 22
> ./suma
ERROR. Debe ejecutarse. > ./suma <entero1> <entero2>
```

5. Escribir un programa que reciba dos cadenas por línea de comandos y compare las cadenas utilizando la función `strcmp`, e indicar si la primer cadena es menor, igual o mayor que la segunda.

6. Escribir un programa que convierta a mayúscuas las cadenas pasadas por línea de comandos

```
> ./ejr6 hola 123
./EJR6
HOLA
123
```