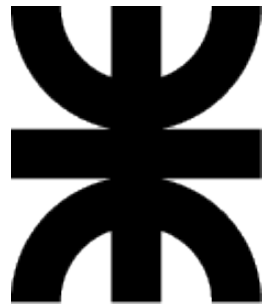
Informática II
Caracteres y cadenas

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

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?

```c
#include <stdio.h>

int main(void)
{
    char cadena[] = {67, 97, 100, 49, 0};
    printf("%s\n", cadena);
    return 0;
}
```

Cadenas

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

```c
#include <stdio.h>

int main(void)
{
    char cadena[] = {67, 97, 100, 49, 0};
    printf("%s\n", cadena);
    return 0;
}
```

- ¿Qué imprime en pantalla el programa anterior?

Cadenas

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

```c
#include <stdio.h>

int main(void)
{
    char cadena[] = {67, 97, 100, 49, 0};
    printf("%s\n", cadena);
    return 0;
}
```

- ¿Qué imprime en pantalla el programa anterior?
- Imprime `Cad1`

Cadenas

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

```c
#include <stdio.h>

int main(void)
{
    char cadena[] = {67, 97, 100, 49, 0};
    printf("%s\n", cadena);
    return 0;
}
```

- ¿Qué imprime en pantalla el programa anterior?
- Imprime `Cad1`

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

Cadenas

Ejemplos:

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

Cadenas

Ejemplos:

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

Cadenas

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

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

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

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

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

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

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 color1[] = "verde";
char *color2 = "azul";
```

- Un arreglo de caracteres debe tener el tamaño adecuado para contener la cadena más 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 color1[] = "verde";
char *color2 = "azul";
```

- Un arreglo de caracteres debe tener el tamaño adecuado para contener la cadena más 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)`

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)`

¿Qué significa `const char *nptr`?

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)`