Cadenas de Caracteres

Antonio Espín Herranz

Cadenas de caracteres

- Es un array de caracteres terminado en \0.
- El número total de caracteres de una cadena C es siempre igual a la longitud de la cadena más 1.
- Se puede ver declarado así: char *s;
- Puntero a carácter que todavía no tiene memoria asignada.

Inicialización de cadenas

char texto[81] = “esto es una cadena”;
char textodemo[255] = {“Esto es una cadena muy larga”};
char cadenaTest[] = {“longitud de cadena”};

- El compilador añade el ‘\0’, reservando para un carácter mas.
- unaCadena = “ABC”; // Da error.
- Para inicializar una cadena fuera de la definición, se utiliza la función strcpy().
- El nombre de la cadena, representa una dirección de memoria.

Lectura e Impresión de cadenas

- **Lectura**:
  - scanf: // Termina al encontrar un blanco o fin de línea.

    char nombre[20];

    scanf("%s", nombre);

    printf("%s \n", nombre);

  - gets: // Termina de leer con el salto de línea. Esta lee blancos.

    gets(nombre);

- **Escritura:**
  - puts: // Es la contraria a gets, imprime una cadena por la salida estándar.

    puts(nombre);

  - printf: recibe una cadena constante formateada y una serie de parámetros.

    Según vayamos a imprimir un tipo de datos u otro, utilizaremos un formato u otro. %c, %d, %f, %s, %u, %o, %x, %X, %p, %ld, %hd

Funciones getchar() / putchar()

- getchar(): leen carácter a carácter de la entrada estándar (stdin), en caso de error devuelve EOF (control + Z).

- putchar(): escribe en la salida estándar stdout.

Funciones getch() / getche()

- No pertenecen a ANSI C.
- Ambas leen un carácter de teclado.
- La diferencia es que con getch() el carácter no se visualiza y con getche(), se hace eco del carácter pulsado y se visualiza por pantalla.
- Se encuentran en conio.h (compiladores antiguos)

Funciones estándar sobre cadenas

- Para trabajar con cadena se incluirá el fichero de cabecera: string.h

- #include <string.h>

Asignación de cadenas

- char *strcpy(char *destino, const char *origen)
  - Copia origen en destino y añade el carácter nulo al final de la cadena: ‘\0’.
  - Ejemplo:

    char nombre[20];

    strcpy(nombre, “cadena a copiar”);
- char *strncpy(char *destino, const char *fuente, size_t num)
  - Copia origen en destino, los caracteres indicados.

Longitud y concatenación de cadenas

- size_t strlen(const char *cadena): Devuelve el número de caracteres de una cadena.
- char *strcat(char *destino, const char *fuente): Concatena en destino la fuente.
- char *strncat(char *destino, const char *fuente, size_t num): Concatena en destino la fuente, pero solo los caracteres indicados.

Comparación de cadenas

- int strcmp(const char *s1, const char *s2): Compara ambas cadenas. Devuelve 0 si son iguales, < 0 si s1 < s2, > 0 cuando s1 > s2.

- int strncmp(const char *s1, const char *s2, size_t num): Compara los num primeros números en ambas cadenas. Devuelve 0 si son iguales, < 0 si s1 < s2, > 0 cuando s1 > s2.

Comparación de cadenas II

- int stricmp(const char *s1, const char *s2): Compara ambas cadenas sin distinguir mayúsculas y minúsculas. Devuelve 0 si son iguales, < 0 si s1 < s2, > 0 cuando s1 > s2.

- int strnicmp(const char *s1, const char *s2, size_t num): Compara los num. primeros números en ambas cadenas sin distinguir mayúsculas y minúsculas. Devuelve 0 si son iguales, < 0 si s1 < s2, > 0 cuando s1 > s2.

Inversión de cadenas / Conversión de cadenas

- char *strrev(char *s): Invierte la cadena que recibe por argumento.
- char *strupr(char *s): Devuelve la cadena convertida en mayúsculas.
- char *strlwr(char *s): Devuelve en minúsculas la cadena recibida por argumento.

Conversión de cadenas a números

- int atoi(const char *cad): transforma a entero la cadena recibida, si no se puede transformar devuelve 0.
- double atof(const char *cad): idem de la anterior pero transforma a double.
- long atol(const char *cad): idem, transformando a largo.

Búsqueda de caracteres y cadenas

- char *strchr(const char *s, int c) Busca un carácter dentro de la cadena, devuelve un puntero a la primera ocurrencia de dicho carácter dentro de s.

- char *strstr(const char *s1, const char *s2) Busca una cadena dentro de otra cadena. Devuelve un puntero al primer carácter de s1 que coincida en s2, si no se encuentra devuelve null.

Cadenas como punteros

- Podemos utilizar punteros en lugar de índices para los arrays de caracteres.
- El nombre del array representa un puntero.
- Ejemplo:

  char alfabeto[27] = “ABCDEFGHIJKLMNOPQRSTUVWXYZ”;

  char *s; // Definimos un puntero a carácter.

  s = &alfabeto[0]; // También vale s = alfabeto;

Recorrer la cadena con el puntero

// Recorre el bucle while, hasta que se encuentra con el \0 que marca el final de la cadena de caracteres.

// *s++ → accede al contenido, es decir, va accediendo carácter a carácter y después incrementa el puntero y accede a la siguiente posición de memoria.

```
while (*s){
        printf("%c", *s++);
}
```

- También podemos definir así las cadenas:
  char *cadena = “Hola que tal”; // se hace la reserva de forma automática.

Definición de un array de punteros a carácter

- char *nombre_meses[12] =
  {
  “Enero”,
  “Febrero”,
  “Marzo”, …,
  “Noviembre”,
  “Diciembre”
  };
- char **p = nombre_meses;

Argumentos de main

- Se utilizan para pasar argumentos a nuestro programa.
- Se declaran en la función main.
- main(int argc, char *argv[])
- main (int argc, char **argv) → También es válida.
- Recibimos el número de argumentos y un array de cadena de caracteres con los argumentos.
- En la posición 0 se registra el nombre de nuestro programa.