Cadenas-fundamentos de programación

Mendoza Bautista Carlos Gabriel

219246329

En el lenguaje de programación C, una cadena o "string" es una secuencia de caracteres terminada en un carácter nulo ('\0'). Este carácter nulo marca el final de la cadena, lo que permite que las funciones que trabajan con cadenas determinen dónde termina la cadena en la memoria.

Declaración y Inicialización de Cadenas

Una cadena en C puede ser declarada de varias maneras, utilizando un arreglo de caracteres o un puntero a caracteres. Aquí algunos ejemplos:

```c
char str1[11] = "Hola Mundo";
char str2[] = "Hola Mundo";
char *str3 = "Hola Mundo";
```

En str1, se especifica el tamaño del arreglo, mientras que str2 infiere el tamaño automáticamente incluyendo el carácter nulo al final. str3 es un puntero que apunta a una cadena literal, la cual es inmutable.

Funciones Básicas de la Biblioteca <string.h>

La biblioteca <string.h> en C proporciona numerosas funciones para manejar cadenas. A continuación, se detallan algunas de las más utilizadas:

strlen(): Retorna la longitud de la cadena sin contar el carácter nulo.

strcpy(): Copia una cadena en otra.

strcat(): Concatena dos cadenas.

strcmp(): Compara dos cadenas lexicográficamente.

Ejemplo de uso de strlen, strcpy, y strcat:

```c
int main() {
    char str1[20] = "Hola";
    char str2[20] = " Mundo";
    char str3[40];

    printf("Longitud str1: %lu\n", strlen(str1));
    strcpy(str3, str1);             // Copia str1 en str3
    strcat(str3, str2);             // Concatena str2 en str3
    printf("str3: %s\n", str3);     // Muestra "Hola Mundo"

    return 0;
}
```

https://www.uco.es/grupos/eatco/informatica/metodologia/cadenasyarrays.pdf

Gestión de Memoria y Cadenas

Al trabajar con cadenas, es crucial gestionar correctamente la memoria para evitar errores como desbordamientos de búfer. Las funciones de copia y concatenación deben usarse con cuidado para asegurarse de que el destino tenga suficiente espacio para la cadena resultante.

Ejemplo de manejo cuidadoso de memoria:

```c
int main() {
    char src[50], dest[50];

    strcpy(src,  "Este es un ejemplo de src");
    strcpy(dest, "Dest");

    // Asegura que hay suficiente espacio en dest
    if (strlen(dest) + strlen(src) + 1 <= 50) {
        strcat(dest, src);
    } else {
        printf("No hay suficiente espacio en 'dest'\n");
    }

    printf("Cadena final: %s\n", dest);
    return 0;
}
```

Comparación y Búsqueda en Cadenas

Para comparar cadenas, strcmp() y strncmp() son esenciales, y para buscar subcadenas o caracteres individuales, funciones como strstr() y strchr() resultan muy útiles.

Ejemplo de comparación y búsqueda:

```c
int main() {
    char str1[] = "Hola Mundo";
    char str2[] = "Hola";

    // Comparación de cadenas
    if (strcmp(str1, str2) == 0) {
        printf("Las cadenas son iguales\n");
    } else {
        printf("Las cadenas son diferentes\n");
    }

    // Búsqueda de subcadena
    char *subcadena = strstr(str1, "Mundo");
    if (subcadena != NULL) {
        printf("Subcadena encontrada: %s\n", subcadena);
    }

    return 0;
}
```

Conclusiones

Las cadenas en C, gestionadas a través de arreglos de caracteres y funciones de la biblioteca <string.h>, son fundamentales en numerosas aplicaciones de programación. Es esencial entender cómo manejar estas cadenas de manera segura y eficiente, especialmente en lo que respecta a la gestión de la memoria y la manipulación de los datos.

La correcta utilización de las funciones proporcionadas por <string.h> facilita la realización de operaciones complejas con cadenas, permitiendo a los desarrolladores implementar soluciones robustas y eficientes en sus programas.