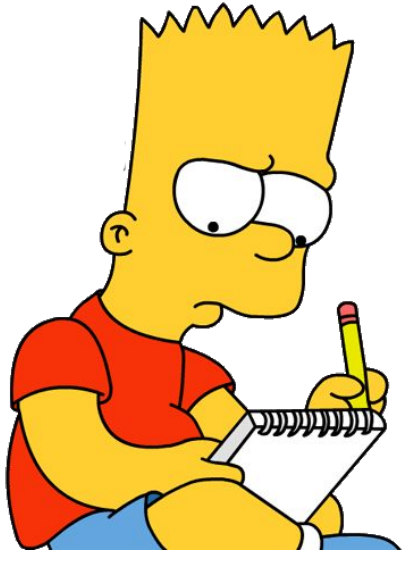
Strings



¿Qué es un String?

- Los “Strings” nos permiten representar texto en C
- C no tiene un tipo de dato nativo para representarlo



¿Cómo representamos una palabra en C?

- Lo más común es pensar en un **vector de caracteres:**

```
char vec_de_chars[MAX_VEC]= {‘H’, ‘o’, ‘l’, ‘a’};
```

Vectores de caracteres

```c
#include <stdio.h>

#define MAX_CARACTERES 50

int main() {
    char mi_vector_de_caracteres[MAX_CARACTERES] = {'H', 'o', 'l', 'a', ' ', 'M', 'u', 'n', 'd', 'o'};
    int tope = 10;
    printf("Contenido del Vector: ");

    for(int i = 0; i < tope; i++) {
        printf("%c", mi_vector_de_caracteres[i]);
    }
    printf("\n");

    return 0;
}
```

¿Cómo representamos una palabra en C?

- Lo más común es pensar en un **vector de caracteres:**

```
char vec_de_chars[MAX_VEC]= {‘H’, ‘o’, ‘l’, ‘a’};
```

- ¿Cómo se resuelve el tema del **tope**?



¿Cómo representamos una palabra en C?

- Si utilizamos un string:

```
char saludo[MAX_SALUDO]= {‘H’, ‘o’, ‘l’, ‘a’, ‘\0’};
```

- ¿Cómo se resuelve el tema del **tope**?

  Utilizamos el ‘\0’ para definir el final del mismo.



¿Qué es un String?



- Los “Strings” nos permiten representar texto en C
- C no tiene un tipo de dato nativo para representarlo
- Los “Strings” son un arreglo de caracteres
- Deben terminar con el “carácter nulo” (\0)

“Strings” vs “Vectores de caracteres”

```c
#include <stdio.h>

#define MAX_CARACTERES 50

int main() {
    char mi_string[MAX_CARACTERES] = "Hola Mundo\0";

    printf("Contenido del String: %s\n", mi_string);

    return 0;
}
```

“Strings” vs “Vectores de caracteres”

Si el texto entre “” se asigna al string completo no hace falta el ‘\0’

```c
#include <stdio.h>

#define MAX_CARACTERES 50

int main() {
    char mi_string[MAX_CARACTERES] = "Hola Mundo";

    printf("Contenido del String: %s\n", mi_string);

    return 0;
}
```

“Strings” vs “Vectores de caracteres”

- ¿Todos los vectores de caracteres son un string?

- ¿Los strings solo contienen caracteres?



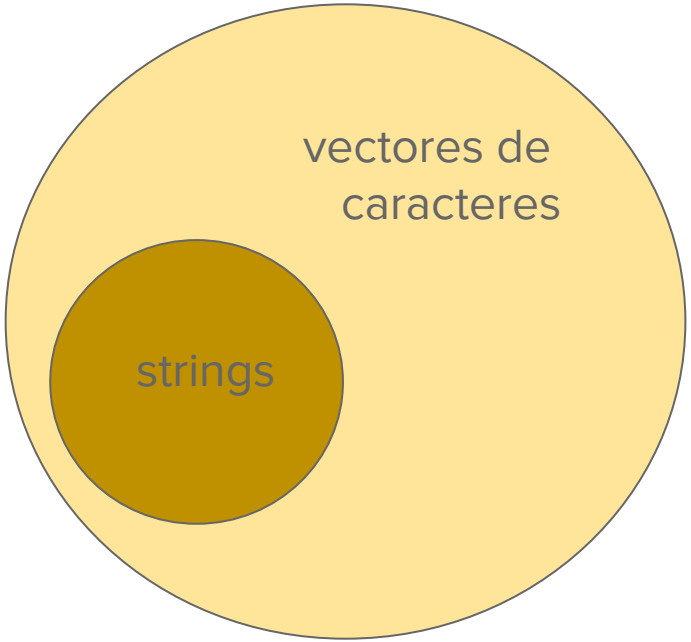
“Strings” vs “Vectores de caracteres”

- **¿Todos los vectores de caracteres son un string?**

No, los strings deben finalizar en `\0`

- **¿Los strings solo contienen caracteres?**

Si, `\0` es un carácter



Los “strings” son “vectores de caracteres”

Tienen las mismas limitaciones que los vectores:

- No se pueden asignar un string a otro
- No se pueden comparar un string con otro usando con los operadores normales
- **La asignación y comparación deben ser elemento a elemento.**

Biblioteca estándar de C: string.h

- C tiene una biblioteca para trabajar con strings: “**string.h**”
- Esta librería tiene fuertemente en consideración la presencia del carácter '\0' para la implementación de sus funciones
- Algunas funciones:
  - strcpy
  - strcmp
  - strlen
  - strcat



Asignación

char* strcpy(char dest[ ], char src[ ]): Copia el string ‘src’ a ‘dest’ . Asume que ‘src’ está bien formado y que se alojó en ‘dest’ memoria suficiente. Copia el carácter finalizador.

Comparación

int strcmp(char str1[ ], char str2[ ]): Compara dos strings carácter por carácter.

- Devuelve ‘0’ si los strings son iguales.
- Devuelve ‘>0’ si el primer carácter no coincidente en str1 es mayor que el de str2.
- Devuelve ‘<0’ si el primer carácter no coincidente en str1 es menor que el de str2.

Otros

unsigned int strlen(char str[]): Devuelve el largo de un string hasta el carácter delimitador (\0), sin contar a este último.

char* strcat(char dst[ ], char src[ ]): Concatena una copia de la cadena ‘src’, al final de la cadena ‘dst’.

Resumen: Destripando ‘string.h’

char* strcpy(char dest[ ], char src[ ]): Copia el string ‘src’ a ‘dest’ . Asume que ‘src’ está bien formado y que se alojó en ‘dest’ memoria suficiente. Copia el carácter finalizador.

int strcmp(char str1[ ], char str2[ ]): Compara dos strings carácter por carácter.

- Devuelve ‘0’ si los strings son iguales.
- Devuelve ‘>0’ si el primer carácter no coincidente en str1 es mayor que el de str2.
- Devuelve ‘<0’ si el primer carácter no coincidente en str1 es menor que el de str2.

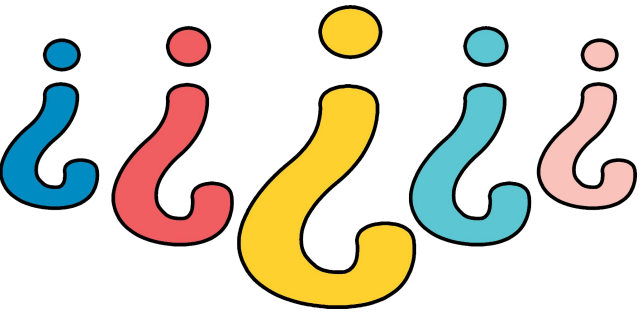
unsigned int strlen(char str[]): Devuelve el largo de un string hasta el carácter delimitador (\0), sin contar a este último.

char* strcat(char dst[ ], char src[ ]): Concatena una copia de la cadena ‘src’, al final de la cadena ‘dst’.



Veamos en código si esto es verdad...





¿Preguntas?



Fin

