STRINGS

Strings (Cadena de caracteres)

- En C no existe un tipo de dato y operadores para el tratamiento de cadenas de caracteres de manera “atómica”

- Un string es una cadena de cero o más caracteres finalizados en cero '0', NUL o carácter '\0' para detectar el final del string

- Declaración de na string
  char saludo[5];

- Asignación
  - en la declaración:

```
char saludo[5] = “hola”;      // termina en NUL
char saludo[5] = {'h','o','l','a','\0'};
```

  - en otro tipo de sentencias usando funciones de librería (string.h)

Error:

```
saludo = “chau”;
```

Strings – Visualización

- Mediante **printf** de **stdio.h**

  ```
  int printf(const char *format, ...);
  ```

  empleando el modificado de formato **%s**

  ```
  char cadena[9]="sin tres";
  . . .
  printf("No hay dos %s", cadena);
  ```

- El formato

  %20s: texto a la derecha, campo de 20 caracteres

  %-15s: texto a la izquierda, campo de 15 caracteres

```
char saludo[5]="hola";
printf("Saludo: %s-mundo\n", saludo);
printf("Saludo: %20s-mundo\n", saludo);
printf("Saludo: %-15s-mundo\n", saludo);
```

```
Saludo: hola-mundo
Saludo:                hola-mundo
Saludo: hola           -mundo
```

- **Strings – Funciones de string.h**
  - Establecer contenido al string

    `char *strcpy (char *destino, char const *origen);`
  - Concatenar

    `char *strcat (char *destino, char const *origen);`

    Es responsabilidad del programador asegurar que
    - la región de memoria destino debe ser suficientemente grande
    - las regiones de memoria destino y origen no deben superponerse

```
#include <stdio.h>
#include <string.h>

int main() {
  char full[80];
  char nombres[32]="Roberto";
  char apellidos [32]="Gomez Bolanos";
  strcpy(full, nombre);       /* full <- "Roberto" */
  strcat(full, " ");          /* full <- "Roberto " */
  strcat(full, apellidos);    /* full <- "Roberto Gomez Bolanos" */
  printf ("Nombre completo: %s\n", full);
  return 0;
}
```

- **Strings – Funciones de string.h**
  - Obtener longitud

    `size_t strlen (const char *s);` // no considera el '\0'
  - Comparar cadenas

    `int strcmp (const char *s1, const char *s2);`

    (retorna >0 si s1>s2, ==0 si s1==s2 y <0 si s1<s2). Ver **`strcasecmp`**

```
#include <stdio.h>
#include <string.h>

int main() {
  char nombre[]="Julio";
  char mes[]="julio";
  char anyo[]="2012";
  printf("Largo de %s = %d\n", nombre, strlen(nombre));
  printf("Largo de %s = %d\n", anyo, strlen(anyo));
  printf("%s y %s son ", nombre, mes);
  if (strcmp(nombre, mes)==0) printf("iguales\n");
  else                        printf("diferentes\n");

  return 0;
}
```

- **Strings – Funciones de string.h**
  - Búsqueda de caracteres y patrones

    `char *strchr (const char *s, int c)`
    Retorna un puntero a la primer ocurrencia del caracter **`c`**

    `char *strrchr (const char *s, int c)`
    Retorna un puntero a la última ocurrencia del caracter **`c`**

    `char *strstr(const char *string, const char *substring)`
    Busca la primer ocurrencia de **`substring`** en **`string`**

    `char *strcasestr(const char *string, const char *substring)`
    Igual que **`strstr`** pero ignora mayúsculas y minúsculas

- **Strings – Funciones de string.h**
  - Memory Functions
    A diferencia de las funciones str__ no finalizan al detectar NUL

    `void *memcpy (void *dest, const void *ori, size_t cant)`
    Copia **`cant`** bytes de un área de memoria a otra (no deben superponerse)
    Retorna un puntero a **`dest`**

    `void *memcmp (const void *a, const void *b, size_t cant)`
    Comportamiento similar a **`strcmop`**

    `void *memmove (void *dest, const void *ori, size_t cant)`
    Copia cant bytes desde **`ori`** a **`dest`**. Las direcciones de memoria pueden superponerse. Retorna un puntero a **`dest`**.

- **Strings – Funciones de string.h**
  - Recursos web

    http://en.wikipedia.org/wiki/C_string_handling

    http://www.acm.uiuc.edu/webmonkeys/book/c_guide/2.14.html

    http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html#string.h

Strings – Ejercicio

- Crear un programa que reciba un texto de no más de 200 caracteres

- Analizar el texto para que ofrezca las siguientes estdísticas
  - Cantidad de palabras (una palabra consiste en uno o más caracteres delimitada por ppio de lineas, espacios, tabuladores, fin de línea y otros signos de puntuación comunes)
  - Crear un histograma con las longitudes de las palabras
  - Cantidad de vocales
  - Cantidad de palíndromos (no case sensitive, Ana, Nuequen, salas)
  - Cantidad de palabras de más de x caracteres
  - Invertir aquellas palabras terminadas en vocal
  - Detectar en cuantas ocasiones aparece un patrón suministrado por el usuario (no case sensitive)

  En principio no importa la cantidad de veces que se parsee el texto