Introducción

Strings

Un dato tipo `string` es un arreglo de caracteres en el que la última posición es el caracter nulo: `\0` que indica el final del `string`.

Por lo anterior, en C el valor `'a'` y `"a"` son diferentes. El primero es un tipo `char` y el segundo es un tipo `string`.

Ejemplo de definición y declaración de un string:

```
#include<stdio.h>
main(){
    char string[] = "Hola, este es un string";
}
```

Lo anterior es similar para un arreglo de enteros:

```
#include<stdio.h>
main(){
    int arreglo[] = {2, 3, -1};
}
```

Podemos imprimir el string completo con el format specifier `%s` en printf:

```
#include<stdio.h>
main(){
    char string[] = "Hola, este es un string";
    printf("string: %s\n", string);
}
```

Algo que no es posible con arreglos de enteros.

Como se vio anteriormente, el nombre `string` es un apuntador. Entonces:

```
#include<stdio.h>
main(){
    char string[] = "Hola, este es un string";
    printf("*string: %c\n", *string);
    printf("*(string+1): %c\n", *(string+1));
}
```

Podemos utilizar funciones del archivo `string.h` para copiar string a string2, obtener la longitud del string:

```
#include<stdio.h>
#include<string.h>
main(){
    char string[]="Hola, este es un string";
    char string2[100];
    int longitud_string;
    strcpy(string2, string);
    printf("string2: %s\n", string2);
    longitud_string = strlen(string);
    printf("Longitud de string: %d\n",longitud_string);
    longitud_string = strlen(string2);
    printf("Longitud de string2: %d\n",longitud_string);
}
```

Observa que no es posible realizar la asignación:

```
#include<stdio.h>
main(){
    char string[]="Hola, este es un string";
    char string2[100];
    string2 = string;
}
```

Pero otra forma de hacer una copia para un string, es con un apuntador:

```
#include<stdio.h>
main(){
    char string[]="Hola, este es un string";
    char *string2;
    string2 = string;
    printf("string2: %s\n", string2);
}
```

Y podemos calcular la longitud del string2:

```
#include<stdio.h>
#include<string.h>
main(){
    char string[]="Hola, este es un string";
    char *string2;
    int longitud;
    string2 = string;
    longitud = strlen(string2);
    printf("longitud de string2: %d\n", longitud);
}
```

Como se mencionó al inicio, un string es un arreglo en el que la última posición se utiliza para guardar el caracter `\0` :

```
#include<stdio.h>
#include<string.h>
main(){
    char string[] = "Hola, este es un string";
    int i=0;
    int longitud_string = strlen(string);
    char *string2;
    string2 = string;
    for(i=0;i<longitud_string;i++)
        string2++;
    if(*string2 == '\0')
        printf("Caracter nulo\n");
    else
        printf("Valor de *string2: %c\n", *string2);
}
```

El último `if ... else` podría haber sido:

```
#include<stdio.h>
#include<string.h>
main(){
    char string[] = "Hola, este es un string";
    int i=0;
    int longitud_string = strlen(string);
    char *string2;
```

```
    string2 = string;
    for(i=0;i<longitud_string;i++)
        string2++;
    if(*string2) //evalúa si *string2 es diferente de nulo
        printf("Valor de *string2: %c\n", *string2);
    else
        printf("Caracter nulo\n");
}
```

Una definición y declaración muy utilizada es de la forma `*arreglo[n]` con `n` un valor positivo constante y entero. Se obtiene un arreglo de tamaño `n` de apuntadores:

```
#include<stdio.h>
main(){
    int *arreglo[4]; //arreglo de tamaño 4, en cada entrada se tiene
                     //un apuntador a un tipo de dato int
}
```

Debemos inicializar a los apuntadores del arreglo. Una forma es por ejemplo con un arreglo multidimensional:

```
#include<stdio.h>
main(){
    int *arreglo[4]; //arreglo de tamaño 4, en cada entrada se tiene
                     //un apuntador a un tipo de dato int
    int arr_mult[4][2];
    int i;
    int longitud_arreglo = sizeof(arreglo)/sizeof(arreglo[0]);
    //Inicializamos el arreglo de apuntadores
    for(i=0;i<longitud_arreglo;i++)
        arreglo[i] = arr_mult[i]; //recordamos que arr_mult es un arreglo de tamaño 4. En
                                  //cada entrada tenemos un arreglo de tamaño 2.
    //Damos valores a arr_mult
    for(i=0;i<longitud_arreglo;i++)
            arr_mult[i][0] = i;
    printf("Imprimimos el arreglo:\n");
    for(i=0;i<longitud_arreglo;i++)
        printf("arreglo[%d]=%d\n",i, *arreglo[i]);
}
```

Así, podríamos haber dado valores a `arr_mult` a partir de `arreglo`:

```
#include<stdio.h>
main(){
    int *arreglo[4]; //arreglo de tamaño 4, en cada entrada se tiene
                     //un apuntador a un tipo de dato int
    int arr_mult[4][2];
    int i, j;
    int longitud_arreglo = sizeof(arreglo)/sizeof(arreglo[0]);
    int num_columnas = sizeof(arr_mult[0])/sizeof(arr_mult[0][0]);
    printf("num_columnas: %d\n", num_columnas);
    //Inicializamos el arreglo de apuntadores
    for(i=0;i<longitud_arreglo;i++)
        arreglo[i] = arr_mult[i];//recordamos que arr_mult es un arreglo de tamaño 4. En
                                 //cada entrada tenemos un arreglo de tamaño 2.
    //Damos valores a arreglo usando dereference
    for(i=0;i<longitud_arreglo;i++)
        for(j=0;j<num_columnas;j++)
```

```
            *(arreglo[i]+j) = i+j;
    printf("Imprimimos arr_mult:\n");
    for(i=0;i<longitud_arreglo;i++){
        for(j=0;j<num_columnas;j++)
            printf("arr_mult[%d][%d] = %d\t", i,j,arr_mult[i][j]);
        printf("\n");
    }
    printf("Imprimimos a arreglo:\n");
    for(i=0;i<longitud_arreglo;i++){
        for(j=0;j<num_columnas;j++)
            printf("*(arreglo[%d] + %d) = %d\t", i,j,*(arr_mult[i] + j));
        printf("\n");
    }
}
```

La ventaja de usar en el ejemplo anterior a `*arreglo[4]` es que en la definición y declaración de este arreglo de apuntadores, sólo se alojan 4 apuntadores y no se hace explícito el número de columnas. Entonces los renglones pueden ser de diferente longitud:

```
#include<stdio.h>
main(){
    int *arreglo[4];
    int arr_mult[3][2];
    int arr_mult2[2][3];
    //Inicializamos a *arreglo:
    arreglo[0] = arr_mult[0];
    arreglo[1] = arr_mult2[1];
    arreglo[2] = arr_mult[2];

    //Algunos valores para arr_mult:
    arr_mult[0][0] = -10;
    arr_mult[0][1] = 2;
    arr_mult[2][0] = 7;
    arr_mult[2][1] = 5;
    //Algunos valores para arr_mult2:
    arr_mult2[1][0] = 4;
    arr_mult2[1][1] = 5;
    arr_mult2[1][2] = 6;
    //imprimimos algunas entradas de arreglo:
    printf("*(arreglo[0]) : %d\n", *arreglo[0]);
    printf("*(arreglo[0] + 1): %d\n", *(arreglo[0]+1));
    printf("*(arreglo[1]) : %d\n", *arreglo[1]);
    printf("*(arreglo[1] +1 ) : %d\n", *(arreglo[1] +1));
    printf("*(arreglo[1] +2 ) : %d\n", *(arreglo[1] +2));
    //Otra forma de imprimir entradas de arreglo:
    printf("arreglo[2][0] : %d\n", arreglo[2][0]);
    printf("arreglo[2][1] : %d\n", arreglo[2][1]);
}
```

Observa en la última forma de impresión que es equivalente para un apuntador inicializado realizar `*apuntador` que `apuntador[0]` o `*(apuntador + 2)` que `apuntador[2]` siempre que esté inicializado `apuntador + 2`.

```
#include<stdio.h>
main(){

    int *apuntador;
```

```
    int variable1 = -10;
    int variable2 = 5;
    apuntador = &variable1;
    printf("apuntador[0]: %d\n", apuntador[0]);
    apuntador = apuntador + 2;
    apuntador = &variable2;
    apuntador = apuntador - 2;
    printf("apuntador[2] : %d\n", apuntador[2]);

}
```

La precedencia del operador `[]` es mayor a la del operador `*`. Si ponemos paréntesis como en el ejemplo siguiente, tenemos un apuntador a un arreglo de tamaño 2 de enteros

```
#include<stdio.h>
main(){
    int (*apuntador)[2]; //apuntador a un arreglo de tamaño 2
    int arr_mult[3][2];
    apuntador = arr_mult +1 ;
    arr_mult[1][0] = -4;
    arr_mult[1][1] = 2;
    printf("(*apuntador)[0] :%d\n", (*apuntador)[0]);
    printf("(*apuntador)[1] : %d\n", (*apuntador)[1]);
}
```

Observa que genera un warning si en la asignación de arreglo se escribe:

```
arreglo = arr_mult[1];
```

pero al ejecutar el `.out` obtenemos lo deseado.

También podemos definir y declarar un arreglo de apuntadores a tipo de datos `char`:

```
#include<stdio.h>
main(){
    char *arreglo_str[4];
    arreglo_str[0] = "Hola,\n";
    arreglo_str[1] = "este es un\n";
    arreglo_str[2] = "ejemplo\t";
    arreglo_str[3] = "de un arreglo de apuntadores\n";
    printf("arreglo_str[0] : %s\n", arreglo_str[0]);
    printf("*arreglo_str : %s\n", *arreglo_str);
    printf("**arreglo_str: %c\n", **arreglo_str);
}
```

O bien con la inicialización:

```
#include<stdio.h>
main(){
    char *arreglo_str[] = {"Hola\n",
                           "este es un\n",
                           "ejemplo\t",
                           "de un arreglo de apuntadores\n"
                           };

    printf("*arreglo_str: %s\n", *arreglo_str);
    printf("*(arreglo_str+1) : %s\n", *(arreglo_str+1));
    printf("*(*(arreglo_str+1)+2) : %c\n", *(*(arreglo_str+1)+2));
```

```
}
```

```
#include<stdio.h>
main(){
    char arreglo_char_mult[][100] = {{"Hola,\n"},
                                     {"este es un\n"},
                                     {"ejemplo\t"},
    {"de un arreglo multidimensional de chars\n"}};
    printf("arreglo_char_mult[0]: %s\n", arreglo_char_mult[0]);
    printf("arreglo_char_mult[0][0]: %c\n", arreglo_char_mult[0][0]);
}
```