



Programación en C

Arreglos bidimensionales (matrices)

Matrices

- Los arreglos son espacios en una dimensión.
- Los arreglos pueden ser bidimensionales y de esta forma pueden contener matrices.
- Dependiendo las dimensiones necesarias son los corchetes q se ponen y en cada corchete el tamaño. Por ejemplo una matriz de 2x2:

```
int matriz[2][2];
```

Matrices

```
int matriz[2][2]= {1,2,3,4};
```

Posiciones
de los
elementos

0,0	0,1
1,0	1,1

Datos que
guardan
cada
elemento de
la matriz

1	2
3	4

```
int matriz[3][3]= {1,2,3,4,5,6,7,8,9};
```

Posiciones
de los
elementos

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Datos que
guardan
cada
elemento de
la matriz

1	2	3
4	5	6
7	8	9

Recorriendo arreglos y matrices

- Para obtener, leer o modificar los elementos de un **arreglo** se usa **1 ciclo for**.
- Para obtener, leer o modificar los elementos de una **matriz** se usan **2 ciclos for**, uno recorre renglones y el otro columnas.
- Aunque sean 2 corchetes NO son 2 variables, sigue siendo 1 variable y por tanto se coloca un especificador de conversión para leer o imprimir.

Recorriendo una matriz

```
int matriz[2][2]= {1,2,3,4};  
  
for(renglon=0; renglon<2; renglon++)  
{  
    for(column=0; column<2; column++)  
    {  
        printf("%d", matriz[renglon][column]);  
    }  
}
```

Operaciones sobre matrices

OPERACIÓN	VARIABLE	MATRIZ
Asignación de valor	<code>x= 0;</code>	<code>matriz[0][0]= 0;</code>
Comparación	<code>if(x==0)</code>	<code>if(matriz[0][0]==0) neutro++; else if(matriz[0][0]>0) pos++; else if(matriz[0][0]<0) neg++;</code>
Multiplicación	<code>x= x * 5;</code>	<code>matriz[0][0]= matriz[0][0] * 5;</code>

Hecho por Huicho :)

Multiplicación de matrices

- La multiplicación NO se hace elemento a elemento como la suma.
- El algoritmo correcto es:

```
matriz3[0][0]= matriz1[0][0]*matriz2[0][0] + matriz1[0][1]*matriz2[1][0];
```

$$19 = 1 * 5 + 2 * 7 \rightarrow 19 = 5 + 14$$

matriz1		matriz2		matriz3	
0,0	0,1	0,0	0,1	0,0	0,1
1,0	1,1	1,0	1,1	1,0	1,1
1	2	5	6	19	22
3	4	7	8	43	50

Programación en C

Arreglos unidimensionales de cadenas de caracteres

Cadenas

- Secuencia de caracteres terminada por el carácter nulo '\0'
- Una cadena siempre termina con el carácter nulo '\0' y los arreglos de caracteres no.
- Al **declarar e inicializar una cadena de caracteres con corchetes vacíos** se reservarán los cuadritos necesarios para cada carácter y uno más al final donde se **asigna automáticamente** el de terminación **nulo '\0'**.
- Si al declarar el arreglo se indican los cuadritos que se creen necesarios **tener en cuenta un cuadro adicional para el carácter de terminación, de no ser así al imprimir mostrará la cadena y lo que siga en memoria hasta encontrar un carácter de terminación.**

Cadenas

```
#include <stdio.h> //printf y scanf

int main(void)
{
    //en las cadenas debe asignarse un caracter de terminacion '\0'
    char m1[]= "hola"; //al no darle tamaño reserva 5 espacios
    char m2[2]= "hi"; //solo reserva para hi y no para el de terminacion
    char m3[3]= "hi"; //reserva manualmente para hi y terminacion

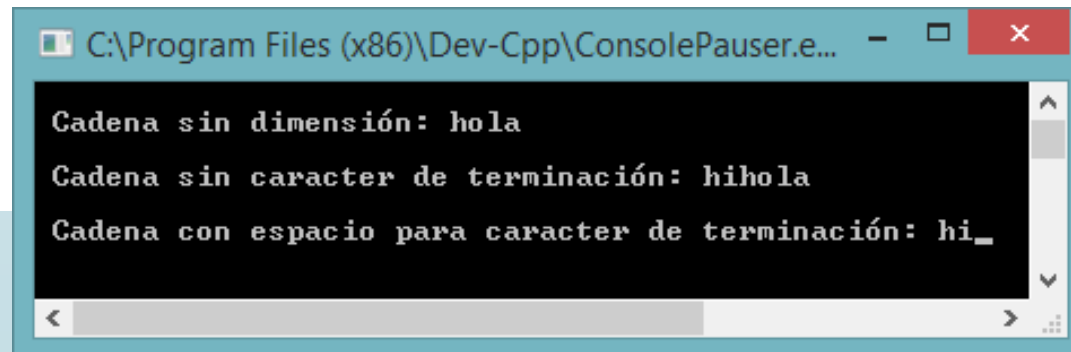
    printf("\n Cadena sin dimensi%cn: %s", 162, m1);

    printf("\n\n Cadena sin caracter de terminaci%cn: %s", 162, m2);

    printf("\n\n Cadena con espacio para caracter de terminaci%cn: %s", 162, m3);

    getchar();
    getchar();
    return 0;
}
```

Hecho por Huicho :)



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.e...
Cadena sin dimensi3n: hola
Cadena sin caracter de terminaci3n: hihola
Cadena con espacio para caracter de terminaci3n: hi_
```

Cadenas inicializadas

- Cuando el arreglo de caracteres se declara **reservando más cuadritos de los necesarios** y es **inicializado con una cadena**, los cuadritos se ocupan carácter por carácter y **al final se asigna el carácter nulo '\0' así como al resto de los cuadritos no usados que fueron reservados**.
- Para comprobar el **contenido** de los cuadritos de la cadena se puede **recorrer cuadrilo por cuadrilo** con un ciclo desde el índice 0 hasta el índice tamaño-1 **imprimiendo el código ascii** con el especificador %d.

Cadenas inicializadas

```
#include <stdio.h> //printf y scanf

int main(void)
{
    // en las cadenas se asigna un caracter de terminacion '\0'
    char mensaje[10]= "hi"; //reserva 10 cuadritos y ocupa 3 espacios
    int conde=0;

    // impresión de la cadena hasta '\0' con %s
    printf("\n Cadena: %s", mensaje);

    // impresión de la cadena cuadrado por cuadrado
    // desde indice 0 hasta tamaño-1
    // se usa %d para imprimir código ascii de los char almacenados
    printf("\n\n Contenido en ascii de cada cuadrado: \n\n");
    for(conde=0; conde<=9; conde++)
        printf(" %d", mensaje[conde]);

    getchar();
    getchar();
    return 0;
}
```

```
C:\Program Files (x86)\Dev-C...
Cadena: hi

Contenido en ascii de cada cuadrado:

104 105 0 0 0 0 0 0 0 0_
```

Hecho por Huicho :)

	Arreglo mensaje
[0]	104 / 'h'
[1]	105 / 'i'
[2]	0 / '\0'
[3]	0
[4]	0
[5]	0
[6]	0
[7]	0
[8]	0
[9]	0

Lectura de carácter

`getch (RECIBE void) → int` `conio.h`

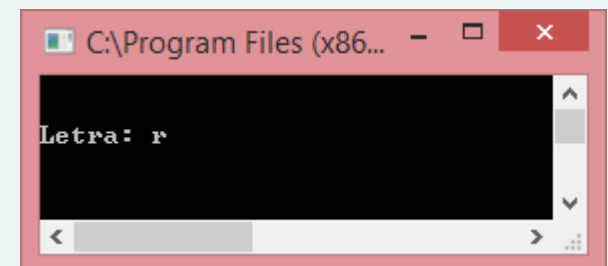
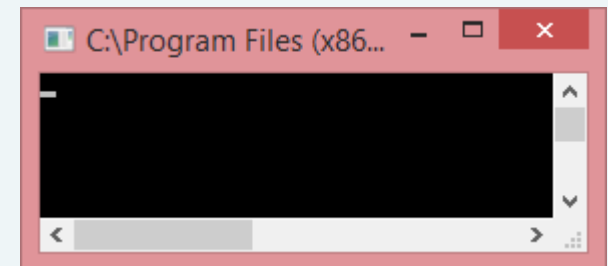
Sirve para leer un carácter del flujo de entrada sin que aparezca en pantalla. Se la llama sin argumentos y devuelve un entero.

```
#include<stdio.h>

int main(int argc, char* argv[])
{
    char letra= '\0';

    letra= getch();
    printf("\n\nLetra: %c", letra);

    getchar();
    return 0;
}
```



Lectura de carácter

`getchar (RECIBE void) → int stdio.h`

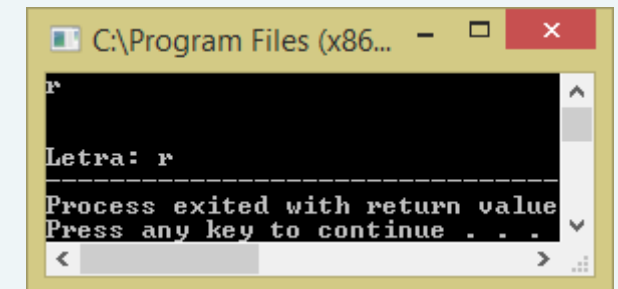
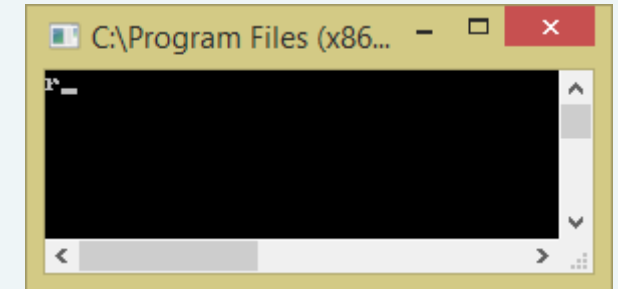
Sirve para leer el siguiente carácter del flujo de entrada. Se la llama sin argumentos y devuelve un entero.

```
#include<stdio.h>

int main(int argc, char* argv[])
{
    char letra= '\0';

    letra= getchar();
    printf("\n\nLetra: %c", letra);

    getchar();
    return 0;
}
```



Lectura de cadena (scanf)

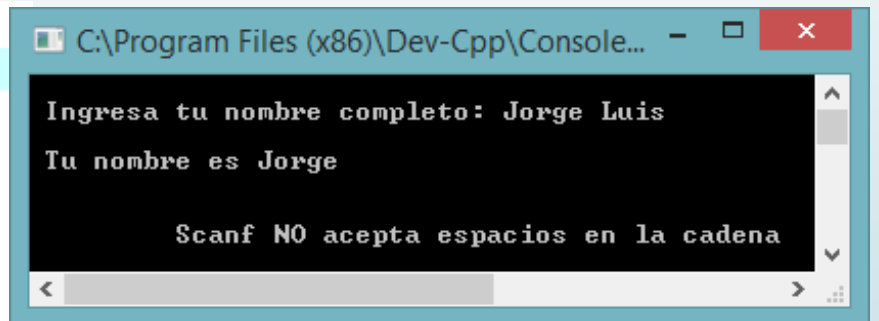
scanf: (RECIBE cadena) → int stdio.h

En el caso de cadenas leerá únicamente hasta donde encuentre un espacio en blanco ignorando todo el demás contenido de dicha cadena, quedando basura en el buffer de lectura y esto lo toma como otra entrada del teclado por cada espacio en blanco que encuentre. Devuelve un entero con los datos leídos.

```
#include <stdio.h> //para scanf, printf, getchar

int main(int argc, char* argv[]) //función principal
{
    char nombre[30]; //se piden 30 cuadritos para arreglo

    printf("\n Ingresa tu nombre completo: ");
    //Leemos la cadena con la función scanf, no lleva "&"
    scanf("%s", nombre);
    printf("\n Tu nombre es %s", nombre);
    printf("\n\n\n\t Scanf no acepta espacios en la cadena");
    fflush(stdin); //borra el buffer del teclado
    getchar(); //mantiene la pantalla estatica
    return 0; //termina la ejecucion devolviendo cero
}
```



```
C:\Program Files (x86)\Dev-Cpp\Console...
Ingresa tu nombre completo: Jorge Luis
Tu nombre es Jorge

Scanf NO acepta espacios en la cadena
```

Lectura de cadena (gets)

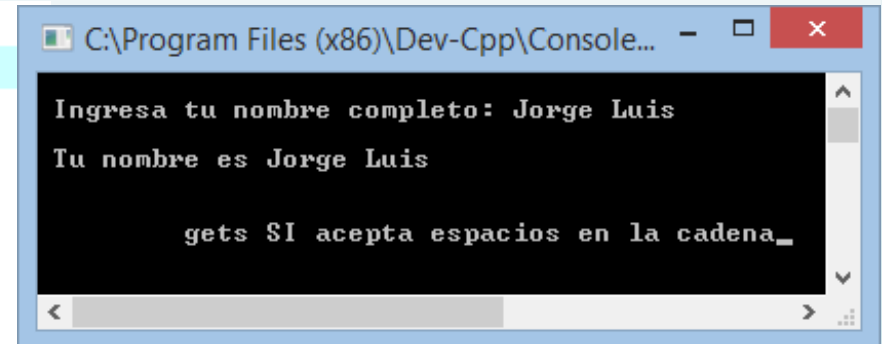
gets (RECIBE cadena)

stdio.h

Lee caracteres del flujo de entrada estándar hasta que encuentra fin de archivo o un salto de línea. Cualquier carácter de línea nueva es descartado, y un carácter nulo es escrito inmediatamente después del último carácter leído.

```
#include <stdio.h> //para gets, printf, getch
int main(int argc, char* argv[]) //función principal
{
    char nombre[30]; //se piden 30 cuadritos para arreglo

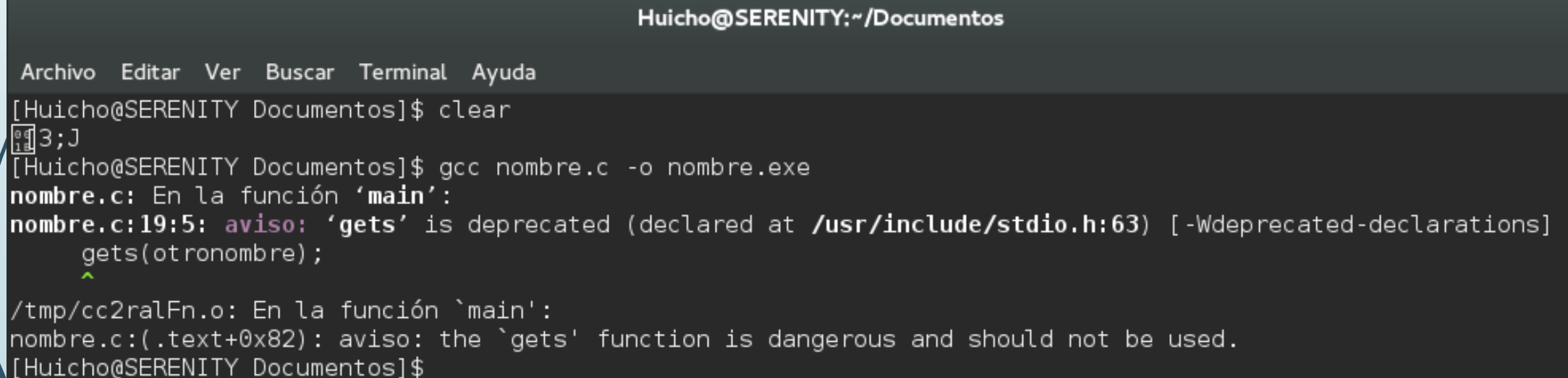
    printf("\n Ingresa tu nombre completo: ");
    //leemos la cadena con la función gets pasando cadena
    gets(nombre);
    printf("\n Tu nombre es %s", nombre);
    printf("\n\n\n\t gets SI acepta espacios en la cadena");
    fflush(stdin); //borra el buffer del teclado
    getch(); //mantiene la pantalla estatica
    return 0; //termina la ejecucion devolviendo cero
}
```



Problema con gets en Linux

Al **compilar** el programa anterior en **Linux** se provocará un **Warning (aviso)** indicando que **gets** es peligroso y no debería usarse debido a que pueden leer una cadena sin límite y desbordar el tamaño del arreglo definido en el código.

Al ser un Warning el programa si compila y se genera el ejecutable.



```
Huicho@SERENITY:~/Documentos

Archivo  Editar  Ver  Buscar  Terminal  Ayuda

[Huicho@SERENITY Documentos]$ clear
[3;J]
[Huicho@SERENITY Documentos]$ gcc nombre.c -o nombre.exe
nombre.c: En la función 'main':
nombre.c:19:5: aviso: 'gets' is deprecated (declared at /usr/include/stdio.h:63) [-Wdeprecated-declarations]
     gets(otronombre);
     ^
/tmp/cc2ralFn.o: En la función `main':
nombre.c:(.text+0x82): aviso: the `gets' function is dangerous and should not be used.
[Huicho@SERENITY Documentos]$
```

Lectura de cadena (fgets)

fgets (RECIBE cadena, tamaño, stdin)

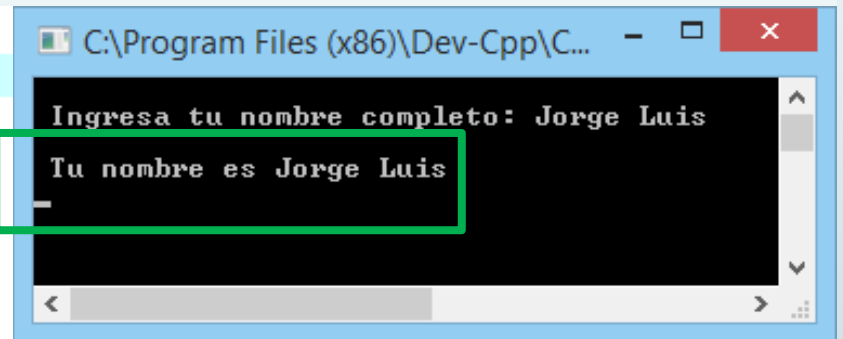
stdio.h

Lee como máximo uno menos que el número de caracteres indicado por **tamaño** del flujo de entrada estándar hasta que encuentra fin de archivo o un **salto de línea** el cual **también es leído** y un carácter nulo es escrito inmediatamente después del último carácter leído.

```
#include <stdio.h> //para gets, printf, getchar

int main(int argc, char* argv[]) //función principal
{
    char nombre[30]; //se piden 30 cuadritos para arreglarlo

    printf("\n Ingresar tu nombre completo: ");
    //pasamos cadena, tamaño del arreglo declarado, stdin
    fgets(nombre, 30, stdin);
    printf("\n Tu nombre es %s", nombre);
    fflush(stdin); //borra el buffer del teclado
    getchar(); //mantiene la pantalla estática
    return 0; //termina la ejecución devolviendo cero
}
```



Lectura de cadena (scanf modificado)

```
scanf(" %[^\\n]", cadena)
```

```
stdio.h
```

El **espacio después de las comillas de apertura** indica que reconocerá todos los caracteres espacio seguidos que haya de donde estemos leyendo o '\\n' que estén en el buffer y **evitar que se salte para leer otra cadena**.

%[^\\n] lee mientras que el carácter no sea un salto de línea y al final asigna el carácter de terminación nulo.

NOTAS:

Los **cuadritos no usados** del arreglo contienen **basura** de la memoria.

Si se van a leer **acentos y letra ñ** en mayúsculas o minúsculas no se almacenarán correctamente por encontrarse en el ASCII extendido que está fuera del rango del tipo de dato char.

Lectura de cadena (scanf modificado)

```
nombre_scanf_modif.c
~/Documentos

#include <stdio.h> //para scanf, printf, getchar

int main(int argc, char* argv[]) //función principal
{
    char nombre[15]; //se piden 15 cuadritos para arreglo
    char ap[15]; //se piden 15 cuadritos para arreglo
    int conde=0;
    printf("\n Ingresa tu nombre: ");
    //leemos con scanf modificado para espacios
    scanf(" %[^\\n]", nombre);
    printf("\n Tu nombre es %s", nombre);

    printf("\n\n Contenido en ascii de cada cuadrito: \\n\\n");
    for(conde=0; conde<=14; conde++)
        printf(" %d", nombre[conde]);

    printf("\n\n\n Ingresa tu apellido paterno: ");
    scanf(" %[^\\n]", ap);
    printf("\n Tu nombre es %s", ap);

    printf("\n\n Contenido en ascii de cada cuadrito: \\n\\n");
    for(conde=0; conde<=14; conde++)
        printf(" %d", ap[conde]);

    getchar(); //atrapa el enter
    getchar(); //mantiene la pantalla estatica
    return 0; //termina la ejecucion devolviendo cero
}
```

C Anchura del tabulador: 8 Ln 7, Col 5 INS

Lectura de cadena (scanf modificado)

```
Huicho@SERENITY:~/Documentos
Archivo Editar Ver Buscar Terminal Ayuda
[Huicho@SERENITY Documentos]$ gcc nombre_scanf_modif.c -o nombre_scanf_modif.exe
[Huicho@SERENITY Documentos]$ ./nombre_scanf_modif.exe

Ingresa tu nombre: Jorge Luis

Tu nombre es Jorge Luis

Contenido en ascii de cada cuadrito:
74 111 114 103 101 32 76 117 105 115 0 -68 60 -21 -65
J o r g e   L u i s \0 b b b b
Ingresa tu apellido paterno: López

Tu nombre es López

Contenido en ascii de cada cuadrito:
76 -61 -77 112 101 122 0 -128 0 0 -48 38 86 -73 75
L e e p e z   b b b b b b b b
```

La letra "b" representa la **basura** que hay en la memoria y no corresponde a ningún carácter de la cadena.

La letra "e" representa una falla en el almacenamiento de un **carácter con acento**.

Lectura de cadena FINAL para Windows y Linux

- Lectura con scanf modificado para aceptar **cadena con espacios** y evitar que se salte al existir espacios o '\n' en el buffer del teclado:

```
scanf(" %[^\\n]", cadena);
```

- Se emplea el modificador "**unsigned**" en la declaración para almacenar correctamente acentos y letra ñ al ampliar el rango del tipo char hasta 255 que comprende el **ascii extendido**.
- Todos los cuadritos de la cadena son **inicializados** con el carácter nulo '\\0' evitando basura de la memoria en los cuadritos no utilizados.

```
nombreCompleto.c
~/Documentos

Abrir  Guardar  [Menu Icon]  [Close Icon]

/*Programa de lectura de nombre con scanf
hecho por Huicho*/

#include <stdio.h> //para gets, scanf y printf

int main(void) //función principal
{
    // se piden 30 cuadritos para arreglo
    // {'\0'} inicializa todos los cuadritos en nulos
    // el tipo es unsigned para almacenar acentos y ñ
    unsigned char nombre[30]={'\0'};

    printf("\n\n Ingresa tu nombre (con scanf modificado): ");
    scanf(" %[^\\n]",nombre);
    printf("\n Tu nombre es %s",nombre);

    //fflush(stdin); //si no opera en linux, usar doble getchar
    getchar(); //atrapa el enter después del scanf
    getchar(); //mantiene la pantalla estatica
    return 0; //termina la ejecucion devolviendo cero
}
```

```
Huicho@SERENITY:~/Documentos
```

Archivo Editar Ver Buscar Terminal Ayuda

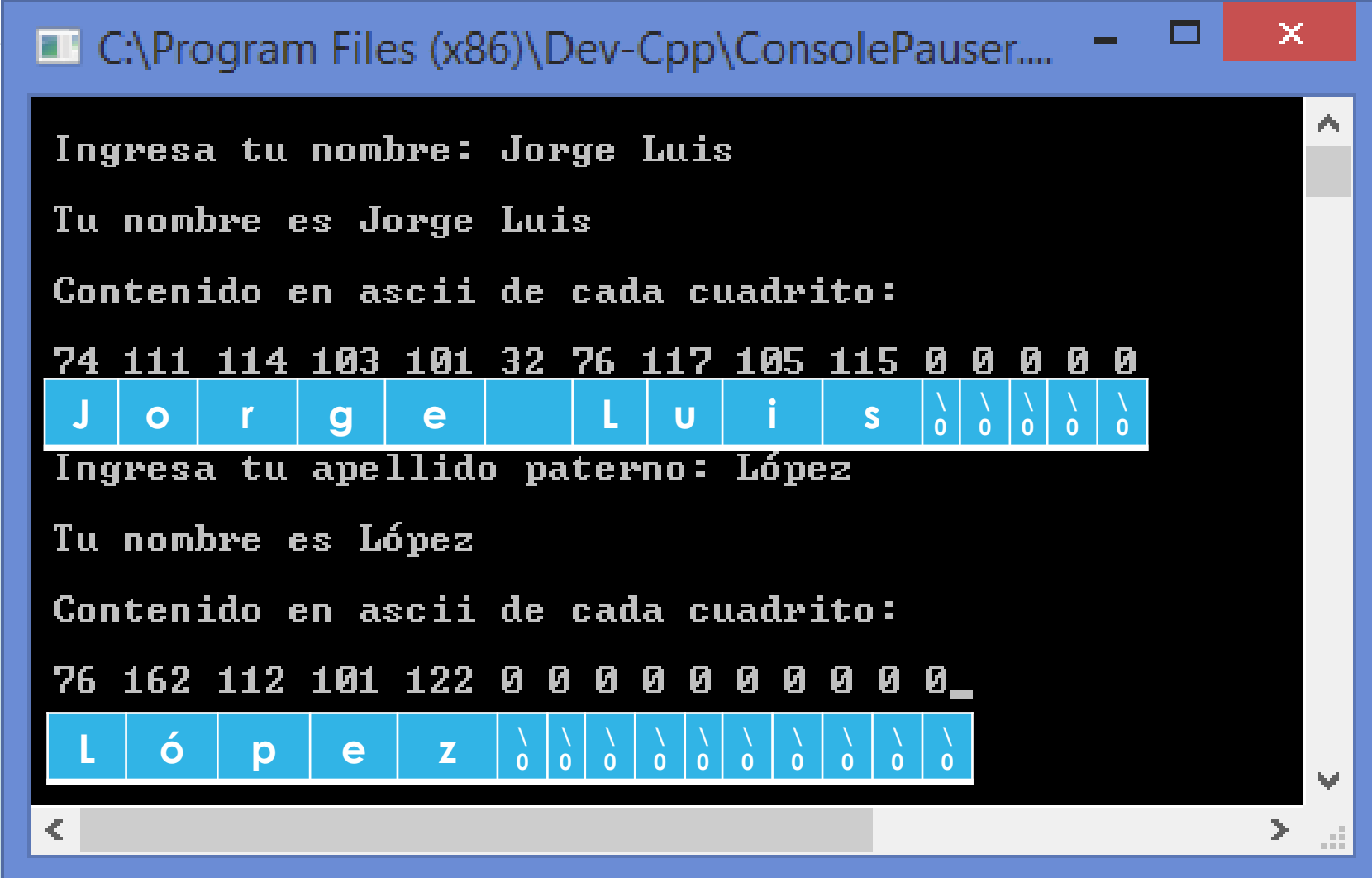
```
[Huicho@SERENITY Documentos]$ gcc nombreCompleto.c -o nombreCompleto.exe
[Huicho@SERENITY Documentos]$ ./nombreCompleto.exe
```

Ingresa tu nombre (con scanf modificado): Jorge Luis

Tu nombre es Jorge Luis



Lectura de cadena FINAL



The screenshot shows a console window titled "C:\Program Files (x86)\Dev-Cpp\ConsolePauser....". The window contains the following text and ASCII representations:

```
Ingresa tu nombre: Jorge Luis
Tu nombre es Jorge Luis
Contenido en ascii de cada cuadrado:
74 111 114 103 101 32 76 117 105 115 0 0 0 0 0
J o r g e   L u i s \0 \0 \0 \0 \0
Ingresa tu apellido paterno: López
Tu nombre es López
Contenido en ascii de cada cuadrado:
76 162 112 101 122 0 0 0 0 0 0 0 0 0 0_
L ó p e z \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
```

The ASCII representations are displayed in a grid format where each character and its corresponding ASCII value are shown in a blue box. The first grid for "Jorge Luis" shows the string followed by five null characters. The second grid for "López" shows the string followed by ten null characters.

Longitud y copia de cadenas

La biblioteca **string.h** contiene funciones de manipulación de cadenas. Algunas son:

Strlen: cadena → int **string.h**

Devuelve la longitud de la cadena dada como argumento.

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

int main(int argc, char* argv[])
{
    char origen[15], destino[15];
    int longitud= 0;

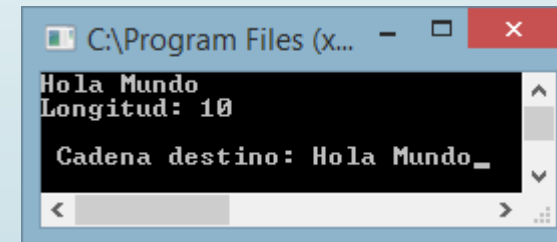
    gets(origen);

    longitud= strlen(origen);
    strcpy(destino, origen);

    printf("Longitud: %d", longitud);
    printf("\n\n Cadena destino: %s", destino);

    getchar();
    return 0;
}
```

Si se introduce el texto "Hola Mundo" la variable tamaño toma el valor 10



Strcpy: cadena_destino x cadena_origen → void **string.h**

Copia la cadena_origen en la cadena_destino

Hecho por Huicho :)

Comparación y concatenación de cadenas

strcmp: cadena1 x cadena2 → int string.h

Esta función compara sus dos argumentos que son de tipo cadena y devuelve un entero.

strcmp(c1,c2)	devuelve 0	si c1 = c2
strcmp(c1,c2)	devuelve un número negativo	si c1 < c2
strcmp(c1,c2)	devuelve un número positivo	si c1 > c2

strcat: cadena_destino x cadena_origen → void string.h

Esta función concatena la cadena fuente a continuación de la cadena destino

```
...
char nombres[15]= "Jorge";
char ap[10]= "Lopez";
char am[10]= "Garcia";
char nombre_completo[50];

...
//strcmp compara si los apellidos son iguales devolviendo 0
if(strcmp(ap, am)==0)
    printf("\n\n Los apellidos son iguales");
else
    printf("\n\n Los apellidos son diferentes");

...
//strcat concatena las cadenas
strcat(nombre_completo, nombres);
...
```

Cadenas de Texto: Arrays de cadenas

Para almacenar varios datos de tipo de cadena en un array, podemos hacerlo de varias formas:

1. Usando la instrucción `typedef`

```
typedef char nombre[30];
...
nombre lista[20];
...
```

Tipos de datos
nombre

Permite al programador definir un nuevo tipo de datos en función de tipos ya conocidos

lista=

"Ana"	"Pepe"		...	"Juan"
0	1			19

2. Usando un array de 2 dimensiones

```
...
char lista[20][30];
...
```

lista=

	0	1	2				29
0	A	n	a	\0			...
1	P	e	p	e	\0		...
...							...
19							