

TEMA 3:

Fundamentos para la construcción de código a partir del algoritmo

TEMA 3:

3.4 Funciones y paso de parámetros

Objetivo

- El alumno construirá programas utilizando el lenguaje de programación C a través de un análisis y modelado algorítmico previo.

Elementos de una función

- **Nombre de la función**
 - Reconoce mayúsculas de minúsculas.
- **Argumentos o parámetros**
 - Los recibe en la llamada a la función en el paréntesis
 - Son datos que necesita para operar que no tiene por sí misma
- **Tipo de retorno**
 - Tipo de dato que devuelve con un valor o resultado aunque sea vacío (void) para ser atrapado en donde fue llamada la función.

Paso por valor

- Criterios para llamar a una función:
 - **IGUAL** nombre de la función.
 - **IGUAL** número de argumentos.
 - **IGUAL** tipo de dato de los argumentos.
 - **IGUAL** o **DIFERENTE** nombre de las variables locales a las que están en la función principal.
- Las variables originales **NO** se modifican.
- Se trabaja sobre una copia de la variable en la función.

Paso por valor

FUNCIÓN	LLAMADA A FUNCIÓN
<pre>void Saludo(void) { printf("\n\t ** Esta funci%cn saluda **\n", 162); }</pre>	<pre>//NO recibe argumentos //NO devuelve Saludo();</pre>
<pre>void Suma(int a, int b) { int c; c= a+b; printf("\n\t La suma de %d + %d = %d", a, b, c); }</pre>	<pre>//RECIBE 2 datos int //NO devuelve int uno, dos; Suma(uno, dos);</pre>
<pre>int Suma2(int a, int b) { int c; c= a+b; return c; }</pre>	<pre>//RECIBE 2 datos int //SI devuelve un int int uno, dos, res; res=Suma2(uno,dos);</pre>

Paso por valor

FUNCIÓN	LLAMADA A FUNCIÓN
<pre>int Resta(int a, int b) { return a-b; }</pre>	<pre>//RECIBE 2 datos int //SI devuelve un int int uno, dos, res; res= Resta(uno, dos);</pre>
<pre>float Divi(int a, int b) { float c; c= (float)a/b; return c; }</pre>	<pre>//RECIBE 2 datos int //SI devuelve un float int uno, dos; float div; div= Divi(uno, dos);</pre>
<pre>int Triple(int a) { return a*3; }</pre>	<pre>//RECIBE 1 dato int //SI devuelve un int int uno, res; res= Triple(uno);</pre>

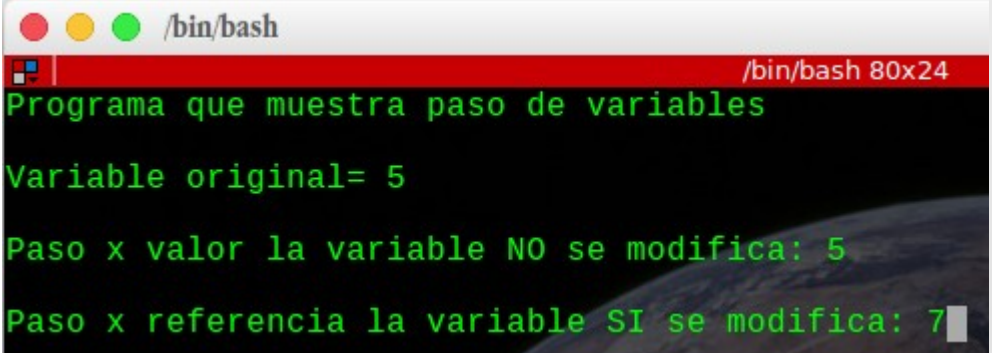
Paso por referencia de una variable

- Referencia es la dirección de memoria de la variable.
- **NO** se crean copias de las variables.
- Se trabaja sobre la variable **ORIGINAL**.
- Las variables originales **SI** se modifican.
- Para pasar una variable por referencia se coloca **&** y el **nombre** de la variable.
- Para recibir la variable por referencia **se requiere un apuntador** al tratarse de una dirección de memoria.

Paso por referencia de una variable

```
1  /*programa con funciones de paso por valor y por referencia
2   * hecho por huicho*/
3
4  #include <stdio.h> //para printf y getchar
5
6  void pasoXValor(int original);
7  void pasoXReferencia(int *original);
8
9  int main(int argc, char* argv[])
10 {
11     int variable=5;
12
13     printf("Programa que muestra paso de variables \n\n");
14     printf("Variable original= %d", variable);
15
16     pasoXValor(variable);
17     printf("\n\nPaso x valor la variable NO se modifica: %d", variable);
18
19     pasoXReferencia(&variable);
20     printf("\n\nPaso x referencia la variable SI se modifica: %d", variable);
21
22     getchar();
23     return 0;
24 }
25
26 void pasoXValor(int original)
27 {
28     original= original+2;
29 }
30
31 void pasoXReferencia(int *original)
32 {
33     *original= *original+2;
34 }
```

- El bloque de código de las funciones puede colocarse entre la biblioteca y la función main o sólo su prototipo ahí y el bloque debajo del cierre de la función principal.
- Llamadas a funciones:
`pasoXValor(variable)`
- `pasoXReferencia(&variable)`
- Por valor se trabaja con una copia y no se altera en la función.
- Por referencia se trabaja con la original y puede ser modificada en la función.



```
/bin/bash
Programa que muestra paso de variables
Variable original= 5
Paso x valor la variable NO se modifica: 5
Paso x referencia la variable SI se modifica: 7
```

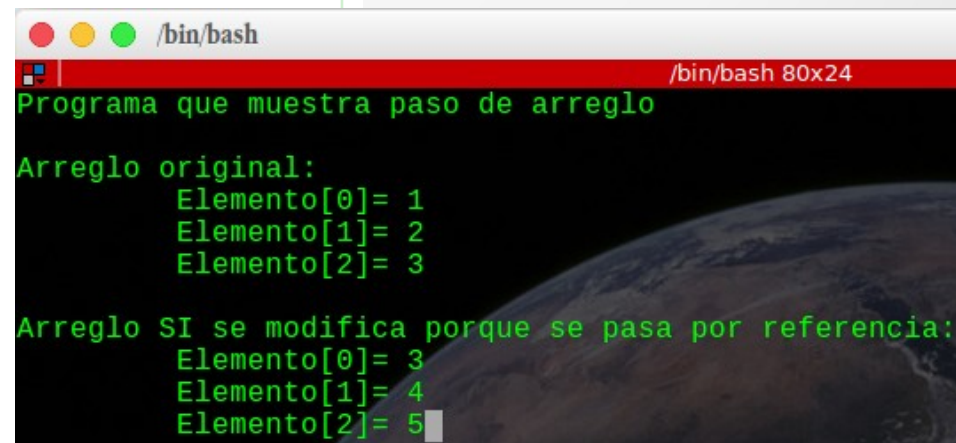
Paso por referencia de un arreglo

- Referencia es la dirección de memoria de la variable.
- **NO** se crean copias de cada cuadrito del arreglo.
- Se trabaja sobre el arreglo **ORIGINAL**.
- El nombre del arreglo es la dirección de memoria del primer cuadrito del arreglo.
- Para pasar un arreglo unidimensional estático en una función se anota su nombre y cuantos elementos tiene.
- Para pasar un arreglo bidimensional estático en una función se anota su nombre, número de renglones y columnas.
- **Un arreglo siempre se pasa por referencia no por valor.**

Paso por referencia de un arreglo

```
1  /*programa con funciones de paso por referencia de arreglo
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf y getchar
5
6  void pasoArreglo(int a[], short cuadritos);
7
8  int main(int argc, char *argv[])
9  {
10     int arreglo[3]={1,2,3};
11     short conde;
12
13     printf("Programa que muestra paso de arreglo \n\n");
14     printf("Arreglo original: ");
15
16     for(conde=0; conde<=2; conde++)
17     {
18         printf("\n\t Elemento[%hd]= %d", conde, arreglo[conde]);
19     }
20
21     pasoArreglo(arreglo, 3);
22
23     printf("\n\nArreglo SI se modifica porque se pasa por referencia: ");
24     for(conde=0; conde<=2; conde++)
25     {
26         printf("\n\t Elemento[%hd]= %d", conde, arreglo[conde]);
27     }
28
29     getchar();
30     return 0;
31 }
32
33 void pasoArreglo(int a[], short cuadritos)
34 {
35     int conde=0;
36
37     for(conde=0; conde<=2; conde++)
38     {
39         a[conde]= a[conde]+2;
40     }
41 }
```

- El bloque de código de las funciones puede colocarse entre la biblioteca y la función main o sólo su prototipo ahí y el bloque debajo del cierre de la función principal.
- Llamada a función:
`pasoArreglo(arreglo, 3)`
- Por referencia se trabaja con el arreglo original y puede ser modificado en la función y mostrado el cambio en la función principal.



```
/bin/bash
Programa que muestra paso de arreglo

Arreglo original:
    Elemento[0]= 1
    Elemento[1]= 2
    Elemento[2]= 3

Arreglo SI se modifica porque se pasa por referencia:
    Elemento[0]= 3
    Elemento[1]= 4
    Elemento[2]= 5
```

Paso de arreglo estático unidimensional

FUNCIÓN	LLAMADA A FUNCIÓN
<pre>//void leerArreglo(int a[MAX], int cuadritos) void leerArreglo(int a[], int cuadritos) { short conde; printf("\n\n Ingresar valores para el arreglo: \n\n"); for(conde=0; conde<=cuadritos-1; conde++) { printf(" Elemento[%hd]= ", conde); scanf("%d", &a[conde]); } }</pre>	<pre>//RECIBE dos argumentos: // -nombre de arreglo // -número de elementos //NO DEVUELVE #define MAX 100 int arreglo[MAX]; short cantidad; imprimirArreglo(arreglo,cantidad);</pre>
<pre>//void imprimirArreglo(int a[MAX], int cuadritos) void imprimirArreglo(int a[], int cuadritos) { short conde; printf("\n\n Arreglo: \n"); for(conde=0; conde<=cuadritos-1; conde++) { printf("\n\t Elemento[%hd]= %d", conde, a[conde]); } }</pre>	<pre>//RECIBE dos argumentos: // -nombre de arreglo // -número de elementos //NO DEVUELVE #define MAX 100 int arreglo[MAX]; short cantidad; leerArreglo(arreglo,cantidad);</pre>

Paso de arreglo estático bidimensional

FUNCIÓN	LLAMADA A FUNCIÓN
<pre>//void leerMatrix(int m[MAX][MAX],short reng,short col) void leerMatrix(int m[][MAX], short reng, short col) { short r,c; printf("\n\n Ingresa valores para arreglo bidi: \n"); for(r=0; r<=reng-1; r++) { for(c=0; c<=col-1; c++) { printf(" Elemento[%hd][%hd]= ", r, c); scanf("%d", &m[r][c]); } } }</pre>	<pre>//RECIBE tres argumentos: // -nombre de arreglo // -número de renglones // -número de columnas //NO DEVUELVE #define MAX 100 int matrix[MAX][MAX]; short reng, col; leerMatrix(matrix, reng, col);</pre>

Paso de arreglo estático bidimensional

FUNCIÓN	LLAMADA A FUNCIÓN
<pre>//void imprimirMatrix(int m[MAX][MAX],short reng,short col) void imprimirMatrix(int m[][MAX], short reng, short col) { short r,c; printf("\n\n Arreglo bidimensional: \n"); for(r=0; r<=reng-1; r++) { for(c=0; c<=col-1; c++) { printf("%5d ", m[r][c]); } printf("\n"); } }</pre>	<pre>//RECIBE tres argumentos: // -nombre de arreglo // -número de renglones // -número de columnas //NO DEVUELVE #define MAX 100 int matrix[MAX][MAX]; short reng, col; imprimirMatrix(matrix,reng,col);</pre>

TEMA 3:

Biblioteca

Biblioteca

- También llamada archivo de encabezado o librería.
- Usar nombre representativo corto y sin espacios como `par.h`
- Incluir bibliotecas al usar funciones externas.
- No es ejecutable por no contar con función `main`.

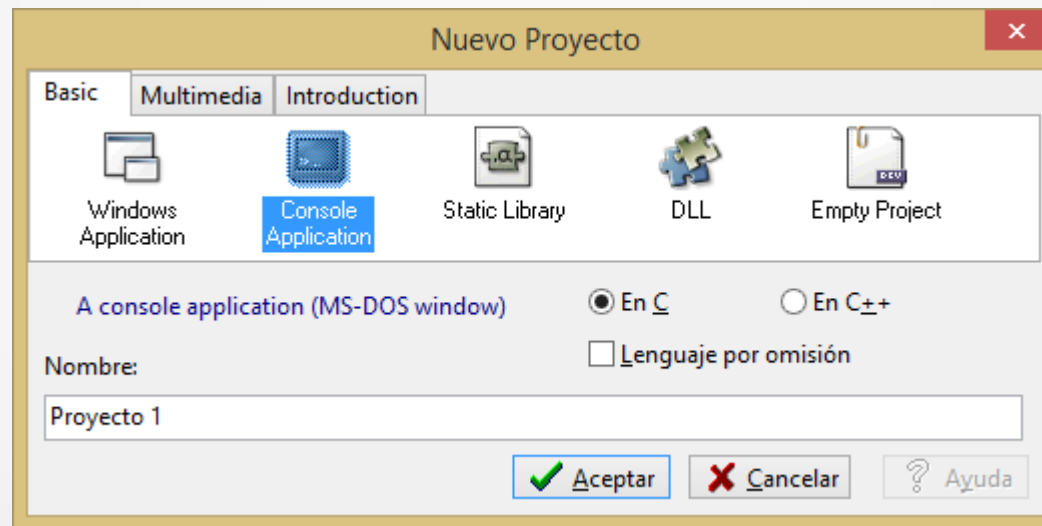
```
#ifndef _PAR_H_
#define _PAR_H_

    //código de las funciones

#endif
```

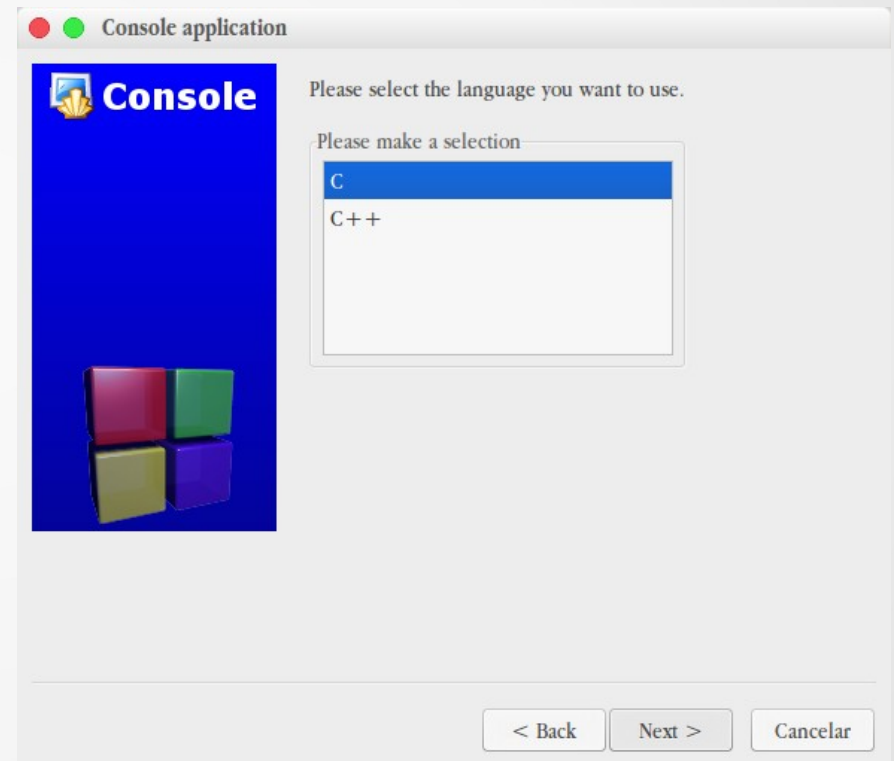
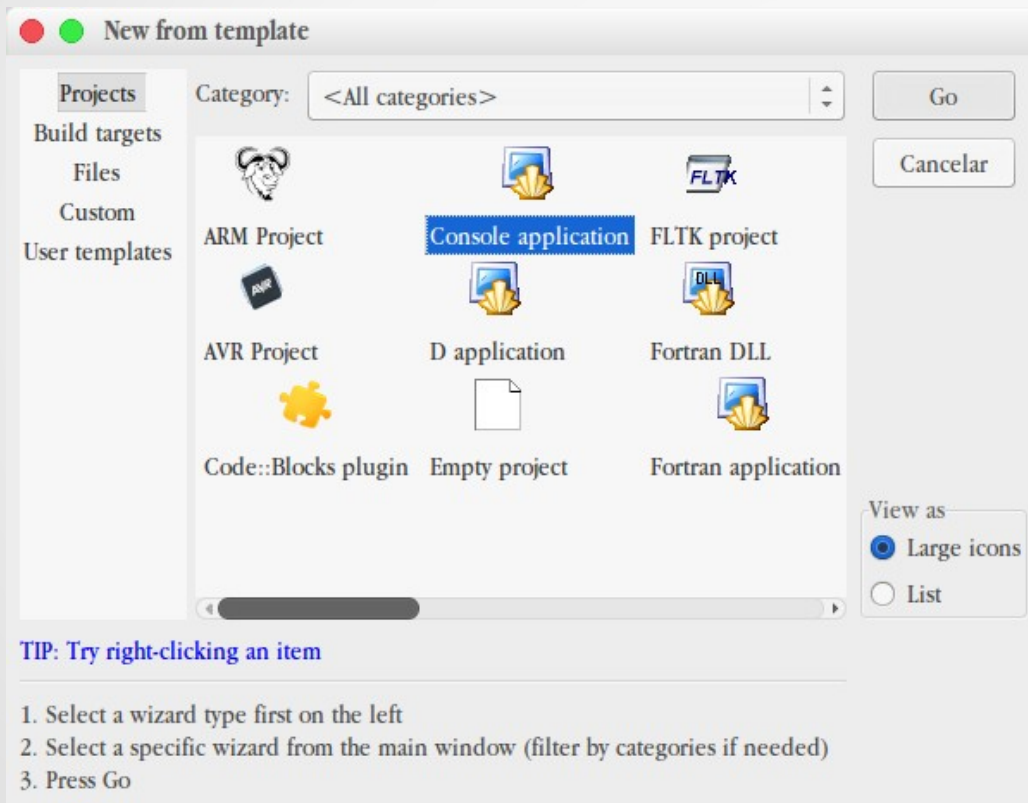

Biblioteca en Dev-C++

- Crear proyecto el cual permite compilar nuestro código fuente en C junto con la biblioteca propia.
- Seleccionar **"Console Application"**, **"En C"** y dar nombre al proyecto.



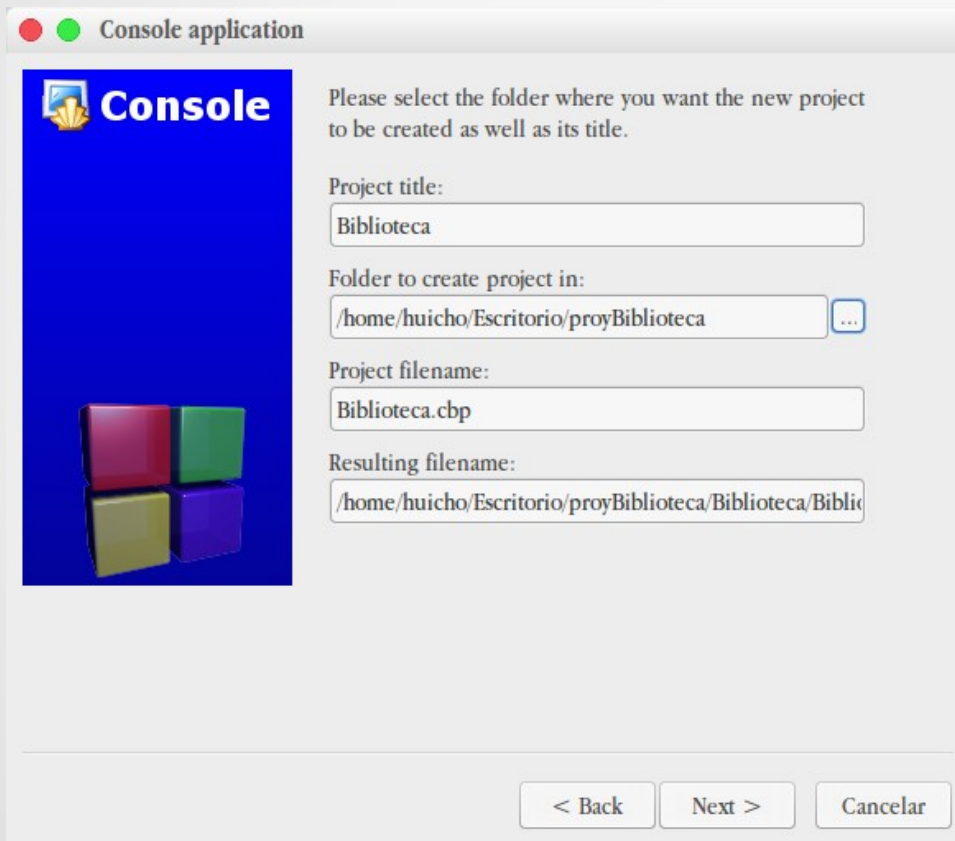
Biblioteca en CodeBlocks

- Crear proyecto, seleccionar "**Console Application**", "**Go**", "**C**" y dar nombre al proyecto.



Biblioteca en CodeBlocks

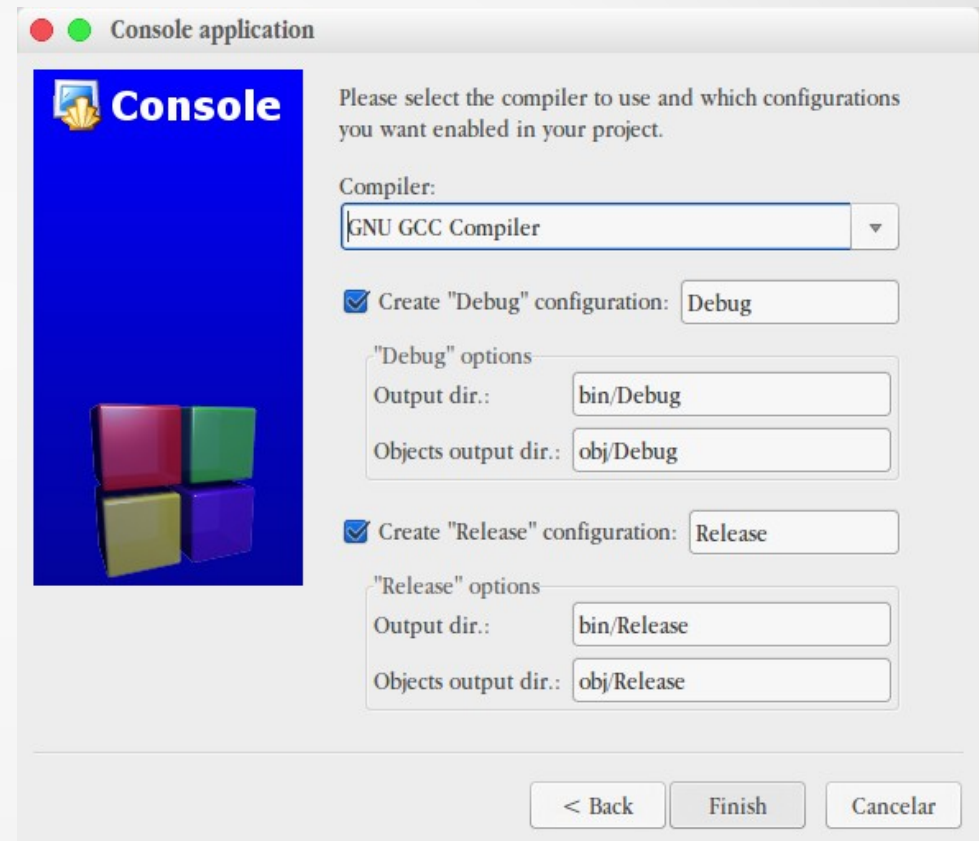
- Dar nombre al proyecto, la ruta y compilador.



This screenshot shows the first step of the CodeBlocks project creation wizard. The window title is 'Console application'. On the left is a blue sidebar with a 'Console' icon and a 3D cube graphic. The main area contains the following fields:

- Please select the folder where you want the new project to be created as well as its title.**
- Project title:** A text box containing 'Biblioteca'.
- Folder to create project in:** A text box containing '/home/huicho/Escritorio/proyBiblioteca' with a browse button (...).
- Project filename:** A text box containing 'Biblioteca.cbp'.
- Resulting filename:** A text box containing '/home/huicho/Escritorio/proyBiblioteca/Biblioteca/Bibli'.

At the bottom are three buttons: '< Back', 'Next >', and 'Cancelar'.



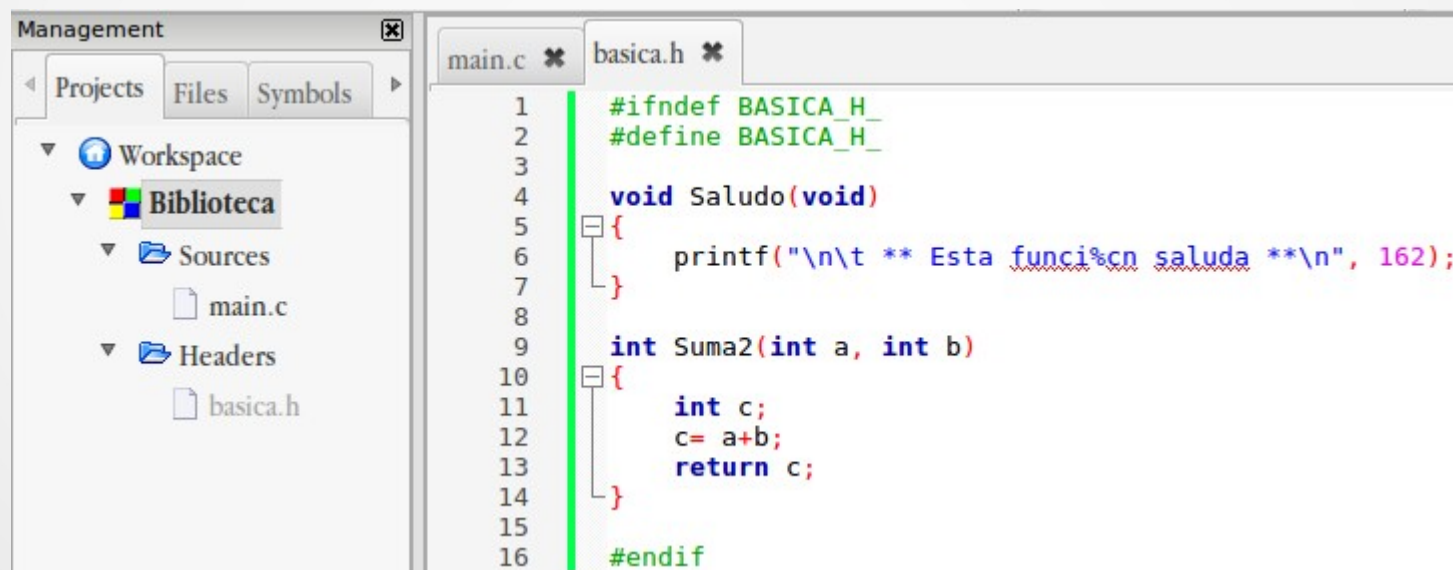
This screenshot shows the second step of the CodeBlocks project creation wizard. The window title is 'Console application'. The sidebar is identical to the first step. The main area contains the following fields:

- Please select the compiler to use and which configurations you want enabled in your project.**
- Compiler:** A dropdown menu showing 'GNU GCC Compiler'.
- Create "Debug" configuration:** A checked checkbox followed by a text box containing 'Debug'.
- "Debug" options:**
 - Output dir.:** A text box containing 'bin/Debug'.
 - Objects output dir.:** A text box containing 'obj/Debug'.
- Create "Release" configuration:** A checked checkbox followed by a text box containing 'Release'.
- "Release" options:**
 - Output dir.:** A text box containing 'bin/Release'.
 - Objects output dir.:** A text box containing 'obj/Release'.

At the bottom are three buttons: '< Back', 'Finish', and 'Cancelar'.

Biblioteca en CodeBlocks

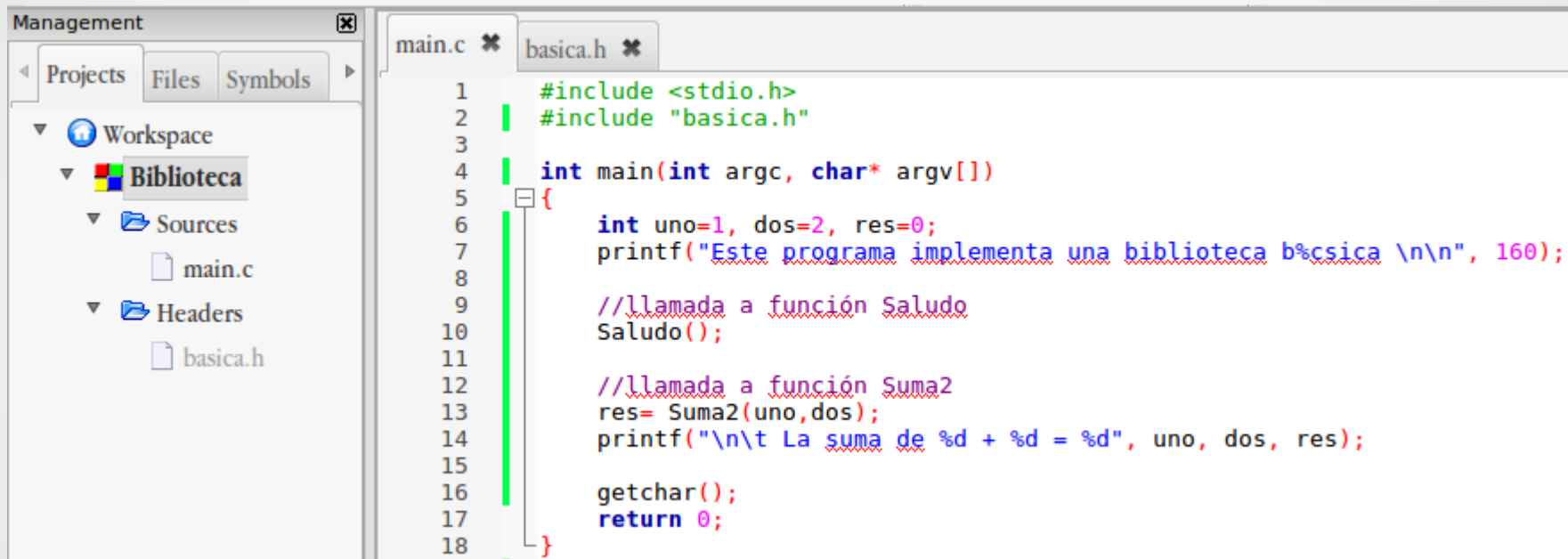
- Podemos incluir en nuestra biblioteca las funciones que hayamos creado para no escribirlas en el programa principal y solo llamarlas desde ahí.



```
1  #ifndef BASICA_H_
2  #define BASICA_H_
3
4  void Saludo(void)
5  {
6      printf("\n\t ** Esta funci%cn saluda **\n", 162);
7  }
8
9  int Suma2(int a, int b)
10 {
11     int c;
12     c= a+b;
13     return c;
14 }
15
16 #endif
```

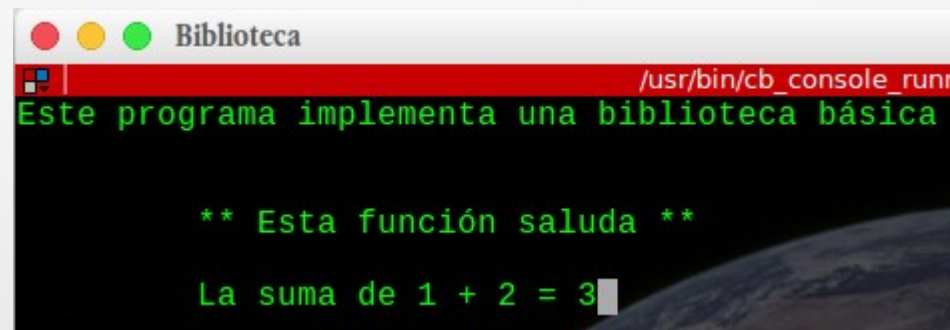
Biblioteca en CodeBlocks

- La biblioteca se incluye en el programa con comillas dobles.



The screenshot shows the CodeBlocks IDE interface. On the left, the 'Management' pane displays the project structure: 'Workspace' contains 'Biblioteca', which has 'Sources' (main.c) and 'Headers' (basica.h). The main editor shows the content of 'main.c' with line numbers 1 to 18. The code includes standard headers and the 'basica.h' library, then calls 'Saludo()' and 'Suma2()' functions, prints the result, and waits for a key press before returning 0.

```
1  #include <stdio.h>
2  #include "basica.h"
3
4  int main(int argc, char* argv[])
5  {
6      int uno=1, dos=2, res=0;
7      printf("Este programa implementa una biblioteca básica \n\n", 160);
8
9      //llamada a función Saludo
10     Saludo();
11
12     //llamada a función Suma2
13     res= Suma2(uno,dos);
14     printf("\n\t La suma de %d + %d = %d", uno, dos, res);
15
16     getchar();
17     return 0;
18 }
```



The screenshot shows the console output of the program. It displays the message 'Este programa implementa una biblioteca básica' followed by a blank line, then '/* Esta función saluda */' followed by another blank line, and finally 'La suma de 1 + 2 = 3'.

```
/usr/bin/cb_console_runn
Este programa implementa una biblioteca básica

/* Esta función saluda */

La suma de 1 + 2 = 3
```

Hecho por Huicho :)