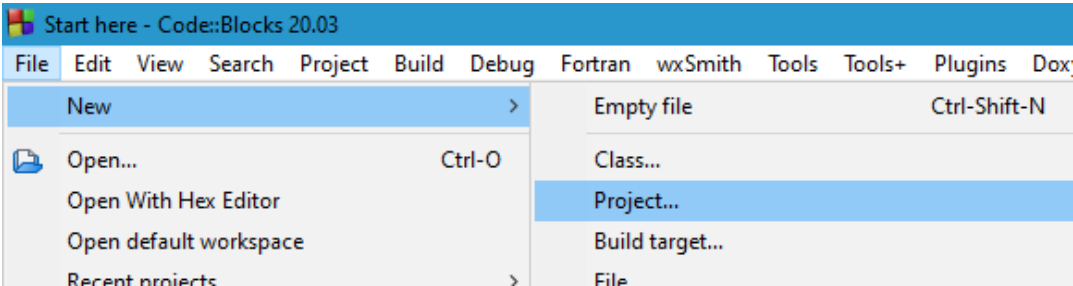


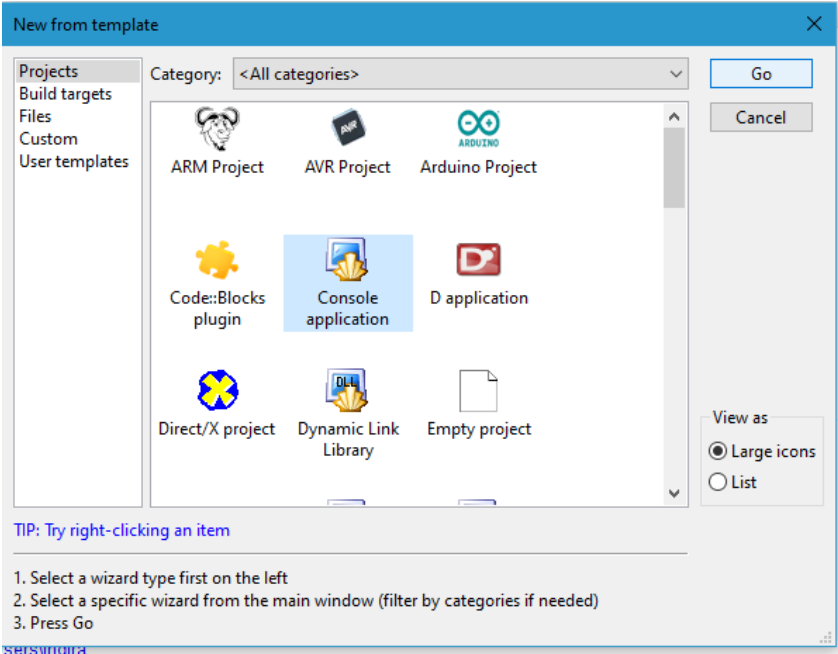
Ejemplo. INSTRUCCIONES: Crear un Gestor de Empleados, para poder realizar la gestión de los datos se deben realizar las operaciones de Alta de registro, Baja de registro, Listar empleados, Consultar datos de empleados, Modificar Salario, Modificar hojas. Debe existir la persistencia de los datos, los empleados deben quedar registrados y cada vez que se ejecute el programa deben poder consultarse los datos almacenados. Los datos que deben ser almacenados son el nombre del empleado, un número de registro, sueldo que gana el empleado y el número de horas que trabaja para la empresa.

Crear la estructura de proyecto

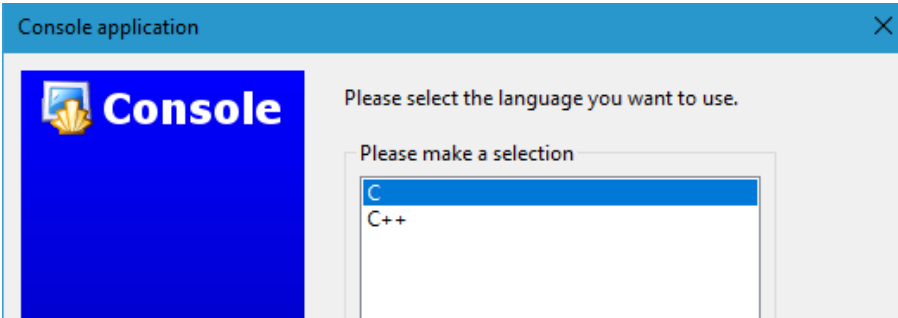
File ->New->Project




Elegir aplicación de Consola




Elegir el lenguaje C



| | | |
|---|--|---------------------------------|
|  | UNIVERSIDAD DE SAN CARLOS DE GUATEMALA | |
| | CENTRO UNIVERSITARIO DE ORIENTE | CARRERAS DE INGENIERÍA |
| | MANEJO E IMPLEMENTACIÓN DE ARCHIVOS | CATEDRÁTICA: ING. INDIRA VALDÉS |
| | PRÁCTICA 3 | |

Asigne un nombre a su proyecto:

**Console**

Please select the folder where you want the new project to be created as well as its title.

Project title:

GestorEmpleados

Folder to create project in:

C:\Users\Indira Valdes\Desktop\Manejo de Archiv...

Project filename:

GestorEmpleados.cbp

Resulting filename:


C:\Users\Indira Valdes\Desktop\Manejo de Archivos\G...

< Back

Next >

Cancel

Seleccionamos el compilador **GNU GCC Compiler** y luego clic en Finalizar

**Console**

Please select the compiler to use and which configurations you want enabled in your project.

Compiler:

GNU GCC Compiler

☒ Create "Debug" configuration:

Debug

"Debug" options

Output dir.:

bin\Debug\

Objects output dir.:

obj\Debug\

☒ Create "Release" configuration:

Release

"Release" options

Output dir.:

bin\Release\

Objects output dir.:

obj\Release\

< Back

Finish

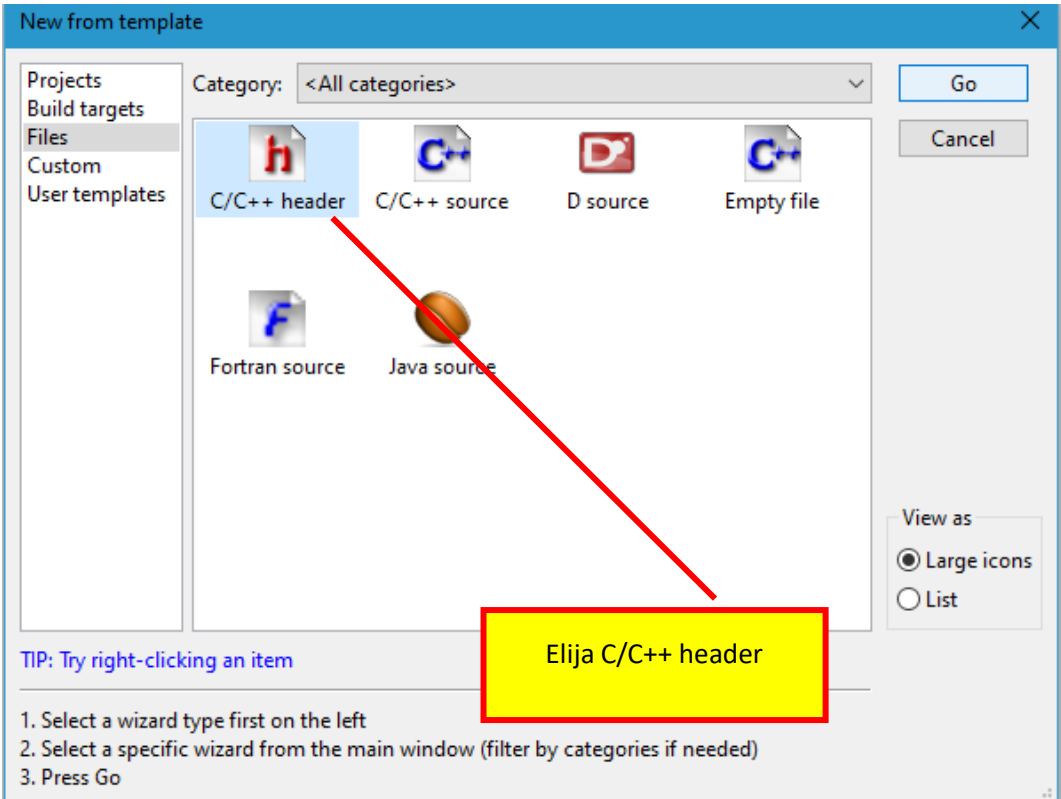
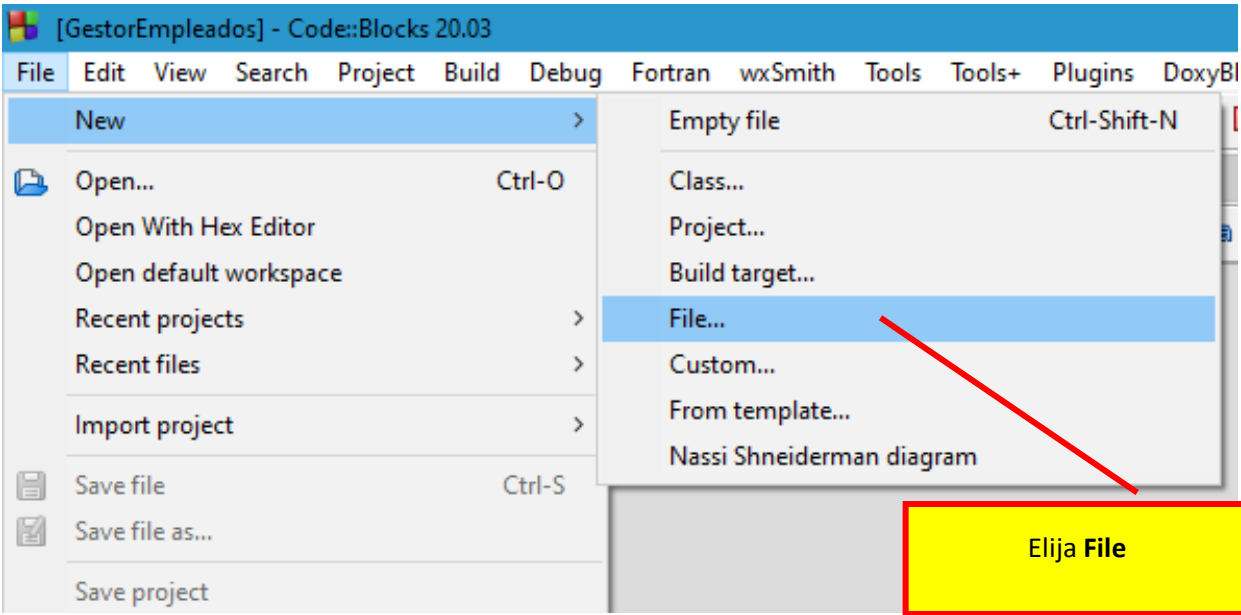
Cancel

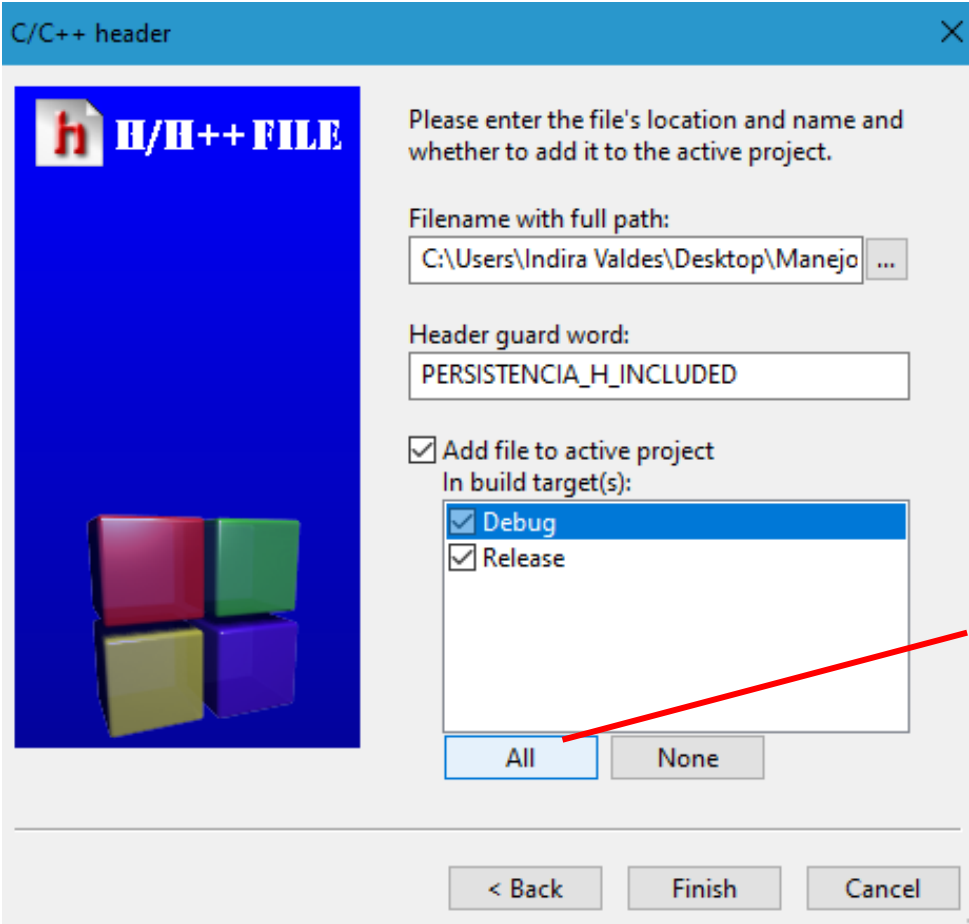
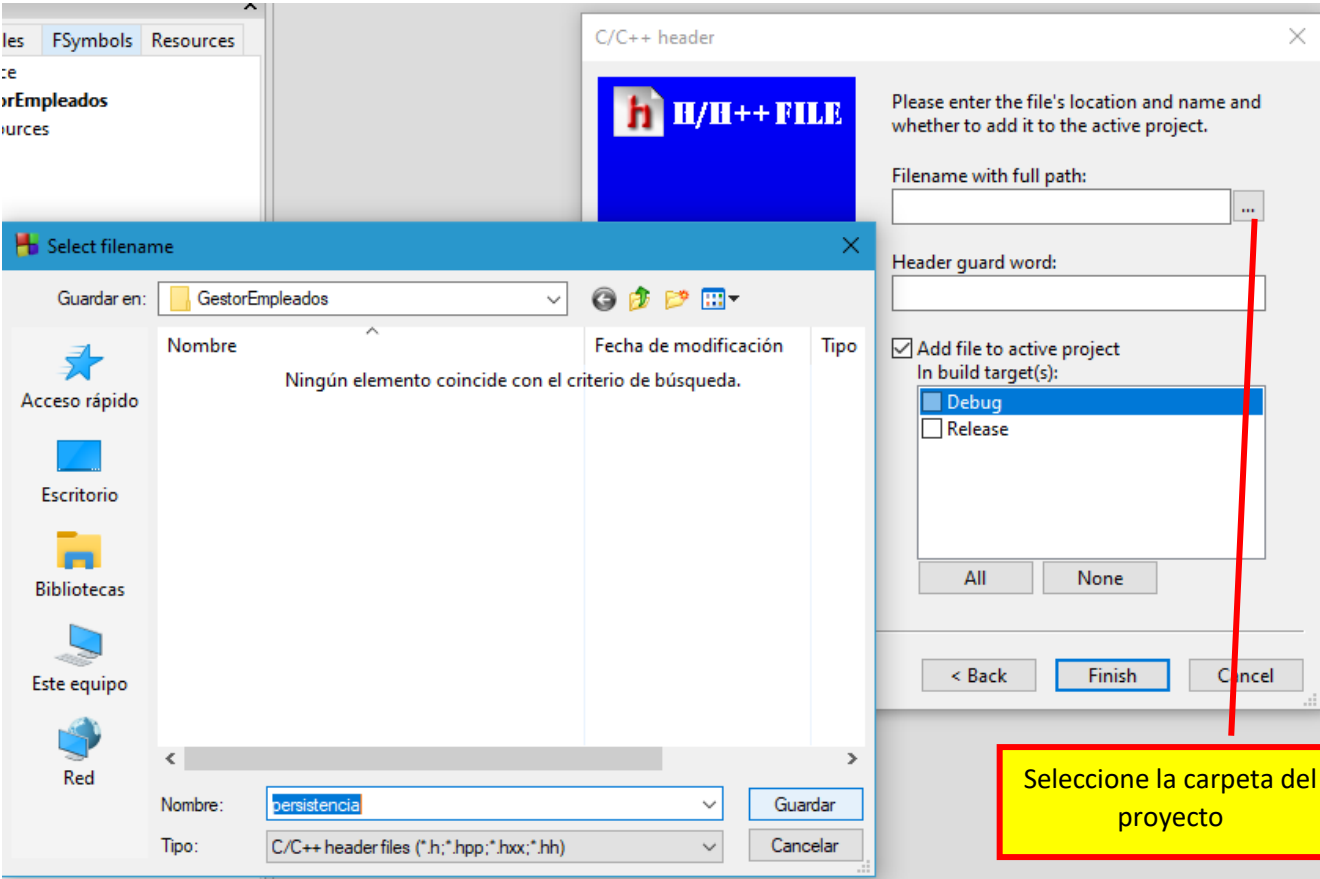
Tendremos un fichero de cabecera con todos los encabezados

Fichero de código fuente con las implementaciones de las funciones

Crearemos un bloque para las operaciones de los datos y manipular archivos.

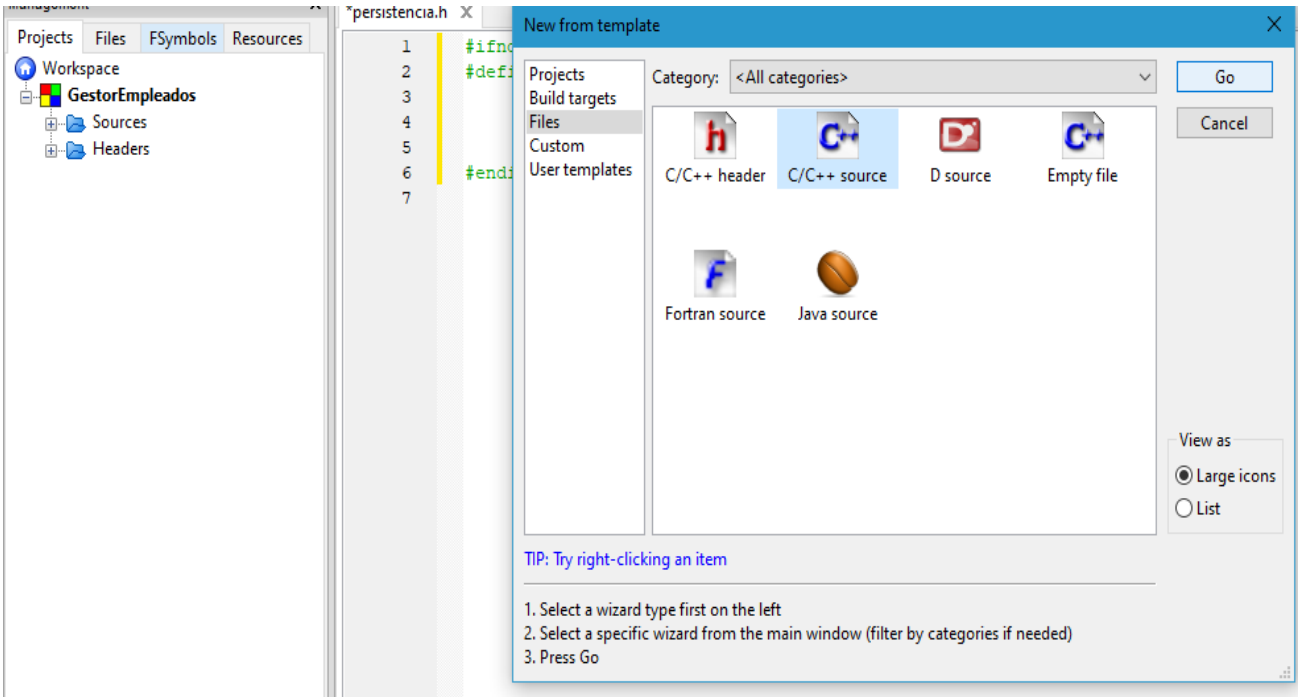
CREACIÓN DE FICHERO DE ENCABEZADO **persistencia.h**



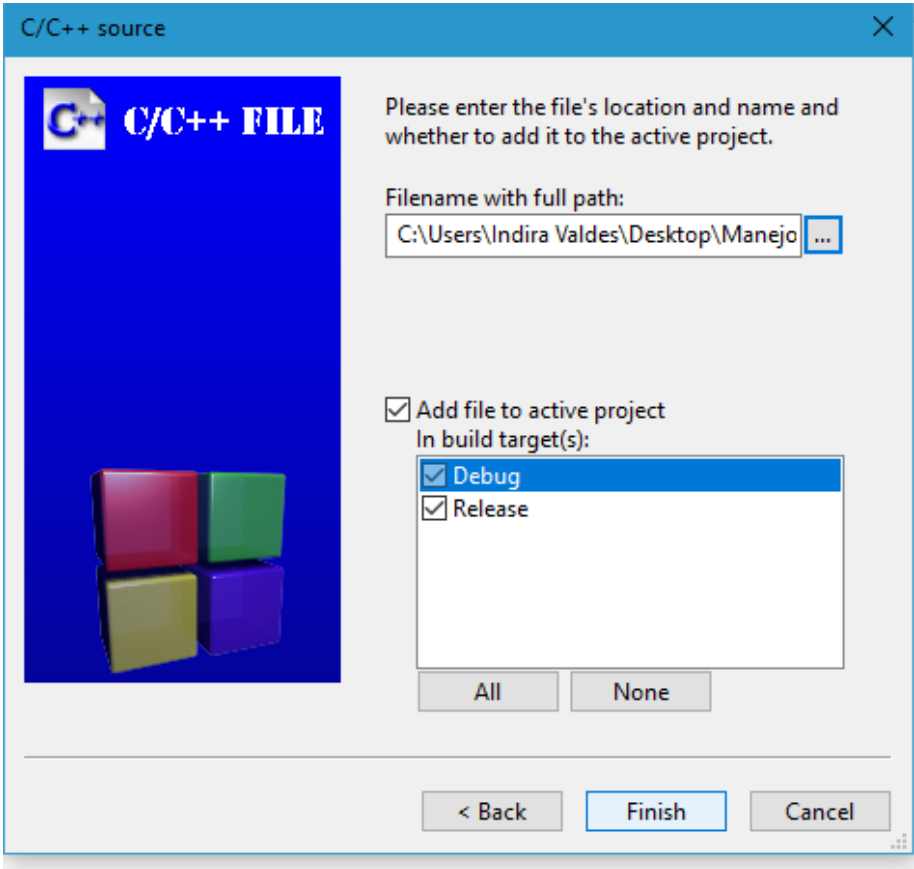


CREACIÓN DE FICHERO DE FUENTE **persistencia.c**


File ->New->File




Recuerde elegir el lenguaje en código C y en activar todas las opciones para manejo del archivo



CREACIÓN DE FICHERO DE ENCABEZADO LLAMADO **operaciones.h**

**H/H++ FILE**



Please enter the file's location and name and whether to add it to the active project.


Filename with full path:
...


Header guard word:

☒ Add file to active project
In build target(s):

☒ Debug
☒ Release

CREACIÓN DE FICHERO DE FUENTE LLAMADO **operaciones.c**

**C/C++ FILE**



Please enter the file's location and name and whether to add it to the active project.

Filename with full path:
...

☒ Add file to active project
In build target(s):

☒ Debug
☒ Release

CREACIÓN DE ARCHIVO DE TEXTO LLAMADO **empleados.txt**

empleados: Bloc de notas

Archivo

Edición

Formato

Ver

Ayuda

pepe;1;4000.00;35

juan;3;2300.00;30

lucia;4;1500.00;20

CREACIÓN DEL MENÚ PRINCIPAL ABRIR **main.c**

El menú estará implementado en la librería **operaciones.h**

Abrimos **operaciones.h** y crearemos una función que desplegará el menú

tencia.h X *operaciones.h X *operaciones.c X main.c X

1

#ifndef OPERACIONES_H_INCLUDED

2

#define OPERACIONES_H_INCLUDED

3

4

void mostrarMenu();//será un procedimiento porque no va devolver nada solo va imprimir

5

6

#endif // OPERACIONES_H_INCLUDED

Luego implementamos el procedimiento en el archivo **operaciones.c**

tencia.h X operaciones.h X operaciones.c X *main.c X

1

#include <stdio.h>

2

#include <stdlib.h>

3

#include <string.h>

4

#include "operaciones.h"

5

6

void mostrarMenu(){//implementamos el procedimiento que mostrará el menú

7

printf("\nIntroduce la opcion que desee realizar:\n");

8

printf("1- Alta empleado\n");

9

printf("2- Baja empleado\n");

10

printf("3- Listar empleados\n");

11

printf("4- Consultar datos empleado\n");

12

printf("5- Modificar sueldo\n");

13

printf("6- Modificar horas\n");

14

printf("7- Salir\n");

15

}

La función `mostrarMenu()` la vamos a utilizar en el `main.c`

tencia.h X operaciones.h X operaciones.c X *main.c X

1

#include <stdio.h>

2

#include <stdlib.h>

3

#include "operaciones.h"

4

#include "persistencia.h"

5

6

int main()

7

{

8

int opcion = 0; //en esta variable vamos a guardar la eleccion del usuario

9

printf("BIENVENIDOS AL GESTOR DE EMPLEADOS\n");

10

mostrarMenu();

11

scanf("%d",&opcion); //pedimos | que elija una opción

12

13

while(opcion != 7){ //mientras la opción del usuario sea distinta de 7

14

if(opcion == 1){ //que se ejecute al ingresar 1 la alta de empleados

15

printf("Alta\n");

16

}

17

else if(opcion == 2){

18

printf("Baja\n");

19

}

20

else if(opcion == 3){

21

printf("Listar\n");

22

}

23

else if(opcion == 4){

24

printf("Consultar\n");

25

}

26

else if(opcion == 5){

27

printf("Sueldo\n");

28

}

29

else if(opcion == 6){

30

printf("Horas\n");

31

}

32

mostrarMenu(); //mostramos el menú nuevamente

33

scanf("%d",&opcion); //pedimos nuevamente que elija una opción

34

}

35

36

printf("GRACIAS POR USAR EL GESTOR DE EMPLEADOS. ¡HASTA PRONTO!\n\n");

37

}

38

Prueba el menú con cada una de las opciones

CREACIÓN DE ESTRUCTURA EMPLEADO ABRIR **operaciones.h**

```

1  #ifndef OPERACIONES_H_INCLUDED
2  #define OPERACIONES_H_INCLUDED
3  //vamos a declarar la estructura empleado
4  struct empleado{
5      char nombre[255];
6      int id;
7      float sueldo;
8      int horas;
9  };
10
11 void mostrarMenu(); //será un procedimiento porq
12
13 #endif // OPERACIONES_H_INCLUDED

```

Ahora necesitamos un ARRAY para almacenar muchos empleados, vamos a crear un arreglo del tipo struct en **main.c**

```

4  #include "persistencia.h"
5
6  int main()
7  {
8      int opcion = 0; //en esta variable vamos a guardar la eleccion del usuario
9      struct empleado empleados[100]; //creamos el arreglo del tipo estructura empleado
10     printf("BIENVENIDOS AL GESTOR DE EMPLEADOS\n");
11     mostrarMenu();
12     scanf("%d",&opcion); //pedimos que elija una opción

```

En la librería **persistencia.h** vamos a crear la función que va permitir cargar en el arreglo todos los empleados que se tengan en el archivo empleados.txt

```

1  #ifndef PERSISTENCIA_H_INCLUDED
2  #define PERSISTENCIA_H_INCLUDED
3  //creamos el encabezado que recibe el arreglo de empleados, y es entero
4  //porque devolverá el número que haya cargado del archivo
5  int cargarEmpleados(struct empleado empleados[100]);
6
7  #endif // PERSISTENCIA_H_INCLUDED
8

```

Ahora vamos a implementar la función `cargarEmpleados` en el archivo `persistencia.c`

```

X persistencia.h X persistencia.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "operaciones.h"//incluimos esta libreria para que reconozca la estructura empleado
5
6  //implementar el método cargarEmpleados
7  int cargarEmpleados(struct empleado empleados[100]){//recibe un array del tipo struct
8      int n_emp = 0;//guarda el numero de empleados que existen en el fichero
9      FILE *f;//creamos una variable del tipo fichero
10     f = fopen("empleados.txt","r");//abrimos el archivo en modo lectura
11     char cadena[255];//almacena cada linea que se lea del fichero
12     char delimitador[] = ";";//esta variable servira para separar las lineas del fichero en campos
13
14     while (feof(f) == 0){//recorremos todas las lineasmientras no hayamos llegado al final del fichero
15
16         struct empleado e;//creamos una estructura para cargar los datos del empleado que este trabajando
17         fgets(cadena,255,f);//obtenemos la línea y guardamos en la variable cadera, el tamaño maximo de la linea
18             //f es el fichero donde se almacena
19         char *token = strtok(cadena,delimitador);//strtok función que se encuentra en la libreria string.h
20             //permite separar una cadena por delimitador devolviendo cada campo y lo almacenamos en token
21         if(token != NULL){//si el token es diferente a vacío
22             int campo = 1;//variable para identificar en que campo me encuentro
23             while (token != NULL){//mientras token sea diferente a nulo
24                 if (campo == 1){//si el campo es el primero es porque leo el campo nombre
25                     strcpy(e.nombre,token);//copio en el campo nombre la cadena recibida
26                 }
27                 else if(campo == 2){//si el campo es igual a 2, estoy en el campo id
28                     e.id = atoi(token);//ahora id es numero entero entonces uso atoi para
29                         // convertir la cadena en entero
30                 }
31                 else if(campo == 3){//si el campo es igual a 3, estoy en el campo sueldo
32                     e.sueldo = atof(token);//uso atof para convertir la cadena en float
33
34                 }
35                 else if(campo == 4){//si el campo es igual a 4, estoy en el campo horas
36                     e.horas = atoi(token);//uso atoi para convertir la cadena en entero
37                 }
38                 campo++;//incremento el campo en 1 ahora será 2
39                 token = strtok(NULL, delimitador);//obtengo el campo siguiente, pasamos un null
40                     //y el delimitador, entonces al ser token =Null saldrá del While
41             }
42             empleados[n_emp] = e;//almacenamos en el arreglo de empleados en la posición que marque n_emp
43             //en e que es el empleado que acabo de cargar (e es la estructura de tipo empleado)
44             n_emp++;//incremento el contado n_emp
45         }
46         fclose(f);//cierro el fichero
47         return n_emp;//retorno la cantidad de empleados que fueron cargados
48     }
49

```

Corremos el código y no debe dar ningún error

Ahora abrimos el **main.c** y creamos la variable `num_emp = 0` y enviamos el arreglo a la función `cargarEmpleados`

*main.c X
persistencia.h X
persistencia.c X

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "operaciones.h"
4  #include "persistencia.h"
5
6  int main()
7  {
8      int opcion = 0; // en esta variable vamos a guardar la eleccion del usuario
9
10     int num_emp = 0; // guarda el número de empleados que hay
11
12     struct empleado empleados[100]; // creamos el arreglo del tipo estructura empleado
13
14     num_emp = cargarEmpleados(empleados); // indicamos que esta variable es igual a
15                                     // la función cargarEmpleados y le paso el ARRAY empleados
16     printf("BIENVENIDOS AL GESTOR DE EMPLEADOS\n");

```

En este punto tenemos cargado todos los registros existentes del archivo en el arreglo del tipo struct empleado

CREACIÓN DE FUNCION LISTAR EMPLEADOS
operaciones.h

Lo vamos a declarar debajo de `mostrarMenu()`

.c X
persistencia.h X
persistencia.c X
*operaciones.h X

```

1  #ifndef OPERACIONES_H_INCLUDED
2  #define OPERACIONES_H_INCLUDED
3  // vamos a declarar la estructura empleado
4  struct empleado{
5      char nombre[255];
6      int id;
7      float sueldo;
8      int horas;
9  };
10
11  void mostrarMenu(); // será un procedimiento porque no va devolver nada solo va imprimir
12  void listarEmpleados(struct empleado empleados[100], int num_emp); // definimos la función
13                                     // listar empleados y recibe el struct y el numero de empleados existentes
14
15  #endif // OPERACIONES_H_INCLUDED
16

```

Abirmos el archivo **operaciones.c** y nos colocamos después de la función **mostrarMenu()** y creamos la función **listarEmpleados** que recibe el arreglo empleados del tipo estructura empleado y recibe el número de empleados que contiene el archivo

```

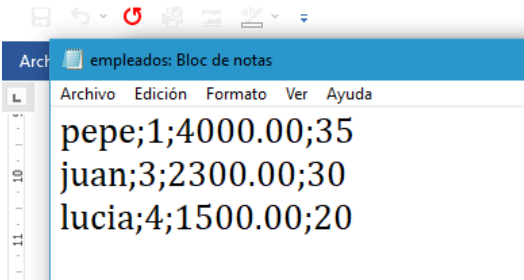
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "operaciones.h"//agregamos la libreria para que sea visible el struct empleados
5
6  void mostrarMenu(){//implementamos el procedimiento que mostrará el menú
7      printf("\nIntroduce la opcion que desee realizar:\n");
8      printf("1- Alta empleado\n");
9      printf("2- Baja empleado\n");
10     printf("3- Listar empleados\n");
11     printf("4- Consultar datos empleado\n");
12     printf("5- Modificar sueldo\n");
13     printf("6- Modificar horas\n");
14     printf("7- Salir\n");
15 }
16 void listarEmpleados(struct empleado empleados[100], int num_emp){
17     printf("Listando los datos del empleado: \n");
18     for (int i=0; i<num_emp; i++){//recorremos con un for desde 0 hasta la posición num_emp
19         //del arreglo empleados del tipo struct empleado
20         printf("Nombre: %s\n", empleados[i].nombre);//imprime cada uno de los campos
21         printf("Id del empleado: %d\n", empleados[i].id);
22         printf("Sueldo: %.2f\n", empleados[i].sueldo);
23         printf("Horas semanales: %d\n",empleados[i].horas);
24         printf("\n");
25     }
26 }
  
```

Ahora actualizaremos el **main.c** para llamar a la función **listarEmpleados()**

```

24     else if(opcion == 2){
25         printf("Baja\n");
26     }
27     else if(opcion == 3){
28         listarEmpleados(empleados,num_emp);
29     }
30     else if(opcion == 4){
31         printf("Consultar\n");
  
```

Recuerde crear el archivo de texto **empleados.txt** que contiene los registros, ingrese previamente algunos



Ahora verifica ejecutando el programa y elegir la opción listar empleados, deben aparecer los registros que tiene almacenados

CREACIÓN DE FUNCION **comprobarId()** AUXILIAR PARA COMPROBAR SI EXISTE UN EMPLEADO POR MEDIO DEL ID **operaciones.c**

Pedimos un ID, y si al momento de crear un empleado existe un empleado con ese ID entonces no permite crearlo porque ya existe un ID con ese empleado, igual servirá para eliminar y modificar. Esta función la crearemos de forma interna en **operaciones**.

```

28 int comprobarId(struct empleado empleados[100], int num_emp, int id){//devolverá un entero
29     //recibe el ARRAY empleados, el número de empleados y recibirá un id
30     int resultado = 0;//numero entero que indica si existe o no existe el id
31
32     for (int i=0; i<num_emp; i++){//recorremos cada registro de empleados
33         if (empleados[i].id == id){//comparamos el id del empleado en la posicion actual
34             resultado = 1;
35         }
36     }
37     //si no encuentra el id resultado seguirá siendo 0
38     return resultado;//devolvemos el resultado que usaran el resto de operaciones del sistema
39 }
40
41

```

CREACIÓN DE LA FUNCIÓN ALTA DE EMPLEADOS **operaciones.h**

Ahora crearemos el resto de funciones que usaran la función auxiliar, estas las declaremos en el archivo de tipo encabezado **operaciones.h**

.c X
persistencia.h X
persistencia.c X
*operaciones.h X
*operaciones.c X

```

1  #ifndef OPERACIONES_H_INCLUDED
2  #define OPERACIONES_H_INCLUDED
3  //vamos a declarar la estructura empleado
4  struct empleado{
5      char nombre[255];
6      int id;
7      float sueldo;
8      int horas;
9  };
10
11 void mostrarMenu();//será un procedimiento porque no va devolver nada solo va imprimir
12 void listarEmpleados(struct empleado empleados[100], int num_emp);//definimos la función
13     //listar empleados y recibe el estruct y el numero de empleados existentes
14 int alta(struct empleado empleados[100], int num_emp);//función que devuelve un numero entero 1 o 0
15     //si se ha dado de alta
16
17 #endif // OPERACIONES_H_INCLUDED
18

```


Nos posicionamos después de la función `comprobarId()`

X
persistencia.h
X
persistencia.c
X
*operaciones.h
X
*operaciones.c
X

```

40 }
41
42 int alta(struct empleado empleados[100], int num_emp){//implementamos la función
43     int creado = 0;//variable que se devuelve si se pudo crear o no el empleado
44     struct empleado emp;//creamos una estructura emp del tipo empleado donde cargare
45         //los datos del nuevo empleado que vamos a crear
46     fflush(stdin);//limpiamos el buffer
47     printf("Introduce nombre y apellidos\n");//pedimos los datos al usuario
48     gets(emp.nombre);//guardamos los datos con la función gets en la estructura emp
49
50     printf("Introduce Id del empleado\n");
51     scanf("%d",&emp.id);
52
53     printf("Introduce sueldo\n");
54     scanf("%f",&emp.sueldo);
55
56     printf("Introduce horas semanales trabajadas\n");
57     scanf("%d",&emp.horas);
58
59     int existe = comprobarId(empleados,num_emp,emp.id);//en esta variable guardamos lo que
60         //devuelve la funcion comprobarId
61
62     if (num_emp < 100){//ahora evaluamos si existe el empleado según el tamaño del ARREGLO 100
63         if(existe == 0){//si no existe el empleado
64             empleados[num_emp] = emp;//guardo la información de la posición actual del arreglo
65                 //en la estructura emp
66             creado = 1;//devuelvo 1 para indicar que el empleado fue creado
67         }
68         else{//si existe es diferente de 0 entonces el empleado ya existe y no se puede crear
69             printf("No se puede crear empleado. ID duplicado\n");
70         }
71     }
72     else{//Si el numero de empleado es igual o mayor de 100 no permitira crear más empleado:
73         printf("No se puede dar de alta el empleado. Cupo alcanzado");
74     }
75
76     return creado;//devuelve el valor de la variable creado
77
78 }

```

Ahora vamos a actualizar el `main()` del archivo `main.c` para llamar al método `alta`, evaluando el valor

*main.c
X
persistencia.h
X
persistencia.c
X
*operaciones.h
X
*operaciones.c
X

```

18     scanf("%d",&opcion);//pedimos que elija una opción
19
20     while(opcion != 7){//mientras la opción del usuario sea distinta de 7
21         if(opcion == 1){//que se ejecute al ingresar 1 la alta de empleados
22             int creado = alta(empleados,num_emp);
23             if (creado == 1){
24                 num_emp++;
25             }
26         }
27         else if(opcion == 2){
28             printf("Baja\n");
29         }

```

Ahora verifica ejecutando el programa e ingresa un registro

CREACIÓN DE OPERACIÓN BAJA DE EMPLEADOS **operaciones.h**

Declaremos la función baja en el archivo de tipo encabezado **operaciones.h**

```

13 //listar empleados y recibe el struct y el numero d
14 int alta(struct empleado empleados[100], int num_emp); //función que devu
15 //si se ha dado de a
16 int baja(struct empleado empleados[100], int num_emp);
17
18 #endif // OPERACIONES_H_INCLUDED

```

Nos posicionamos después de la función **alta()** en el archivo **operaciones.c**

en

```

80 int baja(struct empleado empleados[100], int num_emp){
81     int eliminado = 0;
82     int id, indice;
83
84     printf("Introduce el Id del empleado a dar de baja\n");
85     scanf("%d",&id);
86
87     int existe = comprobarId(empleados,num_emp,id);
88
89     if (existe == 1){
90         for (int i=0; i<num_emp; i++){
91             if (empleados[i].id == id){
92                 indice = i;
93             }
94         }
95
96         for (int j=indice; j<num_emp-1; j++){
97             empleados[j] = empleados[j+1];
98             struct empleado aux;
99             empleados[j+1] = aux;
100         }
101         eliminado = 1;
102     }
103     else{
104         printf("No se puede dar de baja el empleado. ID no existe\n");
105     }
106     return eliminado;
107 }

```

Ahora vamos a actualizar el **main()** del archivo **main.c** para llamar al método baja, evaluando el valor

```

27 else if(opcion == 2){
28     int eliminado = baja(empleados,num_emp);
29     if (eliminado == 1){
30         num_emp--;
31     }
32 }

```

CONSULTAR LOS DATOS UN SOLO EMPLEADO operaciones.h

Declaremos la función `consultarDatosEmpleado()` en el archivo de tipo encabezado `operaciones.h`

```

3 //vamos a declarar la estructura empleado
4 struct empleado{
5     char nombre[255];
6     int id;
7     float sueldo;
8     int horas;
9 };
10
11 void mostrarMenu();//será un procedimiento porque no va devolver nada solo va imprimir
12 void listarEmpleados(struct empleado empleados[100], int num_emp);//definimos la función
13 //listar empleados y recibe el struct y el numero de empleados existe
14 int alta(struct empleado empleados[100], int num_emp);//función que devuelve un numero ent
15 //si se ha dado de alta
16 int baja(struct empleado empleados[100], int num_emp);
17 void consultarDatosEmpleado(struct empleado empleados[100], int num_emp);
18
19 #endif // OPERACIONES_H_INCLUDED

```

Nos posicionamos después de la función `baja()` en el archivo `operaciones.c`

```

109 void consultarDatosEmpleado(struct empleado empleados[100], int num_emp){//implementamos la función
110     int id;//variable que capturará el id que se desea modificar
111     printf("Introduce id del empleado cuyos datos quieres consultar\n");//solicita id
112     scanf("%d",&id);//lee el id ingresado por el usuario, y lo almacena en la variable id
113
114     int existe = comprobarId(empleados,num_emp,id);//asignamos a existe el 0 o 1 de la función comprobarId
115     if (existe == 1){//si existe es igual a 1 se imprimirá los datos del registro con el id solicitado
116         for (int i=0; i<num_emp; i++){
117             if (empleados[i].id == id){
118                 printf("Listando datos del empleado con id: %d\n", empleados[i].id);
119                 printf("Nombre: %s\n",empleados[i].nombre);
120                 printf("Sueldo: %.2f\n", empleados[i].sueldo);
121                 printf("Horas semanales: %d\n",empleados[i].horas);
122             }
123         }
124     }
125     else{//si existe es diferente a 1 entonces no se encontró al empleado
126         printf("No se puede consultar los datos del empleado. ID no existe\n");
127     }
128 }
129
130
131

```

Ahora vamos a actualizar el `main()` del archivo `main.c` para llamar al método `baja`, evaluando el valor

| | | | | |
|-----------|-------------------|------------------|------------------|------------------|
| *main.c X | *persistencia.h X | persistencia.c X | *operaciones.h X | *operaciones.c X |
|-----------|-------------------|------------------|------------------|------------------|

```

33     else if(opcion == 3){
34         listarEmpleados(empleados,num_emp);
35     }
36     else if(opcion == 4){
37         consultarDatosEmpleado(empleados,num_emp);
38     }
39     else if(opcion == 5){

```


MODIFICAR SUELDO DE UN EMPLEADO operaciones.h

Declaremos la función `modificarSueldo()` en el archivo de tipo encabezado `operaciones.h`

```

15                                     //si se ha dado de al
16  int baja(struct empleado empleados[100], int num_emp);
17  void consultarDatosEmpleado(struct empleado empleados[100], int num_emp);
18  void modificarSueldo(struct empleado empleados[100], int num_emp);
19
20  #endif // OPERACIONES_H_INCLUDED

```

Nos posicionamos después de la función `consultarDatosEmpleado()` en el archivo `operaciones.c`

```

132 void modificarSueldo(struct empleado empleados[100], int num_emp){//implementamos la función
133
134     int id;//variable que capturará el id que se desea modificar
135     printf("Introduce id del empleado cuyo sueldo quieres modificar\n");//solicita id
136     scanf("%d",&id);//lee el id ingresado por el usuario, y lo almacena en la variable id
137
138     int existe = comprobarId(empleados,num_emp,id);//asignamos a existe el 0 o 1 de la función comprobarId
139
140     if (existe == 1){//si existe es igual a 1 se imprimirá los datos del registro con el id solicitado
141         float nuevo_sueldo;
142         printf("Introduce el nuevo sueldo del empleado\n");
143         scanf("%f",&nuevo_sueldo);
144         for(int i=0; i<num_emp; i++){
145             if (empleados[i].id == id){
146                 empleados[i].sueldo = nuevo_sueldo;
147                 printf("Sueldo actualizado\n");
148             }
149         }
150
151     }
152     else{
153         printf("No se puede modificar el sueldo del empleado. ID no existe\n");
154     }
155 }

```

Ahora vamos a actualizar el `main()` del archivo `main.c` para llamar al método `modificarSueldo()`, evaluando el valor

```

39     else if(opcion == 5){
40         modificarSueldo(empleados,num_emp);
41     }
42     else if(opcion == 6){
43         printf("Borra\n");

```

GUARDAR LOS CAMBIOS EN EL FICHERO **persistencia.h**

Crearemos la función **guardarEmpleados()** y la declararemos en el archivo **persistencia.h**

```

1  #ifndef PERSISTENCIA_H_INCLUDED
2  #define PERSISTENCIA_H_INCLUDED
3  //creamos el encabezado que recibe el arreglo de empleados, y es entero
4  //porque devolverá el número que haya cargado del archivo
5  int cargarEmpleados(struct empleado empleados[100]);
6  //creamos el encabezado que recibe el arreglo de empleados, y la variable
7  //entera num_emp que contiene el total de registros que contiene el fichero
8  void guardarEmpleados(struct empleado empleados[100], int num_emp);
9
10 #endif // PERSISTENCIA_H_INCLUDED

```

La implementación de la función **guardarEmpleados()** la realizaremos dentro de **persistencia.c**

```

48 }
49 void guardarEmpleados(struct empleado empleados[100], int num_emp){//recibe un array del tipo struct
50     FILE *f;//creamos una variable del tipo fichero
51     f = fopen("empleados.txt","w");//abrimos el archivo en modo escritura, esto borra todo lo que existe y añade
52         //la nueva información que ya tenemos en el arreglo
53     for (int i=0; i<num_emp-1; i++){//recorremos todas las posiciones del arreglo entre 0 y num_emp
54         //entonces el i<num_emp implica que solo imprimiremos hasta el penultimo dato
55         fprintf(f, "%s;%d;%2f;%d\n",empleados[i].nombre,empleados[i].id,empleados[i].sueldo,empleados[i].horas);
56         //imprimimos en el fichero, f es el fichero, y enviamos los campos en los formatos correspondientes
57         //separados por punto y coma y de ultimo tenemos el salto de línea, luego enviamos los datos
58         //almacenando campo por campo en el array
59     }
60     //el salto de línea que colocamos se aplicará al final despues del ultimo registro ingresado, por lo tanto
61     //seria un registro en blanco y eso ocasionaria un problema, entonces debemos de leer por separado el ultimo dato
62     fprintf(f, "%s;%d;%2f;%d",empleados[num_emp-1].nombre,empleados[num_emp-1].id,empleados[num_emp-1].sueldo,empleados[num_emp-1].horas);
63
64     fclose(f);//cerramos el archivo
65 }

```

Ahora vamos a actualizar el **main()** del archivo **main.c** para llamar al método **guardarEmpleados()**, previo al cierre del programa

```

45     mostrarMenu();//mostramos el menú nuevamente
46     scanf("%d",&opcion);//pedimos nuevamente que elija una opción
47 }
48 guardarEmpleados(empleados,num_emp);//antes de cerrar el programa actualizamos el fichero
49 printf("GRACIAS POR USAR EL GESTOR DE EMPLEADOS. ;HASTA PRONTO!\n\n");
50 }

```

Recuerde elegir la opción **SALIR** para que actualice el archivo **empleados.txt**

AHORA USTED DEBERÁ REALIZAR LA ACTUALIZACIÓN DE HORAS DE UN EMPLEADO SOLICITANDO EL ID