

Bibliotecas de funciones

En el lenguaje C se puede trabajar modularmente de muchas formas, sin embargo nos centraremos en una forma ligada al concepto de "Reutilización de código". La idea es plantear el trabajo con funciones pensando que cada vez que desarrollemos un proyecto no lo hagamos todo desde cero. Esto es que cuando diseñemos por primera vez un grupo de funciones las coloquemos, desde un principio, en un archivo especial, de modo que la próxima vez que necesitemos alguna de esas funciones solo tengamos que incorporar ese archivo al proyecto sin tener que volverlo a escribir, reutilizando el código. En otras palabras, el trabajo en este nuevo proyecto se limitará a repetir de modo similar lo que hacemos cuando necesitamos usar por ejemplo la función `printf`, esto es colocar la definición de esa función mediante la cláusula `#include` y luego simplemente utilizarla. Ese archivo especial del que hablamos se suele denominar "**Biblioteca de funciones**" y la idea es que en una de esas bibliotecas se coloque un grupo de funciones que estén relacionadas de alguna forma, por ejemplo funciones como la media, mediana, moda, desviación estándar, etc. se pueden agrupar en una biblioteca de funciones estadísticas, funciones para graficar líneas, círculos, cuadriláteros, etc. pueden agruparse en una biblioteca de funciones gráficas.

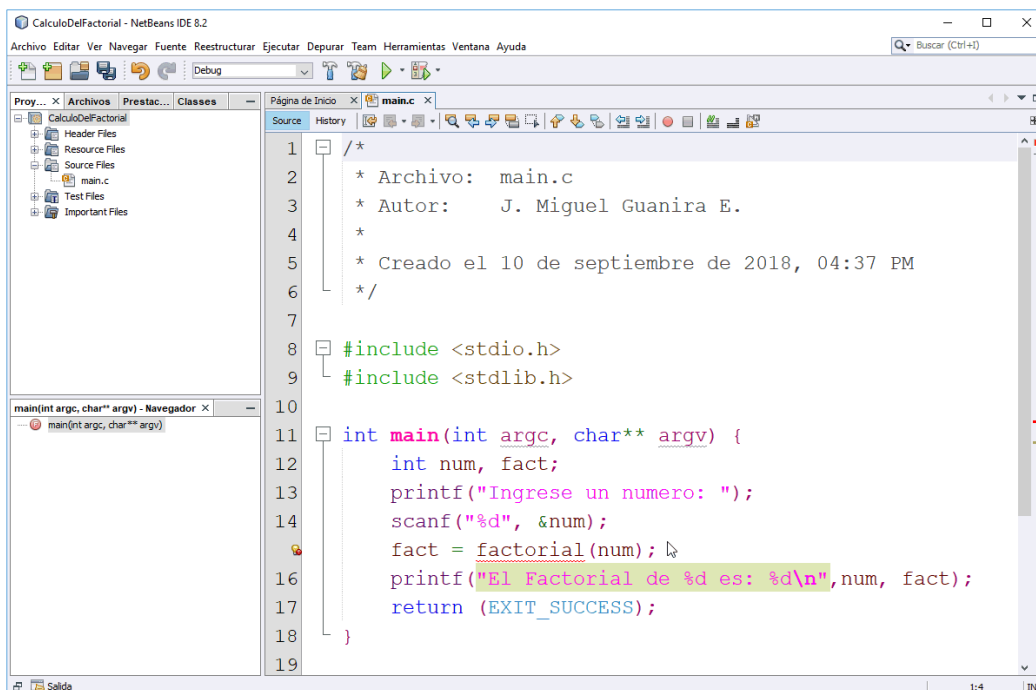
Un ejemplo explicará lo que queremos hacer:

Supongamos que queremos desarrollar un programa que nos permita calcular el factorial de un número, esto es, se ingresa un número entero y el programa calcula y nos devuelve su factorial.

Entonces las tareas que debe realizar el programa sería:

1. Leer el valor entero
2. Calcular el factorial
3. Mostrar el resultado

Plasmar este programa en NetBeans es una tarea sencilla, simplemente creemos un proyecto y escribamos el código siguiente:



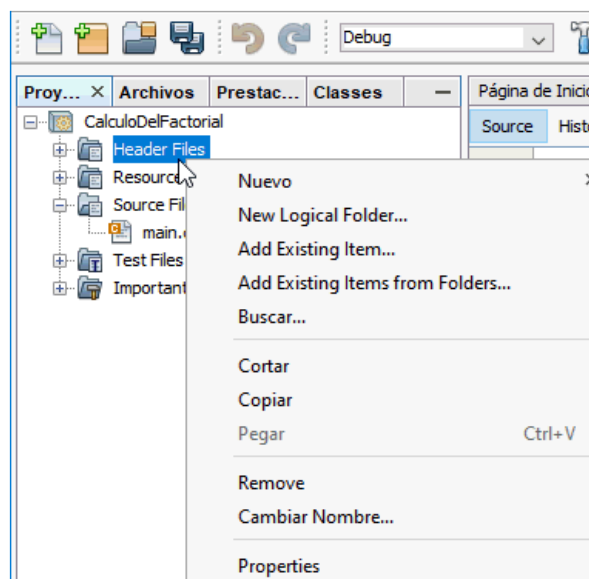
```
1  /*
2   * Archivo:  main.c
3   * Autor:    J. Miguel Guanira E.
4   *
5   * Creado el 10 de septiembre de 2018, 04:37 PM
6   */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 int main(int argc, char** argv) {
12     int num, fact;
13     printf("Ingrese un numero: ");
14     scanf("%d", &num);
15     fact = factorial(num);
16     printf("El Factorial de %d es: %d\n", num, fact);
17     return (EXIT_SUCCESS);
18 }
19
```

Observe que nuestro programa usará una función "factorial" de la misma forma que usamos printf o scanf, con la diferencia que nuestra función factorial está subrayada en rojo, lo que significa que no se reconocerá el significado la palabra "factorial" por lo que si compilamos el programa nos dará una error.

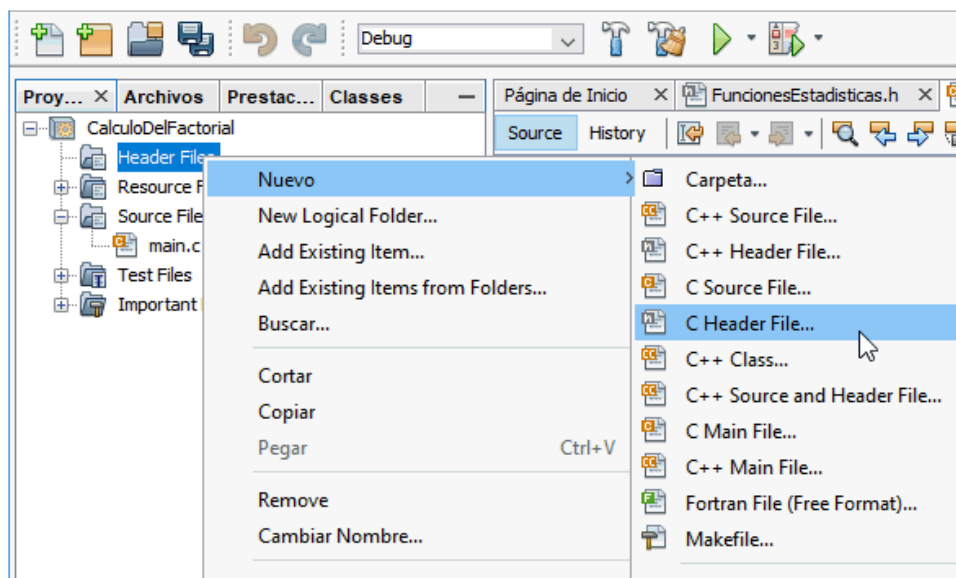
Para corregir el error debemos realizar dos tareas:

1. Declara la función factorial
2. Implementar la función factorial.

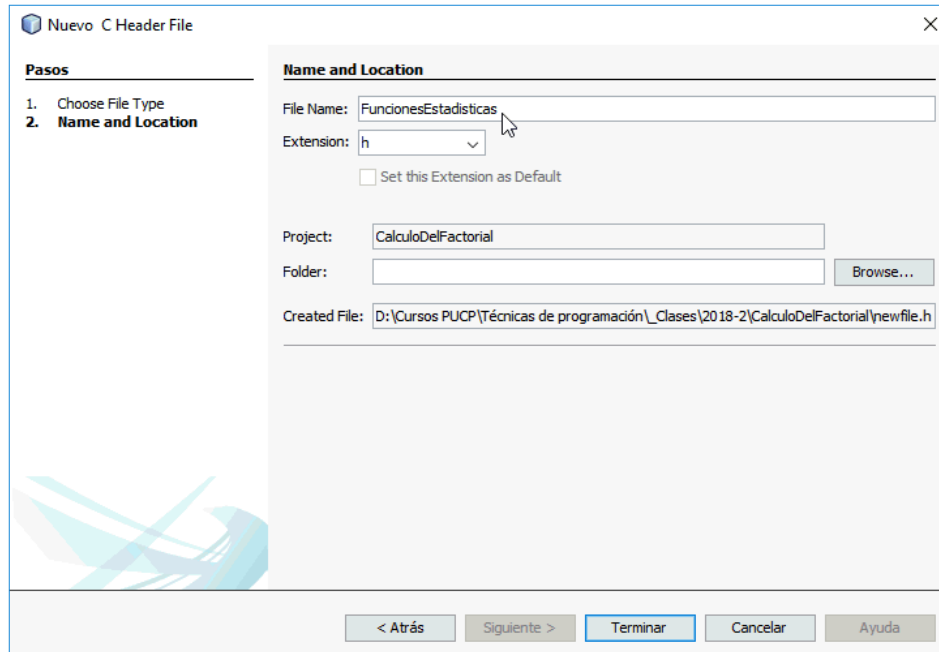
Estas dos tareas deben ser hechas pensando siempre en la reutilización del código. Entonces, debemos crear primero "un archivo de cabecera", "header file" o "archivo.h" donde coloquemos allí su declaración. Debe tener en cuenta que nuestro ejemplo solo define una función pero, para que esto sea práctico, deberíamos pensar en varias funciones que se coloque en este archivo de cabecera. Para esto nos dirigimos a las carpetas del proyecto y en la carpeta "Header Files" presionaremos el botón derecho de mouse para que se despliegue un menú como se ve a continuación:



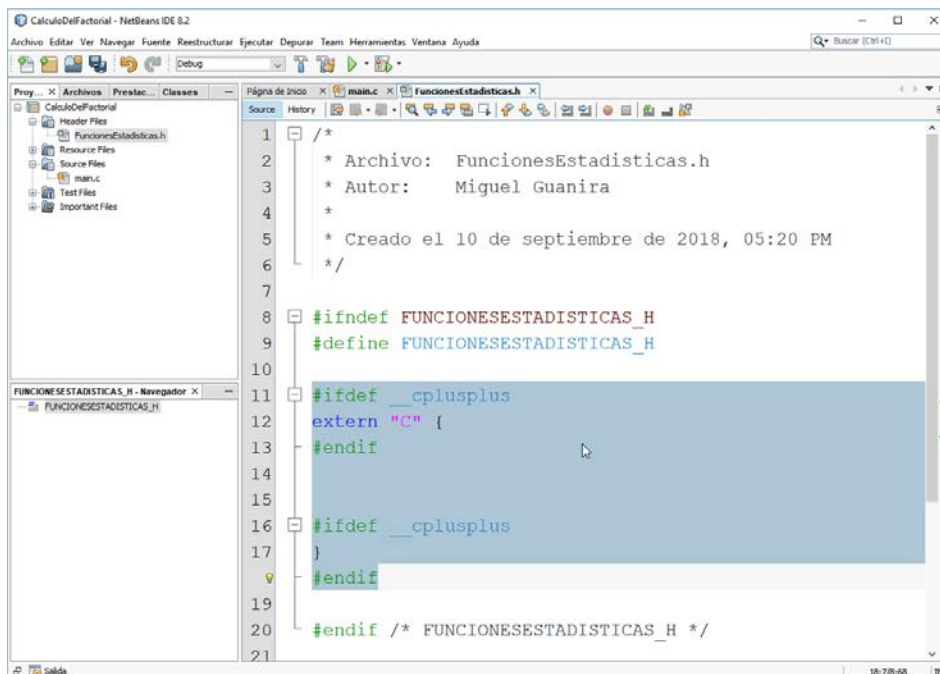
Allí nos dirigimos a la opción "Nuevo" y se desplegará otro menú, allí se elegirá la opción C Header File..., como el que se aprecia a continuación:



Inmediatamente aparecerá una ventana, en ella, en el recuadro "File Name" colocaremos un nombre al archivo, este no debe ser el nombre de la función porque, como veremos luego no solo va a contener la definición de la función "factorial" sino todas las que queramos, por ejemplo podríamos ponerle "FuncionesEstadisticas". Como vemos a continuación:

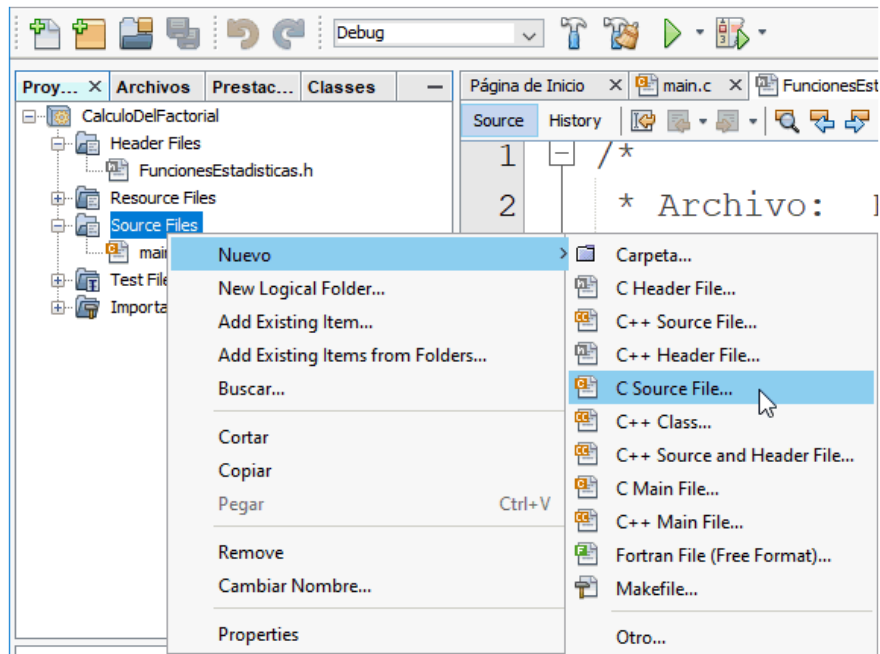


Luego presionamos el botón "Terminar" con lo que podrá observar lo siguiente:

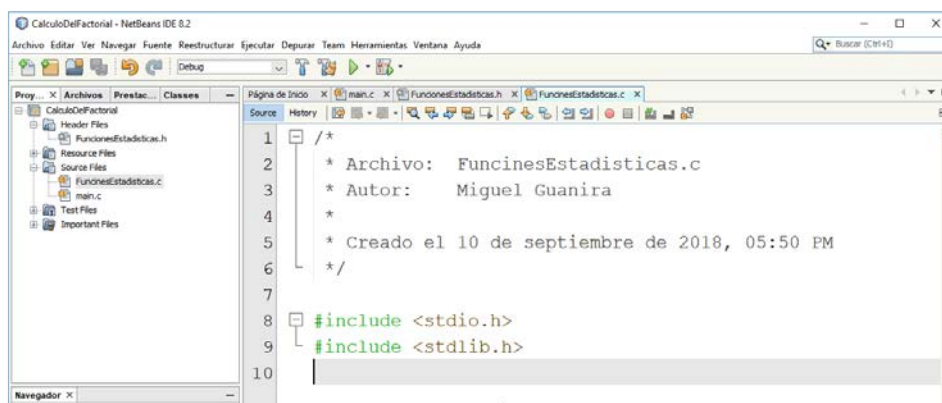


Como puede ver se ha creado el archivo "FuncionesEstadisticas.h". Usted puede borrar la zona marcada en azul porque no la usaremos en este curso.

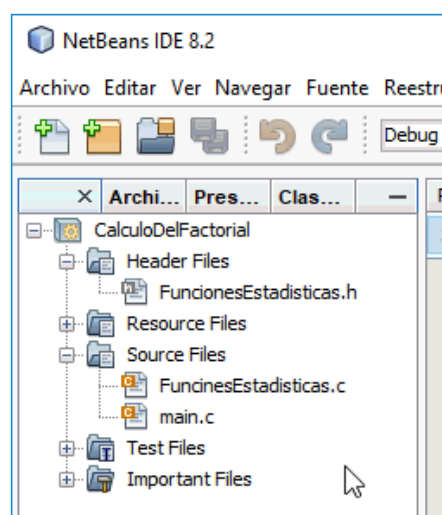
Luego crearemos el archivo donde colocaremos la implementación de nuestra función "factorial" (todas que queremos re utilizar). Para esto repetiremos la creación del archivo como lo hicimos con el "Header Files" pero ahora con el "Source Files", como se ve en la figura siguiente:



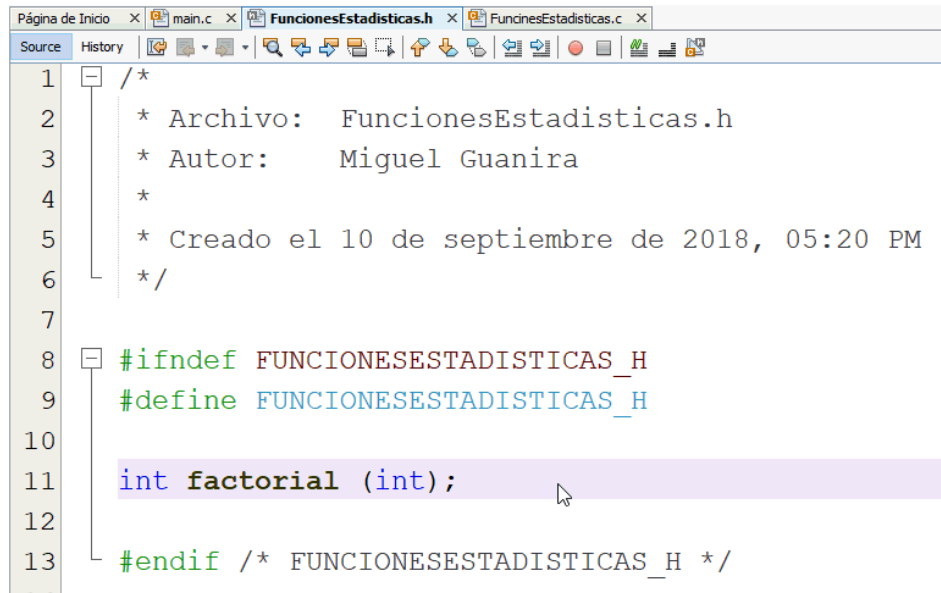
En la ventana que aparece coloque siempre como nombre (como una buena práctica de programación) el mismo que usó para el archivo de cabecera, en este caso `FuncionesEstadisticas`. Con esto ahora tendremos otro archivo con extensión `.c` como se ve a continuación:



Observe que su proyecto ahora está formado por tres archivos. En la ventana de proyectos a la izquierda del NetBeans se puede ver esto.

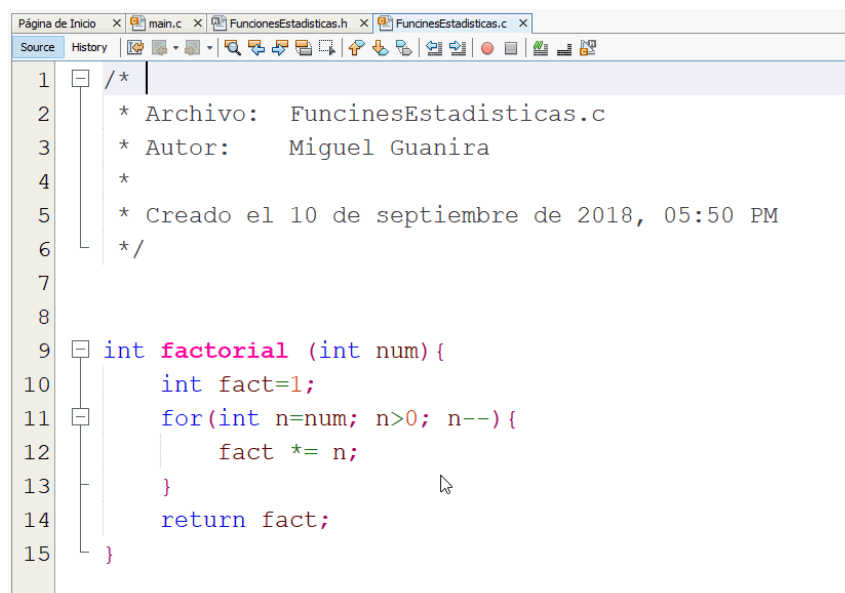


Finalmente escribiremos el código correspondiente, en este caso en el archivo de cabeceras, colocaremos el encabezado de la función factorial, como se ve a continuación:



```
Página de Inicio x main.c x FuncionesEstadisticas.h x FuncinesEstadisticas.c x
Source History
1  /*
2      * Archivo:  FuncionesEstadisticas.h
3      * Autor:    Miguel Guanira
4      *
5      * Creado el 10 de septiembre de 2018, 05:20 PM
6      */
7
8  #ifndef FUNCIONESESTADISTICAS_H
9      #define FUNCIONESESTADISTICAS_H
10
11     int factorial (int);
12
13 #endif /* FUNCIONESESTADISTICAS_H */
```

En el archivo fuente "FuncionesEstadisticas.c" escribiremos la implementación de la función, como se muestra a continuación:



```
Página de Inicio x main.c x FuncionesEstadisticas.h x FuncinesEstadisticas.c x
Source History
1  /*
2      * Archivo:  FuncinesEstadisticas.c
3      * Autor:    Miguel Guanira
4      *
5      * Creado el 10 de septiembre de 2018, 05:50 PM
6      */
7
8
9  int factorial (int num) {
10      int fact=1;
11      for(int n=num; n>0; n--){
12          fact *= n;
13      }
14      return fact;
15  }
```

Finalmente debemos saber que cuando compilemos el proyecto, cada módulo con extensión ".cpp" se compilará de manera independiente y luego se enlazará al programa ejecutable ".exe", por lo tanto cualquier función que utilicemos en un módulo debe ser declarada antes de utilizarla. Por lo tanto, así como cuando usamos en main la función **printf** necesitamos incluir (**#include**) la biblioteca **stdio.h**, la definición de nuestra función factorial debe ser incluida en todos los módulos en donde se use. Regresando a nuestro ejemplo, por esa razón, para que se pueda compilar el módulo "main.c" del programa debemos colocar la definición de la función de la función factorial colocando la orden **#include** correspondiente como se ve a continuación:

```

4      *
5      * Creado el 10 de septiembre de 2018, 04:37 PM
6      */
7
8      #include <stdio.h>
9      #include <stdlib.h>
10     #include "FuncionesEstadisticas.h"
11
12     int main(int argc, char** argv) {
13         int num, fact;
14         printf("Ingrese un numero: ");
15         scanf("%d", &num);
16         fact = factorial(num);
17         printf("El Factorial de %d es: %d\n", num, fact);
18         return (EXIT_SUCCESS);
19     }

```

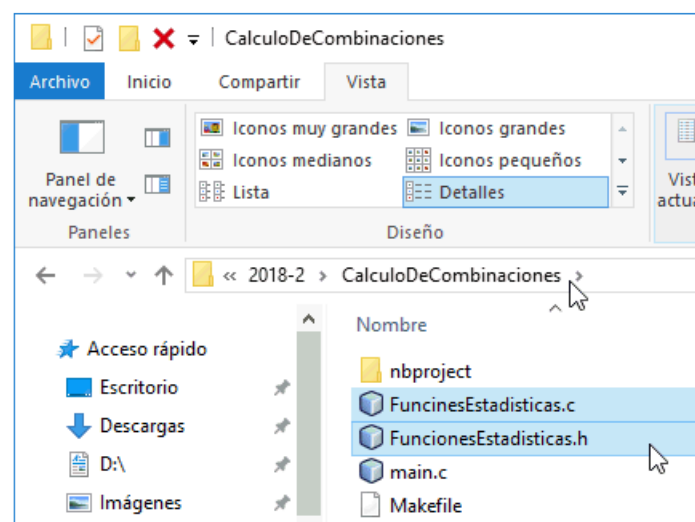
Una vez realizado esto, su programa debe ejecutarse sin problemas. Sin embargo lo más importante de este proceso es que la próxima vez que en un programa necesitemos realizar el cálculo del factorial ya no tendremos que volver a escribirla, solo la usaremos como usamos la función printf.

Reutilización de código

Ahora veremos cómo podemos implementar un proyecto que requiera el cálculo del factorial de un número sin volverlo a escribir. En el siguiente ejemplo queremos desarrollar un proyecto en el que se calcule las combinaciones de "n" elementos tomados de "p" en "p" sin repetición, para esto debemos emplear la siguiente formula:

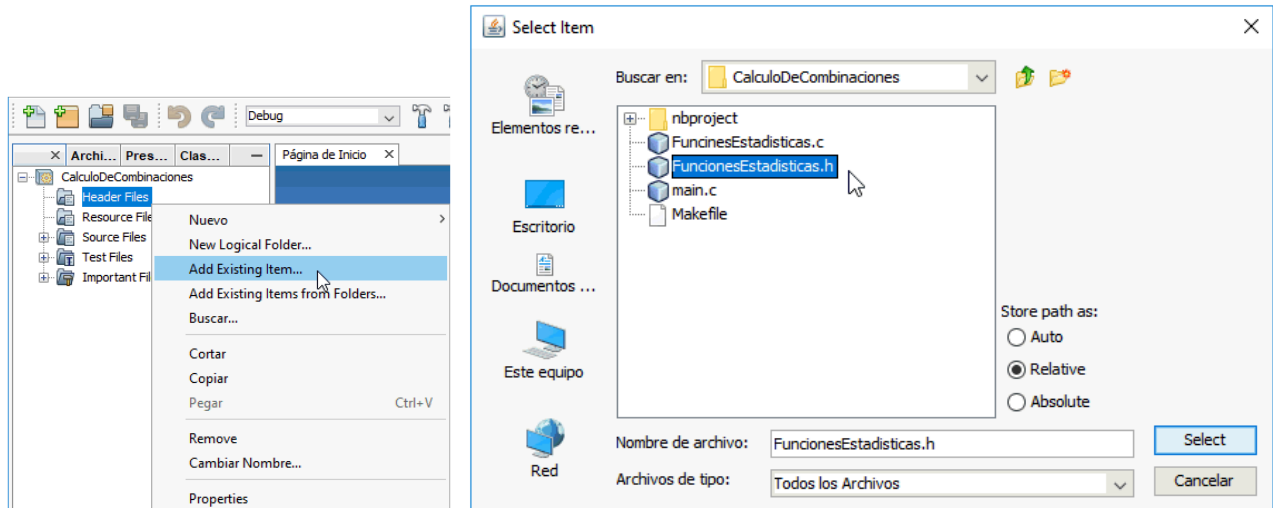
$$C_p^n = \binom{n}{p} = \frac{n!}{p! \cdot (n-p)!}$$

Creemos entonces en NetBeans el proyecto "CalculoDeCombinaciones". Luego, antes de comenzar a escribir el código del programa, debemos copiar, en la carpeta que contiene nuestro nuevo proyecto, el archivo de cabecera (.h) y de implementación (.c) donde están la definición e implementación de nuestra función factorial que se encuentra en la carpeta del proyecto anterior. La carpeta debe quedar como se muestra a continuación:

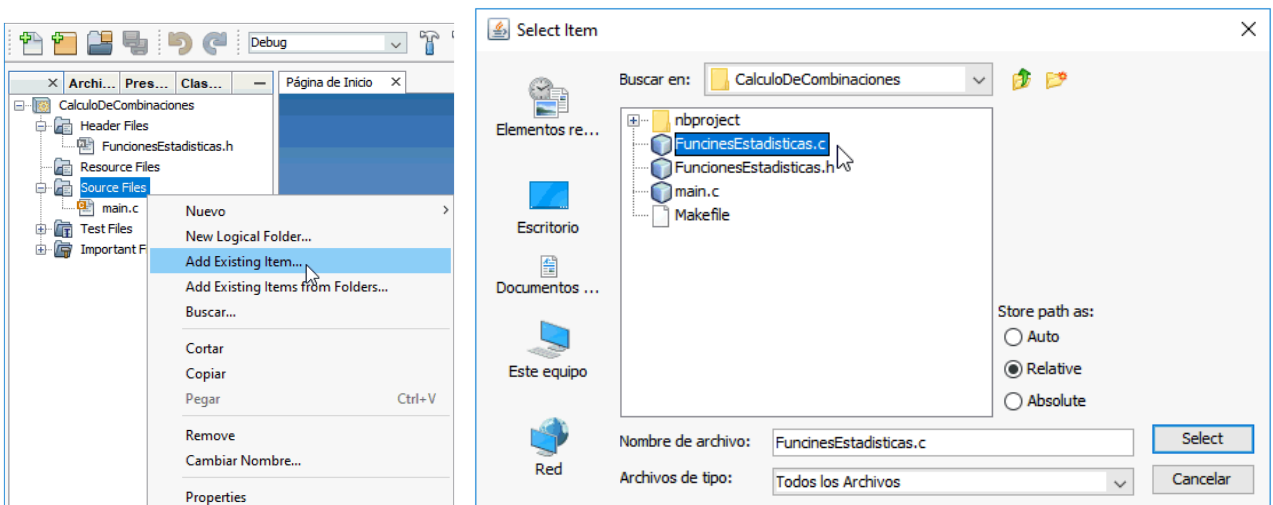


Luego debemos ligar al proyecto estos archivos, para esto nos dirigimos al NetBeans y en la ventana izquierda haremos algo similar a lo que hicimos en el anterior proyecto pero en el menú que se despliega elegiremos la opción "Add Existing Item...", y en la

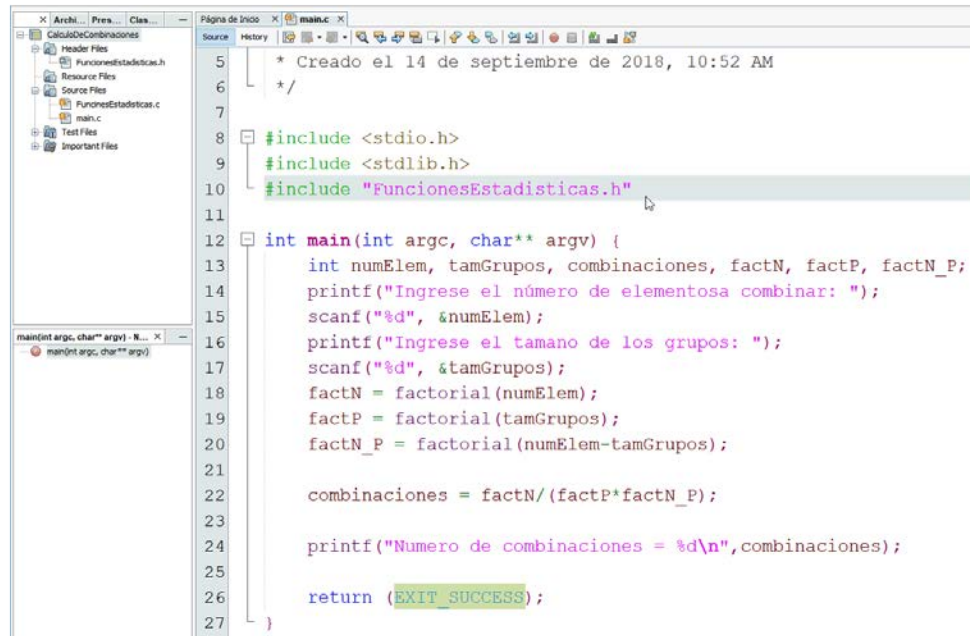
ventana que aparece seleccionamos el archivo "FuncionesEstadistica.h" como se aprecia a continuación:



De la misma manera lo hacemos con el archivo que contiene la implementación de la función factorial, como se ve a continuación:



Una vez hecho esto, podemos escribir y ejecutar nuestro programa de combinaciones solo escribiendo código en la función main como se muestra a continuación:



The screenshot shows a C++ IDE with a project named 'CálculoDeCombinaciones'. The file explorer on the left shows a directory structure with 'Header Files' (containing 'FuncionesEstadisticas.h'), 'Resource Files', 'Source Files' (containing 'FuncionesEstadisticas.c' and 'main.c'), 'Test Files', and 'Important Files'. The main editor window displays the code for 'main.c'. The code includes standard headers and a custom header, defines the main function, and calculates combinations using factorials. A console window at the bottom shows the program's output.

```
5  * Creado el 14 de septiembre de 2018, 10:52 AM
6  */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include "FuncionesEstadisticas.h"
11
12 int main(int argc, char** argv) {
13     int numElem, tamGrupos, combinaciones, factN, factP, factN_P;
14     printf("Ingrese el número de elementosa combinar: ");
15     scanf("%d", &numElem);
16     printf("Ingrese el tamano de los grupos: ");
17     scanf("%d", &tamGrupos);
18     factN = factorial(numElem);
19     factP = factorial(tamGrupos);
20     factN_P = factorial(numElem-tamGrupos);
21
22     combinaciones = factN/(factP*factN_P);
23
24     printf("Numero de combinaciones = %d\n",combinaciones);
25
26     return (EXIT_SUCCESS);
27 }
```

main(int argc, char** argv) - N...
main(int argc, char** argv)