

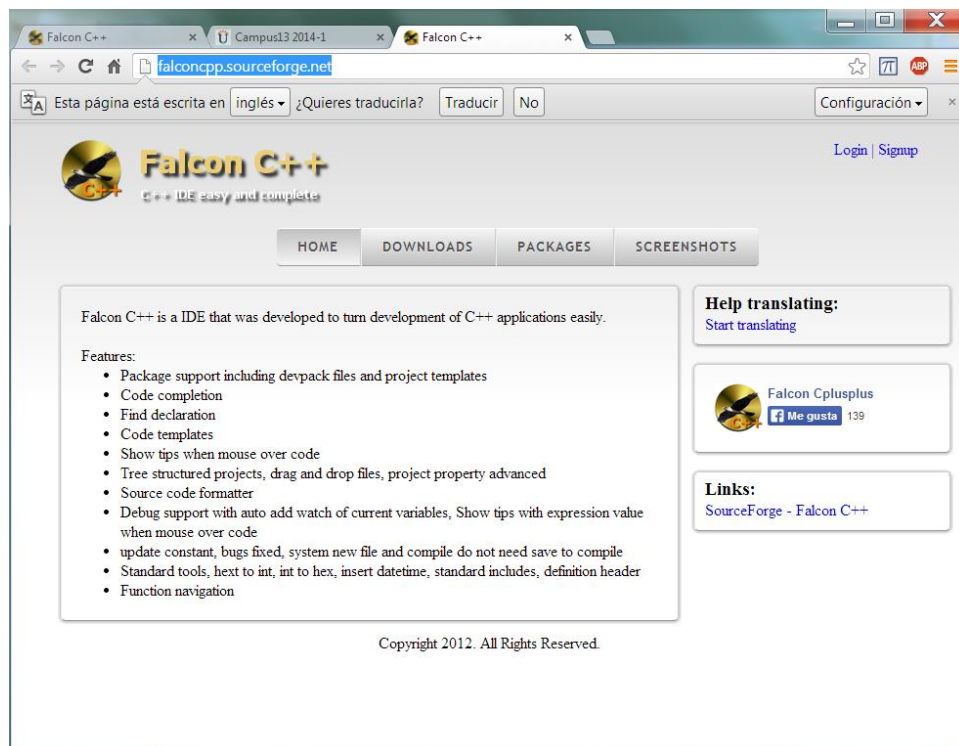
ENTORNO DE DESARROLLO IDE FALCON C++

Falcon c++ es un IDE muy sencillo, moderno, diseñado para estudiantes, y no por ello menos potentes que cualquier otro IDE profesional, practico e intuitivo y fácil de manejar, corre en Pc's bajo Windows de 32 y 64 bits, cosa que no ocurre con algunas versiones del Borland y DevC++.

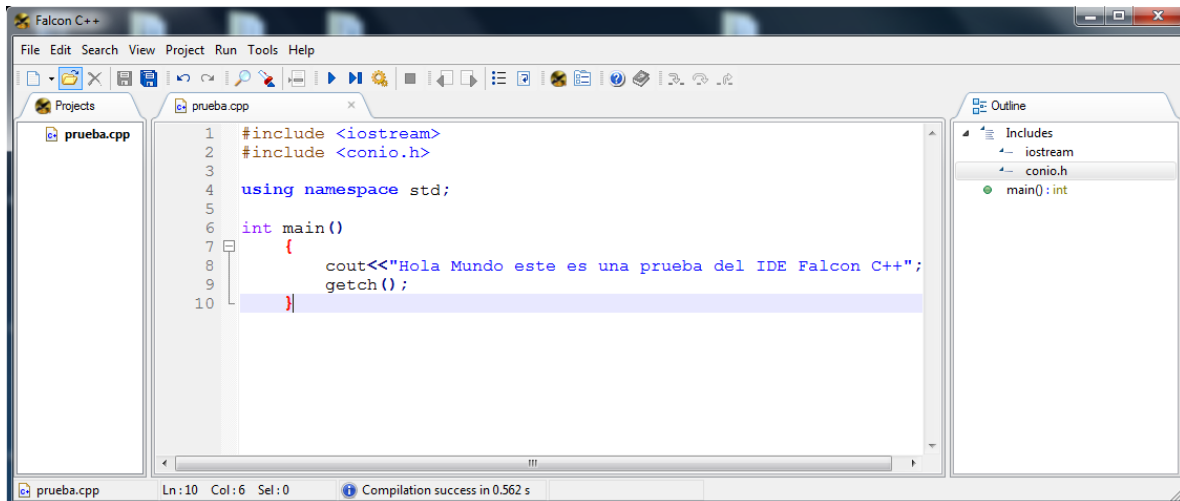
Este IDE tiene una interfaz muy sencilla, pero muy potente, el editor esta actualizado con las últimas novedades, permite insertar las palabras instrucciones en la medida que uno escribe, trae incorporado el compilador Mingw, y no hay necesidad de crear proyectos si no se requiere. También es portable, lo único que hay que hacer es copiar la carpeta **Falcon C++** de archivos de programas(x86) a la memoria y el actualiza todas las rutas.

En conclusión es un IDE para estudiantes, También tiene un controlador de paquetes para librerías no oficiales como la conio.

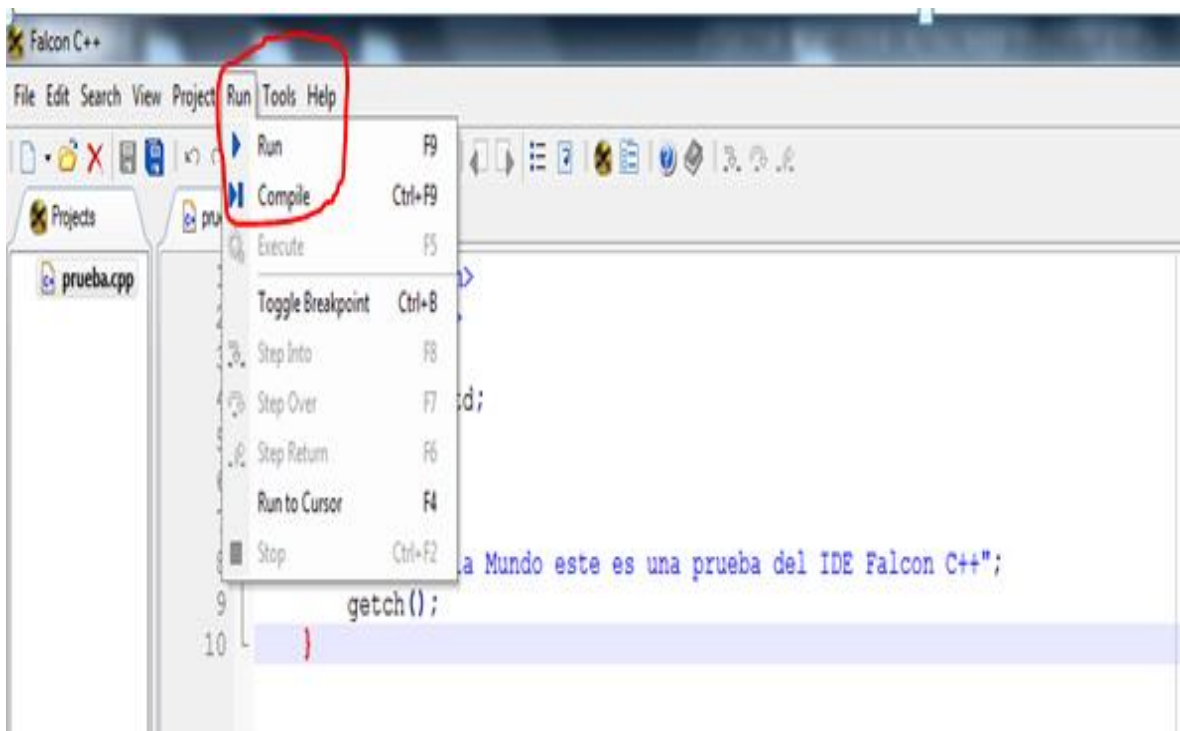
Para mirar cómo se descarga e instala puede acceder al videotutorial en el siguiente link: <https://www.youtube.com/watch?v=vbnXr75bXmA>
Lo pueden descargar en <http://falconcpp.sourceforge.net>



Interfaz de edición de Falcon C++

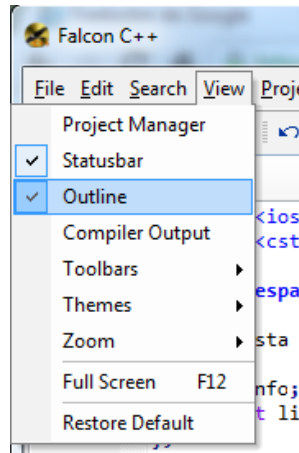


Opciones de Compilación y ejecución

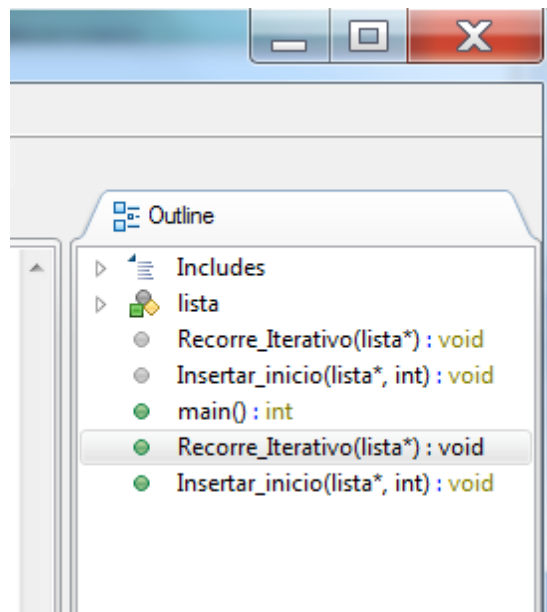


COMO USAR EL DEPURADOR DE FALCON C++


Lo primero que debemos verificar es que la ventana Outline está activa, para activarla vamos al menú wiew(ver) y damos un clic en Outline.





Una vez activada la ventana aparecerá en el loado derecho de la ventana del Falcon c++.



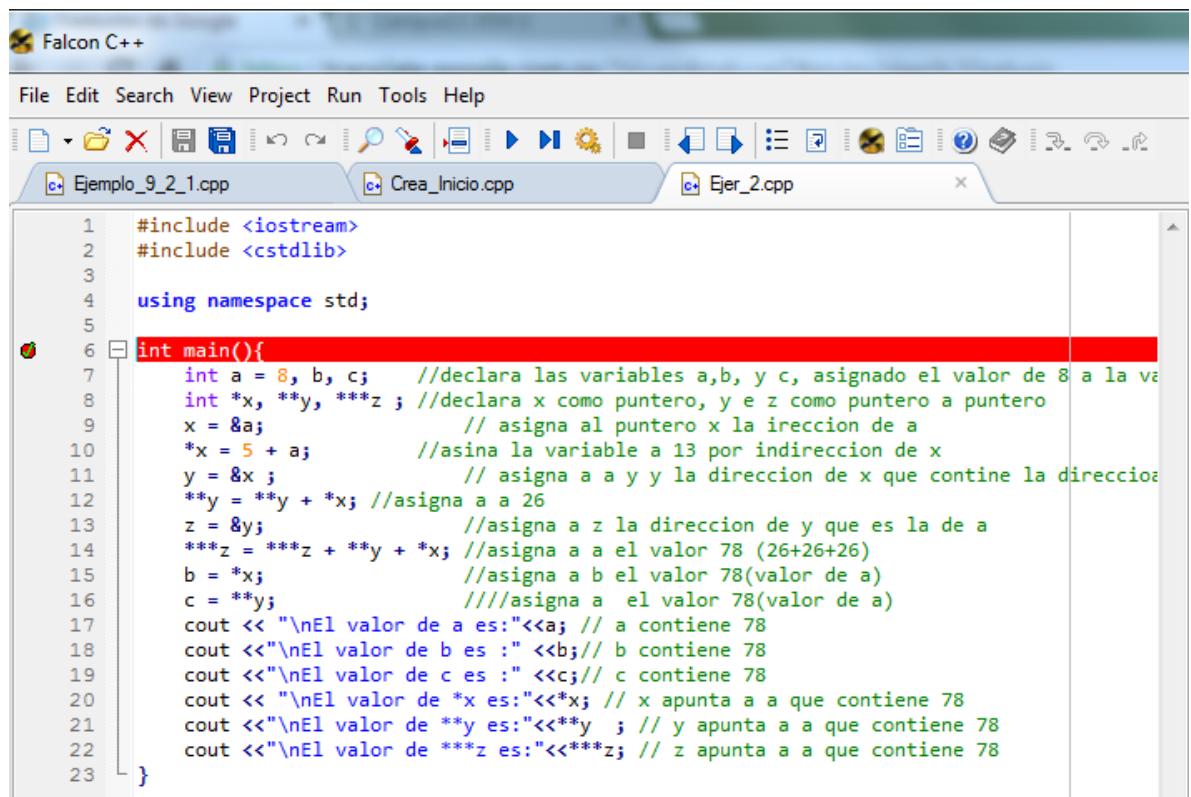
Los botones para manejar el depurador son:

-  **step into**(entrar en). este botón permite ejecutar el programa paso a paso, su comando es f8.

-  **Step Over** (paso por encima). Este botón permite ejecutar el programa por corchetes, es decir cuando ejecuta en solo paso los que este entre corchetes({}). Su comando es F7.
-  **step return**(paso de retorno). Esta opción ejecuta el programa por funciones, es decir, al pulsar el botón ejecuta una función completa y retorna su resultado si es el caso. Su comando es F6.

EJECUCIÓN PASO A PASO



1. Marcamos un punto de interrupción, para ello, damos un clic en el número de la línea en la cual queremos parar el programa.



The screenshot shows the Falcon C++ IDE interface. The menu bar includes File, Edit, Search, View, Project, Run, Tools, and Help. The toolbar contains various icons for file operations, editing, and execution. The project explorer at the top shows three files: Ejemplo_9_2_1.cpp, Crea_Inicio.cpp, and Ejer_2.cpp. The main editor window displays the code for Ejemplo_9_2_1.cpp. A red line is drawn across line 6, indicating a breakpoint. The code is as follows:

```

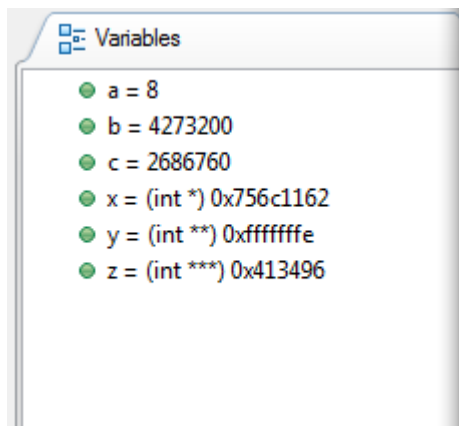
1  #include <iostream>
2  #include <cstdlib>
3
4  using namespace std;
5
6  int main(){
7      int a = 8, b, c; //declara las variables a,b, y c, asignado el valor de 8 a la va
8      int *x, **y, ***z ; //declara x como puntero, y e z como puntero a puntero
9      x = &a; // asigna al puntero x la direccion de a
10     *x = 5 + a; //asigna la variable a 13 por direccion de x
11     y = &x ; // asigna a a y y la direccion de x que contiene la direccion
12     **y = **y + *x; //asigna a a 26
13     z = &y; //asigna a z la direccion de y que es la de a
14     ***z = ***z + **y + *x; //asigna a a el valor 78 (26+26+26)
15     b = *x; //asigna a b el valor 78(valor de a)
16     c = **y; ///asigna a el valor 78(valor de a)
17     cout << "\nEl valor de a es:"<<a; // a contiene 78
18     cout << "\nEl valor de b es : " <<b; // b contiene 78
19     cout << "\nEl valor de c es : " <<c; // c contiene 78
20     cout << "\nEl valor de *x es:"<<*x; // x apunta a a que contiene 78
21     cout << "\nEl valor de **y es:"<<**y ; // y apunta a a que contiene 78
22     cout << "\nEl valor de ***z es:"<<***z; // z apunta a a que contiene 78
23 }
  
```

2. Damos un clic en el botón  run(f9), con lo cual el programa se ejecutara hasta el punto de interrupción que marcamos en el paso anterior.
3. Damos un clic en el botón  step into, o pulsamos f8 desde el teclado, con lo cual se resaltara la línea de código que se está

ejecutando en ese momento y aparece una flecha al lado izquierdo de la respectiva línea.

```
1  #include <iostream>
2  #include <cstdlib>
3
4  using namespace std;
5
6  int main(){
7      int a = 8, b, c;    //declara las variables a,b, y c, asignado el valor de 8 a la v
8      int *x, **y, ***z; //declara x como puntero, y e z como puntero a puntero
9      x = &a;            // asigna al puntero x la direccion de a
10     *x = 5 + a;         //asigna la variable a 13 por direccion de x
11     y = &x;             // asigna a y la direccion de x que contiene la direccion
12     **y = **y + *x;     //asigna a a 26
13     z = &y;             //asigna a z la direccion de y que es la de a
14     ***z = ***z + **y + *x; //asigna a a el valor 78 (26+26+26)
15     b = *x;             //asigna a b el valor 78(valor de a)
16     c = **y;            ///asigna a el valor 78(valor de a)
17     cout << "\nEl valor de a es:"<<a; // a contiene 78
18     cout << "\nEl valor de b es : " <<b; // b contiene 78
19     cout << "\nEl valor de c es : " <<c; // c contiene 78
20     cout << "\nEl valor de *x es:"<<*x; // x apunta a a que contiene 78
21     cout << "\nEl valor de **y es:"<<**y ; // y apunta a a que contiene 78
22     cout << "\nEl valor de ***z es:"<<***z; // z apunta a a que contiene 78
23 }
```

4. En la ventana outline aparecen las variables declaradas, con sus respectivos valores, como podemos observar en la siguiente imagen las variables que al declararlas le hemos asignado un valor, aparece dicho valor, los que no han sido asignados aparecen con un valor arbitrario, como también los punteros.



5. A medida que vamos ejecutando el programa, nos ira mostrando los valores que van almacenando las variables incluidas las direcciones de memoria a la que apuntan los punteros.

```
1 #include <iostream>
2 #include <cstdlib>
3
4 using namespace std;
5
6 int main(){
7     int a = 8, b, c; //declara las variables a,b, y c, asignado el valor de 8 a la va
8     int *x, **y, ***z; //declara x como puntero, y e z como puntero a puntero
9     x = &a; // asigna al puntero x la direccion de a
10    *x = 5 + a; //asina la variable a 13 por direccion de x
11    y = &x; // asigna a y la direccion de x que contiene la direccion
12    **y = *y + *x; //asigna a a 26
13    z = &y; //asigna a z la direccion de y que es la de a
14    ***z = ***z + **y + *x; //asigna a a el valor 78 (26+26+26)
15    b = *x; //asigna a b el valor 78(valor de a)
16    c = **y; ///asigna a el valor 78(valor de a)
17    cout << "\nEl valor de a es:" << a; // a contiene 78
18    cout << "\nEl valor de b es : " << b; // b contiene 78
```

6. En la consola podemos verificar la salida del programa.

```
1 #include <iostream>
2 #include <cstdlib>
3
4 using namespace std;
5
6 int main(){
7     int a = 8, b, c; //declara las variables a,b, y c, asignado el valor de 8 a la va
8     int *x, **y, ***z; //declara x como puntero, y e z como puntero a puntero
9     x = &a; // asigna al puntero x la direccion de a
10    *x = 5 + a; //asina la variable a 13 por direccion de x
11    y = &x; // asigna a y la direccion de x que contiene la direccion
12    **y = *y + *x; //asigna a a 26
13    z = &y; //asigna a z la direccion de y que es la de a
14    ***z = ***z + **y + *x; //asigna a a el valor 78 (26+26+26)
15    b = *x; //asigna a b el valor 78(valor de a)
16    c = **y; ///asigna a el valor 78(valor de a)
17    cout << "\nEl valor de a es:" << a; // a contiene 78
18    cout << "\nEl valor de b es : " << b; // b contiene 78
19    cout << "\nEl valor de c es : " << c; // c contiene 78
20    cout << "\nEl valor de *x es:" << *x; // x apunta a a que contiene 78
```

7. En el momento de edición, la ventana outline nos permite desplazarnos a una determina función con solo dar doble clic sobre el nombre de esta.

```

31         insertar( );
32         break;
33     case 2:
34         extraer( );
35         break;
36     case 3:
37         visualizar( );
38         break;
39     }
40     }while (opc!= 4);
41     return 0;
42 }
43
44 void insertar(void)
45 {
46     AUXILIAR= new cola;
47     system("cls");
48     cout << "Nombre: ";
49     cin >> ws;
50     cin.getline(AUXILIAR -> nombre, 20);
51     AUXILIAR -> sig = NULL;
52     if (FINAL == NULL)
53     {
54         FINAL = CABEZA = AUXILIAR;
55     }
56
57     else
58     {
59         FINAL -> sig = AUXILIAR;
60         FINAL = AUXILIAR;
61     }
62 }

```

Includes

- NULL : struct cola* CABI
- AUXILIAR : struct cola*
- FINAL : struct cola*
- cola
 - nombre : char[20]
 - sig : struct cola*
- insertar() : void
- extraer() : void
- visualizar() : void
- main() : int
- insertar() : void
- extraer() : void
- visualizar() : void

8. Si nuestro programa contiene arreglos, estructuras o arreglos de estructuras también nos muestra como se cargan estas variables en la memoria en que ejecutamos el programa.

```

1  #include <iostream>
2
3  using namespace std;
4
5  struct t_persona
6  {
7      char nombre[30];
8      int edad;
9      int altura;
10     int peso;
11 };
12 typedef struct t_persona persona;
13
14 void mostrar_persona(persona *ptr);
15
16 int main()
17 {
18     int i;
19     persona empleados[]= {{"Mortimer, pepe",47,182,85}, {"Garcia, Luis",39,170,75}, {"Jimenez, Tom
20
21     persona *p; //puntero a estructura
22     p = empleados;
23     for(i=0; i<3; i++, p++)
24     {
25         mostrar_persona(p);
26     }

```

Variables

- i = 0
- empleados
 - [0]
 - nombre = "Mortimer, pepe"
 - [0] = '\0' <repeats 15 times>
 - edad = 47
 - altura = 182
 - peso = 85
 - [1]
 - nombre = "Garcia, Luis"
 - [0] = '\0' <repeats 17 times>
 - edad = 39
 - altura = 170
 - peso = 75
 - [2]
 - nombre = "Jimenez, Tomas"
 - [0] = '\0' <repeats 15 times>
 - edad = 18
 - altura = 175
 - peso = 80
- p = (persona *) 0x28fe84

Espero que este pequeño tutorial sea de utilidad y se animen probarlo... No se arrepentirán...