Lo que queremos hacer es ejecutar 2 o más funciones al mismo tiempo.

Primeramente necesitarás un compilador que soporte la biblioteca windows.h, puede ser el B orland C++ 5.5
o el
Dev-C++ 4.9.9.2
, luego usaremos
hilos
(threads), esto permite ejecutar en forma paralela 2 o más funciones al mismo tiempo.

Lo que haremos es crear un hilo por cada función que queramos ejecutar en forma paralela, asociamos cada hilo a la función a ejecutar. Luego iniciamos los 2 hilos con lo cual se ejecutarán las 2 funciones al mismo tiempo, no hace falta esperar a que la una función termine para que la otra empiece (aunque esto se puede controlar). Y luego haremos un truco de forma que la ejecución del programa continue únicamente cuando los 2 hilos terminen. Finalmente cerramos los hilos cuando su ejecución haya concluído.

Ahora empecemos:

Primero la cabecera:

codeDivStart()#include <windows.h>

Ahora definimos los prototipos de las 2 funciones que queremos ejecutar al mismo tiempo:

codeDivStart()void RunDialogOne();
void RunDialogTwo();

Definimos los 2 controladores, uno por cada hilo:

codeDivStart()HANDLE tPrimary, tSecond;

Nuestra función RunDialogOne y RunDialogTwo simplemente mostrarán un mensaje por pantalla, nada más:

```
codeDivStart()void RunDialogOne()
{
   MessageBox(NULL, "Ejecutando: RunDialogOne", "Hilo 1", MB_OK | MB_ICONINFORMATION);

   return;
}
```

```
void RunDialogTwo()
{
   // WaitForSingleObject(tPrimary, INFINITE);
   MessageBox(NULL, "Ejecutando: RunDialogTwo", "Hilo 2", MB_OK | MB_ICONINFORMATION);

   return;
}
```

Lo importante aquí es la instrucción return, la cual va al final de la función, mediante esta decimos que ha terminado la ejecución del hilo asociado a la función. Siempre hay que llamarla, si no lo hacemos nuestro programa no sabrá si ha terminado o no la ejecución de la función y por lo tanto se colgará. Fíjense en la línea comentada dentro de la función RunDialogTwo
, la cual es:

codeDivStart()WaitForSingleObject(tPrimary, INFINITE);

Si la descomentamos lo que haremos es que la función se continuará ejecutando únicamente cuando la función a la cual hace referencia el hilo tPrimary termine.

La función WaitForSingleObject permite realizar esta espera, el primer parámetro es el controlador del hilo y el segundo el tiempo en milisegundos que hay que esperar, si queremos que espere indefinidamente como en este caso hay que escribir
INFINITE

Bien, ahora vamos a definir la función main que es la que hará el trabajo:

codeDivStart()int main()

```
{
   DWORD ThreadID;

   tPrimary = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)RunDialogOne, NULL, CREATE

   tSecond = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)RunDialogTwo, NULL, CREATE

   ResumeThread(tPrimary);
   ResumeThread(tSecond);

   HANDLE ThreadContainer[2];
   ThreadContainer[0] = tPrimary;
   ThreadContainer[1] = tSecond;

   DWORD dwSuccess = WaitForMultipleObjects(2, ThreadContainer, TRUE, INFINITE);
```

```
    if (dwSuccess == WAIT_FAILED)
        MessageBox(NULL, "Espera a fallado!", "ERROR", MB_OK | MB_ICONERROR);
    else
        MessageBox(NULL, "Exito!", "EXITO", MB_OK | MB_ICONINFORMATION);

    CloseHandle(tPrimary);
    CloseHandle(tSecond);

    return 0;
}
```

No he puesto comentarios para simplificar el código, pero vamos a explicarlo un poco:

- Declaramos la variable ThreadID de tipo DWORD, la cual contendrá el identificador del hilo, auqnue realmente no la usaremos en el presente ejemplo, es necesario declararla.

- Luego creamos un hilo por cada función mediante el método CreateThread, los parámetros en el orden dado son los siguientes:

NULL	Atributos de seguridad. (NULL -> Default)
0	Tamaño de la pila. (0 -> Default)
(LPTHREAD_START_ROUTINE)RunDialogThread	Puntero a la funcion (TypeCasted).
NULL	Parametro de la funcion (si es necesario).
CREATE_SUSPENDED	No iniciar hilo todavia
&ThreadID	ID unico para el hilo. No usado en el ejemplo.

- Luego iniciamos cada hilo mediante la función ResumeThread

- Ahora, mediante la función WaitForMultipleObjects le decimos al programa que espere a que los 2 hilos especificados en el arreglo
ThreadContainer
terminen y luego si poder continuar con la ejecución, los parámetros en el orden dado son los siguientes:

2	Numero de hilos en el arreglo.
ThreadContainer	El arreglo.
TRUE	TRUE -> Espera a TODOS los hilos.
INFINITE	Esperar siempre.

- Luego de terminar la ejecución cerramos los hilos mediante CloseHandle, hay que hacer esto sólo cuando haya terminado la ejecución, si lo hacemos antes, podemos tener problemas