

PRÁCTICA 7: COMPILACIÓN DE PROGRAMAS - DANIEL PÉREZ RUIZ

EJERCICIO 7.1

```
main.cpp: In function 'int main()':  
main.cpp:7:17: error: 'print_hello' was not declared in this scope  
main.cpp:9:52: error: 'factorial' was not declared in this scope
```

Como se ha comentado la línea que incluye la librería no reconoce esas funciones.

EJERCICIO 7.2

Porque no hemos incluido los objetos .o de las funciones del seno, coseno y tangente ni tampoco su biblioteca. Se ha podido podido crear el main.o porque tan sólo llama a las demás funciones.

EJERCICIO 7.3

Ha fallado porque se han movido las librerías a otra carpeta, y la opción -L./ las está buscando en la carpeta actual, y como ahí no están no encuentra la definición de ninguna función.

EJERCICIO 7.4

La salida de la orden make cambia dependiendo de que archivos hayamos modificado, para no tener que ejecutar todos los archivos si solo han cambiado alguno de ellos.

EJERCICIO 7.5

Habria que `./include/function.h` y `-I./includes`.

EJERCICIO 7.6

```
# Nombre archivo: makefileG  
# Uso: make -f makefileG  
# Descripción: Mantiene todas las dependencias entre los módulos y la biblioteca que  
# utiliza el programa2.  
  
# Indica el compilador que se va a usar  
CC=g++  
# Indica el directorio donde se encuentran los archivos de cabecera  
INCLUDE_DIR= ./includes  
# Indica el directorio donde se encuentran las bibliotecas  
LIB_DIR= ./  
#Indica las opciones que se le van a pasar al compilador  
CPPFLAGS= -Wall
```

```

programa2: main2.o factorial.o hello.o libmates.a
    $(CC) -L$(LIB_DIR) -o $@ main2.o factorial.o hello.o -lmates
main2.o: main2.cpp
    $(CC) -I$(INCLUDE_DIR) -c main2.cpp
factorial.o: factorial.cpp
    $(CC) -I$(INCLUDE_DIR) -c factorial.cpp
hello.o: hello.cpp
    $(CC) -I$(INCLUDE_DIR) -c hello.cpp
libmates.a: sin.o cos.o tan.o
    ar -rvs libmates.a sin.o cos.o tan.o
sin.o: sin.cpp
    $(CC) -I$(INCLUDE_DIR) -c sin.cpp
cos.o: cos.cpp
    $(CC) -I$(INCLUDE_DIR) -c cos.cpp
tan.o: tan.cpp
    $(CC) -I$(INCLUDE_DIR) -c tan.cpp
clean:
    rm *.o programa2

```

EJERCICIO 7.7

```

# Nombre archivo: makefileG
# Uso: make -f makefileG
# Descripción: Mantiene todas las dependencias entre los módulos y la biblioteca que
# utiliza el programa2.

CC=g++
INCLUDE_DIR= ./includes
LIB_DIR= ./
CPPFLAGS= -Wall

programa2: main2.o factorial.o hello.o ./libs/libmates.a
    $(CC) -L$(LIB_DIR) -o $@ main2.o factorial.o hello.o -lmates
main2.o: main2.cpp
    $(CC) -I$(INCLUDE_DIR) -c main2.cpp
factorial.o: factorial.cpp
    $(CC) -I$(INCLUDE_DIR) -c factorial.cpp
hello.o: hello.cpp
    $(CC) -I$(INCLUDE_DIR) -c hello.cpp
libmates.a: sin.o cos.o tan.o
    ar -rvs libmates.a sin.o cos.o tan.o
sin.o: sin.cpp
    $(CC) -I$(INCLUDE_DIR) -c sin.cpp
cos.o: cos.cpp
    $(CC) -I$(INCLUDE_DIR) -c cos.cpp
tan.o: tan.cpp
    $(CC) -I$(INCLUDE_DIR) -c tan.cpp
clean:
    rm *.o programa2

```

EJERCICIO 7.8

```
# Nombre archivo: makefileG
# Uso: make -f makefileG
# Descripción: Mantiene todas las dependencias entre los módulos y la biblioteca que
# utiliza el programa2.

CC=g++
INCLUDE_DIR= ./includes
LIB_DIR= ./

programa2: main2.o factorial.o hello.o libmates.a
$(CC) -L$(LIB_DIR) -o programa2 main2.o factorial.o hello.o -lmates
main2.o: main2.cpp
    $(CC) -I$(INCLUDE_DIR) -c main2.cpp
factorial.o: factorial.cpp
$(CC) -I$(INCLUDE_DIR) -c factorial.cpp
hello.o: hello.cpp
    $(CC) -I$(INCLUDE_DIR) -c hello.cpp
libmates.a: sin.o cos.o tan.o
    $(AR) -rvs libmates.a sin.o cos.o tan.o
sin.o: sin.cpp
    $(CC) -I$(INCLUDE_DIR) -c sin.cpp
cos.o: cos.cpp
    $(CC) -I$(INCLUDE_DIR) -c cos.cpp
tan.o: tan.cpp
    $(CC) -I$(INCLUDE_DIR) -c tan.cpp
```

EJERCICIO 7.9

```
# Nombre archivo: makefileH
# Uso: make -f makefileH
# Descripción: Mantiene todas las dependencias entre los módulos que utiliza el
# programa1.

# CC indica que vamos a compilar con c++
CC=g++
# CPPFLAGS nos indica todos los warning que se pueden dar
CPPFLAGS=-Wall -I./includes
# SOURCEMODULES nos indica todos los archivos cpp
SOURCE_MODULES=main.cpp factorial.cpp hello.cpp
# OBJECTMODULES nos indica los archivos .o
OBJECT_MODULES=$(SOURCE_MODULES:.cpp=.o)
# EXECUTABLE nos indica el nombre del archivo ejecutable que se generará
EXECUTABLE=programa1
# all
all: $(OBJECT_MODULES) $(EXECUTABLE)
$(EXECUTABLE): $(OBJECT_MODULES)
    $(CC) $^ -o $@

# Regla para obtener los archivos objeto .o que dependerán de los archivos .cpp
```

Aquí, '\$<' y '\$@' tomarán los valores 'main.cpp' y 'main.o' respectivamente, y así sucesivamente

.o: .cpp

`$(CC) $(CPPFLAGS) $< -o $@`