Manual DEV C++

Mi primer proyecto

Para realizar esta actividad deberás disponer de un ordenador en el que esté instalado el Dev-C++. Debes ir realizando cada uno de los pasos indicados,

Objetivo

Al finalizar esta práctica serás capaz de:

- Crear un proyecto en Dev-C++.
- Dividir tu código en archivos .h y .cpp.
- Aplicar principios básicos de programación orientada a objetos (POO).
- Compilar y ejecutar tu proyecto paso a paso.

Requisitos

- Tener instalado **Dev-C++** .
- Conocimientos básicos de C++ y POO.

Puedes descargar Dev-C++ desde: https://sourceforge.net/projects/orwelldevcpp/

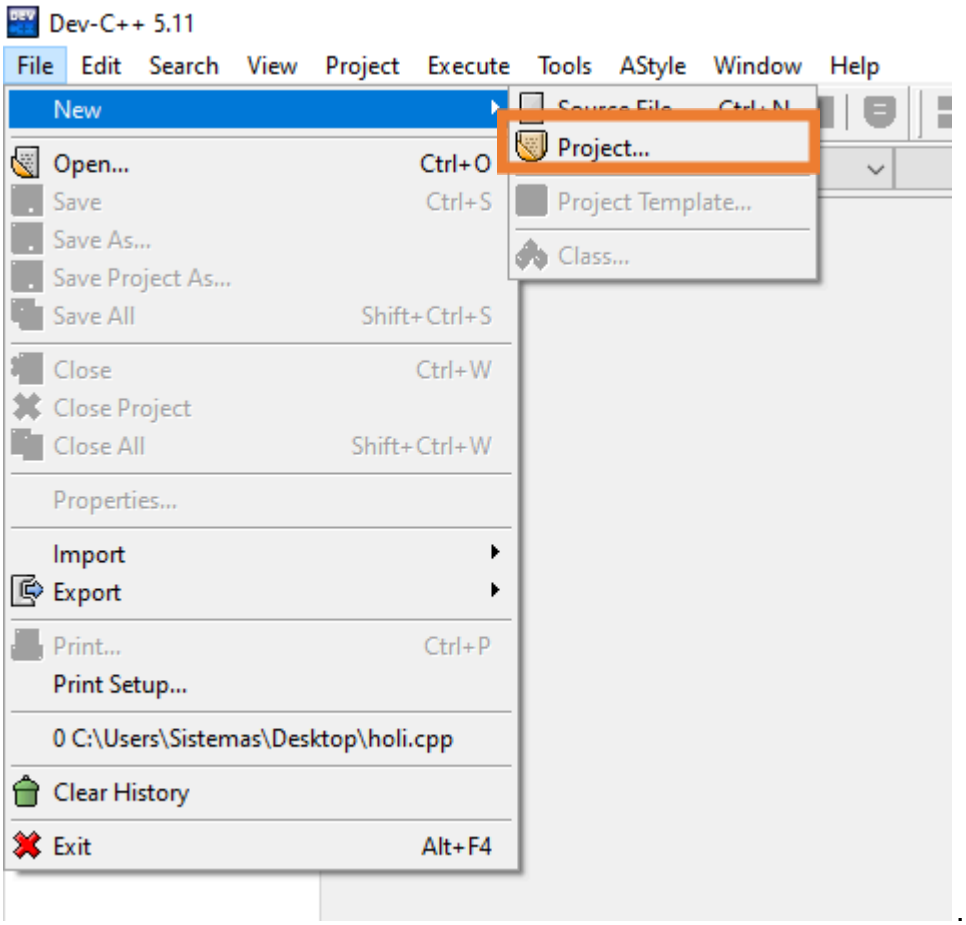
Pasos para crear el proyecto

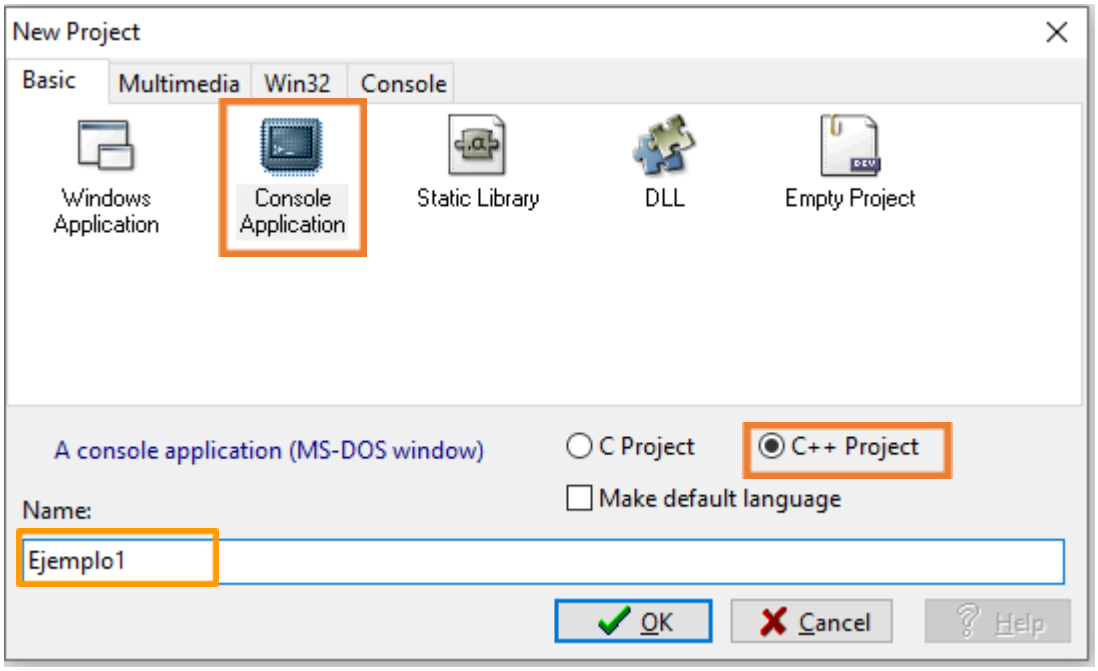
Paso 1: Abrir Dev-C++

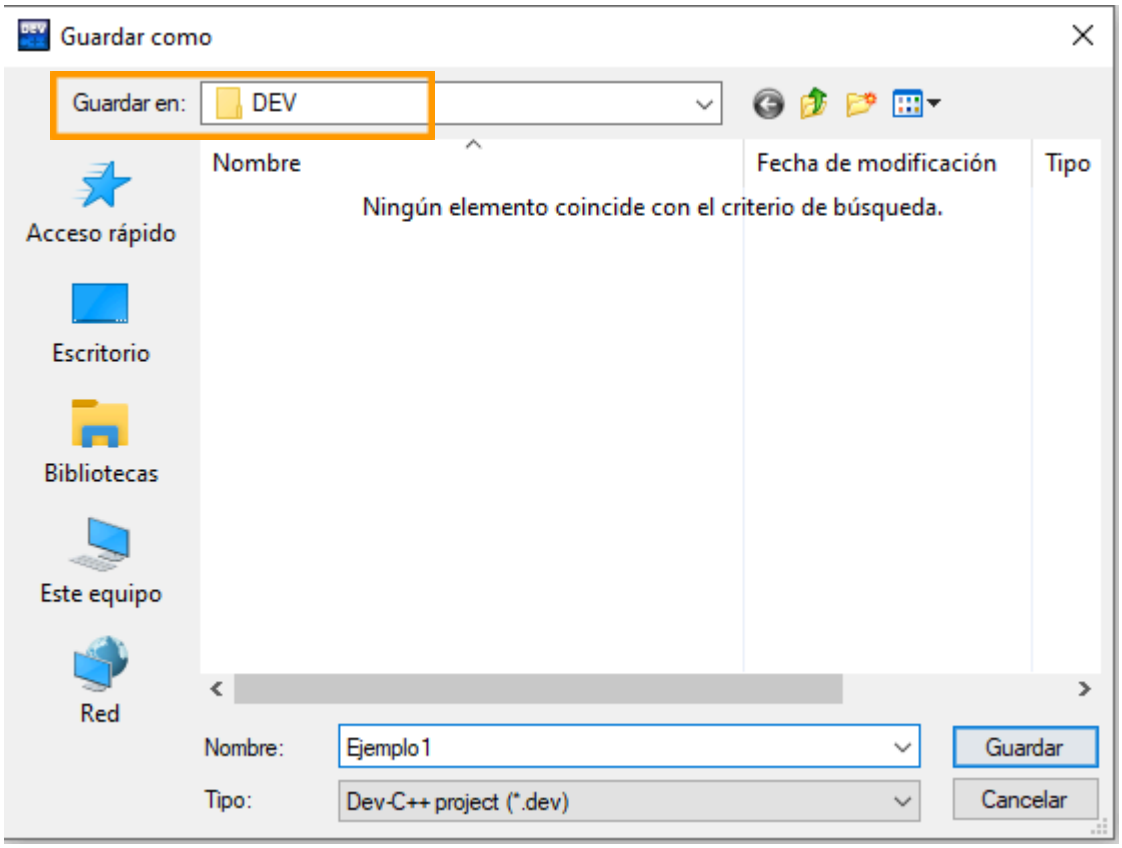
- Abre Dev-C++ desde el menú de inicio.

Paso 2: Crear un nuevo proyecto

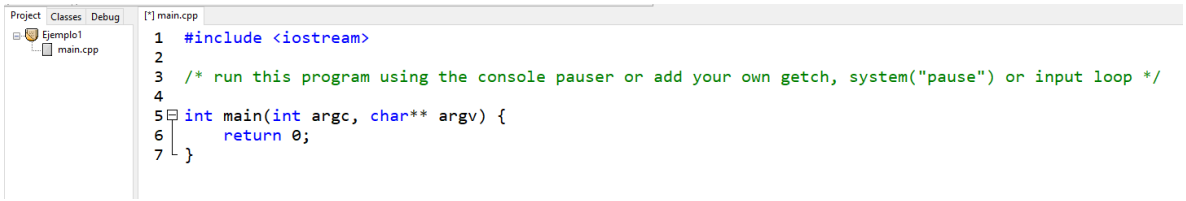
1. Haz clic en **Archivo** → **Nuevo** → **Proyecto**.
2. Selecciona **"Proyecto de Consola"**.
3. Elige el lenguaje **C++**.
4. Escribe el nombre del proyecto: Ejemplo1.
5. Elige una carpeta donde guardar tu proyecto y haz clic en **Aceptar**.


Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
New
Open... Ctrl+O
Save Ctrl+S
Save As...
Save Project As...
Save All Shift+Ctrl+S
Close Ctrl+W
Close Project
Close All Shift+Ctrl+W
Properties...
Import
Export
Print... Ctrl+P
Print Setup...
0 C:\Users\Sistemas\Desktop\holi.cpp
Clear History
Exit Alt+F4
Project...
Project Template...
Class...


New Project
Basic
Multimedia
Win32
Console
Windows Application
Console Application
Static Library
DLL
Empty Project
A console application (MS-DOS window)
C Project
C++ Project
Make default language
Name:
Ejemplo1
OK
Cancel
Help



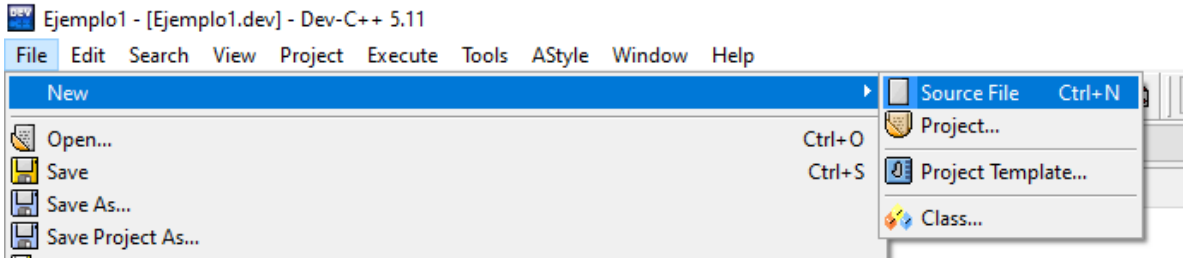
Se crea el siguiente archivo main.cpp por defecto:



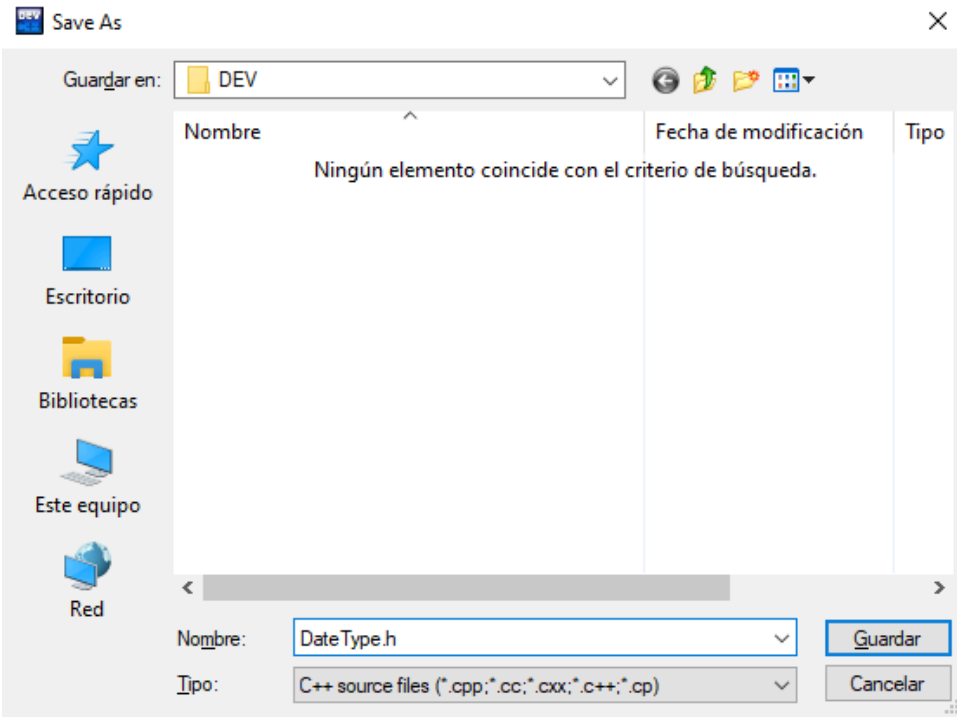
Paso 3: Crear los archivos .h y .cpp

A) Crear archivo DateType.h

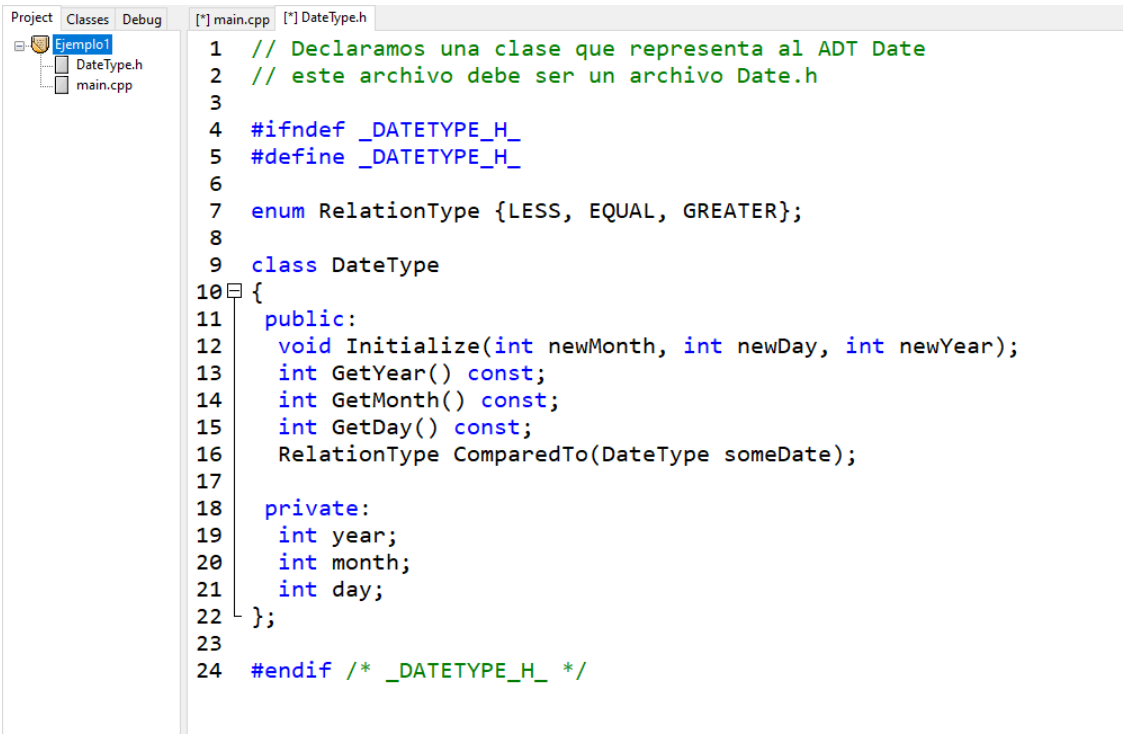
1. Ve a **Archivo → Nuevo → Archivo fuente**.



2. Escribe el siguiente código:



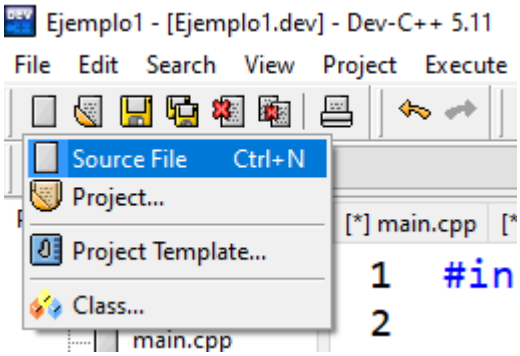
Agregamos la declaración de las variables y funciones a utilizar en donde se importe el header:



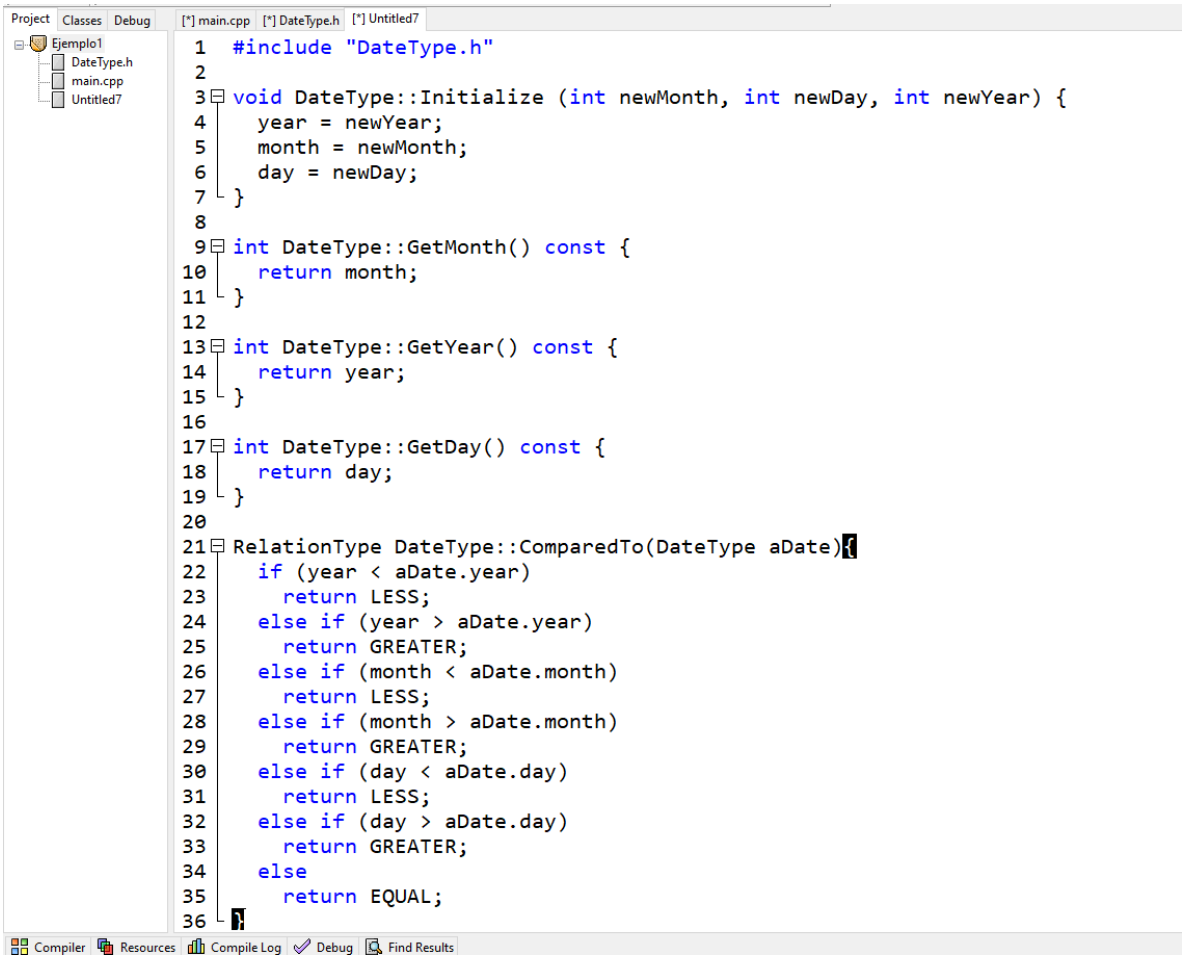

```
1  // Declaramos una clase que representa al ADT Date
2  // este archivo debe ser un archivo Date.h
3
4  #ifndef _DATETYPE_H_
5  #define _DATETYPE_H_
6
7  enum RelationType {LESS, EQUAL, GREATER};
8
9  class DateType
10 {
11  public:
12   void Initialize(int newMonth, int newDay, int newYear);
13   int GetYear() const;
14   int GetMonth() const;
15   int GetDay() const;
16   RelationType ComparedTo(DateType someDate);
17
18  private:
19   int year;
20   int month;
21   int day;
22 };
23
24 #endif /* _DATETYPE_H_ */
```

B) Crear archivo DataType.cpp

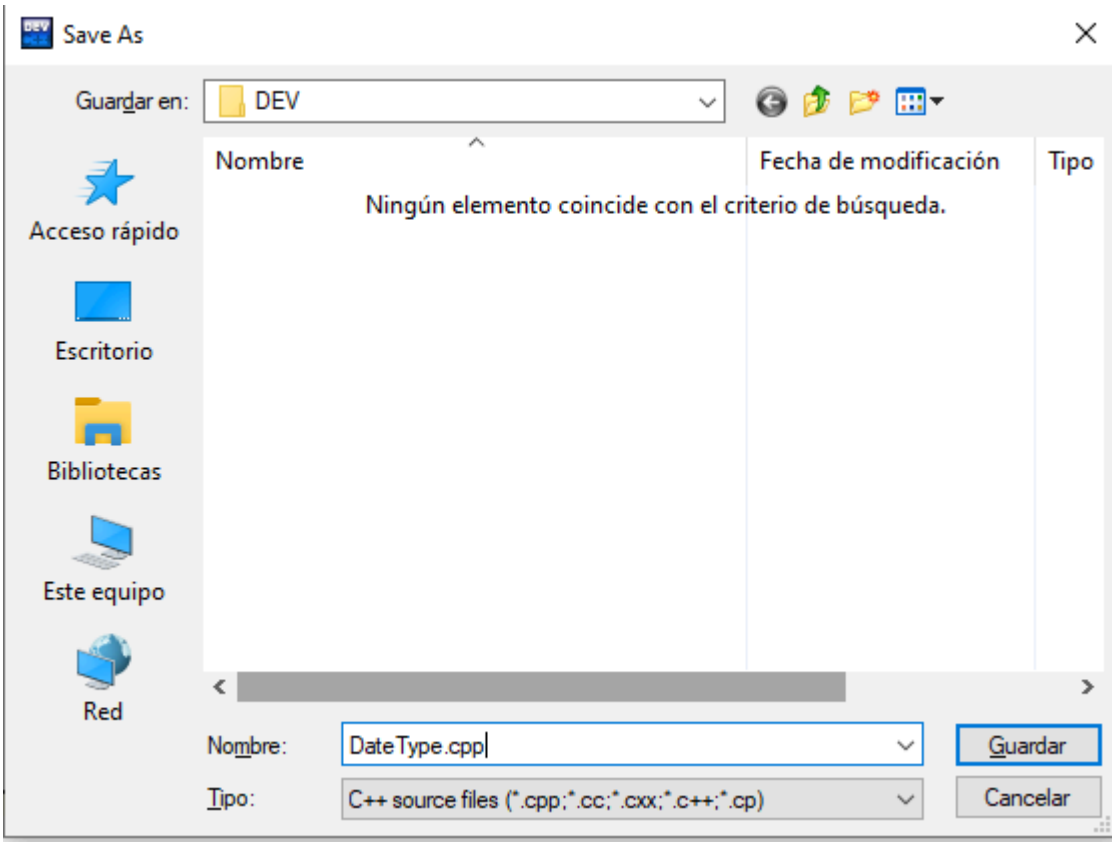
1. Ve al icono de **File** → **Archivo fuente**.



2. Escribe el siguiente código:

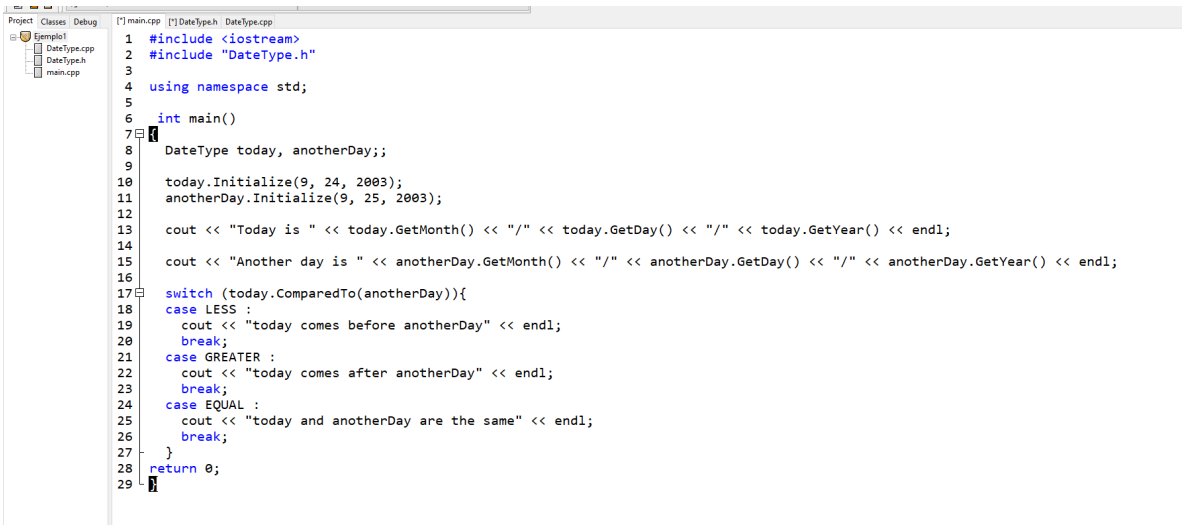


Lo guardamos como DateType.cpp



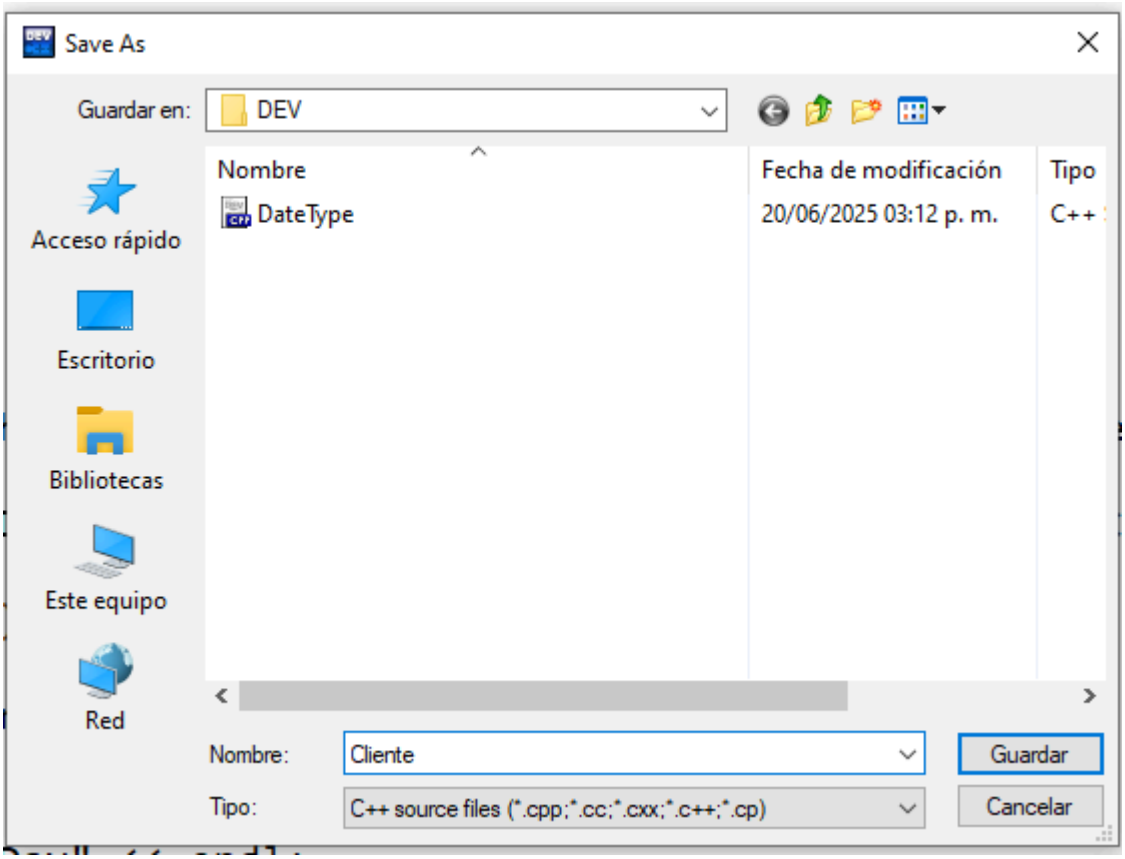
Paso 4: Editar el archivo main.cpp

En Dev-C++, ya viene un archivo principal. Si no aparece, crea uno nuevo llamado main.cpp con el siguiente contenido:



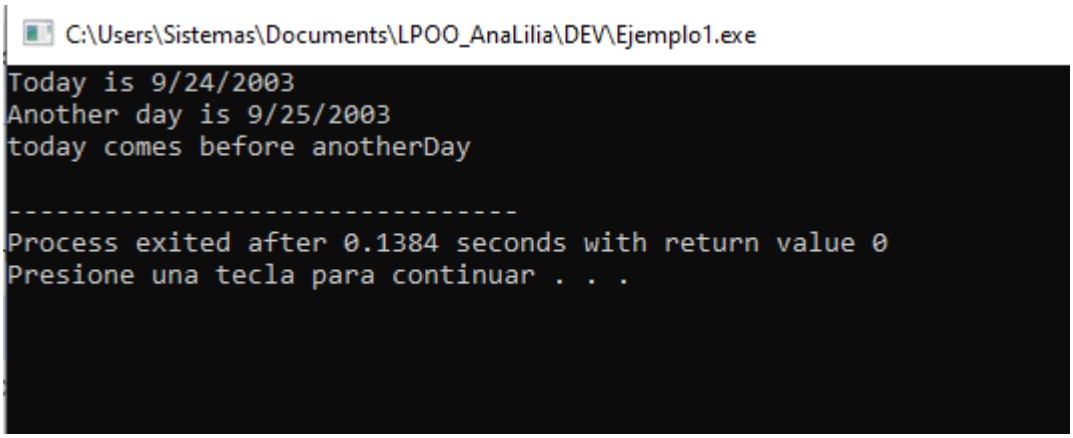
```cpp
#include <iostream>
#include "DateType.h"

using namespace std;

 int main()
{
  DateType today, anotherDay;;

  today.Initialize(9, 24, 2003);
  anotherDay.Initialize(9, 25, 2003);

  cout << "Today is " << today.GetMonth() << "/" << today.GetDay() << "/" << today.GetYear() << endl;

  cout << "Another day is " << anotherDay.GetMonth() << "/" << anotherDay.GetDay() << "/" << anotherDay.GetYear() << endl;

  switch (today.ComparedTo(anotherDay)){
  case LESS :
    cout << "today comes before anotherDay" << endl;
    break;
  case GREATER :
    cout << "today comes after anotherDay" << endl;
    break;
  case EQUAL :
    cout << "today and anotherDay are the same" << endl;
    break;
  }
return 0;
}
```

Lo guardamos como Cliente:



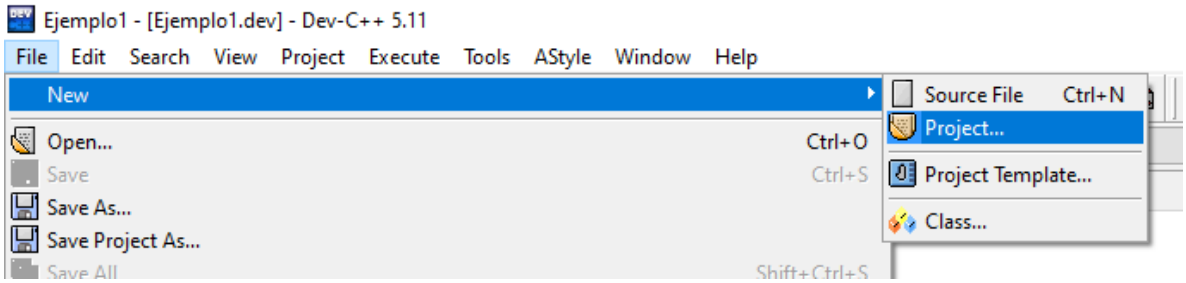
Paso 6: Compilar y ejecutar

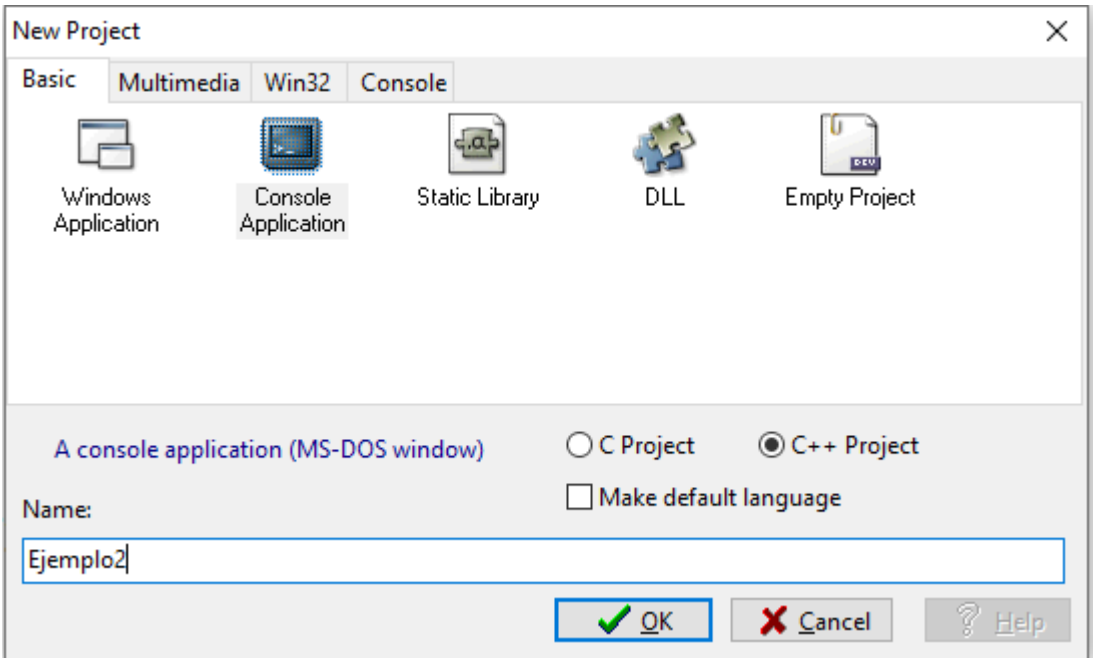
- Haz clic en el botón **"Compilar y Ejecutar"** (o presiona F11).
- Deberías ver la consola con la siguiente salida:

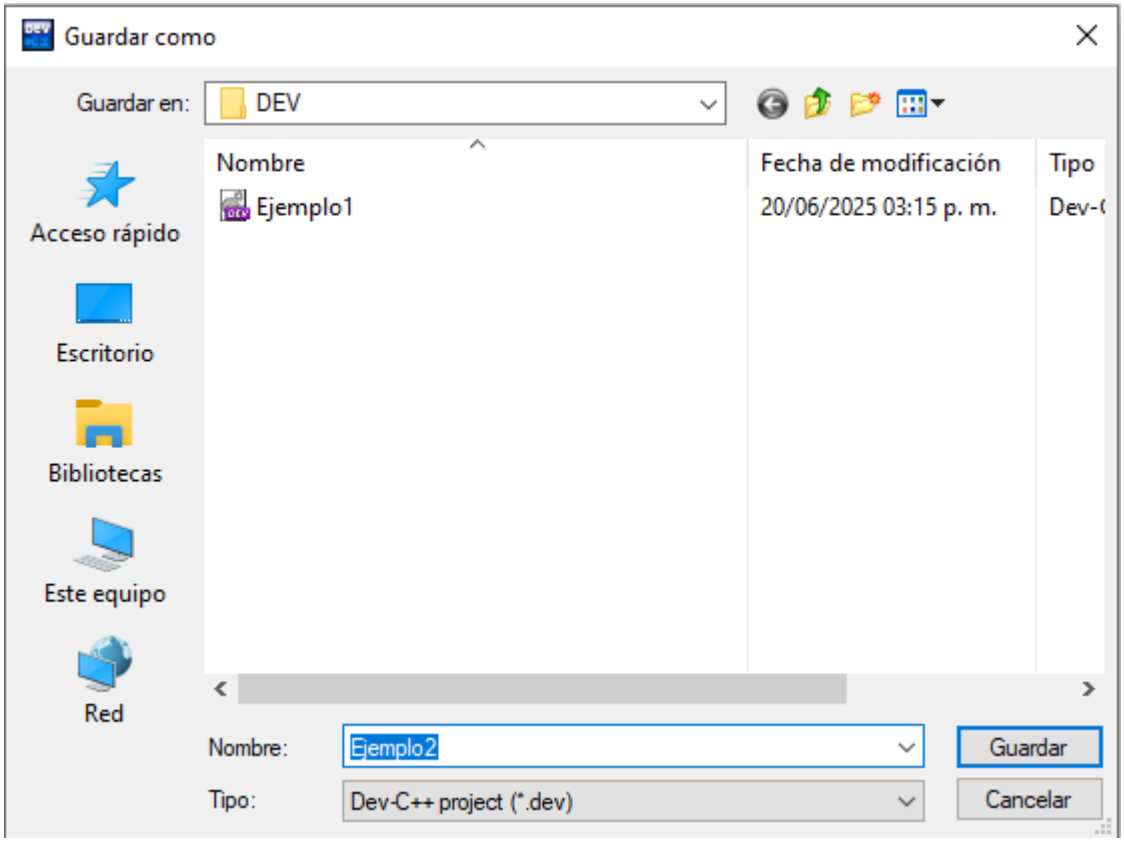


Ejemplo 2:

1. Crear el proyecto:
    - Debe ser console application
    - Indicar el nombre del proyecto y carpeta



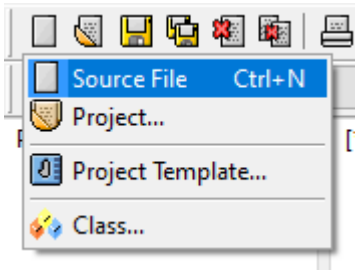


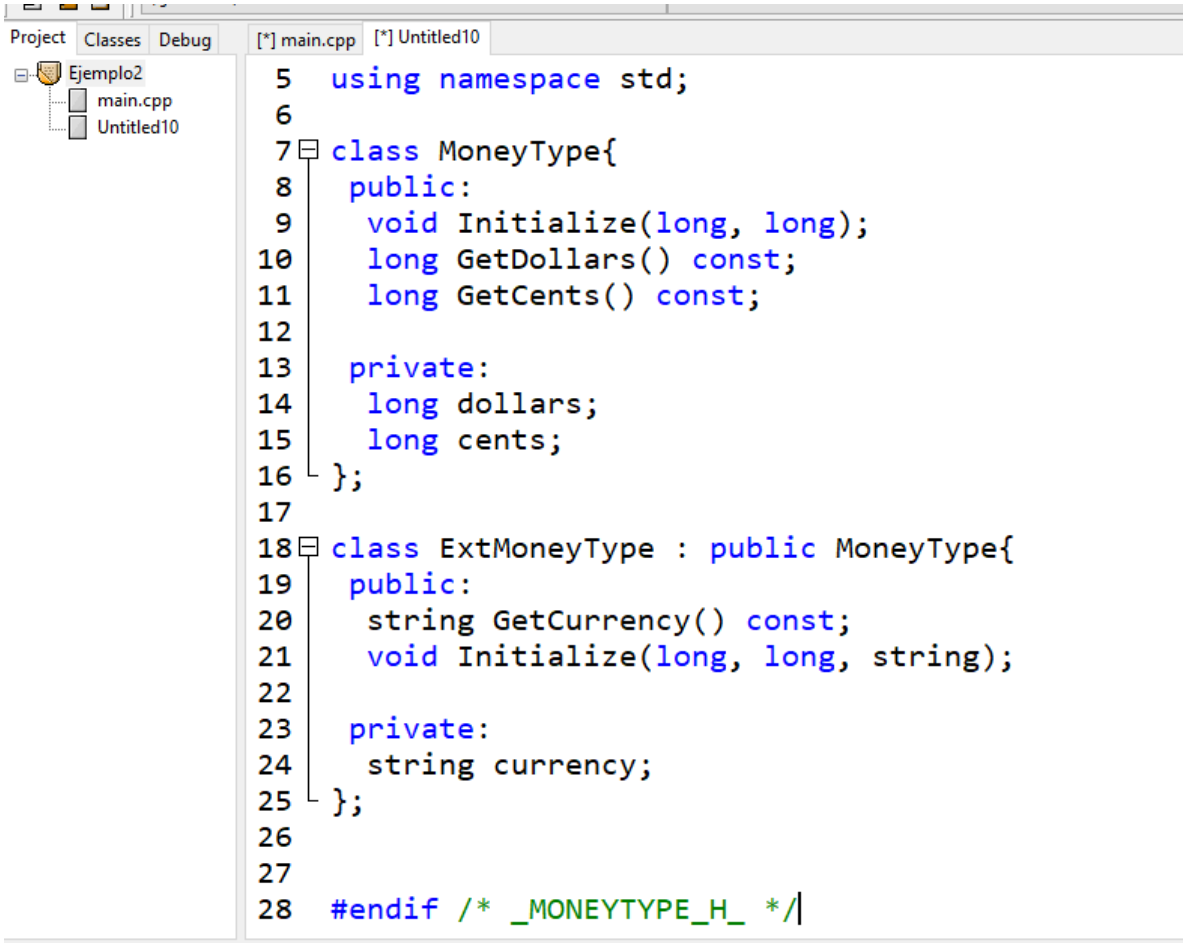


Agregamos el siguiente código en el main que se crea por defecto:

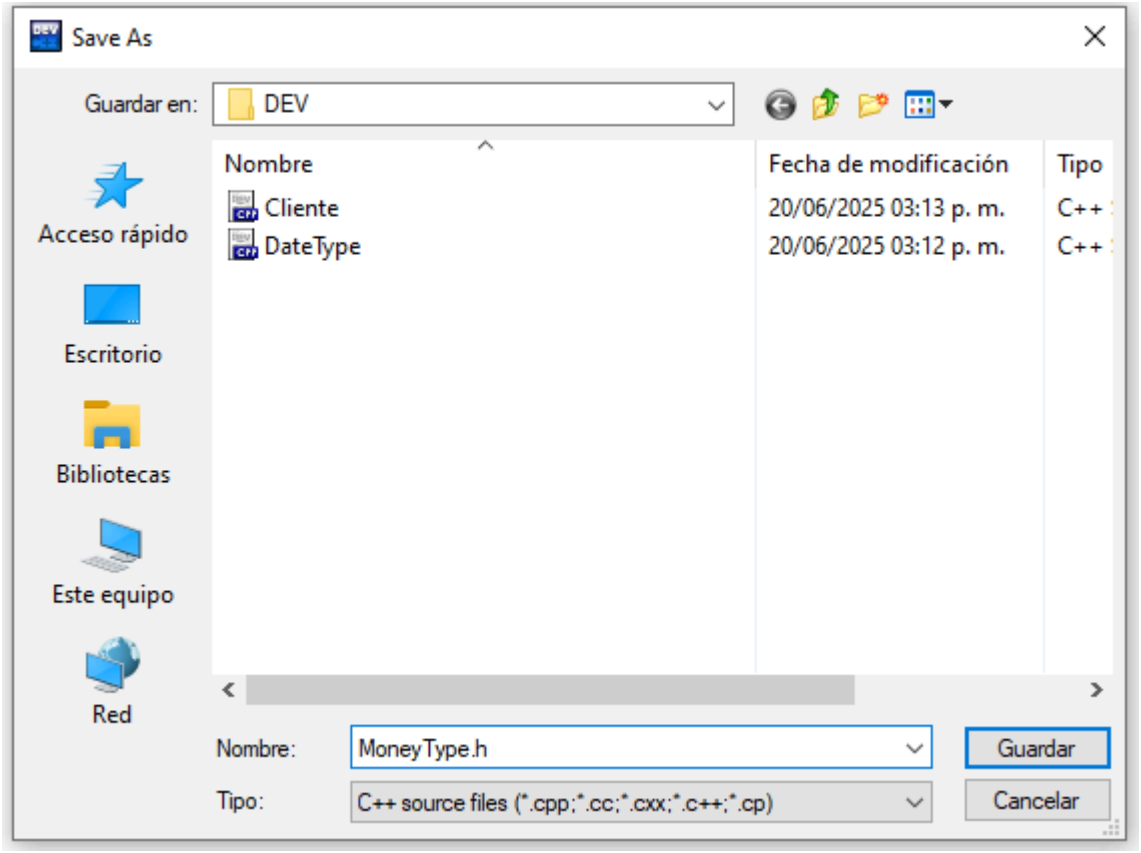
```cpp
#include <iostream>
#include "MoneyType.h"

using namespace std;

main(){
  MoneyType *money = new MoneyType();
  ExtMoneyType *extMoney = new ExtMoneyType();

  money->Initialize(60,20);
  extMoney->Initialize(60,20,"francs");

  cout << "Money: " << money->GetDollars() << " Dollars and " << money->GetCents() << " Cents." << endl;
  cout << "extMoney( "<< extMoney->GetCurrency() <<" ): " << extMoney->GetDollars() << " Dollars and " << extMoney->GetCents() << " Cents"<<

  delete money;
  delete extMoney;
  getchar();
}
```

2. Creamos el header MoneyType.h en donde tendremos la declaración de nuestras variables y funciones a usar.

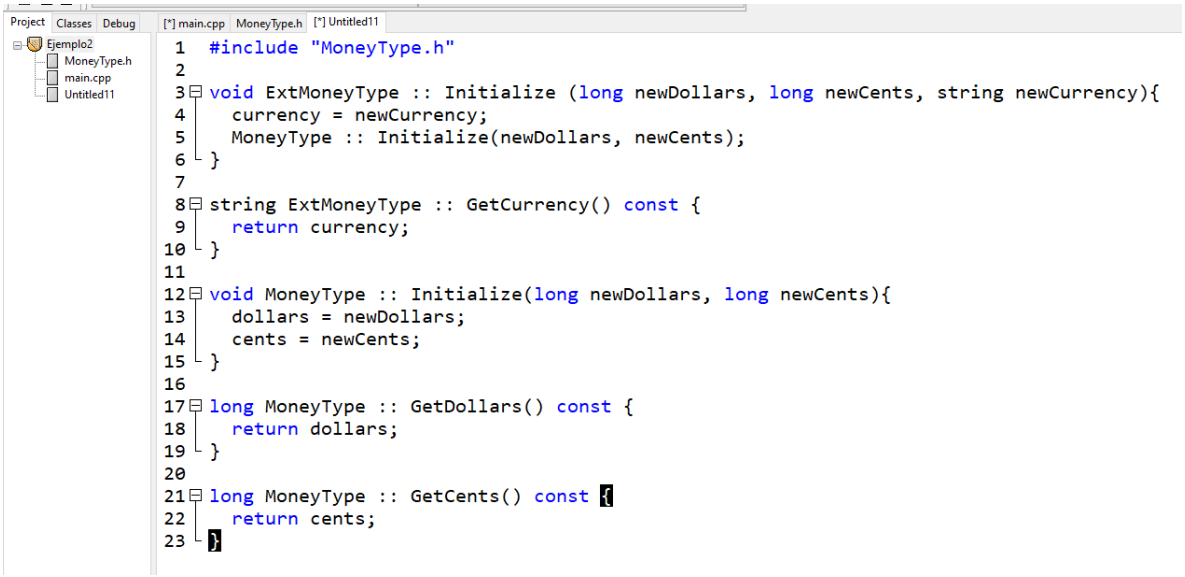


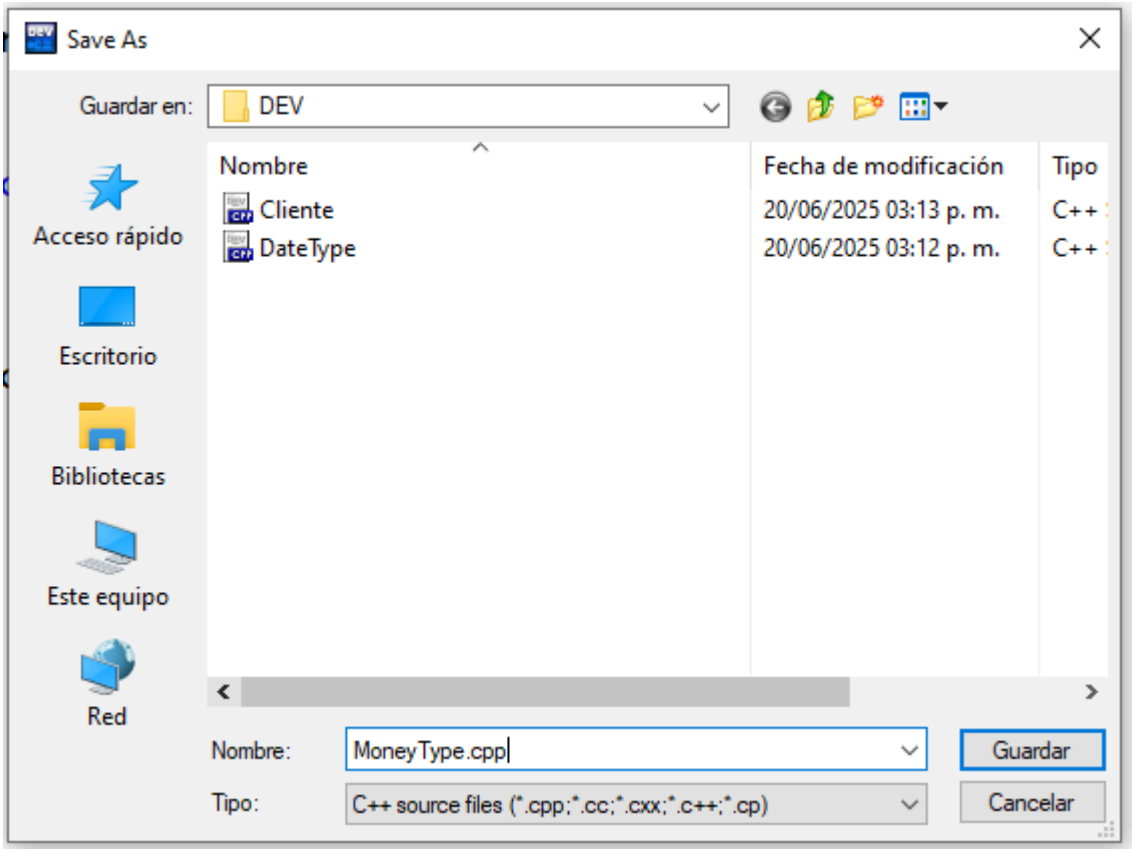



```cpp
using namespace std;

class MoneyType{
 public:
  void Initialize(long, long);
  long GetDollars() const;
  long GetCents() const;

 private:
  long dollars;
  long cents;
};

class ExtMoneyType : public MoneyType{
 public:
  string GetCurrency() const;
  void Initialize(long, long, string);

 private:
  string currency;
};


#endif /* _MONEYTYPE_H_ */
```

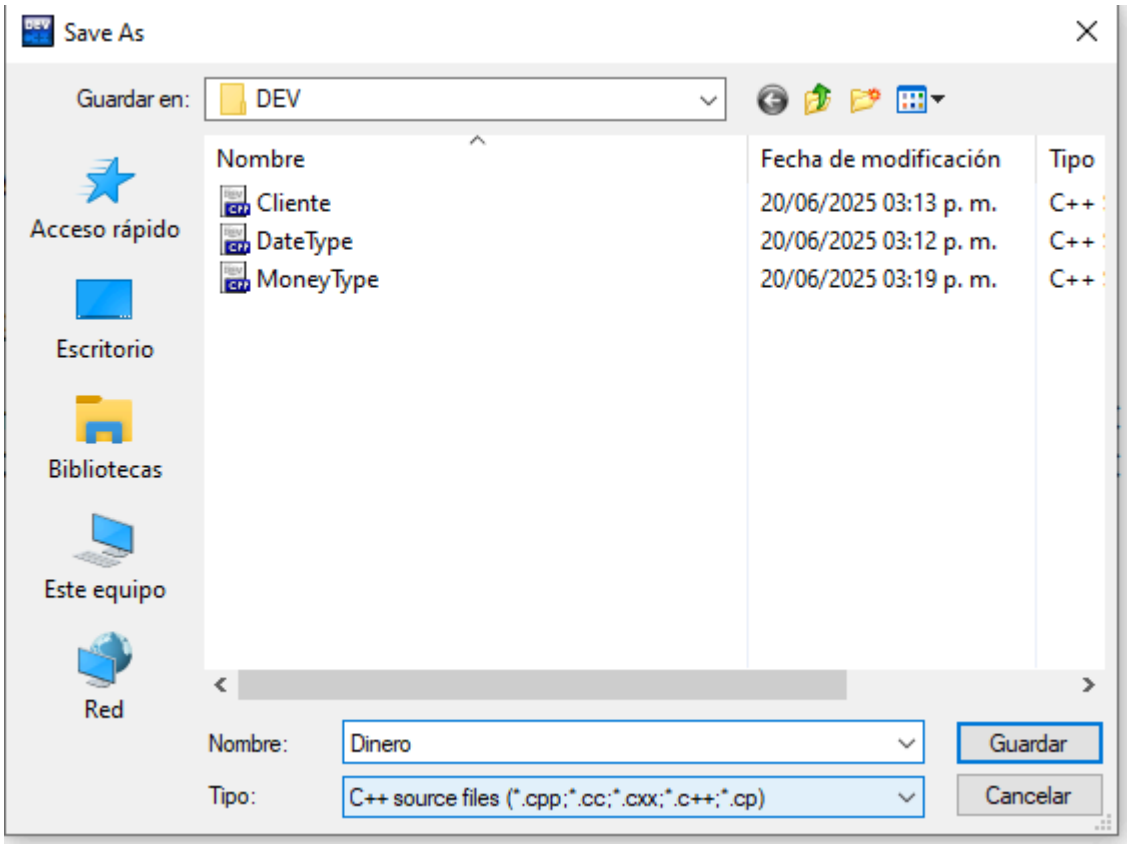


Creamos el archivo MoneyType.cpp en donde estará la implementación de las funciones que se declaran en el header.





Guardamos el archivo main.cpp como Dinero.cpp



Ejecutamos y compilamos: