

# 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/>

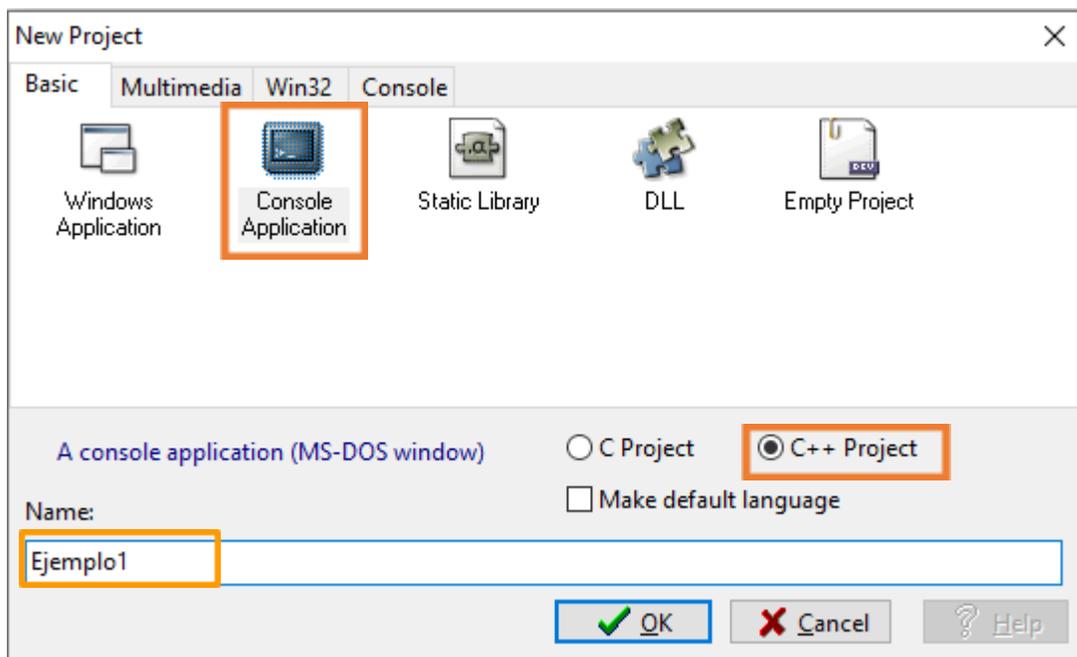
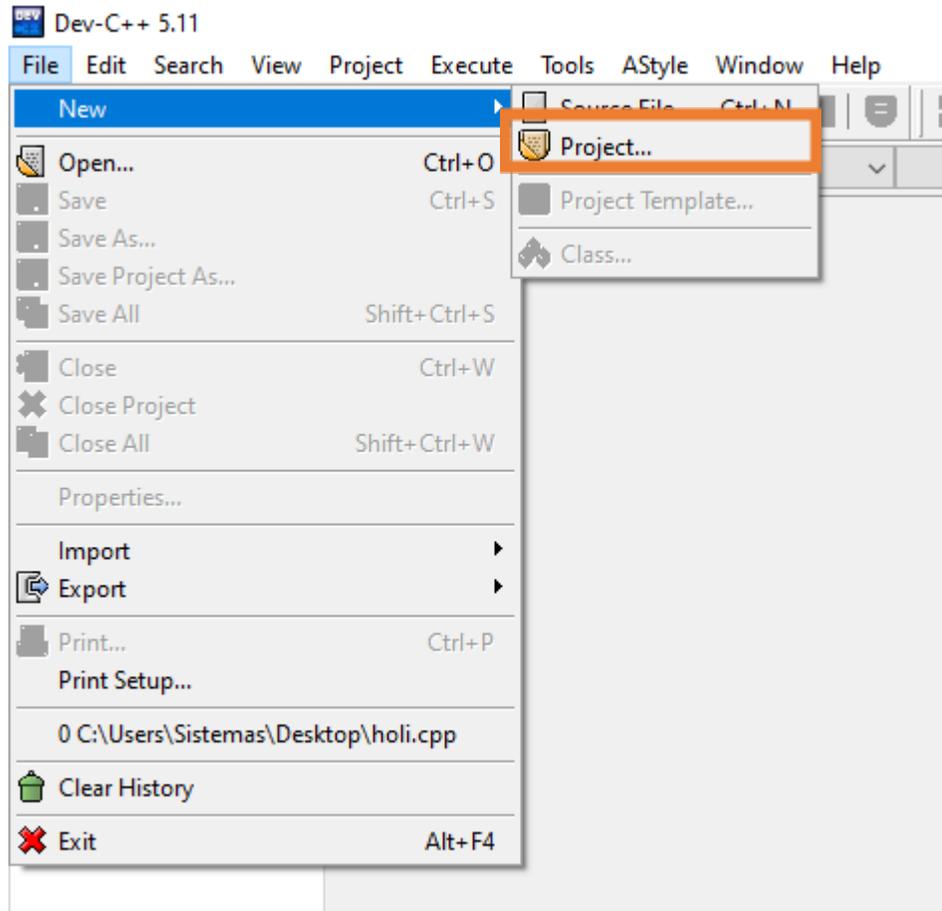
## Pasos para crear el proyecto

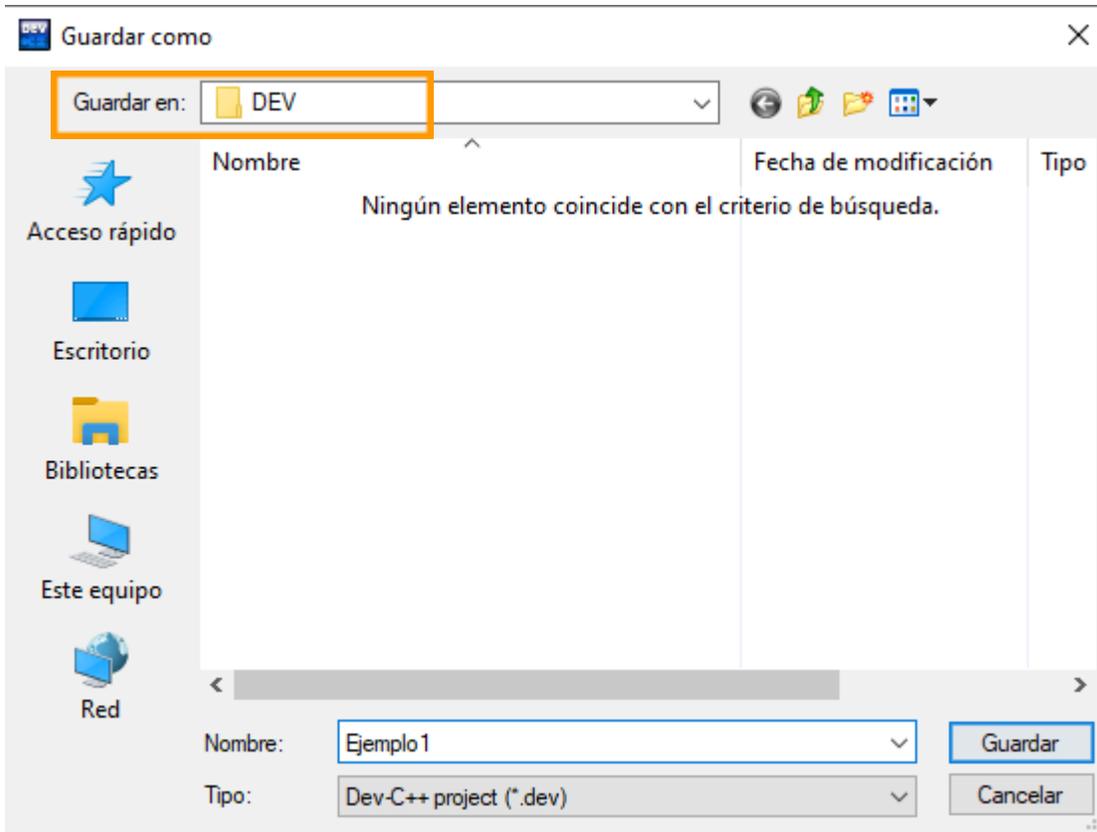
### Paso 1: Abrir Dev-C++

- 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**.





Se crea el siguiente archivo main.cpp por defecto:

```

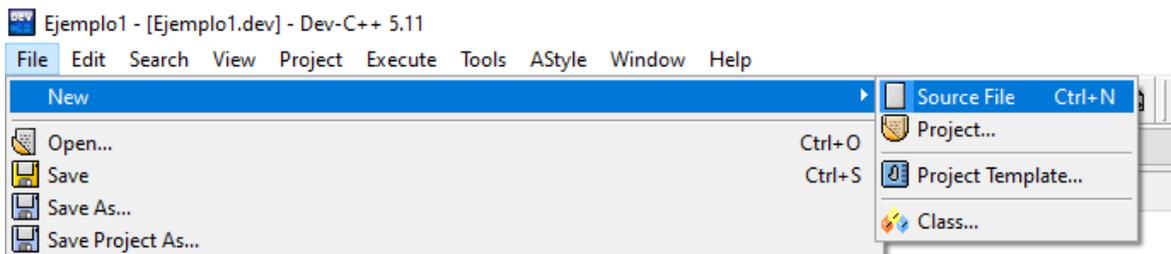
Project Classes Debug
Ejemplo1
└─ main.cpp
1 #include <iostream>
2
3 /* run this program using the console pauser or add your own getch, system("pause") or input loop */
4
5 int main(int argc, char** argv) {
6     return 0;
7 }

```

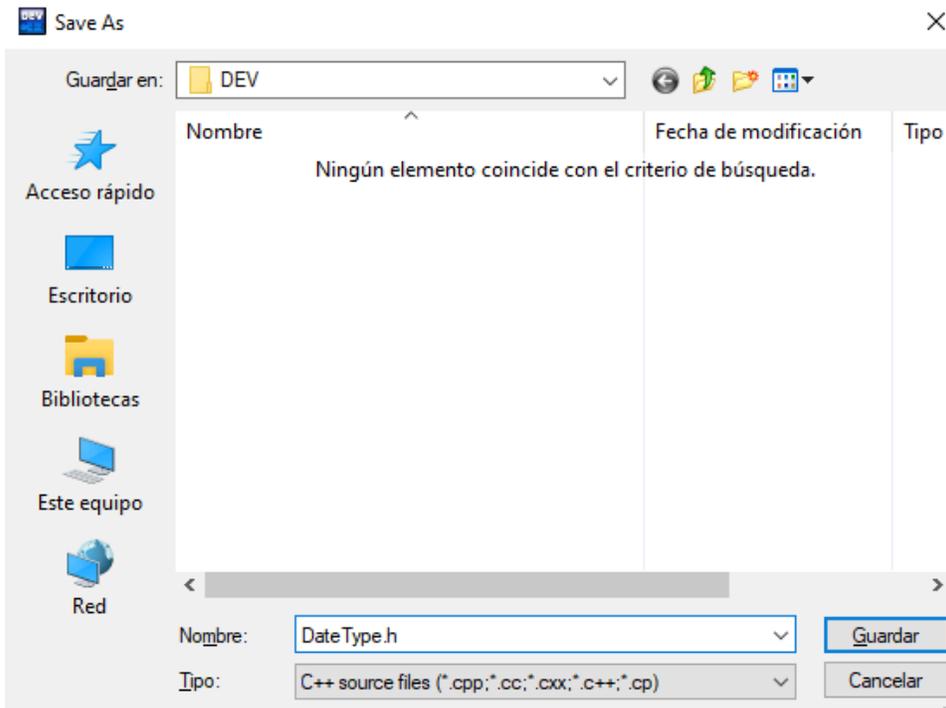
Paso 3: Crear los archivos .h y .cpp

A) Crear archivo *DateType.h*

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



2. Escribe el siguiente código:

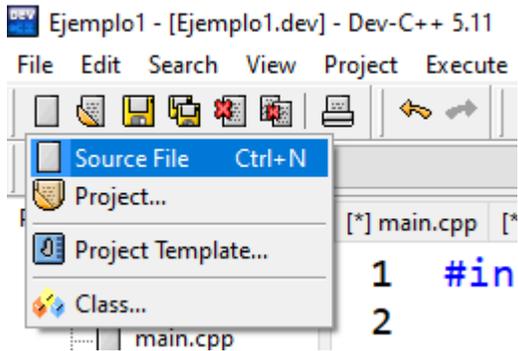


Agregamos la declaración de las variables y funciones a utilizar en donde se importe el header:

```
Project Classes Debug [*] main.cpp [*] DateType.h
Ejemplo
  DateType.h
  main.cpp
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 `DateType.cpp`

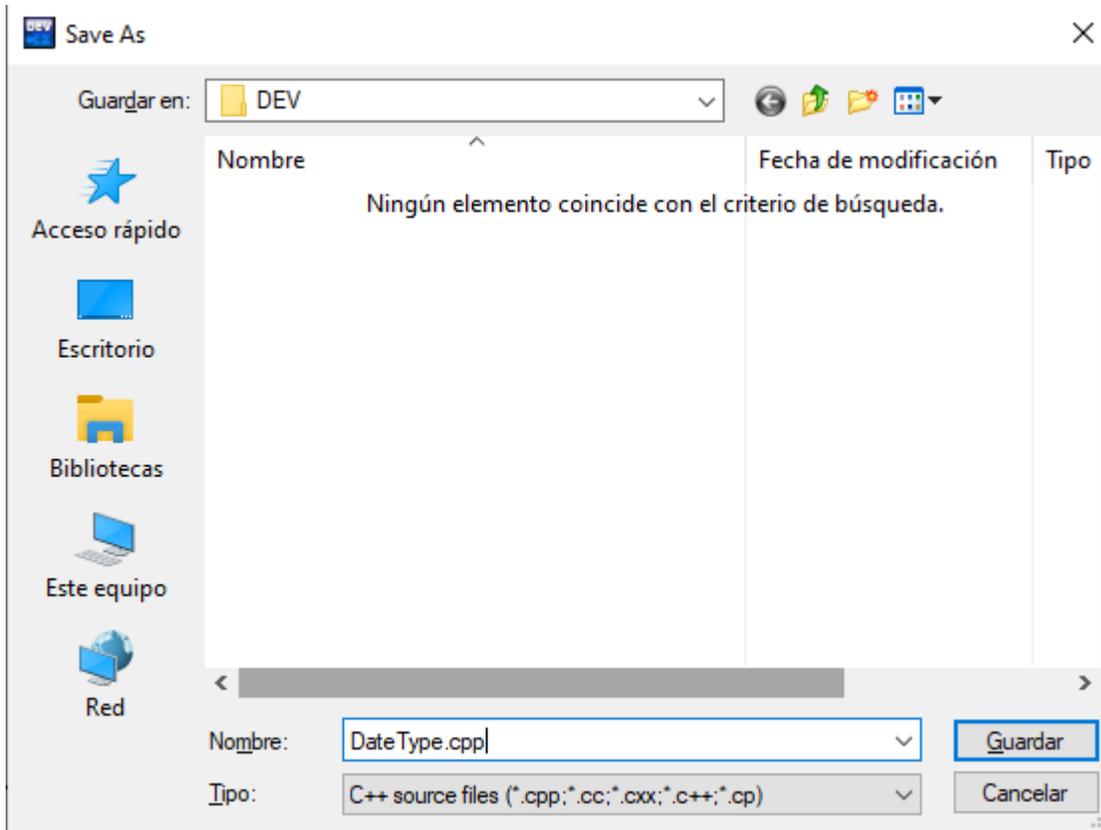
1. Ve al icono de **File** → **Archivo fuente**.



2. Escribe el siguiente código:

```
Project Classes Debug [*] main.cpp [*] DateType.h [*] Untitled7
├── Ejemplo1
│   ├── DateType.h
│   ├── main.cpp
│   └── Untitled7
└──
1 #include "DateType.h"
2
3 void DateType::Initialize (int newMonth, int newDay, int newYear) {
4     year = newYear;
5     month = newMonth;
6     day = newDay;
7 }
8
9 int DateType::GetMonth() const {
10     return month;
11 }
12
13 int DateType::GetYear() const {
14     return year;
15 }
16
17 int DateType::GetDay() const {
18     return day;
19 }
20
21 RelationType DateType::ComparedTo(DateType aDate){
22     if (year < aDate.year)
23         return LESS;
24     else if (year > aDate.year)
25         return GREATER;
26     else if (month < aDate.month)
27         return LESS;
28     else if (month > aDate.month)
29         return GREATER;
30     else if (day < aDate.day)
31         return LESS;
32     else if (day > aDate.day)
33         return GREATER;
34     else
35         return EQUAL;
36 }
```

Lo guardamos como `DateType.cpp`



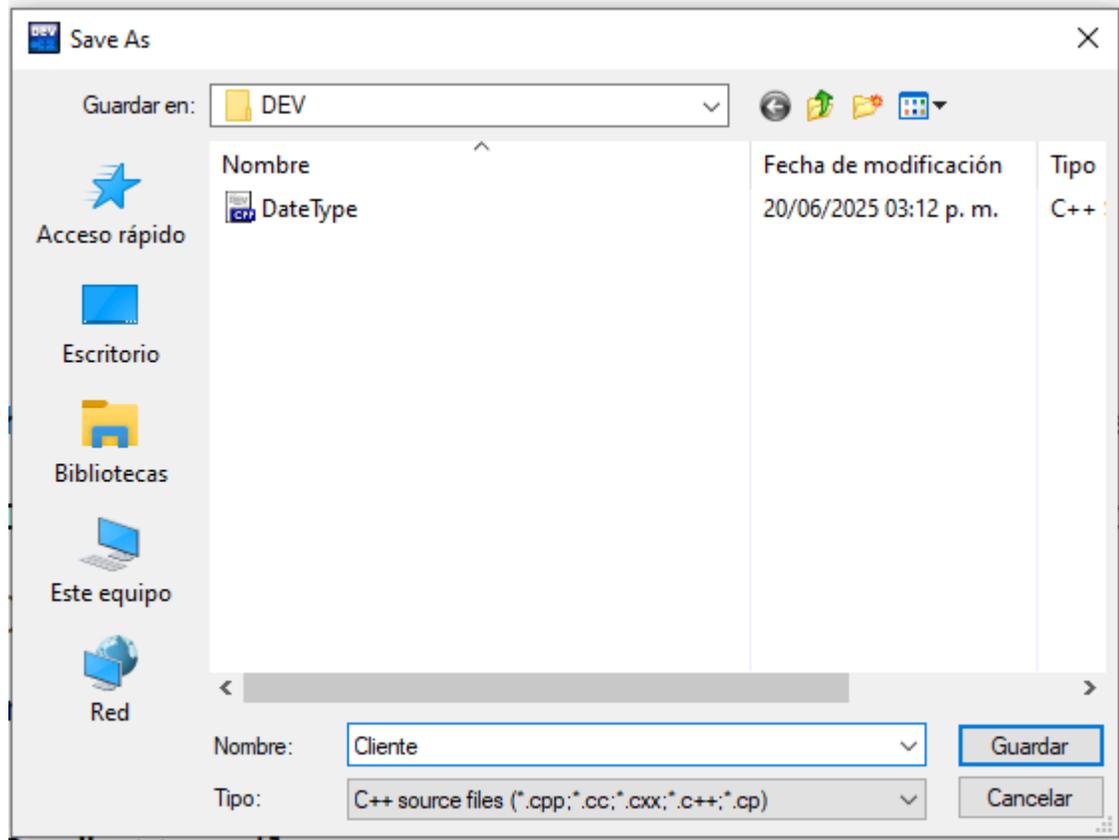
#### Paso 4: Editar el archivo main.cpp

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

```
Project Classes Debug
├── Ejemplo1
│   ├── DateType.cpp
│   ├── DateType.h
│   └── main.cpp
└── main.cpp

1 #include <iostream>
2 #include "DateType.h"
3
4 using namespace std;
5
6 int main()
7 {
8     DateType today, anotherDay;
9
10    today.Initialize(9, 24, 2003);
11    anotherDay.Initialize(9, 25, 2003);
12
13    cout << "Today is " << today.GetMonth() << "/" << today.GetDay() << "/" << today.GetYear() << endl;
14
15    cout << "Another day is " << anotherDay.GetMonth() << "/" << anotherDay.GetDay() << "/" << anotherDay.GetYear() << endl;
16
17    switch (today.ComparedTo(anotherDay)){
18    case LESS :
19        cout << "today comes before anotherDay" << endl;
20        break;
21    case GREATER :
22        cout << "today comes after anotherDay" << endl;
23        break;
24    case EQUAL :
25        cout << "today and anotherDay are the same" << endl;
26        break;
27    }
28    return 0;
29 }
```

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:

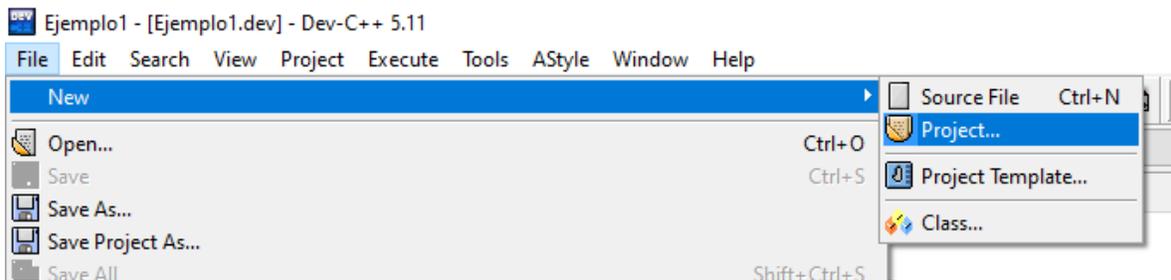


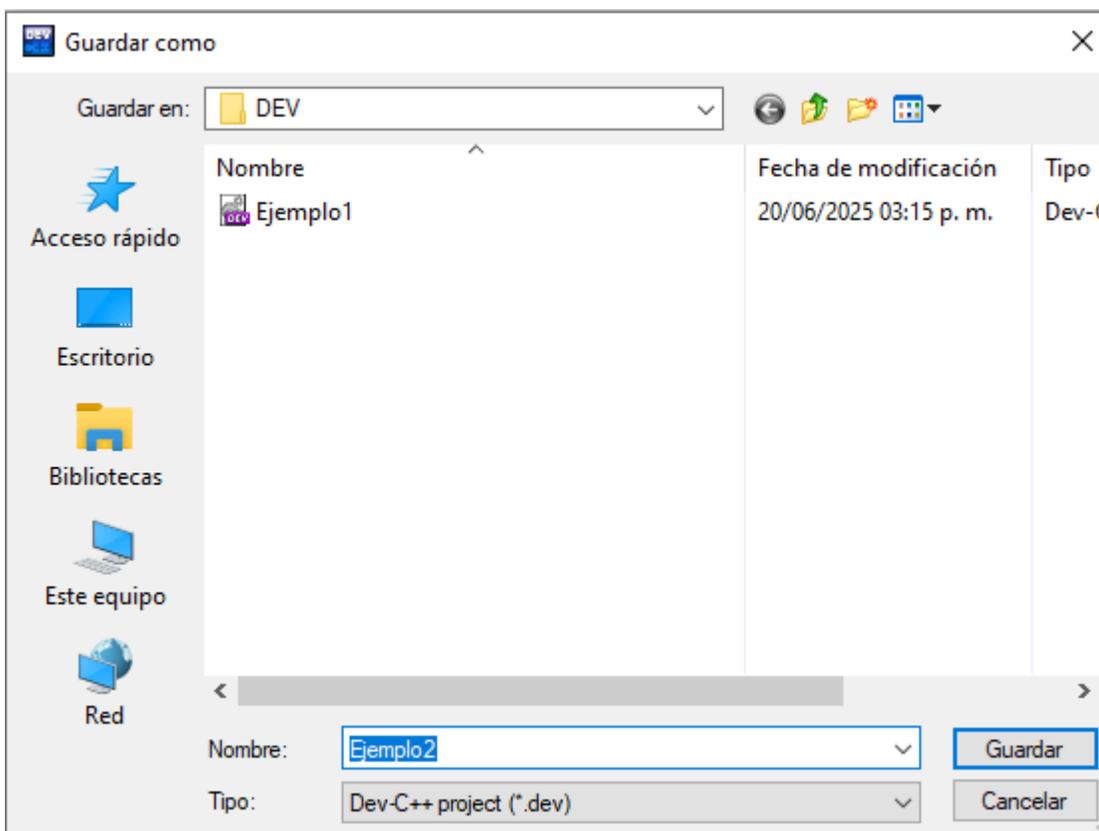
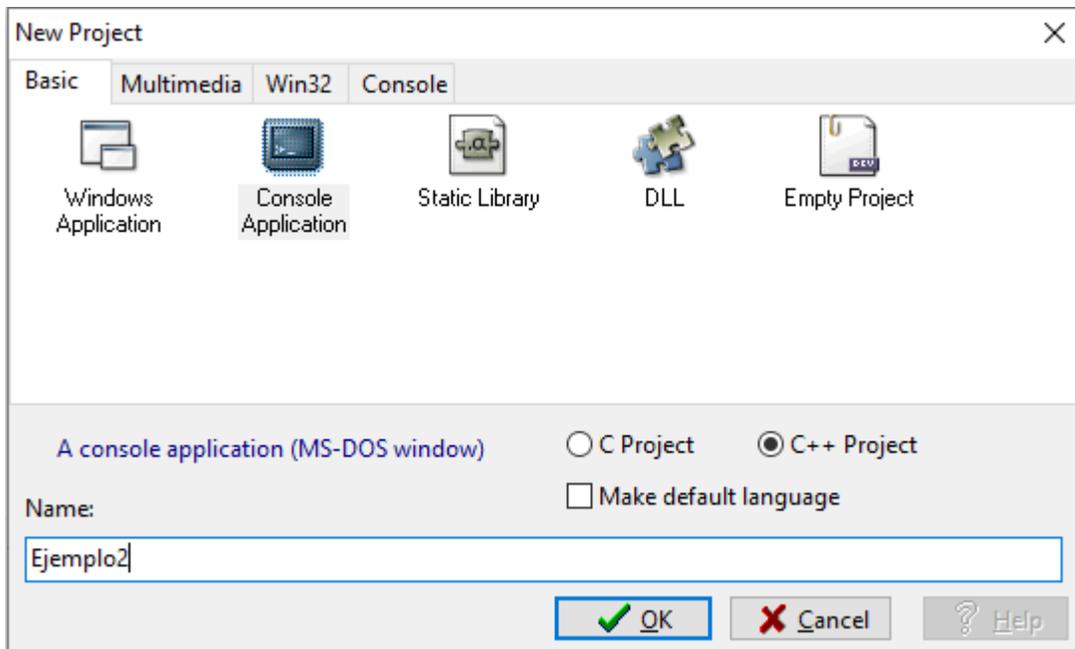
```
C:\Users\Sistemas\Documents\LPOO_Analilia\DEV\Ejemplo1.exe
Today is 9/24/2003
Another day is 9/25/2003
today comes before anotherDay

-----
Process exited after 0.1384 seconds with return value 0
Presione una tecla para continuar . . .
```

## Ejemplo 2:

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

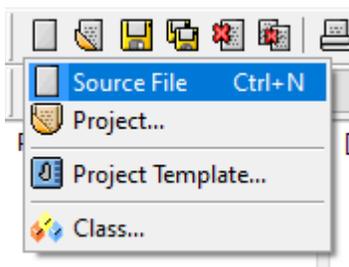




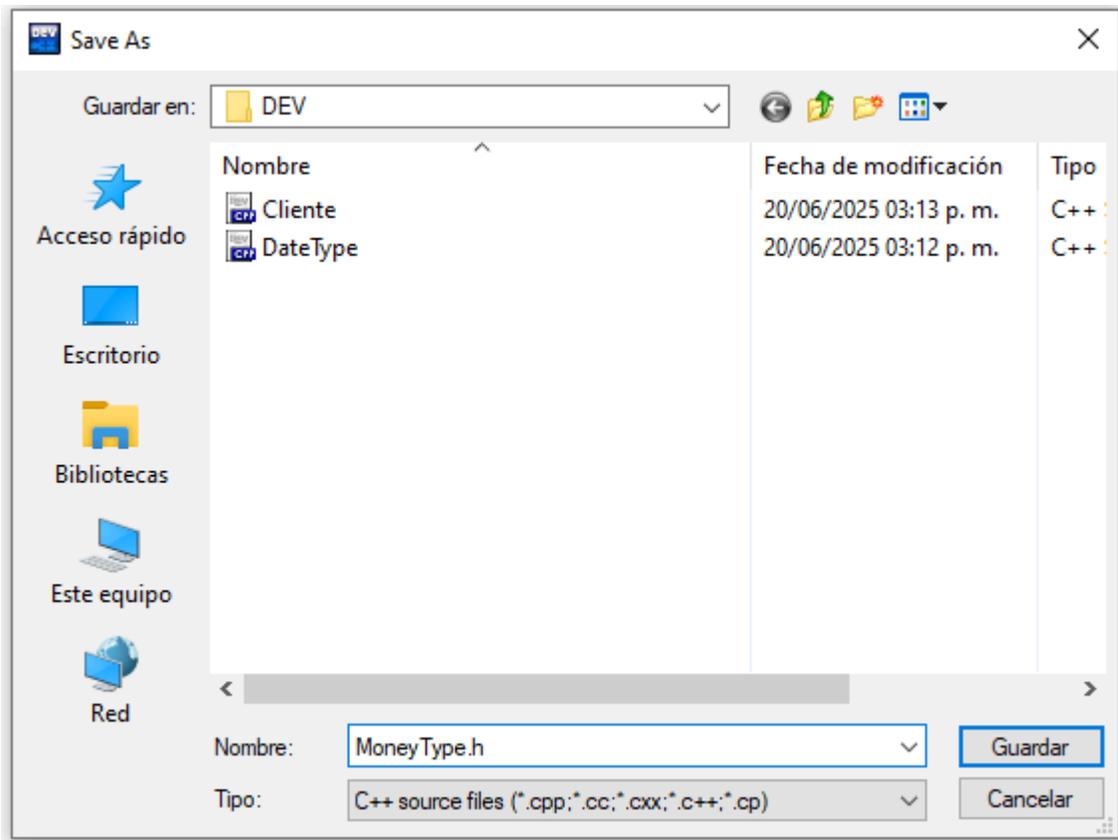
Agregamos el siguiente código en el main que se crea por defecto:

```
Project Classes Debug [*] main.cpp
Ejemplo2
  main.cpp
1 #include <iostream>
2 #include "MoneyType.h"
3
4 using namespace std;
5
6 main(){
7     MoneyType *money = new MoneyType();
8     ExtMoneyType *extMoney = new ExtMoneyType();
9
10    money->Initialize(60,20);
11    extMoney->Initialize(60,20,"francs");
12
13    cout << "Money: " << money->GetDollars() << " Dollars and " << money->GetCents() << " Cents." << endl;
14    cout << "extMoney( "<< extMoney->GetCurrency() <<" ): " << extMoney->GetDollars() << " Dollars and " << extMoney->GetCents() << " Cents"<<
15
16    delete money;
17    delete extMoney;
18    getchar();
19 }
20
```

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

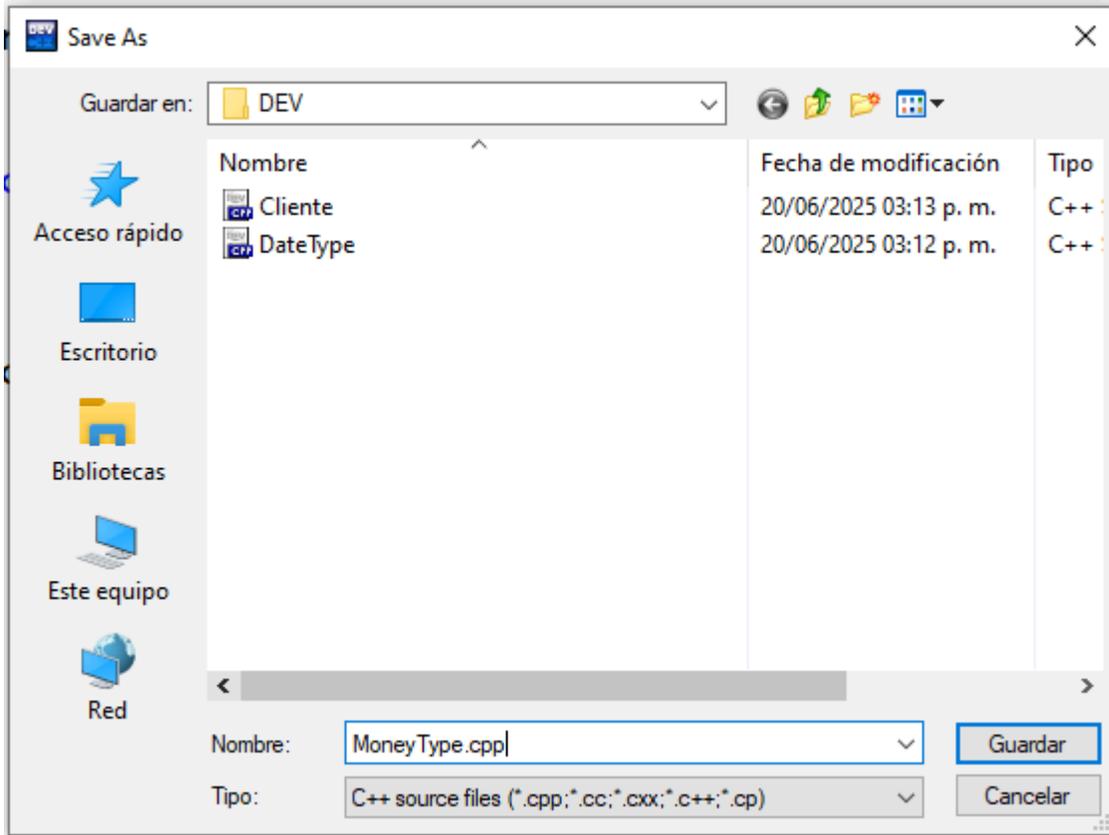


```
Project Classes Debug [*] main.cpp [*] Untitled10
Ejemplo2
  main.cpp
  Untitled10
5 using namespace std;
6
7 class MoneyType{
8     public:
9         void Initialize(long, long);
10        long GetDollars() const;
11        long GetCents() const;
12
13    private:
14        long dollars;
15        long cents;
16 };
17
18 class ExtMoneyType : public MoneyType{
19     public:
20         string GetCurrency() const;
21         void Initialize(long, long, string);
22
23     private:
24         string currency;
25 };
26
27
28 #endif /* _MONEYTYPE_H_ */
```

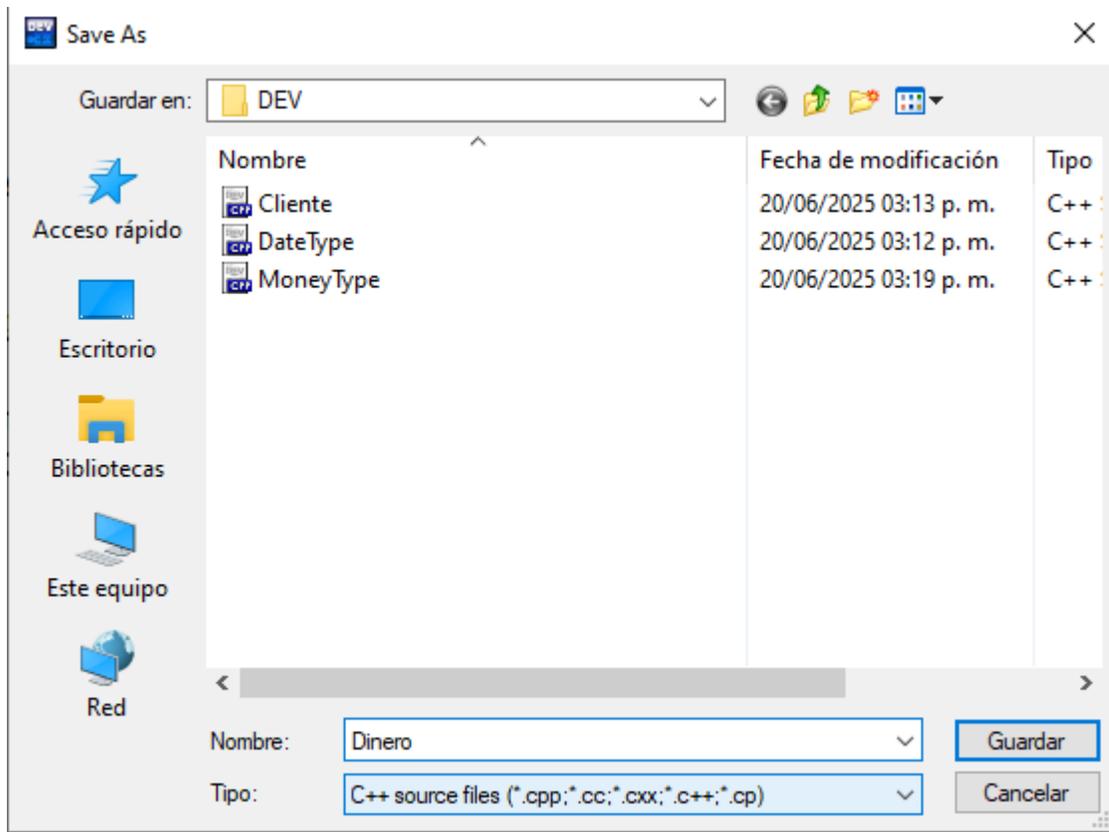


Creamos el archivo MoneyType.cpp en donde estará la implementación de las funciones que se declaran en el header.

```
Project Classes Debug [main.cpp MoneyType.h Untitled11]
Ejemplo2
├── MoneyType.h
├── main.cpp
└── Untitled11
1 #include "MoneyType.h"
2
3 void ExtMoneyType :: Initialize (long newDollars, long newCents, string newCurrency){
4     currency = newCurrency;
5     MoneyType :: Initialize(newDollars, newCents);
6 }
7
8 string ExtMoneyType :: GetCurrency() const {
9     return currency;
10 }
11
12 void MoneyType :: Initialize(long newDollars, long newCents){
13     dollars = newDollars;
14     cents = newCents;
15 }
16
17 long MoneyType :: GetDollars() const {
18     return dollars;
19 }
20
21 long MoneyType :: GetCents() const {
22     return cents;
23 }
```



Guardamos el archivo main.cpp como Dinero.cpp



Ejecutamos y compilamos:

C:\Users\Sistemas\Documents\LPOO\_Analilia\DEV\Ejemplo2.exe

```
Money: 60 Dollars and 20 Cents.  
extMoney( francs ): 60 Dollars and 20 Cents
```