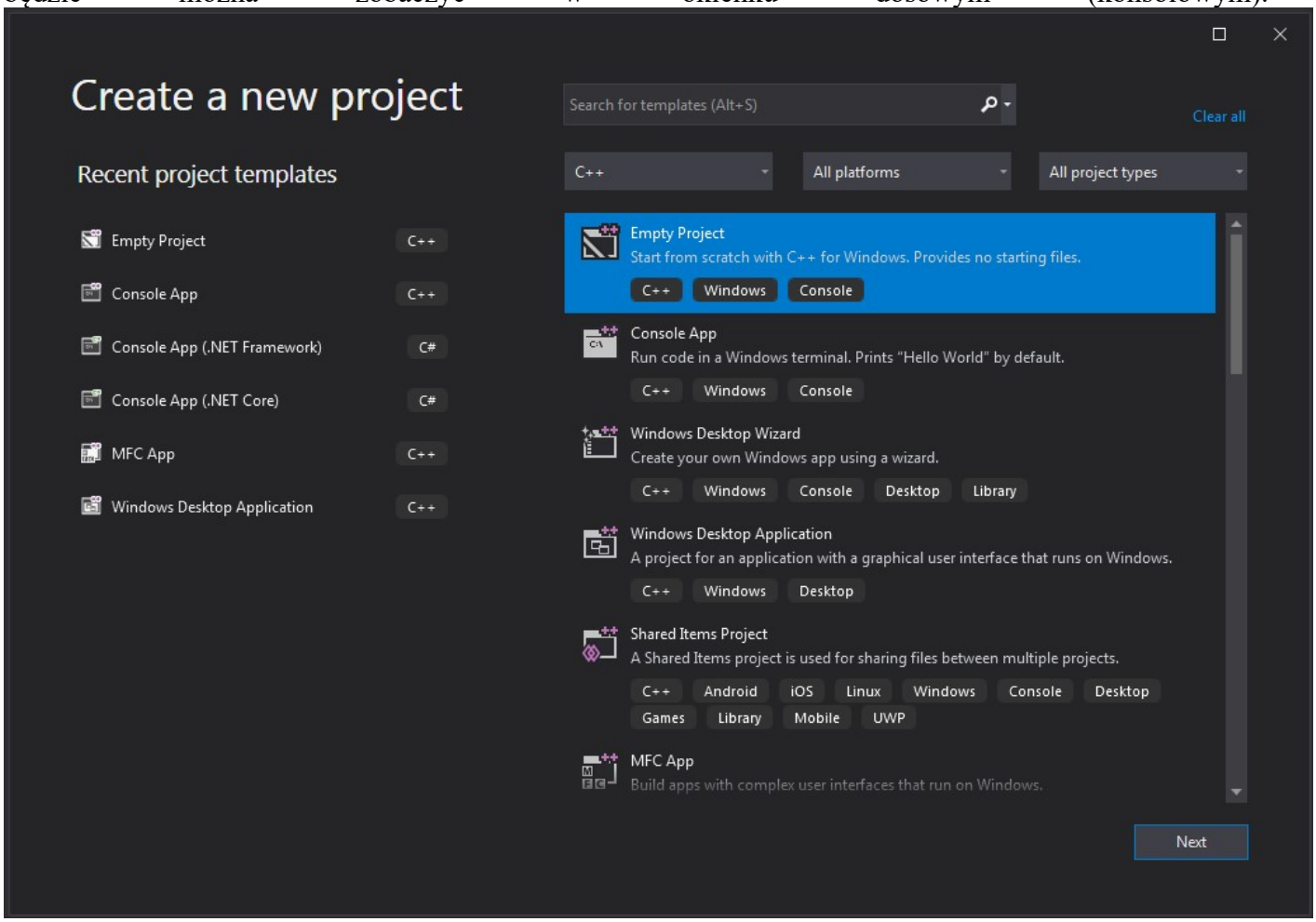
Zajęcia 2: Opracowanie klas: `Data` i `Napis`.

1. Opracowanie klasy.

Celem ćwiczenia jest opracowanie klas, których odpowiedniki znajdziemy w bibliotekach standardowych (np. klasa `Napis` mogłaby być zastąpiona klasą `string` zdefiniowanej w bibliotece standardowej o tej samej nazwie). Dla wykonania założonych celów dydaktycznych w realizowanym projekcie, obiektów klasy **`string` nie wolno używać**. Natomiast przy definiowaniu ciał metod klasy `Napis` można i nawet należy korzystać z funkcji z biblioteki standardowej `string.h (np. strcpy, strlen, strcmp itp.)`.

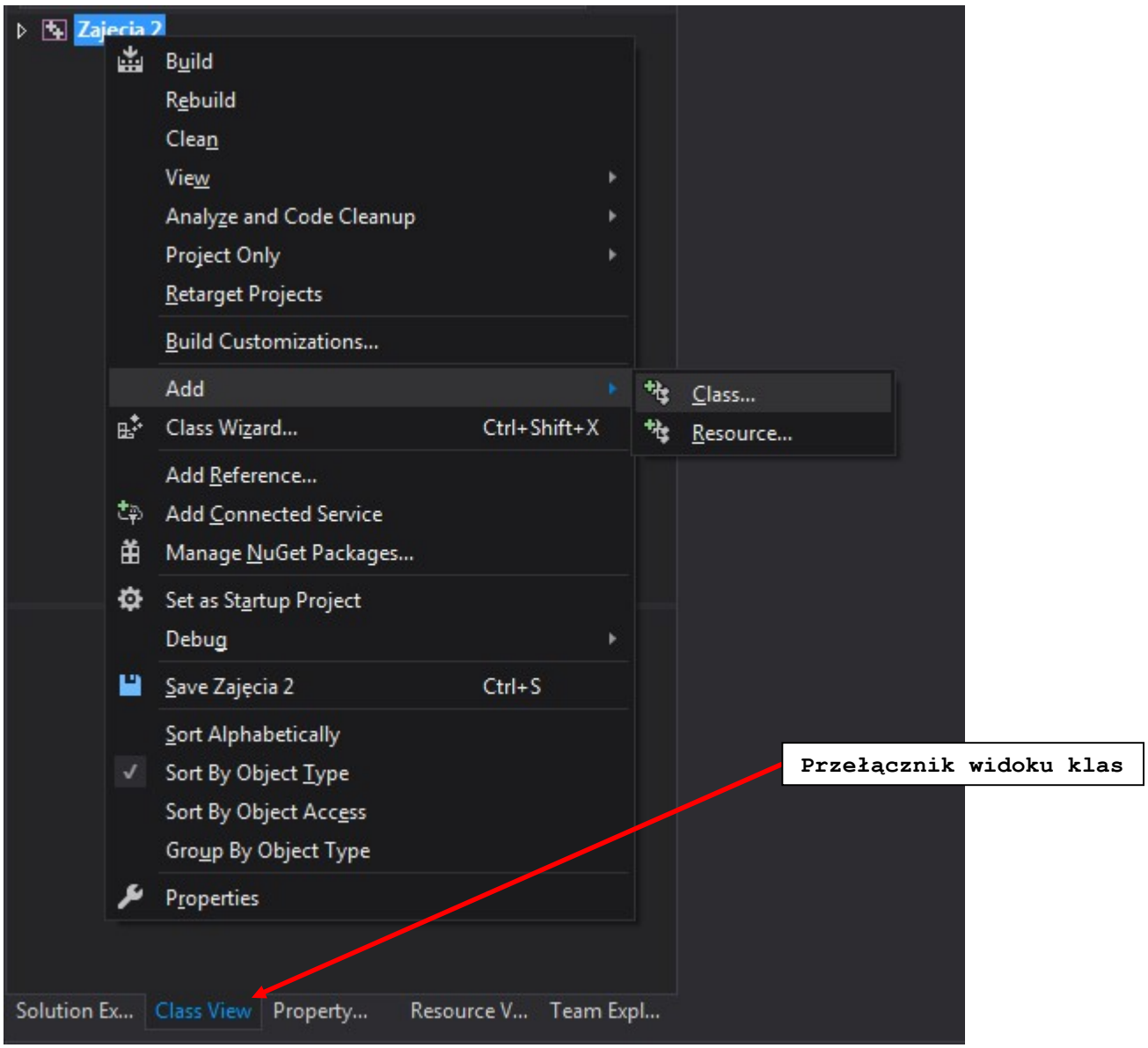
Tworzenie szkieletu klasy

- Utwórz nowy projekt wybierając opcję **New|Project** z menu **File**. Wybierz język C++ (patrz rysunek poniżej), a następnie projekt o nazwie **Empty project** i kliknij przycisk **Next.** W polu tekstowym **Location** wpisz: **h:\Nazwisko_Imię**.. W polu **Project name** podaj dowolną nazwę projektu. Po wciśnięciu przycisku **Create**, pojawi się domyślne okno projektu. Wynik działania takiego projektu będzie można zobaczyć w okienku dosowym (konsolowym).

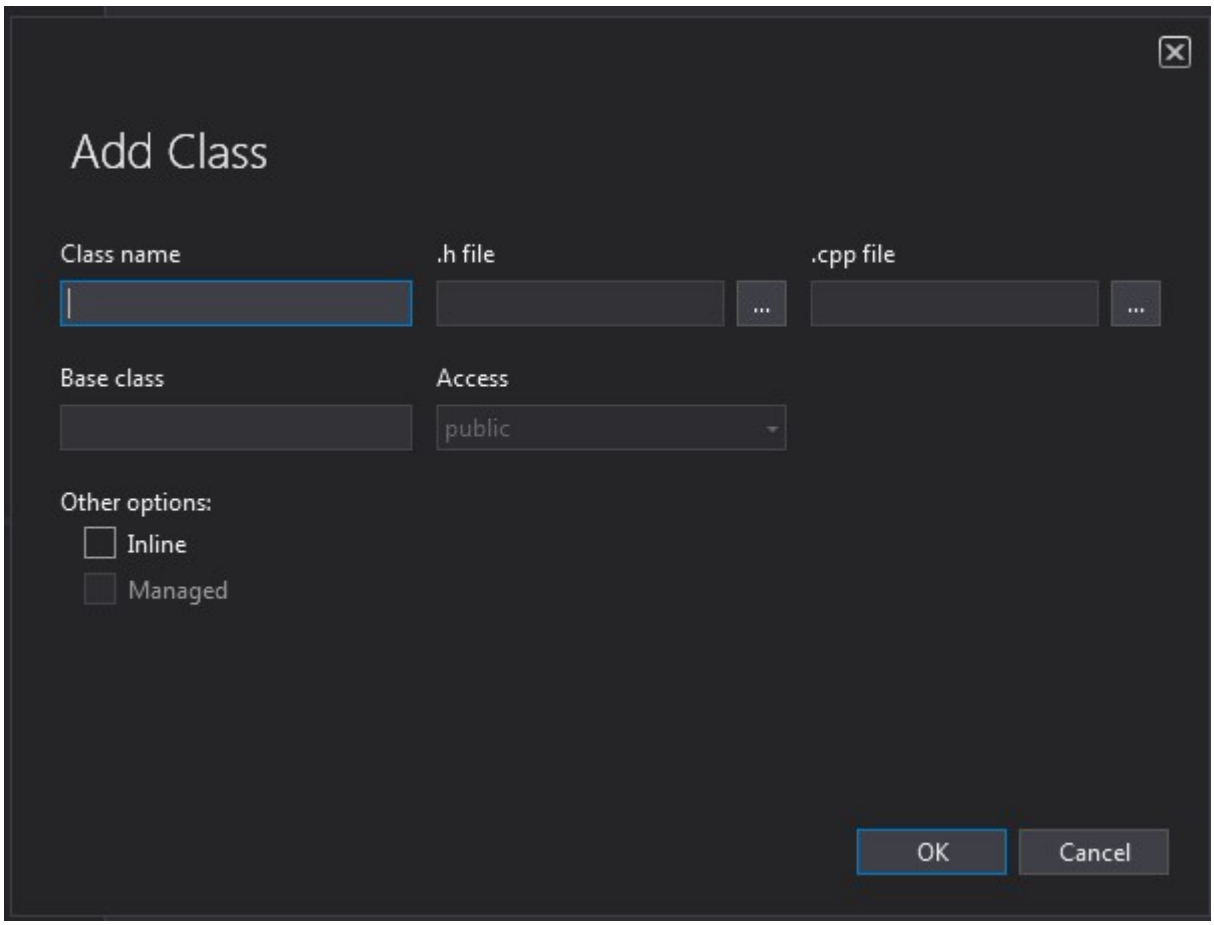


Takie ustawienia spowodują, że projekt nie będzie miał żadnych automatycznie dodanych plików źródłowych i nagłówkowych.

- Przełącz panel z zawartością projektu w tryb widoku klas (**ClassView**):

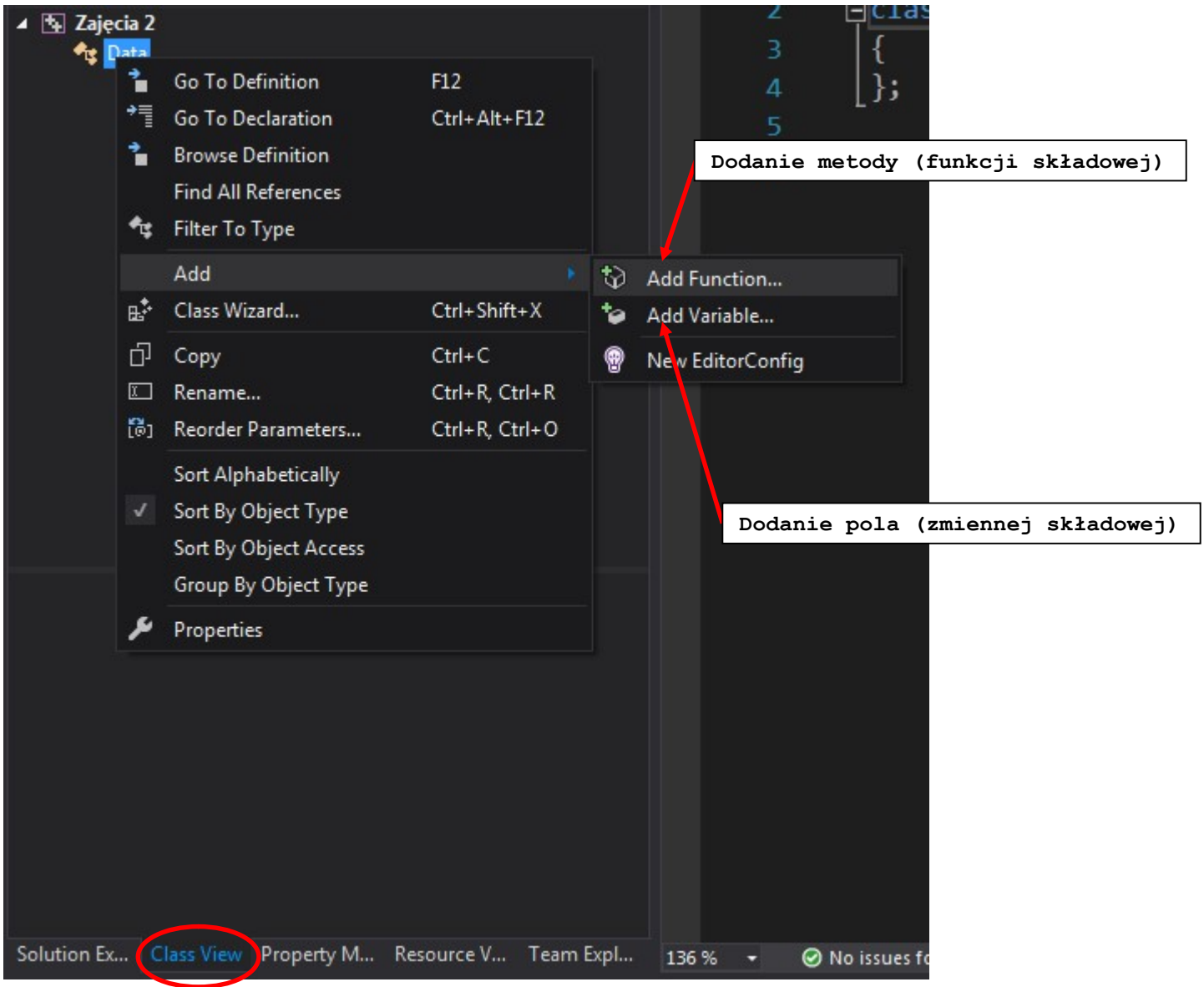


Na ekranie zobaczysz nowo stworzony projekt bez żadnych dodanych plików nagłówkowych i źródłowych. Kliknij prawym przyciskiem myszki na nazwę projektu i z menu lokalnego wybierz opcję **Add**, potem **Class**. Na ekranie pojawi się następujące okienko dialogowe:



Podaj nazwę klasy (**Class name**) `Data` i kliknij **OK**. Warto zauważyć, że kreator środowiska Visual Studio automatycznie wypełnił pola pliku nagłówkowego **Data.h** (tu będzie znajdować się deklaracja klasy) i źródłowego **Data.cpp** (tu będą umieszczane definicje wszystkich metod – funkcji składowych tej klasy).

- Po powrocie do widoku projektu (**Solution Explorer**) zauważysz, że utworzone zostały dwa pliki **Data.cpp** i **Data.h**.
- Przełącz panel na widok **Class View**. Aby za pomocą kreatora dodać składnik klasy - pole (daną składową) bądź metodę (funkcję składową) - należy z menu lokalnego dla klasy `Data` (kliknięcie prawym klawiszem myszki na nazwie klasy) wybrać odpowiednio **Add Variable** (gdy chcemy dodać pola składowe klasy) albo **Add Function** (gdy chcemy dodać funkcje składowe klasy). Obrazuje to kolejny rysunek:



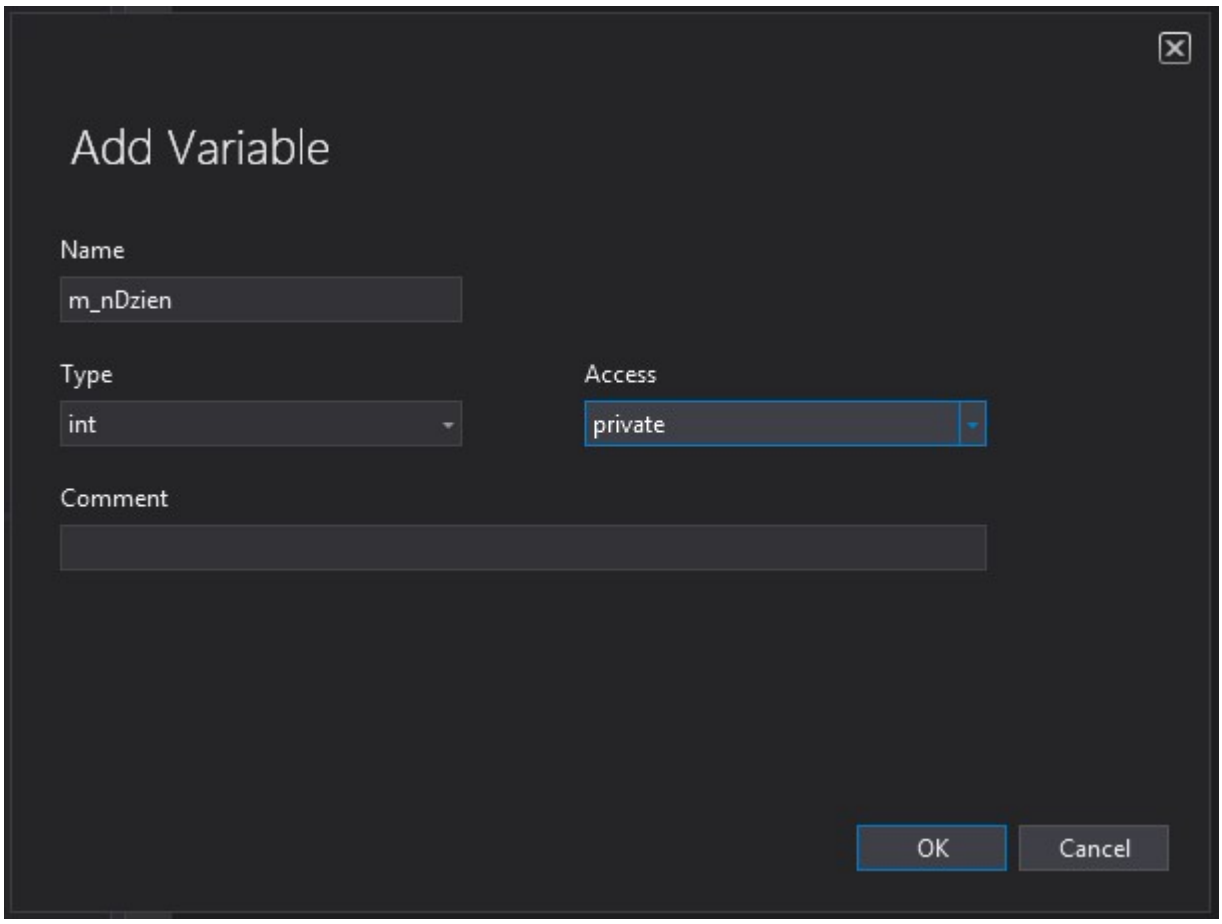
- Zwróć uwagę na menu lokalne dla poszczególnych składników klasy. Znajdują się tam m.in. polecenia **Go to Declaration** i **Go to Definition** ułatwiające poruszanie się zwłaszcza po dużym wielomodułowym projekcie. Staraj się korzystać z tych ułatwień. Dwukrotne kliknięcie na nazwie funkcji/metody powoduje przejście do definicji funkcji/metody.

Dodanie składowych do utworzonej klasy

- Do utworzonej klasy dodaj następujące dane składowe (prywatne):

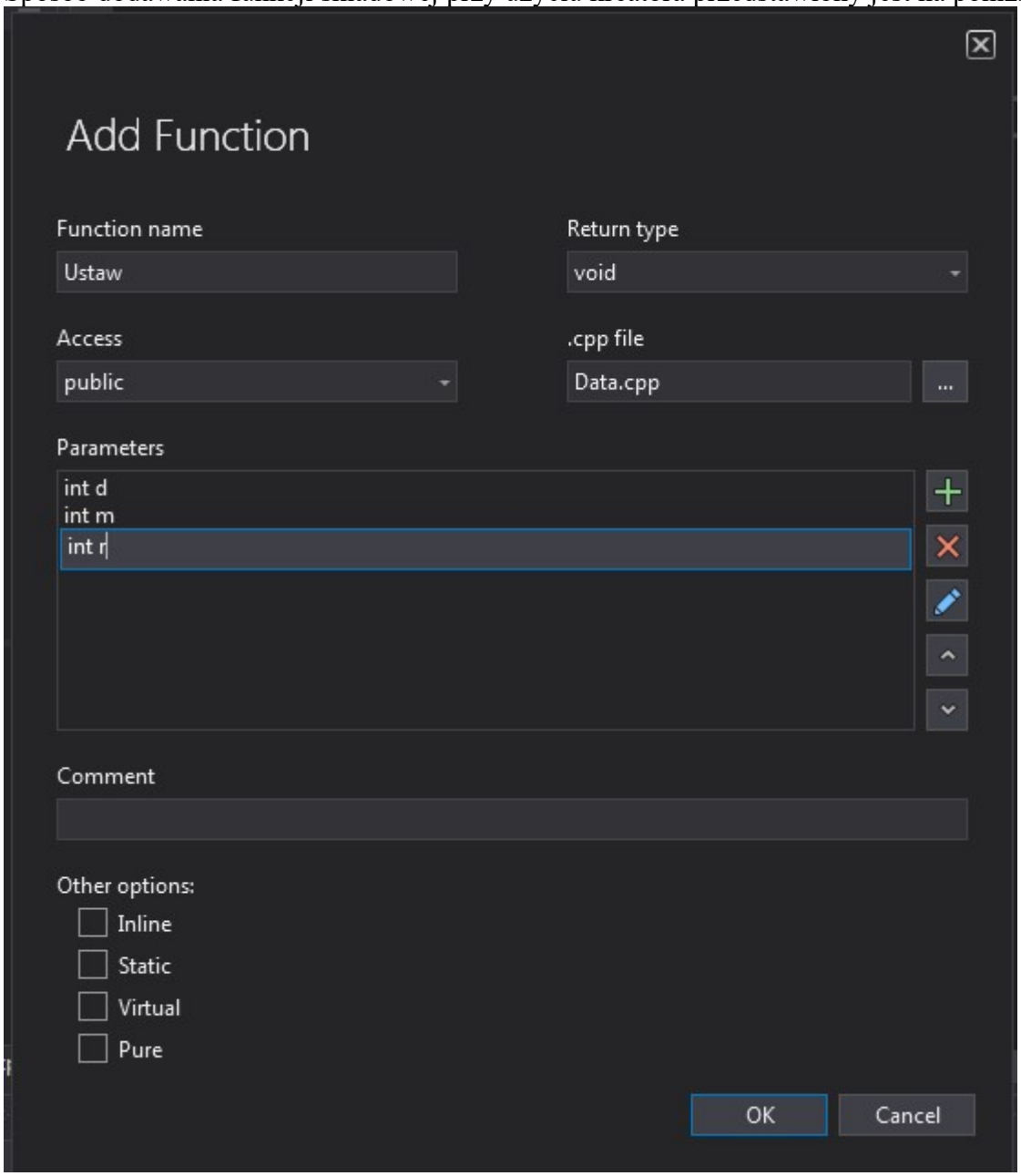
```
int m_nDzien,
int m_nMiesiac,
int m_nRok
```

ręcznie lub korzystając z kreatora:



- metodę (publiczną) ustawiającą poszczególne składniki klasy:
  `void Ustaw(int d, int m, int r);`

Sposób dodawania funkcji składowej przy użyciu kreatora przedstawiony jest na poniższym rysunku:



W polu **Return type** (typ zwracany) wybieramy typ `void`. W polu nazwa funkcji (**Function name**) podajemy nazwę funkcji bez nawiasów (w tym przypadku `Ustaw`). Aby dodać pierwszy parametr (argument) funkcji: `int d`, należy przycisnąć + i wpisać `int d`, Z pozostałymi argumentami funkcji postępujemy analogicznie.

- metody informacyjne (publiczne), mające za zadanie zwrócić wartość poszczególnych pól składowych klasy (`m_nDzien`, `m_nMiesiac`, `m_nRok`):

```
int Dzien() const;
int Miesiac() const;
int Rok() const;
```

Przyrostek `const` należy dopisać ręcznie w deklaracji funkcji (**Data.h**) oraz definicji funkcji (**Data.cpp**); Jeżeli funkcja nie posiada żadnych argumentów, pola **Parameters** zostawiamy puste;

- publiczną metodę wyprowadzającą dane składowe do standardowego strumienia wyjściowego (`cout`):
  `void Wypisz() const;`
  wypisującą datę w formacie dzien-miesiac-rok (np. 19-10-2022)
- publiczną metodę wprowadzającą dane składowe ze standardowego strumienia wejściowego (`cin`):
  `void Wpisz();`
- prywatną metodę korygującą niepoprawnie ustawioną datę:
  `void Koryguj();`
  metoda ma za zadanie poprawić datę na najbliższą właściwą, np.:
  datę `34-1-2020` na `31-1-2020`
  datę `29-2-2023` na `28-2-2023` (bo 2022 nie jest rokiem przestępnym)
  datę `0-1-2024` na `1-1-2024`
  datę `31-4-2017` na `30-4-2017`
  metodę tę należy użyć w metodzie `Ustaw` oraz `Wpisz`
- publiczną metodę porównującą datę (pola składowe klasy) z przekazywanym wzorcem (`wzor`):
  `int Porownaj(const Data & wzor) const;`
  metoda ma zwracać wartość `0` (zero), gdy składowe są identyczne ze składowymi podanego wzorca, `1` gdy data wzorca jest późniejsza od daty obiektu wskazywanego przez `this` albo `-1` w przeciwnym wypadku.
- Jeśli tworzone przez Ciebie metody nie zmieniają składowych, to dodawaj słowo `const` w nagłówku metody np.: `void Wypisz() const;`

2. Opracowanie klasy `Napis`

Tworzenie szkieletu klasy

Przełącz panel z zawartością projektu w tryb widoku klas (**ClassView**) i dodaj do projektu kolejną klasę o nazwie `Napis` (postępuj analogicznie jak w przypadku dodawania klasy `Data`). W widoku klas będą już widoczne dwie dodane do projektu klasy. Po powrocie do widoku projektu (**Solution Explorer**) zauważysz, że utworzone zostały dwa pliki **Napis.cpp** i **Napis.h**.

Dodanie danych składowych do klasy `Napis`

W definicji klasy (plik **Napis.h**) dodaj ręcznie następujące prywatne pole składowe:
`char m_pszNapis[40];`
Następnie w części publicznej klasy zdefiniuj:

- metodę informacyjną (zwracającą pole składowe klasy):
  `const char* Zwroc() const;`
  W polu **Return type** nie ma zwracanego typu `const char*`, należy zatem wpisać go ręcznie;
- metodę składową ustawiającą pole `m_pszNapis` na podstawie przekazanego argumentu:
  `void Ustaw(const char* nowy_napis);`
- metodę wyprowadzającą zawartość pola składowego do standardowego strumienia wyjściowego (cout):
  `void Wypisz() const;`
- metodę wprowadzającą ze standardowego strumienia wejściowego (`cin`) do pola składowego klasy:
  `void Wpisz();`
- metodę porównującą wartość pola składowego z przekazywanym wzorcem:
  `int SprawdzNapis(const char* por_napis) const;`
  metoda ma zwracać wartość `0 (zero)`, gdy pole składowe klasy (`m_pszNapis`) jest identyczne z podanym wzorcem (`por_napis`), wartość większą od zera, gdy pole `m_pszNapis`, jest alfabetycznie dalej niż przekazany argument (`por_napis`) i mniejszą od 0 w przeciwnej sytuacji.