Dzielenie kodu na kilka plików źródłowych

W programowaniu w C++, projekty są zazwyczaj podzielone na pliki nagłówkowe (header files) i pliki źródłowe (source files).

1. Pliki nagłówkowe (.h lub .hpp):

- Pliki nagłówkowe zawierają deklaracje klas, funkcji i zmiennych, ale nie ich implementację.
- Zawierają dyrektywy preprocesora (np. #ifndef, #define, #endif) do zabezpieczenia przed wielokrotnym dołączaniem (header guards).

Opis użytych instrukcji preprocesora:

#ifndef zmienna_preprocesora	Instrukcja preprocesora, która sprawdza czy zmienna preprocesora o podanej nazwie istnieje. Jeśli zmienna preprocesora nie istnieje to wszystkie instrukcje, które się znajdują poniżej #ifndef, zostaną wykonane. Jeśli natomiast zmienna będzie istniała, kompilator pominie wszystkie instrukcje jakie znajdują się pomiędzy słowami preprocesora #ifndef, a #endif.
endif	instrukcja oznacza miejsce końca bloku warunkowego preprocesora.
#define nazwa_zmiennej	Polecenie służy do tworzenia zmiennych preprocesora.

Zaprezentowany przykład czytasz następująco:

#ifndef **dowolna_nazwa**	jeśli nie istnieje **dowolna_nazwa**, wykonuj blok
#define **dowolna_nazwa**	utwórz zmienną o nazwie **dowolna_nazwa**
#endif	koniec bloku warunkowego

- Każdy plik nagłówkowy powinien zawierać tylko i wyłącznie interfejs. Przez słowo interfejs rozumiemy:
  - deklaracje typów
  - deklaracje funkcji
  - deklaracje struktur
  - deklaracje klas
  - deklarację ewentualnych zmiennych globalnych

Przykład pliku nagłówkowego **Punkt.h**:

```
#ifndef PUNKT_H
#define PUNKT_H

class Punkt {
public:
    Punkt(int x, int y);
    void Wyswietl();
private:
    int x, y;
};
```

```
#endif
```

2. Pliki źródłowe (.cpp):

- Pliki źródłowe zawierają implementację funkcji i metod zadeklarowanych w plikach nagłówkowych.

- Każda klasa powinna mieć odpowiadający jej plik źródłowy.

Przykład pliku źródłowego **Punkt.cpp**:

```
#include "Punkt.h" //dołączanie nagłówkowego pliku, który opisuje interfejs jaki znajduje się w tym pliku

#include <iostream>

Punkt::Punkt(int x, int y) : x(x), y(y) {}

void Punkt::Wyswietl() {
    std::cout << "Punkt: (" << x << ", " << y << ")\n";
}
```

3. Plik główny (main.cpp):

- Plik, który zawiera funkcję main() i służy do rozpoczęcia wykonywania programu.

- W tym pliku możesz tworzyć obiekty klas, wywoływać funkcje, itp.

Przykład pliku głównego **main.cpp**:

```
#include "Punkt.h"

int main() {
    Punkt punkt(1, 2);
    punkt.Wyswietl();

    return 0;
}
```

Jak to działa?

Plik main.cpp nie "widzi" zawartości pliku Punkt.cpp tylko na podstawie samego #include "Punkt.h". W C++, pliki źródłowe (.cpp) muszą być skompilowane osobno, a następnie połączone, aby utworzyć ostateczny program.

W typowym procesie kompilacji i linkowania:

Kompilacja plików źródłowych:

1. Każdy plik źródłowy (.cpp) jest kompilowany osobno do pliku obiektowego (.o lub .obj), który zawiera kod maszynowy dla danego pliku.
    a. Na przykład, kompilacja Punkt.cpp stworzy plik obiektowy Punkt.o.
2. Linkowanie:
    a. Kiedy wszystkie pliki źródłowe są skompilowane, linkowanie łączy pliki obiektowe w jeden program.
    b. W przypadku main.cpp, linkowanie łączy plik obiektowy main.o z plikami obiektowymi innych plików źródłowych, w tym Punkt.o.
3. Tworzenie wykonawczego pliku programu:
    a. Po linkowaniu otrzymuje się plik wykonywalny, który można uruchomić.

Podsumowując, main.cpp nie "widzi" zawartości Punkt.cpp w sposób bezpośredni. Każdy plik źródłowy musi zostać skompilowany, a wynikowe pliki obiektowe są łączone podczas linkowania, aby utworzyć ostateczny program. Jednak main.cpp korzysta z deklaracji i definicji zawartych w pliku nagłówkowym Punkt.h.