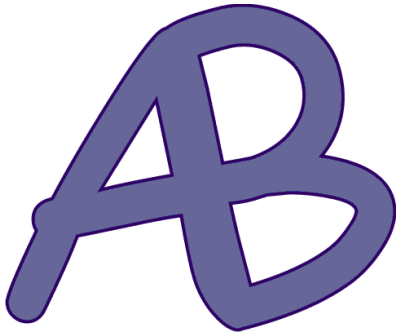


Podział kodu na pliki

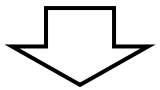
Klasa jako zewnętrzy plik

Podział kodu

W większych projektach programistycznych dobrą praktyką jest podział kodu na pliki:

- Plik nagłówkowy (.h) – zawiera deklaracje klasy, jej metod i pól.
- Plik implementacyjny (.cpp) – zawiera definicje metod klasy.
- Plik główny (często nazywany „main.cpp”) – zawiera funkcję startową main() oraz kod sterujący programem.

Sposób podziału programu na pliki zaprezentujemy przykładzie, znanej z poprzednich wykładów, klasy „Osoba”

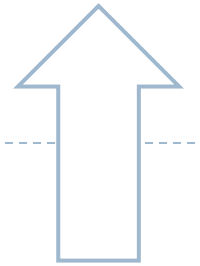


Przykład – na razie wszystko w jednym pliku

```
#include <iostream>
using namespace std;

class Osoba
{
private:
    string imie;
    string nazwisko;
public:
    // Konstruktor
    Osoba(string imie, string nazwisko);

    // Gettery
    string getImie();
    string getNazwisko();

    // Settery
    void setImie(string imie);
    void setNazwisko(string nazwisko);
};
```

```
Osoba::Osoba(string imie, string nazwisko)
{
    this->imie = imie;
    this->nazwisko = nazwisko;
}
string Osoba::getImie()
{
    return imie;
}
string Osoba::getNazwisko()
{
    return nazwisko;
}
void Osoba::setImie(string imie)
{
    this->imie = imie;
}
void Osoba::setNazwisko(string nazwisko)
{
    this->nazwisko = nazwisko;
}
```

Przykład – na razie wszystko w jednym pliku



```
int main()
{
    // Przyklad uzycia klasy Osoba
    Osoba ktos("Jan", "Kowalski");
    cout << "Imie: "
         << ktos.getImie()
         << endl
         << "Nazwisko: "
         << ktos.getNazwisko()
         << endl;
    return 0;
}
```

Na poprzednim slajdzie zwróćmy uwagę, że w ciele klasy „Osoba” znajdują się tylko nagłówki jej metod implementację metod są poza klasą.

Jak widzimy większość programu stanowi klasa. Wyciągnięcie jej do osobnego pliku może między innymi poprawić czytelność. A także umożliwić wykorzystanie już raz przygotowanej klasy w kolejnych programach.

Podział kodu

Plik nagłówkowy – Osoba.h

#ifndef OSOBA_H i #define OSOBA_H
Zapobiega wielokrotnemu dołączeniu tego samego pliku (nagłówkowe zabezpieczenie).

```
#ifndef OSOBA_H
#define OSOBA_H
#include <string>

class Osoba
{
private:
    std::string imie;
    std::string nazwisko;

public:
    // Konstruktor
    Osoba(std::string imie, std::string nazwisko);

    // Gettery
    std::string getImie();
    std::string getNazwisko();

    // Settery
    void setImie(std::string imie);
    void setNazwisko(std::string nazwisko);
};

#endif
```

Dyrektywa zapobiegająca wielokrotnemu dołączeniu bloku kodu o nazwie ”OSOBA_H”

Początek definicji bloku kodu o nazwie „OSBA_H”

Koniec bloku kodu

Podział kodu

Plik implementacyjny – Osoba.cpp

Pamiętać należy aby odwoływać się do przestrzeni nazw klasy z pliku nagłówka owego

Przed nazwą zmiennej string pojawiła się przestrzeń nazw `std::` wynika to stąd, że nie zastosowano dyrektywy `using namespace std;` (nie jest to zalecane)

```
#include <iostream>
#include "Osoba.h"

Osoba::Osoba(std::string imie, std::string nazwisko) {
    this->imie = imie;
    this->nazwisko = nazwisko;
}

std::string Osoba::getImie()
{
    return imie;
}

std::string Osoba::getNazwisko()
{
    return nazwisko;
}

void Osoba::setImie(std::string imie)
{
    this->imie = imie;
}

void Osoba::setNazwisko(std::string nazwisko)
{
    this->nazwisko = nazwisko;
}
```

Dołączenie pliku nagłówkowego

Podział kodu

Plik główny – program w którym zastosujemy klasę „Osoba”

```
#include "Osoba.h"
#include <iostream>

using namespace std;

int main()
{
    Osoba ktos("Jan", "Kowalski");
    cout << "Imie: " << ktos.getImie() << endl
         << "Nazwisko: " << ktos.getNazwisko() << endl;
    return 0;
}
```

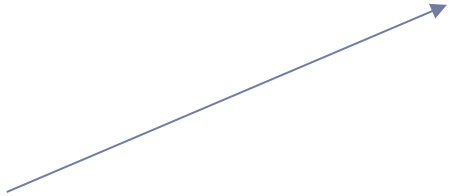
Dołączenie pliku nagłówkowego

Tworzymy obiekt klasy Osoba zdefiniowanej w osobnym pliku

Kompilacja programu

Kompilacja programu składającego się z wielu plików

```
g++ main.cpp Osoba.cpp -o nazwa_programu
```



Komponując program podajemy listę plików c++ z które wchodzą w skład programu - bez plików nagłówków owych.

Pliki nagłówkowe dołączą się automatycznie.

```
g++ main.cpp Osoba.cpp
```

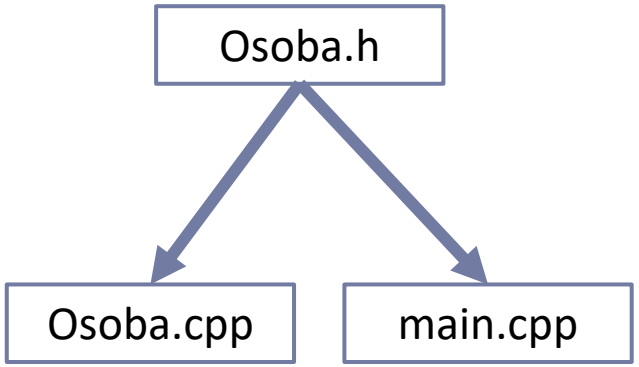
Jeżeli nie podamy parametru „-o” i nazwy pliku utworzony zostanie plik a.exe

Uwagi

#include w wielu plikach – czy to problem?

- ✓ Nie, jeśli używamy go prawidłowo!
- ✓ Każdy plik .cpp powinien mieć #include tylko do tego, czego sam potrzebuje.
- ✓ #include <iostream> w wielu plikach .cpp nie powoduje problemów, ponieważ kompilator kompiluje pliki osobno.

Dlaczego należy zabezpieczać zawartość pliku nagłówkowego przed wielokrotną implementacją?



Plik nagłówkowy dołączony był do obu plików które zostały kompilowane