

WZORCE

PODEJŚCIE DO PROGRAMOWANIA UOGÓLNIONEGO

CZYM SĄ WZORCE

Wzorce (lub inaczej szablony - z ang. *templates*) to sposób na napisanie uogólnionej, sparametryzowanej klasy lub funkcji, której parametrem będzie typ, bądź inna klasa.

SZABLON FUNKCJI

Wzorce (lub inaczej szablony - z ang. *templates*) to sposób na napisanie uogólnionej, sparametryzowanej klasy lub funkcji, której parametrem będzie typ, bądź inna klasa.

```
template <class T>
T coWieksize(T a, T b) {
    T wynik = (a > b) ? a : b;
    if (a == b)
        std::cout << "Oba argumenty maja wartosc ";
    return wynik;
}

int main()
{
    std::cout << coWieksize<int>(5.209, 5.290) << std::endl;
    std::cout << coWieksize<std::string>("Ala", "Alojz") << std::endl;
    std::cout << coWieksize<double>(5.209, 5.290) << std::endl;
}
```

TYPENAME CZY CLASS

Generalnie nie ma różnicy którego słowa użyjemy, efekt działania jest ten sam. Najlepiej jednak używać jednego z nich i trzymać się go w danym projekcie by nie wprowadzać bałaganu.

KLASA SZABLONOWA

Pisząc programy często korzystamy z abstrakcyjnych typów danych, takich jak stos, kolejka czy drzewo. Implementacje takich typów mogą być prawie identyczne, na przykład klasy `lista_liczb` i `lista_znaków` mogą różnić się tylko typem elementu przechowywanego na liście.

Na wzorcach opartych jest wiele technik obiektowych oraz są podstawą nowoczesnego programowania generycznego.

```
template<class T>  
  
class Tablica{  
  
}
```

DEKLARACJA KLASY SZABLONOWEJ

```
Tablica<int> myTab(10);
```

```
template<class T>
class Tablica {
private:
    T* tab;
    int rozmiar;
public:
    Tablica(int n) {
        tab = new T[r]
    };

    ~Tablica() {
        delete[] tab;
    }
};
```

ARGUMENTY WZORCA

```
template<class T,int rozmiar>
class Bufor {
private:
    T bufor[rozmiar];
public:
    void zapisz(int index, T wartosc) {
        bufor[index] = wartosc;
    }
    T odczytaj(int index) {
        return bufor[index];
    }
};
```

```
Bufor<double, 10> bufor;
bufor.zapisz(2, 4.5);

std::cout << bufor.odczytaj(2) << std::endl;
```

4.5

Press any key to continue . . .

WZORZEC OGÓLNY ORAZ SZCZEGÓŁOWY

```
template<class T>
class Compare {
public:
    bool porownaj(T a, T b) {
        return a < b;
    }
};
```

```
template<>
class Compare<char*> {
public:
    bool porownaj(const char* a, const char* b) {
        return std::strcmp(a, b) > 0;
    }
};
```

```
Compare<int> comp;
std::cout << comp.porownaj(2,4) << std::endl;

Compare<char*> comp1;
std::cout << comp1.porownaj("Adam", "Mateusz") << std::endl;
```

```
1
0
Press any key to continue . . .
```

```
1
1
Press any key to continue . . .
```

RTTI

```
template<class T>
class myClass {
public:
    void whatType() {
        std::cout << typeid(*this).name() << std::endl;
    }
};
```

```
myClass<int> typ1;
myClass<double> typ2;
myClass<char*> typ3;
```

```
typ1.whatType();
typ2.whatType();
typ3.whatType();
```

```
class myClass<int>
class myClass<double>
class myClass<char *>
Press any key to continue . . .
```

PROBLEMY Z SZABLONAMI

1. Problemy z walidacją
2. Dużo dodatkowego kodu
3. Skomplikowana składnia