Szablony (wzorce)

ang. templates

Szablony a polimorfizm

Polimorfizm: jeden typ funkcji (metody), wiele możliwych instrukcji do realizacji.

Szablon: jedna zdefiniowana przez programistę instrukcja (zestaw instukcji), wiele różnych typów.

Szablon...

...czyli programowanie uogólnione.

Szablony dotyczą funkcji (metod) i klas.

Wzorce zawierają ogólny, predefiniowany “kształt” obiektu, czyli zawarte w nim metody i zmienne. **Nie wymuszają** jednak odgórnego **wybierania typu** (np. zwracanej zmiennej).

Słowa kluczowe, czyli z czym to się je?

```
template<typename t1, typename t2, … , typename tn>
```

Można użyć więcej niż jednego parametru przy tworzeniu szablonu.

```
typename t1 = class t1
```

class - “bardziej” zgodne ze zdefiniowanym standardem c++

typename - używany zamiennik (żeby nie przeciążać “class”)

Konstrukcja dla klasy i funkcji

```
template<typename t1, typename t2, … , typename tn> // szablon klasy
class nazwa_klasy: public klasa_bazowa<t1>{
      t2 zmienna;
      template <typename T1> // szablon metody
            T1 funkcja (T1 param1, T1* param2){
            …
            }
      …
}

int main(){
      nazwa_klasy *nowa_klasa<int, float, … , double> = new nazwa_klasy<int, float, … , double>();
}
```

To jaki można podać typ?

Podstawowym założeniem jest wykorzystywanie “różnych” typów dla klasy lub funkcji o tej samej konstrukcji.

Można zatem podać wartość liczbową, logiczną, ciąg znaków lub pojedynczy znak.

Prawdziwą przewagę daje możliwość zastosowania typu zdefiniowanego przez programistę (struktury i klasy), a także wskaźników do ww.

Niewygody

Rzadko zdarzają się sytuacje, gdy szablony są niezbędne.

Składnia jest “niecodzienna” (stosowanie “ostrych” nawiasów).

Wbrew pozorom wymagają zdwojonej czujności programisty (np. dodawanie wartości w szablonie funkcji - lecz co gdy podamy wskaźniki?).

Nie “trawią” wszystkiego (praca z ciągami znaków jest skomplikowana).

Rozsądne korzystanie z szablonów...

...popłaca.

STL - Standard Template Library, wykorzystuje zalety programowania uogólnionego (kontenery: vektor, list; algorytmy: sort itd.)

Podsumowanie

```
template <typename/class typ_1> //prosty szablon klasy
class nazwa_klasy{
     //instrukcje
}

template <typename/class typ_2> //prosty szablon funkcji
void nazwa_funkcji(){
     //instrukcje
}

//wykorzystanie
nazwa_klasy<wybrany_typ> nazwa_obiektu();
nazwa_klasy<wybrany_typ> *nazwa_obiektu = nazwa_klasy<wybrany_typ>();
```

Zadanie...

...znajduje się na kartkach.