Szablony

- Szablony reprezentują funkcje, a nawet typy danych tworzone przez programistów (klasy)
  - Ale same nie są funkcjami ani klasami
- Nie zostają one zaimplementowane dla określonego typu danych, ponieważ zostanie on zdefiniowany później
  - W większości sytuacji parametryzowane są typem, ale nie jest to reguła
- Aby użyć szablonu kompilator lub programista musi określić dla jakiego typu ma on zostać użyty

Szablony klas wykorzystanie - tablica

std::array

Defined in header <array>

```
template<
    class T,
    std::size_t N
> struct array;
```
(since C++11)

https://en.cppreference.com/w/cpp/container/array

- Prosta tablica statyczna
  - Alokacja na stosie
  - Elementy określonego typu (możliwe konwersje)
  - Znany rozmiar czasie kompilacji
  - Zamiennik zwykłej tablicy
- Przykład cpp9.0a

Szablony klas wykorzystanie - wektor

std::vector

Defined in header <vector>

```
template<
    class T,
    class Allocator = std::allocator<T>
> class vector;                                                    (1)

namespace pmr {
    template <class T>
    using vector = std::vector<T, std::pmr::polymorphic_allocator<T>>;   (2)  (since C++17)
}
```

1) std::vector is a sequence container that encapsulates dynamic size arrays.

2) std::pmr::vector is an alias template that uses a polymorphic allocator

https://en.cppreference.com/w/cpp/container/vector

- Dynamiczna tablica
  - Alokacja pamięci na stercie
  - Elementy określonego typu (możliwe konwersje)
  - Ciągły obszar pamięci
  - Rozmiar rośnie w miarę potrzeb – UWAGA
- Przykład cpp9.0b

Szablony klas wykorzystanie - string

std::basic_string

Defined in header <string>

```
template<
    class CharT,
    class Traits = std::char_traits<CharT>,                    (1)
    class Allocator = std::allocator<CharT>
> class basic_string;
```

```
namespace pmr {
    template <class CharT, class Traits = std::char_traits<CharT>>
    using basic_string = std::basic_string< CharT, Traits,     (2)  (since C++17)
                                            std::polymorphic_allocator<CharT>>
}
```

- Dynamiczna tablica znaków
  - **`std::string`** to jest **`std::basic_string<char>`**
  - Alokacja pamięci na stercie
  - Elementy określonego typu **`char`**
  - Ciągły obszar pamięci
- Przykład cpp9.0c

Szablony funkcji wykorzystanie - find

std::find

Defined in header <algorithm>

```
template< class InputIt, class T >
constexpr InputIt find( InputIt first, InputIt last, const T& value );
```

https://en.cppreference.com/w/cpp/algorithm/find

- Algorytm do wyszukiwania
  - Obsługuje dowolne typy
  - Znajduje pierwszy element zgodny ze wzorcem
  - Przeszukuje podany zakres - nie musi być cały kontener
  - Są też inne wersje np. find_if
- Przykład cpp9.0d

Szablony funkcji wykorzystanie - sort

std::sort

```
template< class RandomIt >
constexpr void sort( RandomIt first, RandomIt last );
template< class RandomIt, class Compare >
void sort( RandomIt first, RandomIt last, Compare comp );
```

https://en.cppreference.com/w/cpp/algorithm/sort

- Algorytm do sortowania
  - Obsługuje dowolne typy
  - Domyślnie sortuje używając operatora <
  - W wersji drugiej potrafi użyć obiektu funkcyjnego służącego jako narzędzie do porównywania
- Przykład cpp9.0e

Szablony funkcji wykorzystanie – for_each

std::for_each

```
template< class InputIt, class UnaryFunction >
UnaryFunction for each( InputIt first, InputIt last, UnaryFunction f );
```

https://en.cppreference.com/w/cpp/algorithm/for_each

- Algorytm do wykonywania operacji na elementach
  - Stosowany zamienne z zakresową pętlą for
  - Działa w trybie tylko do odczytu albo modyfikowania
  - Przyjmuje jako argumenty zakres oraz funkcję/funktor
- Przykład cpp9.0f

Szablony i STL

- **`std::pair, std::make_pair`**
  - Class template argument deduction (C++17)
  - Przykład cpp9.0g
- **`std::tuple, std::make_tuple`**
  - Strucutral binding (C++17)
  - Przykład cpp9.0h
- **`std::ref, std::cref`**
  - Przykład cpp9.0g
- **`std::unique_ptr`**
  - Przykład cpp9.0i
- **`std::shared_ptr`**
  - Przykład cpp9.0j

Szablony i STL

- **`std::initializer_list`**
  - Przykład cpp9.0k
- **`std::advance`**, **`std::distance`**, **`std::next`** i **`std::prev`**
  - Przykład cpp9.0l
- **`std::less`** i **`std::greater`**
  - Przykład cpp9.0m
- **`std::bind`**
  - Przykład cpp9.0m
- I wiele, wiele innych ciekawych rzeczy