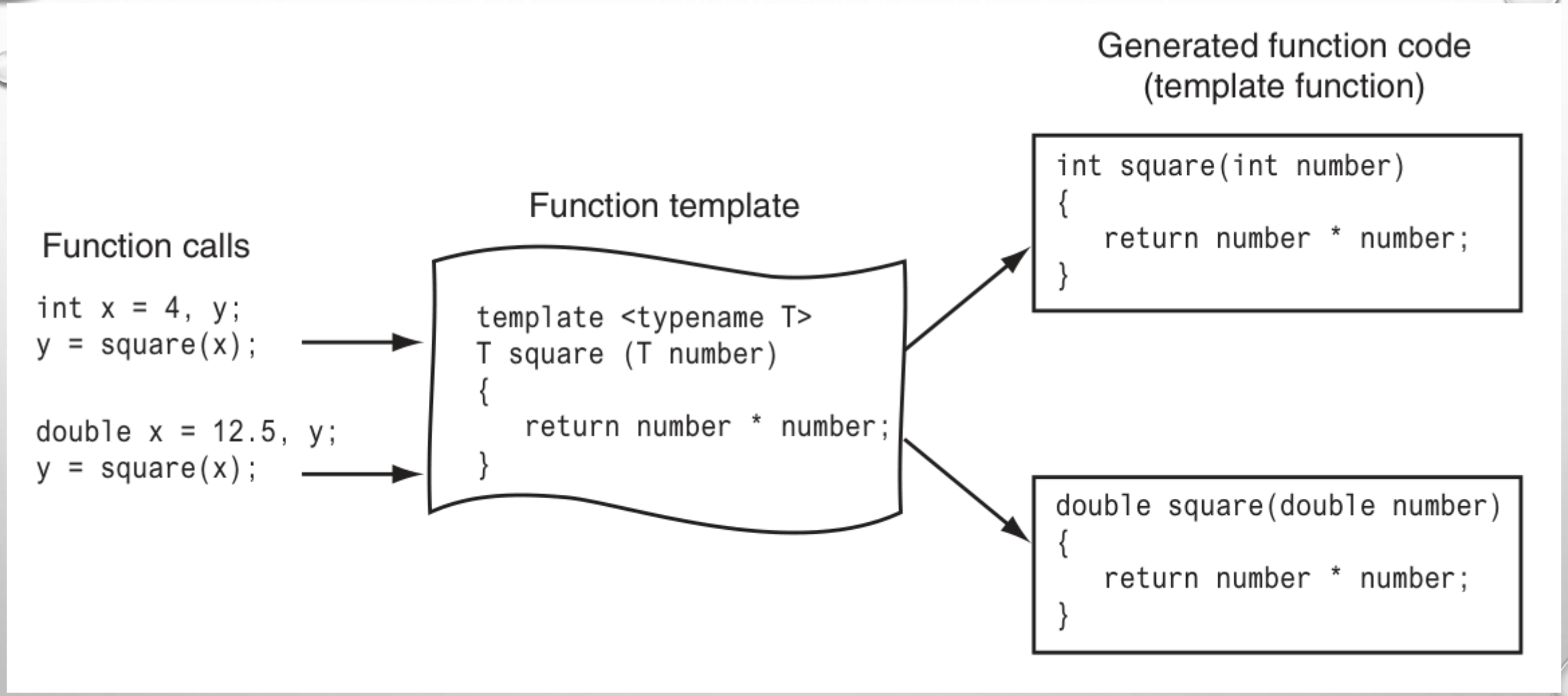
SZABLONY W C++

PRZYKŁAD MOTYWUJĄCY

```
int square(int number) {
    return number * number;
}

double square(double number) {
    return number * number;
}

int main()
{
    cout << square(2) << endl;
    cout << square(2.5) << endl;
}
```

SZABLON FUNKCJI

```
template <typename T>
T square(T number)
{
    return number * number;
}

int main()
{
    cout << square(2) << endl;
    cout << square(2.5) << endl;
}
```


Function calls
int x = 4, y;
y = square(x);
double x = 12.5, y;
y = square(x);
Function template
template <typename T>
T square (T number)
{
return number * number;
}
Generated function code
(template function)
int square(int number)
{
return number * number;
}
double square(double number)
{
return number * number;
}

UWAGA

Szablon funkcji to jedynie specyfikacja funkcji i sam w sobie nie powoduje użycia pamięci. Rzeczywista instancja funkcji jest tworzona w pamięci, gdy kompilator napotka wywołanie funkcji szablonu.

FUNKCJA SWAP

```
template <typename T>
void swapVars(T& var1, T& var2) {
    T temp = var1;
    var1 = var2;
    var2 = temp;
}

int main() {
    int x=7, y=13;
    swapVars(x, y);
    cout << x << " " << y << endl;
    string a="alfa", b="beta";
    swapVars(a, b);
    cout << a << " " << b << endl;
}
```

PRZECIĄŻANIE SZABLONÓW FUNKCJI

```
template <typename T>
T sum(T val1, T val2) {
    return val1 + val2;
}
template <typename T>
T sum(T val1, T val2, T val3) {
    return val1 + val2 + val3;
}

int main() {
    cout << sum(2, 3) << endl;
    cout << sum(1.2, 0.3, 0.1) << endl;
}
```

PROBLEMY

```
template <typename T>
T sum(T val1, T val2) {
    return val1 + val2;
}

int main()
{
    int x = 2;
    double y = 3.1;
    cout << sum(x, y) << endl;
}
```

[Error] no matching function for call to 'sum(int, double)'

ROZWIĄZANIE

```
int main()
{
    int x = 2;
    double y = 3.1;

    cout << sum<double>(x, y) << endl;

    cout << sum((double)x, y) << endl;

    cout << sum(static_cast<double>(x), y) << endl;

}
```

PAMIĘTAJMY, ŻE BIBLIOTEKA STANDARDOWA UŻYWA WZORCÓW

- PONIŻSZY KOD NIE SKOMPILUJE SIĘ

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

int main() {
    cout << max(3,4.1) << endl;
}
```

JAK NAPISAĆ FUNKCJĘ SZABLONOWĄ?

Zazwyczaj najpierw piszemy normalną funkcję,

a potem przekształcamy ją do szablonu funkcji.

WZORCE KLAS

- PRZYPOMNIJMY KLASĘ STACK

KLASA STACK

```
class Stack {
private:
    const static int MAX = 100; // maksymalny rozmiar stosu
    int st[MAX];
    int top;
public:
    Stack() {
        top = -1;
    }
    void push(int var) {
        st[++top] = var;
    }
    int pop() {
        return st[top--];
    }
};
```

SZABLON KLASY STACK

```
template <typename T>
class Stack {
private:
    const static int MAX = 100;
    T st[MAX];
    int top;
public:
    Stack() {
        top = -1;
    }
    void push(T var) {
        st[++top] = var;
    }
    T pop() {
        return st[top--];
    }
};
```

```
int main()
{
    Stack<int> s1;
    s1.push(3);
    cout << s1.pop() << endl;

    Stack<string> s2;
    s2.push("hello");
    cout << s2.pop() << endl;
}
```

UWAGA O DEFINIOWANIU METOD KLAS SZABLONOWYCH POZA DEKLARACJĄ KLAS

```
template <typename T>
class Stack {
private:
    const static int MAX = 100;
    T st[MAX];
    int top;
public:
    Stack() {
        top = -1;
    }
    void push(T var);
    T pop();
};
```

```
template <typename T>
void Stack<T>::push(T var) {
    st[++top] = var;
}

template <typename T>
T Stack<T>::pop() {
    return st[top--];
}
```

STANDARD TEMPLATE LIBRARY

- STANDARD TEMPLATE LIBRARY (STL) TO BIBLIOTEKA OPROGRAMOWANIA PIERWOTNIE ZAPROJEKTOWANA PRZEZ ALEKSANDRA STEPANOWA DLA JĘZYKA PROGRAMOWANIA C++
- OBECNIE JEST ZINTEGROWANA Z BIBLIOTEKĄ STANDARDOWĄ C++
- STL WPŁYNĘŁA NA WIELE ELEMENTÓW BIBLIOTEKI STANDARDOWEJ

DEMO STL

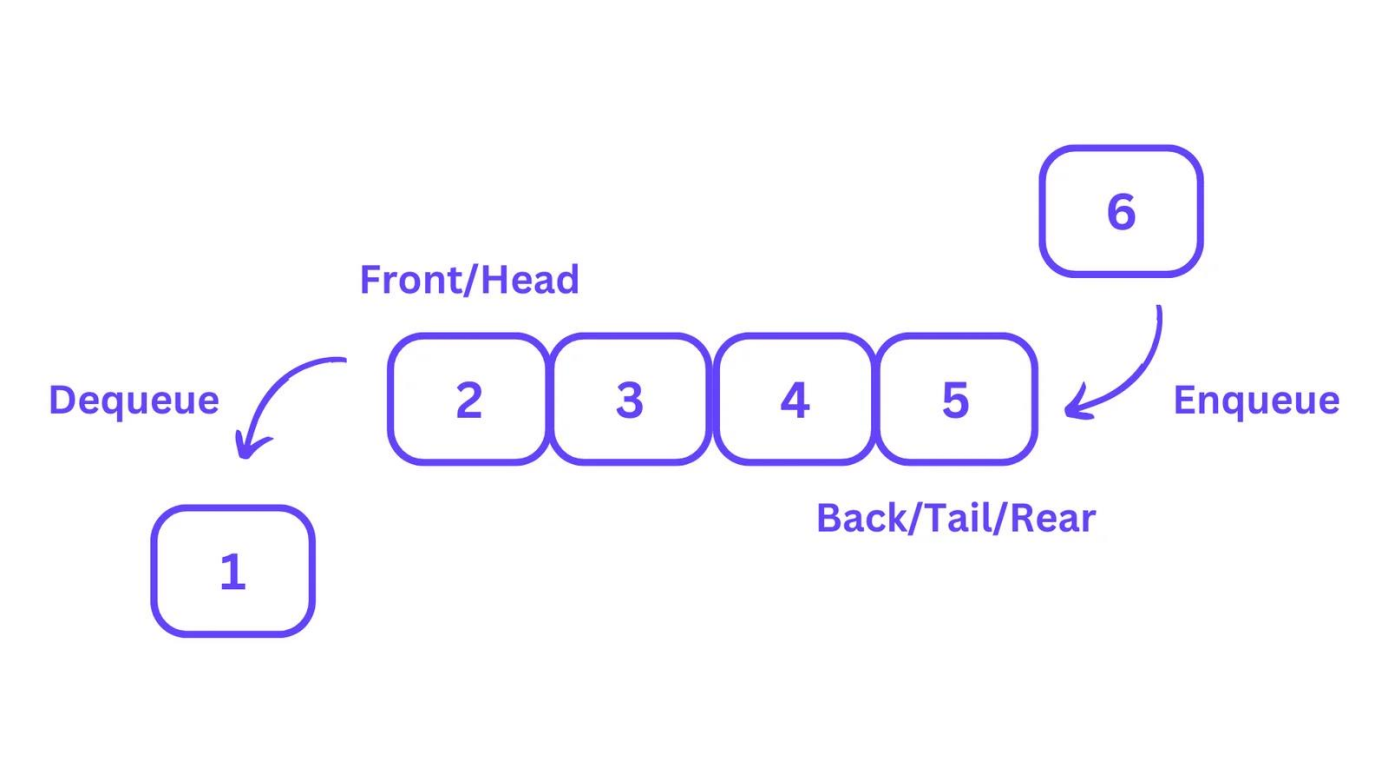
```cpp
#include <iostream>
#include <vector>
#include <utility>
#include <algorithm>
using namespace std;

int main() {
    vector<pair<int,int>> v{{5,2}, {3,4}, {4,6}};
    sort(v.begin(), v.end());
    for(auto p : v)
        cout << p.first << " " << p.second << endl;
}
```

PROGRAMOWANIE UOGÓLNIONE

- **PROGRAMOWANIE UOGÓLNIONE** (GENERYCZNE, Z ANG. *GENERIC PROGRAMMING*) – JEDEN Z PARADYGMATÓW PROGRAMOWANIA. PROGRAMOWANIE UOGÓLNIONE POZWALA NA PISANIE KODU PROGRAMU, W JĘZYKACH TYPOWANYCH STATYCZNIE, BEZ WCZEŚNIEJSZEJ ZNAJOMOŚCI TYPÓW DANYCH, NA KTÓRYCH KOD TEN BĘDZIE PRACOWAŁ.
- W JĘZYKU C++ PROGRAMOWANIE UOGÓLNIONE UMOŻLIWIAJĄ **SZABLONY**.

https://pl.wikipedia.org/wiki/Programowanie_uog%C3%B3lnione

KOLEJKA - QUEUE



https://learnloner.com/queues-in-data-structures-and-algorithms-dsa/

ZADANIE

Zdefiniuj klasę szablonową Queue. Metody zdefiniuj na zewnątrz klasy.

W programie utwórz i przetestuj Queue<int> oraz Queue<string>

WSKAZÓWKA:

https://en.wikipedia.org/wiki/Circular_buffer

KONIEC