# Kolejka piorytetowa

Kolejka priorytetowa różni się od zwykłej kolejki tym, że elementy są usuwane według ich priorytetów, a nie w kolejności dodania.

Standardowa biblioteka C++ dostarcza implementację kolejki priorytetowej w postaci klasy std::priority_queue.

## Przykładowa implementacja kolejki priorytetowej z wykorzystaniem biblioteki STL:

```
#include <iostream>
#include <queue>

int main() {
    std::priority_queue<int> kolejka;

    kolejka.push(3);
    kolejka.push(1);
    kolejka.push(2);

    while (!kolejka.empty()) {
        std::cout << kolejka.top() << " ";
        kolejka.pop();
    }

    return 0;
}
```

## Implemetnacja obiektowa:

```
// Definicja struktury węzła kolejki
struct Node {
    int data;
    int priority; // Priorytet elementu
    Node* next;

    Node(int d, int p) : data(d), priority(p), next(nullptr) {} // Konstruktor
węzła
};
```

## Zadanie

Zamienić zwykłą kolejkę na kolejkę priorytetową we wcześniejszym kodzie:

1. Dodanie pola priorytetu do struktury węzła:
    a. Do struktury Node dodaj pole int priority, które będzie przechowywać priorytet elementu.
2. Modyfikacja funkcji enqueue:
    a. W funkcji enqueue, zamiast dodawać element na koniec kolejki, elementy muszą być dodawane w odpowiedniej kolejności według ich priorytetów.
    b. Przeanalizuj, jak można modyfikować obecną funkcję enqueue tak, aby uwzględniała priorytety.

    c. Pamiętaj, że elementy o niższym priorytecie powinny być dodawane na końcu kolejki.
3. Modyfikacja funkcji dequeue:
    a. W funkcji dequeue, zamiast usuwać zawsze pierwszy element kolejki, usuń element o najwyższym priorytecie.
    b. Zastanów się, jak modyfikować funkcję dequeue, aby usuwała element o najwyższym priorytecie.
4. Testowanie implementacji:
    a. Po dokonaniu powyższych zmian, przetestuj swoją implementację, dodając i usuwając elementy z różnymi priorytetami.
    b. Upewnij się, że elementy są dodawane i usuwane zgodnie z ich priorytetami, a także że kolejka zachowuje się zgodnie z oczekiwaniami.