



**AKADEMIA GÓRNICZO-HUTNICZA**

Dokumentacja do projektu

**Biblioteka do obsługi kolejki  
priorytetowej**

z przedmiotu

**Języki programowania obiektowego**

EiT rok III

*Krzysztof Podoba*

Piątek 11:30

prowadzący: Jakub Zimnol

30.12.2024

# 1. Opis projektu

Tematem projektu jest biblioteka w języku C++ do obsługi kolejki priorytetowej z ograniczonym rozmiarem. Kolejka priorytetowa umożliwia przechowywanie elementów w uporządkowany sposób według ich priorytetów. Dodatkowo, projekt rozszerza funkcjonalność, umożliwiając określenie maksymalnej liczby elementów w kolejce. W przypadku przekroczenia tego limitu, elementy o niższym priorytecie są zastępowane przez te o wyższym. Sortowanie elementów w kolejce odbywa się za pomocą `std::sort`.

Kolejki priorytetowe znajdują zastosowanie w systemach operacyjnych, zarządzaniu zasobami czy grach.

## 2. Struktura projektu

### Zawartość **PriorityQueue.h**:

- Abstrakcyjna klasa **PriorityQueue** – definiuje interfejs do zarządzania elementami w kolejce priorytetowej. Obejmuje metody takie jak:
  - `virtual void insert(int priority, const T& value) = 0` – deklaracja wirtualnej metody do dodawania elementu do kolejki
  - `virtual T pop() = 0` – deklaracja wirtualnej metody do usuwania elementu o najwyższym priorytecie
  - `void printQueue() const` – metoda wypisująca stan kolejki, zaimplementowana w klasie bazowej
  - `bool isEmpty() const` – sprawdza, czy kolejka jest pusta
  - `size_t size() const` – metoda zwracająca liczbę elementów w kolejce
  - `static bool compareNodes(const Node<T>& a, const Node<T>& b)` – statyczna metoda porównująca dwa węzły kolejki
- Klasa **Node** – reprezentuje węzeł kolejki, przechowuje informacje o priorytecie, wartości i identyfikatorze elementu, zawiera przeciążone operatory, które pozwalają na bardziej intuicyjne porównywanie i manipulowanie węzłami (operator<, operator==, operator<<)

Klasa **PriorityQueue** oraz **Node** wykorzystują szablony, co pozwala na obsługę dowolnego typu danych, np. `int`, `std::string`. Zapewnia to uniwersalność i elastyczność.

### Zawartość **BoundedPriorityQueue.h**:

- Klasa **BoundedPriorityQueue** – dziedziczy po **PriorityQueue**, umożliwia ograniczenie rozmiaru kolejki. Jej kluczowe funkcje to: dodawanie elementów z priorytetami (z automatycznym zarządzaniem miejscem w kolejce), obsługa przepełnienia kolejki, zmiana maksymalnego rozmiaru kolejki
  - `void insert(int priority, const T& value)` – dodaje element do kolejki, jeśli kolejka osiągnie maksymalny rozmiar, element o najniższym priorytecie zostaje usunięty, aby zrobić miejsce dla nowego elementu o wyższym priorytecie
  - `T pop()` – usuwa i zwraca element o najwyższym priorytecie

- `void setMaxSize(size_t newSize)` – ustawia maksymalny rozmiar kolejki, jeśli aktualny rozmiar kolejki przekracza nowy limit, elementy o najniższym priorytecie są usuwane
- `bool contains(int priority, const T& value) const` - sprawdza, czy w kolejce znajduje się element o danym priorytecie i wartości

**BoundedPriorityQueue** również wykorzystuje szablony, umożliwiając przechowywanie dowolnych typów danych.

### Zawartość **main.cpp**:

- Funkcja główna **main** – pokazuje działanie kolejki priorytetowej. Prezentuje funkcjonalności takie jak: dodawanie elementów do kolejki, zarządzanie elementami o różnych priorytetach, obsługę pustej kolejki, wykorzystanie różnych typów danych w kolejce

## 3. Kompilacja oraz uruchomienie

Aplikację można skompilować bezpośrednio za pomocą kompilatora **g++**. Poniżej przykładowa komenda:

```
g++ -o main main.cpp
```

Pliki nagłówkowe (**PriorityQueue.h**, **BoundedPriorityQueue.h**) muszą znajdować się w tym samym katalogu co **main.cpp**.

Po kompilacji program uruchamia się poleceniem:

```
./main
```

## 4. Podsumowanie

Projekt prezentuje zastosowanie programowania obiektowego w języku C++. Dzięki wykorzystaniu szablonów, interfejsów i dziedziczenia, rozwiązanie jest elastyczne i łatwe do adaptacji. Kolejka priorytetowa z ograniczonym rozmiarem może znaleźć zastosowanie w wielu scenariuszach, szczególnie w systemach o ograniczonych zasobach.