

# AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych  
Katedra Informatyki

## DOKUMENTACJA PROJEKTOWA ZAAWANSOWANE PROGRAMOWANIE

### **...Algorytm listy dwukierunkowej z zastosowaniem GitHub...**

Autor:  
Mateusz Smaga

Prowadzący:  
mgr inż. Dawid Kotlarski

Nowy Sącz 2024

# Spis treści

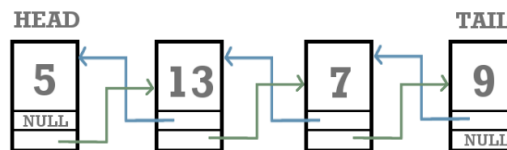
<b>1. Ogólne określenie wymagań</b>	<b>3</b>
1.1. Lista dwukierunkowa, a jednokierunkowa . . . . .	3
<b>2. Analiza problemu</b>	<b>4</b>
2.1. Użyteczność algorytmów list dwukierunkowych . . . . .	4
<b>3. Projektowanie</b>	<b>5</b>
3.1. Właściwości ogólne konfiguracji projektu . . . . .	5
3.2. Język programowania . . . . .	6
3.3. Git . . . . .	6
3.4. Doxygen . . . . .	6
<b>4. Implementacja</b>	<b>8</b>
4.1. Implementacja klasy DoublyLinkedList . . . . .	8
4.2. Przykład implementacji metody show . . . . .	9
4.3. Wynik działania wszystkich metod zawartych w klasie . . . . .	9
<b>5. Wnioski</b>	<b>11</b>
5.1. Wnioski . . . . .	11
<b>Literatura</b>	<b>12</b>
<b>Spis rysunków</b>	<b>12</b>
<b>Spis tabel</b>	<b>13</b>
<b>Spis listingów</b>	<b>14</b>

## 1. Ogólne określenie wymagań

Celem projektu jest stworzenie implementacji listy dwukierunkowej działającej na stercie w języku C++. Program ma być zrealizowany w formie klasy, która będzie oferowała określone funkcje do manipulacji listą. Działanie klasy zostanie przetestowane w funkcji main.

### 1.1. Lista dwukierunkowa, a jednokierunkowa

Polega na sposobie poruszania się na strukturze. W prostszej wersji listy poruszamy się tylko w jednym kierunku. Od początku do końca. Od głowy do ogona. Lista dwukierunkowa umożliwia zmianę azymutu na odwrotny, tzn. teraz możemy przechodzić po elementach od tyłu i cofać się w obrębie listy. Znika więc ograniczenie, które pojawiało się przy pierwszej z list. Mianowicie, gdy chcemy dostać się do np. przedostatniego elementu to nie musimy zaczynać przechodząc od początku listy. Możemy zacząć od jej końca, co pozwoli oszczędzić wiele kroków.[legierski]..



**Rys. 1.1.** Listy dwukierunkowe

Jak widać każdy węzeł ma teraz więcej niż jeden wskaźnik. Teraz wskazuje nie tylko na następny element, ale również na poprzedni.

## 2. Analiza problemu

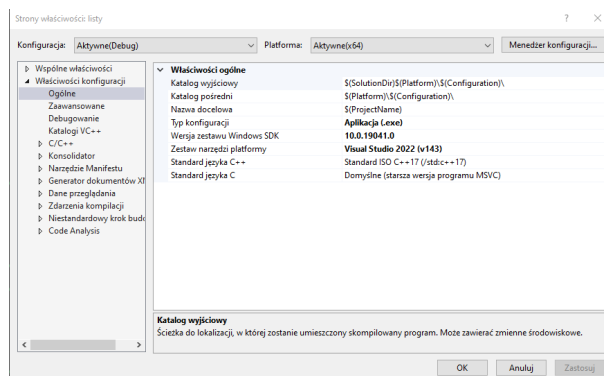
### 2.1. Użyteczność algorytmów list dwukierunkowych

Algorytmy związane z listami dwukierunkowymi (doubly linked lists) są bardzo użyteczne w różnych zastosowaniach, ponieważ oferują elastyczność i efektywność w operacjach związanych z wstawianiem, usuwaniem i przeszukiwaniem elementów w obu kierunkach. Oto kilka przypadków, w których mogą być szczególnie przydatne:

1. **Efektywność operacji wstawiania i usuwania:** Listy dwukierunkowe umożliwiają szybkie wstawianie i usuwanie elementów bez konieczności przesuwania innych elementów.
2. **Przeszukiwanie w obu kierunkach:** Wskaźniki na poprzedni i następny element pozwalają na swobodne poruszanie się w obu kierunkach po strukturze danych.
3. **Dynamiczne zarządzanie pamięcią:** Listy dwukierunkowe umożliwiają dodawanie i usuwanie elementów bez konieczności zmiany rozmiaru struktury.
4. **Mechanizmy cofania i ponawiania:** W aplikacjach typu cofnij/ponów, np. w edytorach, listy dwukierunkowe ułatwiają zarządzanie historią zmian.
5. **Implementacja deque:** Listy dwukierunkowe doskonale nadają się do implementacji podwójnych kolejek, gdzie można dodawać i usuwać elementy z obu końców.

## 3. Projektowanie

### 3.1. Właściwości ogólne konfiguracji projektu



Rys. 3.1. Właściwości konfiguracji

To okno pokazuje **właściwości konfiguracji projektu** w Visual Studio 2022, które zawierają kluczowe ustawienia kompilacji. Oto, co można zaobserwować w tym oknie:

- **Konfiguracja i Platforma:** Ustawione są odpowiednio na *Aktywne (Debug)* i *Aktywne (x64)*, co oznacza, że projekt będzie kompilowany w trybie debugowania na architekturze 64-bitowej.
- **Katalogi Wyjściowy i Pośredni:** Ścieżki do katalogów, gdzie Visual Studio umieszcza skompilowane pliki wyjściowe oraz pliki tymczasowe używane podczas kompilacji.
- **Nazwa docelowa i Typ konfiguracji:** Typ docelowego pliku to aplikacja (.exe).
- **Wersja zestawu Windows SDK:** Używana wersja SDK to 10.0.19041.0, co jest wymagana do kompilacji aplikacji działających na systemie Windows.
- **Zestaw narzędzi platformy:** Używany zestaw narzędzi to Visual Studio 2022 (v143), co wskazuje na wersję kompilatora MSVC dostarczaną z Visual Studio 2022.
- **Standardy języków C++ i C:** Używany standard C++ to *ISO C++17*, co oznacza, że projekt jest kompilowany z użyciem funkcji standardu C++17.

### 3.2. Język programowania

Wybór języka C++ do implementacji list dwukierunkowych jest korzystny z kilku powodów:

- **Efektywne zarządzanie pamięcią:** Dzięki wskaźnikom i operatorom `new` oraz `delete`, C++ umożliwia kontrolowaną alokację pamięci, co jest kluczowe w listach dwukierunkowych.
- **Programowanie obiektowe:** C++ pozwala na modelowanie listy jako klasy `Node` oraz `DoublyLinkedList`, co zwiększa przejrzystość kodu.
- **Bezpośrednia obsługa wskaźników:** C++ oferuje kontrolę nad wskaźnikami, co jest niezbędne przy implementacji węzłów z wskaźnikami na poprzedni i następny element.
- **Wydajność:** Jako język kompilowany, C++ zapewnia wysoką szybkość działania, co jest istotne przy operacjach na listach.
- **Standardowa biblioteka:** C++ posiada STL, która zawiera kontenery takie jak `std::list` do implementacji list dwukierunkowych.

### 3.3. Git

Podczas implementacji list dwukierunkowych korzystałem z Gita, aby:

- **Śledzić zmiany:** Git pozwala mi na kontrolę wersji kodu, dzięki czemu mogę analizować historię modyfikacji.
- **Bezpiecznie eksperymentować:** Możliwość tworzenia gałęzi pozwala testować nowe funkcje bez ryzyka utraty stabilnej wersji projektu.
- **Współpracować i udostępniać:** Git ułatwia współdzielenie kodu i ewentualną współpracę z innymi programistami.

Dzięki Gitowi zarządzanie kodem stało się bardziej efektywne i bezpieczne.

### 3.4. Doxygen

Warto korzystać z Doxygen, ponieważ:

- **Automatyczna dokumentacja:** Doxygen generuje dokumentację bezpośrednio z kodu, co oszczędza czas i zapewnia aktualność.

- **Czytelność kodu:** Dzięki komentarzom w stylu Doxygen kod staje się bardziej przejrzysty dla innych programistów.
- **Wsparcie dla wielu języków:** Doxygen obsługuje popularne języki, takie jak C++ i Python, co czyni go uniwersalnym narzędziem.

Doxygen usprawnia tworzenie dokumentacji i poprawia czytelność projektu.

## 4. Implementacja

### 4.1. Implementacja klasy DoublyLinkedList

```
1 template <class T>
2 class DoublyLinkedList {
3 private:
4     Node<T>* head = nullptr;
5     Node<T>* tail = nullptr;
6     int counter = 0;
7
8 public:
9     DoublyLinkedList() = default;
10    void show() const;
11    void insert_head(const T& value);
12    void insert_tail(const T& value);
13    void set_index(const T& value, int index);
14    void remove_head();
15    void remove_tail();
16    void show_backward() const;
17    void goNext(int index);
18    void goPrevious(int index);
19 };
```

#### Podstawowe metody klasy:

- `void show() const;`  
Wyświetla wszystkie elementy listy od początku do końca.
- `void insert_head(const T& value);`  
Dodaje element na początek listy.
- `void insert_tail(const T& value);`  
Dodaje element na koniec listy.
- `void set_index(const T& value, int index);`  
Dodaje element na wybranej pozycji w liście (jeśli indeks jest prawidłowy).
- `void remove_head();`  
Usuwa element z początku listy.
- `void remove_tail();`  
Usuwa element z końca listy.



- `void show_backward() const;`  
Wyświetla elementy listy w odwrotnej kolejności.
- `void goNext(int index);`  
Przechodzi do następnego elementu po określonym indeksie.
- `void goPrevious(int index);`  
Przechodzi do poprzedniego elementu przed określonym indeksem.

## 4.2. Przykład implementacji metody show

Metoda `show` służy do wyświetlania zawartości listy dwukierunkowej, zaczynając od początku listy (od węzła `head`) i przechodząc aż do końca (do węzła `tail`). Dzięki tej metodzie możemy zobaczyć kolejność elementów znajdujących się w liście.

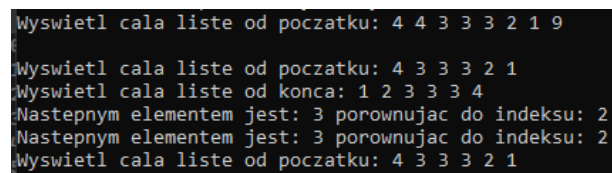
```
1 /**
2  * @brief Wyświetla wszystkie elementy listy od początku do końca.
3  */
4 template <class T>
5 void DoublyLinkedList<T>::show() const {
6     Node<T>* current = head; // Zaczynamy od pierwszego elementu
7     cout << "Wyświetl całą listę od początku: ";
8     while (current != nullptr) { // Przechodzimy przez całą listę
9         cout << current->value << " "; // Wyświetlamy wartość
        bieżącego elementu
10        current = current->next; // Przechodzimy do następnego
        elementu
11    }
12    cout << endl;
13 }
14 };
```

## 4.3. Wynik działania wszystkich metod zawartych w klasie

```
1 int main() {
2     DoublyLinkedList<int> lista;
3     lista.insert_head(1);
4     lista.insert_head(2);
5     lista.insert_head(3);
6     lista.insert_head(4);
7     lista.insert_head(4);
8     lista.insert_tail(9);
9     lista.set_index(3, 2);
```

```
10     lista.set_index(3, 3);
11     cout << "\t---Stan początkowy listy---" << endl;
12     lista.show(); cout << endl;
13     lista.remove_tail();
14     lista.remove_head();
15     lista.show();
16     lista.show_backward();
17     lista.goNext(2);
18     lista.goPrevious(2);
19     lista.show();
20
21     return 0;
22 }
23 };
```

Wynik działania powyższego kodu:



```
Wyswietl cala liste od poczatku: 4 4 3 3 3 2 1 9
Wyswietl cala liste od poczatku: 4 3 3 3 2 1
Wyswietl cala liste od konca: 1 2 3 3 3 4
Nastepnym elementem jest: 3 porownujac do indeksu: 2
Nastepnym elementem jest: 3 porownujac do indeksu: 2
Wyswietl cala liste od poczatku: 4 3 3 3 2 1
```

**Rys. 4.1.** Wynik działania programu

## 5. Wnioski

### 5.1. Wnioski

Implementacja listy dwukierunkowej w C++ prowadzi do kilku istotnych wniosków:

- **Elastyczne zarządzanie pamięcią:** Lista dwukierunkowa jest strukturą dynamiczną, która pozwala na szybkie dodawanie i usuwanie elementów bez potrzeby przesuwania pozostałych danych, co czyni ją bardziej elastyczną niż tablica.
- **Efektywność operacji:** Wstawianie i usuwanie elementów z początku lub końca listy jest operacją o stałej złożoności  $O(1)$ , co jest korzystne w porównaniu do innych struktur danych.
- **Zarządzanie wskaźnikami:** Implementacja wymaga znajomości wskaźników, gdyż każdy element wskazuje zarówno na poprzedni, jak i na następny element. Praca ze wskaźnikami w listach dwukierunkowych uczy zarządzania pamięcią, pomagając zrozumieć potencjalne błędy, jak 'nullptr' czy wiszące wskaźniki.

## Spis rysunków

1.1. Listy dwukierunkowe . . . . .	3
3.1. Wlasciwosci konfiguracji . . . . .	5
4.1. Wynik działania programu . . . . .	10

## **Spis tabel**

## **Spis listingów**