Sprawozdanie 08: Obiegi drzew i przykłady ich zastosowania

Dmytro Sakharskyi - L02 - 31259 5.06.2025

Metody Programowania

Informatyka 1 rok. 2 semestr

Wstęp

Obiegi drzewa to podstawowe operacje wykonywane na strukturze drzewiastej, umożliwiające odwiedzanie wszystkich jej węzłów w określonej kolejności. Wyróżniamy różne typy obiegów, spośród których najczęściej stosowane to:

- **in-order (LPR)** – lewe poddrzewo, korzeń, prawe poddrzewo,
- **pre-order (PLR)** – korzeń, lewe poddrzewo, prawe poddrzewo.

W przypadku drzew binarnych kolejność ta jest jasno zdefiniowana, natomiast dla drzew dowolnego rzędu **(n-arnych)** konieczne jest odpowiednie przystosowanie algorytmu.

Celem niniejszego ćwiczenia było:

- zrozumienie zasad działania obiegów in-order oraz pre-order,
- implementacja tych obiegów w kontekście drzewa n-arnego,
- integracja ich z istniejącym kodem z wcześniejszego zadania,
- a także porównanie wyników z przykładami literaturowymi i graficzne przedstawienie struktury drzewa oraz obiegów.

Zadanie 1

```
void preorder(const shared_ptr<Node>& node) {
    if (!node) return;
    cout << node->data << " ";
    for (const auto& child : node->children) {
        preorder(child);
    }
}
```

Funkcja **preorder** odwiedza najpierw bieżący węzeł, a następnie rekurencyjnie wszystkich jego potomków w kolejności od lewej do prawej.

```
void inorder(const shared_ptr<Node>& node) {
    if (!node) return;
    size_t n = node->children.size();
    for (size_t i = 0; i < n / 2; ++i) {
        inorder(node->children[i]);
    }

    cout << node->data << " ";

    for (size_t i = n / 2; i < n; ++i) {
        inorder(node->children[i]);
    }
}
```

Funkcja **inorder** została zaadaptowana do drzewa n-arnego. W tym przypadku najpierw odwiedzana jest połowa dzieci, następnie bieżący węzeł, a potem pozostała część dzieci. Jest to rozszerzenie klasycznego podejścia stosowanego w drzewach binarnych (**LPR**) i zostało dobrane w sposób logiczny i symetryczny.

Zadanie 2

```
struct Node {
    string data;
    vector<shared_ptr<Node>> children;

    Node(const string& d) : data(d) {}
};
```

Funkcje **preorder**() oraz **inorder**() zostały dostosowane do struktury Node używanej w poprzednim zadaniu. Nie zmieniano typów danych.

- Użyto wskaźników typu **shared_ptr**<Node>, zgodnie z poprzednim kodem.
- Pole children to **vector**<**shared_ptr**<Node>>, co pozwala na przechowywanie wielu dzieci.
- Parametry funkcji mają typ **const shared_ptr**<Node>&, co zapewnia kompatybilność z resztą programu.

Dzięki temu implementacja działa poprawnie i nie narusza definicji struktury Node.

Zadanie 3

Zgodnie z poleceniem, struktura **Node** używana w zadaniu 7 nie została zmieniona. Obie funkcje — **preorder()** oraz **inorder()** — zostały dostosowane do jej oryginalnej definicji:

```
struct Node {
    string data;
    vector<shared_ptr<Node>> children;

    Node(const string& d) : data(d) {}
};
```

Funkcje obiegu drzewa pracują bezpośrednio na vector<shared_ptr<Node>>, co umożliwia obsługę drzewa dowolnego rzędu i spełnia wymagania zadania.

Zadanie 4

Drzewo:

```
--- Struktura drzewa ---
Prezes Sadu Rejonowego
 Samodzielna Sekcja Administracyjna
 I Zespol Kuratorskiej Sluzby Sadowej
 Wydzialy
  I Wydzial Cywilny
  II Wydzial Karny
  III Wydzial Rodzinny i Nieletnich
  IV Wydzial Ksiag Wieczystych
```

Po poprawnej implementacji funkcji preorder() oraz inorder(), obiegi drzewa zostały przedstawione na przykładzie struktury organizacyjnej z poprzedniego zadania.

Preorder:

```
--- Obieg preorder (PLR) ---
Prezes Sadu Rejonowego
Samodzielna Sekcja Administracyjna
I Zespol Kuratorskiej Sluzby Sadowej
Wydzialy
I Wydzial Cywilny
II Wydzial Karny
III Wydzial Rodzinny i Nieletnich
IV Wydzial Ksiag Wieczystych
```

Funkcja **preorder**() najpierw odwiedza bieżący węzeł, a następnie jego dzieci — od lewej do prawej.

Inorder:

```
--- Obieg inorder (LPR) ---
Samodzielna Sekcja Administracyjna
Prezes Sadu Rejonowego
I Zespol Kuratorskiej Sluzby Sadowej
I Wydzial Cywilny
II Wydzial Karny
Wydzialy
III Wydzial Rodzinny i Nieletnich
IV Wydzial Ksiag Wieczystych
```

Funkcja **inorder**() najpierw odwiedzana jest pierwsza część dzieci, następnie bieżący węzeł, a potem pozostałe dzieci (adaptacja do drzewa n-arnego).

Zadanie 5

W celu sprawdzenia poprawności działania funkcji **preorder**() oraz **inorder**(), uzyskane wyniki porównano z przykładami zamieszczonymi w literaturze oraz źródłach internetowych, takich jak GeeksForGeeks i Wikipedia.

- ***Obieg preorder (PLR):***

W funkcji **preorder**() najpierw odwiedzany jest bieżący węzeł, a następnie wszystkie jego dzieci w kolejności od lewej do prawej.

Otrzymany wynik:

```
--- Obieg inorder (LPR) ---
Samodzielna Sekcja Administracyjna
Prezes Sadu Rejonowego
I Zespol Kuratorskiej Sluzby Sadowej
I Wydzial Cywilny
II Wydzial Karny
Wydzialy
III Wydzial Rodzinny i Nieletnich
IV Wydzial Ksiag Wieczystych
```

Wynik ten jest zgodny z definicją obiegu preorder i potwierdza poprawność implementacji.

- ***Obieg inorder (LPR):***

Dla drzew n-arnych klasyczna definicja **in-order** nie istnieje, dlatego przyjęto rozszerzoną wersję, gdzie:

1. najpierw odwiedzana jest **pierwsza część dzieci**,
2. potem bieżący węzeł,
3. następnie **reszta dzieci**.

Otrzymany wynik:

```
--- Obieg inorder (LPR) ---
Samodzielna Sekcja Administracyjna
Prezes Sadu Rejonowego
I Zespol Kuratorskiej Sluzby Sadowej
I Wydzial Cywilny
II Wydzial Karny
Wydzialy
III Wydzial Rodzinny i Nieletnich
IV Wydzial Ksiag Wieczystych
```

Wynik jest zgodny z przyjętą logiką obiegu LPR dla drzew dowolnego rzędu.

Wnioski

W ramach ćwiczenia zaimplementowano dwa typy obiegu drzewa: **pre-order** oraz **in-order**, dostosowane do struktury drzewa dowolnego rzędu (n-arnego). Funkcje te zostały zintegrowane z wcześniej przygotowaną strukturą Node i działają zgodnie z jej logiką, bez konieczności wprowadzania zmian w definicji danych.

Dzięki implementacji oraz testom praktycznym:

- potwierdzono poprawność działania obu algorytmów,
- zaprezentowano wyniki obiegów w postaci tekstowej i graficznej,
- porównano otrzymane rezultaty z przykładami z zewnętrznych źródeł.

Zrealizowane zadania pozwoliły utrwalić różnice między sposobami przetwarzania drzewa oraz wykazać, że możliwe jest ich zastosowanie również w przypadku struktur o zmiennej liczbie dzieci.