ZADAĆA 7

Objektno orijentirano programiranje

Upute:

Zadaću je potrebno predati do 5. svibnja u 08:00 na Teamsu. Diskutiranje zadaće u okviru "study group" je dozvoljen, ali predana zadaća mora biti samostalno riješena. Studenti koji predaju zadaću obavezni su prisustvovati vježbama, u suprotnome zadaća neće biti bodovana. Ako se kod ne prevodi (neovisno o tipu compiler-a), ili se događa greška prilikom izvršavanja koda, zadaća neće biti bodovana.

Zadatak 1. (40 bodova) Nadogradnja binarnog stabla

Na predavanjima ste implementirali binarno stablo. U ovom zadatku potrebno ga je nadograditi sa sljedećim:

- `BinaryTree`(`const` BinaryTree&) – konstruktor kopiranja (deep-copy),
- BinaryTree& `operator=`(`const` BinaryTree&) – preopteretite operator pridruživanja,
- NodeList `BFS`(Node*) – metoda pretraživanja po širini BFS (breadth-first-search).

Ideja BFS-a je sljedeća:

Počevši od nekog korijena stabla s (ili korijena podstabla), sustavno istražujemo sve čvorove koji su susjedi od s (lijevo i desno dijete). Kako smo istražili sve susjede od s, počinjemo pretraživati sve susjede od tih susjeda, itd. dok ne pregledamo sve čvorove u stablu. Drugim riječima, počevši od korijena stabla (podstabla) otkrivamo sve čvorove na dubini d, prije nego li otkrijemo sve čvorove na dubini d+1.

Takav način obilaska može biti implementiran uz pomoć reda – FIFO strukture podataka (first-in, first-out). FIFO struktura podataka implementira operacije enqueue ("postavljanje elementa na kraj reda") i dequeue ("izvlačenje elementa s početka reda"). Možete implementirati vlastitu FIFO strukturu podataka ili koristiti postojeću (npr. std::queue ili std::deque).

Unutar maina pozovite operator pridruživanja, konstruktor kopiranja i BFS.

Zadatak 2. (60 bodova) N-arno stablo

Poopćite implementaciju Binarnog stable na n-arno stablo u klasi `NAryTree`. Čvor u binarnom stablu ima lijevo i desno dijete, dok u slučaju n-arnog stabla, čvor ima proizvoljan broj djece.

Deklaracija čvora u n-arnom stablu je sljedeća:

```
class Node{
protected:
    Elem _elt;
    Node* _par;
    list<Node*> _children;

public:
    Node(Elem);
    Node(const Node&);

    list<Node*>& children();
    Node* parent() const;
    Elem element() const;

    void setParent(Node*);

    void prependChild(Node*);
    void appendChild(Node*);

    void removeFirstChild();
    void removeLastChild();

    void setValue(Elem);

    bool isRoot() const;
    bool isLeaf() const;
};
```

Nakon implementacije klase `Node`, odnosno definiranja metoda u klasi `Node`, poopćite odgovarajuće članove iz klase `BinaryTree` u `NAryTree`, prema sljedećoj deklaraciji:

```
class NAryTree {
public:
    Node* _root;

public:
    NAryTree();
    NAryTree(const NAryTree&);
    ~NAryTree();

    Node* root() const;
    void addRoot(Elem);

    void appendChild(Node*, Elem);
    void prependChild(Node*, Elem);
```

```
    void preorderPrint(Node* ) const;
    void postorderPrint(Node* ) const;

    void preorderNodes(Node*, NodeList&); //radi preorder obilazak i sprema
                                          Node-ove u proslijeđeni NodeList

    NodeList BFS(Node*);

    NAryTree& operator=(const NAryTree&);

};
```

U main dijelu instancirajte barem jedno n-arno stablo i isprobajte implementirane metode.