Pismeni ispit

Objektno orijentirano programiranje

4. srpnja 2019.

Upute

Pismeni se ispit piše 120 minuta. Studenti smiju koristiti isključivo materijale s predavanja i vježbi, C++ referencu te Python dokumentaciju. Niti jedan drugi oblik komunikacije nije dopušten te će se svako nepridržavanje pravila kažnjavati udaljavanjem s ispita i prijavom uredu pročelnika. Nakon što završite s pisanjem ispita, rješenja smjestite po zadacima u mapu `PREZIME_IME_PISMENI4`, tu mapu zipajte u datoteku `PREZIME_IME_PISMENI4.zip` te ju pošaljite na oop@mathos.hr s naslovom e-mail poruke: "PREZIME IME: OOP PISMENI 4".

Zadatak 1. (C++) [35 bodova]

Klase `Shape`, `Rectangle`, `Square` i `Circle` zadane su sljedećim deklaracijama:

```
class Shape {
private:
    double area() const;
};

class Rectangle {
protected:
    double a, b;
public:
    Rectangle(double, double);
    Rectangle(const Rectangle&);
    ~Rectangle();
    double area() const;
};

class Square {
public:
    Square(double);
    Square(const Square&);
    ~Square();
};
```

```
class Circle {
protected:
    double r;
public:
    Circle(double);
    Circle(const Circle&);
    ~Circle();
    double area() const;
};
```

Preinačite ove klase tako da:

- nije moguće instancirati klasu `Shape`
- je uspostavljena odgovarajuća hijerarhija među klasama (npr. klasa `Square` nasljeđuje `Rectangle` itd.)
- nisu enkapsulirani dodatni članovi u niti jednoj od klasa

te definirajte svaku od metoda na odgovarajući način. Zatim instancirajte vektor `vec` koji istovremeno pohranjuje adrese na instance klasa `Rectangle`, `Square` i `Circle`. Taj vektor popunite s 4 vrijednosti, a zatim prođite elementima vektora i ispišite površine objekata na koje ti elementi pokazuju koristeći *range-based* for petlju. Na kraju koristeći `count_if` iz biblioteke `algorithm` prebrojite koliko je objekata čija je površina veća od `10.0`. Primjer ispisa s objektima

```
Rectangle r1{5.5, 7.2};
Square s{2.5};
Rectangle r2{7.8, 7.9};
Circle c{3.1};
```

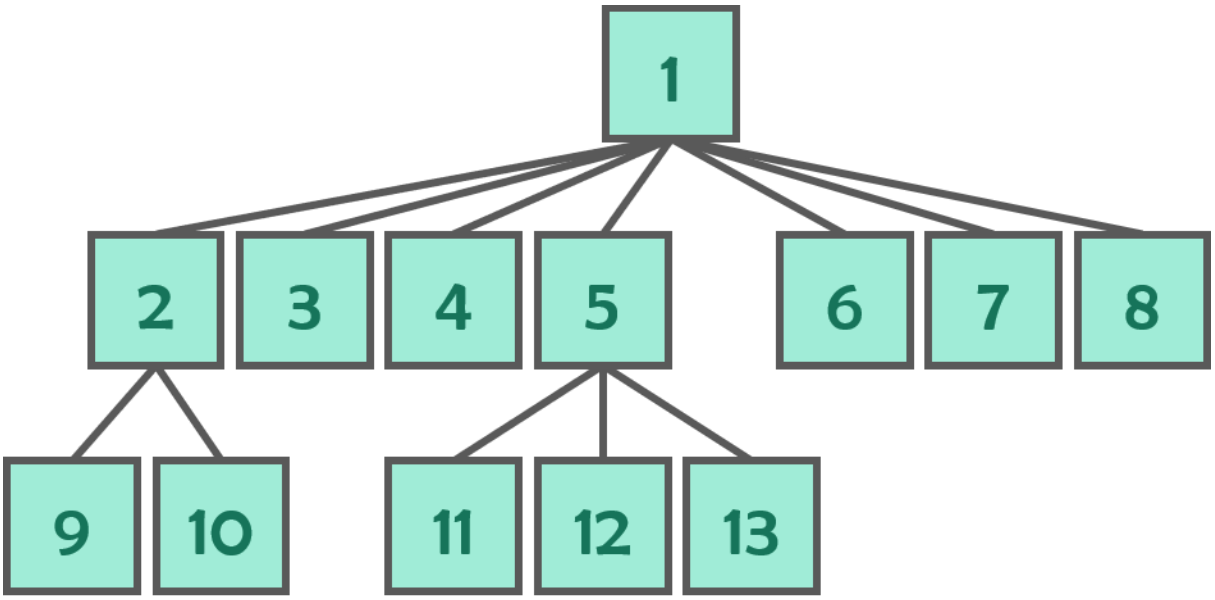
treba biti:

```
39.6    6.25    61.62    30.1907
3
```

pri čemu je `3` u drugome retku vrijednost koju vraća `count_if`.

Zadatak 2. (C++) [35 bodova]

Kako je i rečeno, binarno se stablo može poopćiti na n-arno stablo čiji čvorovi imaju proizvoljan broj djece. Primjer jednog takvog stabla dan je na Slika 1.



Slika 1 n-arno stablo

Struktura koja predstavlja čvor n-arnog stabla zadana je deklaracijom

```
template <typename T>
class Node {
private:
    T element;
    Node* parent;
    list<Node*> children;
public:
    Node(const T&);
    ~Node();

    void setElement(const T&);
    const T& getElement() const;

    void addChild(Node&);
    void removeChild(Node&);
    void setParent(Node&);

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

Pri čemu je `children` lista pokazivača na djecu za pojedini čvor, `parent` pokazivač na roditelja u stablu, a `element` sama vrijednost koju čvor enkapsulira.

`Node(const T&)` inicijalizira `element` s onim što je prosljeđeno kao argument, dok ostale metode rade ono što im i ime govori.

Sama klasa koja predstavlja stablo zadana je s

```
template <typename T>
class Tree {
private:
    Node<T>* root;
public:
    Tree(Node<T>*);
    ~Tree();

    void preorderWalk(Node<T>*, vector<Node<T>*>&);
};
```

gdje `root` je pokazivač na korijen stabla, dok konstruktor `Tree(Node<T>*)` postavlja taj pokazivač na prosljeđenu vrijednost. Metoda `preorderWalk` rekurzivno obilazi stablo na preorder način, dok se trenutni element pri obilasku prirodaje u vektor čija je referenca drugi argument te metode.

Definirajte članove ove klase. Izgradite stablo kao sa Slika 1, pozovite `preorderWalk`, a zatim redom ispišite vrijednosti vektora (pri čemu je nužno koristiti preopterećeni `operator<<` za klasu `Node<T>`) kojeg ste po referenci proslijedili toj metodi. Ispis treba biti:

```
Preorder walk
Node with element 1, parent -1  and children: 2 3 4 5 6 7 8
Node with element 2, parent 1  and children: 9 10
Node with element 9, parent 2  and children:
Node with element 10, parent 2  and children:
Node with element 3, parent 1  and children:
Node with element 4, parent 1  and children:
Node with element 5, parent 1  and children: 11 12 13
Node with element 11, parent 5  and children:
Node with element 12, parent 5  and children:
Node with element 13, parent 5  and children:
Node with element 6, parent 1  and children:
Node with element 7, parent 1  and children:
Node with element 8, parent 1  and children:
```

Zadatak 3. (Python) [30 bodova]

Po uzoru na klasu CSLL koja predstavlja jednostruko povezanu cikličku listu i koju smo implementirali u C++-u, implementirajte takvu strukturu u Pythonu. Budući da Python ne sadrži pokazivače, to nećemo voditi računa o susjedstvu spremajući adrese, već spremajući indekse. Deklaracija klase CSLL je:

```
class CSLL:
    __slots__ = ['value', 'next', 'head']
    def __init__(self):
        pass
    def is_empty(self):
        pass
    def prepend(self, element):
        pass
    def append(self, element):
        pass
    def remove(self, index):
        pass
    def prev(self, index):
        pass
    def __str__(self):
        pass
```

Gdje je head indeks prvog elementa u instanci klase CSLL (ne nužno prvog elementa u listi value), value je lista vrijednosti za pojedine čvorove, a lista susjedstva next. Sve se metode ponašaju kako im i ime govori, dok prev pronalazi prethodnika za pojedini element na indeksu index.

Ako redom na početak ubacujemo vrijednosti od 1 do 10, tada će ispis biti sljedeći:

```
List at 0x7739e0
Node with value 10 at index 9 and neighbor at index 8
Node with value 9 at index 8 and neighbor at index 7
Node with value 8 at index 7 and neighbor at index 6
Node with value 7 at index 6 and neighbor at index 5
Node with value 6 at index 5 and neighbor at index 4
Node with value 5 at index 4 and neighbor at index 3
Node with value 4 at index 3 and neighbor at index 2
Node with value 3 at index 2 and neighbor at index 1
Node with value 2 at index 1 and neighbor at index 0
Node with value 1 at index 0 and neighbor at index 9
```

U drugome primjeru gdje ubacujemo izmjenično vrijednosti i na početak i na kraj

```
m.prepend(7)
m.append(3)
m.prepend(25)
m.append(13)
m.prepend(26)
m.prepend(27)
m.prepend(28)
m.append(-1)
```

ispis treba biti:

List at 0xdf3a08
Node with value 28 at index 6 and neighbor at index 5
Node with value 27 at index 5 and neighbor at index 4
Node with value 26 at index 4 and neighbor at index 2
Node with value 25 at index 2 and neighbor at index 0
Node with value 7 at index 0 and neighbor at index 1
Node with value 3 at index 1 and neighbor at index 3
Node with value 13 at index 3 and neighbor at index 7
Node with value -1 at index 7 and neighbor at index 6

Ako npr. u ovome primjeru uklonimo element koji ima indeks `6`, dakle pozovemo `m.remove(6)`, tada ispis treba biti:

List at 0xdf3a08
Node with value 27 at index 5 and neighbor at index 4
Node with value 26 at index 4 and neighbor at index 2
Node with value 25 at index 2 and neighbor at index 0
Node with value 7 at index 0 and neighbor at index 1
Node with value 3 at index 1 and neighbor at index 3
Node with value 13 at index 3 and neighbor at index 7
Node with value -1 at index 7 and neighbor at index 5

Definirajte odgovarajuće članove ove klase i pozovite primjere.