Pismeni ispit

Objektno orijentirano programiranje

6. rujna 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_PISMENI5`, tu mapu zipajte u datoteku `PREZIME_IME_PISMENI5.zip` te ju pošaljite na oop@mathos.hr s naslovom e-mail poruke: "PREZIME IME: OOP PISMENI 5".

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

Implementirajte klasu koja predstavlja jednostruko povezanu listu SLL<T> i klasu koja predstavlja njoj odgovarajući element Node<T>. Ove su klase deklarirane s:

```
template <typename T>
class Node {
protected:
   T value;
   Node* next;
public:
   Node();
   Node(const T&);
   Node(const Node&);
   ~Node();
};


template <typename T>
class SLL {
private:
   Node<T>* head, * tail;
public:
   SLL();
   SLL(const SLL&);
   ~SLL();
```

```
    bool isEmpty() const;
    void prepend(const T&);
    void append(const T&);

    T headValue() const;
    T tailValue() const;
    bool removeFromHead();
    bool removeFromTail();
    bool remove(Node<T>*);
};
```

Metoda prepend dodaje novi element na početak, a append na kraj liste. Metoda headValue po vrijednosti vraća prvi element, a tailValue zadnji element liste. Metoda removeFromHead uklanja element s početka, removeFromTail s kraja, dok remove uklanja onaj element čiji joj je pokazivač prosljeđen. Implementirajte metode u klasi Node<T> *[5 bodova]* i metode klase SLL<T> *[15 bodova]*. Preopteretite dodatno operatore ispisa za obje klase *[5 bodova]*.

Nakon što implementirate odgovarajuće članove, instancirajte novu listu tipa int, napunite ju s 10 vrijednosti koristeći prepend, zatim uklanjajte elemente dok ne postane prazna koristeći removeFromTail, zatim ju ponovno napunite s deset vrijednosti koristeći append pa uklanjajte elemente koristeći removeFromHead sve dok ne postane prazna *[10 bodova]*.

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

Klasa Expression predstavlja općeniti matematički izraz (npr. sumu više brojeva, derivaciju, integral, umnožak, itd...), a njena evaluate metoda taj izraz izvršava/računa unutar računala. Deklaracija te klase je

```
class Expression {
protected:
    double evaluate();
};
```

Nadogradite ovu klasu tako da ona bude apstraktna *[5 bodova]*, a zatim ju naslijedite u klase SineIntegral, CosineIntegral i Sum<S>.

Klasa SineIntegral, odnosno CosineIntegral, predstavlja određeni integral funkcije sin, odnosno funkcije cos u rasponu od x_1 do x_2. U tu svrhu ove klase enkapsuliraju privatne članove x1 i x2 tipa double, a dodatno i član dx tipa double.

Unutar računala, integral ćemo računati pomoću izraza

$$\sum_{x} f(x)\Delta x$$

za x-eve u rasponu od x_1 do x_2 s međukorakom Δx. Član dx predstavlja dovoljno malu vrijednost izraza Δx pohranjenu unutar računala. Implementirajte ove klase tako da za svaku definirate odgovarajući konstruktor i odgovarajuću metodu evaluate *[15 bodova]*.

Klasa Sum<S> predstavlja sumu, a ona enkapsulira privatno polje numbers tipa double duljine S gdje je S varijabla predloška. Konstruktor klase Sum<S> prihvaća referencu na polje double-ova duljine S, a zatim se vrijednosti elemenata tog polja dodaju u polje numbers enkapsulirano unutar klase Sum<S>. Implementirajte odgovarajući konstruktor i odgovarajuću metodu evaluate za ovu klasu *[10 bodova]*.

Instancirajte nekoliko objekata ovih klasa, npr.:

```
double arr1[5] = {1.0, 2.0, 3.0, 4.0, 5.0};
Sum<5> s1{arr1};
double arr2[3] = {1.5, 2.5, 3.5};
Sum<3> s2{arr2};
SineIntegral sInt{0.0001, 0.0, M_PI};
CosineIntegral cInt{0.0001, 0.0, M_PI / 2.0};
```

Zatim alocirajte polje koje će istovremeno pohranjivati adrese svih njih, prođite elementima polja, evaluirajte svaki izraz te ga ispišite *[5 bodova]*. Primjer ispisa za prethodno instancirane klase treba biti:

```
15
7.5
2
1.00005
```

Zadatak 3. (Python) [30 bodova]

Po uzoru na klasu SLL iz prvog zadatka koja predstavlja jednostruko povezanu listu, implementirajte takvu strukturu u Pythonu. Budući da Python ne sadrži pokazivače, to nećemo voditi računa o susjedstvu spremajući adrese, već spremajući indekse. Deklaracija klase SLL je:

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

Gdje je `head` indeks prvog elementa u instanci klase `SLL` (ne nužno prvog elementa u listi `value`), `tail` indeks zadnjeg elementa u instance klase `SLL` (ne nužno zadnjeg elementa u listi `value`), `value` je lista vrijednosti za pojedine čvorove, a `next` je lista susjedstva. Sve se metode ponašaju kako im i ime govori.

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

```
List at 0x13b3af8
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 -1
```

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

```
m.prepend(5)
m.prepend(8)
m.append(9)
m.prepend(12)
m.prepend(24)
m.append(3)
m.prepend(1)
```

ispis treba biti:

```
List at 0x3523aa8
Node with value 1 at index 6 and neighbor at index 4
Node with value 24 at index 4 and neighbor at index 3
Node with value 12 at index 3 and neighbor at index 1
Node with value 8 at index 1 and neighbor at index 0
Node with value 5 at index 0 and neighbor at index 2
Node with value 9 at index 2 and neighbor at index 5
Node with value 3 at index 5 and neighbor at index -1
```

Ako npr. u ovome primjeru uklonimo element koji ima indeks `3`, dakle pozovemo `m.remove(3)`, tada ispis treba biti:

```
List at 0x3523aa8
Node with value 1 at index 6 and neighbor at index 4
Node with value 24 at index 4 and neighbor at index 1
Node with value 8 at index 1 and neighbor at index 0
Node with value 5 at index 0 and neighbor at index 2
Node with value 9 at index 2 and neighbor at index 5
Node with value 3 at index 5 and neighbor at index -1
```

Definirajte odgovarajuće članove ove klase i pozovite primjere *[30 bodova]*.