OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 1 -

Funkcionalna specifikacija

Na programskom jeziku C++ implementirati statičku biblioteku (.lib) apstraktnih tipova podataka za vreme. Potom napisati glavni program (kao konzolnu .exe aplikaciju) koji testira klase za vreme.

Specifikacija klase Interval

Klasa `Interval` predstavlja apstraktni tip podataka za vremenske intervale. Interval je određen brojem dana (days), sati (hours) i minuta (minutes) i ima predznak (sign: bool). Broj dana je nenegativan ceo broj, broj sati pripada intervalu [0, 23], a broj minuta intervalu [0, 59].

Stvaranje intervala. Obezbediti konstruktor `Interval(int dd, int hh, int mm, bool sign)`. Opciono pri incijalizaciji minuti, sati i predznak mogu da se izostave i u tom slučaju dobijaju podrazumevanu vrednost 0, 0, true (pozitivan) respektivno.

Sabiranje/oduzimanje intervala. Dva intervala se mogu sabrati/oduzeti, pri čemu je rezultat novi interval. Operacija sabiranja ne sme izmeniti stanje operanada. Implementirati sledeće metode:

```
Interval Interval::add(Interval& other);
Interval Interval::subtract(Interval& other);.
```

Interval je nepromenljiv apstraktni tip podataka (immutable). Jednom kad se stvori objekat i definiše stanje, ne sme se ni na koji način više promeniti.

Poređenje intervala. Omogućiti poređenje po jednakosti, kao i utvrđivanje poretka.

```
bool Interval::equals(Interval& other);
bool Interval::lessThan(Interval& other);
bool Interval::greaterThan(Interval& other);
```

Ispisivanje intervala. Omogućiti ispis vremenskih intervala u izlazni tok. Implementirati metodu print koja ispisuje interval u formatu [+/-]*dd.hh:MM*, odnosno u formatu [+/-]*hh:MM,* ako je broj dana jednak nuli. Jednocifrene vrednosti za sate i minute moraju se dopuniti vodećom nulom.

```
void print(ostream& out, Interval& interval);
```

Nula interval. Napraviti javni statički objekat `Interval::`**`ZERO`** čije su sve komponente jednake 0.

Specifikacija klase Timestamp

Klasa `Timestamp` predstavlja apstraktni tip podataka za vremenske trenutke. Vremenski trenutak je određen sledećim podacima: godina, mesec, dan, sati, minuti. Dodatno, vremenski trenutak ima još i interval pomeraja vremenske zone (*zoneOffset*) u odnosu na koordinisano svetsko vreme (eng. *Coordinated Universal Time, UTC*). NIJE DOZVOLJENO koristiti bilo kakve gotove biblioteke koje već implementiraju slične ili iste funkcionalnosti!

Stvaranje vremenskih trenutaka. Potrebno je Implementirati konstruktor za inicijalizaciju `Timestamp::Timestamp(int year, int month, int dayInMonth, int hours, int minutes, Interval* zoneOffset);`.Omogućiti i skraćenu inicijalizaciju sa vremenske trenutke pomoću vrednosti za datum, pri čemu su vremenske koordinate 0:

```
Timestamp::Timestamp(int yy, int mm, int dd, Interval* zoneOffset);
```

U oba slučaja klasa Timestamp mora da napravi sopstevnu dinamičku kopiju objekta za pomeraj vremenske zone. Proveravati pri pravljenju vremena da broj dana za zoneOffset mora biti nula, broj sati u intervalu [0, 12], a predznak pozitivan ili negativan.

Sabiranje/oduzimanje vremena sa intervalom. Dodavanjem vremenskog intervala na vreme, dobija se novo vreme. Operandi ne smeju biti promenjeni, a vremenska zona se ne menja.

```
Timestamp Timestamp::add(Interval& interval);
Timestamp Timestamp::subract(Interval& interval).
```

Oduzimanje dva vremena. Omogućiti oduzimanje dva trenutka pri čemu je rezultat tipa Interval.

`Interval Timestamp::subtract(Timestamp& right)`. Pri oduzimanju, voditi računa o svođenju operanada na istu vremensku zonu. Metoda ne sme izmeniti stanje svojih operanada!

Poređenje dva vremena. Omogućiti poređenje na jednakost, kao i utvrđivanje poretka metodama:

```
bool Timestamp::equals(Timestamp& time);
bool Timestamp::lessThan(Timestamp& time);
bool Timestamp::greaterThan(Timestamp& time).
```

Voditi računa o svođenju vremenskih trenutaka na istu vremensku zonu prilikom poređenja. Metode ne smeju izmeniti stanje svojih operanada, a u skladu sa sledećim zahtevom.

Vreme mora biti nepromenljivi (immutable) tip podataka. Jednom kada se napravi objekat tipa `Timestamp` i jednom kad mu se definiše vrednost, NE SME se više menjati.

Ispisivanje vremena u izlazni tok. Omogućiti ispis vremena u izlazni tok. Implementirati metodu print koja ispisuje vreme u formatu yyyy-mm-dd hh:MM (UTC [+/-]hh:MM), gde komponenta hh:MM za pomeraj vremenske zone treba da bude iskorišćena/pozvana iz klase Interval.

```
void print(ostream& out, Timestamp& time);
```

Primer validnih vrednosti u zadatom formatu: 2016-10-26 10:12 (UTC +02:00),

2015-08-06 08:02 (UTC -04:30).

Test i error funkcija

U projektu glavnog programa treba da postoji funkcija `void test();` koja testira rad sa datim klasama. Studenti treba da implementiraju datu funkciju i uslovno je prevode ako je definisan makro STUDENT_TEST. Predvideti i postojanje makroa PROF_TEST. Ako su oba makroa definisana, PROF_TEST ima prioritet i tada se prevodi tajni test primer koji će biti dat na odbrani. Funkcija main treba samo da pozove test funkciju. U klasama Interval i Timestamp funkcija `test` treba da bude deklarisana kao prijateljska.

Predvideti i uslovno prevoditi funkciju `void errorDetected(string& msg)`. Studentska implementacija treba samo da ispiše grešku i prekine izvršavanje programa.

Tehnički zahtevi i smernice za izradu rešenja

Sve klase i metode moraju biti imenovane prema zahtevima iz domaćeg zadatka. Poljima klase, zaštićenim od direktnog pristupa, se pristupa isključivo pomoću predviđenih metoda za čitanje i pisanje vrednosti polja. Nije dozvoljeno dodavanje nijedne javne funkcije/metode koja nije predviđena specifikacijom (privatne uslužne funkcije mogu). Svaka klasa koja koristi dinamičku memoriju mora imati korektno napisan destruktor. Sve metode smestiti u odgovarajuće klase. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove, pri čemu jedno zaglavlje mora sadržati najviše jednu klasu. Nije dozvoljeno korišćenje globalnih promenljivih za razmenu podataka. U slučaju bilo kakve greške u toku izvršavanja programa, pozvati funkciju `errorDetected`.Ukoliko u zadatku nešto nije dovoljno jasno definisano, treba usvojiti razumnu pretpostavku i na temeljima te pretpostavke nastaviti izgrađivanje svog rešenja.

VAŽNE NAPOMENE

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, .h i .cpp fajlove.

- Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (`utctime.lib`).
- Glavni program napisati u posebnom projektu koji se prevodi kao Win32 Console Application (`test.exe`) fajl i koji treba povezati sa statičkim bibliotekama. Glavni program treba implementirati tako da pozove globalnu funkciju `void test();`.
- Studenti treba da implementiraju svoju verziju ove funkcije tako da demonstriraju operacije sa vremenom i vremenskim intervalima.