II PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA"

Ovdje će biti prikazani samo zadaci (i rješenja) za grupu "A" s obzirom da u zadacima za grupe B, C i D nema nikakve suštinske razlike. Sve razlike su samo svedene na nivo da se spriječi doslovno prepisivanje. Također, ponuđena rješenja su pisana sa ciljem da budu što kraća, što nije nužno i najefikasnije rješenje. Naravno, postoje i brojna druga ispravna rješenja...

Zadatak 1 (8 poena):

Razvijte klasu koja omogućava rad sa uglovima izraženim u stepenima, minutama i sekundama. Interfejs klase treba da sadrži sljedeće elemente:

a) Konstruktor sa tri cjelobrojna parametra, koji redom predstavljaju stepene, minute i sekunde. Ovaj konstruktor treba da inicijalizira ugao na zadani iznos stepeni, minuta i sekundi. Treba dozvoliti mogućnost i da vrijednost parametara kojim se zadaju stepeni, minute i sekunde mogu biti i izvan tipičnog opsega, ali tada automatski treba preračunavati preljev (npr. 130 sekundi treba automatski tretirati kao 2 minute i 10 sekundi, 750 stepeni treba tretirati kao 30 stepeni jer 360 stepeni predstavlja pun krug, itd.). Stoga, na primjer, ugao zadan kao 130 stepeni, 90 minuta i 150 sekundi treba tretirati kao 131 stepen, 32 minuta i 30 sekundi. S druge strane, ukoliko je makar jedan od parametara negativan, konstruktor treba da baci izuzetak.

b) Metodu sa tri parametra koja obavlja načelno istu funkciju kao i konstruktor, a omogućava naknadno postavljanje ugla.

c) Metodu sa tri parametra koja očitava informacije o stepenima, minutama i sekundama koje tvore ugao i smješta ove informacije u odgovarajuće parametre metode.

d) Metodu bez parametara koja daje iznos ugla samo u stepenima (kao realni broj). Na primjer, ova metoda primijenjena na ugao od 40 stepeni i 30 minuta vraća 40.5 kao rezultat.

e) Preklopljeni unarni operator "–" koji primijenjen na neki ugao daje njegovu dopunu do punog kruga od 360 stepeni. Na primjer, ukoliko ugao "a" sadrži 110 stepeni, 20 minuta i 45 sekundi, ugao "–a" treba da sadrži 249 stepeni, 39 minuta i 15 sekundi.

f) Preklopljene binarne operatore "+" i "–" koji sabiraju odnosno oduzimaju dva ugla. Uputa: ukoliko ste konstruktor realizirali kako treba, realizacija ovih operatora trebala bi biti trivijalna.

g) Preklopljeni binarni operator "*" koji množi ugao sa cijelim brojem. Trebalo bi biti podržano kako množenje ugla sa brojem, tako i množenje broja sa uglom.

h) Preklopljene binarne operatore "+=", "-=" i "*=" koji osiguravaju da izrazi "X += Y", "X -= Y" i "X *= Y" uvijek imaju isto značenje kao i izrazi "X = X + Y", "X = X - Y" i "X = X * Y" kadgod ovi izrazi imaju smisla.

i) Preklopljen unarni operator "++" koji povećava ugao na koji je primijenjen za jednu ugaonu sekundu. Potrebno je podržati i prefiksnu i sufiksnu verziju ovog operatora.

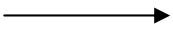
j) Preklopljene relacione operatore "<",">", "==","!=","<=" i ">=" koji uspoređuju dva ugla.

k) Preklopljeni operator "<<" za ispis ugla na ekran u obliku <*stepeni*>d <*minute*>m <*sekunde*>s, na primjer 23d 15m 42s, kao i preklopljeni operator ">>" za čitanje ugla sa tastature, u istom obliku u kojem se i vrši ispis.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve.

Rješenje:

```
class Ugao {
  int d, m, s;  // d = stepeni, m = minute, s = sekunde
public:
  Ugao(int d1, int m1, int s1) { Postavi(d1, m1, s1); }
  void Postavi(int d1, int m1, int s1);
  void Ocitaj(int &d1, int &m1, int &s1) const { d1 = d; m1 = m; s1 = s; }
  double Stepeni() const { return d + m / 60. + s / 3600.; }
  friend Ugao operator -(const Ugao &u) {
    return Ugao(359 - u.d, 59 - u.m, 60 - u.s);
  }
  friend Ugao operator +(const Ugao &u1, const Ugao &u2) {
    return Ugao(u1.d + u2.d, u1.m + u2.m, u1.s + u2.s);
  }
  friend Ugao operator -(const Ugao &u1, const Ugao &u2) {
    return u1 + -u2;
  }
  friend Ugao operator *(const Ugao &u, int n) {
    return Ugao(u.d * n, u.m * n, u.s * n);
  }
  friend Ugao operator *(int n, const Ugao &u) { return u * n; }
  Ugao &operator +=(const Ugao &u) { return *this = *this + u; }
  Ugao &operator -=(const Ugao &u) { return *this = *this - u; }
  Ugao &operator *=(int n) { return *this = *this * n; }
  Ugao &operator ++() { return *this = Ugao(d, m, s + 1); }
  Ugao operator ++(int) { Ugao u = *this; ++*this; return u; }
  friend bool operator ==(const Ugao &u1, const Ugao &u2) {
    return u1.d == u2.d && u1.m == u2.m && u1.s == u2.s;
  }
  friend bool operator !=(const Ugao &u1, const Ugao &u2) {
    return !(u1 == u2);
  }
  friend bool operator <(const Ugao &u1, const Ugao &u2) {
    return u1.Stepeni() < u2.Stepeni();
  }
  friend bool operator <=(const Ugao &u1, const Ugao &u2) {
    return u1.Stepeni() <= u2.Stepeni();
  }
  friend bool operator >(const Ugao &u1, const Ugao &u2) {
    return u1.Stepeni() > u2.Stepeni();
  }
  friend bool operator >=(const Ugao &u1, const Ugao &u2) {
    return u1.Stepeni() >= u2.Stepeni();
  }
  friend ostream &operator <<(ostream &cout, const Ugao &u) {
    return cout << u.d << "d " << u.m << "m " << u.s << "s";
  }
  friend istream &operator >>(istream &cin, Ugao &u);
};
void Ugao::Postavi(int d1, int m1, int s1) {
  if(d1 < 0 || m1 < 0 || s1 < 0) throw "Neispravni parametri!\n";
  s = s1 % 60; m1 += s1 / 60; m = m1 % 60; d1 += m1 / 60; d = d1 % 360;
}
istream &operator >>(istream &cin, Ugao &u) {
  cin >> u.d;
  if(cin.get() != 'd') cin.setstate(ios::failbit);
  cin >> u.m;
  if(cin.get() != 'm') cin.setstate(ios::failbit);
  cin >> u.s;
  if(cin.get() != 's') cin.setstate(ios::failbit);
}
```



Zadatak 2 (12 poena):

Razvijte kontejnersku klasu koja čuva kolekciju podataka o uglovima (ovakva klasa bi, na primjer, mogla služiti za čuvanje podataka o uglovima nekog mnogougla). Pri tome pretpostavite da se podaci o pojedinim uglovima čuvaju u primjercima klase razvijene u prethodnom zadatku. Kontejnerska klasa treba da sadrži pokazivač na prvi element dinamički alociranog niza pokazivača koji pokazuju na dinamički alocirane objekte koji sadrže stvarne podatke o pohranjenim uglovima. Ova klasa još sadrži informaciju o broju pohranjenih uglova, kao i informaciju o kapacitetu, odnosno maksimalnom broju uglova koji se mogu pohraniti (ova informacija se čuva u konstantnom atributu). Inače, svi atributi klase moraju biti u privatnoj sekciji klase. Interfejs klase treba da sadrži sljedeće elemente:

a) Konstruktor sa jednim cjelobrojnim parametrom, koji predstavlja maksimalni broj uglova koji se mogu pohraniti. Konstruktor treba da inicijalizira odgovarajuće atribute i da izvrši neophodnu alokaciju memorije. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip klase.

b) Destruktor, koji oslobađa svu memoriju koja je zauzeta tokom života primjeraka klase.

c) Konstruktor kopije, koji obezbjeđuje da se primjerci ove klase mogu bezbjedno kopirati, prenositi po vrijednosti u funkcije i vraćati kao rezultati iz funkcija.

d) Preklopljeni operator dodjele, koji omogućava bezbjedno međusobno dodjeljivanje primjeraka ove klase. Dodjelu treba podržati samo u slučaju da izvorni i odredišni objekat imaju isti kapacitet, a u suprotnom treba baciti izuzetak.

e) Metodu sa tri parametra, koja upisuje novi ugao u kolekciju. Parametri su broj stepeni, minuta i sekundi. Metoda treba da baci izuzetak u slučaju da je kolekcija popunjena.

f) Metodu koja briše sve pohranjene uglove.

g) Metodu koja ispisuje sve pohranjene uglove na ekran (svaki ugao u novom redu), koristeći operator za ispis definiran u klasi koja opisuje ugao, kao i metodu koja umjesto na ekran vrši ispis u tekstualnu datoteku čije se ime zadaje kao parametar metode.

h) Metodu koja čita podatke o uglovima iz tekstualne datoteke čije se ime zadaje kao parametar. Tekstualna datoteka je formatirana identično kao pri ispisu. Prethodni sadržaj kolekcije prilikom čitanja iz tekstualne datoteke treba izbrisati.

i) Metode koje vraćaju najveći i najmanji ugao pohranjen u kolekciji.

j) Metode koje vraćaju ukupan broj uglova u kolekciji, kao i broj oštrih uglova (tj. uglova manjih od 90 stepeni).

k) Preklopljen operator indeksiranja "`[]`" koji omogućava da se pristupi *i*-tom pohranjenom uglu, gdje je indeks *i* naveden kao parametar u uglastim zagradama. U slučaju da je indeks izvan opsega, treba baciti izuzetak. Ovaj operator bi trebao omogućiti da se vraćeni ugao koristi sa lijeve strane znaka dodjeljivanja kad god tako nešto ima smisla.

l) Metode koje snimaju podatke o uglovima u binarnu datoteku, odnosno obnavljaju sadržaj podataka iz binarne datoteke. Obje metode kao parametar zahtijevaju ime datoteke.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve.

Rješenje:

```
class Kontejner {
  Ugao **uglovi;
  int br;
  const int Kapacitet;
public:
  explicit Kontejner(int n) : Kapacitet(n), br(0), uglovi(new Ugao*[n]) {}
  ~Kontejner() { Brisi(); delete[] uglovi; }
  Kontejner(const Kontejner &k);
  Kontejner &operator =(const Kontejner &k);
  void Upisi(int d, int m, int s);
  void Brisi();
  void Ispisi() const;
  void IspisiUDatoteku(const char ime[]) const;
  void Citaj(const char ime[]);
  Ugao Najveci() const;
  Ugao Najmanji() const;
  int BrojUglova() const { return br; }
  int BrojOstrih() const;
  Ugao &operator [](int i);
  Ugao operator[](int i) const;
  void Sacuvaj(const char ime[]) const;
  void Obnovi(const char ime[]);
};
Kontejner::Kontejner(const Kontejner &k) : Kapacitet(k.Kapacitet),
  br(k.br), uglovi(new Ugao*[k.Kapacitet]) {
    for(int i = 0; i < br; i++) uglovi[i] = new Ugao(*k.uglovi[i]);
}
Kontejner &Kontejner::operator =(const Kontejner &k) {
  if(&k == this) return *this;
  if(Kapacitet != k.Kapacitet) throw "Razliciti kapaciteti!\n";
  Brisi(); br = k.br;
  for(int i = 0; i < br; i++) uglovi[i] = new Ugao(*k.uglovi[i]);
  return *this;
}
void Kontejner::Upisi(int d, int m, int s) {
  if(br == Kapacitet) throw "Kontejner popunjen!";
  uglovi[br++] = new Ugao(d, m, s);
}
void Kontejner::Brisi() {
  for(int i = 0; i < br; i++) delete uglovi[i];
  br = 0;
}
void Kontejner::Ispisi() const {
  for(int i = 0; i < br; i++) cout << *uglovi[i] << endl;
}
void Kontejner::IspisiUDatoteku(const char ime[]) const {
  ofstream izlaz(ime);
  for(int i = 0; i < br; i++) izlaz << *uglovi[i] << endl;
}
void Kontejner::Citaj(const char ime[]) {
  ifstream ulaz(ime);
  Brisi();
  Ugao u(0, 0, 0);
  while(ulaz >> u) uglovi[br++] = new Ugao(u);
}
```

```
Ugao Kontejner::Najveci() const {
  if(br == 0) throw "Nema upisanih uglova!\n"; // S obzirom da niz "uglovi"
  Ugao *pmax(uglovi[0]);                       // sadrži pokazivače, ove
  for(int i = 1; i < br; i++)                  // metode je prilično teško
    if(*uglovi[i] > pmax) pmax = uglovi[i];    // izvesti preko funkcija iz
  return *pmax;                                // biblioteke "algorithm"...
}

Ugao Kontejner::Najmanji() const {
  if(br == 0) throw "Nema upisanih uglova!\n";
  Ugao *pmin(uglovi[0]);
  for(int i = 1; i < br; i++)
    if(*uglovi[i] < pmin) pmin = uglovi[i];
  return *pmin;
}

int Kontejner::BrojOstrih() const {
  int brojac(0);
  for(int i = 0; i < br; i++)
    if(uglovi[i]-> Stepeni() < 90) brojac++;
  return brojac;
}

Ugao &Kontejner::operator [](int i) {
  if(i < 0 || i >= br) throw "Pogrešan indeks!\n";
  return *uglovi[i];
}

Ugao Kontejner::operator[](int i) const {
  if(i < 0 || i >= br) throw "Pogrešan indeks!\n";
  return *uglovi[i];
}

void Kontejner::Sacuvaj(const char ime[]) const {
  ofstream izlaz(ime, ios::out | ios::binary);
  izlaz.write((char*)this, sizeof *this);
  for(int i = 0; i < broj_studenata; i++)
    izlaz.write((char*)uglovi[i], sizeof(Ugao));
}

void Kontejner::Obnovi(const char ime[]) {
  ifstream ulaz(ime, ios::in | ios::binary);
  if(!ulaz) throw "Datoteka ne postoji!\n";
  Brisi(); delete[] uglovi;
  ulaz.read((char*)this, sizeof *this);
  uglovi = new Ugao*[Kapacitet];
  for(int i = 0; i < br; i++) {
    uglovi[i] = new Ugao(0, 0, 0);
    ulaz.read((char*)studenti[i], sizeof(Ugao));
  }
}
```