Zadaća 4

Ova zadaća nosi **5 poena**. Rok za predaju je **nedjelja**, **7. I 2024.** do kraja dana. Zadaće se predaju putem Zamgera. Broj poena koji student dobija ovisi od broja korektno implementiranih stavki koje se traže u zadaći.

Zadatak 1 (2,5 poena)

U ovom zadatku, potrebno je realizirati klasu "`ChebyshevApproximation`" koja služi kao podrška aproksimaciji funkcija pomoću Chebyshevljevih polinoma, kao i numeričkom diferenciranju odnosno numeričkoj integraciji zasnovanoj na ovoj aproksimaciji. Ova klasa treba da ima sljedeći interfejs:

```
template <typename FunType>
  ChebyshevApproximation(FunType f, double xmin, double xmax, int n);
void set_m(int m);
void trunc(double eps);
double operator()(double x) const;
double derivative(double x) const;
ChebyshevApproximation derivative() const;
ChebyshevApproximation antiderivative() const;
double integrate(double a, double b) const;
double integrate() const;
```

Konstruktor prima kao parametre funkciju f čija se aproksimacija vrši, zatim granice intervala na kojima se vrši aproksimacija, te broj uzoraka n na osnovu kojeg se vrši aproksimacija. Unutar ovog konstruktora vrši se proračun odgovarajućih koeficijenata $c_k, k = 0, 1, ..., n$, i inicijalno se postavlja $m = n$ (pri čemu je m stepen polinoma kojim se vrši aproksimacija). Konstruktor je napisan kao generički, da bi se omogućilo da prvi parametar može biti ne samo funkcija, nego i bilo šta što se može koristiti poput funkcije, npr. neki funkcijski objekat (alternativno bi prvi parametar mogao biti i polimorfni funkcijski omotač tipa "`std::function<double(double)>`", ali su generičke tehnike za ovu svrhu efikasnije). Konstruktor treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Bad parameters" ukoliko je $x_{\min} \geq x_{\max}$ ili je $n < 1$. Pozivom metode "`set_m`" postavljena vrijednost m može se naknadno promijeniti, što omogućava da naknadno možemo mijenjati red polinoma kojim se vrši aproksimacija. Ova metoda treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Bad order" u slučaju da je m manji od 1 ili veći od n. Pozivom metode "`trunc`" vrijednost za m proračunava se automatski tako da se odbace svi koeficijenti koji su po apsolutnoj vrijednosti manji od zadanog parametra "`eps`", tj. m se postavlja na najmanju vrijednost takvu da su svi koeficijenti c_k za $k > m$ manji po modulu od tolerancije zadane parametrom. Pri tome se ne smije dozvoliti da m postane manji od 1. Ukoliko bi se to desilo (recimo, ako su svi koeficijenti $c_k, k = 0, 1, ..., n$ manji od "`eps`"), treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Bad tolerance". Ista stvar treba da se desi i ukoliko je parametar "`eps`" negativan.

Računanje aproksimacije u proizvoljnoj tački povjereno je operatorskoj funkciji za operator "`()`". Time primjerci ove klase postaju funkcijski objekti, odnosno mogu se koristiti poput funkcija, kao alternativa za funkciju koju aproksimiraju. Metoda "`derivative`" sa jednim parametrom vrši numeričku aproksimaciju prvog izvoda funkcije f u zadanoj tački (koristeći također Chebyshevljeve polinome). Operator "`()`" i opisana verzija metode "`derivative`" bacaju izuzetak tipa "`domain_error`" uz prateći tekst "Bad argument" ukoliko vrijednost argumenta x nije unutar intervala $[x_{\min}, x_{\max}]$.

Treba podržati i verziju metode "`derivative`" bez parametara koja daje kao rezultat novi funkcijski objekat tipa "`ChebyshevApproximation`" koji predstavlja aproksimaciju prvog izvoda funkcije f na čitavom intervalu na kojem se aproksimacija vrši. Slična ovoj metodi je i metoda "`antiderivative`" (također bez parametara) koja daje novi funkcijski objekat tipa "`ChebyshevApproximation`" koji predstavlja aproksimaciju primitivne funkcije (antiderivacije) funkcije f na čitavom intervalu na kojem se aproksimacija vrši. Da bi se ove dvije metode mogle efikasno izvesti, dobro je uvesti i jedan pomoćni privatni konstruktor koji kreira objekat tipa "`ChebyshevApproximation`" direktno na osnovu poznatih koeficijenata $c_k, k = 0, 1, ..., n$).

Metoda "`integrate`" sa dva parametra vrši procjenu vrijednosti određenog integrala funkcije f na intervalu $[a, b]$ gdje se a i b zadaju kao parametri (naravno, koristeći ponovo Chebyshevljeve polinome), dok metoda "`integrate`" bez parametara daje procjenu vrijednosti određenog integrala funkcije f na

čitavom intervalu $[x_{\min}, x_{\max}]$ koristeći Fejérov algoritam (naravno, treba koristiti činjenicu da su svi neophodni koeficijenti već proračunati). Metoda "`integrate`" sa dva parametra treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Bad interval" ukoliko interval $[a, b]$ nije sadržan unutar intervala $[x_{\min}, x_{\max}]$. Treba dopustiti i mogućnost da granice integracije budu u inverznom poretku,tj. da je $a > b$, pri čemu se u tom slučaju vraća negirana vrijednost integrala kakav bi bio da su granice u ispravnom poretku (što je inače svojstvo određenog integrala poznato iz matematičke analize).

Obavezno napišite i testni glavni program u kojem ćete testirati razvijenu klasu. Najbolje je da za tu svrhu izvršite Chebyshevljevu aproksimaciju neke jednostavne funkcije poput $f(x) = \sin x$ na intervalu $[0, \pi]$ čije se tačne vrijednosti, kao i tačne vrijednosti izvoda i integrala lako nalaze.

Zadatak 2 (2,5 poena)

U ovom zadatku, potrebno je implementirati tri globalne funkcije za podršku numeričkoj integraciji nazvane "`RombergIntegration`", "`TanhSinhIntegration`" odnosno "`AdaptiveIntegration`", koje respektivno implementiraju Rombergov algoritam, zatim algoritam zasnovan na Tahakasi-Mori tanh-sinh pravilu, te adaptivni Simpsonov algoritam numeričke integracije.

Funkcija "`RombergIntegration`" treba da ima sljedeći prototip:

```
template <typename FunType>
  std::pair<double, bool> RombergIntegration(FunType f, double a, double b,
    double eps = 1e-8, int nmax = 1000000, int nmin = 50);
```

Ova funkcija treba da vrati kao rezultat numeričku aproksimaciju određenog integrala funkcije f na intervalu $[a, b]$ koristeći Rombergov algoritam (f, a i b se zadaju putem prva 3 parametra). Slično kao kod ranije opisane metode "`integrate`", i ovdje treba podržati mogućnost da bude $a > b$. Četvrti parametar predstavlja zadanu toleranciju ε (pri čemu je podrazumijevana vrijednost 10^{-10}). Račun se prekida ukoliko se postigne da u dvije susjedne iteracije modul razlike procijenjenih vrijednosti integrala postane manji od ε. Posljednja dva parametra su respektivno maksimalni i minimalni dozvoljeni broj podjela intervala integracije (podrazumijevane vrijednosti su 1000000, što dopušta oko 20 iteracija, odnosno 50, što garantira barem 6 iteracija). Računanje se prekida kada se premaši maksimalni dozvoljeni broj podjela, čak i ukoliko pri tome nije postignuta odgovarajuća tačnost. Kao rezultat funkcije, vraća se uređeni par čija je prva koordinata procijenjena vrijednost integrala, a druga koordinata logička vrijednost "tačno" ili "netačno", ovisno da li je tražena tačnost postignuta ili ne. Ukoliko je parametar "`eps`" negativan, ukoliko su parametri "`nmin`" i "`nmax`" negativni, ili ukoliko je vrijednost "`nmax`" manja od vrijednosti "`nmin`", treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Bad parameter".

Funkcija "`TanhSinhIntegration`" treba da ima sljedeći prototip:

```
template <typename FunType>
  std::pair<double, bool> TanhSinhIntegration(FunType f, double a, double b,
    double eps = 1e-8, int nmax = 1000000, int nmin = 20, double range = 3.5);
```

Ova funkcija je slična prethodnoj, samo računa integral koristeći metod koji se zasniva na primjeni smjene Tahakasi-Mori (poznate kao tanh-sinh ili dvostruko eksponencijalno pravilo) na podintegralnu funkciju, a zatim na primjeni trapeznog pravila na tako dobijenu novu podintegralnu funkciju. Pri tome, treba koristiti strategiju po kojoj se broj intervala udvostručuje polazeći od $N = 2$ dok se ne postigne tražena tačnost. Parametri "`f`", "`a`", "`b`", "`eps`", "`nmax`" i "`nmin`" imaju isto značenje kao u prethodnoj funkciji (i ovdje se dozvoljava da bude $a > b$, a za legalnost parametara vrijede isti uvjeti kao i u prethodnoj funkciji). Međutim, kako je ova funkcija između ostalog namijenjena i za računanje nekih nesvojstvenih integrala koji mogu imati singularitete u granicama integracije, ukoliko se pri računanju dogodi da je vrijednost (transformirane) podintegralne funkcije u nekoj tački beskonačna ili ne-broj, tu tačku treba ignorirati (tj. treba je tretirati kao da je vrijednost funkcije u toj tački jednaka nuli).

Kako smjena Tahakasi-Mori transformira integral u novi integral s beskonačnim granicama, ali pri čemu vrijednosti nove podintegralne funkcije rapidno opadaju kako argument teži u beskonačnost, transformirani integral treba računati kao da je u granicama od $-R$ do R, pri čemu je R pozitivna vrijednost koja se zadaje putem parametra "`range`" (ukoliko se zada negativna vrijednost, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Bad parameter"). Podrazumijevana vrijednost ovog parametra je 3.5, što se pokazalo kao sasvim dobro za gotovo sve primjene. Slično kao i kod funkcije

"RombergIntegration", kao rezultat se vraća uređeni par čija je prva koordinata procijenjena vrijednost integrala, a druga koordinata logička vrijednost "tačno" ili "netačno", ovisno da li je tražena tačnost postignuta ili ne.

Konačno, funkcija "AdaptiveIntegration" treba da ima sljedeći prototip:

```
template <typename FunType>
  std::pair<double, bool> AdaptiveIntegration(FunType f, double a, double b,
    double eps = 1e-10, int maxdepth = 30, int nmin = 1);
```

Ova funkcija daje numeričku aproksimaciju određenog integrala funkcije f na intervalu $[a, b]$ koristeći adaptivni Simpsonov algoritam (f, a i b se zadaju putem prva 3 parametra, a dozvoljeno je da bude $a > b$). Potrebno je implementirati optimiziranu verziju, koja kreće od dvije osnovne procjene dobijene Simpsonovom formulom na osnovu dvije odnosno četiri podjele intervala integracije i u kojoj se ne javljaju nikakva suvišna izračunavanja. Četvrti parametar predstavlja toleranciju ε, koja služi za odluku da li treba ići u dalje podjele (pri čemu je podrazumijevana vrijednost 10^{-10}). Nove podjele se ne vrše ukoliko se dvije osnovne procjene razlikuju po apsolutnoj vrijednosti za manje od ε. Koristite strategiju u kojoj se u svaki dalji rekurzivni poziv prosljeđuje polovica tolerancije kakva je bila u tekućem rekurzivnom pozivu. Peti i šesti parametar predstavljaju respektivno maksimalnu dopuštenu dubinu rekurzije (podrazumijevano 30), te inicijalni broj podjela na koji se dijeli interval integracije prije nego što se započne sa rekurzivnim pozivima (podrazumijevano 1). Ukoliko se dostigne maksimalna dopuštena dubina rekurzivnih poziva, dalje dijeljenje se obustavlja neovisno o tome da li je postignuta tražena tolerancija ili ne. Parametri "eps", "maxdepth" i "nmin" moraju biti pozitivni, inače treba baciti izuzetak tipa "domain_error" uz prateći tekst "Bad parameter". Slično kao i kod prethodne funkcije, i ova funkcija se može koristiti za računanje nekih nesvojstvenih integrala. Zbog toga, ukoliko se dogodi da je vrijednost podintegralne funkcije u nekoj tački beskonačna ili ne-broj, tu tačku treba ignorirati, odnosno tretirati je kao da je u njoj vrijednost funkcije jednaka nuli. Kao rezultat, funkcija "AdaptiveIntegration" vraća uređeni par čija je prva koordinata procijenjena vrijednost integrala, dok je druga koordinata logička vrijednost "tačno" ili "netačno", ovisno da li je cilj postavljen putem tolerancije ε dostignut ili ne. Tačnije, ova vrijednost će biti "tačno" ukoliko se tokom postupka nikada ne dogodi da moramo prekinuti dalje dijeljenje zbog dostizanja maksimalne dopuštene dubine rekurzivnih poziva, a u suprotnom će biti "netačno".

Obavezno napišite i testni glavni program u kojem ćete testirati napisane funkcije. Testiranje obavite na više različitih funkcija za koje su vam tačne vrijednosti integrala poznate. Neka jedna od testnih funkcija bude $f(x) = \sin x$ na intervalu $[0, \pi]$, a za posljednje dvije funkcije (Tahakasi-Mori integracija i adaptivna integracija) uzmite i testnu funkciju $f(x) = 1/\sqrt{x}$ na intervalu $[0, 1]$ (ovo daje nesvojstveni integral). Druge testne funkcije odaberite sami.

NAPOMENA:

Naučite se da testirate programe sami, nemojte čekati na autotestove da vidite da li Vam je nešto ispravno ili ne. Pregledanje zadaće će se vršiti kombinacijom autotestova i ručnog pregledanja. Za metodu, funkciju ili operator koji nisu temeljito testirani u glavnom programu smatraće se da nisu urađeni, te se za njih neće dobiti nikakvi bodovi!