Objektno orijentirano programiranje C++

Predavanje 01 - uvod

Matej Mihelčić

Prirodoslovno-matematički fakultet
Matematički odsjek

04. ožujka 2024.

Predavač

Matej Mihelčić
matmih@math.hr
Konzultacije: utorkom 9-11h, soba 226[1]

[1] najaviti se mailom

Asistent

Sebastijan Horvat
sebastijan.horvat@math.hr

Osnovno o kolegiju

Web stranica:

`https://web.math.pmf.unizg.hr/nastava/opepp/`

Polažu:

- Preddiplomski sveučilišni studij Matematika - 3. godina (izborni kolegij)
- Diplomski sveučilišni studij Računarstvo i matematika - 1. godina (obavezni kolegij)
- Diplomski sveučilišni studij Matematika i informatika - 2. godina (izborni kolegij)
- Diplomski sveučilišni studij Matematika, nastavnički - 2. godina (izborni kolegij)

Nastava:

- 2 sata predavanja
- 2 sata vježbi

Način polaganja

Elementi ocjenjivanja:

- 6 domaćih zadaća (60 bodova).
- Završni ispit (40 bodova).
- Završni ispit se može pisati u svakom terminu ispitnih rokova.

Kako ostvariti prolaznu ocjenu?

- Ostvariti barem 50 bodova na zadaćama/završnom ispitu.

Ocjene

Bodovi	Ocjena
$[50, 60 >$	dovoljan (2)
$[60, 75 >$	dobar (3)
$[75, 86 >$	vrlo dobar (4)
$86+$	izvrstan (5)

Osnovni ciljevi i sadržaj kolegija

Osnovni ciljevi su:

- Detaljnije upoznavanje jezika C++, usvajanje naprednijih koncepata objektno orijentiranog i generičkog programiranja.
- Napredno korištenje standardne biblioteke STL.
- Tehnike programiranja i softverski predlošci.
- Korištenje poznatijih eksternih C++ biblioteka.

Osnovna literatura

Osnovna literatura za predavanja i vježbe:
Slideovi na web stranicama kolegija
(`https://web.math.pmf.unizg.hr/nastava/opepp/`)

Literatura za programiranje u C++-u:
Stanley B. Lippman, Josée Lajoie, Barbara E. Moo: C++ Primer, Fifth Edition, Addison Wesley Professional, 2012.

Savjeti

- **Redovito** pohađati i prolaziti materijale predavanja i vježbi.
- Samostalno rješavati zadatke i domaće zadaće.
- Ponavljati gradivo kolegija prethodnika.

Povijest programskog jezika C++

Programski jezik C++ razvio je Bjarne Stroustrup, zaposlenik Bell Laboratories.

- Razvoj jezika je započeo 1979. godine, a 1983. godine je jezik imenovan C++.
- 1985. godine je izašla prva verzija jezika C++.
- 1989. godine izlazi verzija 2.0 jezika C++.
- 1991. izlazi poboljšanja verzija C++-a 2.0.
- 1992. godine je u jezik ubačen STL.
- 1998. godine izlazi standardizirana verzija 3.0 jezika C++ nazvana ISO/IEC 14882:1998.
- 2003. godine izlazi standardizirana verzija C++03 nazvana ISO/IEC 14882:1998.
- 2011. godine izlazi znatno nadograđena verzija C++11, nazvana ISO/IEC 14882:1998.

Povijest programskog jezika C++

- 2014. godine je dorađen standard C++11, čime nastaje C++14.
- 2017. godine je napravljena velika revizija standarda iz 2011. godine. Tako nastaje C++17.
- 2020. godine je napravljena velika revizija standarda iz 2017. godine. Tako nastaje C++20.
- Sljedeća velika nadogradnja standarda je planirana pod radnim imenom (C++23).

Za informacije o standardu vidjeti: `isocpp.org/std/`

Na kolegiju radimo C++11. Novosti (C++17) će se raditi na kolegiju Napredni C++ profesora Juraka (zimski semestar). Također na stranici `web.math.hr/nastava/opepp/old/` možete naći stare materijale prof. Juraka.

Karakteristike jezika

- Jezik opće namjene visoke efikasnosti.
- Omogućava programiranje niske razine i programiranje s apstrakcijama.
- Slijedi princip da apstrakcija ne smije negativno utjecati na efikasnost koda.

Prevodioci

- GCC (g++) - Linux i Windows (MinGW)
- Clang - Linux i Windows
- Microsoft Visual C++ - Windows (komercijalan)
- AMD Optimizing C/C++ compiler (AOCC) - Linux, optimiziran za AMD platforme Epyc i Ryzen (komercijalan)

Softverske biblioteke

Svaki prevodioc za C++ sadrži standardnu biblioteku `STL` (standard template library).

Neke značajnije biblioteke za C++ su:

- `Boost`, `www.boost.org`. Boost sadrži niz paketa koji podržavaju računanje iz područja linearne algebre, generiranja pseudoslučajnih brojeva, višedretvenog programiranja, obrade slika, regularnih izraza, obrada stringova i teksta te testiranje softvera.
- `Poco`, `https://pocoproject.org/`. Poco je namijenjen izradi portabilnih mrežnih i Internet aplikacija.
- `OpenSSL`, `https://www.openssl.org/`. OpenSSL sadrži biblioteke za kriptografiju i sigurnu komunikaciju.

Softverske biblioteke

- `FFmpeg`, `https://ffmpeg.org/`. FFMpeg je biblioteka za obradu audia i videa.
- `SQLite`, `https://www.sqlite.org/cintro.html`. Omogućava rad sa SQL bazom koristeći C++.
- `JSON for Modern C++`, `https://json.nlohmann.me/`. Omogućava rad sa JSON objektima koristeći C++.
- `OpenCV`, `https://opencv.org/`. Sadrži biblioteke za računalni vid, stojno i duboko učenje te prepoznavanje lica, detekciju objekata i ekstrakciju 3-D modela.
- `Tensorflow`, `https://www.tensorflow.org/`. Jedna od najpoznatijih biblioteka za duboko učenje.

Softverske biblioteke

Neke biblioteke za izradu grafičkog sučelja u C++-u:

- Qt, `https://www.qt.io/`
- WxWidgets, `https://www.wxwidgets.org/`
- Fltk, `https://www.fltk.org/`
- Juce, `https://juce.com/`
- Ultimate++, `https://ultimate.apponic.com/`
- Dear ImGui, `https://www.dearimgui.org/`

Integrirane okoline za razvoj programa

Integrirane okoline za razvoj programa sadrže editor za pisanje programa i različite alate za prevođenje, analizu, ispravljanje programa itd. Često integriraju i sustav za upravljanjem kodom i sustav za distribuciju koda.

- Visual Studio: `msdn.microsoft.com/en-us/vstudio`
- Visual Studio Express: `visualstudio.microsoft.com/vs/express/`
- Qt Creator: `www.qt.io`
- Visual Studio Code: `code.visualstudio.com`
- Code::Blocks: `www.codeblocks.org`
- NetBeans IDE: `netbeans.org`
- Eclipse CDT: `www.eclipse.org/cdt`
- CodeLite: `www.codelite.org`

Objektno orijentirano programiranje

Programska paradigma koja koristi **objekte** za reprezentaciju glavnih elemenata (aktera) problema.

Softversko inženjerstvo uglavnom proučava složene aplikacije kod kojih se mogu javiti problemi s:

- Održavanjem
- Proširivosti
- Prenosivosti

Proceduralna:

- Problem se razlaže na manje cjeline - implementacija pomoću funkcija (procedura).
- Nije moguće definirati apstrakcije koncepata iz aplikacijske domene.
- Nema podrške za postizanje održivosti i proširivosti.

Objektno orijentirana:

- Klase predstavljaju koncepte iz aplikacijske domene.
- Program se sastoji od objekta (instance klasa) i njihove komunikacije.
- Sadržaj i moguće akcije nad objektom ili samog objekta su specificirane definicijom objekta.

Apstrakcija - koncepti iz aplikacijske domene se reprezentiraju neposredno u programu, a izvedbeni detalji su skriveni iza sučelja (eng. interface) koje reprezentira koncept.

```
1  #include <iostream>
2
3  class tocka{
4  private: int a,b;
5
6  public:
7
8  void postavi(int x, int y){   a = x; b = y; }
9
10 void ispisi(){ std::cout<<a<<"␣"<<b<<std::endl;}
11 };
```

Osnovni elementi objektno orijentirane tehnike

```
1  int main(void){
2
3  tocka prva;
4
5  prva.postavi(4,5);
6  prva.ispisi();
7  }
```

Enkapsulacija - sposobnost osiguravanja da se apstrakcija koristi prema svojim specifikacijama. Enkapsulacijom se sprečava narušavanje apstrakcije, odnosno prodiranje implementacijskih odluka izvan granica apstrakcije.

Pretpostavimo da želimo promijeniti implementaciju klase točka.

```
1  #include <iostream>
2
3  class tocka{
4  private: int c,d;
5
6  public:
7
8  void postavi(int x, int y){   c = x; d = y; }
9
10 void ispisi(){ std::cout<<c<<"␣"<<d<<std::endl;}
11 };
```

Gornji main radi bez greške neovisno o promjenama!

Polimorfizam - skrivanje različitih implementacija iza istog sučelja. Osigurava široku primjenjivost koda i reducira zavisnost o implementaciji.

Osnovni primjer polimorfizma

```
1  #include <iostream>
2
3  int main(void){
4      int a = 2, b = 3;
5      std::string c = "aba", d = "baba";
6
7      std::cout<<(a+b)<<std::endl;
8      std::cout<<(c+d)<<std::endl;
9
10     return 0;
11 }
```

Predefiniranje funkcije

```
1  class tocka{
2  private: double c,d;
3
4  public:
5
6  void postavi(double x, double y){  c = x; d = y;}
7  void pomakni(int x, int y){ c+=x; d+=y;}
8  void pomakni(double x, double y){c+=x; d+=y;}
9  void ispisi(){ std::cout<<c<<" "<<d<<std::endl;}
10 };
```

U polimorfizme spadaju i nadjačavanje operatora i funkcija (kod nasljeđivanja), virtualne funkcije.

Nasljeđivanje - konstrukcija novih apstrakcija polazeći od već postojećih.

Nasljeđivanje

```
1  class tocka{
2  protected: double c,d;
3
4  public:
5
6  tocka(double x, double y){c = x; d = y;}
7
8  void postavi(double x, double y){  c = x; d = y;}
9  void pomakni(int x, int y){ c+=x; d+=y;}
10 void pomakni(double x, double y){c+=x; d+=y;}
11 void ispisi(){ std::cout<<c<<" "<<d<<std::endl;}
12 };
```

Nasljeđivanje

```
1 class tockaNaX:tocka{
2   public:
3     tockaNaX(double x):tocka(x,0){};
4 };
```

Generičko programiranje - podrazumijeva generalizaciju softverskih komponenti kako bi se lako mogle koristiti u različitim situacijama. Osnovni elementi generičkog programiranja u C++-u su parametrizirane klase i funkcije.

Generičko programiranje

```
1  #include <iostream>
2
3  template<class T> class tocka{
4        T c, d;
5
6  public:
7     void postavi(T x, T y){  c = x; d = y; }
8     void ispisi(){
9                  std::cout<<c<<" "<<d<<std::endl;}
10 };
```

Generičko programiranje

```
1  int main(void){
2
3  tocka<int> a;
4  a.postavi(2,3);
5
6  tocka<double> b;
7  b.postavi(2.2,3.3);
8
9  tocka<std::string> c;
10 c.postavi("prva","druga");
11
12 a.ispisi(); b.ispisi(); c.ispisi();
13
14     return 0;
15 }
```