Objektno orijentirano programiranje C++

Predavanje 01 - uvod

Marko Živković

Prirodoslovno-matematički fakultet
Matematički odsjek

3. ožujka 2026.

Predavanja

Marko Živković
marko.zivkovic@math.hr
Konzultacije: utorkom 14-15 i 17-18h, soba 306[1]

Predavanja se baziraju na prijašnjim predavanjima prof. Juraka i doc. Mihelčića

Web stranica prošlogodišnjeg kolegija:
https://web.math.pmf.unizg.hr/nastava/opepp/

[1] najaviti se mailom

Osnovno o kolegiju

Polažu:

- Preddiplomski sveučilišni studij Matematika - 3. godina (izborni 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 u praktikumu

Način polaganja

Elementi ocjenjivanja:

- 5 domaćih zadaća (50 bodova).
- Završni ispit (50 bodova).
- Završni ispit se može pisati u svakom terminu ispitnih rokova.

Uvjeti za pristup ispitu:

- sudjelovanje na najmanje 70% predavanja i vježbi,
- predati barem 4 zadaće i na barem dvije ostvariti 20% bodova.

Ocjene

Ocjena se dobiva na temelju zbroja bodova zadaća i ispita.

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 s predavanja i vježbi

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

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.
- 2023. godine je napravljena trenutna verzija standarda, C++23.
- Sljedeća velika nadogradnja standarda je planirana pod radnim imenom (C++26).

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).

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`

Tehnike programiranja

Proceduralna:

- **Fokus na funkcijama** – program je organiziran kao niz funkcija koje izvršavaju korake nad podacima.
- **Podaci i funkcije su odvojeni** – podaci se često prenose između funkcija kao argumenti.
- **Tok izvođenja je sekvencijalan** – program se izvršava korak-po-korak kroz funkcije.

Objektno orijentirana:

- **Fokus na objektima** – program se organizira oko objekata iz klase koja sadrži i podatke i funkcije.
- **Enkapsulacija podataka i ponašanja** – podaci i funkcije koje rade nad njima nalaze se zajedno u klasi.
- **Koncepti poput nasljeđivanja i polimorfizma** – klase mogu dijeliti i proširivati ponašanje.

Primjer klase

```
1  class tocka{
2      int a, b;
3
4  public:
5      void postavi(int x, int y) {
6          a = x;
7          b = y;
8      }
9
10     void pomakniDesno(int x) {
11         a += x;
12     }
13
14     void ispisi() {
15         std::cout << a << "␣" << b << std::endl;
16     }
17 };
```

Korištenje klase

```
1  int main(void){
2      tocka prva;
3      prva.postavi(4,5);
4      prva.ispisi();
5
6      prva.pomakniDesno(2);
7      prva.ispisi();
8  }
```

Preopterećenje funkcija

Preopterećenje je korištenje istog naziva za funkcije koje se razlikuju u parametrima (broju, tipu).
Npr. klasa `tocka` može imati dodatnu funkciju za postavljanje točke na x osi:

```
1 void postavi(int x) {
2     a = x;
3     b = 0;
4 }
```

Enkapsulacija

Enkapsulacija - Enkapsulacija znači skrivanje podataka unutar klase i kontroliranje pristupa tim podacima putem `public` funkcija.

Specifikatori pristupa:

- `private` - podaci i funkcije dostupne samo unutar klase
- `public` - podaci i funkcije dostupne izvan klase
- `protected` - podaci i funkcije dostupni unutar klase i u klasama koje je nasljeđuju.

Ako se ne navedu, u klasama je default `private`.

```
1  class tocka{
2  private:
3      int a, b;
4
5  public:
6      ...
7  };
```

Enkapsulacija

Sljedeći kod javlja grešku jer su `a` i `b` `private` u klasi `tocka`:

```
1 int main(void){
2     tocka prva;
3     prva.a = 4;
4     prva.b = 5;
5 }
```

Mora se koristiti `public` funkcija `postavi()`.

Enkapsulacija služi da se zaštite podaci, ali i da se forsira ispravno i čitko pisanje koda.

Enkapsulacija

Enkapsulacija omogućuje da se promijeni unutarnja logika bez promjene vanjskog ponašanja.

```
1  class tocka{
2      int zbroj, b;
3
4  public:
5      void postavi(int x, int y) {
6          zbroj = x + y;
7          b = y;
8      }
9
10     void pomakniDesno(int x) {
11         zbroj += x;
12     }
13
14     void ispisi() {
15         std::cout << zbroj - b << " " << b << std::endl;
16     }
17 };
```

Polimorfizam

Polimorfizam - ista funkcija može imati različito ponašanje ovisno o objektu nad kojim se poziva.

U polimorfizme spada i nadjačavanje operatora.

polimorfizma operatora +

```
1  int main(void){
2      int a = 2, b = 3;
3      std::string c = "aba", d = "baba";
4
5      std::cout << (a + b) << std::endl;
6      std::cout << (c + d) << std::endl;
7  }
```

Nasljeđivanje

Nasljeđivanjem jedna klasa preuzima podatke i funkcije druge klase. Nova klasa može dodavati nove podatke i funkcije i mijenjati postojeće.

```
1  class tocka3D : public tocka{
2      int c;
3
4  public:
5      void postavi(int x, int y, int z) {
6          postavi (x, y);
7          c = z;
8      }
9
10     void ispisi() {
11         std::cout << a << "␣" << b << "␣" << c << std::endl;
12     }
13 };
```

Da bismo mogli pristupiti podacima a i b u baznoj klasi, one moraju biti deklarirane sa specifikatorom pristupa barem protected:

```
1 class tocka{
2 protected:
3     int a, b;
4
5 public:
6     ...
7 };
```

Nasljeđivanje

```
1  int main(void) {
2      tocka prva;
3      prva.postavi(4, 5);
4      prva.pomakniDesno(2);
5      prva.ispisi();
6
7      tocka3D druga;
8      druga.postavi(3, 6, 4);
9      druga.pomakniDesno(1);
10     druga.ispisi();
11 }
```

Primijetimo da je funkcija `pomakniDesno()` iz klase `tocka` naslijeđena i može se koristiti i u podklasi `tocka3D`.
Primijetimo polimorfizam funkcije `ispisi()`. Isti se naziv koristi za različito ponašanje ovisno o tome kojoj klasi pripada objekt.

Generičko programiranje

Generičko programiranje omogućuje pisanje funkcija i klasa koje rade s različitim tipovima podataka.

```
1  template<class T> class tocka {
2      T a, b;
3
4  public:
5      void postavi(T x, T y) {
6          c = x;
7          d = y;
8      }
9
10     void ispisi() {
11         std::cout<<c<<" "<<d<<std::endl;
12     }
13 };
```

Generičko programiranje

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