

PJ1 USMENI ISPITI

S A D R Ž A J

† 24. 01. 2022	3
† 18.04.2022	5
† 29. 08. 2022	7
† 06. 02. 2023	9
† 12. 06. 2023	11



24. 01. 2022

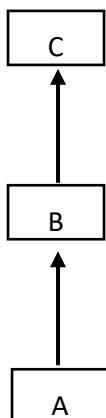
Programski jezik C++
Završni ispit (24.01.2022.)

Ispit se radi **90 minuta**.

1. Šta ne nasljeđuje izvedena klasa?
2. Koje su osobine relacije prijateljstva?
3. Koji *header* file je potrebno uključiti da bi mogli da koristimo funkciju **find** nad nekim objektom kontejnerske klase **vector**?
4. Klasa **A** je izvedena iz klase **B**, a klasa **B** iz klase **C**. Kojim redoslijedom treba poredati **catch** klauzule čiji su tipovi redom **A**, **B** i **C**?
5. Kako treba da izgleda drugi argument konstruktora kopije?
6. Šta treba uraditi da izvedena klasa posjeduje samo jedan podobjekat indirektne osnovne klase?
7. Koji je podrazumijevani način izvođenja klase?
8. Koje osobine operatora ne mogu da se mijenjaju?
9. Kog tipa će biti izuzetak koji je kreiran samo pozivom **throw**; (bez izraza iza **throw**)?
10. Nabrojati i objasniti vrste specijalizacije šablona.

ODGOVORI:

1. Izvedena klasa ne nasljeđuje:
 - Konstruktore osnovne klase
 - Destruktor osnovne klase
 - Operator dodjele '=' osnovne klase
2. Relacija prijateljstva se ne nasljeđuje, nije simetrična ni tranzitivna relacija
3. Potrebno je uključiti header **<algorithm>**
4. Dijagram izgleda ovako (lanac izvođenja):

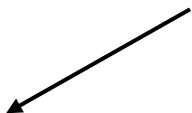


Što znači, generalno se preporučuje da se **catch** klauzule poredaju od najspecifičnijeg do najopštijeg tipa izuzetka. U ovom konkretnom slučaju, **redoslijed** **catch** klauzula trebao bi biti **A, B, C**.

Razlog za ovakav redoslijed je da catch blokovi obrade izuzetaka trebaju biti poredani tako da najopštiji tip izuzetka bude na kraju (tip za osnovnu/baznu klasu), jer će se prvo provjeriti klauzule koje se odnose na specifičnije tipove izuzetaka (izvedene klase). Ako bi redoslijed bio suprotan, tj. C, B, A, catch klauzula za izuzetak tipa C bi uhvatila sve izuzetke, uključujući one koji pripadaju tipovima B i A, te catch klauzule za tipove B i A ne bi bile **nikada dosegnute**.

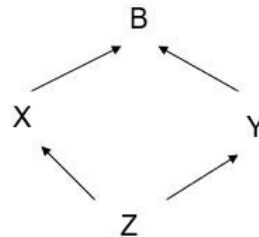
5. Drugi argument konstruktora kopije mora da ima **podrazumijevane vrijednosti**! Npr. ako postoji klasa MyClass:

```
class MyClass{    int myClassValue;
//polje
    MyClass(const MyClass& other, value = 1){
        //...
    }
}
```



6. Ovo je opis dijamant problema! Da bi izvedena klasa posjedovala samo jedan podobjekat indirektno osnovne klase, ta indirektna osnovna klasa se mora naslijediti kao **virtuelna**!

```
class B { /*...*/ };
class X : virtual public B { /*...*/ };
class Y : virtual public B { /*...*/ };
class Z : public X, public Y { /*...*/ };
```



7. Podrazumijevani način izvođenja klase je **private**.
8. Osobine operatora koji se ne mogu mijenjati su **asocijativnost**, **operandi** operatora i **prioritet** operatora.
9. Kada se koristi samo **throw**; bez izraza iza njega, tada će se trenutni izuzetak ponovno baciti dalje bez promjene njegovog tipa. Drugim riječima, tip izuzetka će ostati isti kao i tip izuzetka koji je bio trenutno aktiviran prilikom tog **throw**; izraza (odnosno, biće propagiran dalje sa istim tipom).
10. Postoje dvije vrste specijalizacije šablona:
- **Djelimična specijalizacija** – šablon ima bar jedan parametar, pa se može generisati više klasa (instanci) iz specijalizovanog šablona.
 - **Potpuna specijalizacija** – šablon nema ni jedan parametar, ali se može generisati samo JEDNA instanca šablona (samo jedna klasa npr). Funkcije mogu imati samo ovu specijalizaciju!



18.04.2022

Programski jezici 1 – završni ispit (18.04.2022.)

Ispit se radi 60 minuta.

1. Kako se definiše virtuelna statička članica klase?
2. Kako treba realizovati funkciju `Zakon operator+(int kazna, Osoba v)?`
3. Kako se kreiraju imena operatorskih funkcija?
4. Koje je dejstvo metode `seekg`?
5. Iz kojih funkcija se ne mogu izazvati izuzeci?
6. Šta su iteratori?
7. Šta je apstraktna klasa?
8. Koje vrste specijalizacije su moguće kod generičkih funkcija?
9. Karakteristike privatnog nasljeđivanja.
10. Šta se dobija primjenom specifikatora `explicit` ispred konstruktora?

ODGOVORI:

1. Virtuelna članica klase **ne može biti statička!** (trik pitanje !)
2. Ovo **nije dobar potpis** preklopljenog operatora '+', jer preklopljeni operator se može realizovati na 2 načina:
 - Kao globalna funkcija (tada prima 2 argumenta odgovarajućeg tipa) – `Zakon operator+(const Zakon& lhs, const Zakon& rhs);`
 - Kao funkcija članica (tada prima jedan argument odgovarajućeg tipa, lijevi argument je trenutni objekat) – `Zakon operator+(const Zakon& rhs);`
3. Imena operatorskih funkcija se sastoje od odgovarajućeg povratnog tipa (<tip>), ključne riječi **operator** nakon čega ide odgovarajući operator za preklapanje (<oper.>), a zatim argumenti koje funkcija prima:

`<tip> operator<oper.>(<argumenti>);`
4. Metoda **seekg**(long pozicija) se koristi za postavljanje pozicije čitanja u ulaznu datoteku za broj određen argumentom 'pozicija'. Odnosno, čitanje će krenuti od onog bajta u datoteci određen promjenljivom 'pozicija'. Drugim riječima, na tu poziciju se postavlja vrijednost pokazivača koja će biti pročitana sljedeća.
5. Ne mogu se izazvati izuzeci iz funkcija:
 - koji su deklarirani uz ključnu riječ **noexcept**,
 - **Defaultni konstruktori** i **move konstruktori**: Defaultni konstruktori i move konstruktori koji se generišu automatski od strane kompajlera obično su bezbjedni od izuzetaka.
 - Destruktori klasa (osim ako sadržavaju eksplicitno ključnu riječ **throw**)

6. Iteratori su klase čiji objekti predstavljaju pozicije elemenata u različitim kontejnerskim klasama. Oni se uvijek pridružuju nekoj klasi, a tip zavisi od kontejnerske klase kojoj su pridruženi.
7. Apstraktna klasa je klasa koja ima bar jednu čistu virtuelnu metodu. Čisto virtuelna metoda je funkcija članica koja je deklarirana sa `= 0` na kraju svog potpisa, nema tijelo.
8. Generičke funkcije se mogu samo **potpuno specijalizovati**.
9. Kod privatnog nasljeđivanja privatni članovi nadklase su nepristupačni podklasi, dok će javni i zaštićeni članovi nadklase biti „promovisani“ u privatne članove u podklasi.
10. Specifikator **explicit** onemogućava implicitnu konverziju konstruktora (sprječava njegovo implicitno ponašanje), odnosno moguće je samo eksplicitno navesti konstruktor neke klase (mora biti jasno naveden).



29. 08. 2022

Programski jezici 1 – završni ispit (29.08.2022.)

Ispit se radi **60 minuta**.

1. Objasniti namjenu funkcije **set_unexpected**.
2. Objasniti dejstvo funkcije **istream& read(char* niz, int broj)**.
3. Kako se koriste funkcije kao parametri šablona?
4. Da li destruktork može biti virtuelna funkcija?
5. Objasniti djelovanje specifikatora **final**.
6. Šta se podrazumijevano kreira pri definisanju izvedene klase i koje je ponašanje?
7. Ako lijevi operand operatorske funkcije treba da bude standardnog tipa, kako se mor definisati ova funkcija?
8. Koji operatori ne mogu da se preklapaju?
9. Koliko argumenta može da ima konstruktor kopije?
10. Redoslijed poziva konstruktora i destruktora za automatske lokalne objekte.

ODGOVORI:

1. **std::set_unexpected** je funkcija definisana u biblioteci **<exception>**. Ona se koristi za postavljanje funkcije koja će biti pozvana u slučaju neuhvaćenog izuzetka tipa **std::unexpected_exception**. Odnosno, ako se u nekoj funkciji izazove izuzetak koji nije na spisku naznačenih izuzetaka, izvršava se funkcija **void unexpected()**, zatim poziva se **unexpected_handler()** i podrazumijeva se da ova funkcija poziva funkciju **terminate()**. Ovo se može promijeniti pomoću funkcije **set_unexpected**. Njoj se dostavlja pokazivač na funkciju koju treba da pozove funkcija **unexpected** umesto **terminate**.
2. Funkcija **read** čita sa ulaznog toka niz karaktera od **broj** veličine, te ga smješta u niz **char* niz**. Funkcija će čitati karaktere počevši od trenutne pozicije u toku.
3. Funkcije kao parametri šablona (argumenti) se prenose kao **pokazivači na funkcije**.
4. Destruktor može biti virtuelna funkcija.
5. Specifikator **final** onemogućava izvođenje iz klase kojoj je pridružena ova ključna riječ nakon naziva klase.
6. Funkcija članica operatora = (vrši dodjelu član po član)

Podrazumijevani konstruktor (ima prazno tijelo)

Kopirajući konstruktor (vrši dodjelu vrijednosti član po član svog argumenta objektu koji se konstruiše)

Destruktor (ima prazno tijelo)

7. Ako lijevi operand operatorske funkcije treba da bude standardnog tipa (npr. **int**, **double**, **char**, itd), operatorska funkcija se mora definisati kao **globalna funkcija** ili kao **funkcija članica koja je deklarirana kao prijateljska funkcija** u klasi.
8. Ne mogu da se preklape operatori '.', '.*', '::', '?:' i **sizeof**
9. Konstruktor kopije može da ima jedan argument (konstantna referenca na drugi objekat istog tipa ili referenca na konstantni objekat istog tipa) ili 2 i više argumenata (u tom slučaju 2. ili n-ti argument **mora imati podrazumijevanu vrijednost**).
10. Redoslijed poziva konstruktora i destruktora za automatske lokalne objekte je određen njihovim životnim vijekom, odnosno opsegom u kojem su definisani. Kada se automatski lokalni objekat deklarirše unutar nekog opsega, konstruktor će se pozvati pri ulasku u taj opseg, dok će destruktor biti pozvan pri izlasku iz opsega.



06. 02. 2023

1. Karakteristike zaštićenog nasljeđivanja.
2. Kog je tipa povratna vrijednost postdekrement operatora?
3. Koji header file je potrebno uključiti da bi mogli da koristimo funkciju **sort** nad objektom kontejnerske klase **vektor**?
4. Nabrojati situacije u kojima se ignoriše zahtjev za inline funkcijama.
5. Šta ne nasljeđuje izvedena klasa?
6. Koje je podrazumijevano ponašanje konstruktora za kopiranje?
7. Šta se dobija primjenom specifikatora *explicit* ispred konstruktora?
8. U kojim slučajevima handler tipa **A** može da prihvati izuzetak tipa **B**?
9. Šta mogu da budu parametri šablonske klase?
10. Koji uslov treba da zadovolji poziv funkcije za koju ne postoji odgovarajuća definicija, da bi se generisala odgovarajuća funkcija na osnovu šablona?

ODGOVORI:

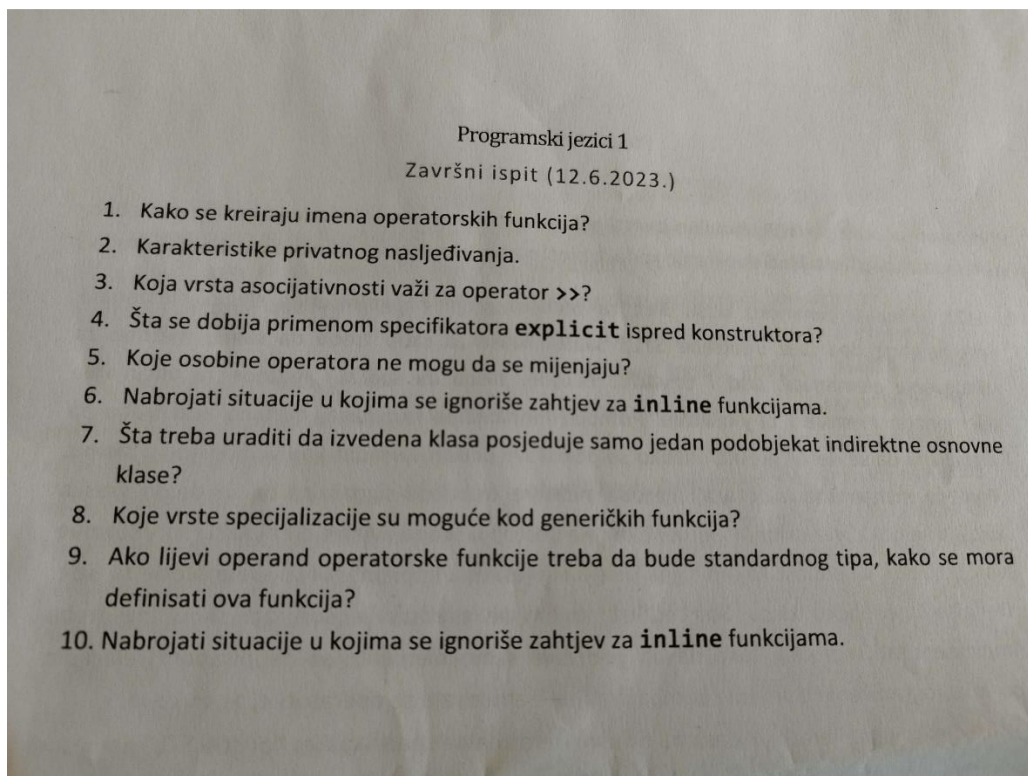
1. Ukoliko klasa naslijedi neku drugu klasu navodeći **protected** kao modifikator nasljeđivanja, svi **public** članovi bazne klase će biti „promovisani“ u **protected**, zaštićeni članovi će ostati zaštićeni, dok privatni članovi bazne klase će ostati privatni članovi (biće naslijeđeni, ali neće biti omogućen direktan pristup). Ako su neka polja/funkcije deklarirani kao **protected** u roditeljskoj klasi, jedino klase nasljednice će imati direktan pristup tim poljima/funkcijama.
2. Postdekrement operator treba da sačuva **kopiju** svog operanda, izmjeni trenutni operand, a zatim vrati kopiju tog operanda.
3. Potrebno je uključiti header file **<algorithm>**. Ukoliko kontejnerska klasa za vektor iz header file-a **<vector>** nije uključena, potrebno je i nju uključiti.
4. Postoje više situacija u kojima se ignoriše taj zahtjev:
 - Ukoliko funkcija sadrži neku **petlju** (for, while, do-while)
 - Ukoliko funkcija **nije** tipa **void** i nije napisana naredba za vraćanje vrijednosti (**return**)
 - Ako je **tijelo** funkcije **suviše dugačko**
 - Ukoliko u funkciji postoji **switch** ili **goto**
 - Ako je funkcija **rekurzivna**
 - Ako funkcija sadrži neke **statičke promjenljive**
5. Izvedena klasa ne nasljeđuje:
 - Preklopljeni operator=
 - Konstruktore osnovne klase
 - Destruktor osnovne klase

Međutim, klasa će dobiti implicitno deklarirane konstruktore, kao i destruktore, dok preklopljeni operator= će vršiti plitko kopiranje, član po član.

6. Podrazumijevano (default) ponašanje konstruktora za kopiranje je da vrši „plitko kopiranje“, član po član. Odnosno, ukoliko se uništi original, uništava se i kopija (i obrnuto!). Za dinamički alocirane objekte članove, vrijednosti pokazivača se kopiraju, tako da sami pokazivači u originalnom objektu i u kopiji pokazuju na istu adresu!
7. Ako se primijeni **explicit** ispred konstruktora, to onemogućava implicitno ponašanje konstruktora, tj. zabranjuje se implicitna konverzija ispred konstruktora. Drugim riječima, to znači da se konstruktor može pozvati samo eksplicitno, odnosno poziv konstruktora mora biti jasno naveden.
8. **Hendler** tipa **A** će da prihvati **izuzetak** tipa **B** ako:
 - Su A i B istog tipa
 - A je javna osnovna klasa za izvedenu klasu B
 - A i B su pokazivački tipovi i B može da se standardnom konverzijom konvertuje u tip A
9. Parametri šablonske klase mogu da budu:
 - Tipovi – **class**, **typename**, **lista_parametara**
 - Vrijednosti – ne mogu da budu floating point, ni korisnički definisanog tipa. Moraju biti globalni, kako bi bili poznati u trenutku kompajliranja. Ti vrijednosni parametri su:
 - Cjelobrojni tip
 - Enumeratori
 - Reference
 - Pokazivač na funkciju
 - Pokazivač na objekat
 - Pokazivač na članove
 - Drugi šabloni, tj. šabloni mogu da budu parametri šablona klase
10. Kada se naiđe na poziv funkcije za koju ne postoji definicija, a postoji odgovarajuća funkcija, prevodilac generiše odgovarajuću definiciju funkcije.



12. 06. 2023



ODGOVORI:

1. Imena operatorskih funkcija se sastoje od odgovarajućeg povratnog tipa (<tip>), ključne riječi **operator** nakon čega ide odgovarajući operator za preklapanje (<oper.>), a zatim argumenti koje funkcija prima:

`<tip> operator<oper.>(<argumenti>);`
2. Kod privatnog nasljeđivanja, svi **public** i **protected** članovi bazne klase će biti „promovisani“ u **private**, to znači da se ti članovi mogu koristiti samo unutar izvedene klase i nisu dostupni izvan nje. Privatni članovi će biti naslijeđeni, ali neće biti dostupni direktno.
3. Operator >> ima asocijativnost LTR (left-to-right).
4. Ako se primijeni **explicit** ispred konstruktora, to onemogućava implicitno ponašanje konstruktora, tj. zabranjuje se implicitna konverzija ispred konstruktora. Drugim riječima, to znači da se konstruktor može pozvati samo eksplicitno, odnosno poziv konstruktora mora biti jasno naveden.
5. Osobine operatora koji ne mogu da se mijenjaju su: asocijativnost, prioritet, broj operanada operatora (da li je unarni/binarni/ternarni).
6. Pogledati 06.02.2023, 4. zad
7. Ako je potrebno da izvedena klasa posjeduje samo jedan podobjekat indirektno osnovne klase onda **osnovnu klasu** treba **deklarirati** kao **virtuelnu** !!! (rješavanje dijamant problema)
8. Za generičke funkcije je moguća **samo potpuna specijalizacija** (trik pitanje!)

9. Ako lijevi operand operatorske funkcije treba da bude standardnog tipa (npr. **int**, **double**, **char**, itd), operatorska funkcija se mora definisati kao **globalna funkcija** ili kao **funkcija članica koja je deklarirana kao prijateljska funkcija** u klasi.
10. Ponovljeno pitanje br. 6.

PITANJA SA OSTALIH ROKOVA:

- 1. Koji su uslovi za kreiranje virtuelnog mehanizma?**
Objektu se pristupa preko pokazivača ili referenci
- 2. Ako je met() metod klase PK koja je prijateljska klasa klasi A, kojim atributima klasa A može da se pristupi u funkciju met()?**
Privatnim, zaštićenim i javnim članovima
- 3. Koja vrsta asocijativnosti važi za složene operatore dodjele?**
RTL
- 4. Redoslijed pozivanja konstruktora pri kreiranju objekata klase C izvedene iz A i B?**
Prvo se poziva konstruktor od klase A, pa B i na kraju od C
- 5. Objasni dejstvo operatora noexcept**
Operator (noexcept)
Rezultat je logičkog tipa (tačno- ne može da baci izuzetak)
Provjera da li bi mogao doći do izuzetka u izrazu
Izraz je proizvoljnog tipa, pa čak i void
Ne izračunava se, samo se provjerava u fazi prevođenja
- 6. Koji je maksimalan broj standardnih tokova koji se mogu kreirati u funkciji članica klase?**
Postoje 4 standardna toka: cin, cout, clog i cerr.
Ova 4 standardna toka se mogu kreirati neograničen broj puta
- 7. Kako treba da izgleda treći argument konstruktora kopije?**
Kao proizvoljna promjenljiva
- 8. Koja vrsta asocijativnosti se vrši za operator %=**
RTL
- 9. Koji su operatori podrazumijevano definisani za korisničke tipove?**
= (dodjela vrijednosti)
& (uzima adresu)



10. Koji je preduslov pozivanja konstruktora pri kreiranju izvedene klase?

Prvo se poziva konstruktor osnovne klase, onda se inicijalizuju podaci članovi pozivanjem njihovih konstruktora i na kraju se poziva konstruktor izvedene klase

11. Koliko se maksimalno može kreirati standardnih tokova?

Nula, ne kreiraju se jer su globalni statički objekti

12. Navesti operatore za pristup članovima klase?

→ (strelica) i . (tačka)

13. Iz kojih funkcija se mogu izazvati izuzeci?

Članice klase, konstruktori, destruktori i operatorske funkcije

14. Kog je tipa povratna vrijednost postdekrement operatora?

Postfiksni operator vraća const objekta MyClass, tj. kopiju *this koja je dekrementovana

15. Kog je tipa povratna vrijednost postinkrement operatora?

Postfiksni operator vraća const objekta MyClass, tj. kopiju *this koja je inkrementovana

16. Da li konstruktor ili destruktore mogu biti virtuelne funkcije?

Destruktor može, konstruktor ne može

17. Anonimni prostor imena

Globalni identifikator sa unutrašnjim povezivanjem

Identifikator iz anonimnog prostora imena: imaju doseg na nivou biblioteke, moraju se

razlikovati od globalnih identifikatora i moraju se koristiti bez navođenja rezolucionog operatora

Isti naziv identifikatora iz anonimnog prostora imena u dvije datoteke označava dva različita entiteta

Poruka koristi anonimni prostor imena umjesto proglašavanja globalnih imena za static