Bosna i Hercegovina
Federacija Bosne i Hercegovine
SREDNJOBOSANSKI KANTON
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

KURIKULUM NASTAVNOG PREDMETA
PROGRAMIRANJE

ZA IT GIMNAZIJU

Travnik, maj 2025.

Bosna i Hercegovina
Federacija Bosne i Hercegovine
SREDNJOBOSANSKI KANTON
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

KURIKULUM NASTAVNOG PREDMETA

PROGRAMIRANJE

ZA IT GIMNAZIJU

Travnik, maj 2025.

Kurikulum nastavnog predmeta
Programiranje

Izdavač: Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

Za izdavača: Bojan Domić, ministar

Stručni tim za razvijanje, prilagođavanje i inoviranje predmetnih kurikuluma i njihovu primjenu u osnovnim i srednjim školama na području Srednjobosanskog kantona u kojima se nastavni proces realizira na bosanskom jeziku:
Nezira Fuško, voditeljica Stručnog tima
doc.dr.sc. Nešad Krnjić, voditelj radne skupine
Amra Mirojević, MA., član
Arnela Šabanović, MA., član

Recenzenti:
prof.dr.sc. Nevzudin Buzađija

Tehnička priprema i uređenje:
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

SADRŽAJ

A/ OPIS PREDMETA

Svrha predmeta Programiranje je sistematski osposobiti učenike za razumijevanje, primjenu i razvoj programskih rješenja, počevši od osnovnih principa do naprednih koncepata. Kroz proces učenja, učenici razvijaju sposobnost algoritamskog i logičkog razmišljanja, rješavanja problema, digitalne pismenosti, kreativnosti i inovativnosti. Predmet ima za cilj potaknuti kod učenika interes za istraživanje i cjeloživotno učenje, jačati vještine timskog rada i komunikacije, te integrisati znanja iz matematike, logike, informacionih tehnologija i projektno orijentisanog učenja. Na taj način, Programiranje postaje značajna karika u pripremi učenika za život u savremenom društvu i tehnološkom okruženju.

U digitalnom dobu, prisustvo računara, pametnih uređaja i tehnologija s integrisanim inteligentnim modulima stvara potrebu za razumijevanjem načina na koji nastaju i funkcionišu softverska rješenja. Programiranje nije samo tehnička vještina, nego i sredstvo razvijanja logike, preciznosti, kreativnog pristupa i sposobnosti rješavanja složenih problema. Kroz programiranje, učenici ovladavaju konceptima algoritama, strukturiranog razmišljanja i modeliranja realnih situacija u digitalnom obliku, čime se otvaraju mogućnosti za interdisciplinarne primjene u nauci, inženjerstvu, poslovanju i umjetnosti.

Izvođenje nastave programiranja zasniva se na postepenom usvajanju znanja – od jednostavnih primjera do kompleksnih aplikacija. Učenici se upoznaju s različitim programskim jezicima i paradigmama (strukturno i objektno-orijentisano programiranje), te stiču sposobnost da razumiju i primjenjuju univerzalne principe neovisno o tehnologiji. Kroz izradu algoritama, njihovo prevođenje u kod, testiranje i optimizaciju, razvijaju se:

- tehničke vještine – rad u različitim programskim okruženjima, poznavanje sintakse i semantike programskih jezika.
- analitičke vještine – sposobnost dekompozicije problema i pronalaženja optimalnih rješenja.
- kreativnost i inovativnost – osmišljavanje novih pristupa i funkcionalnosti.
- komunikacijske i timske vještine – rad u paru, manjim grupama i većim projektnim timovima.

Nastava obuhvata kombinaciju teorije i praktičnog rada. Poseban naglasak stavlja se na:

- rješavanje stvarnih problema kroz projektnu nastavu,
- povezivanje programiranja s drugim predmetima (matematika, logika, informatika, baze podataka),
- upotrebu različitih izvora podataka i tehnologija,
- razvijanje kritičkog stava prema izboru alata i tehnologija.

U višim razredima akcenat je na složenijim projektnim zadacima i primjeni naprednih koncepata, pri čemu učenici razvijaju istraživački pristup, takmičarski duh i spremnost na prilagođavanje promjenama.

Predmet Programiranje izučava se tokom sve četiri godine obrazovanja u IT gimnazi, prateći logički slijed razvoja kompetencija od osnovnog do naprednog nivoa. Učenici ne samo da stiču tehnička znanja, nego i razumiju širu sliku funkcionisanja informaciono-komunikacionih sistema, čime se olakšava prelazak na druge informatičke discipline. Povezanost s matematikom i logikom omogućava dublje razumijevanje algoritamskog pristupa, dok veza s predmetima poput informatike i baza podataka osigurava primjenu znanja u realnim scenarijima.

Završetkom ovog predmeta, učenici će:

- biti osposobljeni za samostalno pisanje, testiranje i održavanje programskog koda,
- razumjeti i primijeniti osnovne i napredne algoritamske strukture,
- povezivati teorijska znanja iz drugih oblasti s praktičnim programerskim zadacima,
- posjedovati razvijene kompetencije za cjeloživotno učenje i prilagodbu novim tehnologijama,
- biti spremni za nastavak obrazovanja u informatičkim i tehničkim disciplinama, kao i za ulazak u profesionalni IT sektor.

B/ CILJEVI UČENJA I PODUČAVANJA PREDMETA

Na osnovu obrazovnih i društvenih zahtjeva savremenog doba te algoritmaskog načina razmišljanja i razumijevanja filozofije programiranja, ciljevi učenja i podučavanja predmeta programiranje su:

1. Razviti sposobnost analitičkog razmišljanja i logičkog povezivanja u procesu rješavanja problema.
2. Razviti sposobnost izrade programskih rješenja pomoću algoritama, pseudokoda i dijagrama toka, uz primjenu osnovnih programskih struktura: sekvenca, selekcija i iteracija.
3. Podsticati kreativnost, inovativnost i prilagodljivost u korištenju različitih programskih jezika, uz primjenu poznatih tehnika razvoja softvera i različitih programskih paradigmi.
4. Razumjeti i primjenjivati osnove programiranja baza podataka.
5. Povezivati i primjenjivati stečena znanja u izradi konkretnih programskih rješenja, uz pravilnu upotrebu sintakse, struktura i tipova podataka.
6. Primjenjivati principe programiranja i projektne pismenosti, uz poseban naglasak na sigurnost web aplikacija. To podrazumijeva prepoznavanje i prevenciju osnovnih sigurnosnih prijetnji, validaciju i filtriranje korisničkog unosa, zaštitu formi i podataka, te primjenu osnovnih koncepata kontrole pristupa.

C/ OBLASNA STRUKTURA PREDMETNOG KURIKULUMA

I razred (programski jezik C++)

1. Uvod u programiranje (pojam programiranja, kompajler, linker, faze programiranja)
2. Programski jezici (historijski razvopj programskih jezika, podjela i osobine generacija programskih jezika)
3. Algoritmi (definicija i svojstva algoritma, analiza problema, etape rješavanja problema, grafički zapis algoritma, algoritamske dtrukture, složeni algoritmi)
4. Struktura programa (upoznavanje sa strukturom programskog jezika (npr. C++) i strukturom programa, ključne riječi, identifikatori, definicija konstanti i promjenljivih, komentari)
5. Tipovi podataka (osnovni tipoivi podaaka u programskom jeziku, unos i prikaz podataka (cout, cin)
6. Izrazi inaredbe (operatori jezika C++, naredbe, izrazi, prvenstvo operatora, operator prdruživanja, aritmetički operatori, relacijski operatori, logičkioperatori, operatori inkrementiranja i dekrementiranja, naredbe pridruživanja, složene naredbe)
7. Tok programa (naredbe grananja (if, if – lse), naredbe višestrukog grananja (switch), naredbe ponavljanja (for, while, do – while), naredbe break i continue, naredba bezuslovnog skoka (go to)).

II razred (programski jezik C++)

1. Nizovi (jednodimenzionalni i višedimenzionalni) (jednodimenzionalni nizovi, pristupanje elementima niza, pretraživanje i sortiranje niza, višedimenzionalni nizovi, pristupanje elementima matrice, ..)
2. Vektori (pretraživanje vektora, rotiranje vektora, invertovanje vektora, sortiranje vektora, sažimanje vektora, proširivanje vektora)
3. Funkcije (definisanje funkcije, parametri i argumenti funkcije, formalni parametri, lokalne i globalne promjenljive, rekurzivne funkcije)
4. Stringovi (definicija stringa, inicijalizacija stringa, pristup elementima stringa pomoću indeksa, upoznavanje sa osnovnim funkcijama za rad sa stringovima iz biblioteke)
5. Dinamičke strukture podataka (dinamičke varijable, pokazivači, jednostruko povezane liste, kružne liste, grafovi, struktura drveta, binarno stablo)
6. Datoteke (Otvaranje i zatvaranje datoteke, rad sa datotekama)

III razred (programski jezik C#)

1. Osnovni koncepti objektno orjentisanog jezika (klasa, objekt, enkapsulacija, konstruktori, destruktori, nasljeđivanje, polimorfizam)
2. Objektni jezik i C jezik, šta je isto a šta različito (logički podaci, definisanje nabrojivog i strukturnog tipa, definisanje podataka na nivou bloka, preklapanje imena funkcije, reference, operator this, dinamički objekti)

3. Klase (Pojam klase, Enkapsulacija podataka, Razlika između klase i strukture, Podrazumijevani konstruktor, Konstruktor sa parametrima, Konstruktor kopije, Metode, Set metode, Get metode, Vaza između klasa)
4. Rukovanje izuzecima (rukovanje izuzecima, prijavljivanje izuzetaka, prihvatanje izuzetaka)
5. Bibliloteka komponenti (izrada projekta, forma, svojstva, metode, događaji, labela, dugme, događaji miša, zajednička svojstva za sve komponente, okvir za tekst (edit), panel, kastovanje komponente nad kojom se desio događaj, okvir za grupu, okvir za potvrdu, grupa radio dugmadi, događaj tastature, komponenta listbox, kombinovani okvir za tekst sa listom combobox, komponenta timer, dinamičko kreiranje komponenti, niz pokazivača na komponente, matrica pokazivača na komponente)

IV razred (tehnologije za strukturiranje i oblikovanje web sadržaja: HTML i CSS; programski jezici: JavaScript i PHP; sistem za upravljanje relacionim bazama podataka: MySQL)

1. Uvod u web tehnologije i osnove HTML-a i CSS-a (arhitektura web aplikacija (klijent–server), uloga HTML-a i CSS-a, struktura HTML dokumenta, elementi i atributi (naslovi, linkovi, liste, tabele, forme), CSS selektori, klase, ID, osnovno pozicioniranje)
2. Uvod u JavaScript i osnove klijentske logike (promjenljive, funkcije, događaji, rad sa DOM-om, validacija formi, povezivanje sa HTML-om, jednostavne interakcije korisnika sa stranicom)
3. Instalacija i konfiguracija PHP i razvojne okoline (instalacija lokalnog servera (XAMPP ili sl.), upoznavanje sa editorom (VS Code / Notepad++ / Visual Studio), prvi PHP program, ugradnja PHP-a u HTML)
4. Osnove PHP-a (sintaksa, promjenljive, izrazi, Uslovni izrazi i petlje, Funkcije i uključivanje fajlova, GET i POST metode, Rukovanje formama sa PHP-om)
5. Rad sa bazom podataka (MySQL) (Kreiranje baze i tabela, Povezivanje PHP-a sa MySQL-om, Unos, čitanje, izmjena i brisanje podataka (CRUD), Osnovne SQL naredbe (SELECT, INSERT, UPDATE, DELETE))
6. Praktičan razvoj dinamičke web stranice (Planiranje izgleda i strukture stranice, Korištenje HTML/CSS/JS + PHP/MySQL u jednom projektu, Validacija podataka, prikaz informacija, jednostavna sesija (login), Modularna struktura koda (header, footer, include))
7. Uvod u sigurnost web aplikacija (Osnovne sigurnosne prijetnje (XSS, SQL injection), Validacija i filtriranje korisničkog unosa, Zaštita formi i podataka, Pristup kontroli (login – osnovni koncept))
8. Izrada završnog mini projekta i prezentacija (Grupni ili individualni rad, Dokumentacija, Prezentacija, Evaluacija i povratna informacija)

Nastava predmeta Programiranje temelji se na razvijanju znanja i vještina koje omogućavaju učenicima da osmisle, kreiraju i analiziraju funkcionalne programske proizvode. Sadržaj predmeta strukturiran je kroz tri međusobno povezane oblasti:

A. Algoritmi
B. Reprezentacija i obrada podataka
C. Kontrola toka i komunikacija.

Svaka od ovih oblasti predstavlja ključni aspekt programiranja, a zajedno obuhvataju cjelovit proces rješavanja problema putem programskog jezika. Iako su tematski diferencirane, ove oblasti se u nastavi ne tretiraju strogo odvojeno. One se prirodno prožimaju i nadopunjuju unutar različitih tematskih cjelina, u skladu s logikom izgradnje programskog rješenja. Bez obzira na programski jezik koji se koristi, učenici se podstiču da prepoznaju i primjenjuju ove koncepte kao integrisane cjeline.

A. Algoritmi

Oblast Algoritmi usmjerena je na razvoj sposobnosti učenika da probleme iz stvarnog svijeta sagledaju kroz prizmu računske logike, te da ih rješavaju konstruisanjem algoritama. To podrazumijeva identifikaciju problema, formulaciju strategije rješavanja i predstavljanje rješenja u obliku sekvence precizno definisanih koraka koje može interpretirati računar. Kroz ovu oblast učenici također analiziraju već postojeće algoritme, upoznaju se s njihovom strukturom i načinom rada, te razvijaju sposobnost poređenja njihove efikasnosti. Time se kod učenika jača algoritamski način razmišljanja i podstiče razumijevanje optimizacije u programiranju.

B. Reprezentacija i obrada podataka

U savremenom digitalnom okruženju, podaci predstavljaju osnovu svakog informacionog sistema. Oblast Reprezentacija i obrada podataka bavi se načinima kako podatke adekvatno predstaviti, pohraniti i efikasno obrađivati u digitalnom obliku. To uključuje rad s različitim tipovima podataka, strukturama podataka, kao i pristupima pohrani i pristupu informacijama. Učenici se kroz ovu oblast upoznaju s različitim modelima predstavljanja podataka, te izučavaju osnovne i napredne metode obrade, uključujući moderne pristupe poput objektno zasnovanog i objektno orijentisanog programiranja. Fokus je na razumijevanju kako struktura i organizacija podataka utiču na brzinu, pouzdanost i funkcionalnost programskog rješenja.

C. Kontrola toka i komunikacija

Oblast Kontrola toka i komunikacija obuhvata logički slijed izvršavanja naredbi unutar programa, kao i interakciju programa s korisnikom, drugim softverima i hardverom. Ključni aspekti ove oblasti odnose se na upravljanje tokom izvršavanja – uključujući grananje, petlje, funkcije i rukovanje izuzecima – što omogućava dinamičko i fleksibilno ponašanje programskog sistema. Pored kontrole toka, oblast uključuje i sve oblike komunikacije programa s okolinom. To podrazumijeva oblikovanje korisničkog interfejsa, upravljanje ulaznim i izlaznim operacijama, kao i interakciju s hardverskim komponentama i memorijskim resursima. Učenici uče kako se programi pisani na jezicima visokog nivoa prevode u mašinski kod putem kompajlera i interpretera, te kako se ti programi pohranjuju i izvršavaju. Poseban

naglasak stavlja se na sigurnost komunikacije kod web aplikacija. To obuhvata zaštitu od uobičajenih prijetnji poput XSS napada i SQL injection-a, provjeru i filtriranje korisničkog unosa, osiguranje formi i pohranjenih podataka, te primjenu osnovnih metoda autentifikacije i kontrole pristupa. Na ovaj način učenici usvajaju principe sigurnog programiranja i razumiju važnost očuvanja integriteta sistema i povjerljivosti podataka. Ova oblast ovog predmetnog kurikuluma, osim tehničkog aspekta, podstiče i razvoj preciznosti, sistematičnosti, te kritičkog i analitičkog mišljenja. Doprinosi oblikovanju samostalnih, pouzdanih i komunikativnih budućih programera, osposobljenih za jasno izražavanje i efikasno rješavanje problema u digitalnom okruženju.

Oblasna struktura predmetnog kurikuluma Programiranja

U nastavku slijedi dio koji se odnosi na odgojno-obrazovne ishode koji su okosnica predmetnog kurikuluma i razrađeni su za svaku od tri oblasti (domene) na kojima se temelji. Odgojno-obrazovni ishodi pomažu nastavnicima u praćenju napretka učenika i u vrednovanju učeničkih postignuća. Tokom pripremanja procesa učenja i podučavanja nastavnik treba povezati odgojno-obrazovne ishode sa sadržajima navedenim u kurikulumu i metodama podučavanja. U tabelama su odgojno-obrazovni ishodi označeni šiframa. Skraćenice poput A.I.1. ili B.IV.3. označavaju redom: oblast kojoj ishod pripada (A. Algoritmi, B. Reprezentacija i obrada podataka i C. Kontrola toka i komunikacija), godinu podučavanja predmeta (I.-prvi razred, II.-razred, III. – treći razred i IV. – četvrti razred u IT gimnaziji), te redni broj odgojno-obrazovnog ishoda koji se podučava u sklopu navedene oblasti (1. – prvi ishod, 2. – drugi ishod, ...). Skraćenice TIT 3.1.1. ili npr. TIT 4.1.2. označavaju poveznice sa Zajedničkom jezgrom nastavnih planova i programa definisanih na ishodima učenja.

D/ ODGOJNO-OBRAZOVNI ISHODI

IT GIMNAZIJA

1. **razred IT gimnazije** /2 nastavna časa sedmično/70 nastavnih časova godišnje/

Oblast: A/Algoritmi	
Ishod učenja	**Razrada ishoda**
A.I.1. Analizira problem i razvija algoritamsko rješenje primjenom osnovnih algoritamskih struktura.	● Prepoznaje ključne elemente jednostavnog problema pogodnog za algoritamsko rješavanje. ● Razlaže problem na manje, logički povezane cjeline. ● Primjenjuje algoritamske strukture slijeda, grananja i ponavljanja pri izradi rješenja. ● Oblikuje jednostavan algoritam u tekstualnom ili grafičkom obliku (npr. pseudokod, dijagram toka). ● Objašnjava logiku i svrhu svake komponente izrađenog algoritma. ● Provjerava tačnost i efikasnost izrađenog algoritma kroz primjere.
Poveznice sa ZJNPP	**TIT-4.1.1 TIT-4.1.2**
Ključni sadržaji	
Definicija i svojstva algoritama. Analiza problema i algoritamski pristup rješavanju. Etape algoritamskog rješavanja problema. Osnovne algoritamske strukture: slijed, grananje, ponavljanje. Pseudokod i dijagram toka kao oblici zapisa algoritma. Testiranje i verifikacija jednostavnog algoritma.	
Preporuke za ostvarenje ishoda	
Kontekstualizovati zadatke: koristiti svakodnevne primjere iz života učenika (npr. algoritam za kuhanje čaja, odlazak u školu). Kombinovati metode prikaza tako da se algoritam podučava i grafički, pomoću dijagrama toka, i tekstualno, kroz pseudokod. Nastava treba biti problemski orijentisana, pri čemu učenici rade u parovima ili grupama na analizi i razradi problema. Posebno je važno podsticati refleksiju i diskusiju, kako bi učenici verbalizovali svoja rješenja i razmjenjivali različite pristupe u razredu. U nastavi se preporučuje korištenje softverskih alata poput Flowgorithm ili scratch.mit.edu za vizualizaciju algoritama, dok se naprednijim učenicima mogu zadavati izazovi kroz poređenje dvaju algoritama za isti problem, bilo po efikasnosti, bilo po jasnoći. Korelacije sa drugim predmetima mogu značajno obogatiti nastavu. U matematici se algoritamski pristupi mogu povezati sa zadacima poput provjere parnosti broja pomoću grananja (if/else), izračunavanja aritmetičke sredine n brojeva uz pomoć petlje (for), pronalaženja minimalne ili maksimalne vrijednosti sekvencijalnom pretragom ili primjene Euklidovog algoritma za NZD pomoću while petlje. Primjer zadatka može biti: „Dati prirodni broj n; napiši pseudokod koji provjerava da li je n prost.“ čime se objedinjuje upotreba grananja i ponavljanja. U engleskom jeziku učenici mogu razvijati dvojezični rječnik pojmova kao što su input, output, condition, loop, flowchart i pseudocode. Aktivnosti mogu uključivati pisanje pseudokoda na engleskom jeziku, npr. za problem „Ticket price with age discount“, gdje jedan učenik zatim usmeno objašnjava logiku algoritma u trajanju od 60 do 90 sekundi, ili definisanje ulaza i izlaza za proces „unos ocjena → prosjek → verbalna ocjena“. U fizici i tehničkom odgoju algoritmi se mogu primijeniti kroz zadatke konverzije mjernih jedinica uz korištenje grananja, poput pretvaranja °C u °F i obratno, ili kroz izbor odgovarajuće formule u zavisnosti od dostupnih veličina.	
A.I.2. Koristi grafičke i tekstualne metode za predstavljanje algoritama.	● Prepoznaje i objašnjava osnovne simbole i pravila korištenja u dijagramima toka.

	● Povezuje elemente dijagrama toka s odgovarajućim algoritamskim strukturama. ● Zapisuje pseudokod koristeći osnovna pravila i strukture algoritma. ● Preoblikuje/prevodi algoritam iz jednog oblika zapisa u drugi (npr. iz dijagrama toka u pseudokod i obratno). ● Procjenjuje i revidira jasnoću i čitljivost algoritma u različitim oblicima zapisa. ● Testira, upoređuje i potvrđuje tačnost i procjenjuje efikasnost izrađenog algoritma kroz primjere
Poveznice sa ZJNPP	**TIT-4.1.1 TIT-4.1.2**

Ključni sadržaji

Oblici zapisa algoritama: dijagram toka, pseudokod.
Elementi i simboli dijagrama toka (početak/kraj, obrada, uslov, tok).
Pravila za pisanje pseudokoda.
Korelacija između algoritamskih struktura i njihovog prikaza.
Pretvaranje algoritma iz dijagrama u pseudokod i obratno.

Preporuke za ostvarenje ishoda

Nastavnik najprije uvodi standardne konvencije pseudokoda (BEGIN, END, IF, WHILE…) i osnovne simbole dijagrama toka kroz zajedničku analizu nekoliko dobro i slabije napisanih primjera, kako bi učenici uočili šta znače jasnoća, konzistentan stil i standardizacija zapisa. Slijedi kratka vježba konverzije iz opisnog teksta u dijagram toka i pseudokod.
Prikazati dobre i loše primjere zapisa algoritama radi razvijanja osjećaja za jasnoću i standardizaciju.
Koristiti radne listove sa zadacima konverzije algoritma iz teksta u dijagram i pseudokod.
Rad u parovima: jedan učenik crta dijagram toka, drugi piše pseudokod za isti algoritam, zatim zamjenjuju uloge.
Digitalni alati: koristiti alate poput Draw.io, Lucidchart ili Flowgorithm za izradu dijagrama toka.
Povezivati s programskim jezikom C++: istaknuti kako se dijelovi pseudokoda mapiraju u konkretan kod.

A.I.3. Razlikuje osnovne etape algoritamskog rješavanja problema.	● Identifikuje osnovne faze procesa algoritamskog rješavanja problema. ● Objašnjava svrhu svake etape (analiza problema, planiranje rješenja, dizajn algoritma, testiranje, evaluacija). ● Opisuje redoslijed koraka od definicije problema do gotovog algoritma. ● Ilustrira svaku fazu na konkretnim problemskim situacijama. ● Analizira posljedice preskakanja ili površne obrade pojedinih faza.
Poveznice sa ZJNPP	**TIT-4.1.1 TIT-4.1.2**

Ključni sadržaji

Faze algoritamskog pristupa rješavanju problema:
- Analiza problema,
- Planiranje rješenja,
- Dizajniranje algoritma,
- Testiranje i evaluacija algoritma.

Uloga svake etape u procesu programiranja.
Primjeri za svaku fazu na jednostavnim problemima.

Preporuke za ostvarenje ishoda

Nastavu započeti kratkim, realnim problemom (npr. provjera da li je broj paran i ispis odgovora) i voditi razred kroz sve etape kao kroz „mini-projekat“. U fazi analize učenici prvo jasno zapisuju šta je ulaz, šta je izlaz i koje su pretpostavke i ograničenja; zatim u planiranju razlažu problem na manje korake i biraju strategiju rješavanja; u dizajnu izrađuju skicu dijagrama toka i pseudokod; u testiranju i evaluaciji izrađuju mali set test-slučajeva, porede očekivano i dobijeno te bilježe prijedloge za doradu. Nastavnik vizualizira cijeli proces kao

kružni tok i namjerno vraća razred korak unazad kada se otkrije nedosljednost, kako bi se uočio iterativni karakter rada. Vizualni prikazi: napraviti dijagram koji prikazuje etape kao međusobno povezane korake u ciklusu razvoja algoritma. Rad u grupama: podijeliti učenike u grupe gdje svaka grupa fokusira jednu fazu i zatim zajednički sklapaju kompletno rješenje. Navesti primjere iz svakodnevnog života. Refleksivne vježbe: zadati učenicima da napišu kratku analizu grešaka koje nastaju ako se neka faza preskoči.

A.I.4. Prepoznaje i primjenjuje poznata algoritamska rješenja u različitim problemskim situacijama. *(Napomena: ovaj ishod je sadržajno proširen u odnosu na prethodne tri tačke i temelji se na stavci "složeni algoritmi", koja implicira rad s poznatim algoritamskim obrascima.)*	● Prepoznaje zadatke koji se mogu riješiti primjenom poznatih algoritama (npr. zamjena vrijednosti, brojanje, pretraga, sortiranje). ● Opisuje način rada poznatih algoritamskih rješenja. ● Implementira algoritam koristeći odgovarajuće algoritamske strukture. ● Prilagođava poznata rješenja konkretnim situacijama kroz modifikaciju ulaznih i izlaznih podataka. ● Analizira prednosti i ograničenja primijenjenog algoritma u odnosu na druge moguće pristupe.
Poveznice sa ZJNPP	**TIT-4.1.2 TIT-4.1.3 TIT-4.2.2**

Ključni sadržaji

Primjeri poznatih algoritama za:
- Razmjenu vrijednosti dvije promjenljive
- Sabiranje i brojanje elemenata
- Pretraživanje niza (linearno)
- Jednostavno sortiranje (npr. selection sort, bubble sort).

Strukture koje podržavaju ove algoritme (petlje, uslovi, pomoćne promjenljive).
Analiza i poređenje efikasnosti jednostavnih algoritama.

Preporuke za ostvarenje ishoda

Zadaci sa šablonom: ponuditi učenicima zadatke koji se mogu riješiti poznatim algoritamskim obrascima.
Upoređivanje algoritama: dati dva algoritma za isti zadatak (npr. sortiranje) i diskutovati koji je efikasniji i zašto.
Zadaci u parovima: jedan učenik daje tekstualni opis problema, drugi bira i implementira poznati algoritam kao rješenje.
Upotreba pseudokoda i dijagrama: predstaviti algoritam u više oblika prije implementacije u kodu.
Refleksivne aktivnosti: učenici objašnjavaju zašto su izabrali određeni algoritam i kako su ga prilagodili zadatku.
Diferencirana nastava: naprednijim učenicima ponuditi složenije algoritme (npr. binarna pretraga) kao izazov.
Formativno praćenje se može realizovati kroz kratke test-slučajeve i vršnjačku provjeru, dok se za sumativnu provjeru zadaje mali zadatak u kojem učenici samostalno prepoznaju obrazac, skiciraju rješenje i implementiraju ga uz kratko obrazloženje izbora.

Oblast: B/Reprezentacija i obrada podataka

Ishod učenja	Razrada ishoda
B.I.1. Prepoznaje i primjenjuje osnovne tipove podataka i pravila za definisanje konstanti i promjenljivih.	● Objašnjava razliku između konstanti i promjenljivih. ● Piše deklaraciju konstante i promjenljive u skladu sa sintaksom programskog jezika C++. ● Navodi osnovne tipove podataka (npr. int, float, char, bool). ● Povezuje tip podatka sa vrstom informacije koju predstavlja. ● Primjenjuje odgovarajuće tipove podataka u jednostavnim programskim zadacima.

Poveznice sa ZJNPP	**TIT-4.2.1**
Ključni sadržaji	
Osnovni pojmovi: konstanta, promjenljiva. Pravila imenovanja i deklarisanja promjenljivih i konstanti. Tipovi podataka: cijeli brojevi (int), realni brojevi (float/double), karakteri (char), logičke vrijednosti (bool). Asignacija i osnovne operacije s promjenljivima.	
Preporuke za ostvarenje ishoda	
Prikazati konkretne primjere: npr. const float PI = 3.14; i int broj = 5; uz objašnjenje svake komponente. Kreirati zadatke za analizu: učenici dobijaju isječke koda i određuju tipove podataka i njihovu namjenu. Rad u paru: jedan učenik zadaje vrstu informacije, drugi bira odgovarajući tip podatka i piše definiciju. Povezati sa svakodnevnim kontekstima: npr. promjenljiva za broj godina, ocjena, temperatura itd. Navesti korelacije s matematikom (razlika između cijelih i realnih brojeva, zaokruživanje) i fizikom (temperatura, mjerenja) Korištenje tablica i dijagrama: za poređenje karakteristika tipova podataka (veličina, domet, format). Greške kao prilika za učenje: analizirati tipične greške kod definisanja promjenljivih (npr. pogrešno ime, neodgovarajući tip).	
B.I.2. Oblikuje izraze i koristi operatore za obradu podataka u skladu sa sintaksom programskog jezika.	• Navodi osnovne operatore (aritmetičke, relacijske, logičke). • Kombinuje operatore u jednostavne izraze u skladu s pravilima sintakse. • Primjenjuje operatore pridruživanja, inkrementacije i dekrementacije. • Primjenjuje pravila prvenstva operatora i koristi zagrade za kontrolu redoslijeda izvršavanja. • Rješava jednostavne problemske zadatke korištenjem izraza i operatora.
Poveznice sa ZJNPP	**TIT-4.2.1**
Ključni sadržaji	
Operator pridruživanja (=). Aritmetički operatori (+, -, *, /, %). Relacijski operatori (==, !=, <, >, <=, >=). Logički operatori (&&, ||, !). Operator inkrementiranja i dekrementiranja (++ i --). Prvenstvo operatora i korištenje zagrada. Primjeri izraza u jeziku C++ i njihovo izvođenje.	
Preporuke za ostvarenje ishoda	
Radionice i mini testovi: kratki zadaci za rješavanje izraza uz obrazloženje redoslijeda izvršavanja. Koristiti metodu pogrešaka (davanje gotovog koda sa namjernim greškama u kodu kako bi učenici otkrivali greške i preponavali tipove grešaka). Vizualizacija operatora: tabela prvenstva operatora dostupna na plakatu ili u digitalnom obliku. Korištenje isječaka koda: učenici analiziraju izraze i određuju rezultat izvođenja. Metodički je korisno „paralelno rješavanje“: najprije računanje na papiru uz očekivani rezultat, potom izvođenje u okruženju i upoređivanje. Usput se skreće pažnja na učestale greške: zamjena = i ==, cjelobrojno dijeljenje umjesto realnog, Zadaci za kreativno izražavanje: npr. zadati konkretan problem i tražiti od učenika da sami kreiraju izraz koji ga rješava. Korelacija s matematikom prirodno se ostvaruje kroz poredak operacija i kroz logičke iskaze, dok se u bosanskom jeziku njeguje precizno formuliranje uslova.	
B.I.3. Demonstrira unos i prikaz podataka korištenjem osnovnih naredbi ulaza/izlaza.	• Objašnjava svrhu naredbi za unos (cin) i prikaz (cout) podataka u jeziku C++. • Povezuje ulazne/izlazne naredbe sa odgovarajućim tipovima podataka. • Piše jednostavne naredbe za unos vrijednosti u promjenljive.

	• Demonstrira tekstualni i numerički izlaz na ekranu koristeći cout. • Integriše unos i izlaz u okviru jednostavnih zadataka i programa. • Primjenjuje escape karaktere i manipulatore izlaza za formatiranje ispisa (\n, \t, setw, endl).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Naredbe cin i cout u programskom jeziku C++. Osnovni operatori ulaza (>>) i izlaza (<<). Korištenje tekstualnih poruka uz unos/izlaz. Escape karakteri (\n, \t). Manipulatori izlaza (endl, setw). Kombinovanje više naredbi ulaza i izlaza. Jednostavni primjeri interakcije sa korisnikom.	
Preporuke za ostvarenje ishoda	
Zadaci sa simulacijom dijaloga: npr. unos imena, godina i ispis poruke dobrodošlice. Rad na konkretnim primjerima, usklađenim s do sada obrađenim tipovima podataka i operatorima: unos kontakt-kartice (ime, prezime, telefon) i ispis jedne formatirane linije; unos naslova knjige i autora te ispis poruke „Autor — Naslov“; unos naziva filma i trajanja (u minutama) i ispis „Film: … — trajanje … min“; unos grada i kratke poruke, uz višeredni ispis koristeći \n i poravnanje rubrika pomoću setw. Korištenje vizualnih vodiča: dijagrami koji prikazuju tok podataka od tastature ka memoriji i od memorije ka ekranu. Formatiranje izlaza: pokazati različite efekte manipulatora izlaza na oblik prikaza. Samoprovjera: učenici analiziraju isječke koda i predviđaju što će biti prikazano na ekranu. Rad u paru: jedan učenik piše kod, drugi testira unos i ispis u IDE-u.	

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.I.1. Strukturira jednostavne programske zadatke primjenom naredbi za grananje i ponavljanje.	• Prepoznaje situacije u kojima se koristi grananje ili ponavljanje. • Primjenjuje naredbe if, if–else, switch za kontrolu toka na osnovu uslova. • Razlikuje vrste petlji (for, while, do–while). • Primjenjuje petlje (for, while, do–while) u odgovarajućem kontekstu. • Kontroliše brojače i uslove ulaska/izlaska iz petlji. (umjesto “upravlja”) • Kombinuje strukture grananja i petlji u jednostavnom zadatku. • Testira funkcionalnost programa i otkriva logičke greške.
Poveznice sa ZJNPP	**TIT-3.2.2 TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Naredbe grananja: if, if–else, switch. Naredbe ponavljanja: for, while, do–while. Ključne riječi: break, continue, goto. Kontrola uslova i promjenljivih unutar petlje. Kombinacija grananja i petlji u rješavanju problema. Primjeri iz svakodnevnog života: kalkulator, brojanje, evaluacija unosa.	
Preporuke za ostvarenje ishoda	
Zadaci iz realnog konteksta (metoda „crne kutije“) • Prvo se navode ulazi, izlazi, preuslovi/postuslovi i očekivani rezultati, bez uvida u implementaciju; učenici dizajniraju testove (tipični, granični, nevažeći), a tek potom pišu kod. • Primjeri bez matematike:	

- *Validacija korisničkog imena:* ulaz string (3–12 znakova, bez razmaka) → izlaz "ispravno"/"neispravno". Testovi: "Ana", "a", "ime sa razmakom", "" .
- *Meni sa* switch: ulaz broj 1–4 → odgovarajuća poruka; testovi: 0, 1, 4, 5, 'x'.
- *Unos do sentinel vrijednosti:* unos niza riječi do "KRAJ" → izlaz broj unosa + spisak; testovi: prazno, jedan unos, "kraj" vs "KRAJ".
- *Login s ograničenjem pokušaja:* max 3 pokušaja → "prijava uspješna"/"zaključano"; testovi: tačno u 1., tačno u 3., 3× pogrešno.
- *Filtriranje poruke:* ulaz tekst → izlaz "poruka prihvaćena/odbijena" prema pravilima (npr. zabranjene riječi).

Poređenje različitih petlji: ista funkcionalnost implementirana kroz for, while i do–while.
Vizualizacija toka: crtanje dijagrama toka za logiku grananja i petlji prije implementacije.
Kod sa greškom: dati učenicima zadatke koji sadrže logičke greške u toku kontrole i tražiti analizu i ispravku.
Razrada algoritma prije koda: insistirati na pisanju pseudokoda ili kratke verbalne analize prije programiranja.
Zadaci sa rastućim nivoom složenosti: od jednostavnih do onih gdje se grananja ugniježđuju unutar petlji i obratno.

C.I.2. Primjenjuje kontrolne naredbe i upravlja tokom izvršavanja programa.	• Objašnjava funkciju kontrolnih naredbi break, continue i goto u okviru petlji i grananja. • Prepoznaje situacije u kojima se koristi prekid i preskakanje dijela programskog toka. • Primjenjuje kontrolne naredbe unutar struktura ponavljanja i grananja. • Analizira posljedice upotrebe kontrolnih naredbi na tok izvršavanja programa. • Ispravlja greške uzrokovane nepravilnim korištenjem kontrolnih naredbi.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Kontrolne naredbe:
- break – prekid izvršavanja petlje ili switch strukture
- continue – preskakanje trenutne iteracije
- goto – bezuslovni skok (uz napomenu o preporučenoj umjerenoj upotrebi).

Lokacija i efekat kontrolnih naredbi unutar različitih struktura.
Uloga i ograničenja kontrolnih naredbi u strukturi programa.
Primjeri iz prakse gdje kontrolne naredbe olakšavaju implementaciju.

Preporuke za ostvarenje ishoda

Ishod je najbolje graditi od „čistog" toka ka ciljanim prekidima. Nastavnik prvo pokaže baznu varijantu petlje ili switch-a bez posebnih kontrolnih naredbi, a zatim uvede problem koji prirodno traži skraćivanje ili preskakanje toka: npr. rana obustava pretrage kada je element pronađen (break) ili preskakanje obrade neispravnog unosa i nastavak s narednim elementom (continue). Učenici na malim skupovima podataka prate kako se tok mijenja kada se doda break ili continue, uz skicu dijagrama toka na kojem se jasno vidi mjesto prekida ili preskoka.
Zadaci sa uočenim greškama: učenici analiziraju kod gdje je break ili continue pogrešno primijenjen.
Grafički prikaz toka: crtati dijagrame toka programa i uočiti gdje dolazi do prekida ili preskoka.
Vođena diskusija: da li i kada koristiti goto ?
Povezivanje s algoritamskom logikom: isticanje da je ispravno upravljanje tokom ključno za efikasnost i razumljivost programa.
Kratko formativno provjeravanje: učenik preuzme jednostavan isječak, objasni usmeno šta će se dogoditi pri izvršavanju, predloži promjenu (break ili continue) i obrazloži šta se dobilo;
Za sumativnu provjeru zadati kratki zadatak u kojem učenik najprije uoči i ispravi greške nastale pogrešnom upotrebom break, continue ili switch, a zatim primijeni odgovarajuću kontrolnu naredbu i u jednoj–dvije rečenice obrazloži svoj izbor.

C.I.3. Objašnjava ulogu kompajlera, linkera i osnovnih faza u procesu razvoja programskog rješenja.	• Navodi osnovne faze razvoja programskog rješenja (analiza, pisanje koda, kompajliranje, povezivanje, testiranje, ispravljanje grešaka). • Objašnjava funkciju kompajlera u prevođenju izvornog koda u mašinski kod. • Opisuje ulogu linkera u povezivanju više modula u izvršnu cjelinu. • Povezuje greške nastale pri kompajliranju i povezivanju s odgovarajućim fazama razvoja. • Razlikuje sintaksne, semantičke i logičke greške u programiranju.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Faze razvoja programskog rješenja: • Analiza problema • Pisanje izvornog koda • Kompajliranje • Linkanje • Testiranje i otklanjanje grešaka. Kompajler: funkcija i uloga u prevođenju koda. Linker: povezivanje biblioteka i modula. Tipovi grešaka u programiranju: sintaksne, semantičke i logičke.	
Preporuke za ostvarenje ishoda	
Prikaz procesa kroz dijagram: vizualizacija toka od pisanja koda do izvršnog programa. Analiza grešaka iz prakse: pokazati greške koje nastaju tokom kompajliranja i linkanja, te ih zajedno analizirati. U IDE-u (npr. Code::Blocks) nastavnik demonstrira isti zadatak u dva koraka: prvo pojedinačni fajl koji se bez problema kompajlira i izvršava, zatim varijantu s dvije datoteke (main.cpp i utils.cpp) u kojoj main poziva funkciju iz drugog modula. Time se jasno vidi uloga kompajlera (prevodi svaki izvorni fajl u odgovarajući .o/.obj) i uloga linkera (spaja sve prevedene dijelove i biblioteke u jedan program). Učenici prate poruke alata i povezuju ih s fazom: sintaksne greške (npr. „expected ';'“) nastaju pri kompajliranju i odnose se na kršenje pravila jezika, semantičke greške (npr. „no matching function for call…“, „cannot convert 'double' to 'int'“), dok linkerske greške (npr. „undefined reference to add“) nastaju pri povezivanju jer definicija pozvane funkcije nije pronađena. Logičke greške se otkrivaju tek kroz test primjere i tragove ispisa. Praktična demonstracija u IDE-u: učenici pišu jednostavan program i prate poruke kompajlera i linkera. Razlikovanje grešaka: vježbe gdje učenici klasificiraju greške (npr. zarez umjesto tačka-zarez, neinicijalizovana promjenljiva). Diskusija o važnosti uredne strukture koda: isticanje veze između jasnoće koda i lakšeg otklanjanja grešaka. Povezivanje sa realnim procesom razvoja softvera: osvijestiti da kompajler i linker nisu samo alati, već važni dijelovi programerskog alata i tokova rada. Prirodna je korelacija s Digitalnim sistemima/Arhitekturom računara — ideja da kompjuter izvršava mašinski kod.	

2. razred IT gimnazije /2 nastavna časa sedmično/70 nastavnih časova godišnje/

Oblast: A/Algoritmi	
Ishod učenja	**Razrada ishoda**
A.II.1. Primjenjuje algoritme za pretraživanje i sortiranje podataka u nizu i vektoru.	● Opisuje osnovne metode pretraživanja (linearna i binarna pretraga). ● Objašnjava principe rada algoritama sortiranja (npr. selection sort, bubble sort). ● Primjenjuje algoritme pretraživanja i sortiranja u programskom jeziku C++ koristeći nizove i vektore.

	● Analizira ulazne i izlazne podatke u funkciji ispravnosti algoritma. ● Upoređuje efikasnost različitih algoritama na jednostavnim primjerima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Jednodimenzionalni nizovi i vektori. Linearna i binarna pretraga. Algoritmi sortiranja: selection sort, bubble sort (osnovni koncepti). Pristup elementima niza i vektora pomoću indeksa. Poređenje složenosti algoritama (osnovni nivo).	
Preporuke za ostvarenje ishoda	
Preporuka je učenike voditi od vidljivog, malog primjera ka generalnom pravilu. Početi s nizom od pet–šest brojeva i prikazati linearnu pretragu indeks po indeks i bilježiti gdje je poređenje uspjelo ili nije. Zatim preći na sortiranje: prvo selection sort, pa bubble sort, uz kratku tablicu trasiranja u kojoj se uočavaju zamjene i napredovanje najmanjeg/najvećeg elementa. Vizualizacija algoritama: ručno crtanje koraka sortiranja i pretrage na tabli ili korištenjem digitalnih animacija. Razvijanje algoritma iz svakodnevnih konteksta: npr. pretraga imena u imeniku, sortiranje ocjena. Eksperimentisanje s različitim ulazima: kako bi učenici uočili razliku u ponašanju i efikasnosti algoritama. Uvod u pojam složenosti algoritma kroz brojanje koraka, bez formalne notacije.	
A.II.2. Analizira i primjenjuje rekurzivne algoritme u rješavanju problemskih zadataka.	● Objašnjava pojam rekurzije i razlikuje ga od iteracije. ● Identifikuje osnovne komponente rekurzivne funkcije: bazni slučaj i rekurzivni poziv. ● Analizira tok izvršavanja rekurzivne funkcije koristeći jednostavne primjere (npr. faktorijel, Fibonaccijev niz). ● Implementira jednostavne rekurzivne algoritme u programskom jeziku. ● Prepoznaje prednosti i ograničenja rekurzije u odnosu na iteraciju.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Definicija rekurzije. Struktura rekurzivne funkcije: bazni slučaj, rekurzivni slučaj. Primjeri: faktorijel, Fibonaccijev niz, stepenovanje broja. Poređenje rekurzije i iteracije (osnovni principi). Pregled načina poziva funkcija (stack pozivi, koncept pozadinskog toka).	
Preporuke za ostvarenje ishoda	
Korištenje dijagrama toka ili stabla poziva: za vizualno predstavljanje toka rekurzivnih poziva. Rad „korak po korak“: analizirati kako se funkcija poziva i vraća kroz svaki nivo rekurzije. Za produbljivanje razumijevanja isti zadatak rješava se i rekurzivno i iterativno (faktorijel, stepenovanje), uz kratku raspravu o čitljivosti, broju koraka i potrošnji memorije. Učenici prvo usmeno opisuju bazni i rekurzivni slučaj „jezikom problema“, pa ga prevode u kod; zatim u paru pregledaju rješenja i traže tipične greške: izostavljen ili pogrešan return, rekurzivni poziv koji ne smanjuje problem, nepotpuno pokriveni rubni slučajevi. Upozorenje na greške: istaknuti značaj baznog slučaja kako bi se izbjegla beskonačna rekurzija. Korištenje debuggera u IDE-u: učenici prate tok izvršavanja rekurzivne funkcije u stvarnom vremenu. Prirodna je korelacija s matematikom (rekurentne definicije, ideja matematičke indukcije).	
A.II.3. Razlikuje korake u razvoju algoritma za rad s dinamičkim strukturama podataka.	● Objašnjava osnovni koncept dinamičkog zauzimanja memorije pomoću pokazivača. ● Identifikuje strukturu jednostruko povezane liste, kružne liste, binarnog stabla i grafova. ● Opisuje osnovne operacije nad dinamičkim strukturama: kreiranje, dodavanje, brisanje, pretraga.

	● Navodi korake potrebne za oblikovanje algoritma koji koristi dinamičke strukture. ● Uspoređuje dinamičke i statičke strukture prema prednostima i ograničenjima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Dinamičke promjenljive i rad s memorijom. Pokazivači i njihova uloga u dinamičkoj alokaciji. Struktura: čvor (node), veza (link), početni i završni pokazivač. Jednostruko povezana lista, kružna lista, binarno stablo, graf. Osnovne operacije: inicijalizacija, dodavanje, uklanjanje, pretraga elemenata. Poređenje statičkih i dinamičkih struktura.	
Preporuke za ostvarenje ishoda	
Vizualizacija struktura podataka: crtati liste, stabla i grafove sa prikazima čvorova i veza. Simulacija koraka algoritma: npr. dodavanje elementa u listu – učenici fizički povezuju „kartice“ s pokazivačima. Korištenje pseudo-koda: radi postupnog razumijevanja logike bez fokusa na sintaksu. Upotreba animiranih alata: Vizualizaciju pojačati kratkim animacijama (npr. Visualgo) i jednim trasiranjem u debuggeru da učenici vide „živi“ tok: vrijednosti pokazivača prije i poslije operacije, gdje nastaje greška ako se preskoči ažuriranje veze ili zaboravi delete. Paralelno poređenje: prikazati isti problem riješen uz pomoć niza i uz pomoć liste, zatim diskutovati o razlici. Rad u grupama: svaki tim razvija algoritam za jednu operaciju nad strukturom, zatim ga prezentira razredu. Istaknuti korelaciju s matematikom (stabla i grafovi u kombinatorici) i informatikom/arhitekturom računara (memorija i adresiranje), kako bi se dodatno učvstilo razumijevanje apstraktnih veza između modela i njihove realizacije u programu.	

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.II.1. Manipuliše elementima jednodimenzionalnih i višedimenzionalnih nizova.	● Piše deklaracije i inicijalizacije jednodimenzionalnih i višedimenzionalnih nizova u C++. ● Primjenjuje indeksiranje za pristup pojedinačnim elementima niza. ● Primjenjuje petlje za unos, obradu i ispis elemenata niza i matrica. ● Primjenjuje operacije kao što su suma, prosjek, maksimum/minimum, transponovanje matrice. ● Implementira jednostavne algoritme za pretraživanje i sortiranje elemenata niza.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Jednodimenzionalni i dvodimenzionalni nizovi (matrice). Deklaracija, inicijalizacija i pristup elementima. Obrada nizova pomoću petlji (for, while). Osnovne operacije nad nizovima (računanje sume, prosjeka, sortiranje, pretraga). Obrada i prikaz matrica (unos, ispis, transponovanje).	
Preporuke za ostvarenje ishoda	
Zadaci sa realnim kontekstom: kroz kratke, realne scenarije (lista ocjena, temperature tokom sedmice, raspored sjedenja u učionici) učenici najprije olovkom upisuju i čitaju vrijednosti po zadatim indeksima, zatim isti tok prenose u C++, deklaracijom, inicijalizacijom i pristupom elementima. Grafički prikaz strukture: koristiti tabele da bi se lakše vizualizirao izgled niza i matrice. Vođena implementacija u učionici: učenici korak po korak pišu kod uz objašnjenja nastavnika. Upoređivanje petlji: rješavanje istog problema pomoću različitih struktura ponavljanja. Učenici rade u parovima: jedan opisuje postupak riječima i osmišljava test-slučajeve (element na početku/sredini/kraju, nepostojeći element, duplikati), drugi implementira i provjerava, pa mijenjaju uloge. Greške kao alat za učenje: analizirati pogreške kod pristupa elementima izvan granica niza.	

Mini-projekti: npr. unos i prikaz matrice, pronalazak elementa najveće vrijednosti, transponovanje kvadratne matrice.	
B.II.2. Koristi vektore i stringove za pohranu i obradu podataka.	● Piše deklaracije i incijalizacije vektorai stringova u C++ (<vector>, <string>). ● Primjenjuje indeksiranje za pristup elementima vektora i stringa. ● Primjenjuje osnovne metode i funkcije za obradu vektora (dodavanje, uklanjanje, sortiranje, rotiranje, sažimanje). ● Primjenjuje funkcije iz biblioteke za rad sa stringovima (npr. length(), substr(), append(), find()). ● Kombinuje rad s vektorima i stringovima u jednostavnim programskim zadacima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Vector i string klase u C++. Inicijalizacija i pristup elementima vektora i stringova. Operacije nad vektorima: dodavanje (push_back), uklanjanje (erase), sortiranje, rotiranje. Operacije nad stringovima: pristup znakovima, dužina (length()), izdvajanje podstringa (substr()), spajanje (append()), pretraga (find()). Petlje i funkcije za obradu podataka u vektorima i stringovima.	
Preporuke za ostvarenje ishoda	
Prikaz realnih primjera: npr. manipulacija listom imena (vektor stringova), analiza rečenice (rad sa znakovima). Korištenje biblioteka: uvesti praksu uključivanja <vector> i <string> uz objašnjenje njihove uloge. Zadaci za vježbu: sortiranje brojeva u vektoru, pronalaženje riječi u stringu, spajanje dva stringa. Vođene demonstracije: pokazati razliku između statičkih nizova i fleksibilnosti vektora. Rad u parovima: jedan učenik kreira strukturu podataka, drugi implementira operacije. Igre i kvizovi: npr. pravljenje programa za provjeru palindroma, brojanje riječi u rečenici, manipulacija karakterima.	
B.II.3. Implementira osnovne operacije nad dinamičkim strukturama podataka (pokazivači, liste, stabla).	• Objašnjava osnovne pojmove: pokazivač, čvor, veza, korijen stabla. • Primjenjuje pokazivače za dinamičko zauzimanje i oslobađanje memorije (new, delete). • Kreira i povezuje čvorove jednostruko povezane liste. • Primjenjuje osnovne operacije nad listama: dodavanje, uklanjanje i prikaz elemenata. • Implementira osnovne operacije nad binarnim stablom (npr. umetanje čvora, ispis preorder/inorder/postorder). • Objašnjava razliku između linearnih (liste) i hijerarhijskih (stabla) struktura.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Pokazivači i operatori * i &. Dinamička alokacija memorije: new, delete. Struktura čvora (npr. struct Node { int data; Node* next; };). Jednostruko povezane liste: kreiranje, dodavanje, brisanje, ispis. Osnove binarnog stabla: pojam korijena, lijevog i desnog podstabla. Operacije nad stablima: umetanje čvorova, pretraga, ispis u različitim redoslijedima.	
Preporuke za ostvarenje ishoda	

Započeti uvod u ishoe kroz prepoznatljive situacije u kojima su „fleksibilni nizovi" prirodan izbor: lista imena razreda, evidencija bodova na kvizu ili analiza jedne rečenice. Najprije jasno pokazati razliku u odnosu na statičke nizove: uključivanjem zaglavlja <vector> i <string> dobijamo strukture koje se same šire i smanjuju, pa učenici bez brige o fiksnoj dužini koriste push_back za dodavanje, a zatim preko indeksa pristupaju elementima i provjeravaju granice brojeći od nule do size()-1. Kroz kratke, razgovorne zadatke učenici manipulišu vektorom brojeva (dodaju, brišu erase-om, poređuju sort-om, rotiraju rotate-om) i odmah vizualizuju promjene na mini-primjerima, dok se kod stringa usredsređuju na smislen rad s tekstom: mjere dužinu length()-om, izdvajaju dio naslova substr()-om, spajaju append()-om i traže uzorak find()-om.
Učenike naviknuti da prije svake operacije „pročitaju" šta žele postići u jeziku problema, a potom to prevedu u C++ korake; tako se spontano grade dobri obrasci za indeksiranje, provjeru granica i izbor prikladnih metoda.
Korištenje vizualnih pomagala: crtanje struktura lista i stabala sa pokazivačima između čvorova.
Postepena gradnja koda: od deklaracije strukture čvora do povezivanja i ispisa elemenata.
Simulacija memorije: rad sa „papirnim" čvorovima i pokazivačima u razredu.
Rad u grupama: svaka grupa implementira jednu operaciju (dodavanje, ispis, brisanje), zatim ih integrišu.
Zadaci s greškama: analiza i ispravljanje koda koji sadrži tipične greške sa pokazivačima.
Korištenje online alata (npr. Visualgo.net): vizualizacija ponašanja struktura u realnom vremenu.

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.II.1. Organizuje strukturu programa korištenjem funkcija s parametrima i povratnim vrijednostima.	• Piše funkcije s odgovarajućim povratnim tipom i listom parametara. • Razlikuje formalne i stvarne (realne) parametre funkcije. • Primjenjuje prosljeđivanje argumenata po vrijednosti i po referenci pri pozivu funkcije. • Objašnjava razliku između lokalnih i globalnih promjenljivih u kontekstu funkcija. • Organizuje program u više funkcionalnih cjelina (funkcija) radi bolje modularnosti i čitljivosti. • Primjenjuje return izraz za vraćanje rezultata iz funkcije
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Definisanje funkcija: zaglavlje, tijelo funkcije, return vrijednosti. Parametri funkcija: formalni i stvarni, prosljeđivanje po vrijednosti i po referenci. Povratni tipovi: void, int, float, string. Modularizacija programskog koda. Opseg promjenljivih: lokalne i globalne promjenljive.	
Preporuke za ostvarenje ishoda	
Demonstracija modularnog pristupa: jedan veći zadatak podijeliti u manje funkcije (npr. unos, obrada, ispis). Praktični zadaci: izrada funkcije koja vraća maksimum iz dva broja, računanje prosjeka, validacija unosa itd. Poređenje načina prosljeđivanja: zadaci u kojima učenici posmatraju efekat promjene vrijednosti nad parametrom. Tipične greške (zaboravljen return, neusklađen povratni tip, pogrešan redoslijed parametara, oslanjanje na globalne varijable) tretiraju se kao prilika za učenje kroz ciljane mikro-primjere i kratke ispravke. Za završnu provjeru smislen je mali zadatak u kojem učenik dizajnira tri povezane funkcije (unos, obrada, ispis) s barem jednim parametrom po referenci, jasno razlikuje formalne i stvarne parametre te u dvije–tri rečenice objasni kako je modularizacija poboljšala čitljivost i testiranje. Prirodna je korelacija s Matematikom (funkcija kao preslikavanje ulaza u izlaz) i bosanskim jezikom (jasna, smisleno imenovana „ugovorna" specifikacija svake funkcije).	
C.II.2. Razlikuje lokalne i globalne promjenljive u kontekstu funkcionalnog toka programa.	• Opisuje pojmove lokalne i globalne promjenljive. • Objašnjava opseg važenja (scope) promjenljivih u okviru funkcija i izvan njih. • Ustanovljava konflikte i greške koje nastaju usljed preklapanja imena promjenljivih.

	• Uspoređuje ponašanje lokalnih i globalnih promjenljivih tokom izvršavanja programa. • Koristi lokalne i globalne promjenljive u skladu sa zahtjevima zadatka.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Opseg (scope) promjenljivih: lokalni i globalni kontekst.
Pravila deklaracije i vidljivosti promjenljivih.
Životni vijek promjenljive (lifetime).
Sukobi imena i senčenje (shadowing).
Primjeri interakcije između lokalnih i globalnih promjenljivih.
Praktični primjeri u C++.

Preporuke za ostvarenje ishoda

Uvod u temu započeti kroz mali program u kojem postoje dvije promjenljive istog imena na različitim mjestima: jedna deklarisana izvan svih funkcija (globalna), druga unutar main-a ili pomoćne funkcije (lokalna).
Kroz kratko „praćenje toka“ učenici bilježe gdje je koja promjenljiva vidljiva i koliko dugo traje.
Praktična demonstracija u IDE-u: učenici mijenjaju vrijednosti promjenljivih unutar i izvan funkcija.
Korištenje dijagrama vidljivosti: vizualni prikaz hijerarhije i važenja promjenljivih u programu.
Zadaci za upoređivanje: isti program sa lokalnim i sa globalnim promjenljivim – uočiti razliku u ponašanju.
Zadaci sa greškom: analizirati kod u kojem dolazi do konflikta između lokalne i globalne promjenljive.
Grupna diskusija: prednosti i mane korištenja globalnih promjenljivih, koncept „dobre prakse“ u programiranju.
Vođena refleksija: kada i zašto koristiti lokalne promjenljive radi sigurnosti i modularnosti koda.

C.II.3. Primjenjuje osnovne postupke za rad s datotekama u cilju čuvanja i razmjene podataka.	• Objašnjava svrhu rada s datotekama u kontekstu trajnog čuvanja podataka. • Primjenjuje tokove ofstream i ifstream iz biblioteke <fstream>. • Primjenjuje postupke otvaranja i zatvaranja tekstualnih datoteka za upis i čitanje. • Primjenjuje operacije upisa i čitanja podataka korištenjem naredbi (<<, >>, .open(), .close()). • Rješava jednostavne programske zadatke koji uključuju rad s datotekama (npr. čuvanje unosa, ispis sadržaja). • Rješava tipične greške prilikom rada s datotekama (npr. neuspješno otvaranje datoteke).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Uključivanje biblioteke <fstream>.
Objekti ofstream (output file stream) i ifstream (input file stream).
Metode .open(), .close().
Operatori za upis i čitanje (<<, >>).
Provjera uspješnosti otvaranja datoteke (.is_open()).
Osnovni primjeri rada s tekstualnim datotekama (upis i čitanje linija, brojeva, stringova).

Preporuke za ostvarenje ishoda

Učenike najprije usmjerite na ideju da datoteka omogućava trajno čuvanje podataka izvan memorije programa: ono što se unese danas mora biti dostupno i poslije ponovnog pokretanja. U kratkoj demonstraciji u IDE-u pokažite tok rada „od nule“: uključivanje <fstream>, kreiranje objekta za upis, pokušaj otvaranja i provjera uspješnosti, upis nekoliko vrijednosti i uredno zatvaranje; zatim simetrično čitanje iz iste datoteke uz prikaz rezultata na ekranu. Posebno naglasiti naviku provjere stanja toka (.is_open() ili provjera uslova toka) i smisleno rukovanje greškama – ako otvaranje ne uspije, program treba jasno javiti korisniku šta se desilo i predložiti rješenje (npr. provjeriti putanju ili dozvole).
Realni zadaci: npr. unos i čuvanje liste imena u datoteku, ispis sadržaja datoteke u konzolu.
Vođena demonstracija: korak-po-korak prikaz otvaranja datoteke, pisanja i zatvaranja.
Upotreba IDE-a: uočavanje grešaka pri nepostojećim ili neotvorenim datotekama.

Rad u parovima: jedan učenik piše funkciju za upis, drugi funkciju za čitanje, zatim ih testiraju zajedno. Diskusija o prednostima datoteka: zašto je važno sačuvati podatke i kada to radimo. Zadaci sa simulacijom datoteka: rad sa stvarnim .txt fajlovima u učionici – unos podataka i ručna provjera sadržaja.

3. razred IT gimnazije /2 nastavna časa sedmično/70 nastavnih časova godišnje/

Oblast: A/Algoritmi	
Ishod učenja	**Razrada ishoda**
A.III.1. Analizira i primjenjuje osnovne koncepte objektno orijentisanog programiranja u dizajnu jednostavnih programskih rješenja.	● Objašnjava osnovne principe OOP-a: klasa, objekt, enkapsulacija, nasljeđivanje, polimorfizam. ● Identifikuje komponente klase: atribute, metode, konstruktore i destruktore. ● Povezuje koncept klase sa realnim objektima iz svakodnevnog života. ● Konstruiše jednostavne klase i objekte u C# jeziku. ● Primjenjuje konstruktore za inicijalizaciju objekata. ● Upotrebljava enkapsulaciju za zaštitu podataka i pristupa im putem metoda (get/set).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Pojmovi: klasa, objekt, atribut, metoda. Konstruktori (podrazumijevani, sa parametrima, kopije) i destruktori. Enkapsulacija: zaštita podataka, pristup putem metoda. Povezivanje stvarnih entiteta s klasama u programiranju. Razlika između klase i strukture u C#.	
Preporuke za ostvarenje ishoda	
Uvod u OOP graditi od svijeta koji je učenicima blizak: zajednički odaberiti jedan „stvarni“ entitet (npr. Automobil ili Učenik). Tražizi od učenika da najprije jezikom svakodnevice kažu šta taj entitet „ima“ (atributi poput boje, snage, prosjeka ocjena) i šta „radi“ (metode poput Pokreni, Zaustavi, IzračunajProsjek). Tek nakon toga to prevoditi u C# klasu u Visual Studiu: privatna polja i javne metode ili, idiomatski, svojstva (properties) sa get/set pristupnicima, uz kratko obrazloženje zašto enkapsulacija čuva ispravnost podataka (npr. odbijanjem negativne ocjene ili brzine). Konstruktore uvoditi postepeno: podrazumijevani za „prazan“ objekat i konstruktor s parametrima za smisleno početno stanje. Radionički pristup u Visual Studiu: učenici kreiraju jednostavne klase, instanciraju objekte i pozivaju metode. Vođeni zadaci: korak-po-korak implementacija klase sa enkapsulacijom i jednostavnim ponašanjem. Upotreba get i set metoda: umjesto direktnog pristupa atributima, uz objašnjenje sigurnosnih razloga. Grupni rad: svaki član tima dizajnira klasu za jedan dio zajedničkog projekta (npr. igrice – Igrac, Protivnik, Igra).	
A.III.2. Razvija algoritamsku logiku korištenjem nasljeđivanja, enkapsulacije i polimorfizma.	● Objašnjava pojam i svrhu nasljeđivanja u OOP-u. ● Kreira hijerarhiju klasa i primjenjuje koncept nadklase i podklase. ● Primjenjuje enkapsulaciju u baznoj i izvedenim klasama. ● Objašnjava koncept polimorfizma i koristi metode koje se ponašaju različito u različitim kontekstima. ● Koristi override i virtual ključne riječi u C# jeziku. ● Razvija algoritme u kojima se objekti različitih tipova obrađuju na zajednički način (polimorfizam u akciji).

Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Nasljeđivanje (: base, protected, override, virtual).
Enkapsulacija unutar hijerarhije klasa.
Polimorfizam: preklapanje metoda, pozivi preko referenci bazne klase.
Primjeri: klase Zivotinja → Pas, Macka s metodom Glas().
Praktične implikacije: proširivost i ponovna upotreba koda.

Preporuke za ostvarenje ishoda

Početi od male, razumljive domene. Zajedno s učenicima osmisliti hijerarhiju koja „traži“ nasljeđivanje: na primjeru Zaposleni kao bazne klase uvesti atribute i enkapsulaciju preko svojstava, a zatim izvesti Profesor i Administrator, vodeći računa da zajednički podaci i ponašanja ostanu u nadklasi, a specifičnosti odu u podklase. Učiniti polimorfizam vidljivim tako što u baznoj klasi definišemo metodu označenu kao virtual (npr. PrikaziDetalje()), a u izvedenim klasama je preklopimo s override, pa u glavnom programu obradimo kolekciju tipa List<Zaposleni> i pozovemo istu metodu nad različitim objektima. Učenici tako uočavaju da „jedan algoritamski tok“ prolazi kroz kolekciju i uniformno poziva metodu, dok se konkretno ponašanje mijenja u zavisnosti od stvarnog tipa objekta.
Kreiranje funkcija koje primaju referencu bazne klase: radi prikaza snage polimorfizma.
Vizualni prikaz hijerarhije: crtanje stabla nasljeđivanja, s prikazom preklopljenih metoda.
Zadaci u kojima učenici samostalno dizajniraju jednostavnu OOP hijerarhiju.
Kratki kvizovi i provjere znanja: fokus na prepoznavanje ispravne upotrebe override, virtual, base.

A.III.3. Implementira mehanizme rukovanja izuzecima u funkciji stabilnosti i kontrole toka programa.	● Objašnjava pojam izuzetka i razliku između greške i izuzetka. ● Identifikuje situacije u kojima može doći do izuzetaka (npr. dijeljenje nulom, pristup nedostupnom resursu). ● Primjenjuje ključne riječi try, catch, finally u jeziku C# za obradu izuzetaka. ● Upotrebljava odgovarajuće klase izuzetaka (Exception, DivideByZeroException, itd.) pri obradi izuzetaka. ● Povezuje rukovanje izuzecima sa stabilnošću i pouzdanošću programa.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Izuzetci (exceptions): definicija, primjer iz prakse.
Blokovi try, catch, finally.
Klase izuzetaka (System.Exception i izvedene klase).
Kreiranje vlastitih izuzetaka (osnovni nivo).
Uloga rukovanja izuzecima u profesionalnom softveru.

Preporuke za ostvarenje ishoda

Započeti sa primjerima koji pokazuju razliku između greške i izuzetka: sintaksna greška nastaje pri prevođenju i program se ne pokreće, dok izuzetak nastaje u toku izvođenja i može se kontrolisano obraditi. U Visual Studiu prikažite dvije situacije iz svakodnevnog rada: dijeljenje unesenim brojem gdje učenik ponekad unese nulu, te čitanje iz datoteke koja možda ne postoji. Isti zadatak najprije pokrenite bez zaštite da bi se pojavio izuzetak, a zatim uvedite try blok oko rizičnog dijela i hvatanje specifičnih izuzetaka.
Simulacija grešaka u kodu: npr. unos nule kao djelitelja, pokušaj otvaranja nepostojeće datoteke.
Analiza poruka o izuzetku: učenici objašnjavaju značenje i porijeklo prikazane greške.
Korištenje try–catch u svakodnevnim zadacima: npr. unos broja s tastature i obrada pogrešnog unosa.
Primjeri dobrog i lošeg rukovanja izuzecima: učenici komentarišu kodove i uče iz pogrešaka.
Diskusija o profesionalnom razvoju softvera: zašto su izuzeci ključni za korisničko iskustvo i stabilnost sistema.
Vježbe sa više catch blokova: obrada različitih vrsta izuzetaka u istom programu.

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.III.1. Modelira strukture podataka pomoću klasa i objekata u programskom jeziku C#.	● Konstruiše klase koje predstavljaju apstraktne i konkretne strukture podataka (npr. korisnik, proizvod, igra, figura). ● Dizajnira atribute i metode relevantne za modeliranu strukturu. ● Primjenjuje različite vrste konstruktora za inicijalizaciju objekata. ● Implementira stvaranje i upravljanje instancama objekata zasnovanim na klasama. ● Povezuje klase u funkcionalne cjeline (npr. Igra koristi Igrac, Tabla, Potez). ● Piše komentare koji opisuju strukturu podataka u kodu
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Klasa kao model realnog entiteta. Atributi i metode kao elementi ponašanja objekta. Inicijalizacija objekata pomoću konstruktora. Organizacija međusobno povezanih klasa (sastavljanje i korištenje objekata). Primjeri: klase Knjiga, Korisnik, Narudžba, Igrač, Pozicija. Pregled razlike između modeliranja statičkih i dinamičkih odnosa.	
Preporuke za ostvarenje ishoda	
Primjeri iz svakodnevnog života: krenuti od prepoznatljivog konteksta i zajednički s učenicima „izvući“ model iz stvarnosti: najprije na papiru ili tabli nacrtati jednostavan klasni prikaz (UML) za odabrani domen, npr. Biblioteka, tako da se jasno vide entiteti. Zatim taj model preslikajti u C# klasu s jasnim atributima i ponašanjima, pri čemu učenici umjesto javnih polja koriste enkapsulirana svojstva (properties) i smisleno imenovane metode. Konstruktore uvoditi postupno: podrazumijevani za „prazno“ početno stanje i konstruktor s parametrima za cjelovitu inicijalizaciju, uz kratko objašnjenje gdje je zgodno ugraditi provjere ispravnosti. U glavnom programu učenici instanciraju više objekata, mijenjaju njihova stanja kroz metode i promatraju kako se klase prirodno povezuju u funkcionalnu cjelinu Zadaci sa više klasa: uvesti relacije između klasa (has-a, uses-a) kroz praktične projekte. Diskusija o apstrakciji: zašto nije potrebno da sve klase sadrže detalje implementacije, već samo opis ponašanja. Projektni rad: modeliranje jednostavnog sistema (igrica, naručivanje hrane).	
B.III.2. Koristi reference, nabrojive i strukturne tipove podataka u cilju optimizacije programskog koda.	● Objašnjava razliku između vrijednosnih i referentnih tipova podataka u jeziku C#. ● Primjenjuje referentne tipove (npr. klase, nizovi, objekti) za fleksibilan i efikasan kod. ● Piše deklaracije nabrojivih tipova (enum) za ograničeni skup vrijednosti. ● Primjenjuje strukturne tipove (struct) u kontekstu jednostavnih, manje kompleksnih objekata. ● Uspoređuje upotrebu klasa i struktura u različitim scenarijima. ● Kombinuje reference i strukturne/nabrojive tipove u konkretnim zadacima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Vrijednosni i referentni tipovi podataka u C#. Reference i ponašanje objekata kroz referencu. Nabrojivi tipovi (enum) – definicija, dodjela i upotreba.	

Strukturni tipovi (struct) – razlika u odnosu na klase.
Scenariji korištenja enum i struct za čitljivost i organizaciju koda.
Povezivanje s OOP konceptima.

Preporuke za ostvarenje ishoda

Vizuelni prikaz razlika: tabela ili dijagram koji pokazuje razliku između class, struct i enum.
Rad na jednostavnim zadacima: npr. enum Boja { Crvena, Zelena, Plava } korišten u klasi Automobil.
Zadaci sa referencama: prikazati kako promjena nad objektom putem jedne reference utiče na sve reference ka istom objektu.
Diskusija u razredu: kada koristiti struct umjesto class – kriteriji za donošenje odluke.
Kombinovanje koncepata: npr. struktura koja koristi enum, klasa koja koristi pokazivač na tu strukturu.
Upareni rad: jedan učenik piše enum, drugi koristi vrijednosti u logici programa. U korelaciji s bosanskim jezikom treba istaći jasno, dosljedno imenovanje tipova i članova, a s matematikom povezati enum s konačnim skupovima vrijednosti.

B.III.3. Primjenjuje dinamičko kreiranje i upravljanje objektima u radu sa komponentama i podacima.	• Objašnjava koncept dinamičkog kreiranja objekata i komponenti u C#. • Kreira kontrole u toku izvršavanja programa koristeći operatore new i pokazivače na komponente. • Prilagođava svojstva komponenti dinamički (inicijalizaciju, pozicioniranje, atribute poput teksta i dimenzija). • Organizuje nizove i matrice pokazivača na komponente za sistematsko upravljanje većim brojem elemenata. • Upravlja životnim ciklusom dinamički kreiranih objekata (stvaranje i pravovremeno oslobađanje resursa). • Kombinuje logiku upravljanja podacima sa vizuelnim prikazom i interakcijom korisnika.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Dinamičko kreiranje objekata i komponenti (new, dodavanje kontrola u Form).
Pokazivači na komponente (npr. Button[], TextBox[,]).
Upravljanje svojstvima, događajima i rasporedom kontrola.
Nizovi i matrice pokazivača na kontrole.
Povezivanje sa logikom programa i podacima korisnika.
Primjeri: generisanje forme sa 10 dinamičkih dugmadi, kreiranje kviza, igrice itd..

Preporuke za ostvarenje ishoda

Zadaci sa postepenim proširivanjem: početi od jednog dugmeta, zatim ga pretvoriti u niz, pa u matricu. Početi od male, vidljive demonstracije u kojoj se na formi u toku izvršavanja kreira jedna kontrola pozivom new, postavljaju joj se osnovna svojstva (tekst, veličina, pozicija) i dodaje se u kolekciju Controls. Zatim isti postupak generički „zapakovati" u pomoćnu metodu (npr. KreirajDugme(int i)) i pozoviti je u petlji kako bi učenici uočili prednost sistematskog pristupa: niz Button[] za deset dugmadi, pa po potrebi i matrica TextBox[,] raspoređena u mrežu.
Demonstracija preko igara ili animacija: dinamički kreirani elementi koji reaguju na događaje (npr. klikni me igre).
Koristiti petlje za kreiranje većeg broja komponenti: učenici uvide prednosti sistematične dinamike.
Zadaci sa povezivanjem na podatke: npr. dinamičko kreiranje polja na osnovu broja učenika iz datoteke.
Povezivanje s događajima: svaki dinamički objekat reaguje na klik, tastaturu ili tajmer.
Vođena diskusija: zašto i kada koristiti dinamičko kreiranje – fleksibilnost, skalabilnost, kontrola toka.

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.III.1. Razvija grafički korisnički interfejs korištenjem komponenti iz biblioteke vizuelnih elemenata.	• Objašnjava osnovne pojmove GUI-a (grafički korisnički interfejs) i svrhu njegovog korištenja. • Identifikuje osnovne komponente GUI-a u C# (npr. Form, Label, Button, TextBox, Panel, CheckBox, RadioButton, ListBox, ComboBox). • Primjenjuje razvojno okruženje (Visual Studio) za postavljanje i podešavanje vizuelnih elemenata na formi. • Prilagođava svojstva komponenti (tekst, boja, veličina, položaj, font). • Kreira osnovni izgled aplikacije kroz raspored komponenti i njihovo grupisanje (npr. korištenje GroupBox, Panel). • Implementira događaje za osnovnu interakciju između korisnika i aplikacije (npr. Click, TextChanged).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Grafički korisnički interfejs – osnovni pojmovi. Osnovne GUI komponente: Label, TextBox, Button, ListBox, ComboBox, CheckBox, RadioButton, Panel, GroupBox. Podešavanje svojstava komponente: Name, Text, Enabled, Visible, BackColor, ForeColor, Font, Size, Location. Grupisanje komponenti i organizacija forme. Raspoređivanje elemenata za čitljivost i upotrebljivost.	
Preporuke za ostvarenje ishoda	
Rad u Visual Studiu kroz demonstraciju i vježbu: učenici prate korake nastavnika, a zatim samostalno primjenjuju naučeno. Zadaci za dizajn forme: npr. izraditi formular za prijavu korisnika, unos podataka, kviz ili mini-kalkulator. Korištenje storyboard tehnike: učenici prvo nacrtaju izgled forme na papiru prije izrade u IDE-u. Postepena izgradnja GUI-a: dodavanje jedne po jedne komponente uz objašnjenje funkcije i svojstava. Interakciju uvesti postupno: na jednostavnom primjeru formulara za prijavu ili mini-kalkulatora događaj Click čita vrijednosti iz TextBox-a, provjerava ih i ažurira Label, a TextChanged pokreće diskretnu provjeru unosa i daje povratnu informaciju korisniku. Učenici rade zadatak postupno od „prikaza“ tako što logiku smještaju u pomoćne metode, dok forma ostaje tanak sloj koji prikuplja ulaz i prikazuje rezultat; to rješenje olakšava testiranje i kasnije proširenje. Podsticanje estetskog razmišljanja: diskusija o dizajnu i upotrebljivosti – jednostavnost, čitljivost, boje. Uključiti kreativne zadatke: npr. uređivanje vlastite kontakt forme, aplikacije za evidenciju, igrice s GUI elementima.	
C.III.2. Organizuje interakciju između korisnika i programa kroz događaje tastature, miša i tajmera.	• Objašnjava osnovne pojmove događaja i upravljanja događajima u C# (event-driven programming). • Implementira događaje miša (MouseClick, MouseMove, MouseDown) i tastature (KeyDown, KeyPress, KeyUp). • Primjenjuje komponentu Timer za izvođenje akcija u pravilnim vremenskim intervalima. • Kreira jednostavne aplikacije koje reaguju na korisničke akcije (npr. pomicanje objekta tastaturom, promjena boje na klik miša). • Kombinuje više događaja za postizanje kompleksnijeg interaktivnog ponašanja aplikacije. • Analizira ponašanje programa u zavisnosti od korisničkog unosa i vremenskih događaja.

Poveznice sa ZJNPP	TIT-4.2.1 TIT-4.2.2

Ključni sadržaji

Događaji u C# i princip obrade događaja.
Osnovni događaji miša: Click, MouseMove, MouseDown, MouseUp.
Osnovni događaji tastature: KeyDown, KeyPress, KeyUp.
Komponenta Timer: svojstva (Interval, Enabled) i događaj Tick.
Veza između GUI komponente i događaja (+= new EventHandler(...)).
Primjeri: upravljanje objektom strelicama, reagovanje na klik, animacija pomoću Timer-a.

Preporuke za ostvarenje ishoda

Demonstracije u učionici: nastavnik pokazuje primjer reagovanja na događaj, učenici ga implementiraju i prilagođavaju. Nastavu započeti kratkim uvodom u „event-driven" logiku: program miruje dok ne stigne događaj, a naš posao je da na pravi element „prikačimo" odgovarajući obrađivač i u njemu jasno zapišemo šta se dešava. Učenike voditi od jednostavnog do složenijeg: najprije na formi postaviti nekoliko osnovnih komponenti (npr. Panel kao „scenu" i Label kao „lik"), pa pokazati kako MouseClick mijenja boju lika, MouseMove prikazuje koordinate kursora, a KeyDown/KeyUp pomjeraju lik strelicama.
Zadaci s konkretnim ciljem: npr. napraviti program gdje se kvadrat pomjera strelicama, a tajmer ga automatski vraća u početni položaj.
Igre i animacije kao motivacija: učenici izrađuju mini-igrice sa jednostavnim mehanizmima (klikni da pogodiš, reakcija na tastaturu).
Praktične vježbe za povezivanje teorije s praksom: učenici testiraju efekte različitih događaja na istu komponentu.
Vođena diskusija: kako događaji omogućavaju fleksibilno i responzivno ponašanje korisničkog interfejsa.
Rad u paru: jedan učenik definiše izgled i osnovne komponente, drugi piše događaje za njih.

C.III.3. Kreira jednostavne interaktivne programe i animacije korištenjem pokazivača na komponente i nizova.	• Piše deklaracije nizova i matrica pokazivača na komponente (Button[], PictureBox[,] itd.). • Prilagođava svojstva i funkcionalnost komponenti kroz petlje (inicijalizacija, pozicioniranje, dodjela uloga). • Povezuje dinamički kreirane komponente s odgovarajućim događajima (npr. klik, tajmer). • Organizuje logiku programa u skladu sa strukturom komponenti i njihovim odnosima. • Kombinuje tajmer, tastaturu i događaje miša za osnovnu animaciju ili interaktivnu igru. • Rješava nepravilnosti u ponašanju komponenti na osnovu rezultata testiranja.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Nizovi i matrice pokazivača na GUI komponente (Label, Button, PictureBox, TextBox, itd.).
Dinamičko kreiranje i upravljanje komponentama u petlji.
Dodjeljivanje događaja komponentama pomoću delegata.
Upotreba Timer komponente za animaciju.
Koordinacija logike i vizuelnog prikaza kroz nizove.
Primjeri: igrica memorije, kviz, klikni me, zmija, vremenske reakcije.

Preporuke za ostvarenje ishoda

Projektni zadaci: izrada jednostavnih igara (npr. klikni na pravo dugme, kviz s više ponuđenih odgovora).
Krenuti od najmanjeg mogućeg primjera i postepeno ga pretvarati u sistem: jedan vizuelni element najprije dinamički kreirajti pozivom new, podesiti mu osnovna svojstva i dodajti ga na formu, a zatim isti postupak generalizovati u petlju kako bi dobili niz ili matricu komponenti (npr. Button[] ili PictureBox[,]). Svakoj komponenti odmah dodijeliti prepoznatljiv identitet—koristiti indeks ili par indeksa (red, kolona) upisan u svojstvo Tag—i vezati isti obrađivač događaja; u handleru prepoznati „ko je kliknut" kroz sender i kastovanje. Paralelno graditi i „model podataka" (niz cjelobrojnih ili tekstualnih vrijednosti) koji odražava stanje igre/zadatka, pa GUI niz držiti samo kao prikaz tog modela. Time učenici prirodno uče da logiku organizuju oko strukture komponenti i njihovih odnosa: petlja kreira, indeksira i pozicionira, događaj čita ili mijenja stanje, a ekran se osvježava iz modela. Kada se obrazac ustali, jednostavne projekte (memorija, kviz, „klikni me") širiti inkrementalno: miješanje elemenata, brojanje pogođenih parova, zaključavanje kliknutih polja i sl.

Za animacije i kompleksniju interakciju povežite tastaturu, miš i tajmer u jedinstven tok: KeyDown postavlja smjer (uz KeyPreview = true), Timer.Tick u pravilnim intervalima pomjera „igrača“ ili element scene i provjerava sudare/ivice, a MouseClick mijenja stanje ili pokreće posebnu akciju. Učenike vodite da sistematski testiraju i otklanjaju nepravilnosti: provjera granica indeksa pri pristupu matrici, usklađivanje Tag vrijednosti nakon preslagivanja elemenata, gašenje tajmera kad scena miruje, te kratki tragovi ispisa ili breakpointi za uvid u tok. Na kraju svake iteracije tražiti kratku refleksiju: kako je niz komponenti povezan s nizom podataka, kojim događajima se upravlja tokom i šta je promijenjeno nakon testiranja da bi se ponašanje stabilizovalo i postalo predvidivo.

4. razred IT gimnazije /2 nastavna časa sedmično/60 nastavnih časova godišnje/

Oblast: A/Algoritmi	
Ishod učenja	**Razrada ishoda**
A.IV.1. Primjenjuje skriptne programske jezike za obradu korisničkih zahtjeva i razvoj dinamičkih web funkcionalnosti.	● Objašnjava razliku između klijentskih (JavaScript) i serverskih (PHP) skriptnih jezika. ● Integriše JavaScript i PHP kod u HTML stranicu i objašnjava njihovu ulogu u različitim fazama izvršavanja. ● Primjenjuje PHP za obradu podataka sa forme (GET i POST metode). ● Primjenjuje JavaScript za validaciju korisničkog unosa prije slanja na server. ● Kombinuje klijentsku i serversku logiku u jednostavnim zadacima (npr. registracija, prijava, kalkulator). ● Provjerava funkcionalnost aplikacije i prepoznaje gdje se izvršava koji dio koda (preglednik vs. server).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Osnove skriptnih jezika: JavaScript (klijent), PHP (server). Umetanje JavaScript i PHP koda u HTML. GET i POST metode. Obrada formi i validacija podataka. Interakcija korisnik–server: primjer rada web forme. Zajedničko korištenje HTML, JavaScript i PHP-a.	
Preporuke za ostvarenje ishoda	
Početni zadaci sa jednostavnim formama: npr. unos imena i prikaz poruke u PHP-u, provjera praznog polja u JavaScript-u. Uporedna demonstracija: pokazati isti zadatak s validacijom u JavaScript-u i u PHP-u – gdje, kada i zašto koristiti oba. Rad u parovima: jedan učenik radi klijentsku validaciju, drugi serversku obradu, zatim zajedno integrišu rješenje. Simulacija protoka podataka: učenici crtaju tok zahtjeva od korisnika do servera i nazad. Testiranje na lokalnom serveru (XAMPP ili sl.): učenici odmah vide kako server odgovara na zahtjev. Diskusija o prednostima i ograničenjima skriptnih jezika u web programiranju.	
A.IV.2. Kreira funkcionalne dijelove aplikacije koristeći uslovne izraze, petlje i funkcije u JavaScript-u i PHP-u.	● Prepoznaje sintaksu uslovnih izraza i petlji u JavaScript-u i PHP-u. ● Primjenjuje grananje (if, else, switch) za upravljanje tokom izvršavanja u skriptnim jezicima.

	● Primjenjuje petlje (for, while, foreach) za obradu podataka. ● Piše funkcije s parametrima i povratnim vrijednostima. ● Razlikuje lokalni i globalni opseg promjenljivih unutar funkcija. ● Kombinuje funkcije, petlje i uslove u izgradnji modularnog i funkcionalnog dijela aplikacije.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Sintaksa i struktura uslovnih izraza i petlji u JavaScript-u i PHP-u.
Tipične funkcije: validacija, obrada unosa, formatiranje podataka.
Deklaracija i poziv funkcija.
Prosljeđivanje parametara i povratne vrijednosti.
Opseg promjenljivih (var, let, const u JS; lokalne/globalne u PHP).
Kombinovanje više struktura u jednoj funkciji.

Preporuke za ostvarenje ishoda

Zadaci s realnim primjerima: npr. provjera ispravnosti lozinke, brojanje brojeva u nizu, kreiranje tablice na osnovu korisničkog unosa.
Pisanje paralelnih primjera u JS i PHP: isti zadatak (npr. zbir brojeva od 1 do n) prikazati u oba jezika i analizirati razlike.
Rad u parovima ili grupama: jedna grupa implementira funkciju u JS, druga u PHP, zatim razmjena i analiza.
Kreiranje jednostavnog "modula": funkcije koje se pozivaju iz forme (npr. kalkulator sa više operacija).
Učenicima kojima je potrebna podrška ponudite početni kostur sa potpisima funkcija i mjestima za grane i petlje, dok napredniji mogu preformulisati ponovljene dijelove u pomoćne funkcije.
Postavljanje pitanja "šta ako?": mijenjanje uslova i praćenje uticaja na tok programa.
Upotreba vizualnih sredstava: dijagram toka koji prikazuje povezane funkcije i odluke unutar programa.

A.IV.3. Organizuje logiku programa u skladu s principima modularnosti i jasnoće server-klijent arhitekture.	● Objašnjava osnovne principe modularnog programiranja i razdvajanja funkcionalnosti. ● Organizuje HTML, CSS, JavaScript i PHP kod u odvojene datoteke i dijelove. ● Primjenjuje include/require naredbe u PHP-u za dijeljenje zajedničkog koda (npr. header.php, footer.php). ● Strukturira logiku klijenta i servera u odvojene slojeve (interakcije/validacije naspram obrade/pohrane). ● Razvija aplikaciju tako da je lako razumljiva, nadogradiva i ponovo upotrebljiva. ● Primjenjuje principe dobre strukture projekta (nazivi datoteka, direktorija, dokumentacija).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Modularnost u web programiranju.
Podjela između klijentskog i serverskog koda.
Organizacija koda: odvojeni fajlovi za HTML, CSS, JS, PHP.
Korištenje include, require, require_once.
Struktura projekta: index.php, form.php, db.php, scripts.js, style.css, itd..
Praktični primjeri: login forma, kontakt forma, sistem za unos i prikaz podataka.

Preporuke za ostvarenje ishoda

Primjeri dobre prakse: pokazati primjer dobro strukturiranog i loše organizovanog projekta; uporediti čitljivost i fleksibilnost.
Nastavu je korisno voditi kroz „razlaganje“ jedne male web funkcionalnosti na jasne slojeve, tako da učenici istovremeno uče modularnost i klijent–server logiku. Početi namjerno „zbijenim“ primjerom u kojem su HTML, CSS, JavaScript i PHP pomiješani u jednoj datoteci, a zatim korak po korak refaktorisati: izgled ostaje

u HTML-u i style.css-u, interakcija i validacija prelaze u scripts.js, dok obrada unosa i pohrana podataka pripadaju PHP-u.
Praktičan zadatak: učenici organizuju svoj web projekat koristeći jasno strukturirane fajlove i direktorije.
Rad u timovima: podjela rada – jedan učenik zadužen za klijentsku logiku, drugi za serversku, treći za izgled.
Vođena analiza: razlaganje postojećeg projekta na funkcionalne cjeline i prikaz njihove međusobne povezanosti.
Upotreba skica ili mapa sajta: za planiranje funkcionalnosti i strukture projekta.
Refleksivni zadaci: učenici opisuju u čemu im modularna organizacija pomaže i kako doprinosi razumljivosti koda.

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.IV.1. Struktuira i upravlja podacima u relacijskim bazama korištenjem SQL jezika i povezivanjem sa PHP-om.	● Objašnjava osnovne pojmove relacijskih baza podataka (tabela, red, kolona, primarni ključ). ● Kreira jednostavne baze i tabele u MySQL okruženju. ● Piše osnovne SQL naredbe za unos, čitanje, izmjenu i brisanje podataka (INSERT, SELECT, UPDATE, DELETE). ● Povezuje PHP skriptu sa bazom podataka pomoću mysqli ili PDO biblioteka. ● Provodi operacije nad bazom kroz PHP forme i prikazuje rezultate u web aplikaciji. ● Rješava i rješava osnovne greške u povezivanju i radu sa bazom.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Relacijski model podataka. SQL naredbe: CREATE, INSERT, SELECT, UPDATE, DELETE. MySQL alati (npr. phpMyAdmin, MySQL shell). Povezivanje PHP-a sa bazom (mysqli_connect, query, fetch_assoc). Izvršavanje SQL naredbi iz PHP-a. Prikaz rezultata u HTML tabeli.	
Preporuke za ostvarenje ishoda	
Praktični rad uz phpMyAdmin: kreiranje baze, unos testnih podataka, vizuelno učenje strukture. Učenike voditi od modela ka funkcionalnoj aplikaciji: najprije zajedno skicirati mali ER-dijagram (npr. „Korisnik“ sa ključem i nekoliko atributa), pa u phpMyAdminu kreirati bazu i tabelu te unijeti nekoliko probnih redova. Na tim podacima graditi osjećaj za SQL - tako što učenici najprije čitaju i filtriraju (SELECT s jednostavnim WHERE i ORDER BY), zatim dopunjavaju (INSERT), mijenjaju (UPDATE) i brišu (DELETE), uz kratko objašnjenje zašto je primarni ključ garancija prepoznatljivog reda. Tek kada je jezik upita postao razumljiv, preći na povezivanje s PHP-om: u maloj, čitljivoj skripti uspostaviti vezu (po mogućnosti PDO, uz try–catch i uključeni „exception mode“) i iz već postojeće HTML forme pošalati podatke POST metodom. Zadaci u parovima: jedan učenik piše SQL upite, drugi ih integrira u PHP aplikaciju. Vizualizacija relacija: crtanje ER dijagrama za jednostavne sisteme (npr. učenici i predmeti). Upotreba try–catch blokova za povezivanje: učenici testiraju spajanje i rješavaju greške. Povezivanje sa realnim primjerima: npr. sistem za rezervaciju, unos komentara, prijava korisnika. Diferencirani pristup: napredniji učenici koriste pripremljene upite i parametrizaciju za sigurnije rukovanje.	
B.IV.2. Koristi HTML forme i serverski kod za unos, validaciju, prikaz i ažuriranje podataka.	● Kreira HTML forme za unos različitih tipova podataka (tekst, brojevi, datumi, izbori). ● Povezuje forme sa PHP skriptama pomoću action i method atributa. ● Primjenjuje $_GET i $_POST za dohvat podataka iz forme. ● Primjenjuje validaciju unosa u JavaScriptu (klijent) i PHP-u (server).

	● Provodi upis podataka u bazu i prikaz rezultata u HTML tabeli. ● Izvodi ažuriranje i brisanje prethodno unesenih podataka kroz web interfejs.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

HTML elementi forme: <form>, <input>, <textarea>, <select>, <button>.
Atributi: name, value, method, action.
Prikupljanje podataka: $_POST, $_GET.
Validacija unosa: JavaScript (osnovna), PHP (obavezna polja, tip podataka).
Umetanje podataka u bazu i prikaz rezultata.
Ažuriranje i brisanje podataka pomoću forme.

Preporuke za ostvarenje ishoda

Nastavu graditi „u fazama“, tako da učenici jasno vide tok podataka od korisnika do servera i nazad. Početi jednostavnom formom (npr. rezervacija termina) u kojoj se promišljeno biraju tipovi polja — tekst, e-pošta, broj, datum, izbor iz liste — uz jasne natpise i name atribute, pa se forma veže na PHP skriptu preko action i odgovarajuće method vrijednosti. U pregledniku najprije radi brza provjera u JavaScriptu (obavezna polja, raspon broja, datum u budućnosti), ali se odmah naglašava da server ponavlja ključne provjere u PHP-u jer klijentska validacija nikad nije dovoljna.
Učenici u DevTools/Network kartici prate razliku između GET i POST zahtjeva, vide gdje se podaci nalaze i objašnjavaju zašto se osjetljivi unosi šalju POST-om. Tek kada je unos ispravan, PHP skripta podatke čita iz $_POST/$_GET, zapisuje u bazu i vraća jasnu poruku te ažurirani prikaz podataka u tabeli; ovdje se njeguje uredan HTML ispis s kolonama i zaglavljima, što prirodno povezuje Informatiku s bosanskim jezikom (kratke, razumljive poruke i nazivi polja).
Analiza sigurnosti: diskusija o osnovnim prijetnjama (npr. direktan unos SQL naredbi).
Rad u fazama: kreirati formu → povezati s PHP-om → dodati validaciju → pohraniti u bazu → prikazati podatke.
Zadaci sa greškom: učenici analiziraju i otklanjaju greške u povezivanju forme i servera.
Učenje kroz refaktorisanje: unapređenje forme (npr. dodavanje select opcije, više tipova inputa).

B.IV.3. Primjenjuje osnovne sigurnosne mehanizme za zaštitu podataka i korisničkog unosa.	● Objašnjava osnovne sigurnosne prijetnje u web aplikacijama (npr. SQL injection, XSS). ● Prepoznaje ranjive tačke u aplikaciji povezane s korisničkim unosom i radom sa bazom. ● Primjenjuje validaciju i filtriranje korisničkog unosa na strani klijenta i servera. ● Primjenjuje PHP funkcije za zaštitu od SQL injekcija (npr. mysqli_real_escape_string) ● Primjenjuje funkcije za filtriranje korisničkog unosa. ● Demonstrira pravilno rukovanje osjetljivim podacima i pristupom (npr. login, session).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Osnovne prijetnje: SQL injection, XSS, krađa podataka.
Validacija i sanitizacija podataka.
PHP funkcije za zaštitu ulaza (htmlspecialchars, trim, strip_tags).
SQL zaštita: korištenje prepared statements sa mysqli ili PDO.
Razlika između klijentske i serverske zaštite.
Primjeri sigurnog i nesigurnog koda.

Preporuke za ostvarenje ishoda

Tema se otvara kratkim „negativnim“ primjerom: namjerno jednostavna forma za prijavu ili pretragu bez ikakve zaštite, gdje učenici vide kako se nesiguran unos prelije direktno u SQL upit ili u HTML ispis. Nastavnik tada demonstrira tipične simptome napada (npr. unos ' OR '1'='1 u polje za korisničko ime ili umetanje <script> u komentar) i odmah ih prevede u pojmove SQL injection i XSS, uz naglasak da klijentska validacija služi samo za ugodniji doživljaj, dok se stvarna zaštita mora provesti na serveru. Isti primjer se zatim popravlja sloj po sloj: na klijentu JavaScript provjerava obavezna polja i format, a na serveru PHP prvo

normalizira i validira podatke (trim, provjera tipa i raspona, npr. filter_input), potom ih unosi u bazu isključivo preko pripremljenih upita (PDO ili mysqli prepared statements) umjesto spornog „spajanja stringova“. Za izlaz prema pregledniku svaki korisnički sadržaj se „iskačući“ štiti (htmlspecialchars(..., ENT_QUOTES, 'UTF-8')) kako bi se eliminisalo izvršavanje umetnutog koda. Učenici eksplicitno razlikuju tri koraka: validacija (da li je unos smislen), sanitizacija (uređivanje formata) i „output escaping“ (siguran prikaz). Primjena u postojećim formama: proširivanje prethodnih zadataka dodatnim slojem sigurnosti. Diskusija o stvarnim napadima: (primjereno uzrastu) – zašto sigurnost nije tehnička opcija, već nužnost. Grupni rad: timovi analiziraju aplikaciju i predlažu mjere zaštite za unesene podatke. Uvođenje termina poput “input sanitization” i “output escaping” kroz praktične primjere.

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.IV.1. Razvija interaktivan korisnički interfejs korištenjem HTML-a, CSS-a i JavaScript-a.	• Organizuje HTML dokument koristeći semantičke elemente (header, nav, main, footer, itd.). • Primjenjuje CSS pravila za izgled i raspored elemenata (boje, margine, fontovi, pozicioniranje). • Povezuje CSS s HTML dokumentom putem <style> ili eksternih stilskih datoteka. • Koristi JavaScript za dodavanje dinamike (npr. promjena sadržaja, validacija forme, interakcija na klik). • Prilagođava DOM strukturou za izmjenu elemenata u realnom vremenu. • Kombinuje HTML, CSS i JavaScript u funkcionalnu i estetski prihvatljivu cjelinu.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Struktura HTML stranice (elementi, atributi, forme). Osnovni i napredni CSS selektori, pravila i pozicioniranje (display, flex, grid). Uvod u JavaScript: promjenljive, funkcije, događaji. DOM manipulacija (getElementById, innerHTML, addEventListener). Validacija formi i jednostavne vizualne reakcije. Primjeri: kalkulator, meni na klik, dinamički unos.	
Preporuke za ostvarenje ishoda	
Nastavu započnite kratkim „žičanim okvirom“ (wireframe) na papiru: učenici najprije rasporede sadržaj u semantičke cjeline — zaglavlje sa navigacijom, glavni dio i podnožje — pa taj raspored vjerno preslikaju u HTML. Već u prvoj iteraciji razdvojite slojeve: izgled ide u eksterni style.css, ponašanje u script.js, a u HTML-u ostaje čista struktura s jasnim klasama i imenima elemenata. Učenici postepeno grade raspored prvo s flex-om, a zatim s jednostavnim grid-om, vodeći računa o tipografiji, kontrastu boja i razmacima. Dinamiku uvoditi malim koracima: u script.js dodaje se osluškivanje događaja preko addEventListener. Korištenje browser developer alata (F12): učenici prate promjene DOM-a uživo. Primjeri iz svakodnevnog korištenja weba: analiziranje UI elemenata poznatih stranica (prijava, registracija, dropdown meni...). Zadaci u kojima učenici kombinuju tehnologije: svaka promjena u unosu reflektuje se kroz JavaScript i stilizira kroz CSS. Estetski izazovi: učenici dizajniraju „najljepši“ kontakt formular koji uključuje osnovnu interakciju. Učenje kroz igru: npr. kviz sa dugmadima koja reaguju na klik.	
C.IV.2. Povezuje klijentsku i serversku logiku kroz HTTP zahtjeve i rukovanje podacima.	• Objašnjava osnovne koncepte HTTP protokola: zahtjev (request), odgovor (response), metode GET i POST. • Povezuje HTML formu sa PHP skriptom koja obrađuje podatke. • Primjenjuje $_GET i $_POST za preuzimanje podataka sa klijentske strane.

	• Izvodi prikaz rezultata obrade podataka iz PHP-a unutar HTML stranice. • Provodi jednostavnu dvosmjernu komunikaciju: unos → obrada → prikaz rezultata. • Rješava greške u komunikaciji između klijenta i servera.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

HTTP protokol: zahtjev i odgovor.
Metode GET i POST – razlike i upotreba.
Prijenos podataka između klijenta i servera.
PHP superglobali $_GET, $_POST, $_REQUEST.
Prikaz rezultata iz serverske obrade u HTML-u.
Osnovna obrada grešaka i validacija zahtjeva.

Preporuke za ostvarenje ishoda

Ilustracija toka podataka: grafički prikaz – korisnik → forma → server → baza → server → korisnik. Učenici najprije na tabli skiciraju put podataka — korisnik popunjava HTML formu, preglednik šalje HTTP zahtjev (GET ili POST), server ga prima i PHP skripta obrađuje ulaz, zatim vraća odgovor koji se ugrađuje u HTML — pa isti tok odmah reprodukuju u učionici
Praktičan zadatak: forma za izračunavanje vrijednosti (npr. kalkulator, konverzija valuta) uz serversku obradu.
Rad u parovima: jedan učenik kreira HTML formu, drugi PHP skriptu za obradu, zatim zamjene uloge.
Zadaci sa praćenjem protoka podataka: učenici koriste var_dump() i echo da razumiju gdje se šta izvršava.
Upotreba alata kao što su "Network" tab u browser developer alatima: učenje kroz vizualni prikaz HTTP zahtjeva.
Greške se namjerno koriste kao dio učenja: nastavnik kratko demonstrira tipične promašaje — pogrešan method, neusklađen name, netačan action — i traži od učenika da pomoću echo/var_dump() i Network prikaza lociraju uzrok i predlože ispravku

C.IV.3. Kreira i prezentira funkcionalnu web aplikaciju integrisanjem različitih tehnologija.	• Organizuje strukturu aplikacije (funkcionalnosti, raspored stranica, povezanost elemenata). • Kombinuje HTML za strukturu, CSS za stilizaciju, JavaScript za interakciju i PHP za serversku logiku. • Povezuje aplikaciju sa MySQL bazom podataka za pohranu i prikaz podataka. • Primjenjuje osnovne principe sigurnosti u radu s korisničkim podacima. • Rješava funkcionalne i logičke greške na osnovu provjere aplikacije. • Prikazuje aplikaciju pred razredom uz objašnjenje korištenih tehnologija, logike rada i izazova tokom izrade.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Planiranje projekta i strukture direktorijuma.
Integracija HTML, CSS, JS i PHP koda.
Rad sa bazom podataka – dinamičko prikazivanje i unos podataka.
Sigurnosne mjere (validacija, sanitizacija, rad sa sesijama – osnovni nivo).
Testiranje i otklanjanje grešaka.
Dokumentacija i prezentacija projekta.

Preporuke za ostvarenje ishoda

Rad na ovom ishodu najbolje teče kao mali projekat „od ideje do demonstracije". Nastavnik s razredom najprije skicira temu i tok korištenja (npr. evidencija knjiga, mini-blog ili anketa), pravi jednostavnu mapu stranica i ER-dijagram, a zatim postavlja jasnu strukturu direktorija: semantički HTML i style.css za izgled, script.js za interakcije na klijentu i PHP fajlove za serversku logiku, uz izdvojeni db.php za konekciju s MySQL-om. Učenici prvo isporučuju „jezgro" funkcionalnosti: forma koja šalje podatke metodom POST, JavaScript koji pruža trenutnu, pristojnu provjeru unosa, te PHP koji podatke ponovo provjerava, sanira i sprema kroz PDO i pripremljene upite; rezultat se vraća kao čitljiv HTML prikaz (npr. tabela s listom zapisa). Usput se uvode

osnovne mjere sigurnosti: validacija i filtriranje na serveru, htmlspecialchars pri ispisu, pažljivo rukovanje sesijom u scenarijima prijave i, gdje je smisleno, hashiranje lozinki. Rad napreduje iterativno: učenici male cjeline razvijaju i testiraju „od kraja do kraja“ — unos, obrada, pohrana, prikaz — pri čemu uče da jasno odvoje odgovornosti klijenta (doživljaj i brza provjera) i servera (tačnost, trajnost, sigurnost). Rad u grupama ili samostalno: podjela na frontend i backend zadatke, planiranje i praćenje napretka. Kroz naredne korake projekat sazrijeva: dodaju se pretraga i sortiranje podataka, jednostavno uređivanje i brisanje postojećih zapisa i poruke o ishodu operacija koje su razumljive i korisniku i razvojnome timu. Vođenje dnevnika rada: učenici bilježe šta su uradili, s kojim izazovima su se susreli i kako su ih riješili. Korištenje predloženog obrasca za dokumentaciju: naslov, opis projekta, korištene tehnologije, opis funkcionalnosti. Formativna evaluacija: učenici prezentuju projekat kroz kratku demonstraciju i objašnjenje. Refleksija: učenici komentarišu šta bi unaprijedili, kako su sarađivali i koje su vještine razvili.

E/UČENJE I PODUČAVANJE

Učenje i podučavanje predmeta Programiranje u IT gimnaziji osmišljeno je kao dinamičan, savremen i učeniku prilagođen proces koji traje tokom sve četiri godine školovanja. Programiranje se realizuje kroz dva časa sedmično, a preporučuje se izvođenje nastave u blok-časovima kako bi se omogućio dublji rad na zadacima i projektnim aktivnostima. Nastava je usmjerena ka aktivnom učenju i razvoju digitalnih, analitičkih i kreativnih kompetencija koje učenicima omogućavaju da programiranje razumiju kao alat za rješavanje problema i izražavanje ideja.

U svakoj nastavnoj godini učenici postupno usvajaju znanja iz različitih područja programiranja, od osnovnih pojmova i algoritama, preko rada sa strukturama podataka i funkcijama, do objektno orijentisanog programiranja, skriptnih jezika i razvoja web aplikacija. Sadržaji su raspoređeni tako da učenici kroz postepeno složenije zadatke razvijaju algoritamski način razmišljanja, usvajaju logiku i strukturu programskih jezika, te uče kako se programski koncepti primjenjuju u stvarnim situacijama.

Preporučuje se izvođenje nastave u savremeno opremljenom informatičkom kabinetu. Poželjno je da svaki učenik ima pristup vlastitom računaru, dok u minimalnim uslovima najviše dva učenika mogu dijeliti jedan računar. Računari trebaju biti umreženi i imati pristup internetu, te imati instalirana razvojna i izvršna okruženja za C++, C#, JavaScript i PHP. Za C++ se preporučuju Code::Blocks (uz MinGW-w64) ili Visual Studio/VS Code; Dev-C++ se može koristiti isključivo ako je u pitanju aktivno održavana varijanta (Embarcadero Dev-C++). Preporučuje se da raspored računara u učionici omogućava dobru preglednost i kretanje nastavnika među učenicima, npr. postavka u obliku potkovice, što olakšava interakciju, nadzor i podršku u radu.

Uloga nastavnika je da učenike potiče na istraživanje, samostalno učenje i timsku saradnju. Nastavnik podstiče učenike da analiziraju probleme, planiraju rješenja, pišu kod, testiraju i ispravljaju greške, vode računa o dokumentaciji, te se razvijaju u odgovorne i savjesne digitalne stvaraoce. Pri planiranju aktivnosti nastavnik uzima u obzir predznanja, stilove učenja i interesovanja učenika. Zadaci se oblikuju tako da podstiču rješavanje problema, logičko razmišljanje, tehničku preciznost i kreativnu slobodu.

Poseban značaj imaju projektni zadaci, kroz koje učenici prolaze cijeli razvojni ciklus: od identifikacije problema i planiranja rješenja, preko implementacije i testiranja, do refleksije i prezentacije rezultata. Projekti mogu biti individualni ili timski, a uključuju izradu konkretnih programskih rješenja koja imaju svrhu i primjenu. Takav rad razvija kod učenika istraživački duh, osjećaj odgovornosti, sposobnost samoevaluacije i vještine komunikacije. Učenici vode evidenciju o svom radu, analiziraju uspjehe i poteškoće, dijele iskustva sa vršnjacima i uče iz međusobne interakcije.

Nastava se temelji na inkluzivnim principima gdje se uvažavaju individualne razlike, tempo rada prilagođava se mogućnostima učenika, a učenici koji izostaju ili imaju specifične potrebe uključuju se u rad kroz digitalne alate i platforme. Učionica programiranja treba biti prostor koji potiče saradnju, međusobno poštovanje, samostalnost, odgovornost i želju za učenjem.

Programiranje je multidisciplinarno i prirodno se povezuje s drugim nastavnim predmetima. Posebno je usklađeno s matematikom, tehničkom i informatičkom edukacijom, maternjim i stranim jezicima, ali i sa predmetima društvenih nauka kroz analizu korisničkih potreba, interfejs dizajn i komunikaciju sa softverom. Ovakav pristup učenicima pruža dublje razumijevanje tehnoloških procesa i omogućava im da razviju ne samo tehničke, već i životne vještine potrebne za uspjeh u savremenom digitalnom društvu.

F/VREDNOVANJE U PREDMETNOM KURIKULUMU

Vrednovanje u predmetu Programiranje ima ključnu ulogu u podršci učenicima tokom procesa učenja i napredovanja. Budući da ovaj predmet ne podrazumijeva usvajanje činjenica kroz memorisanje, već razvijanje logike, razumijevanje koncepata, praktičnu primjenu znanja i sposobnost rješavanja problema, vrednovanje mora biti kontinuirano, svrhovito i usmjereno na cjelokupni razvoj učeničkih kompetencija.

Učenici, osim što stiču teorijsko znanje, razvijaju i logičko, algoritamsko i kritičko mišljenje, sposobnost saradnje i rješavanja složenijih problema. Vrednovanje zato obuhvata različite pristupe i metode, usklađene s ciljevima učenja i razvojnim potrebama učenika. U predmetu Programiranje posebno je važno vrednovati kako krajnji rezultat, tako i proces rješavanja, uloženi trud, napredak i refleksiju učenika o vlastitom učenju.

Nastavnik prati učenike dok promišljaju, testiraju ideje, otklanjaju greške, sarađuju, dokumentuju rješenja i objašnjavaju svoj način razmišljanja. U tom smislu, uspjeh učenika se ne ogleda samo u ispravnosti koda, već i u njegovom pristupu, dosljednosti, kreativnosti i spremnosti da razumije zašto nešto (ne) radi. Posebnu pažnju treba posvetiti tome da se prepozna napredak, da se učenika vrednuje ne samo po tome gdje je stigao, već i koliko je od početne tačke uznapredovao.

Zato su zadaci u programiranju prilika za učenike da pokažu kako razmišljaju, a ne samo šta znaju. Svaka greška je dio učenja, i vrednovanje to mora uzeti u obzir. Umjesto ocjene koja samo konstatuje stanje, učeniku se daje prilika da svoje rješenje unaprijedi, da ga objasni, uporedi sa drugim pristupima i eventualno sam procijeni njegovu funkcionalnost i efikasnost.

Učenici se vrednuju kroz različite oblike aktivnosti, kao što su:

- usmeno i pisano izražavanje razumijevanja pojmova i koncepata (npr. objašnjenje algoritama, struktura podataka, logike koda),
- praktični zadaci na času ili u obliku domaćih zadataka (kodiranje, pronalaženje i otklanjanje grešaka, rad s nizovima, funkcijama, klasama, bazama podataka...),
- projektni zadaci i timski rad (izrada aplikacija, razvoj jednostavnih web rješenja, rješavanje problema iz stvarnog svijeta),
- vođenje projektne dokumentacije i tehničkog dnevnika,
- samovrednovanje i vršnjačko vrednovanje, gdje učenici razvijaju vještine objektivne procjene, kritičkog mišljenja i odgovornosti.

Ovakav pristup obuhvata tri domene učenja:

- kognitivnu - razumijevanje i tumačenje koncepata, izraza i struktura;
- psihomotornu - sposobnost praktične primjene znanja pri pisanju, testiranju i doradi koda;
- afektivnu - stavove učenika prema programiranju, motivaciju, saradnju i samostalnost.

Poseban značaj pridaje se funkcionalnom vrednovanju, gdje se ocjenjuje koliko učenik može samostalno osmisliti, implementirati i testirati programsko rješenje u realnim ili simuliranim situacijama, uz pravilnu primjenu standarda i dobre prakse programiranja.

U tom procesu važnu ulogu ima i učenička refleksija o sopstvenom radu, mogućnost da komentariše tuđa rješenja, da predloži poboljšanja i da prepozna dobre strane svog rada, što doprinosi vrednovanju koje razvija odgovornost i samostalnost. Nastavnik u tom procesu ne donosi zaključke sam, nego uključuje učenike u diskusiju o kvaliteti rješenja.

Ocjenjivanje se temelji na jasno postavljenim kriterijima koji su učenicima poznati i razumljivi. Ti kriteriji ne mjere samo ispravnost koda, već i njegovu čitljivost, logičku strukturu, primjerenost rješenju, dokumentaciju i korisničko iskustvo. Uzimaju se u obzir i elementi koji nisu odmah vidljivi, kao što je napor uložen u traženje rješenja, upornost u rješavanju izazova, prepoznavanje problema i predlaganje alternativnih pristupa.

Na kraju, vrednovanje u programiranju nije nešto što se dešava samo na kraju nastavne cjeline ili polugodišta. To je stalni dijalog između učenika i nastavnika, kroz koji se učeniku pomaže da razumije gdje se nalazi, kako napreduje i kako da svoje znanje dalje razvija.

Sve aktivnosti vrednovanja usklađene su sa Standardima učeničkih postignuća - Agencije za predškolsko, osnovno i srednje obrazovanje - APOSO [1], koji obezbjeđuju dosljednost, transparentnost i objektivnost u ocjenjivanju znanja, vještina i stavova učenika.

Instrumenti (formativno i sumativno):

Formativno

- Liste provjere (checkliste): binarne provjere ključnih elemenata (ulaz/izlaz, izbor strukture, 2+ test-slučaja).
- Rubrike s opisnim nivoima: kratka kvalitativna povratna informacija (npr. 1–4) za jasnoću algoritma, stil koda, testiranje.
- Exit ticket / minutni papir: 1–2 pitanja na kraju časa (npr. “navedi rubni test za svoj algoritam”).
- Peer-review: razmjena pseudokoda/koda i popuna kratke rubrike.
- Digitalni portfolio: dijagrami toka, pseudokod, verzije koda, test dnevnik + refleksija.

Sumativno

- Pisani test: kratki zadaci (dopuna pseudokoda, “trag izvršavanja”, odabir ispravnog izlaza).
- Praktična provjera: izrada malog programa uz zadate zahtjeve i test-slučajeve.
- Mini-projekat: mali tim (2–3 učenika) — specifikacija, implementacija, demonstracija, kratki izvještaj.

Primjeri tipova zadataka

Formativno (u toku učenja)

- Preoblikuj dijagram toka → pseudokod (3–6 koraka) i zapiši ulaz/izlaz + 2 rubna testa.
- Ulaz/izlaz (C++): napiši tri cout linije s setw i \n (bez matematike: kartica kontakta, postavke korisnika).
- Grananje/petlje: sentinel unos do “KRAJ”; dodaj provjeru praznog unosa i ispis broja stavki.

Sumativno (na kraju cjeline)

[1] *Standardi učeničkih postignuća agencije za predškolsko, osnovno i srednje obrazovanje – APOSO*, [https://aposo.gov.ba/sadrzaj/uploads/SUP-teh-i-IT-BOS-FINAL.pdf]

- Kombinovani test (30–40 min): dopuni pseudokod → prati izvršavanje petlje → poveži OOP pojmove → kratki zadatak koda.
- Praktična provjera (45 min): program koji validira unos, ispisuje izvještaj (min/max/prosjek) i prilaže 3 test-slučaja.

Minimalni i optimalni standardi

Kriterij 1 – Funkcionalnost i ispravnost (40%)
- *Minimalni:* radi za tipične ulaze; rubni slučajevi djelimično pokriveni.
- *Optimalni:* robustno za tipične i rubne ulaze; obrađuje greške unosa.

Kriterij 2 – Algoritamska jasnoća (20%)
- *Minimalni:* postoji dijagram ili pseudokod koji prati osnovne korake.
- *Optimalni:* jasni dijagram i pseudokod, usklađeni, s označenim ulazima/izlazima i testovima.

Kriterij 3 – Kvalitet koda (20%)
- *Minimalni:* razumljiva imena, osnovno uvlačenje, poneki komentar.
- *Optimalni:* dosljedan stil, jasna struktura, komentari gdje je potrebno.

Kriterij 4 – Testiranje i refleksija (20%)
- *Minimalni:* najmanje 2 testa (tipičan + rubni) s očekivanim izlazima.
- *Optimalni:* skup testova (tipični, rubni, nevažeći) + kratka refleksija učenika o greškama i poboljšanjima.

Kriterij	Minimalni standard	Optimalni standard
Funkcionalnost	radi za tipične ulaze	robustno pokriva rubne + greške unosa
Algoritam	dijagram ili pseudokod	jasni dijagram i pseudokod, usklađeni
Kôd	osnovna čitljivost	dosljedan stil, smisleni nazivi, komentari
Test & refleksija	2 testa (tipičan+rubni)	više testova + kratka refleksija

Lista provjere (Da/Ne) – praktični zadatak

1. Jasno zabilježeni ulaz/izlaz i pretpostavke.
2. Priložen dijagram toka ili pseudokod prije koda.
3. Odabrana odgovarajuća struktura (slijed/grananje/ponavljanje).
4. Validiran korisnički unos (prazan/nevažeći).
5. Priložena najmanje 2 testa s očekivanim izlazom.
6. Kôd se prevodi bez grešaka i radi traženu funkcionalnost.
7. Poštovan osnovni stil (uvlačenje, nazivi, komentari).

Digitalni alati za vrednovanje

- Brze provjere znanja: MS Forms / Google Forms / Moodle Quiz / Kahoot / Quizizz.
- Zadaci s kodom i predaja: Replit / GitHub Classroom / IDE u školi (verzioniranje, povratne informacije).
- Portfoliji i evidencija: OneDrive/SharePoint ili Google Classroom/Drive.
- Praćenje procesa: kratke refleksije učenika i dnevnik rada uz svaku iteraciju rješenja.

Rubrika 1 — OOP mini-projekat (C#)

Kriterij (težina)	Minimalni standard	Optimalni standard
Funkcionalnost i ispravnost (30%)	Aplikacija pokriva osnovne zahtjeve; radi za tipične ulaze.	Pokriva tipične **i** rubne slučajeve; stabilna u radu; nema kritičnih grešaka.
Dizajn klasa i enkapsulacija (25%)	Postoje klase s atributima i metodama; osnovna enkapsulacija.	Jasne odgovornosti klasa; dosljedna enkapsulacija; smisleni pristupni modifikatori; konstruktori smisleno inicijalizuju stanje.
Interakcija objekata (15%)	Osnovne veze između objekata; jednostavne sekvence poziva.	Dobro definisane veze i protok podataka; izbjegnuta kružna zavisnost; prikladni obrasci (npr. agregacija/kompozicija gdje ima smisla).
Kvalitet koda i stil (15%)	Čitljiv kôd s osnovnim uvlačenjem i imenima.	Dosljedan stil, jasna imena i organizacija datoteka; komentari samo gdje su potrebni; nema dupliranja koda.
Testiranje i dokumentacija (15%)	Priložena 2 testa (tipičan + rubni) i kratka napomena o radu.	Sistematični testovi (tipični, rubni, nevažeći); kratka dokumentacija (upute za pokretanje, poznata ograničenja) + 3–5 rečenica refleksije.

Kratke napomene za primjenu:

- Uz zadatak objaviti rubriku i težine unaprijed.
- Po mogućnosti koristiti digitalne alate za predaju i povratne informacije (npr. GitHub Classroom / Classroom / Forms).

G/PROFIL I STRUČNA SPREMA NASTAVNIKA

- Nastavu programiranja na osnovu člana 26. i 81. Zakona o srednjoj školi („Službene novine SBK“, broj 11/01 i 17/04) i Dopune nastavnih planova i programa za srednje škole broj: 01-38-953/09 mogu izvoditi lica sa završenim II (Drugim) ciklusom odgovarajućeg studija visokog obrazovanja (diplomski studij), sa akademskom titulom i stručnim zvanjem magistra za određenu obast kojim stiče 300 ECTS bodova.
- Nastavu predmeta Programiranje mogu izvoditi lica koja su završila odgovarajući fakultet na kome se stiče zvanje:
  - profesor informatike,
  - diplomirani informatičar,
  - diplomirani inženjer informatike,
  - master softverskog inženjerstva
  - profesor matematike i informatike ili drugog dvopredmetnog fakulteta u kojem je informatika ravnopravan predmet,
  - diplomirani inženjer elektrotehnike, smjer elektronika, smjer informatika ili računarstvo,
  - diplomirani inženjer informacijskih tehnologija,
  - Lica ostalih fakulteta koji obrazuju informatički kadar (VII/1; 300 ECTS), a odslušali su 4 semestra informatike (nastavni plan i program mora verifikovati Nastavno naučno vijeće na elektrotehničkom ili drugom srodnom tehničkom fakultetu).
- Ako osoba angažirana za izvođenje nastave u srednjoj školi tokom studija nije položila ispit iz pedagoško-psihološko-didaktičko-metodičke grupe predmeta, dužna je te ispite položiti u roku koji je utvrđen kantonalnim Zakonom o srednjem školstvu.
  Potrebnim pedagoško-psihološkim obrazovanjem nastavnika, stručnih saradnika i saradnika u srednjim školama smatra se pedagoško-psihološko obrazovanje koje obuhvaća obrazovna područja opće pedagogije, didaktike, metodike i psihologije odgoja i obrazovanja.(Dopune nastavnih planova i programa na bosanskome jeziku za srednje škole u SBK, broj:01-38-1525/12-2)
- Ukoliko lice u toku studija nije polagalo ispit iz pedagoško-psihološko-metodičke grupe predmeta, dužno je ove ispite položiti u roku od godine dana od dana stupanja na posao. (Dopuna nastavnih planova i programa za srednje škole, broj: 01-38-953/09)
- Profesori koji nisu navedeni u profilima, a stekli su pravo na izvođenje nastave po prethodnim propisima, zadržavaju stečeno pravo pod uvjetom da su zaposleni na neodređeno vrijeme. (Izmjena i dopuna nastavnih planova i programa za devetogodišnje osnovne škole koje nastavu realiziraju na bosanskom jeziku u Srednjobosanskom kantonu, broj: 01-34-76/2021)