

1. ОСТВАРИВАЊЕ ОБРАЗОВНО-ВАСПИТНОГ РАДА – ОБЛИЦИ И ТРАЈАЊЕ

РАЗРЕД	НАСТАВА				УКУПНО
	Теоријска настава	Вежбе	Практична настава	Настава у блоку	
I	-	70	-	30	100
II	-	70	-	30	100
III	-	102	-	30	132
IV	-	93	-	30	123

Напомена: у табели је приказан годишњи фонд часова за сваки облик рада

2. ЦИЉЕВИ УЧЕЊА

- Развијање способности за декомпозицију проблема на елементарне кораке;
- Развијање логичког закључивања и критичког мишљења;
- Оспособљавање за писање структурираних програма;
- Обучавање за анализу алгоритама и програма ради отклањања формалних и логичких грешака;
- Оспособљавање за савладавање модуларног приступа решавању проблема;
- Обучавање за примену напредних техника у раду са једнодимензионалним и дводимензионалним пољима;
- Упознавање са основним техникама у раду са показивачима;
- Оспособљавање за израду програма у којима се изводе операције над сложеним типовима података (стрингови, структуре, текстуалне и бинарне датотеке);
- Развијање способности за рад са динамичким структурама података;
- Оспособљавање за писање програма заснованих на објектно-оријентисаном концепту;
- Оспособљавање за писање корисничких класа и њихову имплементацију у пројекту;
- Оспособљавање за креирање програма са графичким корисничким интерфејсом;
- Оспособљавање за креирање шаблона, разумевање и коришћење стандарних шаблона у раду са низом, мапом и листом;
- Упознавање са коришћењем основних елемената графике;
- Оспособљавање за писање апликација које користе базе података.

3. НАЗИВ И ПРЕПОРУЧЕНО ТРАЈАЊЕ ТЕМА ПРЕДМЕТА

Разред: први

Ред. бр	НАЗИВ ТЕМЕ	Препоручено трајање теме (часови)			
		Т	В	ПН	Б
1	Алгоритамско решавање проблема	-	30	-	6
2	Структура језика и типови података	-	6	-	3
3	Изрази и наредбе	-	6	-	3
4	Разгранате програмске структуре	-	10	-	6
5	Цикличне програмске структуре	-	10	-	6
6	Једнодимензионални низ – вектор	-	8	-	6

Разред: други

Ред. бр	НАЗИВ ТЕМЕ	Препоручено трајање теме (часови)			
		Т	В	ПН	Б
1	Једнодимензионални низ – вектор	-	6	-	3
2	Показивачи	-	6	-	3
3	Вишедимензионални низ	-	8	-	3
4	Функције	-	16	-	6
5	Стрингови и текстуалне датотеке	-	14	-	6
6	Структуре и бинарне датотеке	-	14	-	6
7	Динамичке структуре података	-	6	-	3

Разред: трећи

Ред. бр	НАЗИВ ТЕМЕ	Препоручено трајање теме (часови)			
		Т	В	ПН	Б
1	Увод у објектни језик и основни концепти ООП	-	15	-	-
2	Класе	-	15	-	6
3	Руковање изузецима	-	9	-	-
4	Наслеђивање и полиморфизам	-	15	-	6
5	Увод у визуелно програмирање	-	6	-	-
6	Контроле графичког корисничког интерфејса	-	42	-	18

Разред: четврти

Ред. бр	НАЗИВ ТЕМЕ	Препоручено трајање теме (часови)			
		Т	В	ПН	Б
1	Компоненте графичког корисничког интерфејса	-	9	-	3
2	Рад са фајловима и фолдерима	-	12	-	3
3	Рачунарска графика	-	15	-	6
4	Колекције и генеричке класе	-	18	-	6
5	Рад са базама података	-	39	-	12

4. НАЗИВИ ТЕМА, ИСХОДИ УЧЕЊА, ПРЕПОРУЧЕНИ САДРЖАЈИ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА

Разред: први

НАЗИВ ТЕМЕ: Алгоритмско решавање проблема	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни појам алгоритма и његову намену; – разликује основне алгоритамске структуре; – објасни појам променљиве; – објасни разлику променљиве и константе; – одреди потребне променљиве (улазне, излазне, улазно-излазне); – креира и тестира линијске, разгранте и цикличне алгоритме; – уочи места гранања и форумише услове гранања; – уочи основне елементе циклуса (припрема, тело, критеријум изласка); – одреди природу циклуса: бројачки или условни (са постусловом или предусловом); – комбинује различите алгоритамске структуре.	– Графички запис алогритама. – Алгоритми са простом линијском структуром. – Алгоритми са разгранатом структуром. – Алгоритми са цикличном структуром. – Алгоритмаи са угњежденим разгранатим и цикличним структурама – Провера исправности алгоритама. Кључни појмови: алгоритам, променљива, константа, линијска структура, разграната структура, вишеструко гранање, цикличне структуре.
НАЗИВ ТЕМЕ: Структура језика и типови података	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни концепт основних типова података; – примењује правила именовања променљивих и константи – разликује кључне речи језика од идетификатора променљивих и константи; – уочава променљиве и константе у програму; – објасни разлику симболичких и вредносних константи; – декларише променљиве и константе у програму; – објасни основну структуру програма; – пише програмски код за унос и приказ података у задатом формату.	– Структура програма. – Кључне речи, константе, променљиве и њихови идентификатори. – Основни типови података. – Декларација и иницијализација променљивих. – Унос и приказ података. – Наредбе и функције за унос и приказ података. – Елементи развојног окружења. – Рад са развојним окружењем (едитор, преводилац, дебагер) демонстриран кроз примере једноставних програма. Кључни појмови: декларација, иницијализација, променљива, константа, тип података, наредба улаза, наредба излаза.
НАЗИВ ТЕМЕ: Изрази и наредбе	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– наведе класификацију и приоритет основних типова оператора; – одреди вредност датог израза формираног коришћењем математичких, логичких и релацијских оператора; – пише изразе и наредбе користећи операторе; – објасни разлику наредбе доделе вредности променљивој и наредбе поређења једнакости вредности променљиве са вредношћу друге променљиве тј. израза; – објасни појам стандардне функције; – користи стандардне математичке функције при формирању израза.	– Оператори језика. Првенство оператора. – Оператор доделе вредности. – Аритметички оператори. – Оператори инкрементирања и декрементирања. – Релацијски оператори. – Логички оператори. – Додатни оператори доделе вредности. – Оператори над битовима. – Стандардне функције. Кључни појмови: оператор, приоритет, израз, наредба, додела, поређење.
НАЗИВ ТЕМЕ: Разгранате програмске структуре	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– наведе основне наредбе гранања; – користи наредбу основне селекције; – дефинише израз на основу кога се одређује ток извршавања алгоритма и програма; – примењује угњеждавање наредби основне селекције; – реализује вишеструко гранање каскадном уградњом наредби основне селекције; – користи наредбу скретнице за реализацију вишеструког гранања; – комбинује наредбе основне селекције са наредбом скретнице при решавању проблема.	– Секвенца и селекција. – Наредба основне селекције. – Вишеструка гранања угњеждавањем наредби основне селекције – Вишеструка гранања коришћењем наредбе скретнице. Кључни појмови: основна селекција <i>if-else</i>, каскадна веза <i>if-else-if-else</i>, скретница <i>switch</i>.
НАЗИВ ТЕМЕ: Цикличне програмске структуре	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни разлику бројачких и условних понављања; – наведе основне наредбе за формирање програмских циклуса; – користи наредбе за формирање бројачких циклуса, условних циклуса са предусловом и условних циклуса са постусловом; – користити наредбу за превремени излазак из циклуса скоком на прву наредбу после циклуса; – користи наредбу за скок на крај циклуса; – решава проблеме угњеждавањем наредби понављања.	– Основни циклус са излазом на врху (<i>while</i>). – Генерализовани циклус са излазом на врху (<i>for</i>). – Основни циклус са излазом на дну (<i>do-while</i>). – Превремени излазак из из циклуса (<i>break</i>). – Скок на крај циклуса (<i>continue</i>). – Угњеждавање циклуса Кључни појмови: предуслов (<i>while, for</i>), постуслов (<i>do-while</i>), скокови (<i>continue, break, return</i>).

НАЗИВ ТЕМЕ: Једнодимензионални низ – вектор	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– дефинише низовни тип; – декларише једнодимензионалне низовне променљиве; – објасни везу између индекса и вредности елемента вектора; – примењује технику линеарног претраживања вектора; – одреди максимум и минимум вектора и/или дела вектора; – програмски реализује уређење вектора у задатом редослед.	– Дефиниција и декларација вектора. – Иницијализација вектора. – Приступање елементима вектора. – Линеарно (секвенцијално) претраживање вектора. – Одређивање минималног и максималног вектора. – Сортирање вектора основним алгоритмом сортирања <i>Selection Sort</i>. Кључни појмови: вектор, декларација, иницијализација, линерана претрага, сортирање.

Разред: други

НАЗИВ ТЕМЕ: Једнодимензионалн низ – вектор	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– користи различите алгоритме сортирања вектора; – користи технике линеарне и бинарне претраге вектора; – примењује технике рада са векторима: формирање, ротирање елемената, сажимање и проширивање, спајање и декомпозиција вектора.	– Линеарно и бинарно претраживање вектора. – Алгоритми за сортирање вектора (<i>Insertion sort</i>, <i>Bubble sort</i>). – Циклично померање (ротирање) елемената вектора. – Сажимање и проширивање вектора. – Спајање више вектора у један и обрнуто – раздвајање почетног вектора у више засебних. Кључни појмови: линеарна претрага, бинарна претрага, сортирање.
НАЗИВ ТЕМЕ: Показивачи	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни разлику вредносне и показивачке променљиве; – декларише и иницијализује показивачку променљиву; – приступи садржају меморије преко показивачке променљиве; – приступа елементима вектора преко показивача; – примењује адресну аритметику показивача кроз вектор; – објасни разлику између статичке и динамичке декларације вектора; – користи функције за динамичку алокацију и реалокацију меморије (<i>malloc</i>, <i>calloc</i> и <i>realloc</i>).	– Декларација и иницијализација показивача. – Адресна аритметика. – Вектори и показивачи. – Динамичка алокација и реалокација меморијског порстора. – Генерички показивачи. Кључни појмови: референца/адреса/показивач, алокација меморије, адресна аритметика.
НАЗИВ ТЕМЕ: Вишедимензионални низ	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– разликује једнодимензионални од вишедимензионалног низа; – декларише и иницијализује дводимензионални низ – матрицу; – користи индексе за приступ елементима матрице; – користи показиваче за приступ елементима матрице; – програмски реализује унос и приказ дводимензионалног низа; – програмски реализује формирање матрице по задатим критеријумима; – реализује претрагу елемената у матрици.	– Декларација и иницијализација вишедимензионалног низа. – Формирање матрице, унос и приказ елемената матрице. – Дијагонале матрице. – Проласци кроз матрицу врста по врста и колона по колона. – Декларација матрице у динамичкој зони меморије. – Зупчаста матрица. Кључни појмови: матрица, редови и колоне, правоугаона матрица, квадратна матрица, главна дијагонала, споредна дијагонала, зупчаста матрица.
НАЗИВ ТЕМЕ: Функције	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни сврху декомпозиције проблема на потпрограмске целине; – уочи могућност декомпозиције проблема; – разликује прототип, дефиницију функције и позив функције; – објасни смисао фиктивних и стварних параметара; – објасни разлику глобалних и локалних променљивих; – користи пренос параметара по вредности и по референци; – објасни улогу показивача у бочним ефектима; – користи вектор и матрицу као параметар функције; – креира функције које враћају показивачке променљиве; – објасни појам рекурзивних функција; – формира сопствену библиотеку функција.	– Смисао и особине функција. – Прототип функције и дефинисање тела функције. – Позивање функције. – Предаја параметара по вредности. – Предаја параметара по референци. – Једно- и дводимензионални низови и функције. – Функције које враћају показиваче. – Показивачи на функције. – Рекурзивне функције. – Формирање сопствене библиотеке функција. Кључни појмови: фиктивни параметри, стварни параметри, повратна вредност, локалне променљиве, глобалне променљиве, пренос параметара по вредности, пренос параметара по референци, повратна вредност, бочни ефекти.
НАЗИВ ТЕМЕ: Стрингови и текстуалне датотеке	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни разлику између вектора карактера и стринга; – примењује технике рада са векторима при раду са стрингом; – креира програме за унос, формирање и приказ стринга; – реализује програме за претрагу стрингова; – примењује стандардне функције из библиотека за рад са карактерима (<i>ctype.h</i>) и стринговима (<i>string.h</i>); – примењује адресну аритметику показивача над стрингом; – наведе врсте датотека; – користи текстуалну датотеку за читавање улазних података за обраду; – смешта резултате обраде у излазни текстуални фајл; – користи наредбе форматираног и неформатираног читања и уписа у текстуални фајл; – користи аргументе командне линије за приступ датотекама.	– Декларација и иницијализација стринга. – Унос и испис стринга. – Стрингови и показивачи. – Стандардне функције за рад са карактерима. – Стандардне функције за рад са стринговима. – Дефиниција и подела датотека. – Форматирани и неформатирани пренос знакова из текстуалног фајла. – Форматирани и неформатирани пренос стрингова из текстуалног фајла. – Приступ фајлу из командне линије. Кључни појмови: карактер, стринг, датотека, форматирани УИ, неформатирани УИ.

НАЗИВ ТЕМЕ: Структуре и бинарне датотеке	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– дефинише структурни тип и декларише структурну променљиву; – приступа пољима структуре помоћу оператора . или оператора →; – користи низове структура; – креира функцију која манипулише структурним променљивама; – објасни разлику бинарне и текст датотеке; – користи функције за упис и читање података у бинарну датотеку; – користи функције за позиционирање унутар датотеке; – изводи операције претраживања унутар бинарне датотеке; – реализује операције ажурирања садржаја бинарне датотеке.	– Дефиниција структуре. – Декларација и иницијализација структурне променљиве. – Приступ пољима структуре. – Низови структура. – Структуре и функције. – Бинарне датотеке. – Функције за упис и читање података из бинарне датотеке. – Функције за позиционирање унутар датотеке. Кључни појмови: структура, показивач на структуру, бинарни фајл.
НАЗИВ ТЕМЕ: Динамичке структуре података	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– дефинише једноструко спрегнуту листу; – формира једноструко и двоструко спрегнуту листу; – додаје чвор у листу; – брише чвор из листе.	– Иницијализација једноструко спрегнуте листе. – Додавање чвора у листу. – Брисање чвора из листе. – Стек, ред. – Двоструко спрегнута листа (иницијализација, додавање и брисање чворова). Кључни појмови: једноструко спрегнута листа, стек, ред, двоструко спрегнута листа.

Разред: трећи

НАЗИВ ТЕМЕ: Увод у објектни језик и основни концепти ООП	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– дефинише појмове објектно оријентисаног програмирања: класа, објекат, инстанцирање објекта; – објасни и користи декларацију променљивих на нивоу блока; – користи податке логичког (<i>bool</i>) типа; – декларише низовне променљиве, и објасни разлику између дводимен-зионалног поља и низа низова (<i>jagged array</i>); – користи <i>foreach</i> циклус; – користи листе и методе за рад са листама; – користи податке набројивог и структурног типа; – декларише методе уз пренос параметра по вредности и по референци; – креира и користи методе са истим именом, а другим параметрима; – користи датотеке за читање и упис података.	– Шта је радни оквир (<i>.NET Framework</i>) и организација системских класа и њихових метода у развојном окружењу. – Дефинисање података на нивоу блока. – Дефинисање низовних променљивих и коришћење системских метода за рад са пољима. – Набројиви тип (<i>enum</i>) података. – Структурни типа података (<i>struct</i>). – Смештање података у листу и манипулација листом. – Креирање и позивање метода са преносом параметра по вредности и референци. – Преклапање имена функције (<i>overload</i>). – Упис и читање података из текстуалне датотеке. Кључни појмови: <i>.NET Framework</i>, системске класе, резервисане речи: <i>static</i>, <i>bool</i>, <i>enum</i>, <i>struct</i>, <i>Array</i>, <i>ArrayList</i>, <i>List</i><>, преоптерећивање метода или <i>overload</i>, <i>params</i>.
НАЗИВ ТЕМЕ: Класе	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– дефинише појам класе и објекта; – објасни однос класе и објекта; – објасни разлику класе и структуре; – објасни улогу и начин позива инстанчног и статичког конструктора класе; – креира конструктор са и без параметара; – дефинише појмове <i>boxing</i> и <i>unboxing</i>; – образложи потребу енкапсулације података; – користи класификаторе приступа члановима класе; – објасни разлику између поља (атрибута) у класи и њему придруженог својства (<i>property</i>); – објасни смисао кључне речи <i>this</i>; – дефинише статичке и инстанчне методе класе; – креира и тестира класе у којима је имплементирано преклапање оператора (не код <i>Jave</i>); – пише и тестира програме у којима се демонстрира примена класе; – организује класе у динамичке библиотеке (<i>dll</i>).	– Појам и елементи класе. – Разлика између класе и структуре. – Подразумевани конструктор. – Конструктор са параметрима. – Конструктор копије. – Кључна реч <i>this</i>. – Енкапсулација података. – Гет и Сет методе. – Класификатори приступа. – Статичке и инстанчне методе класе. – Преклапање метода у класи. – Преклапање (<i>overload</i>) оператора у класи. – Појам парцијалне класе. – Динамичке библиотеке класа. Кључни појмови: класа, објекат, конструктор, кључна реч <i>this</i>, својство, статички члан класе, инстанчни члан класе, преклапање метода, преклапање оператора.
НАЗИВ ТЕМЕ: Руковање изузецима	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– наводи типове грешака у програму; – објасни разлику грешке у току извршења програма (<i>runtime</i>) и логичке грешке у примењеном алгоритму решења; – објасни како изузеци утичу на ток извршења кода; – примењује „бацање” изузетака из корисничке класе; – реализује „хватање” и обраду изузетака „бачених” из корисничке класе; – пише и тестира програме у којима се користи руковање изузецима.	– Креирање прилагођених изузетака који одговарају специфичним условима вашег програма. – Пријављивање тј. избацивање изузетака (<i>throw</i>). – Прихватање (обрада) изузетака. – Примењивање <i>try-catch-finally</i> блокова. – Употреба <i>using</i> блока за управљање ресурсима. – Коришћење изузетака за контролу тока програма. Кључни појмови: изузетак, <i>run-time</i> грешка, <i>try-catch-finally</i>, <i>throw</i>, <i>using</i>.

НАЗИВ ТЕМЕ: Наслеђивање и полиморфизам	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни оправданост и потребу за наслеђивањем класа; – објасни поступке генерализације и специјализације при креирању хијерархије класа; – формира конструкторе изведених класа; – објасни разлику између сакривања и преклапања метода родитељске класе; – реализује преклапање (<i>override</i>) метода родитељске класе; – реализује сакривање (<i>new</i>) метода родитељске класе; – објасни разлику између кастовања, ковертовања објекта у стринг и позива методе <i>ToString</i>; – пише и тестира програме у којима се користи наслеђивање; – пише и тестира програме у којима се користи полиморфизам; – објашњава употребу апстрактних класа; – објашњава намену интерфејса; – пише и тестира програме у којима се користе апстрактни типови података.	– Дефиниција изведене класе. – Употреба чланова изведене класе. – Конструктори изведених класа. – Редифинисање и сакривање метода у изведеној класи. – Композиција класе. – Апстрактна класа. – Заједнички чланови класе. – Интерфејси и наслеђивање интерфејса. Кључни појмови: <i>virtual, override, new, abstract class, interface</i>, класификатор приступа <i>protected</i>.
НАЗИВ ТЕМЕ: Увод у визуелно програмирање	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни компоненте окружења за развој визуелних апликација; – формира пројекат у <i>Windows</i> окружењу; – објасни структуру пројекта; – објасни начине додавања ресурса у пројекат; – дода ресурсе и приступи ресурсима пројекта.	– Израда пројекта. – Структура пројекта. – Дизајнер и код. – Прозори развојног окружења. – Додавање ресурса пројекта. – Смисао програмирања вођеног догађајима. Кључни појмови: солуција, дизајнер, кодни фајл, ресурси пројекта.
НАЗИВ ТЕМЕ: Контроле графичког корисничког интерфејса	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– примењује стандарне компоненте из библиотеке компоненти; – формира и позиционира компоненте; – подешава величину и изглед компоненти; – користи својства и методе компоненти; – уочава заједничка својства компоненти; – разуме смисао програмирања вођеног догађајима; – уочава заједничке и специфичне догађаје компоненти; – програмира догађаје миша; – програмира догађаје тастатуре; – примењује динамичко креирање компоненти.	– Својства контрола корисничког интерфејса. – Методе контрола корисничког интерфејса. – Заједничка својства и методе основних контрола. – Придруживање догађаја контролама корисничког интерфејса. – Аргументи догађаја. – Форма. – Лабела. – Дугме. – Оквир за текст (<i>TextBox, MaskedTextBox</i>). – Оквир за слику (<i>PictureBox</i>). – Нумеричко поље (<i>NumericUpDown</i>). – Компонента <i>Timer</i>. – Контерјнерске контроле. – Контроле избора (<i>checkBox, radioButton</i>). – Оквир са листом (<i>ListBox</i>). – Комбиновани оквир са листом (<i>ComboBox</i>). – Оквир са листом за вишеструки избор (<i>checkListBox</i>). – Контроле за рад са датумима. – Динамичко креирање компоненти. – Догађаји миша. – Догађај тастатуре. Кључни појмови: контроле, својства контрола, методе контрола, догађаји.

Разред: четврти

НАЗИВ ТЕМЕ: Компоненте корисничког интерфејса	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– манипулише дијалозима; – реализује прихват и реакцију на одговор у дијалог прозору; – користи <i>ColorDialog</i> за избор боје; – користи <i>FontDialog</i> за одабир фонта; – користи дијалоге за лоцирање, отварање фајла и памћење фајла; – наведе различите типове менија; – уочи разлику између менија и палете команди; – користи дијалоге за отварање и снимање саджаја датотеке; – пише апликације са више форми.	– Рад са дијалозима (<i>ColorDialog, FontDialog</i>). – Рад са фонтовима. – Рад са дијалозима за приступ фајловима и фолдерима (<i>OpenFileDialog, SaveFileDialog, FolderBrowserDialog</i>). – Рад са менијима. – Рад са линијама са алатима. – Рад са више форми и комуникација између форми. – <i>MDI</i> апликације. Кључни појмови: <i>ColorDialog, FontDialog, OpenFileDialog, SaveFileDialog, FolderBrowserDialog</i>, мени, линија са алатима, <i>MDI</i>.

НАЗИВ ТЕМЕ: Рачунарска графика	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни основне концепте рачунарске графике (пиксел, резолуција, боја, координатни систем); – креира битмапу; – користи графички објекат и његове методе за цртање на битмапи; – примењује трансформације графичког објекта при цртању; – користи дијалоге за избор боје и фонта; – реализује једноставну анимацију објеката и интеракцију са корисником кроз миша и тастатуру.	– Класа <i>Graphics</i> за рад са графиком. – Апстрактна класа <i>Image</i> и функционалности класе <i>Bitmap</i>. – Структуре именског простора <i>System.Drawing: Color, Point, PointF, Size, SizeF, Rectangle, RectangleF</i>. – Креирање објеката класе <i>Pen, SolidBrush, TextureBrush, LinearGradientBrush, HatchBrush</i>. – Поддешавање јединице мере димензија графичког објекта. – Цртање праве и изломљене линије, терминатори линије (<i>LineCap</i>). – Цртање криве линије (цртање „слободном руком”). – Цртање правоугаоника и квадрата (оквир и обојена површина). – Цртање елипсе и круга. – Цртање лукова и кружног исечка. – Цртање стринга (банера). – Трансформације графичког објекта: трансляција, ротација, скалирање, ресет графичког објекта. – Чување 2Д слике у меморији у изабраном формату. Кључни појмови: <i>Bitmap, Graphics, Point, Rectangle, Pen, Brushes, TransformTranslate, TransformRotate, ScaleTransform, Draw</i> методе, <i>Fill</i> методе.
НАЗИВ ТЕМЕ: Рад са фајловима и фолдерима	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни појам тока (<i>stream</i>); – објасни разлику у примени класа <i>FileStream</i> и <i>MemoryStream</i>; – реализује читање и упис у бинарни фајл; – приступа текстуалним фајловима ради читања и писања; – креира нови, брише, копира и премешта постојећи фајл са једне локације на другу; – провери постојање фајла на датој локацији; – приступи мета подацима фајла; – прочита садржај директоријума; – манипулише директоријумом.	– Улазни и излазни токови. – Серијализација и десеријализација. – Рад са бинарним фајловима. – Својства и методе класа <i>FileStream</i> и <i>MemoryStream</i>. – Отварање фајла (<i>FileMode, FileAccess</i>). – Читање података из бинарног тока (<i>BinaryReader</i>) и упис у бинарни ток (<i>BinaryWriter</i>). – Рад са фајловима коришћењем својстава и метода класа <i>File</i> и <i>FileInfo</i>. – Рад са фолдерима коришћењем својстава и метода класе <i>Directory</i> и <i>DirectoryInfo</i>. – Својства и методе класе <i>Path</i>. Кључне речи: <i>FileStream, MemoryStream, Directory, Path, File, BinaryReader BinaryWriter</i>.
НАЗИВ ТЕМЕ: Генеричке класе и колекције	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– објасни предности употребе генеричких конструкција; – познаје синтаксу дефинисања генеричких класа; – дефинише генеричке методе у генеричкој и негенеричкој класи; – примењује ограничења на генеричке типове; – наведе основне негенеричке колекције података; – објасни карактеристике и примену различитих врста колекција; – наводи технике за итерацију кроз елементе колекција; – користи ламбда оператор; – реализује операције са листама: додавање, брисање, претрага, сортирање; – објасни принципе приступа <i>FIFO</i> и <i>LIFO</i> код стека и реда; – користи стек и ред у апликацији; – наведе разлике <i>Dictionary</i> и <i>HashSet</i> објеката; – реализује додавање и брисање елемената, претрагу и итерацију кроз елементе <i>Dictionary</i> и <i>HashSet</i>;	– Креирање генеричке класе, генеричка поља и методе. – Инстанцирање објекта генеричке класе. – Генерички метод у негенеричкој класи. – Основне колекције и њихове генеричке верзије. – Итерација кроз елементе колекција. – Ламбда оператор. – Рад са колекцијама података: <i>ArrayList, List, SortedList</i>. – Организација и приступ подацима у колекцијама типа магацин (<i>Stack</i>) и ред (<i>Queue</i>). – Рад са колекцијом типа <i>HashSet</i>. – Мапирање кључева на вредности кроз колекцију типа <i>Dictionary</i>. Кључне речи: <i>Generics, ArrayList, List<T>, Stack, Queue, HashSet, Dictionary</i>, Ламбда оператор.
НАЗИВ ТЕМЕ: Рад са базама података	
ИСХОДИ	ПРЕПОРУЧЕНИ САДРЖАЈ И КЉУЧНИ ПОЈМОВИ САДРЖАЈА
По завршетку теме ученик ће бити у стању да:	
– разликује конекциони и бесконекциони режим приступа подацима у бази; – формира конекциони и командни објекат за комуникацију са базом; – користи класе <i>ADO.NET</i> архитектуре; – позива <i>SQL</i> упите као текст или као <i>stored procedure</i> из базе; – преноси вредносне параметре командном објекту; – преноси вектор података као структурирани параметар командног објекта; – реализује прихват повратне вредности и преузимање вредности излазних параметара из <i>stored procedure</i>; – користи контроле за приказ података прихваћених из базе; – планира и израђује интерфејс за комуникацију са базом; – врши читање и анализу прочитаних података из базе; – врши упис, ажурирање и брисање података у бази.	– Конекциони и бесконекциони приступ бази података. – Конекциони стринг. – Класе <i>Data Provider</i>-а за креирање конекционог и командног објекта. – Пренос параметара командном објекту. – Коришћење <i>DataReader</i> објекта и рад у конектованом окружењу. – Коришћење <i>DataAdapter</i> објекта и рад у дисконектованом окружењу. – Коришћење класа <i>System.Data</i> именског простора <i>ADO.NET</i>-а: <i>DataSet, DataTable, DataRow, DataColumn, DataView</i> објекта. – Учитавање података из базе. – Уписивање података у базу. – Ажурирање података у бази. – Брисање података из базе. – Коришћење контрола за приказ података: <i>DataGridView, ListView, Chart, TreeView</i>. – Прављење интерфејса апликације за повезивање и рад над базом података. Кључни појмови: <i>ADO.NET</i>. Конекција. Команда. <i>Data Provider. DataReader. DataAdapter. DataSet. DataTable</i>.

5. УПУТСТВО ЗА ДИДАКТИЧКО-МЕТОДИЧКО ОСТВАРИВАЊЕ ПРОГРАМА

Дидактичко-методичко упутство намењено је наставницима како би се поједноставио и уједначио процес планирања и организације наставе у свим школама, али и стручним сарадницима, директору и другим лицима задуженим за праћење и вредновање рада школе.

Облици наставе: Настава се реализује кроз вежбе и наставу у блоку.

Место реализације наставе: Часови вежби и наставе у блоку се реализују у кабинетима где сваки ученик има засебан рачунар на коме самостално ради.

Подела одељења на групе: У првом и другом разреду ученици се деле на две групе до 15 ученика, у трећем и четвртном разреду ученици се деле у три групе до 10 ученика.

Помоћни наставник: (трећи и четврти разред) Помаже наставнику/ментору у припреми и разради радних задатака у процесу припреме ученика за полагање практичног дела стручне матуре; врши инсталацију и реинсталацију софтвера у договору са особом задуженом за одржавање информационих система и технологија, све у договору и координацији са предметним наставником/наставницима исте или сличне групе предмета; води рачуна, у сарадњи са предметним наставником, о мерама безбедности и заштите здравља на раду ученика при реализацији реализацији наставе, вежби и насатве у блоку и употреби заштитне опреме.

Препоруке за планирање наставе:

Часове реализовати у блоку од по 2 часа недељно по групи, у првом и другом разреду и у блоку од по 3 часа недељно по групи, у трећем и четвртном разреду. Часови се одржавају у кабинетима где сваки ученик има засебан рачунар на коме ради.

Приликом планирања часа, исходе предвиђене програмом, уколико је потребно разложити на мање и на основу њих планирати активности за конкретан час. Треба имати у виду да се исходи разликују, да се неки могу брже и лакше остварити, док је за неке потребно више времена, активности и рада на различитим садржајима.

При обради нових садржаја треба се ослањати на постојеће искуство и знање ученика и настојати, где год је то могуће, да ученици самостално изводе закључке.

На часовима комбиновати различите методе и облике рада у циљу мотивације ученика да што боље усвоје садржаје и достигну исходе.

Предложени број часова по темама је оквиран, на наставнику је да процени потребан и довољан број часова по темама узимајући у обзир знања и вештине које ученици имају из претходног школовања и животног искуства.

Програм дозвољава слободу избора програмског језика. За реализацију плана наставе и учења програмирања у првом и другом разреду препоручује се програмски језик *C* и актуелно, а уједно и бесплатно развојно окружење: *Microsoft Visual Studio Express Edition*. За реализацију плана у трећем и четвртном разреду препоручује се програмски језик *C#* у истом развојном окружењу (*Microsoft Visual Studio Express Edition*).

Препоруке за остваривање наставе:

Разред: први

На почетку сваке вежбе ученицима дати теоријске основе неопходне за разумевање и извођење вежбе.

У оквиру теме **Алгоритамско решавање проблема**, ученици пре свега треба да овладају декомпозицијом дефинисаног проблема на елементарне кораке решавања. На почетку теме ученике упознати са етапама у изради софтверског решења (од дефинисања проблема, до извршног кода) и инсистирати на значају почетних корака који претходе самом кодирању: анализа проблема, скицирање решења и састављање алгоритма. При томе направити паралелу са поступком решавања задатака из математике или физике (нпр. неки од задатака са завршног испита осмих разреда) где читањем текста задатка прво препознамо шта је дато (улазне вредности), шта се тражи (излазне вредности), а затим и који је редослед корака (међувредности) потребан у процесу добијања решења. Обзиром да појам променљиве може да буде прилично апстрактан ученицима на самом почетку учења, искористи више једноставних примера за вежбање препознавање потребних улазних, излазних и помоћних променљивих (нпр. дат је полупречник основе и висина, наћи површину и запремину ваљка; одредити када се филм завршава ако је дато време почетка филма у облику сат:мин и дужина филма у мин...). У почетку се препоручује избор примера у којима се користе променљиве бројног типа (цели и реални бројеви). За визуелизацију процеса који се дешава у меморији рачунара у моменту декларисања променљивих, пожељно је користити бесплатан програм Flowgorithm који омогућава цртање алгоритма и покретање његовог извршења уз приказ дешавања у меморији рачунара у сваком кораку извршења. Коришћењем овог алата ученицима постаје очигледна разлика декларисане, а неиницијализоване променљиве у односу на ону која је декларисана, а потом јој и додељена вредност. Наведени алат се може користити не само за реализацију и визуелизацију извршавања линијских алгоритама, већ има могућност и креирања разгранатих (*if-else*) и цикличних структура (*for*, *while*, *do-while*). Када су у питању основне управљачке структуре, Flowgorithm (бар за сада) једино нема могућност симулације селекције помоћу скретнице (*switch*).

Теме **Структура језика и типови података** подразумева увођење у кодирање у изабраном програмском језику (препука је језик *C*). У оквиру ове теме, надовезујући се на примере обрађене у претходној теми и у њима поменуте типове података (*int* и *float*), објаснити остале типове података који постоје у језику *C*. Објаснити појмове имплицитне и експлицитне конверзије типова података. Скренути пажњу на доделу променљиве или константе једног типа променљивој другог типа – када је могућа директна додела уз имплицитну конверзију, а када је неопходно извршити експлицитну конверзију пре доделе. Објаснити када неку вредност треба декларисати као променљиву одређеног типа којој се вредност додељује током извршења програма било наредбом улаза или наредбом доделе, а када је нека вредност константа, било симболичка или вредносна (нпр. у наредби доделе $P=r^2\pi$, P је излазна променљива, r улазна променљива, а π је константа која се не уноси, већ се унапред зна њена вредност). У оквиру ове теме већ треба увести синтаксу наредби форматираног уноса и приказа вредности променљивих. Реализовати једноставне кодове уноса различитих типова променљивих и њиховог приказа применом расположивих могућности формирања (одређивање ширине поља за приказ, лево или десно поравнање, допуна водећим нулама, одређивање потребног броја децимала, одабир начина приказа реалног броја...). Објаснити да суштински у језику *C* постоје само бројни типови података. Тестирати тако што се нпр. *char* вредност прикаже кроз *%c*, а затим и *%d* конверзију.

Кроз тему **Изрази и наредбе** објаснити расположиве групе оператора. Код аритметичких оператора посебно обратити пажњу на оператор дељења и зависност резултата дељења од типа дељеника и делioca. Објаснити начин коришћења оператора *%* за одређивање остатка дељења и немогућност његовог коришћења над реалним вредностима. Објаснити операторе инкрементирања и декрементирања и нагласити разлику префиксне и суфиксне нотације. Објаснити релацијске операторе. Објаснити логичке операторе. Размотрити редослед извршавања операција у изразу који комбинује аритметичке, релацијске и/или логичке операторе када приоритет није одређен заградом. Нагласити разлику наредбе доделе вредности и наредбе поређења вредности. Поред основног, објаснити и додатне операторе доделе (*+=*, *-=*,...). За дате секвенце наредби, ученицима поставити задатак да одреде вредности променљивих по извршењу секвенце. У овој фази рада, довољно је поменути да постоје оператори над битовима. Упознати се синтаксом позива стандардних функција библиотеке *math.h* (*pow*, *sqrt*, *abs*, *fabs*...). Овладати писањем комплексних израза који комбинују стандардне математичке функције са претходно обрађеним операторима (нпр. изрази за одређивање решења квадратне једначине)

У оквиру теме **Разгранате програмске структуре** осврнути се на примере разгранатих алгоритама који су објашњени у оквиту прве тематске целине и објаснити начин њихове програмске реализације тј кодирања. Обратити пажњу на коришћење вичастих загра-

да {} за груписање наредби у програмске блокове чиме се блок наредби смешта у одговарајућу грану разгранате структуре. Насупрот кодирања разгранатих алгоритамских структура, инсистирати и на обрнутом процесу – читање готовог кода. Дати примере кодова са различитим позиционираним вичитастим заградама уз захтев да се на основу датог кода скицира одговарајући алгоритам тока извршења. Објаснити синтаксу *switch* наредбе и када се више каскадних *if-else-if-else* наредби може заменити једном *switch* наредбом. Објаснити излазак из *switch* наредбе са акцентом на могућност „пропадања” кроз гране *switch*-а. Програмски реализовати примере са међусобном уградњом наредби основне селекције и селекције уз помоћ скретнице.

На почетку теме **Цикличне програмске структуре** од ученика захтевати да објасне разлику између бројачких и условних петљи и у листи датих формулација препознају бројачку или условну природу циклуса. Објаснити да се у програмском језику *C* сви циклуси свODE на условне циклусе који се извршавају док је услов испуњен. Објаснити синтаксу *while* наредбе. Објаснити синтаксу *for* наредбе уз објашњење како се циклус реализован *for* наредбом може заменити *while* наредбом. На различитим примерима демонстрирати могућности писања *for* наредбе: иницијализација једне и више променљивих на улазу у циклус, коришћење комплексних улова за излазак из циклуса, мењање вредности једне или више променљивих на крају сваког проласка кроз тело циклуса, изостављање једног или више израза у заглављу *for* наредбе *for(;;) ... for(i=0, j=n; ;) ... for(i<j; i++, j--) ...* Објаснити синтаксу *do-while* наредбе. Објаснити коришћење наредби скокова *break*, *continue*, *return*. Као и у претходној теми, инсистирати на читању кода, тј за готове примере кода са угњезденим наредбама понављања и гранања, тражити да ученици алгоритамски представе ток извршења програма.

У оквиру теме **Једнодимензионални низ-вектор** треба поставити основу за даље изучавање сложених типова података у другом разреду. У завршној фази првог разреда треба објаснити креирање низовне променљиве, чему она служи, од каквих се елемената састоји и како се приступа појединим елементима унутар вектора. За приступ елементима вектора користити индексе. Објаснити претраживање вектора линеарном претрагом сукцесивних елемената на примерима одређивања средње вредности свих или појединих елемената који задовољавају задати услов, пребројавање елемената, измена вредности елемената вектора по услову... Објаснити принцип тражења највећег и најмањег елемента у вектору или делу вектора (нпр. *min/max* целог вектора, *min/max* парни елемент...). Објаснити основни алгоритам сортирања поређењем сваког са сваком елементом.

Адекватни примери задатка за све теме могу се наћи на сајту <https://petlja.org/kurs/11231/1/> и <https://petlja.org/biblioteka/r/kursevi/Zbirka>

Разред: други

Тема **Једнодимензионални низ-вектор** наставак је изучавања започетог у претходном разреду. Обновите основне манипулације вектором: декларација, иницијализација, унос и приказ елемената вектора. На примеру вектора случајних бројева, обновити поступак линеарне претраге вектора. Објаснити принцип бинарне претраге вектора и упоредити брзину алгоритма линеарне и бинарне претраге. Обновити принцип функционисања алгоритма сортирања Selection sort и објаснити и демонстрирати сортирање методом Insertion sort и Bubble sort. Направити поређење перформанси алгоритма сортирања у различитим сценаријима. У текућој фази учења ученици још нису упознати са појмом функције и рекурзивног позива функције, тако да се не могу помињати алгоритми типа Merge и Quick sort (пожељно је обрадити ове алгоритме касније у оквиру теме Функције), али је препорука демонстрирати Merge два већ сортирана вектора у нови сортирани вектор. Препоручени су следећи примери трансформација вектора: циклично померање елемената вектора улево и у десно, уметање новог елемента на задату позицију, уметање новог елемента у уређени вектор тако да се сачува његова уређеност, избацивање елемента са задате позиције, избацивање свих елемената који задовољавају дати услов уз очување релативног поретка преосталих елемената, спајање елемената два вектора у резултујући вектор по задатом правилу, одвајање елемената почетног вектора у два или више излазних вектора (нпр. одвојити парне и непарне, веће и мање од просечне вредности...). Као напредне примере реализовати избацивање елемената који се понављају, генерисање вектора случајних бројева без понављања...

На почетку теме **Показивачи** направити теоријски увод кроз који треба објаснити разлику између адресе и податка који се налази на датом адреси. Објаснити декларацију показивачке променљиве и коришћење оператора * и &. Објаснити извођење аритметичких операција над показивачима (инкремент, декремент, одузимање два показивача) и физички смисао података који се добијају као резултат извршења ових операција. Разумевање извођења операција адресне аритметике проверити тако што се за дату секвенцу наредби тражи процена стања променљивих по извршењу секвенце. Указати на смисао идентификатора вектора као адресе почетног елемента у вектору. Демонстрирати пролаз кроз вектор и приступ елементима коришћењем показивача на елементе вектора. Формирати вектор у динамичкој зони меморије. Објаснити наредбе за резервацију и реалокацију меморијског простора. Објаснити разлику између статички и динамички декларисаног вектора. Објаснити важност ослобађања алоциране меморије употребом функције free. Разумевање природе и функционисања показивача проверити како имплементацијом кодова који захтевају рад са показивачима, тако и читањем и тумачењем готових кодова који користе показиваче.

У оквиру теме **Вишедимензионални низ** објаснити поступак декларације и иницијализације матрице. На почетку реализовати програме за формирање матрице уносом елемената са тастатуре, формирање матрице попуњавањем елемената на основу утврђене законитости и приказ елемената формиране матрице уз одговарајуће форматирање приказа по редовима и колонама. Објаснити приступ елементима матрице коришћењем индекса. Нагласити разлику између правоугаоне и квадратне матрице. Наводити ученике у процесу извођења услова за приступ елементима на, изнад и испод главне и споредне дијагонале квадратне матрице. Препоручује се израда примера који комбинују коришћење матрице и вектора (нпр. формирати вектор који садржи просек/min/max... по колонама/врстама матрице, улазни вектор преписати у прву врсту матрице, а остале врсте формирати тако да свака наредна буде циклично померена у односу на претходну...). Као пример задатка напредног нивоа препоручује се реализација спиралног обиласка матрице. Објаснити начин реализације матрице у динамичкој зони меморије. Објаснити креирање и могуће примене зупчасте матрице.

Са порастом сложености примера, значај декомпозиције проблема на функционалне целине све интензивније долази до изражаја. У том контексту, тема **Функције** има централно место и захтева посебну пажњу. На примерима раније реализованих алгоритма и кодова указати на целине које се понављају у једном или у више различитих програма. Почетна објашњења декларације и позива функција ослонити на примере који су ученицима познати из математике, физике или електротехнике (функција која одређује факторијел природног броја, функција која одређује површину или запремину тела на основу задатих димензија, функција која одређује еквивалентну отпорност два редно везана отпорника, функција која налази еквивалентну отпорност редно везаних отпорника...). Објаснити синтаксу декларације функције, пре свега повратни тип функције и листу фиктивних параметара. Објаснити значење void повратне вредности. Објаснити позивање функције при чему се врши замена листе фиктивних, листом стварних аргумената. Нагласити разлику у начину позива функције која има void повратну вредност и функције чија повратна вредност је променљива неког од раније објашњених простих типова података. Разумевање начина позива функције проверити тако што се за наведени прототип функције, тражи од ученика да у листи понуђених позива препознају оне који су коректно извршени. Препоручује се и обрнути процес, на основу датог позива функције ученик треба да препозна одговарајући прототип у листи понуђених. Објаснити смисао преноса параметара по вредности и зашто се промене

које се дешавају унутар функције на параметрима пренетим по вредности, не одражавају на вредности стварних параметара по повратку у позивајућу функцију. Објаснити бочне ефекте и пренос параметара по референци. Реализовати примере функција које као фиктивне параметре користе вектор и сагледати промене у пренетом вектору по изласку из функције. Реализовати примере функција које као повратну вредност имају показивач. Сагледати ограничења при преносу матрице као фиктивног аргумента функције. Објаснити појам досега вредности променљиве у зависности од тога где је променљива декларисана: на локалном или глобалном нивоу. Објаснити креирање корисничке библиотеке функција на примеру функција за рад са векторима и матрицама: функције за унос и приказ вектора и матрице, функција која налази највећи (најмањи) елемент вектора и матрице, функција која од задатог вектора формира матрицу, функција која од матрице формира вектор преписивањем задате колоне или врсте, функција за транспонување матрице, функције за сортирање вектора...). Објаснити појам рекурзије на примеру функција као што су рекурзивно одређивање факторијела броја, рекурзивно одређивање степена природног броја. Обновити бинарну претрагу вектора и написати рекурзивну функцију за бинарну претрагу која враћа позицију нађеног елемента у вектору пренетом као параметар функције.

На самом почетку теме **Стрингови и текстуалне датотеке** нагласити разлику између обичног вектора карактера и стринга. Објаснити и демонстрирати наредбе за форматирање и неформатирање унос и приказ појединачних карактера. Објаснити наредбе за форматирање и неформатирање унос и приказ стрингова. Реализовати програме уз коришћење функција библиотеке `sture.h` за рад са карактерима. Реализовати програме уз коришћење функција библиотеке `string.h` за рад са стринговима. Креирати библиотеку корисничких функција за рад са стринговима (претварање свих слова у велика/мала, промена кејсинга тако да свака реч текста почиње великим словом, а сва остала су мала (*Capitalize Each Word*), замена првог појављивања задатог карактера другим карактером, замена свих појављивања задатог карактера другим карактером, замена првог појављивања задатог подстринга другим подстрингом, замена свих појављивања задатог подстринга другим подстрингом, избацивање задатог подстринга, шифровање и дешифровање стринга по задатом правилу, избацивање вишеструких празнина у стрингу...). Објаснити поделу датотека и поступак отварања, читања, уписа и додавања садржаја на крај постојеће датотеке. Нагласити да ажурирање произвољног садржаја у текстуалном фајлу није могуће, већ се измена привидно постиже тако што се стари фајл обрише, па се под истим именом меморише нови фајл са изменама. Реализовати пример програма који податке за тестирање повлачи из текстуалног фајла, а резултате извршења уписује у излазни `txt` фајл.

У оквиру теме **Структуре и бинарне датотеке** обновити разлику у понашању вредносних и референцијских типова података и пре свега њиховог понашања када се преносе функцији као њени параметри. Декларисати различите структурне типове који у себи садрже како просте, тако и сложене типове података – вектор, матрицу, стринг, другу структуру или показивач на другу структуру. Објаснити приступ пољима структуре у зависности да ли има се приступа преко структурне променљиве или преко показивача на структурну променљиву. Генерисати функције које као улазне параметре примају структурну променљиву и на практичном примеру показати да, иако су сложеног типа, структурне променљиве се понашају као вредносни типови. Тестирати разумевање тумачењем кодова и одређивањем вредности поља структурне променљиве пре и после позива функције чији је она параметар, при чему у датим примерима кодова комбиновати различите начине преноса структурних променљивих функцији. Обновити разлику између текстуалних и бинарних фајлова и начин приступа бинарном фајлу ради читања и уписа. Реализовати примере читања и уписа у бинарни фајл једног по једног податка, као и целог вектора података одједном. Објаснити наредбе за позиционирање унутар бинарног фајла, као и поступак ажурирања података у постојећем бинарном фајлу. Демонстрирати примере читања из бинарног и уписа у текстуални фајл и обрнуто.

На почетку теме **Динамичке структуре података** обновити разлику између поља (вектора и матрица) декларисаних и статичкој и у динамичкој зони меморије. Размотрити предности и мане динамичких структура података. Подсетити се знања стечених кроз тему Показивачи и алокације и делокације меморије функцијама `malloc()`, `calloc()`, `realloc()` и `free()`. Објаснити начин креирања и програмски имплементирати креирање динамичких листа (једноструко и двоструко спрегнутих). Објаснити поступке додавања и избацивања чворова из листе. Објаснити начин приступа подацима у стеку и реду. Имплементирати стек и ред помоћу низова. Имплементирати стек и ред помоћу динамичких листа.

Адекватни примери задатка за све теме могу се наћи на сајту <https://petlja.org/kurs/11234/1/>

Разред: трећи

Кроз почетну тему **Увод у објектни језик и основни концепти ООП** ученике упознати на појмовном нивоу са значењем термина класа и објекат и односом класа-објекат класе. Објаснити како је позивају методе у објектно оријентисаном окружењу (позив из објекта и позив из класе) у обиму који је потребан да би се у наставку теме користиле системске класе и у њима дефинисане статичке и инстанчне методе методе. У наставку теме упознати ученике са синтаксом `C#` језика кроз примере сличне оним какви су рађени у првом и другом разреду. Размотрити сличности и разлике у синтакси језика `C` и `C#`. Одрадити примере са применом основних управљачких структура (`if`, `switch`, `for`, `while`, `do-while`). Нагласити разлику код наредбе скретнице (нема пропадања кроз `switch` као у `C`-у) и обрадити `foreach` циклус. Објаснити начин креирања вектора и матрица у језику `C#`. Нагласити разлику између 2Д поља и низа низова (*jagged array*). Посебну пажњу обратити на креирање метода и пренос параметра методама (улазни параметри, излазни `out` параметри, улазно-излазни `ref` параметри). Реализовати примере метода које као повратну вредност и-или параметар користе поља. Објаснити кључну реч `params` и креирати методе са променљивим бројем параметара. Када се говори о преоптерећивању (*overload*) метода, почети од указивања на *overload*-е који су дефинисани за коришћене методе класе *Math*. Потом креирати преоптерећене корисничке методе нпр: метод чија прва верзија ради приказ целог вектора, друга верзија методе приказује само елементе између две задате позиције, трећа само првих или последњих *N* елемената... Увести појмове листе *ArrayList* и *List<T>*, без улажења у објашњење смисла генеричких листа, већ их само размотрити као листе које садрже елементе истог типа. Упоредити понашање листе и вектора и демонстрирати коришћење метода за рад са пољима и листама.

Тему **Класе** започети објашњењем елемената који чине класу као шаблон по коме се креирају објекти. У првој фази увести појмове поље класе, конструктор класе (подразумевани и конструктори са параметрима) и методе класе. Као почетни пример може послужити класа Тачка у којој се могу реализовати методе растојања две тачке и то у инстанчној и статичкој верзији. На овом примеру, блиском из математике, могуће је очигледно објаснити шта треба пренети као аргумент методи када се она реализује као инстанчна метода растојања *this* тачке од неке задате тачке, а шта се преноси када се она реализује као статичка метода растојања неке две задате тачке. Истовремено, демонстрирати начин позива ових метода. У наставку, препорука је реализовати примере класа које моделирају неке ентитете из окружења, нпр: текући рачун са пољима број рачуна, име власника, износ на рачуну, конструктор којим се поставља број и власник рачуна, а износ поставља на нулу, оверлоад конструктора који поставља поред броја и имена власника и иницијални износ на рачуну, методе уплата, исплата, пренос на други рачун или са другог рачуна... У оквиру ове теме потребно је објаснити и преклапање оператора у корисничкој класи. Погодан пример, који би се наслањао на претходно одрађени пример класе тачка, био би класа вектор са пољима почетна и крајња тачка и методама за преклапање оператора `+` и `-` за сабирање и одузимање два вектора. У наставку, увести појам енкапсулације. Објаснити смисао и значење класификатора приступа *private*, *public* и *internal*. Објаснити улогу својства (*property*), његове гет и сет

методе и нагласити разлику између поља класе и њему придруженог својства. Представити начин контроле вредности које се додељују пољу уз помоћ сет методе, нпр. класа наставни предмет, у њој поље оцена и својство којим се кроз сет методу дозвољава само унос вредности од 1 до 5. Уколико се покуша додела неке вредности ван дозвољеног опсега, у овом моменту (још није уведено бацање изузетака), поставити неку подразумевану вредност која би указивала на покушај недозвољене доделе вредности. Објаснити организацију класа унутар именских простора и приступ класи из другог именског простора. Демонстрирати креирање библиотеке класа (*dll*), као и приступ класама дефинисаним унутар *dll*-а. Упоредити понашање структуре и класе, првенствено шта се дешава при декларацији, а затим и шта се дешава при додели једне променљиве другој у зависности од тога да ли су декларисане као структурне променљиве или као објекти класе. Размотрити разлику структуре у односу на класу када су у питању конструктори (у структури конструктори са параметрима не „маскирају” подразумевани конструктор) и својства (поље структуре се не може иницијализовати кроз сетер својства).

На почетку теме **Руковање изузетцима** обновити типове грешака које се могу јавити у програмима, са посебним акцентом на грешке током извршења програма (*run time*). Објаснити потребу обраде насталих грешака. Надовезати се на градиво изложено у оквиру теме Класе о гет и сет методама за контролу приступа пољима класе и демонстрирати бацање изузетака уместо постављања подразумеваних вредности у случају недозвољених додела вредности пољима класе. Објаснити коришћење *try-catch-finally* блокова за хватање и обраду насталих грешака. Објаснити коришћење *using* блока. Инсистирати на томе да ученици самостално формирају тест примере за проверу креираних класа и тачности извршења програма који користе креиране корисничке класе.

Тему **Наслеђивање и полиморфизам** започети навођењем ученика у процесу закључивања о томе које је класе логично наследити из задате „надкласе” (нпр превозна средства). Објаснити принцип генерализације и специјализације у поступку осмишљавања ланца наслеђивања класа. Демонстрирати наслеђивање класа на једноставним примерима (особа-наставник-разредни старешина, запослени-руководилац, рачун-девизни рачун...). Објаснити класификатор приступа *protected*. Објаснити принцип функционисања конструктора изведене класе и у том контексту, значење кључне речи *base*, наспрам раније објашњеног *this*. На примеру ентитета из окружења указати на заједничке функционалности (методе) које се код сродних ентитета (класа) могу мањи или више разликовати (надовезујући се на почетни пример особа-наставник-разредни старешина, указати да и наставник и разредни старешина имају методу која израчунава плату, али се она одређује на другачији начин). Објаснити синтаксу преклапања (*override*) метода родитељске класе. Насупрот преклапању (*override*) објаснити заклањање метода родитељске класе коришћењем кључне речи *new*. На примерима објаснити и демонстрирати како се објекти креирани конструктором изведене класе могу доделити променљивама родитељске класе и у склопу овог, увести оператор *is*. Креирати низ теоријски задатака у којима се креирају објекти класа из датог ланца наслеђивања које у себи садрже редефинисани-заклоњени метод и одредити шта тај метод враћа у зависности из ког се објекта позива. На пример: класе деда, отац и син имају редефинисани или закљоњени метод прикажи који враћа име класе. Ако се декларишу и иницијализују објекти *Deda d1=new Deda()*, *Deda d2 = new Otac()*, *Deda d3 = new Sin()*, *Otac d4=new Sin()*... одредити шта ће метод *Prikazi()* вратити у случају позива из сваког од објеката респективно.

Истаћи системску класу *Object* као имплицитног родитеља како свих системских класа у *.NET* окружењу, тако и кориснички дефинисаних класа. Указати на методе класе *Object* које се наслеђују код свих дефинисаних класа. Објаснити шта враћа метода *ToString()* из класе *Object* и редефинисати методу *ToString()* у креираној корисничкој класи.

Увести појам и објаснити улогу интерфејса. Демонстрирати наслеђивање интерфејса на примерима системских интерфејса *IComparable* и *IComparable* кроз примере који захтевају сортирање и-или поређење објеката дефинисаних корисничких класа. Објаснити намену, понашање и демонстрирати коришћење апстрактних класа. У контексту наслеђивања, објаснити кључну реч *sealed*.

Осмислити примере где се од ученика очекује да креирају корисничку класу на основу већ дефинисане почетне класе. Од ученика се очекује да на основу формулације захтева процене да ли ће нова класа бити наслеђена из почетне или ће нова класа као своје поље садржати објекат (или листу објеката) почетне класе. **Пример 1:** дата је класа роба. Треба креирати класу кварљива роба наслеђену из класе роба. **Пример 2:** дата је класа особа. Креирати класу текући рачун која у себи садржи власника рачуна као поље типа особа. **Пример 3:** дата је класа особа. Из ње треба наследити класе ученик и професор. Креирати класу одељење која садржи разредног старешину као поље типа професор и поље које чува податке о ученицима типа листа или вектор објеката класе ученик.

Садржајима теме **Увод у визуелно програмирање** ученици се уводе у израду апликација са графичким корисничким интерфејсом. На почетку је битно истаћи разлику у начину писања и организацији кода код визуелних апликација у односу на конзолне. Увести појам догађаја и програмирања вођеног догађајима и на примерима указати на организацију кода унутар руковоаца догађајима. При демонстрацији и објашњењу начина коришћења сваке од контрола наведених у исходима, размотрити својства, методе и догађаје придружене посматраној контроли. Посебан акценат ставити на објашњење аргумената руковоаца догађајем: контрола изазивач догађаја *sender* и аргумент догађаја *e*. Указати на чињеницу да када је аргумент догађаја *e* објекат класе изведене из основне класе *EventArgs*, он у себи садржи додатне информације о догађају који се десило (координате клика мишем, притиснути тастер тастатуре...). Насупрот изради дизајна апликације коришћењем дизајнера развојног окружења, одрадiti примере у којима се захтева динамичко креирање контрола (нпр креирање потребног броја оквира за слику у зависности од вредности унете у нумеричко поље), подешавање својстава динамички креираних контрола и додавање руковоаца догађајима из кода.

Када се пређе са конзолног на визуелно програмирање ученици често имају тенденцију коришћења цикличне структуре у захтевима који у својој формулацији имају неку понављање, нпр: кликом на дугме покренути слајд шоу којим се приказује 10 слика. Демонстрирати шта се дешава уколико се унутар руковоаца догађајем кликом на дугме стави петља која се понавља 10 пута и у сваком свом пролазу приказује једну од 10 слика. Са друге стране, реализовати ситуацију где се кликом на дугме укључује тајмер, а потом у сваком откуцају тајмера приказује једна слика. Указати на ситуације када променљиве треба декларисати глобално у класи, а када је оправдана локална декларација унутар метода руковоаца догађајима. После почетног овладавања радом са појединим контролама корисничког интерфејса, реализовати сложеније примере који захтевају имплементацију корисничке класе. Податке из фајла учитати у листу објеката креиране класе, а потом на контролама форме праказати и по захтеву обрадити учитане податке.

Разред: четврти

Тема **Компоненте графичког корисничког интерфејса** наставак је теме из трећег разреда (Увод у визуелно програмирање) кроз коју су обрађене основне контроле графичког корисничког интерфејса. У овој фази рада треба обрадити рад са дијалозима. Пре свега разморити повратну вредност методе за приказ дијалога *DialogResult*. Кренути од приказивања *MessageBox*-а уз коришћење *overlay*-а методе за приказ којим се бирају понуђени тастери (*OK/Cancel, Yes/No, Yes/No/Cancel, Abort/Retry/Ignore*), препознавања избора и програмирања одговарајућег одговора на избор корисника. У наставку обновити својства и методе структуре *Color*, а затим обрадити дијалог за избор боје. Такође, објаснити рад са класом *Font* и начин избора фонта дијалогом *FontDialog*. Избор боје и фонта, после почетних једноставних примера промене боје и фонта текст својства контроле *RichTextBox*, предлог је креирати једноставни текст едитор у коме би се реализовала промена боје и облика фонта селектованог текста, означеног (*highlight*) текста, све осим означеног текста. Такође, реализовати програмски промену стила фонта (*Bold, Underline,Italic, StrickeThrough*). У наставку, на истом примеру је могуће увести рад

са менијима (главни и контекстни мени), као и рад са линијом са алатима, нпр избор облика и величине фонта из падајуће листе линије са алатима, стиловање фонта преко дугмади на линији са алатима, као и одабир боје фонта. Обработити дијалоге за приступ фајловима и фолдерима – меморисати унети форматирани текст под одређеним именом, отворити нови или постојећи фајл, направити промене и запамтити за под новим именом. У оквиру ове теме презентовати и рад са више форми, начин како се подаци унети на једној форми могу прочитати и користити на другој форми исте апликације. Објаснити отварање форме у модалном и немодалном режиму рада и разлику у понашању. Даље проширење протходно предложеног примера једноставног едитора текста ишло би у смеру имплементације функционалности која омогућава кориснику да отвори више докумената, прикаже их унутар главне форме као под-прозор (MDI) и да манипулише њима независно један од другог. Објаснити однос *Parent* и *Child* форме.

Тему **Рад са фајловима и фолдерима** започети обнављањем дефиниција фајла и фолдера. Објаснити концепт путање и како се користи за референцирање фајлова и фолдера. Обновити коришћење класа за читање и упис у текстуалне фајлове (које по препоруци треба обработити у трећем разреду). Објаснити својства, методе и енумерације класа именског простора *System.IO*, пре свих класа *Path* и *Directory*. Креирати нови фолдер на одређеној путањи, објаснити како прегледати садржај одређеног фолдера. Пример: Написати програм који листује све фајлове и подфолдере на одређеној путањи. Објаснити како програмски преименовати и обрисати постојеће фајлове и фолдере, како радити са путањама, комбиновати их и извучити информације из њих.

Обновити разлику између текстуалних и бинарних фајлова. Објаснити рад са бинарним датотекама у језику *C#*. На пример, написати програм који чита из текстуалног и пише у бинарни фајл и обрнуто. Објаснити како радити са стреамовима за читање и писање података.

Кроз тему **Рачунарска графика** потребно је упознати се са основним опцијама коришћења GDI+ (Graphics Device Interface) библиотеке за цртање 2Д графике у *C#*. Подсетити се појма пиксел, растерска графика, векторска графика. Обработити основне структуре и класе имеског простора *System.Drawing*. Пре свега, увести у рад структуре *Point*, *PointF*, *Rectangle* и *RectangleF*. Објаснити класе *Image* и *Bitmap* и нагласити да се у случају класе *Image* ради о апстрактној класи, док је класа *Bitmap* наслеђена из класе *Image*, може инстанцирати и њена инстанца се може доделити објекту класе *Image*. Објаснити коришћење основних елемената као што су *Graphics* објекат, *Pen* и *Brush* за цртање. Када се говори о објекту класе *Graphics* представити га као објекат који врши цртање на површини на којој је креиран и демонстрирати разлику при цртању графичких облика директно на контолама форме и на битмапи. Израдити једноставне апликације које користе GDI+ за цртање различитих облика и линија. Примери на којима се може вежбати исцртавање графичких облика: цртање серије облика различитих боја на сваки откуцај тајмера, цртање квадрата и правоугаоника повлачењем миша од почетног темена у правцу повучене дијагонале, цртање круга и елипсе из темена описаног квадрата/ правоугаоника или из центра повлачењем миша за дужину полупречника, цртање изломљене линије тако што се на сваки клик мишем црта наредни сегмент, а цртање се завршава двокликом, исцртавање банера итд... Обработити основне трансформације попут транслације, ротације и скалирања. Пример: графичко решавање система линеарних једначина – нацртати координатни систем, избеждатири осе координатног система, а по уносу коефицијената прваца и слободних чланова две линеарне једначине, нацртати праве и тачку пресека.

На почетку теме **Колекције и генеричке класе** објаснити концепт генеричких класа у *C#*. Објаснити предности генеричких метода, а потом и имплементирати једноставне генеричке методе: генерички метод за замену вредности две променљиве, генерички метод за налажење максимума и претрагу по траженом елементу у пољу, генерички метод за креирање листе попуњене датом иницијалном вредношћу итд. Објаснити позив генеричких метода. Имплементирати једноставне генеричке класе. Демонстрирати примере ограничења типова у генеричким класама: ограничење типа параметара на вредносне типове, на референцне типове, на типове који имају подразумевани конструктор, на типове који наслеђују родитељску класу, на типове који имплементирају одређени интерфејс, комбинована ограничења. Имплементирати генеричку класу са различитим ограничењима и апликацију која користи генерисану класу. Објаснити рад са колекцијама: *List*, *Dictionary*, *Queue*, *Stack*. Када су у питању генеричке листе *List<T>*, оне су већ коришћене током трећег разреда, а овде треба обновити речено о листама и коришћење метода *Add*, *Remove*, *Find*, *Sort*. Када је реч о речницима (*Dictionary<TKey, TValue>*), кренути од смисла парова кључ-вредност и објаснити претрагу по кључу. У наставку, прво дати примере сценарија у којима би користили податке организоване у стек или ред (*Queue<T>*, *Stack<T>*). Представити напредне типова колекција (*HashSet*, *SortedList*, *SortedListDictionary* итд.)

Тема **Рад са базама података** је централна тема у четвртм разреду на коју се наслана практични задатака на завршном испиту. Стога треба предвидети довољно време за детаљну обраду ове теме. Почетни од објашњења принципа, предности и недостатака конекционог и бесконекционог приступа бази. Објаснити основне провајдерске класе (*Connection*, *Command*, *DataAdapter*, *DataReader*...). Као почетне примере одрадити читање података из табеле базе података у објекат типа *DataSet* и *DataTable* и представити прочитане податке на назначеној контроли интерфејса. Препорука је не писати упите у коду, већ користити ускладиштене процедуре. Реализовати читање података из базе који се извршавају по задатом критеријуму претраге уз пренос улазних параметара ускладиштеној процедури. Реализовати позив командних упита из апликације. При томе реализовати враћање вредности из ускладиштене процедуре (нпр. упит враћа вредност аутоматске шифре која је креирана при упису новог реда у табелу). Реализовати позив процедура које захтевају пренос низа улазних вредности (тј. пренос структурираног параметра). При читању и изменама у бази података, обезбедити апликацију од пуцања коришћењем механизма *try-catch* и-или одговарајућим проверама стања у контролама на форми. После увода у ову тему, примери погодни за вежбање су управо задаци из приручника за полагање матурског испита.

6. УПУТСТВО ЗА ФОРМАТИВНО И СУМАТИВНО ОЦЕЊИВАЊЕ УЧЕНИКА

Разред: први

У процесу вредновања потребно је континуирано пратити рад ученика. Током школске године пратити однос ученика према раду, начину учења и савладавању градива, степен залагања, ниво самосталности. Један од параметара формативног оцењивања може бити и оригиналност идеја и разноврсност предложених решења (тј. различитост у односу на поступке решавања демонстриране на часу). При формативном оцењивању могу се узети у обзир и брзина одговора на захтеве, креативност у раду, ефикасност алгорита, самосталност у процесу уочавања и отклањања грешака у алгоритму и коду.

Препоручује се да коначна оцена сваког ученика буде комбинација различитих оцењивања:

- активност на часу и учествовање у разговору и дискусији,
- израда домаћих задатака,
- тестови теоријских знања и
- тестови практичне израде задатака.

Користити редовне домаће задатке који ће наставнику омогућити да стекне бољи увид у остварености исхода кроз анализу примера који ученици нису знали да ураде, а ученику да прати своје напредовање.

На крају појединих тема препоручује се сумативна провера знања:

– **Алгоритамско решавање проблема** – креирати алгоритме са задатом функционалношћу

– **Структура језика и типови података и Изрази и наредбе** – стечена знања проверити теоријским тестом кратких питања и задатака. Тестови могу бити рађени на папиру или у форми тестова на електронској платформи за учење (у случају коришћења питања отвореног типа, наставник треба ручно да оцени одговор и образложи дату оцену)

– **Разгранате програмске структуре, Цикличне програмске структуре и Једнодимензионални низови** – тестови теоријских знања и тестови практичних вештина: кодирање датог алгоритма, превођење датог кода у алгоритам, комплетно решавање задатка од описа захтева до креирања извршног кода (инсистирати да се задатак реализује прво алгоритамска, а тек по изради алгоритма приступити кодирању). При сумативном оцењивању у фокусу треба да буде тачност алгоритма и кода, а не ефикасност и оптималност (посебно не у почетним фазама учења)

Вредновање активности у оквиру тимског рада може се обавити тако да се од сваког члана тражи објашњење елемената урађеног рада и процена сопственог доприноса у оквиру тима.

За било коју активност којом се вреднује рад, унапред упознати ученике са критеријумима вредновања. На сваком часу подстицати ученике да дају своје мишљење и активно учествују у настави.

Приликом сваког вредновања дати ученику повратну информацију на основу које ће моћи да разуме грешке и побољша свој резултат и учење.

Охрабривати ученике да процењују сопствени напредак у учењу.

Разред: други

У процесу вредновања потребно је континуирано пратити рад ученика.

Током школске године пратити однос ученика према раду, начину учења и савладавању градива. степен залагања. Као један од параметара формативног оцењивања може бити и уредност, разноврсност предложених решења, брзина одговора на захтеве, степен самосталности при изради програма и отклањању грешака.

Препоручује се да коначна оцена сваког ученика буде комбинација различитих оцењивања:

- активност на часу и учествовање у разговору и дискусији,
- израда домаћих задатака,
- тестови теоријских знања и
- тестови практичне израде задатака.

Користити редовне домаће задатке који ће наставнику омогућити да стекне бољи увид у остварености исхода кроз анализу примера који ученици нису знали да ураде, а ученику да прати своје напредовање.

На крају појединих тема препоручује се сумативна провера знања организовањем тестова теоријских знања (кратки теоријски задаци, могу се реализовати у форми електронских тестова) и тестова практичних вештина. Примери захтева за тестове практичних вештина су (по темама):

– **Једнодимензионални низ** – претрага и трансформације вентора померањем, уметањем и избацивањем елемената

– **Показивачи** – реализација вектора у динамичкој зони и приступ вектору преко показивача

– **Вишедимензионални низ** – претрага по врстама и колонама, формирање матрице од вектора и обрнуто, манипулација зупчастом матрицом

– **Функције** – функције са преносом по референци, функције са бочним ефектима, рекурзивне функције

– **Стрингови и текстуалне датотеке** – манипулација стринговима и трансформације стрингова коришћењем функција библиотеке *ctype.h* и *string.h*. Креирање и позивање корисничких функција које за аргумент или као повратну вредност имају стринг (улазне податке за тестирање читати из текстуалног фајла)

– **Структуре и бинарне датотеке** – манипулација структурама, креирање и претрага вектора и матрица структура. Креирање и позивање корисничких функција које за аргумент или као повратну вредност имају структуру или показивач на структуру (улазне податке читати из фајла)

Вредновање активности у оквиру тимског рада може се обавити тако да се од сваког члана тражи објашњење елемената урађеног рада и процена сопственог доприноса у оквиру тима.

За било коју активност којом се вреднује рад, унапред упознати ученике са критеријумима вредновања. На сваком часу подстицати ученике да дају своје мишљење и активно учествују у настави.

Приликом сваког вредновања дати ученику повратну информацију на основу које ће моћи да разуме грешке и побољша свој резултат и учење.

Охрабривати ученике да процењују сопствени напредак у учењу.

Разред: трећи

У процесу вредновања потребно је континуирано пратити рад ученика.

Током школске године пратити однос ученика према раду, начину учења и савладавању градива. степен залагања. Као један од параметара формативног оцењивања може бити и уредност, разноврсност предложених решења, брзина одговора на захтеве, степен самосталности при изради програма и отклањању грешака.

Препоручује се да коначна оцена сваког ученика буде комбинација различитих оцењивања:

- активност на часу и учествовање у разговору и дискусији,
- израда домаћих задатака,
- тестови теоријских знања и
- тестови практичне израде задатака.

Користити редовне домаће задатке који ће наставнику омогућити да стекне бољи увид у остварености исхода кроз анализу примера који ученици нису знали да ураде, а ученику да прати своје напредовање.

На крају појединих тема препоручује се сумативна провера знања организовањем тестова теоријских знања (кратки теоријски задаци, могу се реализовати у форми електронских тестова) и тестова практичних вештина. Примери захтева за тестове практичних вештина су (по темама):

– **Увод у објектни језик и основни концепти ООП** – манипулација пољима и структурама у програмском језику *C#*. Задатке реализовати уз коришћење метода и имплементацију технике преоптерећених метода

– **Класе, Руковање изузетцима** – креирање корисничке класе по датој спецификацији уз имплементацију бацања изузетака из класе. Тестирати методе креиране класе у главном програму. Улазне податке читати из текстуалног фајла.

– **Наслеђивање и полиморфизам** – креирање (минимално) две класе, родитељске и изведене по датој спецификацији. Друга могућност је да се за дату родитељску класу захтева имплементација изведене класе по задатим критеријумима. Креирање класе која ће као поље имати листу објеката друге корисничке класе. Тестирати методе креираних класа у главном програму. Улазне податке читати из текстуалног фајла.

– **Увод у визуелно програмирање** – манипулација основним контролама корисничког интерфејса. У оквиру ове теме, препоручује се израда једног или више пројектних задатака који се могу реализовати индивидуално или у мањим групама. Дизајн и функционалности пројектног задатка могу бити прецизно специфициране. У завршној фази рада, пожељно је да ученици самостално, у складу са самопроценом способности осмисле дизајн и функционалност завршног пројекта.

Вредновање активности у оквиру тимског рада може се обавити тако да се од сваког члана тражи објашњење елемената урађеног рада и процена сопственог доприноса у оквиру тима.

За било коју активност којом се вреднује рад, унапред упознати ученике са критеријумима вредновања. На сваком часу подстицати ученике да дају своје мишљење и активно учествују у настави.

Приликом сваког вредновања дати ученику повратну информацију на основу које ће моћи да разуме грешке и побољша свој резултат и учење.

Охрабривати ученике да процењују сопствени напредак у учењу.

Разред: четврти

У процесу вредновања потребно је континуирано пратити рад ученика.

Током школске године пратити однос ученика према раду, начину учења и савладавању градива, степен залагања. Као један од параметара формативног оцењивања може бити и уредност, разноврсност предложених решења, брзина одговора на захтеве, степен самосталности при изради програма и отклањању грешака.

Препоручује се да коначна оцена сваког ученика буде комбинација различитих оцењивања:

– активност на часу и учествовање у разговору и дискусији,

– израда домаћих задатака,

– тестови теоријских знања и

– тестови практичне израде задатака.

Користити редовне домаће задатке који ће наставнику омогућити да стекне бољи увид у остварености исхода кроз анализу примера који ученици нису знали да ураде, а ученику да прати своје напредовање.

На крају појединих тема препоручује се сумативна провера знања организовањем тестова теоријских знања (кратки теоријски задаци, могу се реализовати у форми електронских тестова) и тестова практичних вештина. Примери захтева за тестове практичних вештина су (по темама):

– **Компоненте графичког корисничког интерфејса** – у оквиру ове теме, препоручује се израда једног пројектног задатка који се може реализовати индивидуално или у мањим групама. Дизајн и функционалности пројектног задатка могу бити прецизно специфициране или задате само основне функционисања, а ученици могу сходно знању и преференцама додати проширење функционалности. При оцењивању узети у обзир личну иницијативу.

– **Рад са фајловима и фолдерима** – читање садржаја бинарних фајлова, преписивање из једног у други фајл. Ажурирање садржаја фајла, читање и преглед садржаја фолдера.

– **Колекције и генеричке класе** – читање садржаја из бинарних и-или текстуалних фајлова и смештање података у колекцију, претрага колекције и приказ траженог резултата. Измена података у колекцији и смештање измењених података назад у фајл.

– **Рад са базама података** – читање података из базе, приказ на специфицираној контроли. Остварити промену, брисање и унос нових редова у циљану табелу базе података. Приказ података добијених укрштањем података из више табела на специфицираној контроли (,,).

Вредновање активности у оквиру тимског рада може се обавити тако да се од сваког члана тражи објашњење елемената урађеног рада и процена сопственог доприноса у оквиру тима.

За било коју активност којом се вреднује рад, унапред упознати ученике са критеријумима вредновања. На сваком часу подстицати ученике да дају своје мишљење и активно учествују у настави.

Приликом сваког вредновања дати ученику повратну информацију на основу које ће моћи да разуме грешке и побољша свој резултат и учење.

Охрабривати ученике да процењују сопствени напредак у учењу.

Назив предмета: Физика

Циљ учења Физике јесте стицање функционалне научне писмености, оспособљавање ученика за уочавање и примену физичких закона у свакодневном животу, развој логичког и критичког мишљења у истраживањима физичких феномена.

Разред **први**
Недељни фонд часова **2 часа**

ИСХОДИ По завршетку разреда ученик ће бити у стању да:	ТЕМА и кључни појмови садржаја програма
– објасни значај физике као фундаменталне науке и њену везу са природним и техничким наукама; – користи научни језик за описивање физичких појава; – решава квалитативне и квантитативне проблеме; – наведе основне физичке величине и њихове мерне јединице и објасни како се добијају јединице изведених физичких величина; – изврши директна мерења дужине, масе и времена и прикаже резултат мерења; – дефинише и описује основне кинематичке физичке величине; – разликује скаларне и векторске величине; – користи појмове брзине и убрзања при описивању механичког кретања; – разликује равномерно праволинијско кретање и равномерно променљиво праволинијско кретање и примењује законе кретања у једноставним примерима; – анализира графике равномерног и равномерно променљивог кретања; – објасни релативност брзине на примерима;	1. УВОД У ФИЗИКУ Физичке величине, ознаке, мерење и мерне јединице.