



Ministerul Educației al Republicii Moldova
Centrul de Excelență în Informatică și Tehnologii Informaționale



"Aprob"

Directorul Centrului de Excelență în
Informatică și Tehnologii Informaționale

 Vitalie Zavadschi

20 decembrie 2016

Curriculumul modular
F.04.O.016 Asistență pentru programarea orientată pe obiecte

Specialitatea: 61220 Administrarea bazelor de date
Calificarea: Asistent pentru baze de date

Chișinău 2016

Curriculumul a fost elaborat în cadrul Proiectului *EuropeAid/133700/C/SER/MD/12*
"Asistență tehnică pentru domeniul învățământ și formare profesională
în Republica Moldova",
implementat cu suportul financiar al Uniunii Europene



Autori:

Gîncu Silviu, doctor în pedagogie, grad didactic superior.

Pîrvan Evgheni, grad didactic superior, Colegiul „Iulia Hașdeu” din Cahul.

Șarapanovscaia Irina, grad didactic doi, Centrul de excelență în informatică și tehnologii informaționale.

Aprobat de:

Consiliul metodic-științific al Centrului de Excelență în Informatică și Tehnologii Informaționale.



Director

Vitalie Zavadschi

20 decembrie 2016

Recenzenți:

1. „EBS Integrator” SRL, adresa: str. Ion Inculeț 33, mun. Chișinău, director Aremes Vitalie.
2. ÎCS „Cedacri International” SRL, adresa: str. Ștefan cel Mare 171/1, mun. Chișinău, manager departament: Francesco Pipio.

Adresa Curriculumului în Internet:

Portalul național al învățământului profesional tehnic

<http://www.ipt.md/ro/produse-educationale>.

Cuprins

I. Preliminarii	4
II. Motivația, utilitatea modulului pentru dezvoltarea profesională.....	5
III. Competențele profesionale specifice modulului	5
IV. Administrarea modulului	5
V. Unitățile de învățare	6
VI. Repartizarea orientativă a orelor pe unități de învățare	11
VII. Studiu individual ghidat de profesor	11
VIII. Lucrările practice recomandate	12
IX. Sugestii metodologice	12
X. Sugestii de evaluare a competențelor profesionale	14
XI. Resursele necesare pentru desfășurarea procesului de studii	16
XII. Resursele didactice recomandate elevilor	17

I. Preliminarii

Programarea orientată pe obiecte este o paradigma de programare, axată pe ideea încapsulării, adică grupării datelor și codului care operează asupra lor într-o singură structură. Programarea orientată pe obiecte reprezintă unul din cei mai importanți pași făcuți în evoluția limbajelor de programare spre o mai puternică abstractizare în implementarea programelor. Ea a apărut din necesitatea exprimării problemei într-un mod mai natural ființei umane. Astfel, unitățile care alcătuiesc un program se apropie mai mult de modul uman de a gândi.

Statutul Curriculumului. Curriculumul modular “Asistență pentru programarea orientată pe obiecte” este un document normativ și obligatoriu pentru realizarea procesului de pregătire a tehnicienilor în învățământul profesional tehnic postsecundar, care, în conformitate cu sarcinile de lucru, vor acorda asistență în elaborarea și dezvoltarea diverselor aplicații și produse-program.

Funcțiile Curriculumului. Funcțiile de bază ale Curriculumul sunt:

- act normativ al procesului de predare, învățare, evaluare și certificare în contextul pedagogiei axate pe competențe;
- reper pentru proiectarea didactică și desfășurarea procesului educațional din perspectiva unei pedagogii axate pe competențe;
- componentă de bază pentru elaborarea strategiei de evaluare și certificare;
- orientare a procesului educațional spre formare de competențe la elevi;
- componentă fundamentală pentru elaborarea manualelor tipărite, manualelor electronice, ghidurilor metodologice, instrumentarului de evaluare.

Beneficiarii Curriculumului. Curriculumul este destinat:

- profesorilor din instituțiile de învățământ profesional tehnic postsecundar;
- autorilor de manuale și ghiduri metodologice;
- elevilor care își fac studiile la specialitatea în cauză;
- membrilor comisiilor pentru examenele de calificare;
- membrilor comisiilor de identificare, evaluare și recunoaștere a rezultatelor învățării, dobândite în contexte non-formale și informale.

Scopul studierii acestui modul constă în formarea și dezvoltarea competenței profesionale specifice de acordare de asistență în programarea orientată pe obiecte și de organizare a codului programelor la nivel de clase, în mentenanța și actualizarea produselor-program (softurilor) de sistem precum și a celui de aplicații. De asemenea, modulul contribuie la dezvoltarea competenței profesionale generale de respectare și de promovare a normelor de drept informatic.

Unitățile de curs ce în mod obligatoriu trebuie certificate până la demararea procesului de instruire la modulul în cauză:

- Programarea structurată.
- Programarea procedurală.
- Programarea calculatorului.

II. Motivația, utilitatea modulului pentru dezvoltarea profesională

Studierea acestui modul va contribui la formarea și dezvoltarea competențelor profesionale ce corespund nivelului patru de calificare:

- cunoștințe factice, principii, procese și concepte generale din domeniul elaborării produselor program;
- abilități cognitive și practice necesare pentru elaborarea aplicațiilor de consolă conform tematicilor incluse;
- asumarea responsabilității pentru mentenanța aplicațiilor.

Competențele formate și dezvoltate în cadrul acestui modul vor fi necesare pentru studierea unităților de curs orientate spre acordarea de asistență în elaborarea și dezvoltarea produselor-program. De asemenea, ele vor fi de un real folos în activitatea profesională a tehnicianului, în special, în ocupațiile legate de gestiunea produselor-program utilizate în companii.

III. Competențele profesionale specifice modulului

În cadrul modulului vor fi formate și dezvoltate următoarele competențe profesionale specifice:

CS1. Organizarea codului de program în stilul orientat pe obiecte.

CS2. Setarea domeniului de valori și a setului de operații admisibile ale obiectelor.

CS3. Implementarea conceptelor programării orientate pe obiecte în limbaje de programare.

CS4. Modificarea stării și comportamentului obiectelor în dependență de specificul problemei.

CS5. Acordarea de asistență în elaborarea aplicațiilor orientate pe obiecte.

IV. Administrarea modulului

Semestrul	Numărul de ore				Modalitatea de evaluare	Numărul de credite
	Total	Contact direct		Lucrul individual		
		Prelegeri	Practică/ Seminar			
IV	90	30	30	30	examen	3

V. Unitățile de învățare

Unități de competență	Unități de conținut	Abilități
1. Concepte ale programării orientată pe obiecte		
UC1. Implementarea conceptelor de abstractizare și încapsulare	<p>1. Stiluri de organizarea a codului unui program:</p> <ul style="list-style-type: none"> – programarea nestructurată; – programarea procedurală; – programarea modulară; – programarea orientată pe obiect. <p>2. Clase și obiecte. Sintaxă și semantică:</p> <ul style="list-style-type: none"> – structura unei clase; – modificatori de acces; – date și metode; – funcții inline; – pointerul *this; – obiecte. 	<p>A1. Utilizarea terminologiei specifice conceptelor de abstractizare și încapsulare.</p> <p>A2. Prezentarea situațiilor de utilizare a claselor.</p> <p>A3. Reprezentarea claselor prin diagrame.</p> <p>A4. Crearea unei clase conform specificațiilor propuse.</p> <p>A5. Declararea datelor și metodelor unei clase.</p> <p>A6. Protejarea membrilor clasei conform specificațiilor propuse.</p> <p>A7. Definirea metodelor unei clase.</p> <p>A8. Declararea obiectelor.</p> <p>A9. Accesarea datelor și metodelor unui obiect.</p> <p>A10. Accesarea datelor și metodelor unui obiect de tip pointer.</p> <p>A11. Inițializarea datelor unui obiect la nivel de metode.</p> <p>A12. Gestionarea obiectelor în cadrul unui program.</p> <p>A13. Elaborarea algoritmilor pentru tipuri de date de tip clasă.</p> <p>A14. Translarea algoritmilor pentru tipuri de date de tip clasă în limbajul de programare.</p> <p>A15. Implementarea algoritmilor pentru tipuri de date de tip clasă în limbajul de programare.</p>
UC2. Utilizarea constructorului și destructorului	<p>3. Constructori. Tipuri de constructori:</p> <ul style="list-style-type: none"> – constructor implicit; 	<p>A16. Definirea constructorului implicit al unei clase.</p> <p>A17. Definirea destructorului.</p>

Unități de competență	Unități de conținut	Abilități
	<ul style="list-style-type: none"> – constructor explicit; – constructor de copiere. 4. Destructor.	A18. Crearea de constructori expliți conform specificațiilor propuse. A19. Crearea constructorilor de copiere. A20. Inițializarea datelor unui obiect la nivel de constructor. A21. Apelarea constructorului/destructorului în cazul obiectelor de tip pointer. A22. Argumentarea utilizării constructorilor și destructorului.
2. Moștenire		
UC3. Implementarea conceptului de moștenire	5. Moștenire. 6. Clase de bază, clase derivate. 7. Derivarea claselor: <ul style="list-style-type: none"> – clase derivate; – clase de bază; – protejarea membrilor clasei; – modificatori de acces. 8. Constructorii și destructorii claselor derivate. 9. Moștenire multiplă.	A23. Reprezentarea mecanismului de moștenire prin diagrama claselor. A24. Prezentarea situațiilor de aplicare a moștenirii. A25. Derivarea unei clase. A26. Derivarea constructorului. A27. Moștenire datelor și metodelor unei clase conform specificațiilor propuse. A28. Derivarea metodelor unei clase. A29. Protejarea datelor între clase. A30. Verificarea corectitudinii definirii structurii claselor ce se află în relații de moștenire. A31. Inițializarea datelor unui obiect prin intermediul constructorului. A32. Crearea ierarhiilor de clase aflate în relația de moștenire conform specificațiilor propuse. A33. Elaborarea algoritmilor pentru clasele de bază/derivate. A34. Translarea algoritmilor clasele de bază/derivate în

Unități de competență	Unități de conținut	Abilități
		<p>limbajul de programare.</p> <p>A35. Implementarea algoritmilor pentru clasele de bază/derivate în limbajul de programare.</p> <p>A36. Aplicarea conceptului de moștenire în activitatea profesională.</p>
3. Polimorfism		
UC4. Implementarea polimorfismului în relația de moștenire	<p>10. Polimorfism în cadrul relației de moștenire:</p> <ul style="list-style-type: none"> – conceptul de polimorfism, – polimorfismul pur; – funcții virtuale; – funcții virtuale pure. <p>11. Clase abstracte și clase virtuale.</p>	<p>A37. Descrierea conceptului de polimorfism.</p> <p>A38. Prezentarea situațiilor de aplicare a polimorfismului.</p> <p>A39. Declararea funcțiilor virtuale.</p> <p>A40. Redefinirea metodelor claselor în relația de moștenire.</p> <p>A41. Utilizarea obiectelor de tip pointer.</p> <p>A42. Apelarea metodelor claselor de bază/virtuale ale unui obiect de tip pointer.</p> <p>A43. Declararea claselor abstracte.</p> <p>A44. Declararea claselor virtuale</p> <p>A45. Apelarea constructorilor în clasele virtuale.</p> <p>A46. Elaborarea fragmentelor de program bazate pe clase și funcții virtuale.</p> <p>A47. Elaborarea algoritmilor bazați pe conceptul de polimorfism.</p> <p>A48. Translarea algoritmilor bazați pe conceptul de polimorfism în limbajul de programare.</p>

Unități de competență	Unități de conținut	Abilități
		<p>A49. Implementarea algoritmilor p bazați pe conceptul de polimorfism în limbajul de programare.</p> <p>A50. Aplicarea conceptului de polimorfism în activitatea profesională.</p>
UC5. Implementarea polimorfismului ad-hoc	<p>12. Polimorfismul ad-hoc. Modalități de implementare:</p> <ul style="list-style-type: none"> – funcții membru; – funcții friend. 	<p>A51. Prezentarea situațiilor de aplicare a polimorfismului ad-hoc.</p> <p>A52. Declararea funcțiilor de tip friend în cadrul unei clase.</p> <p>A53. Declararea funcțiilor de tip friend în cadrul a două clase.</p> <p>A54. definirea funcțiilor friend.</p> <p>A55. Apelarea funcțiilor friend.</p> <p>A56. Elaborarea algoritmilor bazați pe funcții friend.</p> <p>A57. Supraîncărcarea operatorilor aritmetici prin intermediul funcțiilor membru/friend.</p> <p>A58. Supraîncărcarea operatorilor de intrare/ieșire.</p> <p>A59. Supraîncărcarea operatorilor relaționali.</p> <p>A60. Supraîncărcarea operatorului de atribuire.</p> <p>A61. Supraîncărcarea sub supraveghere a unor operatori conform specificațiilor propuse.</p> <p>A62. Utilizarea operatorilor supraîncărcați în cadrul programelor.</p> <p>A63. Aplicarea conceptului de polimorfism ad-hoc în activitatea profesională.</p>
4. Ierarhii de clase		
UC6. Prelucrarea tipurilor dinamice de date la nivel de obiecte	<p>13. Clase de obiecte.</p> <p>14. Obiecte de tip listă.</p>	<p>A64. Declararea unei clase de obiecte.</p> <p>A65. Definirea metodelor unei clase de obiecte.</p>

Unități de competență	Unități de conținut	Abilități
		<p>A66. Implementarea la nivel de metode a operațiilor specifice unei structuri dinamice de date.</p> <p>A67. Elaborarea algoritmilor pentru tipuri de date dinamice la nivel de obiecte.</p> <p>A68. Translarea algoritmilor pentru obiecte de tip listă.</p> <p>A69. Implementarea algoritmilor pentru obiecte de tip listă în limbajul de programare.</p>
UC7. Implementarea funcțiilor și claselor template	<p>15. Programarea generică.</p> <p>16. Funcții template.</p> <p>17. Clase template.</p>	<p>A70. Prezentarea situațiilor de aplicare a programării generice.</p> <p>A71. Elaborarea funcțiilor template.</p> <p>A72. Utilizarea claselor template conform specificațiilor propuse.</p> <p>A73. Elaborarea algoritmilor pentru funcții și clase template.</p> <p>A74. Translarea algoritmilor pentru funcții și clase template.</p> <p>A75. Implementarea algoritmilor pentru funcții și clase template în limbajul de programare.</p>
UC8. Implementarea conceptului de agregare	18. Ierarhii de clase.	<p>A76. Reprezentarea agregării în diagrama claselor.</p> <p>A77. Crearea ierarhiilor de clase formate din 3-5 clase.</p> <p>A78. Elaborarea algoritmilor pentru ierarhii de clase.</p> <p>A79. Translarea algoritmilor pentru ierarhii de clase.</p> <p>A80. Implementarea algoritmilor pentru ierarhii de clase în limbajul de programare.</p> <p>A81. Aplicarea conceptelor: abstractizare, incapsulare, moștenire, polimorfism și agregare în activitatea profesională.</p>

VI. Repartizarea orientativă a orelor pe unități de învățare

Nr. crt.	Unități de învățare	Numărul de ore			
		Total	Contact direct		Lucrul individual
			Prelegeri	Practică/ Seminar	
1.	Concepte ale programării orientată pe obiecte	24	8	8	8
2.	Moștenire	18	6	6	6
3.	Polimorfism	24	8	8	8
4	Ierarhii de clase	24	8	8	8
	Total	90	30	30	30

VII. Studiu individual ghidat de profesor

Materii pentru studiul individual	Produse de elaborat	Modalități de evaluare	Termeni de realizare
1. Concepte ale programării orientată pe obiecte			
Clase și obiecte	Portofoliu: Set de aplicații de consolă cu utilizarea claselor și obiectelor. Referat: Caracteristici ale stilurilor de programare	Prezentarea portofoliului Comunicare	Săptămâna 3
Constructorii și destructorii	Portofoliu: Set de aplicații de consolă cu utilizarea constructorilor și destructorilor	Prezentarea portofoliului	Săptămâna 4
2. Moștenire			
Moștenire simplă	Portofoliu: Set de aplicații de consolă bazate pe utilizarea relației de moștenire între clase.	Prezentarea portofoliului	Săptămâna 6
Moștenire multiplă	Portofoliu: Set de aplicații de consolă bazate pe utilizarea relației de moștenire multiplă între clase.	Prezentarea portofoliului	Săptămâna 7

Materii pentru studiul individual	Produse de elaborat	Modalități de evaluare	Termeni de realizare
3. Polimorfismul			
Polimorfismul dinamic	Portofoliu: Set de aplicații de consolă bazate pe implementarea polimorfismului dinamic	Prezentarea portofoliului	Săptămâna 9
Polimorfismul ad-hoc	Portofoliu: Set de aplicații de consolă bazate pe implementarea polimorfismului ad-hoc	Prezentarea portofoliului	Săptămâna 11
4. Ierarhii de clase			
Programare generică	Portofoliu: Set de aplicații de consolă cu utilizarea funcțiilor și claselor template	Prezentarea portofoliului	Săptămâna 12
Ierarhii de clase	Portofoliu: Set de aplicații de consolă cu crearea ierarhiilor de clase constituite din 3-5 clase	Prezentarea portofoliului	Săptămâna 15

VIII. Lucrările practice recomandate

Lucrările practice vor fi efectuate în formă de lucrări de laborator. Tematica lucrărilor recomandate:

1. Prelucrarea tipurilor de date frecvent utilizate (structuri, masive etc) la nivel de clase.
2. Utilizarea constructorilor și destructorului.
3. Implementarea relației de moștenire simplă.
4. Implementarea moștenirii multiple.
5. Implementarea polimorfismul dinamic.
6. Implementarea polimorfismul ad-hoc.
7. Prelucrarea tipurilor de date dinamice la nivel de clase.
8. Prelucrarea datelor la nivel funcții și clase template.
9. Implementarea ierarhiilor formate din 3-5 clase.

IX. Sugestii metodologice

Elementul de bază al Curriculumului sunt competențele ce trebuie formate și dezvoltate în procesul de formare profesională. Acestea vor fi formate prin organizarea eficientă a procesului de instruire. Pentru aceasta sunt necesare două condiții:

1. Organizarea activităților. Pentru buna organizare a procesului didactic ambii participanți necesită de a-și organiza activitățile. De modul cum sunt organizate acestea depinde în mare

măsură nivelul de formare a competențelor. În această ordine de idei, în procesul de organizare a activităților se vor asigura:

- condiții optime pentru buna colaborare dintre elev și profesor;
- un set de procese care duc la îmbunătățirea relațiilor dintre părți;
- un nivel de implicare a părților acționând în baza unor reguli și acțiuni prestabilite.

2. *Selectarea adecvată a metodelor de instruire.* Se recomandă utilizarea metodelor de instruire precum:

Simularea și modelarea. Simularea este utilizată pentru prezentarea la faza inițială a unor concepte, oferind posibilitatea de ghidare a activității elevului în bază de situații practice. Prin intermediul acestei metode se pot reda, prin analogie, diverse situații, raționamente, care pot să reprezinte relații dintre obiecte, fenomene, procese etc. Această metodă se recomandă pentru predarea-învățarea-evaluare următoarelor materii Sintaxa și semantica claselor, moștenire, ierarhii de clase ș. a.

Problematizarea mai poate fi denumită și predare prin rezolvare de probleme sau predare productivă de probleme. Conform acestei metode instruitului este pus în fața unor dificultăți create în mod deliberat, și prin depășirea lor învață ceva nou. „Punctul forte” al metodei îl constituie situația-problemă. Din această cauză este necesar de a formula corect situația. La crearea situației de tip problemă se va ține cont de următoarele caracteristici:

- A. Situația trebuie să prezinte o dificultate pentru instruit, iar pentru a găsi soluția, acesta se va confrunta cu efort de gândire;
- B. Situația trebuie să prezinte interes, astfel încât acesta să acționeze spre a rezolva problema;
- C. Situația trebuie să orienteze activitatea instruitului spre a rezolva problema și de al cointeresa pe acesta de a dobândi noi cunoștințe;
- D. Rezolvarea situației nu va fi posibilă fără a apela la resurselor recent dobândite.

Prin intermediul situației create, instruitul este cointerestat de a studia, analiza și a participa la rezolvarea problemei. Aplicarea acestei metode presupune parcurgerea a patru etape:

1. Formularea problemei – este descrisă situația problemă, explicarea, după necesitate a diferitor puncte cheie, care ar permite instruitului să perceapă problema;
2. Studiarea problemei – se lucrează în mod independent, sunt reactualizate anumite resurse;
3. Determinarea soluției – în cadrul acestei etape sunt pregătite resursele necesare, se descoperă mijloacele care duc la rezolvarea problemei și este analizat modul de aplicare a acestora în determinarea soluției;
4. Obținerea rezultatului final – se analizează rezultatul obținut și formate anumite concluzii.

Această metodă se recomandă pentru predarea-învățarea-evaluare următoarelor materii constructori, moștenire simplă/multiplă, polimorfismul dinamic ș. a.

Algoritmizarea reprezintă o metodă de predare-învățare bazată pe utilizarea și valorificarea algoritmilor în procesul de instruire. Algoritmul de instruire se reprezintă sub forma unui grup

de scheme, unui set de operații, iar prin parcurgerea lor într-o ordine bine stabilită duce la rezolvarea unui set de probleme caracteristice unei familii de situații. În rezultatul aplicării acestei metode se va oferi posibilitatea elevului de a elabora treptat propriile scheme, aplicabile în diferite circumstanțe didactice. Această metodă se recomandă pentru predarea-învățarea-evaluare următoarelor materii polimorfismul ad-hoc, programarea generică, ierarhii de clase ș. a.

Metoda studiul de caz valorifică o situație reală care se analizează și se rezolvă. Așa cum problemele rezolvate în stilul orientat pe obiecte au un grad sporit de dificultate, sunt cazuri când este necesar de a prezenta elevului probleme deja rezolvate. Avantajul metodei, constă în faptul că fiecare dintre elev își va aduce aportul la analiza și rezolvarea problemei. În utilizarea acestei metode se conturează câteva etape: 1) Selectarea și prezentarea cazului; 2) Organizarea echipelor de lucru; 3) Prelucrarea și conceptualizarea; 4) Structurarea finală a studiului. Această metodă se recomandă pentru predarea-învățarea-evaluare următoarelor materii ierarhii de clase, polimorfismul dinamic.

X. Sugestii de evaluare a competențelor profesionale

Evaluarea competențelor profesionale este procesul prin care sunt colectate și analizate dovezile necesare pentru judecarea competenței în raport cu cerințele calificării profesionale. Calificarea profesională este documentul în care se descriu rezultatele învățării în concordanță cu cerințele pieței muncii, specificate în standardul ocupațional/ profilul ocupațional. Evaluarea competențelor profesionale este un proces complet diferit de sistemul tradițional de evaluare a cunoștințelor. Evaluarea competențelor profesionale este un proces care presupune consultarea și colaborarea dintre elev și profesor. Evaluarea competențelor are loc prin furnizarea de către elev a dovezilor de competență care sunt interpretate de către profesor. Dovezile de competență acumulate sunt rezultate considerate parțiale și atât elevul cât și profesorul pot solicita clarificări suplimentare.

Procedura de evaluare a competențelor profesionale pentru modulul *Asistență pentru programarea orientată pe obiecte*, va oferi elevilor posibilitatea de a-și demonstra atât cunoștințele teoretice și practice. Metodele folosite în procesul de evaluare vor evidenția cunoștințele și deprinderile necesare pentru efectuarea activităților de muncă și, mai ales, capacitatea elevului de a obține rezultatele practice așteptate.

Activitățile de evaluare vor fi orientate spre motivarea elevilor și obținerea unui feedback continuu, fapt ce va permite corectarea operativă a procesului de învățare, stimularea autoevaluării și a evaluării reciproce, evidențierea succeselor, implementarea evaluării selective sau individuale. Pentru a eficientiza procesele de evaluare, înainte de a demara evaluările, cadrul didactic va aduce la cunoștința elevilor tematica lucrărilor, modul de evaluare (bareme/grile/criterii de notare) și condițiile de realizare a fiecărei evaluări.

Evaluarea curentă/formativă se va realiza prin diverse modalități: observarea comportamentului elevului, analiza rezultatelor activității elevului, discuția/conversația, prezentarea proiectelor individuale de activitate. Prin evaluarea curentă/formativă, cadrele didactice informează elevul despre nivelul de performanță; îl motivează să se implice în dobândirea competențelor profesionale.

Evaluarea sumativă se realizează la finele modulului în baza simulării în atelier a unei situații de problemă din contexte profesionale variate, care solicită elevului demonstrarea competenței profesionale. Cadrele didactice vor elabora sarcini prin care vor orienta comportamentul profesional al elevului spre demonstrarea sistemului de cunoștințe și abilități. În acest scop, vor fi clar stabiliți indicatorii și descriptorii de performanță ai procesului și produsului realizat de către elev.

Portofoliul reprezintă o metodă complexă de evaluare în care un rezultat al evaluării este elaborat pe baza aplicării unui ansamblu variat de probe și instrumente de evaluare. Portofoliul, de regulă este realizat pe o perioadă mai îndelungată (în decursul mai multor ore). Conținutul unui portofoliu este reprezentat de rezultatele la: lucrări practice, studiul individual, investigații, referate și proiecte, observarea sistematică la clasă, autoevaluarea elevului, chestionare de atitudini etc. Alegerea elementelor ce formează portofoliul este realizată de către profesor (astfel încât acestea să ofere informații concludente privind pregătirea, evoluția, atitudinea elevului) sau chiar de către elev (pe considerente de performanță, preferințe etc.). Structurarea evaluării sub forma de portofoliu se dovedește deosebit de utilă, atât pentru profesor, cât și pentru elev sau părinții acestuia. Pentru a realiza o evaluare pe bază de portofoliu, profesorul:

- va comunica elevilor intenția de a realiza un portofoliu, adaptând instrumentele de evaluare ce constituie “centrul de greutate” ale portofoliului la specificul unității de învățare;
- va alege componentele ce formează portofoliul, dând și elevului posibilitatea de a adăuga piese pe care le consideră relevante pentru activitatea sa;
- va evalua separat fiecare piesă a portofoliului în momentul realizării ei, dar va asigura și un sistem de criterii pe baza cărora să realizeze evaluarea globală și finală a portofoliului;
- va pune în evidență evoluția elevului, particularitățile de exprimare și de raportare a acestuia la aria vizată;
- va integra rezultatul evaluării portofoliului în sistemul general de notare.

Competențele elevului se manifestă prin produse concrete, care sunt analizate de către profesor în raport cu aspectele critice stabilite pentru unitate/unitățile de competență pentru care este evaluat. Dovezile de competență sunt informațiile produse de un elev din care rezultă că îndeplinește toate aspectele descrise de unitatea/unitățile de competență pentru care este evaluat, respectiv are cunoștințele și deprinderile necesare.

Evaluarea nivelului de dezvoltare a competențelor în cadrul orelor:

teoretice se va realiza prin teste, exemple de aplicare a cunoștințelor teoretice în practică, machete etc.;

de laborator se va realiza prin elaborarea de către elev, în termeni concreți, a aplicațiilor web avînd la bază unitățile de conținut studiate în cadrul orelor teoretice precum și abilitățile anterior dezvoltate;

de studiu individual se va realiza prin studierea de către elev a materialelor suplimentare decît cele oferite în cadrul orelor de tip contact direct și prezentarea de portofolii pentru anumite unități de conținut prin care elevul își va demonstra abilitățile formate.

Probe de evaluare a competențelor, în baza situațiilor de problemă de la viitoarele locuri de muncă:

- implementarea conceptelor programării orientate pe obiecte;
- crearea structurilor de clase;
- organizarea codului de program în stilul orientat pe obiecte;
- reprezentarea conceptelor programării orientate pe obiecte prin diagrama claselor;
- modificarea stării și comportamentului obiectelor conform specificațiilor propuse;
- testarea aplicațiilor de consolă elaborate.

În calitate de **produse pentru măsurarea competențelor** se vor folosi:

- aplicații de consolă elaborate conform specificațiilor propuse;
- algoritmi elaborați conform specificațiilor propuse;
- obiecte gestionate conform specificațiilor propuse.

Criteriile de evaluare a produselor pentru măsurarea competenței vor include:

- Utilizarea corectă a mecanismelor programării orientate pe obiecte.
- Corectitudinea claselor elaborate.
- Fundamentarea deciziilor.
- Ținuta lingvistică.
- Respectarea termenilor de elaborare.

XI. Resursele necesare pentru desfășurarea procesului de studii

Cerințe față de sălile de curs	
Pentru orele teoretice	Cabinet de informatică cu 15 calculatoare Proiector
Pentru orele de laborator	Laborator de informatică care asigură fiecărui elev un calculator
Cerințe tehnice	
Parametri tehnici minimi ale calculatorului	Procesor: 2 GHz Memorie operativă: 4 GB Unitate de stocare: 500 GB Afișaj și grafică: size: 22'', resolution: 1366x768 Network: Ethernet, 100 Mb
Software	Sistem de Operare Microsoft Windows Code::Blocks Dev C/CPP Visual Studio 2015

XII. Resursele didactice recomandate elevilor

Nr. crt.	Denumirea resursei	Locul în care poate fi consultată/ accesată resursa
1.	A. Braicov, S. Gîncu, Borland C++ Builder. Chişinău, 2009. http://en.calameo.com/read/002801569838bdc5a9164	Internet
2.	A. Ruceanu, Programarea orientată pe obiecte: limbajul C++, 2007. http://www.ruceanu.ro/adrian/wp-content/cursuri/poo2013.php	Internet
3.	I. Lupu, V. Cabac, S. Gîncu Formarea şi dezvoltarea competenţei de programare orientată pe obiecte la viitorii profesori de informatică, Chişinău, 2013. http://en.calameo.com/books/0028015690bbcf6a9e03	Internet
4	S. Gîncu Metodologia rezolvării problemelor de informatică în stilul orientat pe obiecte, Chişinău, 2012. http://en.calameo.com/read/002801569ce96f41ef59f	Internet
5	Programarea orientată pe obiect şi programarea vizuală. http://colegiulbratianu.ro/wp-content/themes/theme53309/documente/software/DotNet.pdf	Internet
6	Booch G. Object-Oriented Design with Applications. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994. http://dl.acm.org/citation.cfm?id=174890	Internet
7	Фаулер М. UML. Основы / Пер. с англ. 3-е изд. СПб: Символ-Плюс, 2004. http://www.studfiles.ru/preview/1500382/	Internet