

## FIȘA DISCIPLINEI

### 1. Date despre program

1.1 Instituția de învățământ superior	Universitatea Babeș-Bolyai Cluj-Napoca
1.2 Facultatea	Facultatea de Matematica și Informatică
1.3 Departamentul	Departamentul de informatică
1.4 Domeniul de studii	Matematica
1.5 Ciclul de studii	Licenta
1.6 Programul de studiu / Calificarea	Matematica – limba de studiu română

### 2. Date despre disciplină

2.1 Denumirea disciplinei	Programare orientată obiect						
2.2 Titularul activităților de curs	Diosan Laura						
2.3 Titularul activităților de seminar	Diosan Laura						
2.4 Anul de studii	1	2.5 Semestrul	2	2.6. Tipul de evaluare	E	2.7 Regimul disciplinei	Obligativ

### 3. Timpul total estimat (ore pe semestru al activităților didactice)

3.1 Număr de ore pe săptămână	5	Din care: 3.2 curs	2	3.3 seminar/laborator	3
3.4 Total ore din planul de învățământ	70	Din care: 3.5 curs	28	3.6 seminar/laborator	42
Distribuția fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie și notițe					30
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					25
Pregătire seminarii/laboratoare, teme, referate, portofolii și eseuri					20
Tutoriat					2
Examinări					3
Alte activități: .....					
3.7 Total ore studiu individual		80			
3.8 Total ore pe semestru		150			
3.9 Numărul de credite		6			

### 4. Precondiții (acolo unde este cazul)

4.1 de curriculum	<ul style="list-style-type: none"> <li>Algoritmica, structuri de date, statistica</li> </ul>
4.2 de competențe	<ul style="list-style-type: none"> <li>Abilități medii de programare într-un limbaj de nivel înalt (orientat obiect)</li> </ul>

### 5. Condiții (acolo unde este cazul)

5.1 De desfășurare a cursului	<ul style="list-style-type: none"> <li></li> </ul>
5.2 De desfășurare a seminarului/laboratorului	<ul style="list-style-type: none"> <li>Pentru activitatea de laborator este nevoie de calculatoare cu medii de programare avansate</li> </ul>

## 6. Competențele specifice acumulate

<b>Competențe profesionale</b>	<p>C1.1 Descrierea adecvată a paradigmatelor de programare și a mecanismelor de limbaj specifice, precum și identificarea diferenței dintre aspectele de ordin semantic și sintactic.</p> <p>C1.3 Elaborarea codurilor sursă adecvate și testarea unitară a unor componente într-un limbaj de programare cunoscut, pe baza unor specificații de proiectare date</p>
<b>Competențe transversale</b>	<p>CT1 Aplicarea regulilor de muncă organizată și eficientă, a unor atitudini responsabile față de domeniul didactic-științific, pentru valorificarea creativă a propriului potențial, cu respectarea principiilor și a normelor de etică profesională</p> <p>CT3 Utilizarea unor metode și tehnici eficiente de învățare, informare, cercetare și dezvoltare a capacităților de valorificare a cunoștințelor, de adaptare la cerințele unei societăți dinamice și de comunicare în limba română și într-o limbă de circulație internațională</p>

## 7. Obiectivele disciplinei (reieșind din grila competențelor acumulate)

7.1 Obiectivul general al disciplinei	<ul style="list-style-type: none"> <li>Programarea orientată pe obiect are drept obiectiv însușirea principiilor de bază a programării cu ajutorul tipurilor abstrakte de date (obiectelor).</li> </ul>
7.2 Obiectivele specifice	<p>După însușirea materialului prezentat la această disciplină studenții ar trebui:</p> <ul style="list-style-type: none"> <li>sa poată rezolva probleme de dimensiuni mici și medii într-o manieră orientată pe obiecte</li> <li>sa poată evidenția diferența între proiectarea funcțională tradițională și proiectarea orientată pe obiecte</li> <li>sa înțeleagă rolul mostenirii, polimorfismului, legării dinamice și a structurilor generice în dezvoltarea unor programe reutilizabile</li> <li>sa explice și sa folosească diferite strategii de programare referitor la tratarea excepțiilor și asertiuni formale</li> <li>sa poată scrie programe C++ de dimensiuni mici/medii</li> <li>în timpul rezolvării unei probleme sa folosească clase scrise de alți programatori</li> <li>sa înțeleagă și sa folosească structurile de date fundamentale: colecții, mulțimi, tabele, liste, stive, cozi, arbori, grafe</li> </ul>

## 8. Conținuturi

8.1 Curs	Metode de predare	Observații
<p>În prima parte a cursului sunt introduse gradat conceptele programării orientate pe obiecte. Pentru exemplificări este folosit limbajul C++. Sunt supuse dezbaterilor diferite structuri de date ce au fost prezentate în cursul de introducere în informatică. În partea a doua sunt prezentate subiecte mai avansate de programare C++: ierarhii de clase standard, programarea dirijată de evenimente, componente C++ pentru interfața cu utilizatorul, tratarea excepțiilor.</p> <p>Curs 1 Elemente de bază ale limbajului C++.</p>	<p>Expunerea Conversația Problematizarea</p>	

<ul style="list-style-type: none"> <li>- Elemente lexicale. Operatori. Conversii.</li> <li>- Tipuri de date. Variabile. Constante.</li> <li>- Domeniu de vizibilitate si durata de viata a variabilelor.</li> <li>- Spatii de nume.</li> <li>- Instructiuni C++.</li> <li>- Declararea si definitia functiilor in C++.</li> <li>- Supraincercarea functiilor. Functii inline.</li> </ul>		
<p>Curs 2</p> <p>Tipuri de date derivate si utilizator si alocare dinamica in C++.</p> <ul style="list-style-type: none"> <li>- Tipurile tablou si structura.</li> <li>- Tipurile pointer si referinta.</li> <li>- Alocarea si dealocarea memoriei.</li> <li>- Pointeri la functii si pointeri void.</li> </ul> <p>Programare modulara in C++.</p> <ul style="list-style-type: none"> <li>- Fisiere header. Biblioteci.</li> <li>- Implementari modulare de tipuri abstracte de date.</li> <li>- Folosirea tipului pointer void pentru obtinerea genericitatii.</li> </ul>	<p>Expunerea</p> <p>Conversația</p> <p>Demonstrația didactică</p> <p>Algoritmizarea</p>	
<p>Curs 3</p> <p>Metoda programarii orientate-obiect in C++.</p> <ul style="list-style-type: none"> <li>- Clase si obiecte.</li> <li>- Membrii unei clase. Specificatori de acces.</li> <li>- Constructori / destructori</li> <li>- Diagrame UML pentru clase (membrii, acces).</li> </ul>	<p>Expunerea</p> <p>Conversația</p> <p>Demonstrația didactică</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 4</p> <ul style="list-style-type: none"> <li>- Supraincercarea operatorilor.</li> <li>- Membrii statici.</li> <li>- Functii Friend.</li> </ul>	<p>Expunerea</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 5</p> <ul style="list-style-type: none"> <li>- Clase: containere si iteratori <ul style="list-style-type: none"> <li>o Lista simplu inlantuita si lista dublu inlantuita</li> <li>o Stiva, Coada, Tabele de dispersie</li> <li>o Relatia de asociere/agregare intre clase - reprezentare UML</li> </ul> </li> </ul>	<p>Expunerea</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 6</p> <ul style="list-style-type: none"> <li>- Relatia de mostenire <ul style="list-style-type: none"> <li>o Mostenire simpla. Clase derivate.</li> <li>o Principiul substitutiei.</li> <li>o Suprascrierea metodelor.</li> <li>o Mostenire multipla.</li> <li>o Relatia de specializare/generalizare intre clase - reprezentare UML.</li> </ul> </li> </ul>	<p>Expunerea</p> <p>Conversația</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 7</p> <ul style="list-style-type: none"> <li>- Clase abstracte si Polimorfism. <ul style="list-style-type: none"> <li>o Metode virtuale.</li> <li>o Legare dinamica.</li> <li>o Mostenire virtuala.</li> <li>o Reutilizare cod (mostenire/compozitie).</li> <li>o Conversii (upcast/downcast).</li> </ul> </li> </ul>	<p>Expunerea</p> <p>Demonstrația didactică</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 8</p> <ul style="list-style-type: none"> <li>- Proiectare orientata-obiect si proiectare bazata pe interfete. <ul style="list-style-type: none"> <li>o Clase abstracte, interfete.</li> <li>o Reprezentare UML pentru interfete.</li> <li>o Proiectarea orientata-obiect a unei biblioteci de structuri de date.</li> </ul> </li> </ul>	<p>Expunerea</p> <p>Conversația</p> <p>Demonstrația didactică</p> <p>Algoritmizarea</p> <p>Problematizarea</p>	
<p>Curs 9</p> <ul style="list-style-type: none"> <li>- Sabloane (Template).</li> </ul>	<p>Expunerea</p> <p>Conversația</p>	

<ul style="list-style-type: none"> <li>○ Functii template. Clase template.</li> <li>○ Reutilizarea codului sursa.</li> </ul>	Algoritmizarea Problematizarea	
<p>Curs 10</p> <ul style="list-style-type: none"> <li>- Operatii de intrare/iesire. <ul style="list-style-type: none"> <li>○ Fluxuri de intrare/ iesire. Ierarhii de clase pentru I/O.</li> <li>○ Formatare. Manipulatori.</li> <li>○ Lucrul cu fisiere.</li> </ul> </li> </ul>	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
<p>Curs 11</p> <ul style="list-style-type: none"> <li>- Tratarea exceptiilor. <ul style="list-style-type: none"> <li>○ Notiunea de exceptie in programare</li> <li>○ Tratarea exceptiilor in C++.</li> </ul> </li> </ul>	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
<p>Curs 12</p> <ul style="list-style-type: none"> <li>- Biblioteca STL <ul style="list-style-type: none"> <li>○ Clase container si iterator.</li> <li>○ Folosirea algoritmilor din STL.</li> </ul> </li> </ul>	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
<p>Curs 13</p> <ul style="list-style-type: none"> <li>- Interfete grafice <ul style="list-style-type: none"> <li>○ MCF</li> <li>○ VCL</li> <li>○ QT</li> </ul> </li> </ul>	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
<p>Curs 14</p> <ul style="list-style-type: none"> <li>- Sabloane de proiectare <ul style="list-style-type: none"> <li>○ Singleton</li> <li>○ Model View Controller</li> <li>○ Repository</li> </ul> </li> </ul>	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
<p>Bibliografie</p> <ol style="list-style-type: none"> <li>1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley Publ., Massachusetts, 1983.</li> <li>2. R. Andonie, I. Garbacea, Algoritmi fundamentali. O perspectiva C++, Editura Libris,</li> <li>3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002</li> <li>4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.</li> <li>5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.</li> <li>6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.</li> <li>7. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.</li> <li>8. Dan Roman, Ingineria programarii obiectuale, Editura Albastra, Cluj_Napoca, 1996.</li> <li>9. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.</li> <li>10. Bruce Eckel, Thinking in C++, <a href="http://www.bruceeckel.com">www.bruceeckel.com</a></li> </ol>		
8.2 Seminar / laborator	Metode de predare	Observa ții

<p>S 1. Probleme simple in C++. Functii. Variabile (locale si globale) si vizibilitatea lor. Vectori (uni si multidimensionali) si structuri.</p> <p>L 1. Strategii și metode inteligente de rezolvare a jocurilor</p>	<p>Conversația Algoritmizarea Descoperirea Studiul individual Exercițiul</p>	<p>Fiecare seminar dureaza 2 ore si se desfasoara o data la 2 saptamani</p>
<p>S 2. TAD container cu elemente generice (void*): reprezentare vizibila si ascunsa. Citiri si scrieri din/in fisiere.</p> <p>L 2. Specificarea, proiectarea si implementarea unor probleme simple in C/C++. Aspecte generale ale limbajelor C si C++. Versiuni structurate si modulare ale aplicatiilor C/C++. Specificarea, proiectarea si implementarea unui TAD in C/C++. Reprezentarea unui TAD. Operatiile unui TAD. Utilizarea unui TAD.</p>	<p>Conversația Algoritmizarea Problematizarea Studiul de caz Cooperarea Studiul individual Exercițiul</p>	<p>Fiecare laborator dureaza 2 ore si se desfasoara o data la 2 saptamani</p>
<p>S 3. Clase. Clase simple. Supraincararea operatorilor. Clase cu date membre de tip obiect.</p> <p>L 3. Specificarea, proiectarea si implementarea unui TAD container in C/C++. Utilizarea elementelor generice pentru popularea containerului. Reprezentarea TAD vector cu elemente generice. Operatiile TAD vector cu elemente generice. Utilizarea TAD vector cu elemente generice. Specificarea, proiectarea si implementarea unui TAD container dinamic in C/C++. Utilizarea elementelor generice si a iteratorilor. Reprezentarea TAD container dinamic cu elemente generice. Operatiile TAD container cu elemente generice. Utilizarea TAD container cu elemente generice. Utilizarea iteratorilor.</p>	<p>Conversația Algoritmizarea Problematizarea Descoperirea Simularea Studiul individual Exercițiul</p>	<p>Fiecare laborator dureaza 2 ore si se desfasoara o data la 2 saptamani</p>
<p>S 4. Clase de tip lista dinamica si iteratori. Mostenire.</p> <p>L 4. Clase si obiecte. Specificarea, proiectarea si implementarea unei clase in C++. Constructori, destructor, accesori, mutatori. Supraincararea operatorilor. Obiecte statice si dinamice. Clase cu date membre de tip obiect. Specificarea, proiectarea si implementarea acestor clase. Obiecte dinamice.</p>	<p>Conversația Algoritmizarea Problematizarea Studiul de caz Brainstorming-ul Studiul individual Exercițiul</p>	
<p>S 5. Clase abstracte si interfete. Polimorfism.</p> <p>L 5. Clase si obiecte. Containere dinamice cu obiecte (liste dinamice, dictionare, tabele de dispersie). Iteratori pentru aceste containere. Specificarea, proiectarea si implementarea unei clase in C++. Constructori, destructor, accesori, mutatori. Clase cu date membre de tip obiect. Obiecte dinamice. Supraincararea operatorilor. Mostenire.</p>	<p>Conversația Algoritmizarea Problematizarea Descoperirea Studiul de caz Studiul individual Exercițiul</p>	
<p>S 6. Sabloane si exceptii.</p> <p>L 6. Clase si obiecte. Mostenire. Polimorfism. Specificarea, proiectarea si implementarea claselor derivate. Clase abstracte si interfete. Sabloane. Sabloane si exceptii. Specificarea, proiectarea si implementarea claselor in C++. Implementarea unei probleme respectand o diagrama UML. Sabloane de proiectare.</p>	<p>Conversația Algoritmizarea Studiul de caz Simularea Studiul individual Exercițiul</p>	
<p>S 7. Probleme complexe implementate pe baza unei diagrame UML. Sabloane de proiectare.</p> <p>L 7. -</p>	<p>Conversația Algoritmizarea Problematizarea Studiul de caz Brainstorming-ul Studiul</p>	

**Bibliografie**

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley Publ., Massachusetts, 1983.
2. R. Andonie, I. Garbacea, Algoritmi fundamentali. O perspectiva C++, Editura Libris,
3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicade, Editura Teora, 2002
4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.
8. Dan Roman, Ingineria programarii obiectuale, Editura Albastra, Cluj\_Napoca, 1996.
9. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
10. Bruce Eckel, Thinking in C++, [www.bruceeckel.com](http://www.bruceeckel.com)
11. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.

**9. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatori reprezentativi din domeniul aferent programului**

- Cursul respecta recomandările curriculare IEEE și ACM pentru studiile în informatică
- Cursul există în programa de studiu a numeroase facultăților de profil din întreaga lume
- Companiile de software consideră conținutul cursului ca fiind util în dezvoltarea abilităților de modelare și programare ale studenților

**10. Evaluare**

Tip activitate	10.1 Criterii de evaluare	10.2 metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	<ul style="list-style-type: none"> <li>• Cunoașterea conceptelor de bază ale domeniului</li> <li>• Aplicarea principiilor de programare orientată obiect din conținutul cursului pentru rezolvarea problemelor complexe și dificile</li> </ul>	Examen scris	40%
	<ul style="list-style-type: none"> <li>• Aplicarea principiilor de programare orientată obiect din conținutul cursului pentru rezolvarea problemelor complexe și dificile</li> </ul>	Proba practică	20%
10.5 Seminar/laborator	<ul style="list-style-type: none"> <li>• Specificarea, proiectarea, implementarea și testarea aplicațiilor</li> <li>• Rezolvarea efectivă a problemelor cu ajutorul metodelor anterior implementate</li> </ul>	Observarea sistematică a studentului în timpul rezolvării temelor	10%

	<ul style="list-style-type: none"> <li>• Specificarea, proiectarea, implementarea și testarea aplicațiilor</li> <li>• Rezolvarea efectivă a problemelor cu ajutorul metodelor anterior implementate</li> </ul>	Observarea sistematică a studentului în timpul rezolvării temelor de laborator	30%
--	--	--	-----

#### 10.6 Standard minim de performanță

- Fiecare student trebuie să demonstreze că a atins un nivel acceptabil de cunoaștere și înțelegere a domeniului, că este capabil să exprime cunoștințele într-o formă coerentă, că are capacitatea de a stabili anumite conexiuni și de a utiliza cunoștințele în rezolvarea unor probleme.
- Pentru a promova examenul studentul trebuie să:
  - Realizeze cel puțin 70% din temele de laborator.

Data completării

30 aprilie 2015

Semnătura titularului de curs

Conf. Dr. Diosan Laura

Semnătura titularului de seminar

Conf. Dr. Diosan Laura

Data avizării în departament

.....

Semnătura directorului de departament

Prof. Dr. Parv Bazil