Anexa 2 la Ordinul ministrului educaţiei şi cercetării nr. 3252 / 13.02.2006

MINISTERUL EDUCAŢIEI ŞI CERCETĂRII

CONSILIUL NAŢIONAL PENTRU CURRICULUM

PROGRAME ŞCOLARE PENTRU CICLUL SUPERIOR AL LICEULUI

I N F O R M A T I C Ă

Specializarea: Matematică-informatică, intensiv informatică

CLASA A XI-A

Aprobat prin ordinul ministrului
Nr. 3252 / 13.02.2006

Bucureşti, 2006

NOTA DE PREZENTARE

Prezentul document cuprinde programa şcolară de *Informatică-intensiv*, pentru clasa a XI-a, ciclul superior al liceului, **specializarea matematică-informatică**.

În elaborarea prezentei programei şcolare au fost respectate principiile de proiectare curriculară, specifice curriculumului naţional, valorificându-se în acelaşi timp tendinţele domeniului pe plan internaţional şi opinii ale unor profesori cu o bogată experienţă didactică.

În baza planurilor-cadru pentru ciclul superior al liceului, aprobate prin Ordinul ministrului educaţiei şi cercetării nr. 5718/ 22.12.2005, la clasele de matematică-informatică – intensiv informatică, acestei discipline i se alocă în total **7 ore** / săptămână.

Programa are următoarele componente:

- Notă de prezentare
- Competenţe generale
- Valori şi atitudini
- Competenţe specifice şi conţinuturi
- Sugestii metodologice.

Competenţe specifice, definite pentru disciplina *Informatică – intensiv*, la nivelul clasei a XI-a, sunt derivate din competenţele generale şi reprezintă ansambluri structurate de cunoştinţe şi deprinderi ce urmează a fi dobândite de către elevi prin învăţare, pe durata anului de studiu.

Studiul disciplinei *Informatică* se va desfăşura cu întreg colectivul de elevi ai clasei pentru activităţile teoretice şi cu colectivul de elevi organizat pe grupe, obligatoriu în laboratorul de informatică pentru activităţile practice.

În procesul de predare-învăţare, activitatea va fi orientată pe probleme: analiza unor situaţii practice (generale sau specifice unui anumit domeniu), identificarea fluxului informaţional, elaborarea unui model algoritmic de rezolvare, implementarea algoritmilor într-un limbaj de programare.

Exemplele utilizate la predare vor fi preponderent alese din aria curriculară ”Matematică şi ştiinţe ale naturii”, în colaborare cu profesorii de *Matematică*, *Fizică, Chimie* şi *Biologie*.

COMPETENŢE GENERALE

1. Identificarea datelor care intervin într-o problemă şi aplicarea algoritmilor fundamentali de prelucrare a acestora
2. Elaborarea algoritmilor de rezolvare a problemelor
3. Implementarea algoritmilor într-un limbaj de programare

VALORI ŞI ATITUDINI

- Exprimarea unui mod de gândire creativ, în structurarea şi rezolvarea problemelor
- Conştientizarea impactului social, economic şi moral al informaticii
- Formarea obişnuinţelor de a recurge la concepte şi metode informatice de tip algoritmic specifice în abordarea unei varietăţi de probleme.
- Manifestarea unor atitudini favorabile faţă de ştiinţă şi de cunoaştere în general
- Manifestarea iniţiativei şi disponibilităţii de a aborda sarcini variate

COMPETENŢE SPECIFICE ŞI CONŢINUTURI

1. Identificarea datelor care intervin într-o problemă şi aplicarea algoritmilor fundamentali de prelucrare a acestora

Competenţe specifice	Conţinuturi [1]
1.1. Transpunerea unei probleme din limbaj natural în limbaj de grafuri, folosind corect terminologia specifică 1.2. Analizarea unei probleme în scopul identificării datelor necesare şi alegerea modalităţilor adecvate de structurare a datelor care intervin într-o problemă 1.3. Descrierea unor algoritmi simpli de verificare a unor proprietăţi specifice grafurilor. 1.4. Descrierea algoritmilor fundamentali de prelucrare a grafurilor şi implementarea acestora într-un limbaj de programare 1.5. Descrierea operaţiilor specifice listelor simplu înlănţuite şi elaborarea unor subprograme care să implementeze aceste operaţii. 1.6. Descrierea operaţiilor specifice structurilor arborescente şi elaborarea unor subprograme care să implementeze aceste operaţii. 1.7. Analizarea în mod comparativ a avantajelor utilizării diferitelor metode de structurare a datelor necesare pentru rezolvarea unei probleme 1.8. Aplicarea în mod creativ a algoritmilor fundamentali în rezolvarea unor probleme concrete	**Grafuri neorientate şi grafuri orientate** • Terminologie (graf neorientat, graf orientat, lanţ, drum, ciclu, circuit, grad, graf parţial, subgraf, conexitate, tare conexitate, arbore, graf ponderat, arbore parţial, arbore parţial de cost minim) • Tipuri speciale de grafuri (graf complet, graf hamiltonian, graf eulerian, graf bipartit, graf turneu) • Reprezentarea grafurilor (matrice de adiacenţă, liste de adiacenţă, lista muchiilor, matricea costurilor) • Algoritmi de prelucrare a grafurilor - Parcurgerea grafurilor în lăţime şi în adâncime - Determinarea componentelor conexe ale unui graf neorientat - Determinarea componentelor tare conexe ale unui graf orientat - Determinarea matricei lanţurilor/drumurilor - Determinarea drumurilor de cost minim într-un graf (algoritmul lui Dijkstra, algoritmul Roy-Floyd) - Arbori parţiali de cost minim (algoritmul lui Kruskal sau algoritmul lui Prim) • Rezolvarea unor probleme cu caracter practic. **Structuri de date alocate dinamic (definiţii, utilitate)** • Liste simplu înlănţuite • Liste dublu înlănţuite • Liste circulare • Operaţii elementare pe liste înlănţuite (inserare element, ştergere element, parcurgere) **Structuri de date arborescente** • Arbori cu rădăcină (definiţie, proprietăţi, reprezentare cu referinţe ascendente, reprezentare cu referinţe descendente) • Arbori binari (definiţie, proprietăţi specifice; reprezentarea arborilor binari cu referinţe descendente; operaţii specifice) • Tipuri speciale de arbori binari - Arbore binar complet – definiţie, proprietăţi, reprezentare secvenţială - Arbore binar de căutare – definiţie, proprietăţi, operaţii specifice (inserare nod, ştergere nod, căutare element) - Heap-uri – definiţie, proprietăţi, operaţii specifice (inserare nod, extragerea nodului cu cheie maximă/minimă). • Rezolvarea unor probleme cu caracter practic.

[1] Conţinuturile sunt prezentate în tabele, grupate pe competenţe şi asocierea acestora este obligatorie. Este la decizia cadrului didactic/ a autorului de manual şcolar ordinea abordării conţinuturilor, cu respectarea logicii interne a domeniului.

2. Elaborarea algoritmilor de rezolvare a problemelor

Competenţe specifice	Conţinuturi
2.1 Analiza problemei în scopul identificării metodei de programare adecvate pentru rezolvarea problemei 2.2 Aplicarea creativă a metodelor de programare pentru rezolvarea unor probleme intradisciplinare sau interdisciplinare, sau a unor probleme cu aplicabilitate practică 2.3 Analizarea comparativă a eficienţei diferitelor metode de rezolvare a aceleiaşi probleme şi alegerea celui mai eficient algoritm de rezolvare a unei probleme	**Metode de programare** • Metoda de programare *Greedy* (descrierea generală a metodei, utilitate, aplicaţii). • Metoda de programare *Backtracking* (descrierea generală a metodei, utilitate, aplicaţii) • Metoda de programare *Divide et Impera* (descrierea generală a metodei, utilitate, aplicaţii) • Metoda programării dinamice (descrierea generală a metodei, utilitate, aplicaţii). **Analizarea eficienţei unui algoritm**

3. Implementarea algoritmilor într-un limbaj de programare

Competenţe specifice	Conţinuturi
3.1 Declararea corectă a unui pointer de date 3.2 Utilizarea corectă a pointerilor de date 3.3 Utilizarea subprogramelor predefinite de alocare şi eliberare dinamică a memoriei	**Tipuri specifice pentru adresarea zonei de memorie alocate unei variabile** (pointeri / referinţe). Declarare, operaţii specifice. **Alocarea dinamică a memoriei**. (operaţii şi mecanisme specifice)

SUGESTII METODOLOGICE

Predarea informaticii va fi orientată pe *rezolvarea de probleme*, utilizându-se preponderent metode activ-participative şi punându-se accent pe *analiza problemei*. Pentru buna desfăşurare a orelor şi aplicarea programei se sugerează următoarele activităţi de învăţare:

- discuţii despre activităţi cotidiene şi modelarea acestora în limbaj algoritmic;
- activităţi de dezvoltare a deprinderilor de organizare a informaţiei în diferite structuri de date;
- identificarea modalităţilor eficiente de reprezentare a datelor necesare pentru rezolvarea unei probleme
- descompunerea rezolvării unei probleme în subprobleme;
- prezentarea unor situaţii practice familiare elevilor care pot fi modelate în termenii teoriei grafurilor
- reprezentarea grafică a grafurilor, listelor, arborilor şi ilustrarea prin exemple reprezentate grafic a diferitelor noţiuni şi proprietăţi specifice
- demonstrarea modului de realizare a operaţiilor elementare specifice diferitelor structuri de date pe exemple reprezentate grafic.
- aplicarea algoritmilor fundamentali din teoria grafurilor pe exemple relevante
- adaptarea creativă a algoritmilor fundamentali de prelucrare a datelor pentru rezolvarea unei probleme
- identificarea unor situaţii în care alegerea unui algoritm prezintă avantaje în raport cu altul;
- exersarea creării şi aplicării programelor pentru rezolvarea unor probleme întâlnite de elevi în studiul altor discipline şcolare;
- evidenţierea greşelilor tipice în elaborarea algoritmilor;
- proiectarea/modelarea unor algoritmi şi implementarea acestora;
- implementarea structurilor de date alocate dinamic;
- testarea şi analizarea comportamentului programelor pentru diferite date de intrare;
- încurajarea discuţiilor purtate între elevi, exprimarea şi ascultarea părerilor fiecăruia.

Conţinuturile din prezenta programa vor fi susţinute prin rezolvarea unor probleme intradisciplinare sau interdisciplinare, respectiv probleme aplicabilitate practică în viaţa cotidiană.

Exemple de aplicaţii recomandate

I. Grafuri orientate şi grafuri neorientate

1. Algoritmi simpli de verificare a însuşirii terminologiei sau de verificare a unor proprietăţi specifice grafurilor (de exemplu, calcularea gradelor vârfurilor unui graf, verificarea faptului că o succesiune de vârfuri reprezintă lanţ, drum, ciclu sau circuit în graf, identificarea tuturor ciclurilor de lungime 3 într-un graf, verificarea proprietăţii de graf complet sau graf turneu, etc.)
2. Probleme practice, care solicită aplicarea creativă a algoritmilor din teoria grafurilor, cum ar fi:
   - Determinarea unei modalităţi de conectare a unor calculatoare în reţea astfel încât costurile de conectare să fie minime
   - Determinarea unui traseu de lungime minimă între două localităţi a căror poziţie pe hartă este specificată
   - Determinarea unei modalităţi de transmitere a unui mesaj într-o interreţea astfel încât numărul total de servere prin intermediul cărora este transmis mesajul să fie minim.
   - Determinarea structurii relaţionale a unui grup de persoane

II. Structuri de date alocate dinamic

1. Aplicaţii simple care să necesite implementarea operaţiilor elementare pe liste înlănţuite cum ar fi: crearea unei liste prin inserări succesive, astfel încât la fiecare pas lista să fie ordonată, inversarea legăturilor într-o listă simplu înlănţuită, numărarea elementelor dintr-o listă cu o anumită proprietate, etc.
2. Probleme mai complexe, în care elevii să identifice eficienţa utilizării listelor simplu înlănţuite, cum ar fi: reprezentarea grafurilor prin liste de adiacenţă alocate dinamic, reprezentarea polinoamelor rare în formă condensată şi implementarea operaţiilor specifice polinoamelor, etc.

III. Structuri de date arborescente

1. Aplicaţii simple care să necesite implementarea operaţiilor elementare pe structuri arborescente cum ar fi: parcurgerea unui arbore în scopul identificării tuturor nodurilor cu o anumită proprietate, determinarea înălţimii unui arbore, copierea unui arbore, etc.
2. Probleme mai complexe, în care elevii să identifice eficienţa utilizării structurilor de date arborescente, cum ar fi: realizarea eficientă a unui dicţionar, sortarea unei secvenţe de valori cu ajutorul *heap*-urilor (*heapsort*), optimizarea algoritmului lui *Kruskal* prin organizarea muchiilor ca *heap*.

IV. Metoda de programare *Greedy*

1. Problema rucsacului în variantă continuă
2. Determinarea arborelui parţial de cost minim (algoritmul lui Kruskal, algoritmul lui Prim)

V. Metoda de programare *Divide et Impera*

1. Sortarea eficientă a unei mulţimi de valori aplicând metoda *Divide et Impera* (sortarea rapidă, sortarea prin interclasare)
2. Căutarea eficientă a unui element într-o mulţime ordonată aplicând metoda *Divide et Impera* (căutarea binară)
3. Generarea unor modele fractale

VI. Metoda de programare *Backtracking*

1. Generarea permutărilor, combinărilor, aranjamentelor, funcţiilor surjective, partiţiilor unui număr, partiţiilor unei mulţimi
2. Generarea tuturor posibilităţilor de a ieşi dintr-un labirint
3. Generarea tuturor grafurilor parţiale ale unui graf
4. Determinarea tuturor ciclurilor hamiltoniene într-un graf

VII. Metoda programării dinamice

1. Determinarea unui subşir crescător de lungime maximă
2. Înmulţirea optimală a unui şir de matrice.
3. Problema rucsacului în varianta discretă
4. Algoritmul Roy-Floyd de determinare a drumurilor de cost minim între oricare două vârfuri ale grafului

VIII. Analiza comparativă a rezolvării unei probleme prin diferite metode de programare. De exemplu, problema determinării unui traseu de la vârful unui triunghi către baza acestuia, astfel încât suma elementelor care aparţin traseului să fie minimă; deplasările posibile sunt din elementul curent la unul dintre elementele situate sub el, în stânga sau în dreapta.