PROIECT DIDACTIC

LECŢIE PENTRU DOBÂNDIREA DE NOI CUNOŞTINŢE

Disciplina: INFORMATICĂ

Clasa: a XI-a, specializarea Matematică – Informatică

Profesor: .....

Unitatea de învăţare: Metode de programare

Tema: Metoda de programare **Divide et Impera**

Tipul lecţiei: însuşire de noi cunoştinţe

Locul desfăşurării: Laboratorul de informatică

Nivelul iniţial al clasei:

- elevii şi-au însuşit noţiunile teoretice legate de recursivitate;
- elevii utilizează şi operează corect cu funcţiile şi structurile de date alocate static;

Competenţă generală: Elaborarea algoritmilor de rezolvare a problemelor

Competenţe specifice:

C1. Analiza problemei în scopul identificării metodei de programare adecvate pentru rezolvarea problemei

C2. Aplicarea creativă a metodelor de programare pentru rezolvarea unor probleme intradisciplinare sau interdisciplinare, sau a unor probleme cu aplicabilitate practică

C3. Analizarea comparativă a eficienţei diferitelor metode de rezolvare a aceleiaşi probleme şi alegerea celui mai eficient algoritm de rezolvare a unei probleme

Obiectivul general al lecţiei: Aplicarea tehnicilor recursive în rezolvarea de probleme

Obiective operaţionale:

La sfârşitul lecţiei elevii vor fi capabili:

O1: să utilizeze corect funcţii implementate recursiv;

O2: să aplice cunoştinţele acumulate în rezolvarea de aplicaţii practice;

O3: să aleagă modlaităţi diferite de implementare ale rezolvării unei aceleiaşi probleme;

O4: să aplice metoda Divide et Impera în rezolvarea de probleme.

Obiective educaţionale:

Obiective cognitive:

- să utilizeze corect noţiunile teoretice însuşite pentru a rezolvarea aplicaţii;
- să compare între ele diferite metode rezolvare

Obiective afective:

- să argumenteze anumite situaţii create în etapele de dezvoltare a unei aplicaţii;
- să manifeste interes faţă de problemele propuse şi dorinţa de învăţare prin descoperirea proprie a adevărului ştiinţific;
- să aprecieze corect soluţiile oferite de ceilalţi colegi.

Obiective psihomotorii:

- să-şi dezvolte gândirea logică, flexibilă, creatoare;
- să-şi dezvolte atenţia concentrată şi spiritul de observaţie;
- să utilizeze corect noţiunile teoretice însuşite.

Strategii didactice:

- **Principii didactice:**

- principiul participării active
- principiul asigurării progresului gradat al performanţei
- principiul conexiunii inverse**.**

- **Metode de învăţământ:**

- metode de comunicare orală: expunere, conversaţie, problematizare, algoritmizare, modelare
- metode de acţiune: exerciţiul, învăţare prin descoperire, prin rezolvare de probleme
- problematizarea prin crearea situaţiilor problemă
- conversaţia de consolidare în toate etapele lecţiei

- **Forme de organizare:** frontală şi individuală
- **Forme de dirijare a învăţării:** dirijată de profesor sau independentă
- **Resurse materiale:**

- material bibliografic:

- Tudor Sorin – *Informatică. Manual pentru clasa a X-a. Varianta C++* - Editura L&S Soft, Bucureşti, 2006

- fişă de probleme

- **Metode de evaluare:**

- evaluare iniţială: întrebări adresate frontal clasei, aplicaţii practice
- set de aplicaţii

Momentele lecţiei:

- **Moment organizatoric:**
  - Pregătirea lecţiei:
    - întocmirea proiectului didactic
    - pregătirea setului de întrebări
    - pregătirea setului de aplicaţii
    - pregătirea temei
  - Organizarea şi pregătirea clasei: verificarea frecvenţei
  - Captarea atenţiei clasei:
    - anunţarea subiectului lecţiei;
    - anunţarea obiectivelor urmărite;
    - anunaţarea modului de desfăşurare a activităţii

- **Reactualizarea cunoştinţelor**

Se desfăşoară o scurtă conversaţie introductivă, reactualizarea cunoştinţelor făcându-se apoi treptat, pe măsură ce vor fi întâlnite diverse situaţii.

Evaluarea în etapa de reactualizare se realizează frontal.

- **Transmiterea noilor cunoştinţe**

Metoda **Divide et Impera** este o tehnică de programare care se poate aplica problemelor care se pot descompune în alte subprobleme ce pot fi tratate independent, sunt similare cu problema iniţială, sunt de dimensiuni mai mici şi a căror rezolvare este mai uşoară.

Pot fi identificate trei etape în rezolvarea unei probleme folosind Divide et Impera:

1. **Divide**, descompunera problemei în două (sau mai multe) subprobleme
2. **Impera**, rezolvarea subproblemelor
3. **Combină** rezultatele parţiale

Divide et impera este o tehnică ce admite atât implementare iterativă cât şi recursivă.

Vom utiliza implementarea recursivă a algoritmilor. Astfel, identificăm următoarele situații:

- Am ajuns la o problemă cu o dimensiune ce ne permite rezolvarea problemei;
- Impărțim problema în subprobleme cu scopul de a fi rezolvate ulterior.

1. Maximul dintr-un vector

Se citeşte un vector cu n componente, numere naturale. Se cere să se tipărească componenta cu valoarea maximă.

```
#include <iostream>
using namespace std;
int v[10],n;

int max(int i, int j)
{
 int a, b, m;
 if (i==j) return v[i];
 else
 {
  m = (i+j)/2;
  a = max(i, m);
  b = max(m+1, j);
  if  (a>b) return a;
  else return b;
 }
}

int main( )
```

```
{
 cin>>n;
 for (int i=1; i<=n; i++)
 {
  cin>>v[i];
 }
 cout<<max(1,n);
 return 0;
}
```

2. Căutare binară

Se consideră un vector ***v*** având ***n*** componente, numere întregi, ordonate crescător şi o valoare întreagă ***x***. Dacă valoarea ***x*** se găseşte printre componentele vectorului ***v*** să se afişeze indicele corespunzător, în cazcontrar să se afişeze mesajul „nu se gaseste”. Numărul de componente ***n***, cele ***n*** componente ale vectorului ***v*** şi valoarea lui ***x*** se citesc de la tastatură.

O parcurgere secvenţială în care ***x*** se compară pe rând cu toate cele ***n*** componente ale vectorului reperzintă o abordare simplistă şi ineficientă (nu ţine seama de faptul că vectorul este ordonat crescător), având complexitatea O(n).

O abordare folosind Divide et Impera:

- dacă ***x*** are valoarea componentei aflată pe poziţia de indice m=(s+d)/2, se afisează indicele şi se revine din apel (problema a fost rezolvată).
- în caz contrar, dacă nu au fost verificate toate componentele vectorului, atunci problema se descompune astfel:
    - ✓ dacă valoarea ***x*** este mai mică decât valoarea componentei testate (cea de pe poziţia din mijloc), înseamnă că ***x*** se poate afla numai pe una din poziţiile din partea stângă, cu indice între s şi m - 1, se reapelează funcţia cu aceşti parametri;
    - ✓ dacă valoarea ***x*** este mai mare decât valoarea componentei testate (cea de pe poziţia din mijloc), înseamnă că ***x*** se poate afla numai pe una din poziţiile

din partea dreaptă, cu indice între m+1 şi d, se reapelează funcţia cu aceşti parametri;

- dacă nu mai sunt alte componente de testat, înseamnă că ***x*** nu se găseşte printre componentele vectorului.

Întrucât vectorul este sortat, căutarea se va face în continuarea pasului curent, într-o problemă de dimensiune egală cu jumătatea celei curente. În consecință, complexitatea acestui algoritm va fi mult mai mică decât cea a unei căutări secvențiale, și anume $O(\log_2 n)$. Ex: pentru un $n=2^{10}$, căutarea se va finaliza după maxim 10 operații și nu 1024 ca în cazul celălalt!

```
#include <iostream>
using namespace std;
int v[100], n,x;

void cauta(int s, int d)
{
 int m = (s+d)/2;
 if (s<=d)
 {
  if (x==v[m])
     cout<<x<<" se gaseste pe pozitia : "<<m;
  else
    if  (x<v[m])
      cauta(s,m-1);
    else
      cauta(m+1,d);
 }
 else cout<<"nu se gaseste";
}
int main( )
{
```

```
   cin>>n;
   for (int i=1; i<=n; i++)
   {
     cin>>v[i];
   }
   cin>>x;
   cauta (1,n);
   return 0;
}
```

2. Căutare binară, #508 – sursa pbinfo.ro

Se dă un vector x cu n elemente numere naturale, ordonate crescător, și un vector y cu m elemente, de asemenea numere naturale. Verificați pentru fiecare element al vectorului y dacă apare în x.

```
#include <iostream>
using namespace std;

int caut(int x[25001],int s,int d,int a)
{
   int mij;
   if(s<=d)
   {
      mij=(s+d)/2;
      if(x[mij]==a)
         return 1;
      if(a<x[mij])
         return caut(x,s,mij-1,a);
      else
         return caut(x,mij+1,d,a);
   }
   else
```

```
    return 0;
}
int main()
{
   int x[25001],n,m,i,a;
   cin>>n;
   for(i=1;i<=n;i++)
      cin>>x[i];
   cin>>m;
   for(i=1;i<=m;i++)
   {
      cin>>a;
      if(caut(x,1,n,a))
         cout<<1<<" ";
      else
         cout<<0<<" ";
   }
   return 0;
}
```

Activitatea profesorului

Profesorul va propune elevilor recapitularea cunoştinţelor cu ajutorul întrebărilor.

Etapa de comunicare a noilor cunoştinţe va consta într-o combinaţie de aplicaţii practice şi scurte definiţii.

Pentru fixarea cunoştinţelor, profesorul va propune elevilor un set de probleme ce vor fi rezolvate cu Divide et Impera.

Activitatea elevilor

Elevii ar trebui să răspundă corect întrebărilor adresate lor de către profesor (oral sau la tablă, ori pe calculator). Ei vor îndeplini sarcinile de lucru. Vor lua notiţe.

- **Intensificarea reţinerii şi asigurarea transferului de informaţii.**

Se realizează cu aplicaţii pentru verificarea utilizării corecte a noţiunilor învăţate.

- **Dirijarea învăţării pentru obţinerea performanţei (tema de lucru – conţinutul unei fişe de probleme)**

- **Tema pentru acasă**

Tema pentru acasă va fi constituită din problemele care rămân nerezolvate din fişa de lucru.

FIŞĂ DE LUCRU
DIVIDE ET IMPERA

Utilizând metoda **Divide et Impera** rezolvaţi următoarele probleme:

1. **Produsul a n numere**

Din fişierul de intrare **şir.in**, se citesc:

- ◊ de pe prima linie un număr natural **n**, reprezentând numărul de componente ale unui şir **a**;
- ◊ de pe a doua line, separate prin spaţiu, cele **n** componente ale şirului **a.**

Cerinţă:

Să se afişeze produsul componentelor şirului citit.

2. **Verificarea ordinii componentelor unui vector**

Din fişierul de intrare **vector.in**, se citesc:

- ◊ de pe prima linie un număr natural **n**, reprezentând numărul de elemente ale vectorului **a**;
- ◊ de pe a doua line, separate prin spaţiu, cele **n** elemente ale vectorului **a.**

Cerinţă:

Dacă elementele vectorului **a** sunt ordonate crescător afişaţi mesajul „ok”, în caz contrar afişaţi mesajul „Vectorul nu este ordonat”.

3. Determinarea celui mai mare divizor comun al elementelor unui şir

Din fişierul de intrare **cmmdc.in**, se citesc:

- ◊ de pe prima linie un număr natural **n**, reprezentând numărul de componente ale unui şir **a**;
- ◊ de pe a doua line, separate prin spaţiu, cele **n** componente ale şirului **a,** numere naturale**.**

Cerinţă:

Să se afişeze cel mai mare divizor comul ale componentelor şirului citit.

4. Verificarea primalităţii unui număr

Se consideră un număr natural **n.** Afişaţi un mesaj dacă acesta este prim.

5. Suma divizorilor proprii ai unui număr natural

Se dă un număr natural **n**. Afişaţi suma divizorilor proprii ai acestuia utilizând metoda Divide et Impera.

6. Numere pozitive

Se dă un şir de numere întregi. Verificaţi dacă toate elementele sunt pozitive.