PROIECT DIDACTIC

Liceul Teoretic „Ion Mihalache" Topoloveni

Disciplina: Informatica

Clasa: a IX-a

Profesor: POPA IONELA ELIZA

Unitatea de învăţare: Elaborarea algoritmilor de rezolvare a problemelor

Tema: Structuri de control. Structura liniara.

Tipul lecţiei: Dobândire de noi cunoştinţe

Obiectiv cadru:Realizarea de aplicaţii utilizând algoritmi specifici

Obiective de referinţă: - Identificarea datelor, a relaţiilor dintre acestea

- Formarea deprinderilor de a transpune în limbaj algoritmic o problemă dată

Obiective educaţionale:

Obiective cognitive:

- Să definească corect noţiunile teoretice însuşite
- Să urmărească etapele de realizare a unei aplicaţii
- Să analizeze modul de funcţionare a structurii liniare
- Să elaboreze algoritmi respectând principiile programării structurate
- Să aplice corect noţiunile însuşite în aplicaţii concrete

Obiective afective:

- Să recunoască necesitatea utilizării structurilor liniare
- Sa argumenteze corect alegerea unei variante
- Să manifeste interes faţă de problemele puse şi dorinţa de învăţare prin descoperire proprie
- Să studieze individual şi în echipă, în colaborare şi în competiţie
- Să aprecieze corect soluţiile oferite de ceilalţi elevi
- Să se autoevalueze în raport cu obiectivele şi cu clasa

Obiective psihomotorii:

- Să dezvolte gândirea algoritmică, logică, flexibilă, creatoare
- Să-şi dezvolte atenţia concentrată şi spiritul de observaţie

- Să utilizeze corect noţiunile teoretice însuşite
- Să conceapă programe pentru aplicaţiile propuse
- Să compună probleme care necesită structuri liniare

Obiective operaţionale:

- Să reproducă şi să explice forma generală şi interpretarea pentru structura liniara
- Să descrie diagrama de sintaxă pentru structura liniara
- Să justifice necesitatea utilizării structurilor liniare
- Să înţeleagă exemplele date
- Să conceapă noi aplicaţii care necesită utilizarea structurii liniare.

Strategii didactice:

Principii didactice:

– principiul participării şi învăţării active
– principiul asigurării progresului gradat al performanţei
– principiul conexiunii inverse

Metode de învăţământ:

– metode de comunicare orală: expunere, conversaţie, problematizare
– metode activ participative: exerciţiul, învăţare prin descoperire

Procedee de instruire:

– explicaţia în etapa de comunicare
– învăţarea prin descoperire
– problematizarea prin crearea situaţiilor problemă
– conversaţia de consolidare în etapa de fixare a cunoştinţelor

Forme de organizare: frontală şi individuală

Forme de dirijare a învăţării: dirijată de profesor sau independentă

Resurse materiale:material bibliografic – manualul (Mariana Miloşescu, Informatică - Manual pentru clasa a IX-a, profilul real, specializarea matematică-informatică, ştiinţe ale naturii, Editura didactică şi pedagogică, 2004)

Metode de evaluare:

– evaluare iniţială: întrebări orale

– set de aplicaţii

Desfăşurarea lecţiei:

1. **Moment organizatoric**

o Pregătirea lecţiei: - întocmirea proiectului didactic;
- pregătirea setului de întrebări;
- pregătirea setului de aplicaţii;
- pregătirea temei;

o Organizarea şi pregătirea clasei: - verificarea frecvenţei;

o Captarea atenţiei clasei:- verificarea temei de casă
- anunţarea subiectului pentru tema respectivă;
- anunţarea obiectivelor urmărite;
- anunţarea modului de desfăşurare a activităţii;

2. Reactualizarea cunoştinţelor

Se realizează un set de întrebări pentru reactualizarea cunoştinţelor teoretice, ca mai jos :

Întrebare	**Răspuns aşteptat**
Ce este algoritmul	Mulţime ordonată şi finită de paşi executabili prin care se defineşte fără echivoc modul de rezolvare a unei clase de probleme
Caracteristicile unui algoritm	Orice algoritm trebuie să îndeplinească trei caracteristici fundamentale: 1. generalitate 2. claritate (determinare) 3. finitudinea
Cum se reprezintă	Schemă logică Limbaj pseudocod Limbaj de programare
Ce este limbajul pseudocod	Limbaj artificial, apropiat de limbajul natural, care foloseşte pentru reprezentarea algoritmilor cuvinte cheie. Pseudocod are doua tipuri de propoziţii: propoziţii standard, care vor fi prezentate fiecare cu sintaxa si semnificaţia (semantica) ei şi propoziţii nestandard (sunt texte care descriu părţi ale

Întrebare	Răspuns aşteptat
	algoritmului încă incomplet elaborate)
Ce este structura de control	Entitate din cadrul algoritmului prin care se descrie modul în care paşii algoritmului îşi predau controlul unul altuia
Tipuri de structuri de control	- liniară (secvenţială) - alternativă - repetitivă
Programarea structurată	Programarea structurată este un stil de programare care cere respectarea unei discipline de programare şi folosirea riguroasa a câtorva structuri de control. Ca rezultat se va ajunge la un algoritm uşor de urmărit, clar şi corect. Bohm si Jacopini au demonstrat ca orice algoritm poate fi compus din numai trei structuri de control: - structura secvenţială; - structura alternativă; - structura repetitivă.
Structura liniară	Paşii se execută în mod necondiţionat, o singură dată, în ordinea în care au fost scrişi.

– metode : conversaţia de fixare
– evaluare frontală

3. Comunicarea noilor cunoştinţe

STRUCTURI DE CONTROL.
STRUCTURA SECVENTIALA

Algoritmul proiectat pentru rezolvarea unei anumite probleme trebuie implementat intr-un limbaj de programare; prelucrarea datelor se realizeaza cu ajutorul instructiunilor. Instructiunea descrie un proces de prelucrare pe care un calculator il poate executa. O instructiune este o constructie valida (care respecta sintaxa limbajului) urmata de ; . Ordinea in care se executa instructiunile unui program defineste asa-numita structura de control a programului.

Limbajele moderne sunt alcatuite pe principiile programarii structurate. Conform lui C. Bohm si G. Jacobini, orice algoritm poate fi realizat prin combinarea a trei structuri fundamentale:

- structura secventiala;
- structura alternativa (de decizie, de selectie);
- structura repetitiva (ciclica).

IMPLEMENTAREA STRUCTURII SECVENTIALE

Structura secventiala este o insiruire de secvente de prelucrare (instructiuni), plasate una dupa alta, in ordinea in care se doreste executia acestora.

Reprezentarea structurii secventiale cu ajutorul pseudocodului:

```
instr1;
instr2;
. . . . .
```

Implementarea structurii secventiale se realizeaza cu ajutorul instructiunilor:

Instructiunea vida

Sintaxa: ;

Instructiunea vida nu are nici un efect. Se utilizeaza in constructii in care se cere prezenta unei instructiuni, dar nu se executa nimic (de obicei, in instructiunile repetitive).

Exemple:

```
int a;
. . . . . .
int j;
  ;
for (;;)
{
. . . .
}
```

Instructiunea expresie

Sintaxa:

expresie;

sau:

apel_functie;

Exemple:

```
int b, a=9;
double c;
```

b=a+9;

cout<<a;

c=sqrt(a);

clrcsr();//apelul functiei predefinite care sterge ecranul; prototipul in headerul conio.h

Instructiunea compusa (instructiunea bloc)

Sintaxa: {

declaratii;

instr1;

instr2;

. . . .

}

Intr-un bloc se pot declara si variabile care pot fi accesate doar in corpul blocului.

Instructiunea bloc este utilizata in locurile in care este necesara prezenta unei singure instructiuni, insa procesul de calcul este mai complex, deci trebuie descris in mai multe secvente.

******Structura liniară*** (numită şi ***secvențială***) este alcatuită din urmatoarele instrucţiuni:

- **comentarii**
- **declararea variabilelor**
- **instrucţiunea de citire**
- **instrucţiunea de scriere**
- **instrucţiunea de atribuire**
- **instrucţiunea compusă (sau blocul de instrucţiuni)**

Comentarii

Putem adaugă comentarii în cadrul algoritmului pentru a descrie operaţiile efectuate sau a da indicaţii necesare la implementare. Adeseori, când se lucrează în echipă, comentariile sunt foarte necesare.

Sunt mai multe variante în care putem să scriem comentarii. În general, fiecare programator va folosi ceea ce crede că este mai uşor de înteles sau mai rapid de scris.

În algoritmii prezentaţi în acest modul, comentariile încep cu semnul „//” şi se vor scrie la începutul fiecarui rând de comentariu. Nu este necesară scrierea semnului şi la sfârşitul rândului.

Prezentam ca exemplu două tipuri de comentarii

Exemplu 1

// Acesta este un comentariu

// Fiecare rând de comentariu începe cu semnul „//”

Exemplu 2

/* Acest comentariu se poate scrie
pe mai multe linii, iar sfarsitul comentariului
se face cu */

Declararea variabilelor

La începutul algoritmului trebuie să se precizeze datele de intrare, datele de ieşire, datele de manevră şi tipul lor. O variabilă nu se poate declara de mai multe ori în cadrul aceluiaşi algoritm.

variabila tip

//Exemple

a întreg
b real
x caracter

Instrucţiunea de citire

Efectul instrucţiunii este de a da valori (de la tastatură sau dintr-un fişier) variabilelor de intrare cu care lucrăm.

Citeşte expresie1, expresie2, expresie3

//Exemplu

Citeste a, b, x

Instrucţiunea de scriere

Instrucţiunea afişează pe ecran sau în fişier valorile variabilelor.

Scrie expresie1, expresie2, expresie3

//Exemplu

Scrie a, b

Instrucţiunea de atribuire

Efectul instrucţiunii este acela de a atribui valoarea din dreapta săgeţii variabilei specificată în stanga. În cazul în care în dreapta avem o expresie, aceasta se va evalua şi apoi valoarea va fi atribuită variabilei din stanga.

```
variabilă ← expresie

//Exemplu:
a ← 56
b ← a-2*a
c ← c+1
```

Ultima atribuire are un sens deosebit, adică variabila c va lua valoarea avută la pasul anterior al algoritmului marită cu 1.

Blocul de instrucţiuni

Este folosit pentru a efectua mai multe instrucţiuni, în ordinea în care sunt scrise. Sunt mai multe variante de marcare a începutului şi sfarşitului de bloc de instrucţiuni. Mai jos prezentăm două dintre ele, urmând ca pe parcursul modulului să folosim varianta cu paranteze.

```
//Exemplu 1
  | instructiune1
  | instructiune2
  | instructiune3
  |_■

//Exemplu 2
  { instructiune1
    instructiune2
    instructiune3
  }
```

APLICAŢII: Prezentăm în continuare doi algoritmi liniari importanţi:

Interschimbarea a două valori (numită şi regula celor trei pahare)

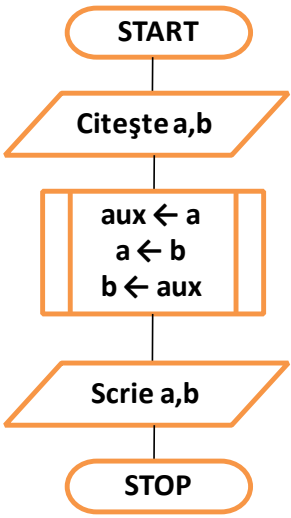
Fie două variabile întregi a şi b. Valorile lor se citesc de la tastatură. Să se interschimbe valorile celor două variabile apoi să se afişeze noile valori, pe acelaşi rând cu un spaţiu între ele.

Exemplu: dacă pentru variabilele a şi b se citesc valorile 5 şi 8, se va afişa: 8 5

```
a, b întregi //date de intrare
aux întreg //date de manevră
citeşte a, b
aux ← a
a ← b
b ← aux
scrie a, b
```

Explicarea algoritmului: Pentru a interschimba valorile, se foloseşte o variabilă auxiliară, care preia valoarea lui a, apoi a ia valoarea lui b, urmând ca în final b să ia valoarea lui aux, adică valoarea lui a avută iniţial. Algoritmul de interschimbare se mai numeşte şi "**Regula celor trei pahare**", deoarece este necesară o a treia variabilă pentru a face interschimbarea.

Algoritmul reprezentat cu ajutorul schemei logice este urmatorul:



Acest algoritm este întâlnit în algoritmi precum sortarea numerelor.

Cifrele unui număr

Fie x un număr întreg format din exact 5 cifre. Să se afişeze cifra unităţilor şi cea a sutelor, pe acelaşi rând, cu un spaţiu între ele.

Exemplu: dacă pentru x se citeşte valoarea **12345** se va afişa **5 3**.

```
x întreg //date de intrare
c1,c2 întregi //date de manevră
citeşte x
//reţin cifra unităţilor în c1
c1 ← x % 10
x ← x/100 //elimin cifra unităţilor şi a zecilor
//reţin cifra sutelor în c2
c2 ← x % 10
scrie c1, c2
```

Explicarea algoritmului: Pentru a obţine cifrele unui număr trebuie să efectuăm împărţiri la 10. Am arătat că operatorul „%” returnează restul împărţirii. În cazul în care un număr se împarte la 10, atunci restul este chiar ultima cifră, iar câtul împărţirii este numărul fară ultima cifra. În cazul împărţirii la 100 restul returnează ultimele 2 cifre, iar câtul este numărul fară ultimele 2 cifre. Pentru a afişa cifra sutelor este suficient să eliminăm ultimele 2 cifre (prin împărţire la 100) şi să afişăm ultima cifră a numărului nou obţinut.

-Utilizare :

-Sintaxa

-Funcţionare

-**Reprezentarea în schemă logică**:

-Exemplu

4. Asigurarea feedback-ului şi evaluarea performanţei

Aplicaţie

Se citesc doua numere a si b reale. Să se determine: suma, diferenta, produsul dintre a si b, restul impartirii lui a la b.

Realizati algoritmul in pseudocod si schema logica.

5. Fixarea cunoştinţelor

Se discută rezultatele obţinute – faptul că trebuie şi cum trebuie folosită structura liniara.

- se semnalează şi se corectează eventualele erori apărute;
- se evidenţiază şi se notează elevii ce au răspuns.

6. Tema pentru acasă

Probleme din manual.