PROIECT DIDACTIC

DATA: 19.05.2024
TIMP ALOCAT: 50 min.
CLASA: a X-a
LOCUL DESFĂŞURĂRII: sala de clasă
PROFESOR: Costache Rebeca
DISCIPLINA: Informatică
UNITATEA DE ÎNVĂŢARE: Subprograme
TEMA: Transmiterea parametrilor prin valoarea în limbajul C++
TIPUL LECŢIEI: Lecţie de predare-învăţare
SCOPUL LECŢIEI: însuşirea noţiunilor de bază despre transmiterea parametrilor prin valoare în cadrul subprogramelor C++

COMPETENŢE GENERALE

- identificarea datelor care intervin într-o problemă şi aplicarea algoritmilor fundamentali de prelucrare a acestora;
- elaborarea algoritmilor de rezolvare a problemelor
- implementarea algoritmilor într-un limbaj de programare

COMPETENŢE SPECIFICE

- Descompunerea problemelor în subprobleme şi identificarea modului de transmitere a informaţiilor între acestea
- Construirea unor subprograme pentru rezolvarea subproblemelor unei probleme
- Identificarea mecanismului corect de transmitere a parametrilor între subprogramele definite

OBIECTIVE OPERAŢIONALE:

Elevii trebuie să fie capabili:

- să reproducă şi să explice noţiunile de bază despre subprograme
- să evalueze corect evoluţia parametrilor transmişi prin valoare
- să rezolve aplicaţii ce presupun utilizarea noţiunilor dobândite

OBIECTIVE EDUCAŢIONALE

OBIECTIVE COGNITIVE:

- să definească corect noţiunile teoretice însuşite;
- să aplice corect noţiunile însuşite în aplicaţii concrete.

OBIECTIVE AFECTIVE:

- să argumenteze anumite situaţii create în etapele de rezolvare a unei aplicaţii;
- să manifeste interes faţă de problemele puse şi dorinţa de învăţare prin descoperire proprie a adevărului ştiinţific;
- să studieze individual şi în echipă, în colaborare şi în competiţie, cunoscând scopul învăţării temei date;
- să aprecieze corect soluţiile oferite de ceilalţi colegi.

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.

NIVELUL INIŢIAL AL CLASEI:

Colectiv eterogen

STRATEGII DIDACTICE

- **PRINCIPII DIDACTICE:**
  - principiul participării şi învăţării active;
  - principiul asigurării progresului gradat al performanţei;
  - principiul conexiunii inverse.
- **METODE DE ÎNVĂŢARE:**
  - metode de comunicare orală: conversaţia, explicaţia;
  - metode activ participative: învăţarea prin descoperire, problematizarea, exerciţiul.

- **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 prin materiale didactice;
  - independentă.
- **RESURSE MATERIALE:**
  - materiale bibliografice (manualul, culegeri, prezentare PowerPoint);
  - fişe de lucru;
  - set de aplicaţii;
- **METODE DE EVALUARE:**
  - observarea şi aprecierea verbală;
  - chestionare orală;
  - set de aplicaţii.
- **DESFĂŞURAREA ACTIVITĂŢII:**

A. Moment organizatoric :

- **Pregătirea lecţiei**
  - întocmirea proiectului didactic;
  - pregătirea setului de aplicaţii.
- **Organizarea şi pregătirea clasei (2 min.):**
  - verificarea prezenţei;
  - verificarea temei prin sondaj;
  - verificarea existenţei resurselor materiale.
- **Captarea atenţiei clasei (2 min.):**
  - anunţarea subiectului pentru tema respectivă;
  - anunţarea obiectivelor urmărite;
  - anunţarea modului de desfăşurare a activităţii

B. Comunicarea noilor cunoştinţe (30 min.):

Transmiterea prin valoare

Se utilizează atunci când suntem interesaţi ca subprogramul să lucreze cu acea valoare, dar să nu poată modifica parametrul efectiv corespunzător din blocul apelator.

Se pot transmite prin valoare:

1. **Valorile reţinute de variabile**. În acest caz parametrii efectivi trebuie să fie numele variabilelor.

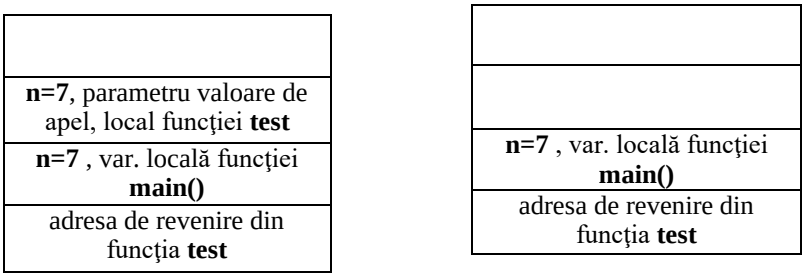
Exemplu:

```
#include<iostream.h>
void test(int n)
{
        n++;
        cout<<n<<endl;  // tipăreşte n=8
}
void main()
{
        int n=7;
        test(n);
        cout<<n<<endl;  // tipăreşte n=7
}
```

Parametrul **n** este transmis prin valoare. În funcţia **main()** acest parametru este iniţializat cu valoarea 7. Când apelăm funcţia **test()**, se rezervă spaţiu pe stivă, spaţiu care are numele parametrului formal (în acest caz, tot **n**) şi care este iniţializat cu valoarea memorată de variabila **n** a programului principal. Altfel spus, pe stivă se **copie** valoarea parametrului efectiv de apel. În funcţie, variabila **n** (care este

locală acestei funcţii) este incrementată şi devine 8, valoare care va fi tipărită. La ieşirea din funcţie, variabila **n** din stivă se pierde, adică nu mai are spaţiu alocat, prin urmare valoarea 8 este pierdută. În **main()** se tipăreşte valoarea variabilei **n** (locală acesteia) care are valoarea 7.

Pentru exemplul anterior, conţinutul stivei, în momentul apelului şi după execuţia funcţiei **test()**, este următorul:



după execuţia funcţiei test

Se observă că, în momentul apelului funcţiei **test()**, pe stivă sunt alocate două variabile cu acelaşi nume **n**. Prima variabilă este variabila locală funcţiei **main()** care se salvează pe stivă în momentul apelului pentru a putea reface contextul funcţiei **main()** după încheierea apelului. A doua variabilă este parametrul formal tip valoare **n**, vizibil numai în funcţia **test()** şi iniţializat în momentul apelului cu valoarea 7. Indiferent ce valori primeşte acest **n** în corpul funcţiei **test()**, după încheierea execuţiei acestei funcţii, spaţiul său este de alocat din stivă, adică variabila respectivă este distrusă. Din acest motiv, după execuţia funcţiei **test()**, conţinutul stivei este cel din dreapta. Se reface contextul din care s-a lansat apelul funcţiei **test()**, adică se recuperează din stivă valoarea variabilei locale **n=7** şi adresa de revenire, adică adresa instrucţiunii **cout**.

1. **Expresii**. În acest caz, parametrii efectivi sunt expresii, care pot conţine şi funcţii şi care mai întâi se evaluează. Exemplu:

```
#include<iostream.h>
#include<math.h>
void test(int n)
{
        cout<<n<<endl;
}
void main()
{
        test(5);              // se va tipări 5
        test(7+(int)sqrt(45)); // se va tipări 13
}
```

În funcţie se creează o variabilă numită **n**, reţinută pe stivă, care la primul apel va primi valoarea 5 şi la al doilea apel valoarea 13. La ieşirea din funcţie conţinutul acestei variabile se pierde.

Transmiterea parametrilor prin valoare se utilizează când nu dorim ca subprogramul apelat să poată modifica parametrii efectivi de apel. Acesta este modul implicit de transmitere a parametrilor în limbajul C. Dacă nu ar exista decât transmiterea prin valoare, ar fi imposibil să modificăm valoarea anumitor valori care sunt declarate în blocul apelator.

C. Fixarea noilor cunoştinţe şi realizarea feed-back-ului (15 min.):

1. Precizaţi care dintre următoarele linii de program reprezintă corect, din punct de vedere sintactic, lista de parametri şi valoarea returnată de o funcţie cu numele **test**, dacă cerem ca parametrii formali să fie două variabile de tip întreg, **x** şi **y**, şi o variabilă de tip real **z**, şi să returneze un rezultat întreg.
   a) int test(int x,y,float z)
   b) int test(int x,y;float z)
   c) int test(int x,int y,float z)
   d) test(int x;int y;float z) int
   e) test(int x,int y,float z) int
2. Care dintre afirmaţiile de mai jos sunt adevărate?
   a) la apelul unei funcţii, se produce înlocuirea parametrilor formali cu parametrii actuali
   b) tipul parametrilor daţi la apelul unei funcţii trebuie să coincidă sau să fie compatibil cu tipul celor definiţi în antetul funcţiei
   c) la apelul unei funcţii, se salvează pe stivă adresa de revenire, precum şi variabilele locale şi parametrii modulului apelat
   d) orice funcţie trebuie să aibă întotdeauna cel puţin un parametru
   e) nici una dintre afirmaţiile de mai sus
3. Considerăm o funcţie demo, de tip void pentru care se definesc ca parametri trei variabile întregi. Cum realizăm apelul funcţiei, astfel încât la apel să dăm ca parametri variabilele întregi a,b şi c?
   a) demo(int a,int b,int c);
   b) demo(int a;int b;int c);
   c) demo(a,b,c);
   d) demo(a;b;c);
   e) demo(int a,b,c);

4. Scrieţi o funcţie D care primeşte ca parametru un număr întreg a şi returnează valoarea lui a+2.
   a) int D(int a);{D=(a+2);}
   b) int D(int a){D=a+2;}
   c) int D(int a);{return(a+2);}
   d) int D(int a){return a+2;}
   e) nici una dintre variantele anterioare
5. Care dintre afirmaţiile de mai jos sunt adevărate?
   a) parametrii definiţi în antetul unei funcţii se numesc actuali, iar cei care apar la apelul funcţiei se numesc formali
   b) valoarea returnată de către o funcţie poate fi transmisă ca parametru altei funcţii
   c) variabilele de tip tablou nu se pot transmite ca parametri funcţiilor
   d) variabilele globale sunt cunoscute pe tot parcursul programului în care au fost declarate în toate modulele care urmează declaraţiei
   e) corpul unei funcţii trebuie cuprins între "{" şi "}", numai dacă este alcătuit din cel puţin două instrucţiuni distincte
6. Avantajele utilizării funcţiilor într-un program sunt:
   a) se poate obţine o economisire a spaţiului de memorie rezervat variabilelor folosite în cadrul programului
   b) viteză mai mare în execuţia programului
   c) posibilitatea de a executa de mai multe ori instrucţiunile cuprinse într-o funcţie
   d) un program care conţine funcţii poate fi urmărit şi corectat mai uşor
   e) nici unul dintre avantajele de mai sus

7. Scrieţi o funcţie cu numele divizori care primeşte prin parametrul n un număr natural nenul cu maxim 9 cifre şi returnează numărul divizorilor proprii ai numărului n.

8. Deduceţi şirul de valori care se afişează în urma execuţiei programului de mai jos:

```
#include<iostream.h>
int i,j,k;
int test(int x,int y)
```

```
{
        return (x-y);
}
void calcul(int p,int q)
{
        int u,v;
        u=p-i; v=q+j;
        i=test(u,q);
        j=test(v,p);
}
void main()
{
        i=2; j=3;
        calcul(i,j);
        cout<<i<<" "<<j<<endl;
        calcul(j,i);
        cout<<i<<" "<<j<<endl;
}
```

a) 2 3 2 3 b)3 2 3 2 c)2 3 –3 4 d)–3 4 4 –3 e) –3 4 10 –3

9. Fie programul:

```
#include<iostream.h>
int j,x;
int test(int,int);
void main()
{
        x=2;
        cout<<test(x,x+3);//(1)
        cout<<x;        //(2)
}
int test(int a,int b)
{
        int x=a+b;
        for(j=1;j<=x;j++)
          if(j>a && j<b) return j;
        return 0;
}
```

Precizaţi care dintre afirmaţiile de mai jos sunt adevărate:

a) instrucţiunea (1) afişează valoarea 3
b) instrucţiunea (2) afişează valoarea 7
c) funcţia **test** returnează 0, indiferent care ar fi parametrii daţi la apel
d) corpul funcţiei **test** este eronat, deoarece conţine două instrucţiuni return
e) programul este eronat, deoarece variabila **x** a fost declarată de două ori

10. Determinaţi valorile pe care le afişează programul de mai jos:

```
#include<iostream.h>
int x,y;
int T(int m,int n)
{
        m=n+x; n+=1;
        return(n+y+m);
}


void main()
{
        y=10; x=12;
        cout<<T(x,y);
        cout<<x<<","<<y;
}
```

a) 43,22,10 b) 43,12,10 c) 47,10,12 d) 44,22,11 e) 44,12,11

D. Temă pentru acasă (1 min.)

Aplicaţii: - exerciţiile 1, 2, 3, 5, 6 din manual, pagina 165.