**PROIECT DE LECȚIE**

**Clasa:** a X-a

**Student:**

**Disciplina:** INFORMATICĂ

**Unitatea de învățare**: **SUBPROGRAME Titlul lecției:** STRUCTURA FUNCȚIILOR **Tipul lecției:** Formarea de priceperi şi deprinderi **Durata:** 50 min

**Locul de desfăşurare*:*** Sala de clasă

**Competențe specifice:**

2.1. Utilizarea corectă a subprogramelor predefinite şi a celor definite de utilizator 2.2. Construirea unor subprograme pentru rezolvarea subproblemelor unei probleme

**Obiective operaționale:**

• *cognitive*

o să cunoască alcătuirea unei funcții;

o să cunoască structura antetului unei funcții;

o să cunoască atributele unei variabile;

o să cunoască variabilele globale şi locale, precum şi atributele lor;

o să construiască subprograme pentru rezolvarea subproblemelor unei probleme.

• *afective*

o să se implice afectiv cu plăcere şi interes la toate etapele lecției;

o să reacționeze pozitiv la aprecierile sau criticile profesorului;

• *psiho-motorii*

o să utilizeze corect noțiunile teoretice dobândite.

**Strategii didactice :**

• *principii didactice*

o principiul participării active a elevilor

o principiul conexiunii inverse (feedback-ului)

• *metode:*

o conversația euristică, problematizarea, explicația, exemplificarea, exercițiul, observația, descoperirea dirijată, conversația de consolidare

• *forme de organizare a activității instructive:* frontală, individuală şi pe grupe câte 2 elevi

• *forme de dirijare a activității:* dirijată de profesor şi independentă

• *resurse materiale:* tabla, creta, fişe de lucru.

**Metode de evaluare:** evaluare continuă pe parcursul lecției

**Bibliografie:**

[1] Tudor Sorin, Informatică: curs pentru clasele a IX-a şi aX-a. Editura L&S INFO-MAT, Bucureşti – 2008

**DESFĂŞURAREA LECȚIEI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
| **1. Moment organizatoric.** | Asigurarea condițiilor optime pentru desfăşurarea lecției (curățenie, lumină, ținută…). Verificarea prezentei elevilor. | conversația | în grup |
| **2. Anunțarea temei şi a obiectivelor** | **STRUCTURA FUNCȚIILOR** | conversația | în grup |

Pagina 1 din 7

|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
| **3. Captarea atenției şi verificare cunoştințelor** | Scrieți o funcție care calculează suma a două numere întregi.  **1.**  **2.**  Scrieți o funcție care calculează suma primelor n numere naturale. |  |  |
| **4. Prezentarea conținutului şi dirijarea învățării** | O **funcție** este alcătuită din **antet** şi o instrucțiune **compusă**.  **Antetul** conține informații importante necesare compilatorului cum ar fi numele funcției, lista parametrilor formali, tipul rezultatului.  Structura antetului este:  tip nume (lista paramentrilor formali) Lista paramentrilor formali este de forma:  parametru1, parametru2, ... parametrun Fiecare parametru are forma: tip nume  Există posibilitatea ca lista parametrilor formali să fie vidă.  **Instrucțiunea compusă** cuprinde declarațiile variabilelor locale şi instrucțiunile propriu – zise. **Observație:** Poate fi tip al unei funcții orice tip de dată cu excepția masivelor.  **Exemple de antete**:  int suma (int a, int b)  funcția se numeşte suma, returnează un rezultat de tip int şi are doi parametrii formali de tip int, numiți a şi b.  void t (int n, float v[20])  funcția se numeşte t, este de tip void (nu returnează rezultat prin nume), are doi parametrii formali, primul numit n, de tip int, al doilea numit v şi este un vector cu 20 de componente de tip float.  float f (float a, float b, float v[20])  funcția se numeşte f, returnează un rezultat de tip float şi are doi parametrii formali de tip float, numiți a şi b, iar al treilea parametru formal este numit v şi este un vector cu 20 de componente de tip float.  O funcție returnează rezultatul la întâlnirea instrucțiunii return, care are forma: return expresie;  **Observație:** Tipul expresiei trebuie să coincidă cu tipul funcției.  La întâlnirea instrucțiunii retrun, după atribuirea valorii, execuția funcției se încheie şi se revine la funcția care a apelat-o. În absența instrucțiunii return, execuția funcției se încheie după execuția ultimei instrucțiuni. În acest caz nu se întoarce nicio valoare.  O funcție poate fi apelată de sine stătător (prin nume şi lista parametrilor efectivi), dar poate fi inclusă şi în cadrul expresiilor, caz în care, la evaluarea expresiei este apelată. Această ultimă formă nu este valabilă în cazul  funcțiilor de tip void. **Exemplu:**  #include <iostream> using namespace std; int produs (int x, int y) {  return x\*y;  } int main () { | conversația  euristică  Exemplul  metoda  exercițiului  problematizarea | în grup şi individual |

Pagina 2 din 7

|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
|  | int x=2,y=3;  cout <<1+produs(x,y); return 0;  }  În programul de mai sus este apelată o funcție care calculează produsul a două numere întregi. Programul tipăreşte suma dintre 1 şi produsul calculat. Apelul funcției s-a realizat din interiorul expresiei:  1+produs(x,y). **Observații:**  • În cadrul expresiei, apelul este un operand. El intră în calcul cu valoarea returnată de funcție.  • După apelul funcției se continuă cu evaluarea expresiei.  • La apel, ordinea de evaluare a parametrilor nu este definită. De exemplu, dacă apelăm funcția test astfel: test(2-3, 2+3) nu ştim dăcă înainte se efectuează scăderea sau adunarea.  Sistemul de operare alocă fiecărui program trei zone distincte în memoria internă în care se găsesc memorate variabilele programului. Aceste zone sunt: segmentul de date, segmentul de stivă şi zona liberă (Heap). O variabilă se caracterizează prin **patru atribute**:  **1. Clasa de memorare;**  **2. Vizibilitatea;**  **3. Durata de viață;**  **4. Tipul variabilei.**  **Clasa de memorare** precizează locul unde este memorată variabila respectivă. O variabilă poate fi memorată în **segmentul de date**, în cel de **stivă**, în **heap** sau într-un **registru** al microprocesorului.  **Vizibilitatea** precizează liniile textului sursă din care variabila respectivă poate fi accesată. Astfel avem:  **a. Vizibilitate la nivel de bloc** (instrucțiune compusă);  **b. Vizibilitate la nivel de fişier** – în cazul în care programul ocupă un singur fişier sursă, singurul caz pe care îl tratăm acum.  **c. Vizibilitate la nivel de clasă** – în legătură cu progrmarea orientată pe obiecte.  **Durata de viață** reprezintă timpul în care variabila respectivă are alocat spațiu în memoria internă. Astfel avem:  **a. Durată statică** – variabila are alocat spațiu în tot timpul execuției programului.  **b. Durată locală** – variabila are alocat spațiu în timpul în care se execută instrucțiune din blocul respectiv.  **c. Durată dinamică** – alocarea şi dezalocarea spațiului necesar variabilei respective se face de către programator prin operatori sau funcții speciale.  În C++ variabilele pot fi împărțite în trei mari categorii: **locale, globale** şi **dinamice**.  **Variabilele globale** se declară în afara corpului oricărei funcții. La declarare, variabilele globale sunt inițializate cu 0. |  |  |

Pagina 3 din 7

|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
|  | **Exemplu:**  #include <iostream> using namespace std; int a;  int t ()  { a=3;  cout<<a; }  int b; int main () {  b=4; cout<<a<<”\n”; t();  }  Variabilele a şi b sunt golobale. În astfel de cazuri, variabilele respective pot fi utilizate de toate funcțiile care urmează în textul sursă declarației variabilei respective. Din acest motiv astfel de variabile se numesc globale.  **Atributele variabilelor globale** sunt:  **1. Clasa de memorare** – segmentul de date.  **2. Vizibilitatea** – în cazul în care declarațiile acestora sunt înaintea tuturor funcțiilor, acestea sunt vizibile la nivelul întregului program (fişier). Dacă anumite funcții sunt plasate înaintea declarațiilor acestor variabile, atunci ele sunt vizibile doar pentru funcțiile care sunt plasate după aceste declarații. În exemplul de mai sus, variabila a poate fi accesată din corpul oricărei funcții, dar variabila b poate fi accesată doar de funcția main ().  **3. Durata de viață** a variabilelor globale este statică. Ele au spațiu rezervat în tot timpul execuției programului.  **Variabilele locale** sunt declarate în corpul funcțiilor. Pot fi declarate în orice bloc (instrucțiune compusă) al acestora. Variabilele declarate în corpul funcției **main()** sunt tot locale.  **Exemplu:** void t() {  int a=3; } int main () {  int b=4; }  Variabilele a şi b sunt locale. Variabila a este declarată în corpul funcției t(), iar variabila b este declarată în  corpul funcției main (). **Atributele variabilelor locale** sunt:  **1. Clasa de memorare** a variabilelor locale este, ***implicit***, **segmentul de stivă**. |  |  |

Pagina 4 din 7

|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
|  | **2. Vizibilitatea** variabilelor locale este la nivelul blocului la care au fost declarate.  **Exemplu:** void t()  { int b=4; {  int c=3; cout<<b<<” “<<c;  } }  Variabila **b** este vizibilă la nivelul funcției, iar variabila **c** este vizibilă doar la nivelul blocului în care a fost  declarată. **Exemplu:**  #include <iostream> using namespace std;  int a; void t()  { int a=4; {  int a=3; cout<<a<<”\n “;  } cout<<a<<”\n “;  } int main ()  { a=5; t();  cout<<a; }  Sunt declarate trei variabile, toate numite a. Una este globală, iar două sunt locale, dar declarate în blocuri  diferite. Programul va afişa conținutul tuturor acestor variable, pe rânduri diferite, adică (3, 4, 5). **Observații:**   |  |  | | --- | --- | |  | În cazul în care, într-un anumit bloc sunt vizibile (se pot accesa) mai multe variabile, toate cu acelaşi |   nume, dar cu domenii de vizibilitate diferite, se accesează variabila cu cu vizibilitatea cea mai mică.  Există posibilitatea ca, un ciclu for să conțină  declarația unei variabile locale. În secvența următoare se calculează suma primelor 4 numere naturale. Variabila i este declarată (şi în consecință  vizibilă) doar în blocul subordonat lui for: int n=4,s=0;  for(int i=1; i <=n; i++) s+=i;  **3. Durata de viață** a variabilelor locale este atât timp cât durează executarea blocului respectiv. |  |  |
| **5. Muncă independentă** | Test pentru verificarea cunoştințelor (vezi **Anexa 1**) | conversația euristică | individual |

Pagina 5 din 7



c)

float b (float x) {

x=2; int y=3;

return x+y+b-23; }



d)

float b (float x) {

x=2; int y=3; x+=y+b-23;

}



|  |  |  |  |
| --- | --- | --- | --- |
| **Etapele lecției** | **Conținutul lecției** | **Strategii didactice** | |
| **Metode şi procedee** | **Procedee de evaluare** |
| **6. Concluzii şi aprecieri** | Fac aprecieri cu privire la activitatea elevilor. | conversația discuția libera | în grup şi individual |
| **7. Tema pentru acasă** | Împart fişele de lucru cu tema pentru acasă (vezi **Anexa 2**). |  |  |

**Anexa 1 - Test pentru verificarea cunoştințelor**

2p **1.** Care antet de subprogram este corect?

**a)** int nume (int x; int y);

**b)** int nume (int x, y, float v[20]);

**c)** float nume (n: int);

**d)** float nume (float a, float b, float v[20]).

2p **2.** Care este clasa de memorare, durata de viață şi vizibilitatea variabilelor globale?

a) Segmentul de date, 2- 3 secunde, la nivelul întregului program;

b) Segmentul de stivă, atât cât durează programul, în blocul care conține definiția lor;

c) Segmentul de date, atât cât durează programul, la nivelul întregului program;

d) Segmentul de de stivă, cât timp le utilizăm, la nivelul întregului program.

2p 3. Care este clasa de memorare, durata de viață şi vizibilitatea variabilelor locale?

a) Segmentul de stivă, cât timp rulează subprogramul, la nivelul subprogramului;

b) Segmentul de stivă, atât cât durează programul, în blocul care conține definiția lor;

c) Segmentul de date, atât cât durează programul, la nivelul întregului program;

d) Segmentul de stivă, atât cât durează programul, la nivelul subprogramului.

3p 4. Care din funcțiile de mai jos este corectă sintactic şi logic?

a)

float a (float x) {

x=2;

a=x+2; }

b)

float b (float x) {

x=2; int y=3; return x+y+3;

}

Se accordă un punct din oficiu.

Pagina 6 din 7

**Anexa 2 - Fişa de lucru - tema pentru acasă**

**1.** Scrieți o funcție care primeşte ca parametru lungimea laturii unui pătrat şi returnează aria sa.

**2.** Scrieți o funcție care primeşte ca parametrii lungimea şi lățimea unui dreptunghi şi returnează aria sa.

**3.** Scrieți o funcție care primeşte ca parametru lungimea laturii unui pătrat şi returnează lungimea diagonalei.

**4.** Scrieți o funcție care primeşte ca parametrii lungimea şi lățimea unui dreptunghi şi returnează lungimea diagonalei.

**5.** Scrieți o funcție care primeşte ca parametrii de intrare lungimile celor două catete ale unui triunghi dreptunghic şi returnează lungimea ipotenuzei.

**6.** Scrieți o funcție care primeşte trei parametrii de tip real, cu semnificația de lungimi pentru trei segmente. Funcția va returna 1, dacă cele trei segmente pot forma un triunghi şi 0, în caz contrar.

**7.** Scrieți o funcție care primeşte ca parametru un număr natural de tip long şi returnează numărul care se obține citind cifrele în ordine inversă. Exemplu: dacă parametrul efectiv este 123, funcția va returna 321.

**8.** Scrieți o funcție care returnează ultima cifră a unui număr natural. Exemplu: dacă parametrul efectiv este 123, funcția va returna 3.

**9.** Pentru fiecare metodă de sortare cunoscută, scrieți câte o funcție care sortează descrescător un vector cu componente numere reale.

Pagina 7 din 7