Laboratorul 6

Expresii aritmetice

Exerciţiul 1

Din punct de vedere al compilatorului, instrucţiunile C++ nu trebuie formatate. Este, însă, extrem de important ca programele voastre să poată fi citite uşor atât de voi cât şi de alte persoane care ar dori să le folosească. Atunci când scrieţi un text în limba română, folosiţi câteva reguli simple de aliniere pentru a îl face uşor de citit. Studiaţi programul următor care calculează costul pe metru pătrat al unei locuinţe. El va fi compilat şi rulat corect, dar nu este conform niciunui standard de formatare.

```
//Program CostLocuinta.cpp
//Acest program calculeaza costul pe metru patrat al spatiului
locuibil
   //dintr-o casa, date fiind dimensiunile casei, numarul de
etaje
//si suprafata nelocuibila
#include <iostream>
#include<iomanip>// Pentru setw() si setprecision()
using namespace std;
const double LATIME=15.0; //Latimea casei
     const double LUNGIME =12.0;//Lungimea casei
const double ETAJE = 2;//Numarul de etaje, inclusiv parterul
const double SPATIU_NELOCUIBIL = 31.0;//Garaj, WC, etc.
const double PRET = 128000.0;//Pretul de vanzare
int main() {double suprafataTotala; //Suprafata totala
    double suprafataLocuibila; //Suprafata locuibila
double costPerMP; //Costul pe metru patrat al suprafetei
locuibile
cout.setf(ios::fixed,    ios::floatfield);    //    Stabilirea
formatului de iesire
cout.setf(ios::showpoint); //pt. float
suprafataTotala = LUNGIME*LATIME*ETAJE; suprafataLocuibila =
suprafataTotala  -  SPATIU_NELOCUIBIL;  costPerMP  =  PRET  /
suprafataLocuibila;
cout  <<  "Costul  pe  metru  patrat  este  "  <<  setw(6)  <<
setprecision(2)
<< costPerMP << " lei." << endl; return 0;}
```

Acelaşi program formatat:

```
//Program CostLocuinta.cpp
//Acest program calculeaza costul pe metru patrat al spatiului
locuibil
//dintr-o  casa,  date  fiind  dimensiunile  casei,  numarul  de
etaje
```

```
//si suprafata nelocuibila
#include <iostream>
#include<iomanip>// Pentru setw() si setprecision()
using namespace std;

const double LATIME=15.0; //Latimea casei
const double LUNGIME =12.0;//Lungimea casei
const double ETAJE = 2;//Numarul de etaje, inclusiv parterul
const double SPATIU_NELOCUIBIL = 31.0;//Garaj, WC, etc.
const double PRET = 128000.0;//Pretul de vanzare

int main()
{
     double suprafataTotala; //Suprafata totala
  double suprafataLocuibila; //Suprafata locuibila
     double costPerMP; //Costul pe metru patrat al suprafetei
locuibile

     cout.setf(ios::fixed,  ios::floatfield);  //  Stabilirea
formatului de iesire
     cout.setf(ios::showpoint); //pt. float

     suprafataTotala = LUNGIME*LATIME*ETAJE;
     suprafataLocuibila = suprafataTotala - SPATIU_NELOCUIBIL;
     costPerMP = PRET / suprafataLocuibila;

     cout << "Costul pe metru patrat este " << setw(6) <<
setprecision(2)
                << costPerMP << " lei." << endl;

     return 0;
}
```

1. Exemplu de programare.

Problema

Aveţi de petrecut o zi într-un oraş şi vă planificaţi să vizitaţi Muzeul de Istorie Naturală, un magazin de discuri, o galerie de artă, o librărie şi apoi veţi merge la un concert. Aveţi o hartă care vă arată unde se găsesc aceste puncte. Scara hărţii este de patru centimetri pentru un sfert de kilometru. Doriţi să stabiliţi distanţele dintre ele şi distanţa totală pe care o aveţi de parcurs de-a lungul întregii zile.

Ieşiri

Programul va afişa distanţele dintre punctele vizitate şi distanţa totală rotunjită la zecime de kilometru. Se vor tipări şi valorile pe care se bazează calculele, pentru verificare.

Discuţie

Puteţi măsura distanţele de pe hartă folosind o riglă. Programul trebuie să afişeze distanţele în kilometri, deci aceste distanţe trebuie multiplicate cu 0,25.

Afişăm apoi rezultatul rotunjit la prima zecimală pentru fiecare distanţă. Calculăm, în final distanţa totală şi o afişăm.

Singura problemă este rotunjirea la prima zecimală. Pentru rotunjirea la primul întreg folosim următoarea formulă:

```
int( unReal * 0.5 );
```

Deci pentru rotunjirea la prima zecimală folosim formula:

```
double( int( unReal * 10.0 + 0.5 ) ) / 10.0;
```

Programul

```
//*********************************************************
//Programul Excursie.cpp
//Acest program calculeaza distanta totala parcursa
//intr-o excursie, date fiind masuratorile facute pe o harta
//*********************************************************
#include <iostream>
#include <iomanip>
using namespace std;

const double DISTANTA1 = 1.5; //Masuratoarea primei distante
const double DISTANTA2 = 2.5; //Masuratoarea  celei  de-a  doua
distante
const double DISTANTA3 = 5.0; //Masuratoarea  celei  de-a  treia
distante
const double DISTANTA4 = 4.0; //Masuratoarea  celei  de-a  patra
distante
const double SCARA = 0.25;    //Scara hartii

int main()
{
  double distantaTotala;
  double km;

  cout.setf(ios::fixed, ios::floatfield);
  cout.setf(ios::showpoint);

  distantaTotala = 0.0;

  //Calculeaza  distanta  reala  pentru  fiecare  citire  de  pe
harta

  km = double(int( DISTANTA1 * SCARA * 10.0 +0.5)) / 10.0;
  cout << "Pentru masuratoarea " << setprecision(1)
       << DISTANTA1
       << " prima distanta este de " << km << " km."
       << endl;
  distantaTotala = distantaTotala + km;

  km = double(int( DISTANTA2 * SCARA * 10.0 +0.5)) / 10.0;
  cout << "Pentru masuratoarea " << setprecision(1)
       << DISTANTA2
       << " a doua distanta este de " << km << " km."
```

```
        << endl;
  distantaTotala = distantaTotala + km;

  km = double(int( DISTANTA3 * SCARA * 10.0 +0.5)) / 10.0;
  cout << "Pentru masuratoarea " << setprecision(1)
       << DISTANTA3
       << " a treia distanta este de " << km << " km."
       << endl;
  distantaTotala = distantaTotala + km;

  km = double(int( DISTANTA4 * SCARA * 10.0 +0.5)) / 10.0;
  cout << "Pentru masuratoarea " << setprecision(1)
       << DISTANTA4
       << " a patra distanta este de " << km << " km."
       << endl;
  distantaTotala = distantaTotala + km;

  //Afisarea distantei totale
  cout << endl;
  cout << "Distanta totala care trebuie parcursa este "
       << distantaTotala << " km."
       << endl;

  return 0;
}
```

Sugestii pentru debugging

- Verificaţi întotdeauna expresiile pentru a fi siguri că operaţiile sunt executate in ordinea pe care o doriţi;
- Evitaţi combinarea în aceeaşi expresie a valorilor întregi şi reale. Dacă este nevoie de aşa ceva, folosiţi conversiile explicite pentru a evita greşelile;
- Pentru fiecare asignare, verificaţi că rezultatul expresiei are acelaşi tip de dată ca şi variabila din stânga operatorului =. Daca nu, folosiţi conversii explicite pentru claritate şi siguranţă;
- Pentru fiecare funcţie de bibliotecă folosită, asiguraţi-vă că aţi inclus (`#include`) fişierul header potrivit;
- Examinaţi fiecare apel de funcţie pentru a vedea dacă aţi folosit numărul corect de parametri şi că tipurile lor de date sunt corecte.

Exerciţiul 2

Adăugaţi conversiile de tip următoarelor instrucţiuni pentru a le face clare şi explicite. Răspunsurile voastre trebuie să conducă la aceleaşi rezultate ca şi instrucţiunile originale:

```
unReal = 5 + unIntreg;
unIntreg = 2.5 * unIntreg / unReal;
```

Exerciţiul 3

Cum scrieţi următoarea formulă ca o expresie C++ care produce un rezultat de tip `double`?

9c/5 + 32

Exerciţiul 4

Reformataţi următorul program pentru a îl face clar şi uşor de citit:

```
//*************************************
                  //Programul SumProd.cpp
          //Acest program calculeaza
//suma si produsul a doua numere intregi
//*************************************

#include <iostream>
using namespace
std;
const int INT1 = 20; const int INT2=8;int main() { cout <<
"Suma lui " << INT1 << " si "
<< INT2 << " este " << INT1+INT2 << endl;cout
<< "Produsul lor este " << INT1*INT2 << endl;return 0;}
```

Exerciţiul 5

Ce valoare se păstrează în variabila `rezultat` de tip `int` în fiecare dintre următoarele instrucţiuni:

```
rezultat = 15 % 4;
rezultat = 7 / 3 + 2;
rezultat = 2 + 7 * 5;
rezultat = 45 / 8 * 4 + 2;
rezultat = 17 + (21 % 6) * 2;
rezultat = int(4.5 + 2.6 * 0.5);
```

Exerciţiul 6

Transformaţi următoarea instrucţiune în notaţie algebrică:

```
y = -b + sqrt(b * b – 4.0 * a * c);
```

Exerciţiul 7

Formatarea incorectă a unui program provoacă erori? (adevărat/fals)

Exerciţiul 8

Scrieţi o instrucţiune de asignare care calculează suma primelor *n* numere naturale folosind formula lui Gauss:

$$sum = n(n+1)/2.$$

Păstraţi rezultatul în variabila `sum` de tip `int`.

Exerciţiul 9

Completaţi următorul program C++. Programul trebuie să găsească şi să afişeze perimetrul şi aria unui dreptunghi, date fiind lungimea şi lăţimea. Folosiţi

mesaje sugestive care să însoţească afişarea rezultatelor. Nu uitaţi să folosiţi comentarii.

```
//*****************************************
//Programul Dreptunghi.cpp
//Acest program calculeaza perimetrul si aria
//unui dreptunghi cand se cunosc lungimea si latimea sa.
//*****************************************

#include <iostream>
using namespace std;

int main()
{
  double lungime;        //Lungimea dreptunghiului
  double latime;         //Latimea dreptunghiului
  double perimetrul;          //Perimetrul dreptunghiului
  double aria;      //Aria dreptunghiului

  lungime = 10.7;
  latime = 5.2;
```

Exerciţiul 10

Scrieţi un program C++ care converteşte temperaturile în grade Celsius în echivalentele lor în grade Fahrenheit. Formula este:

Fahrenheit = 9 * *Celsius* / 5 + 32

Folosiţi o constantă simbolică pentru temperatura în grade Celsius. Programul trebuie să tipărească ambele valori, cu mesaje explicite. Folosiţi comentarii în program, alegeţi nume semnificative pentru identificatori şi aliniaţi instrucţiunile ca în programele exemplu.

Exerciţiul 11

Scrieţi un program care calculează diametrul, lungimea şi aria unui cerc având raza 6,75. Asignaţi raza unei variabile `double` şi apoi afişaţi raza împreună cu un mesaj de identificare. Declaraţi constanta simbolică `PI` cu valoarea 3,14159. Programul trebuie să afişeze diametrul, lungimea şi aria cercului, fiecare pe linii separate, cu etichete de identificare. Tipăriţi fiecare valoare cu 5 zecimale într-un câmp de lungime 10. Folosiţi comentarii, identificatori cu nume semnificative şi aliniaţi instrucţiunile.

Exerciţiul 12

Limbajul de programare C++ stadard introduce patru noi operatori de cast. Aceştia sunt `static_cast`, `const_cast`, `reinterpret_cast` şi `dynamic_cast`. Aceşti operatori de conversie sunt dedicaţi câte unei situaţii precise, în timp ce filosofia metodei tradiţionale este *one cast fits all*. Cast-ul este o operaţie care este adeseori o sursă de erori, de aceea noii operatori introduşi în standardul C++ dau programatorului un mai mare control asupra codului. Sintaxa operaţiei de conversie prin `static_cast` este:

`static_cast<` *tipData* `>(`*parametru*`)`

unde *tipData* este tipul de dată către se face conversia, iar *parametru* este obiectul care urmează să fie convertit. Sintaxa celorlalţi trei operatori este asemănătoare cu aceasta. Operatorul `static_cast` se foloseşte pentru implementarea conversiilor între diverse tipuri de dată. Verificarea tipurilor de dată se face în momentul compilării.

Putem să afişăm codul ASCII al unui caracter prin instrucţiunea

```
cout << ”Caracterul (” << ’a’ << ”) are codul ASCII ”
     << static_cast< int >(’a’) << endl;
```

Scrieţi un program care afişează sub forma unui tabel caracterele corespunzătoare codurilor ASCII 35, 64, 65, 96, 97, 122, 150. Rezultatele ar trebui să apară pe ecran în formatul următor:

```
    caracter   codAscii
```

Indicaţie. Folosiţi conversia unei valori numerice – codul ASCII – către `char`.

Exerciţiul 13

Scrieţi un program care separă cifrele numerelor intregi de cinci cifre şi le afişează cu câtre trei spaţii între ele. Puteţi declara în programul vostru o constantă care stochează un astfel de număr. De exemplu, dacă numărul este `42339`, programul ar trebui să afişeze:

```
4   2   3   3   9
```

Indicaţie. Folosiţi împărţirea întreagă şi operatorul modulo.