Tipuri structurate de date

Probleme tip grilă

1. Se da urmatoarea structura:

C++

```
struct Student
{
    int varsta;
    char nume[25];

};
```

Pascal

```
type Student = record
    varsta : Integer;
    nume : array [1..25] of char;
  end;
```

Care din urmatoarele functii calculeaza si returneaza corect studentul cel mai in varsta dintr-o lista de studenti?

a)

```
Student celMaiBatran(Student studenti[], int lungimeListaStudenti) {
    Student rezultat = studenti[0];
    for (int i = 1; i < lungimeListaStudenti; i++) {
        if (rezultat->varsta < studenti[i]->varsta) {
            rezultat = studenti[i];
        }
    }
    return rezultat;
}
```

```
function celMaiBatran(studenti : array of Student; lungimeListaStudenti : integer) : Student;
var
  j : integer;
begin
  celMaiBatran := studenti[1];
  for j:=2 to lungimeListaStudenti do
  begin
    if (celMaiBatran[varsta] < studenti[j][varsta]) then
      celMaiBatran := studenti[j];
  end;
end;
```

b) (CROECT)

```
Student celMaiBatran(Student studenti[], int lungimeListaStudenti) {
    Student rezultat = studenti[0];

    for (int i = 1; i < lungimeListaStudenti; i++) {
        if (rezultat.varsta < studenti[i].varsta) {
            rezultat = studenti[i];
        }
    }

    return rezultat;
}
```

```
function celMaiBatran(studenti : array of Student; lungimeListaStudenti : integer) : Student;
var
  j : integer;
begin
  celMaiBatran := studenti[1];
  for j:=2 to lungimeListaStudenti do
  begin
    if (celMaiBatran.varsta < studenti[j].varsta) then
      celMaiBatran := studenti[j];
  end;
end;
```

c)

```
int celMaiBatran(Student studenti[], int lungimeListaStudenti) {
    Student rezultat = studenti[0];
    for (int i = 1; i < lungimeListaStudenti; i++) {
        if (rezultat[varsta] < studenti[i][varsta]) {
            rezultat = studenti[i];
        }
    }
    return rezultat;
}
```

```
function celMaiBatran(studenti : array of Student; lungimeListaStudenti : integer) : integer;
var
  j : integer;
begin
  celMaiBatran := studenti[1];
  for j:=2 to lungimeListaStudenti do
  begin
    if (celMaiBatran->varsta < studenti[j]->varsta) then
      celMaiBatran := studenti[j];
  end;
end;
```

d)

```
Student celMaiBatran(Student studenti[], int lungimeListaStudenti) {
    Student rezultat = studenti[0];
    for (int i = 1; i < lungimeListaStudenti; i++) {
        if (rezultat.nume < studenti[i].nume) {
            rezultat = studenti[i];
        }
    }
    return rezultat;
}
```

```
function celMaiBatran(studenti : array of Student; lungimeListaStudenti : integer) : Student;
var
  j : integer;
begin
  celMaiBatran := studenti[1];
  for j:=2 to lungimeListaStudenti do
  begin
    if (celMaiBatran.nume < studenti[j].nume) then
      celMaiBatran := studenti[j];
  end;
end;
```

2. Avand urmatoarele structuri:

C++

```cpp
struct Producator {
    char nume[100];
};

struct Marca {
    char nume[100];
    Producator producator;
};

struct Motor {
    int putere;
    int cm3;
};

struct Masina {
    Marca marca;
    Motor motor;
    int anFabricatie;
};


Masina masinaMea;
```

Pascal

```pascal
type Producator = record
    nume : array [1..100] of char;
  end;

type Marca = record
    nume : array [1..100] of char;
    producator : Producator;
  end;

type Motor = record
    putere : integer;
    cm3 : integer;

  end;

type Masina = record
    marca : Marca;
    motor : Motor;
    anFabricatie : integer;
  end;

var masinaMea : Masina;
```

Care dintre variantele urmatoare sunt corecte daca dorim sa accesam numele producatorului pentru variabila „masinaMea”?

a)

C++

```cpp
masinaMea->marca.producator.nume
```

Pascal

```pascal
masinaMea^.marca.producator.nume
```

b)

C++

```cpp
masinaMea->marca->producator->nume
```

Pascal

```pascal
masinaMea^.marca^.producator^.nume
```

c) (CORECT)

C++

```cpp
masinaMea.marca.producator.nume
```

Pascal

```pascal
masinaMea.marca.producator.nume
```

d)

C++	Pascal
`masinaMea[marca][producator].nume`	`masinaMea[marca][producator].nume`

Problema 1

Enunț

Scrieti o aplicatie care opereaza cu numere rationale pozitive. Se vor implementa urmatoarele operatii:

- Adunare
- Scadere
- Inmultire
- Impartire

Analiză

- in primul rand ar trebui sa ne gandim la o reprezentare convenabila pentru un numar rational, reprezentare care sa contina numaratorul si numitorul;
- urmatorul pas ar fi sa ne gandim la cum sa ne structuram programul in sub-programe: ar trebui sa avem o metoda de afisare, una de citire si cate o metoda care sa rezolve cele 4 operatii cerute in enuntul problemei;
- pentru ca nu retine in memorie numere rationale mari care se pot simplifica, ar trebuie sa ne gandim sa retinem doar numarul rational simplificat. Acest lucru il putem obtine daca am calcula cel mai mare divizor comun al numaratorului si numitorului si am impartii atat numaratorul, cat si numitorul la acest cmmdc imediat ce am citit de la tastatura numarul rational.

Implementare

Varianta C++

```cpp
#include <iostream>

struct Rational {
    int numarator;
    int numitor;
};

/*
 * Metoda ajutatoare pentru calcularea celui mai mare divizor comun.
 */
```

```
int cmmdc(int a, int b) {
    if (0 == b) return a;
    else return cmmdc(b, a % b);
}

void afisareNumarRational(Rational a) {
    cout << a.numarator << " / " << a.numitor << "\n";
}

/*
 * O metoda care ne ajuta sa cream un numar Rational simplificat, avand ca si parametrii de
intrare numaratorul si numitorul.
 *
 */
Rational creazaNumar(int numarator, int numitor) {
    Rational rational;

    // folosim cmmdc pentru a simplifica numarul rational final
    int cmmDivizorComun = cmmdc(numarator, numitor);

    rational.numarator = numarator / cmmDivizorComun;
    rational.numitor = numitor / cmmDivizorComun;
    return rational;
}

/*
 * O metoda care citeste de la tastatura 2 numere intregi care reprezinta numaratorul si
numitorul numarului Rational
 */
Rational citesteNumar() {
    int numarator, numitor;

    cout << "Dati numartorul:";
    cin >> numarator;

    cout << "Dati numitorul:";
    cin >> numitor;

    // verificam daca cele 2 numere citite de la tastatura sunt pozitive, iar in caz contrar
oprim executia programului
    if (numarator < 0 || numitor < 0) {
        cout << "Eroare: Numaratorul si numitorul trebiue sa fie numere pozitive!";
        exit(1);
    }

    // numitorul unui numar rational nu poate fi 0
    if (numitor == 0) {
        cout << "Eroare: Numitorul nu poate fi 0 (zero)!";
        exit(1);
    }

    Rational rational = creazaNumar(numarator, numitor);

    return rational;
}

/*
 * Functia de inmultire a 2 numere rationale. Inmultim numaratorii intre ei si numitorii
intre ei.
 */
Rational inmultire(Rational a, Rational b) {
    int numarator = a.numarator * b.numarator;
    int numitor = a.numitor * b.numitor;
```

```
    // returnam un nou numar rational simplificat
    return creazaNumar(numarator, numitor);
}
/*
 * Functia de impartire a 2 numere rationale.
 */
Rational impartire(Rational a, Rational b) {
    int numarator = a.numarator * b.numitor;
    int numitor = a.numitor * b.numarator;

    // returnam un nou numar rational simplificat
    return creazaNumar(numarator, numitor);
}

/*
 * Functia de adunare a 2 numere rationale.
 */
Rational adunare(Rational a, Rational b) {
    int numarator = (a.numarator * b.numitor) + (a.numitor * b.numarator);
    int numitor = a.numitor * b.numitor;

    // returnam un nou numar rational simplificat
    return creazaNumar(numarator, numitor);
}

/*
 * Functia de scadere a 2 numere rationale.
 */
Rational scadere(Rational a, Rational b) {
    int numarator = (a.numarator * b.numitor) - (a.numitor * b.numarator);
    int numitor = a.numitor * b.numitor;

    // returnam un nou numar rational simplificat
    return creazaNumar(numarator, numitor);
}


int main() {
    Rational a = citesteNumar();
    Rational b = citesteNumar();

    cout << "Numarul A = ";
    afisareNumarRational(a);

    cout << "Numarul B = ";
    afisareNumarRational(b);

    Rational inmultireAB = inmultire(a, b);
    cout << "Rezultatul INMULTIRII este: ";
    afisareNumarRational(inmultireAB);


    Rational adunareAB = adunare(a, b);
    cout << "Rezultatul ADUNARII este: ";
    afisareNumarRational(adunareAB);

    Rational scadereAB = scadere(a, b);
    cout << "Rezultatul SCADERII este: ";
    afisareNumarRational(scadereAB);

    Rational impartireAB = impartire(a, b);
    cout << "Rezultatul IMPARTIRII este: ";
```

```
    afisareNumarRational(impartireAB);

    return 0;
}
```

Varianta Pascal

```
Program NumereReale(input, output);

type
  Rational = record
    numarator : integer;
    numitor : integer;
  end;


{
  Metoda ajutatoare pentru calcularea celui mai mare divizor comun.
}
function cmmdc(a : integer; b : integer) : integer;
begin
  if 0 = b then
    cmmdc := a
  else cmmdc := cmmdc(b, a mod b);
end;

procedure afisareNumarRational(a : Rational);
begin
  write(a.numarator);
  write('/');
  write(a.numitor);
  writeln();
end;

{
O metoda care ne ajuta sa cream un numar Rational simplificat, avand ca si parametrii de
intrare numaratorul si numitorul.
}
function creazaNumar(numarator : integer; numitor : integer) : Rational;
var
  cmmDivizorComun : integer;
begin

  cmmDivizorComun := cmmdc(numarator, numitor);

  creazaNumar.numarator := numarator div cmmDivizorComun;
  creazaNumar.numitor := numitor div cmmDivizorComun;

end;

{
O metoda care citeste de la tastatura 2 numere intregi care reprezinta numaratorul si
numitorul numarului Rational
}
function citesteNumar() : Rational;
var
  numarator, numitor : integer;
begin

  write('Dati numartorul:');
  readln(numarator);
```

```
  write('Dati numitorul:');
  readln(numitor);

  { verificam daca cele 2 numere citite de la tastatura sunt pozitive, iar in caz contrar
oprim executia programului }
  if ((numarator < 0) or (numitor < 0)) then
  begin
    writeln('Eroare: Numaratorul si numitorul trebiue sa fie numere pozitive!');
    exit;
  end;

  {numitorul unui numar rational nu poate fi 0}
  if numitor = 0 then
  begin
    writeln('Eroare: Numitorul nu poate fi 0 (zero)!');
    exit;
  end;

  citesteNumar := creazaNumar(numarator, numitor);
end;

{
  Functia de inmultire a 2 numere rationale. Inmultim numaratorii intre ei si numitorii intre
ei.
}
function inmultire(a : Rational; b : Rational) : Rational;
var
  numarator, numitor : integer;
begin
  numarator := a.numarator * b.numarator;
  numitor := a.numitor * b.numitor;

  { returnam un nou numar rational simplificat }
  inmultire := creazaNumar(numarator, numitor);
end;

{
  Functia de impartire a 2 numere rationale.
}
function impartire(a : Rational; b : Rational) : Rational;
var
  numarator, numitor : integer;
begin
  numarator := a.numarator * b.numitor;
  numitor := a.numitor * b.numarator;

  { returnam un nou numar rational simplificat }
  impartire := creazaNumar(numarator, numitor);
end;

{
  Functia de adunare a 2 numere rationale.
}
function adunare(a : Rational; b : Rational) : Rational;
var
  numarator, numitor : integer;
begin
  numarator := (a.numarator * b.numitor) + (a.numitor * b.numarator);
  numitor := a.numitor * b.numitor;

  { returnam un nou numar rational simplificat }
  adunare := creazaNumar(numarator, numitor);
```

```
end;

{
  Functia de scadere a 2 numere rationale.
}
function scadere(a : Rational; b : Rational) : Rational;
var
  numarator, numitor : integer;
begin
  numarator := (a.numarator * b.numitor) - (a.numitor * b.numarator);
  numitor := a.numitor * b.numitor;

  { returnam un nou numar rational simplificat }
  scadere := creazaNumar(numarator, numitor);
end;


var
  a, b, adunareAB, scadereAB, inmultireAB, impartireAB : Rational;
begin
  a := citesteNumar();
  b := citesteNumar();

  write('Numarul A = ');
  afisareNumarRational(a);
  write('Numarul B = ');
  afisareNumarRational(b);

  inmultireAB := inmultire(a, b);
  write('Rezultatul INMULTIRII este: ');
  afisareNumarRational(inmultireAB);


  adunareAB := adunare(a, b);
  write('Rezultatul ADUNARII este: ');
  afisareNumarRational(adunareAB);

  scadereAB := scadere(a, b);
  write('Rezultatul SCADERII este: ');
  afisareNumarRational(scadereAB);

  impartireAB := impartire(a, b);
  write('Rezultatul IMPARTIRII este: ');
  afisareNumarRational(impartireAB);

end.
```

Rezultate executie:

Dati numartorul: 3
Dati numitorul: 7
Dati numartorul: 4
Dati numitorul: 9
Numarul A = 3 / 7
Numarul B = 4 / 9

Rezultatul INMULTIRII este: 4 / 21
Rezultatul ADUNARII este: 55 / 63

Rezultatul SCADERII este: 1 / -63
Rezultatul IMPARTIRII este: 27 / 28