Conf. dr. Iuliana Bocicor

Probleme consultații - 17 februarie 2024
Algoritmi care lucreaza pe numere

Problema 1

Enunț

Să se scrie un subprogram care să afișeze perechile de numere naturale „prietene” dintr-un interval dat [a,b], a, b – numere naturale, a < b. Două numere naturale **x** şi **y** sunt „prietene” dacă suma divizorilor lui **x** este egală cu suma divizorilor lui **y**. Pentru un număr, se vor considera toți divizorii, inclusiv 1 și numărul însuși.

Analiză

Vom scrie un subprogram care verifică daca **x** si **y** din [**a,b**] au aceeași sumă **s** a divizorilor.
Avem nevoie şi de un subprogram care determină suma divizorilor. Vom realiza acest subprogram cu cea mai bună complexitate, adică $O(\sqrt{n})$.

Specificarea subalgoritmilor

- Subalgoritmul **SumDiv(n)**:
  Descriere: Calculează suma divizorilor numărului dat.
  Date: *n* – număr natural.
  Rezultate: *s* – număr natural, reprezentând suma divizorilor numărului dat.

- Subalgoritmul **AfisPrietene(a, b)**:
  Descriere: Afișează toate numerele prietene din intervalul [a, b].
  Date: a, b – numere naturale, a < b
  Rezultate: Se afișează toate numerele prietene din intervalul dat.

Implementare

Varianta C++

```
long sumDiv(long n)
{
	long S = n + 1;
	long d = 2;
	for (d = 2; d * d < n; d++)		//se verifica pentru d = 2,3,...,sqrt(n)-1
		if (n % d == 0)
			S += d + n / d;		//se aduna factorii complementari
	if (d * d == n)
			S += d;			// daca n este patrat perfect se mai aduna
	return S;
```

```
}
```

O variantă recursivă a acestui subprogram:

```
long SumDivRec(long n, long d)
{
	if (d * d > n) return 1 + n;

	//daca n este patrat perfect
	if (d * d == n) return d + SumDivRec(n, d + 1);

	//se verifica pentru d= 2, ... , sqrt(n)
	//se aduna factorii complementari
	if (n % d == 0)
		return d + (n / d) + SumDivRec(n, d + 1);
	return SumDivRec(n, d + 1);		//daca d nu e divizor
}

long SumDiv(long n)
{
	return SumDivRec(n, 2);
}

void AfisPrietene(long a, long b)
{
	for (long x = a; x < b; x++)
		for (long y = x + 1; y <= b; y++)
			if (SumDiv(x) == SumDiv(y)) cout << x << " si " << y << " sunt
prietene \n";
}
```

Exemple

Date de intrare	Rezultate
1, 20	6, 11 10, 17 14, 15
15, 25	15, 23 16, 25
1, 10	-
100, 140	102, 110 104, 116 114, 135 115, 119 132, 140
1023, 1050	1030, 1035

Problema 2

Enunț

Să se realizeze câte o funcție recursivă cu un singur parametru (numărul *n*) pentru:

a) determinarea cifrei minime a lui *n*, se va returna cifra minimă;

b) determinarea cifrei pare maxime a lui n, se va returna cifra pară maximă sau -1, dacă *n* nu are cifre pare;

Analiză

Vom scrie formulele recursive pentru cele două cerințe.

a)

$$cifraMinima(n) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ \min\{n\ mod\ 10,\quad cifraMinima\left(\left[\frac{n}{10}\right]\right),\quad altfel \end{cases}$$

O altă variantă recursivă pentru determinarea cifrei minime este:

$$cifraMinima(\overline{a_1a_2\dots a_{n-1}a_n}) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ cifraMinima(\overline{a_1a_2\dots \min\{a_{n-1}, a_n\}}),\quad altfel \end{cases}$$

În această variantă apelul recursiv se face pentru numărul obținut prin înlocuirea ultimelor două cifre ale sale cu minimul dintre acestea.

b)

$$cifraParaMaxima(n) = \begin{cases} n, dacă\ e\ este\ format\ dintr-o\ singură\ cifră\ pară \\ -1, dacă\ n\ este\ format\ dintr-o\ singură\ cifră\ impară \\ cifraParaMaxima\left(\left[\frac{n}{10}\right]\right),\quad daca\ n\ mod\ 10\ este\ impara \\ \max\{n\ mod\ 10, cifraParaMaxima\left(\left[\frac{n}{10}\right]\right\}, daca\ n\ mod\ 10\ este\ para \end{cases}$$

Specificarea funcțiilor

Funcția **cifraMinima(n)**:

Descriere: Returneaza cifra minima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra minima a numarului dat.

Funcția **cifraParaMaxima(n)**:

Descriere: Returneaza cifra para maxima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra para maxima a numarului dat.

Implementare

Varianta C++

```
#include <iostream>

using namespace std;

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V1(int n)
{
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int cifraMinima_restul_numarului = cifraMinima_V1(n / 10);
	return (n % 10 < cifraMinima_restul_numarului ? n % 10 :
cifraMinima_restul_numarului);
}

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V2(int n)
{
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int minim_ultimele_doua_cifre = n % 10;
	int penultima_cifra = (n / 10) % 10;
	if (penultima_cifra < minim_ultimele_doua_cifre)
		minim_ultimele_doua_cifre = penultima_cifra;
	return cifraMinima_V2((n/100) * 10 + minim_ultimele_doua_cifre);
}

/*
Descriere:  Returneaza cifra para maxima a unui numar dat.
Date : n – numar natural.
Rezultate : cifra para maxima a numarului dat.
*/
int cifraParaMaxima(int n)
{
	if (n <= 9)          // daca n este format dintr-o singura cifra
	{
		if (n % 2 == 0)             // daca este par
			return n;
		else
			return -1;
	}

	int ultima_cifra = n % 10;
	if (ultima_cifra % 2 != 0)
```

```
		return cifraParaMaxima(n / 10);
	int cifraParaMaxima_restul_numarului = cifraParaMaxima(n / 10);
	return (ultima_cifra > cifraParaMaxima_restul_numarului ? ultima_cifra :
cifraParaMaxima_restul_numarului);
}

int main()
{
	int n = 0;
	cout << "Introduceti numarul: ";
	cin >> n;

	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V1(n) <<
endl;
	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V2(n) <<
endl;
	cout << "Cifra para maxima a numarului " << n << " este: " << cifraParaMaxima(n)
<< endl;

	return 0;
}
```

Problema 3

Enunț

Fie **i,j,k** trei numere naturale. Să scrie un subprogram care returnează restul împărțirii numărului **(i^j)** la **k**, deci **(i^j)**modulo **k** (iterativ şi recursiv).

Exemple:

(100^{100}) modulo 7 = 2
(125^{199}) modulo 999 = 800
(2^{10}) modulo 9 = 7

Analiză

- practic se înmulțesc resturile modulo *k*; nu e nevoie de calculul expresiei i^j;
- se calculează la fiecare pas de înmulțire restul produsului (modulo k);

Aceasta deoarece se știe că:

(a * b) mod c = (a mod c * b mod c) mod c

Atunci:

a^x mod c = (a mod c * a^{x-1} mod c) mod c

Pentru varianta recursivă, formula recursivă se deduce din formula anterioară astfel:

$$rest(i,j,k) = \begin{cases} 1, & daca\ j = 0 \\ ((i\ mod\ k) * rest(i, j-1, k))\ mod\ k \end{cases}$$

Specificarea funcției

Funcția **Rest(i, j, k)**:

Descriere: returnează restul împărțirii lui (i^j) la k.

Date: i,j,k - numere naturale

Rezultate: R un număr natural: R∈{0,1,...k-1}

Implementare

Varianta iterativă C++

```
/*
Descriere:  returneaza restul impartirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un numar natural: R in {0,1,...k-1}.
*/
int Rest(int i, int j, int k)
{
	int r = i % k;
	int rest = 1;
	while (j > 0)
	{
		rest = (rest * r) % k;
		j--;
	}
	return rest;
}
```

Varianta recursivă C ++

```
/*
Descriere:  returneaza restul impartirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un numar natural: R in {0,1,...k-1}.
*/
int Rest(int i, int j, int k)
{
	if (j == 0)
		return 1;
	return ((i % k) * Rest(i, j - 1, k)) % k;
}
```

Exemple

Date de intrare	Rezultate
100, 100, 7	2
125, 199, 999	800

2, 10, 9	7
8, 0, 100	1
5, 151, 5	0

Problema 4

Enunț

Dat fiind un număr real *x*, să se determine rădăcina pătrată *r* a acestuia, cu aproximarea *epsilon*.

Analiză

Se va folosi metoda înjumătățirii intervalului: pornind de la un interval inițial pentru rădăcina pătrată, se selectează iterativ subintervalul în care se află rădăcina, până când se ajunge la o lungime maximă a intervalului mai mică decât aproximarea cerută.

Specificarea subalgoritmilor

- Subalgoritmul **citesteNumarSiAproximare(x, eps)**:
  Descriere: Citește două numere reale.
  Date: -
  Rezultate: *x, eps* – numere reale, x > 0.

- Subalgoritmul **determinaRadacinaPatrata(n, eps)**:
  Descriere: Determina rădăcina pătrata a unui număr dat, folosind o aproximare dată.
  Date: x, eps – numere reale, x > 0
  Rezultate: rădăcina pătrată a lui *x*, cu aproximarea *eps*.

Implementare

Varianta C++

```
/*
Descriere: Citeste doua numere reale, reprezentand numarul caruia trebuie sa ii fie
calculata radacina patrata si aproximarea (eroarea).
Date: -
Rezultate: se citesc cele doua numere.
*/
void citesteNumarSiAproximare(double& x, double& eps)
{
	cout << "Introduceti numarul real si apoi precizia: ";
	cin >> x >> eps;
}
```

```
/*
	Descriere: Determina radacina patrata a unui numar dat, folosind o aproximare
data.
	Date: x, eps - numere reale.
	Rezultate: radacina patrata a lui x, cu aproximarea eps.
*/
double determinaRadacinaPatrataIterativ(double x, double eps)
{
	double dreapta = x;
	double stanga = 0;
	double mij = (dreapta + stanga) / 2;

	while (abs(dreapta - stanga) > eps)
	{
		if (mij * mij > x)
			dreapta = mij;
		else
			stanga = mij;

		mij = (dreapta + stanga) / 2;
	}

	return mij;
}

double determinaRadacinaPatrataRec(double x, double eps, double stanga, double dreapta)
{
	double mij = (dreapta + stanga) / 2;

	if (abs(dreapta - stanga) < eps)
		return mij;

	if (mij * mij > x)
		return determinaRadacinaPatrataRec(x, eps, stanga, mij);

	return determinaRadacinaPatrataRec(x, eps, mij, dreapta);
}

/*
	Descriere: Determina radacina patrata a unui numar dat, folosind o aproximare
data.
	Date: x, eps - numere reale.
	Rezultate: radacina patrata a lui x, cu aproximarea eps.
*/
double determinaRadacinaPatrataRecursiv(double x, double eps)
{
	return determinaRadacinaPatrataRec(x, eps, 0, x);
}

int main()
{
	double x = 0, eps = 0;
	citesteNumarSiAproximare(x, eps);
	cout << "Iterativ: ";
	cout << "Radacina patrata a numarului " << x << ", cu aproximarea " << eps << "
este: " << determinaRadacinaPatrataIterativ(x, eps) << endl;
	cout << endl;
	cout << "Recursiv: ";
```

```
	cout << "Radacina patrata a numarului " << x << ", cu aproximarea " << eps << "
este: " << determinaRadacinaPatrataRecursiv(x, eps) << endl;
	return 0;
}
```

Exemple

Date de intrare	Rezultate
5, 0.01	2.23145
5, 0.0001	2.23606
100, 0.0000001	10
100, 0.1	10.0098
2, 0.03	1.41406

Probleme tip grilă

1. Se considera subalgoritmul ceFace(n), unde ***n*** este un numar natural, $1 \leq \boldsymbol{n} \leq 1000000000$. S-au folosit notația *x%y* pentru restul împărțirii numărului întreg *x* la numărul întreg *y* și *[x]* pentru partea întreagă a numărului real *x.*

```
Algorithm ceFace(n)
	If ([n / 10]) == 0)
		If (n % 2 == 0)
			return n;
		EnfIf
		return 0;
	Else
		If (n % 10 % 2 = 1)
			return ceFace([n / 10]);
		EndIf
		return ceFace([n / 10]) * 10 + n % 10;
	EndIf
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?

A. La apelul ceFace(2528) se va returna 2528
B. La apelul ceFace(335) se va returna 0
C. Cand algoritmul este apelat pentru n – numar impar, se va returna 0
D. Cand algoritmul este apelat pentru n – numar par, se va returna n

De câte ori se apelează subalgoritmul pentru a = 253401976?

a. De 3 ori
b. De 9 ori

c. De 6 ori
d. De 7 ori

2. Știind că variabilele *a* și *i* sunt întregi, stabiliți ce reprezintă valorile afișate de algoritmul de mai jos. S-au folosit notațiile *x%y* pentru restul împărțirii numărului întreg *x* la numărul întreg *y*, și *[x]* pentru partea întreagă a numărului real *x*.

```
Algorithm afis()
            a ← 10
            For i ← 1,6
                  write [a/7]
                  a ← a % 7 * 10
            EndFor
 EndAlgorithm
```

a. primele 6 zecimale ale lui 10/7
b. primele 7 zecimale ale lui 1/6
c. primele 6 cifre ale lui 10/7
d. primele 7 cifre ale lui 10/6

3. Ce va afișa algoritmul pseudocod de mai jos pentru două numere naturale nenule *a* și *b*? S-a notat cu *x%y* restul împărțirii numerelor întregi *x* și *y*.

```
Algorithm afis()
        read a,b
        c ← 1
        While a * c % b ≠ 0
             c ← c + 1
        EndWhile
        write a*c
EndAlgorithm
```

a. a^b
b. cel mai mic multiplu comun
c. cel mai mare divizor comun
d. a*b

4. Care este rezultatul execuției următoarei secvențe?

```
Algorithm ceFace()
      Read n,c {n, natural>0, c cifra}
      z ← 0
```

```
        While n % 10 = c
                n ← [n/10]
                z ← z + 1
        EndWhile
        write z
EndAlgorithm
```

a. Afişează 3 pentru n=123 şi c=3;
b. Afişează 2 pentru n=12003 şi c=0;
c. Afişează n, pentru n=c=1;
d. Afişează 4, pentru n=1277771 şi c=7
e. Afişează 3, pentru n=12555 şi c=5.

5. Care este rezultatul execuției următoarei secvențe (x, m sunt numere intregi)?

```
Algorithm ceFace(x, m)
        y ← 1;
        While (m > 0)
                If (m % 2 = 0)
                        m ← [m / 2]
                        x ← x * x;
                Else
                        m ← m - 1;
                        y ← y * x;
                EndIf
        EndWhile
        return y
EndAlgorithm
```

a. Afişează x, dacă m<=0.
b. Afişează x^2, dacă m=2.
c. Afişează 1, pentru m < 0.
d. Afişează x^k, dacă $m=2^k$.
e. Afişează un număr negativ dacă x < 0.

6. Care funcţie schimbă valorile întregi ale lui a şi b între ele?

```
A.
void F(int& a, int& b){
      a-=b;
      b+=a;
      a=b-a;
}
```

```
B.
void F(int& a, int& b){
      a=a+b;
      b=a-b;
      a=a-b;
}
```

```
C.
void F(int& a, int& b){
      a=a/b;
      b=a*b;
      a=b/a;
}
```

```
D.
void F(int& a, int& b){
      a=a*b;
      b=a/b;
      a=a/b;
}
```

```
E.
void F(int& a, int& b){
      b=b-a;
      b=a*b;
      a=b;
}
```

7. Care subprogram determină dacă numărul întreg n este prim (*true* dacă n e prim, *false* altfel)?

```
a)
bool Prim(int n){
  int d=2;
  while(d*d<=n){
     if(n%d==0) return false;
     d=(d==2)?3:d+2;
  }
  return true;
}
```

```
b)
bool Prim(int n){
  if(n<=1) return false;
  for (int d=2;d*d<=n;
d=(d==2)?3:d+2)
     if(n%d==0) return false;
  return true;
}
```

```
c)
bool Prim(int n, int d){//apel
extern, d=2
  if(n<=1) return false;
  if(d*d>n) return true;
  if(n%d==0) return false;
  return Prim(n,(d==2)?3:d+2);
}
```

8. Se considera subprogramul de mai jos, unde **n** este un numar întreg.

```
Algorithm SP(n)
  For i ← 2, n
    c ← 0
    While (n % i == 0)
       n ← n / i;
       c ← c + 1
```

```
    EndWhile
    If (c > 0) write i, c
    EndIf
   EndFor
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?

A. Subprogramul afiseaza perechi de numere prime, pentru n>=2.
B. Subprogramul afiseaza numerele prime pana la n, pentru n>=2.
C. Subprogramul afiseaza factorii primi ai lui n, pentru n>=2.
D. Subprogramul afiseaza factorii primi ai lui n si puterile lor, pentru n>=2.
E. Subprogramul nu afiseaza nimic, daca n < 2.

9. Se considera urmatorul algoritm, unde **n** este un numar natural:

```
Algorithm ceFace(n)
      t ← 1
      c ← n % 10;
      n ← n / 10;
      While (t = 1 && n > 0)
            If (n % 10 > c)
                  t ← 0
            EndIf
            c ← n % 10;
            n ← n / 10;
      EndWhile
      return t
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?

A. Algoritmul afiseaza 1, daca n > 0.
B. Algoritmul afiseaza 0, daca n = 1234.
C. Algoritmul afiseaza 0, daca n = 4321.
D. Algoritmul afiseaza 1, daca n = 5555.
E. Algoritmul afiseaza 0, daca n = 0