Problema 51

Stan Ana-Maria, Grupa 241

Se numește număr “bine ordonat crescător” un număr natural cu proprietatea că cifrele sale citite de la stânga la dreapta sunt în ordine crescătoare.

Exemplu: 3478.

Enunțul propus:

Fiind dat un număr natural citit de la tastatură, să se stabilească dacă este “bine ordonat crescător” sau nu, afișându-se un mesaj corespunzător.

Discutarea enunțului:

Întrebam elevii cum se gândesc sa compare cifrele numărului dat, de ce variabile auxiliare avem nevoie si luam mai multe numere ca exemplu sa stabilim împreună daca sunt “bine ordonat crescător”.

Exemple:

3478->3<=4<=7<=8 -> “bine ordonat crescător”

254 -> 2<=5>4 -> nu este “bine ordonat crescător”

1 -> “bine ordonat crescător”

11-> 1<=1 ->“bine ordonat crescător”

112->1<=1<=2 ->“bine ordonat crescător”

Prin exemple si discuții stabilim împreună:

- Daca avem un număr format dintr-o cifra acesta o sa fie “bine ordonat crescător”.
- Daca avem un număr care are pe poziții alăturate aceeași cifra (33, 334) acesta o sa fie “bine ordonat crescător”, deoarece enunțul cere ca cifrele sa fie in ordine crescătoare nu strict crescătoare.
- Variabilele de care avem nevoie si modurile cele mai ușoare si optime pentru a rezolva problema.

Urmează sa discutam doua moduri in care putem sa rezolvam problema:

Variabilele auxiliare:

Cazul 1:

O variabila de tip **boolean** in care o sa salvam **true**, daca numărul respecta regula sau **false** daca acesta nu o sa respecte regula.

Cazul 2:

Doua variabile de tip **int**(numere naturale), in prima salvam numărul de cifre ale numărului – 1, iar in a doua numărul de perechi pe poziții auxiliare care respecta regula data.

Date de intrare:

n – număr NATURAL citit de la tastatura

Afișare pe ecran:

Mesajul "Numărul este bine ordonat crescător" daca numărul respecta regula data sau "Numărul nu este bine ordonat crescător" in caz contrar.

Algoritmul din cazul 1:

Definirea variabilelor si citirea numărului dat de la tastatura.

```
int n;
bool ok = true;

cin>>n;
```

Folosirea unei bucle de tip **while** pentru a putea accesa fiecare cifra din număr. Aceasta are condiția n>9, deoarece vrem ca bucla sa ruleze cat timp numarul are cel putin doua cifre(pentru a putea sa le comparam). In plus, aceasta condiție din **while** ne asigura ca daca numărul este de o cifra acesta o sa fie “bine ordonat crescător”.

```
while(n>9){
    if(n%10<n/10%10)
      {
        ok = false;
        break;
      }
    n = n/10;
}
```

In condiția **if** verificam daca ultima cifra a lui n (n%10) este strict mai mica decât penultima (n/10%10), daca aceasta este atunci numărul nu respecta regula, schimbam variabila auxiliara in false si apelam break pentru a nu continua sa verificam si restul numărului (știm deja ca numărul nu respecta regula si asta ne ajuta la optimizare). La final, ștergem ultima cifra a numărului si repetam tot pana numărul devine o cifra (sau se apelează break).

Daca variabila nu s-a schimbat si a rămas **true**, numărul respecta regula si afișam acest lucru, daca variabila auxiliara este **false** înseamnă ca numărul nu respecta regula si afișam "Numărul nu este bine ordonat crescător".

```
if(ok==true)
    cout<<"Numarul este bine ordonat crescator";
else
    cout<<"Numarul nu este bine ordonat crescator";
```

Algoritmul din cazul 2:

Definirea variabilelor si citirea numărului dat de la tastatura.

```
int n;
int contor1=0,contor2=0;

cin>>n;
```

Contor1 reprezintă numărul de perechi pe poziții consecutive care respecta regula. Contor2 reprezintă numărul de cifre ale numărului – 1, pentru ca bucla **while** o sa se oprească când numărul mai are o cifra si nu o mai luam in considerare.

Condiția din **while** este aceeași ca cea de mai sus, schimbat fiind doar conținutul buclei. De data aceasta in **if** condiția este ca cele doua numere consecutive sa respecte regula(mai sus era daca nu respecta regula) si daca aceasta este respectata contor1 o sa crească cu 1. In final, ștergem ultima cifra a numărului n si creștem contorul2 cu 1, reținând astfel numărul de cifre-1 din numărul dat de la tastatura.

```
while(n>9){
    if(n%10>=n/10%10)
        contor1=contor1+1;
    n = n/10;
    contor2=contor2+1;
}
```

```
if(contor1==contor2)
    cout<<"Numarul este bine ordonat crescator";
else
    cout<<"Numarul nu este bine ordonat crescator";
```

Daca variabila contor1 este egala cu contor2, numărul respecta regula si afișam acest lucru, daca acestea diferă înseamnă ca numărul nu respecta regula si afișam "Numărul nu este bine ordonat crescător".

Discuție finala:

Ambele soluții au complexitate maxima O(n), dar prima rezolvare este mai eficienta, deoarece folosim o singura variabila auxiliara si daca găsim o pereche de cifre consecutive din număr care nu respecta regula nu mai parcurgem tot numărul.

Barem

Oficiu 1p

Cunoștințe generale necesare 1p

Rezolvare 6p

Stil(comentarii, identare) 2p

Puncte bonus:

Rezolvare mai eficienta 2p bonus

Răspunsuri suplimentare in timpul orei 0,5p

Cele doua rezolvari complete:

Cazul 1:

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   bool ok = true;
   cin>>n;
   while(n>9){
      if(n%10<n/10%10)
        {
          ok = false;
          break;
        }
      n = n/10;
   }
```

```
    if(ok==true)
        cout<<"Numarul este bine ordonat crescator";
    else
        cout<<"Numarul nu este bine ordonat crescator";
    return 0;
}
```

Cazul 2:

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    int contor1=0,contor2=0;
    cin>>n;
    while(n>9){
        if(n%10>=n/10%10)
            contor1=contor1+1;
        n = n/10;
        contor2=contor2+1;
    }
    if(contor1==contor2)
        cout<<"Numarul este bine ordonat crescator";
    else
        cout<<"Numarul nu este bine ordonat crescator";
    return 0;
}
```