

## Siruri recursive

### • Sirurile Recursive

- Un șir se numește recursiv dacă se autoapelează adică, dacă în corpul său există un modul care se autoapelează.
- Recursivitatea se realizează cu ajutorul subprogramelor.
- Un subprogram care se autoapelează se numește subprogram recursiv. Prin urmare un algoritm se numește recursiv sau un program se numește recursiv dacă conține cel puțin un subprogram care se autoapelează.
- Prin urmare orice subprogram recursiv trebuie să îndeplinească următoarele condiții:
  - să se poată executa cel puțin o dată fără a se autoapela;
  - toate autoapelurile să se producă astfel încât la un moment dat să se ajungă la îndeplinirea condiției de oprire.

### Cum se realizează recursivitatea?

- Recursivitatea se realizează în felul următor:
  - Din afara subprogramului facem un apel al subprogramului;
  - Dacă nu este îndeplinită o condiție (numită condiție de oprire), algoritmul se autoapelează de un anumit număr de ori (până când e îndeplinită condiția). La fiecare nouă autoapelare a subprogramului se reexecută secvența de instrucțiuni din corpul său, eventual cu alte date, creându-se un lanț de autoapeluri recursive.

Observatie! - Dacă condiția de oprire nu este îndeplinită niciodată sau nu există, algoritmul va avea un rezultat asemănător intrării într-un ciclu infinit.

### • Sirul lui Fibonacci

#### Ce este șirul lui Fibonacci?

- Șirul lui Fibonacci este o secvență infinită de numere în care fiecare număr este suma celor două numere anterioare. Forma sa standard este: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Și continuă la infinit. Adică, fiecare număr din șir (începând cu al treilea număr) este suma celor două numere precedente.



### Observații!

- Intuitiv, putem spune că un algoritm recursiv are același efect ca și o buclă (ciclu, instrucțiune repetitivă) – repetă execuția unui set de instrucțiuni. În mod analog, deducem că repetarea nu trebuie să se realizeze de un număr infinit de ori. De aici provine necesitatea existenței condiției de oprire.
- Varianta de implementare rămâne la latitudinea programatorului.
- Varianta recursivă este recomandată în cazul în care problemele sunt definite printr-o relație de recurență însă acest tip de algoritmi sunt mai greu de urmărit și de obicei necesită un timp de execuție mai mare.

### Probleme:

1.

#255

### Enunț:

Se consideră șirul Fibonacci, definit astfel:  $f_1=1$ ,  $f_2=1$ ,  $f_n=f_{n-1}+f_{n-2}$ , dacă  $n>2$ .

### Cerință:

Se dă un număr natural  $n$ . Să se afișeze în ordine crescătoare, primii  $n$  termeni ai șirului lui Fibonacci. Programul citește de la tastatură numărul  $n$ . Programul afișează pe ecran primii  $n$  termeni ai șirului lui Fibonacci, în ordine crescătoare, separați printr-un spațiu.

Restricții:  $1 \leq n \leq 40$ ;

### Exemplu:

Date de intrare: 5

Date de ieșire: 1 1 2 3 5

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n, a=1, b=1, c;
6      cin>>n;
7      if (n==1)
8          cout<<1;
9      else{
10         cout<<1<<" "<<1<<" ";
11         for(int i=3; i<=n; i++){
12             c=a+b;
13             cout<<c<<" ";
14             a=b;
15             b=c;
16         }
17         return 0;
18     }
19 }
```

```
5
1 1 2 3 5
Process returned 0 (0x0)   execution time : 1.509 s
Press any key to continue.
```

2.

Cerință:

Să se afișeze toți termenii șirului lui Fibonacci mai mici decât un număr natural  $n$  introdus de la tastatură.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int n, a=1, b=1, c;
8      cin>>n;
9      cout<<a<<" ";
10     while(b<n)
11     {
12         cout<<b<<" ";
13         c=a+b;
14         a=b;
15         b=c;
16     }
17
18     return 0;
19 }
20
```

```
6
1 1 2 3 5
Process returned 0 (0x0)   execution time : 1.115 s
Press any key to continue.
```

3.

#423

Enunț:

Se consideră șirul Fibonacci, definit astfel:  $f_1=1$ ,  $f_2=1$ ,  $f_n=f_{n-1}+f_{n-2}$ , dacă  $n>2$ .

Cerință:

Se dă un număr natural  $n$ . Să se afișeze în ordine crescătoare, termenii șirului lui Fibonacci mai mici sau egali cu  $n$ . Programul citește de la tastatură numărul  $n$ . Programul afișează pe ecran termenii șirului lui Fibonacci mai mici sau egali cu  $n$ , în ordine crescătoare, separați printr-un spațiu.

Restricții:  $1 \leq n \leq 500.000.000$ ;

Exemplu:

Date de intrare: 40

Date de ieșire: 1 1 2 3 5 8 13 21 34

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int n, a=1, b=1, c;
8      cin>>n;
9      if(n==1)
10         cout<<1;
11     else{
12         cout<<1<<" "<<1<<" ";
13         while (n>=b) {
14             c=a+b;
15             a=b;
16             b=c;
17             if (b<=n)
18                 cout<<b<<" ";
19         }
20     }
21
22
23     return 0;
24 }
25

```

```

40
1 1 2 3 5 8 13 21 34
Process returned 0 (0x0)   execution time : 1.374 s
Press any key to continue.

```