

### Tema: Prelucrarea șirurilor de caractere în C/C++

#### Obiective:

- să definească noțiunea de șir de caractere;
- să aplice algoritmi de diferite complexități în prelucrarea șirurilor de caractere;
- să cunoască/aplice funcții standard utilizate la prelucrarea șirurilor de caractere.

#### 9.1. Sarcină pentru soluționare

De elaborat un program care exclude din șirul-sursă un subșir.

#### 9.2. Exemplu de soluționare a sarcinii

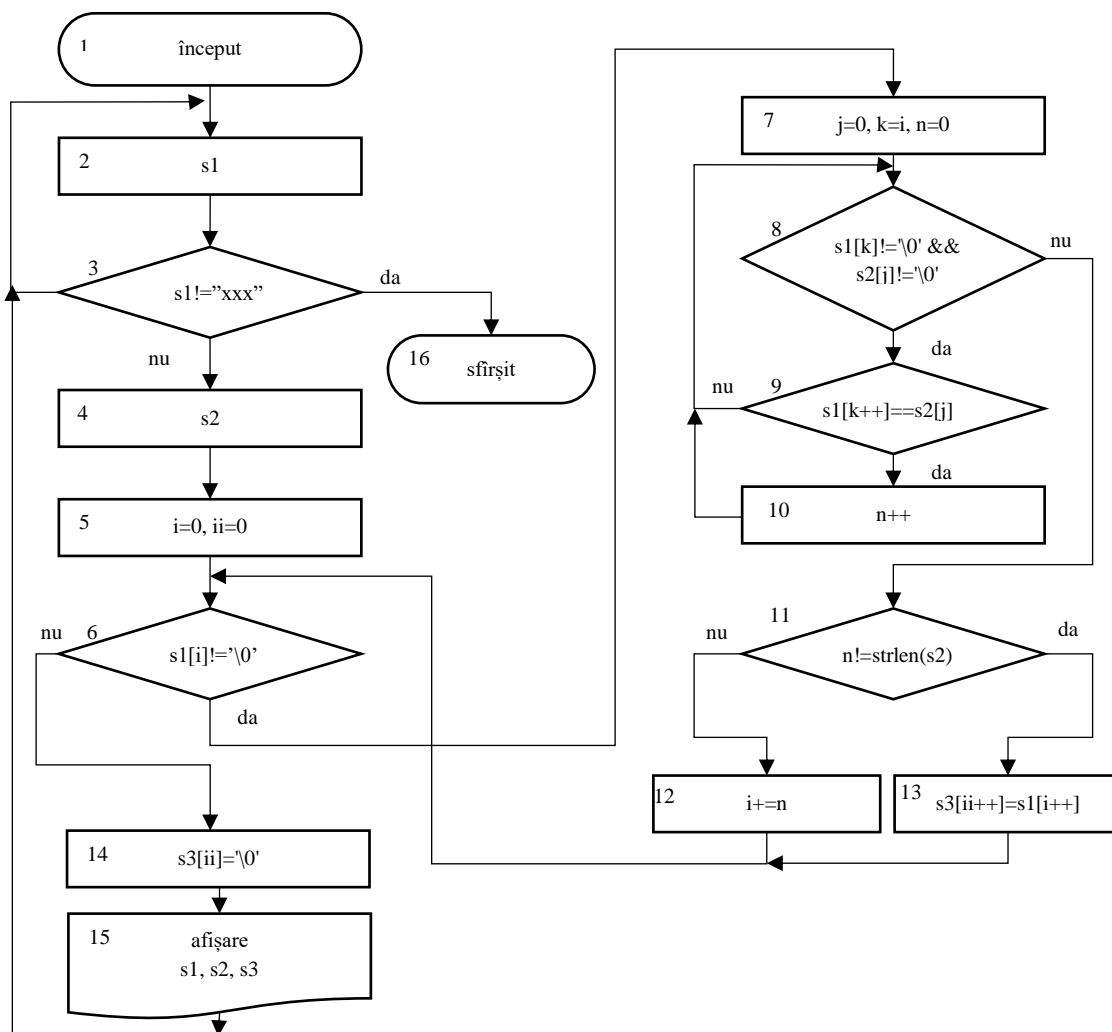
##### Elaborarea algoritmului

Algoritmul de lucru începe cu anunțarea ciclului infinit care include tot corpul funcției principale *main()* (blocurile 2-15, Figura ), organizat cu scopul de a îndeplini repetat sarcina propusă, fără a finaliza programul. Pe poziție de parametru, în ciclu, este utilizată variabila *s1*, care la fiecare iterație a ciclului se solicită să fie setată (blocul 2), valoarea setată se compară cu șirul de caractere „xxx” care este codul de finisare a programului (blocul 3). În continuare se citește subșirul (blocul 4).

Prelucrarea șirului are loc în următoarele două cicluri (blocul 5-15). Ciclul extern are ca parametru contorul *i* care ne permite să parcurgem șirul sursă (blocul 5) și contorul *ii* plasat suplimentar pentru a forma șirul final *s3*. Ciclul se întrerupe la identificarea simbolului sfârșit de șir – ‘\0’ din șirul sursă (blocul 6). Respectiv, la fiecare iterație a ciclului extern se îndeplinește integral ciclul intern (blocurile 7-10). În antetul ciclului intern se inițiază contorul *j* utilizat pentru a parcurge subșirul *s2*, parametrul *k* ia valoarea contorului *i* (se evită modificarea nedorită a valorii contorului *i*), variabila *n* se inițiază cu zero (blocul 7). Ciclul intern numără caracterele care coincid la verificarea șirului sursă și subșir pe segmentul curent (blocurile 9-10). Ciclul se finalizează când se întâlnește simbolul ‘\0’ în șirul sursă sau subșir (blocul 8).

După finisarea ciclului intern se face verificarea numărului de caracter care au coincis cu numărul de caractere din subșir (blocul 11). Dacă acestea diferă ( subșirul nu a fost găsit), caracterul curent se înscrie în variabila *s3* (blocul 13). În cazul când subșirul este găsit, contorul *i* se modifică cu *n* poziții (blocul 12) pentru a sări subșirul. Atenționăm, că contorul ciclului extern se modifică în blocurile 12-13.

La finalizarea ciclului extern, programul continuă cu plasarea simbolului sfârșit de șir pentru variabila *s3* (blocul 14) și afișarea variabilelor *s1*, *s2*, *s3*. Se revine la blocul 3, pentru a finaliza programul urmează de setat șirul “xxx”.



**Figura 9.1.** Schema bloc

### Determinarea variabilelor programului

Pentru realizarea algoritmului, sunt prevăzute trei variabile de tip șir de caractere cu dimensiunea de 80 caractere:

```
char s1[80]; //sirul
char s2[80]; //subsirul
char s3[80]; //rezultatul
```

Variabilele pentru prezentarea indicelui de poziție în șiruri la diferite etape:

```
int i, ii, j, k;
```

Variabila  $n$  de tip întreg, utilizată pentru a înregistra numărul de caractere din șirul sursă și subșir care coincid pe segmentul curent.

```
int n;
```

### Elaborarea textului programului

Textul programului începe cu declararea funcției main() și declararea variabilelor.

```
char s1[80]; //sirul
char s2[80]; //subsirul
char s3[80]; //rezultatul
```

```
int i, ii, j, k;      //indicii de pozitie
int n;               //numarul de caractere coincise
```

După declararea variabilelor, textul funcției *main()* constă dintr-un singur ciclu infinit. La fiecare iterație, mai întâi de toate, se afișează mesajul pentru introducerea șirului sursă.

```
cin.getline(s1,80);
```

Următoarea instrucțiune necesită o examinare mai detaliată:

```
if(!strcmp(s1,"xxx")) break;
```

La executarea ei, șirul *s1* se compară cu ajutorul funcției *strcmp()* cu constanta șir-"xxx". Dacă ele sunt egale, *strcmp()* returnează 0, atunci valoarea expresiei logice în operatorul condițional este TRUE și se îndeplinește ieșirea din ciclul infinit. Prin urmare, ciclul se va executa până nu va fi setat șirul "xxx".

Citirea șirurilor este organizată cu funcția *getline()*, care permite citirea șirurilor de caractere care au și caractere albe.

```
cin.getline(s2,80);
```

Procesul de formare a șirului final *s3* este organizat în blocurile 5-15, este realizată pe baza a două cicluri. Ciclul extern permite parcurgerea șirului sursă element cu element. Ciclul se finalizează odată cu stabilirea simbolului '\0'. Contorul *i* stabilit pentru parcurgerea șirului sursă în ciclul extern se inițializează cu zero, modificarea lui este organizată în blocurile 12 și 13. Deoarece modificarea contorului *i* nu este constantă și depinde de faptul dacă s-a identificat subșirul sau nu. Contorul *ii* este prevăzut pentru șirul final *s3*.

```
for(i=0, ii=0; s1[i]!='\0'; )
```

În ciclul plasat sunt inițializate mai multe variabile: variabila contor *j* - indicele de poziție pentru subșir; variabila *k* - indicele de poziție pentru șirul sursă în ciclul plasat; *n* - variabila care numără perechile de caractere ce coincid. Ciclul se finalizează când se identifică sfârșitul șirului sursă sau al subșirului.

```
for(j=0, k=i, n=0; s1[k]!='\0' && s2[j]!='\0'; j++)
```

Instrucțiunea condițională din ciclul intern verifică dacă caracterele șirului sursă coincid cu caracterele subșirului. În caz de coincidență, variabila *n* se incrementează:

```
if(s1[k++]==s2[j]) n++;
```

La ieșirea din ciclu, instrucțiunea condițională (blocul 11) verifică dacă lungimea subșirului (*strlen(s2)*) coincide cu variabila *n* (numărul de caractere care au coincis). În caz de coincidență, contorul *i* își modifică valoarea cu *+n* (blocul 12). În caz contrar, caracterul curent din șirul sursă se înscrie în șirul *s3* și contorul *i* se incrementează (blocul 13):

```
if(n!=strlen(s2)) s3[ii++]=s1[i++];
```

```
else i+=n;
```

La revenire în ciclul intern, variabila *n* se resetează (blocul 7).

Textul integral al programului este adus mai jos.

```

[*] Lab_11.cpp
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  int main(){
6      char s1[80];        //sirul
7      char s2[80];        //subsirul
8      char s3[80];        //rezultatul
9      int i, ii, j, k;    //indicii de pozitie
10     int n;               //numarul de caractere coincise
11
12     for( ; ; ) {        //ciclu infinit
13         cout<<"\n\n Seteaza sirul: ";
14         cin.getline(s1,80);
15         if(!strcmp(s1,"xxx")) break;
16         cout<<"\n Seteaza subsirul: ";
17         cin.getline(s2,80);
18
19         for(i=0, ii=0; s1[i]!='\0'; ){
20
21             for(j=0, k=i, n=0; s1[k]!='\0' && s2[j]!='\0'; j++)
22                 if(s1[k++]==s2[j]) n++;
23                 if(n!=strlen(s2)) s3[ii++]=s1[i++];
24                 else i+=n;
25
26         }
27         s3[ii]='\0';
28         cout<<"\n sirul s1: "<<s1;
29         cout<<"\n subsirul s2: "<<s2;
30         cout<<"\n sirul nou s3: "<<s3;
31     }
32     return 0;
33 }

```

**Figura 9.2.** Textul programului

### Ajustarea programului

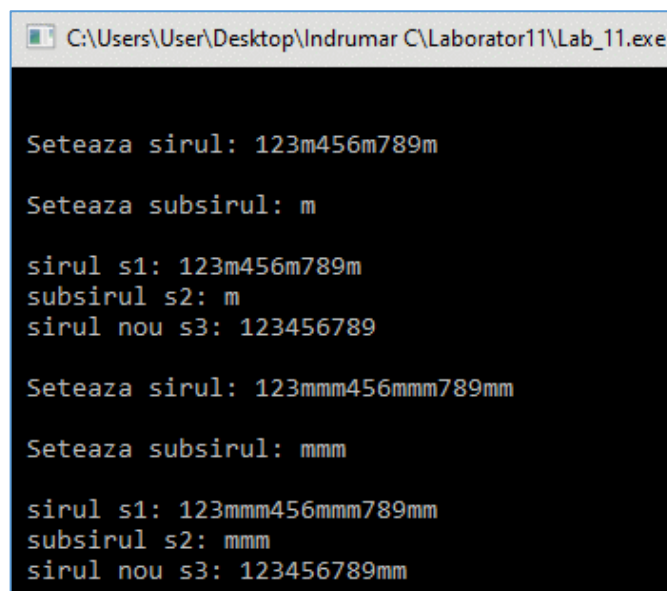
La ajustarea programului se vor urmări:

- corectitudinea calculării variabilelor  $i$  și  $n$ ;
- corectitudinea copierii simbolurilor;
- corectitudinea formării definitive a șirului rezultat (o greșeală foarte răspândită este lipsa indicelui sfârșitului de rând în șirul rezultat).

Este foarte important de ales pentru ajustare astfel de date de ieșire, care permit urmărirea corectitudinii funcționării programului pe toate ramurile algoritmului său.

### Rezultatele derulării programului

Drept rezultat al lucrului programului la ecran au fost afișate următoarele:



```
C:\Users\User\Desktop\Indrumar C\Laborator11\Lab_11.exe

Seteaza sirul: 123m456m789m

Seteaza subsirul: m

sirul s1: 123m456m789m
subsirul s2: m
sirul nou s3: 123456789

Seteaza sirul: 123mmm456mmm789mm

Seteaza subsirul: mmm

sirul s1: 123mmm456mmm789mm
subsirul s2: mmm
sirul nou s3: 123456789mm
```

Fig. 12.3. Rezultatele furnizate de program

### 9.3. Subiecte de evaluare

#### Exerciții:

1. Să se implementeze programul din exemplul prezentat.
2. Se consideră șirul de caractere  $S$ . De scris un program care dublează caracterele din șir.
3. De găsit caracterul cel mai des întâlnit în șir: primul sau ultimul caracter al șirului.
4. Propuneți un algoritm de lucru care ar permite extragerea dintr-un șir de caractere-sursă a unui subșir specificat prin poziția în cadrul șirului sursă și al numărului de caractere pe care le conține.
5. Elaborați un program care lichidează un subșir dintr-un șir de caractere dat. Subșirul se va specifica prin poziția și numărul de caractere.
6. Elaborați un program care citește  $n$  șiruri de caractere și afișează șirul cel mai lung.

#### Verificarea cunoștințelor:

1. Definiți noțiunea de șir de caractere.
2. Prezentați exemple de adresare către elementele șirului de caractere.
3. Descrieți prioritatea prezentării informației într-un șir de caractere.
4. Descrieți modul de amplasare în memorie a șirurilor de caractere.
5. Identificați operatorii ce pot fi aplicați asupra șirurilor de caractere. Prezentați exemple.
6. Se știe că un șir de caractere poate fi „tratat” ca un tablou unidimensional de tip *char*. Relatați despre avantajul lucrului cu un șir de caractere în comparație cu un tablou de tip *char*.
7. Enumerați funcțiile standard utilizate la prelucrarea șirurilor de caractere.

### 9.4. Bibliografie

1. Kris Jamsa & Lars Klander, *Totul despre C și C++*, Editura Teora, București, 2006.
2. Herbert Schildt, *C++ manual complet*, Editura Teora, București, 1999. Disponibil: <https://drive.google.com/file/d/1BrQtITgykcWk03xxtl3Q0-nvcRRFAsy7/view?usp=sharing>
3. Tutorials C++ Language. Operators. Disponibil: <https://m.cplusplus.com/doc/tutorial-ro/ntcs/>