Șiruri de caractere în C++. Clasa **string**

***Cuvinte cheie***

* șir de caractere
* lungimea șirului de caractere
* concatenarea șirurilor
* subșir de caractere
* eliminarea caracterelor
* inserarea caracterelor
* căutarea subșirurilor
* înlocuirea subșirurilor
* conversii de tip

**Ce este un șir de caractere?**

Un **șir de caractere** reprezintă o succesiune de caractere care se interpretează ca un tot întreg. În limbajul C++, șirul se referă sau la un tablou de caractere sau la ***clasa*** **string** și se declară astfel:

**char** s[20]="calculator";

**string** s;

Implicit după declarare, șirul este **vid**, deci nu conține nici un caracter.

Inițializarea uni șir de caractere poate fi făcută, prin instrucțiunea de atribuire:

s="calculator";

sau îndată la declararea șirului de caractere

**string** s="calculator";

**string** s1("calculator");

**string** s2(s);

O **constantă** ***pe șir de caractere*** în limbajul C++ se înclude între **ghilimele**.

Pentru crearea șirurilor din aceleași caractere se procedează astfel:

**string** chain(5, '-'); // (1)

chain = **string** (10, '+'); // (2)

Numărul din paranteze exprimă numărul de repetări, iar valoarea a doua – caracterul care se repetă. În rândul 1 se construiește un șir de caractere **chain** din 5 semne "*minus*", iar în cel de-al doilea valoarea variabilei **chain** se modifică într-un șir de 10 semne "*plus*".

**Citirea și afișarea unui șir de caractere**

Șirul de caractere poate fi citit de la tastatură astfel:

**cin >> s;**

Acest mod de citire are un dezavantaj mare: toate caracterele de după primul spațiu nu vor fi reținute în șirul **s**.

Pentru șirurile care conțin spații, se recomandă citirea cu funcția getline:

**getline(cin, s);**

unde:

* **cin** este sursa de unde provine fluxul de date de intrare (tastatura);
* **s** este variabila în care se va memora datele de intrare (destinația)

Afișarea unui șir de caractere se face în mod obișnuit astfel:

**cout << s;**

**Lungimea unui șir de caractere**

Pentru determinarea lungimii șirului **s** (numărul de caractere din șir) se folosește funcția membru **s.size()**. Șirul de caractere în C++ nu este un tip simplu de date (ca de exemplu **int** sau **float**), dar reprezintă un obiect care posedă anumite proprietăți (аtribute) și metode. *Metoda* semnifică o funcție de prelucrare al datelor obiectului. Scrierea **s.size()** înseamnă că metoda **size** este aplicată obiectului **s**.

Metoda **size** este definită pentru obiectele de tip **string**, apelul acesteia pentru alte tipuri de date va produce de cele mai multe ori eroare. Această metodă returnează un întreg – numărul de caractere din șir, care poate fi memorat într-o variabilă de tip întreg:

**strlen**

**int** n = s.size();

***Exemplu***:

|  |  |
| --- | --- |
| **string** sir="Welcome to C++ Programming";  cout << "Lungimea sirului este = " << sir.length() << endl;  cout << "Lungimea sirului este = " << sir.size() << endl; | Lungimea sirului este = 26  Lungimea sirului este = 26 |

**Сompararea șirurilor**

Șirurile pot fi comparate între ele la fel ca și numerele. De exemplu, verificăm egalitatea a două șiruri de caractere:

**string** password;

getline(cin, password) ;

**if** (password == "sEzAm")

cout << "Sezam deschide-te!";

**else**

cout << "No pasaran!";

Operațiile de comparare "**<**" și "**>**", funcționează la fel ca și în **Pascal** sau **Python**:

**string** s1 = "NOROCOS";

**string** s2 = "NOROIOS";

cout << (s1 < s2);

Acest fragment de program va afișa valoarea **1**, deoarece șirul **s1<s2**. Compararea se face după codul ASCII al primelor caractere distincte din șirurile comparate (codul literei 'C' este **67**, iar al literei **'I'** este **73**, **67<73🡪'C'<'I'**, deci **"NOROCOS"<"NOROIOS"**).

**Concatenarea șirurilor**

Operatorul **'+'** este folosit pentru concatenarea (alipirea sau juxtapunerea) șirurilor de caractere. Operația poartă denumirea de operație de concatenare.

***Exemplu***:

**string** s1 = "Hello";

**string** s2 = "World";

**string** rez = s1 + " " + s2 + "!";

În variabila **rez** va fi scrisă valoarea

**"Hello World!"**

**Accesarea unui caracter din șir**

Fiecare caracter din șirul de caractere are un număr de ordine (*indice* sau *poziție*) binedeterminat. Numerotarea caracterelor în C/C++ se începe ca de obicei de la **0**.

Referirea unui caracter se face prin indicele său:

**string** hello = "Salutare!";

cout << hello[1] << endl; // a

cout << hello[5] << hello[2] << "t"; // alt

**Parcurgerea unui șir de caractere**

Deoarece la un caracter din șir ne putem adresa direct prin indicele său, pentru parcurgerea șirului putem folosi o instrucțiune de ciclu cu parameru care parcurge caracter cu caracter întreg șirul:

Pentru afișarea codurilor fiecărui caracter dintr-un șir de caractere, putem proceda astfel:

**string** capitala = "Moldova Chisinau";

**for** (**int** i=0; i<capitala.size(); i++)

cout << capitala[i] << " " << **int** (capitala[i]) << endl;

Un caracter se păstrează în memorie ca un număr întreg corespunzător codului ASCII, de aceea, folosind conversia de tip int, putem obține codul caracterului curent. Începând cu versiunea C++11, ne putem lipsi și de variabila contor, pentru a parcurge șirul de caractere:

**string** capitala = "Moldova Chisinau";

**for** (**char &**ch: capitala)

cout << ch << " " << **int** (ch) << endl;

Instrucțiunea **for (char ch: capitala)** se aseamănă cu instrucțiunea de ciclu din limbajul Python care se scrie astfel:

**for ch in capitala:**

toate caracterele din șirul inițial se scriu pe rând în variabila temporară **ch**.

Spre deosebire de alte limbaje, șirurile de caractere din C++ sе consideră ***obiecte dinamice***. Prin urmare, putem modifica, schimba anumite caractere (sau toate!) direct în zona de memorie unde a fost plasat șirul după declararea sa.

***Exemplu***:

**string** s;

getline(cin,s);

**for** (**int** i = 0; i < s.size(); i++)

**if** (s[i] == 'a')

s[i] = 'e';

cout << s;

sau

// incepand cu C++11

**string** s;

getline(cin,s);

**for** (**char**& ch: s)

**if** (ch=='a')

ch='e';

***Observație***: Variabila temporară **ch** este declarată ca o referință la caracter (**char&**), de aceea ea poate fi folosită pentru modificarea caracterelor șirului **s**.

**Subșir de caractere**

Pentru extragerea unui subșir de caractere (subșir, în engleză – *substring*), în C++ se folosește metoda **substr()**.

***Exemplu*:** Următorul fragment extrage din șirul inițial **s** un subșir de cinci caractere începând cu caracterul cu indicele **3** și le depune în șirul **s1**:

**string** s = "0123456789";

**string** s1 = s.substr(3, 5);

cout << s1 << endl; // "34567"

Cele două argumente reprezintă indicele primului caracter și numărul de caractere. Dacă parametrul al doilea lipsește, atunci se extrag toate caracterele din poziția specificată până la sfârșitul șirului.

***Exemplu*:**

**string** s = "0123456789";

**string** s1 = s.substr(3); // "3456789"

**Eliminarea și inserarea**

Pentru eliminarea unui subșir de caractere, trebuie apelată metoda **erase()**, indicând indicele primului caracter de eliminat și numărul de caractere eliminate.

**string** s = "0123456789";

s.erase(3, 6); // "0129"

În șir rămâne valoarea **"0129"** (sunt eliminate **6** caractere, începând cu elementul **s[3]**). Menționăm faptul că după aplicarea metodei **erase**, șirul inițial este modificat.

Dacă parametrul al doilea lipsește, atunci sunt eliminate toate caracterele până la sfârșitul șirului.

**string** s = "0123456789";

s.erase(3); // "012"

Metoda **clear** (engl. – *curăță, elimină, șterge*) elimină toate caracterele din șir și eliberează spațiul de memorie ocupat de șir. După această metoda **size** returnează valoarea **0**, iar metoda logică **empty** (engl. – *vid*) valoarea de adevăr (**1**).

s.clear();

**if** (s.empty()) // sau if (s.size() == 0)

cout << "Șir vid!" << endl;

Pentru **inserarea** unui subșir de caractere, putem utiliza metoda **insert**. La apelarea metodei se trensmite în calitate de parametri ***subșirul de inserat***, cât și ***indicele*** *caracterului de la care se începe inserarea*.

***Exemplu*:**

**string** s = "0123456789";

s.insert(3, "ABC"); // "012АВС3456789"

Metoda sau funcția membru **insert** de asemenea modifică șirul inițial, deoarece după inserare se mărește lungimea șirului inițial **s**.

**Căutarea în șirurile de caractere**

În limbajul C++ pentru căutarea unui caracter sau al unui subșir de caractere este utilizată metoda **find**. Această funcție returnează poziția primei apariții al caracterului căutat, iar în cazul căutării unui subșir de caractere, indicele primei apariții al subșirului dat. Dacă subșirul nu este găsit, atunci, este returnată valoarea specială **string::npos** (din engleză *no position* – lipsă poziție). Convertind valoarea la tipul **int**, obținem **-1**.

***Exemplu*:**

**string** s = "Soarele de dimineata";

**int** n = s.find('с');

**if** (n!=**string**::npos)

cout <<"litera с este in pozitia: s["<< n << "]" << endl;

**else**

cout << "Nu exista litera с in sir!" << endl;

Exact în același mod este căutat un subșir de caractere:

**string** s = "Soarele de dimineata";

**int** n = s.find("rele");

**if** (n!=**string**::npos)

cout << " subsirul rele se incepe din pozitia s["<< n<< "]";

**else**

cout << "Nu exista asa sir!";

Dacă căutarea trebuie să se înceapă dintr-o poziție oarecare și nu de la începutul șirului, atunci funcției membru **find** trebuie săi transmitem și cel de-al doilea argument indicele caracterului de la care se va începe căutarea.

**string** sir1 = "Soarele de dimineata";

**int** n = sir1.find('e', 2);

cout << n << endl; // 4

Pentru căutarea de la dreapta spre stânga (începând de la sfârșit spre început) se va utiliza metoda **rfind** (engl. ***reverse find*** *–* căutare inversă) care va determina ultima apariție a subșirului dat în șirul inițial.

**string** sir1 = "Soarele de dimineata";

n = sir1.rfind('e');

cout << n << endl; // 16

**Înlocuirea**

Metoda **replace** (din engleză – *înlocuiește*) permite înlocuirea unui subșir de caractere cu alt subșir dat.

***Exemplu*:** Pentru înlocuirea a două caractere **"/"** din scrierea unei date calendaristice cu caracterul **"."**, sunt necesare două apeluri ale funcției date:

**string** date = "12/02/2018";

date.replace(2, 1, "."); // "12.02/2018"

date.replace(5, 1, "."); // "12.02.2018"

Primul argument al metodei **replace** este indicele primului caracter ce trebuie înlocuit, al doilea parametru – numărul de caractere de înlocuit, iar cel de-al treilea parametru, subșirul cu care se înlocuiește. Pentru înlocuirea tuturor intrărilor unui subșir cu altul, trebuie scrisă o secvență aparte, deoarece nu există metodă standard care să relizeze acest lucru.

***Exemplu*:**

Fie în șirul **fileName** se păstrează numele unui fișier cu extensia **.com**:

**string** fileName = "example.com";

Se cere schimbarea numelui fișierului în **spam**, lăsând extensia aceeași.

Pentru aceasta, determinăm poziția ultimului punct folosind **rfind**, apoi aplicăm metoda **replace**:

**int** pos = fileName.rfind('.');

fileName.replace(0, pos, "spam"); // "spam.com"

**Transformări șir de caractere în număr**

În limbajul C++ pentru transformarea unui șir de caractere (tipul **string**) într-un număr, putem folosi funcțiile **atoi** (*șir* 🡪 *număr întreg*) sau **atof** (*șir* 🡪 *număr real*), funcții predefinite din biblioteca **<cstdlib>**.

**#include <cstdlib>**

Funcțiile **atoi** și **atof** au fost moștenite din limbajul **C**, de aceea ele folosesc o altă formă de reprezentarea a șirurilor de caractere – secvențe de caractere care se încheie cu caracterul **'\0'** la sfârșit (*null-terminated strings*). Pentru conversia unui șir de caractere de tip **string**, în unul de tip **tablou de caractere**, trebuie de apelat funcția metodă **c\_str**.

În versiunea **C++11**, au fost introduse funcțiile **stoi** și **stof** care lucrează cu obiecte de tip **string** și care nu necesită conversia șirului de caractere la alt format.

**string** s = "12.345be-be-be6789";

**int** n = atoi (s.c\_str()); // 12

**float** x = atof(s.c\_str()); // 12.345

Procesul de conversie realizat de funcțiile **atoi** și **atof** se încheie odată cu depistarea primului caracter ce nu corespunde tipului necesar de date.

Transformarea *inversă* (din *număr* în *șir de caractere*) în **C++** se face cu ajutorul fluxurilor de ieșire care scriu rezultatul într-un șir de caractere și nu la consolă (ecran). Pentru aceasta, va fi necesar să includem biblioteca **<sstream>** și apoi să direcționăm fluxul de ieșire spre șirul de caractere.

**#include** <sstream>

**int** main()

{

ostringstream temp;

}

Aici **temp** reprezintă un obiect de tipul **ostringstream** – un flux de ieșire auxiliar, care mai apoi poate di transformat în șir de caractere folosind metoda **str**:

**int** N = 123;

temp << N;

**string** s = temp.str(); // "123"

Pentru conversia următorului număr, fluxul de ieșire trebuie curățit, scriind în el un șir vid de caractere:

temp.str(""); // curățire flux

La afișarea ***numerelor reale***, ar fi bine să utilizăm ***manipulatorii de afișare***, similari celor de la fluxul de ieșire **cout**, din biblioteca **<iomanip>**, care permite stabilirea ***numărului de poziții de afișare*** cât și ***precizia de afișare*** (numărul de zecimale în partea fracționară a numărului).

***Exemple***:

**#include** <iomanip>

...

**double** X = 123.456123;

temp << fixed << setprecision(3) << setw(10) << X; // in virgulă fixă

s = temp.str(); // " 123.456"

temp.str(""); // curatire flux

temp << scientific << setprecision (6) << setw(13) << X; // sau in virgula mobila (stiintific)

s = temp.str(); // "1.234561E+002"

În biblioteca standard C++11 există funcția predefintită **to\_string**, care transformă date numerice în șiruri de caractere:

**int** N = 123;

**string** sN = to\_string(N); // 123

**double** X = 123.45;

**string** sX = to\_string(X); // 123.450000

**Șirurile de caractere în funcții de prelucrare**

Șirurile de caractere pot fi transmise unor funcții de prelucrare a șirurilor, în calitate de argumente, rezultatul prelucrării fiind de asemenea un șir de carctere.

Să încercăm să construim un subprogram care ***înlocuiește*** toate intrările unui șubșir de caractere cu alt subșir de caractere, de exemplu **wOld** cu **wNew**.

O primă idee de algoritm al rezolvării problemei date:

**while** ( /\* cuvântrul wOld există în șirul inițial s \*/ )

{

// elimină subșirul wOld din șir

// inserează în locul lui subșirul wNew

}

Acest algoritm va funcționa incorect în cazul dacă subșirul **wOld** va fi parte componentă a lui **wNew**. De exemplu, cu ajutorul acestui algoritm, nu vom putea înlocui **"12"** cu **"A12B"** (ciclu infinit!)

Pentru eliminarea acestui neajuns major, vom acumula rezultatul în altă variabilă pe șir de caractere **result**, eliminând din șirul inițial s, partea prelucrată deja.

Inițial, **result** este vid, apoi dacă **wOld** nu există în șirul **s**, atunci șirul dat **s** este scris la sfârșitul șirului **result** și algoritmul se încheie.

result

wOld

s

result

s

s

Dacă în șirul inițial **s** a fost găsit subșirul **wOld**:

result

wOld

s

s

result

s

result

result

s

wNew

wNew

wOld

s

1. Acea parte a șirului **s**, care se află la stânga de **wOld**, se alipește la finele lui **result**;
2. Se adaugă la sfârșitul lui **result** subșirul nou **wNew**;
3. Se elimină din șirul inițial **s**, partea de început împreună cu subșirul găsit **wOld**.

Operațiile date, începând cu căutarea subșirului **wOld** se repetă până când șirul **s** ***devine șir vid***.

În tabelul de mai jos este fixat protocolul de funcționare al algoritmului dat, pentru șirul inițial **"12.12.12"**, în care trebuie schimbat **"12"**, cu subșirul **"A12B"**:

|  |  |
| --- | --- |
| **Șirul final result** | **Șirul inițial s** |
| "" | "12.12.12" |
| "А12В" | ".12.12" |
| "А12В.А12В" | ".12" |
| "А12В.А12В.А12В" | "" |

**void** replaceAll(**string** &s, **string** wOld, **string** wNew)

{

**string** result ="";

**int** lengthOld = wOld.size();

**while** (s.size()>0)

{

**int** posOld = s.find(wOld);

**if** (posOld == **string**::npos)

**break**;

result = result + s.substr(0,posOld) + wNew;

s.erase(0, posOld + lengthOld);

}

result = result + s;

s = result;

}

**Unele funcții de tip string**

|  |  |  |
| --- | --- | --- |
| **Expression** | **Effect** | **Exemplu de cod** |
| varstr.at(index) | Returnează elementul în poziția specificată de **index**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Abracadabra";  **char** ch;  **int** lung = varstr.length();  **for** (**int** i = 0; i < lung; i++)  {  ch = varstr.at(i);  cout << ch << " ";  }  **return** 0;  }  **Ieșire**:A b r a c a d a b r a |
| varstr[index] | Returnează elementul în poziția specificată de **index**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Abracadabra";  **char** ch;  **for** (**int** i = 0; i < varstr.length(); i++)  {  ch = varstr[i];  cout << ch << " ";  }  **return** 0;  }  **Ieșire**: A b r a c a d a b r a |
| varstr.append("sir", n) | Adaugă **n** caractere ale tabloului de caractere la șirul **varstr**, unde **"sir"** este o constantă pe șir de caractere (tablou de caractere, stil-C) . | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** str1("Salutare ");  cout << "Sirul initial str1: " << str1 << endl;  str1.append("Alexandru", 4);  cout << "Dupa append() : ";  cout << str1 << endl;  **return** 0;  }  **Ieșire:**  Sirul initial str1: Salutare  Dupa append() : Salutare Alex |
| varstr.append(n, ch) | Adaugă **n** copii ale lui **ch** la șirul **varstr**, unde **ch** este o variabilă sau o constantă de tip **char**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** str1("Salut!");  cout << "Sirul initial str1: " << str1 << endl;  str1.append(10,'#'); // adauga 10 caractere $  cout << "Dupa append() : ";  cout << str1 << endl;  **return** 0;  }  **Ieșire**:  Sirul initial str1: Salut!  Dupa append() : Salut!########## |
| varstr.append(str) | Adaugă subșirul de carctere **str** la șirul inițial **varstr**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** str1("Salut! ");  **string** str2("Buna ziua!");  cout << "Sirul initial str1: " << str1 << endl;  str1.append(str2); // adauga str2 in str1  cout << "Folosind append() : ";  cout << str1 << endl;  **return** 0;  }  **Ieșire**:  Sirul initial str1: Salut!  Folosind append() : Salut! Buna ziua! |
| varstr.clear() | Elimină toate caracterele din șirul **varstr**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Abracadabra";  cout << "Inaite de eliminare: ";  cout << varstr << endl;  // Elimina toate caracterele din sir  varstr.clear();  cout << "Dupa eliminare: ";  cout << varstr;  **return** 0;  }  **Ieșire**:  Inaite de eliminare: Abracadabra  Dupa eliminare: |
| varstr.compare(str) | Returnează **1** dacă **varstr>str**; **0** dacă **varstr==str**; și **-1** dacă **varstr<str**. | **#include**<iostream>  **using namespace** std;  **int** main()  {  **string** S1="NOROCOS";  **string** S2="NOROIOS";  cout << (S1.compare(S2)) << endl;  // returns a value < 0 (s1 is smaller then s2)  **if** ((S1.compare(S2))<0)  cout << S1 << " este mai mic decat " << S2 << endl;  **else**  **if** ((S1.compare(S2))>0)  cout << S1 << " este mai mare decat " << S2 << endl;  **else**  cout << S1 << " este egal cu " << S2 << endl;  **return** 0;  }  **Iesire:**  -1  NOROCOS este mai mic decat NOROIOS |
| varstr.empty() | Returnează **true/1** dacă **varstr** este vid, altfel returnează **false/0**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** s = "Elisabeth";  **string** s1 = "";  cout<<"Sirul este: "<<s<<"\n";  cout<<"Lungimea lui: "<<s.length()<<"\n";  cout<<"Sirul este vid? "<<s.empty()<<"\n\n";  cout<<"Sirul este: "<<s1<<"\n";  cout<<"Lungimea lui: "<<s1.length()<<"\n";  cout<<"Sirul este vid? "<<s1.empty();  **return** 0;  }  **Iesire:**  Sirul este: Elisabeth  Lungimea lui: 9  Sirul este vid? 0  Sirul este:  Lungimea lui: 0  Sirul este vid? 1 |
| varstr.erase() | Elimină toate caracterele din **varstr**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Ala-bala-portocala";  cout << "Inaite de eliminare: ";  cout << varstr << endl;  // Elimina toate caracterele din sir  varstr.erase();  cout << "Dupa eliminare: ";  cout << varstr;  **return** 0;  }  **Ieșire:**  Inaite de eliminare: Ala-bala-portocala  Dupa eliminare: |
| varstr.erase(pos, n) | Elimină **n** caractere din **varstr** începând cu poziția specificată **pos**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Abracadabra";  cout << "Inaite de eliminare: ";  cout << varstr << endl;  // Elimina toate caracterele din sir  varstr.erase(4,5);  cout << "Dupa eliminare: ";  cout << varstr;  **return** 0;  }  **Ieșire:**  Inaite de eliminare: Abracadabra  Dupa eliminare: Abrara |
| varstr.erase(pos); | Elimină toate caracterele din **varstr** începând de la poziția specificată prin indicele **pos**. **pos** este un parametru de tip **string::size\_type**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** varstr="Abracadabra";  cout << "Inaite de eliminare: ";  cout << varstr << endl;  // Elimina toate caracterele din sir  varstr.erase(4);  cout << "Dupa eliminare: ";  cout << varstr;  **return** 0;  }  **Ieșire:**  Inaite de eliminare: Abracadabra  Dupa eliminare: Abra |
| varstr.find(str) | Returnează indicele primei apariții a lui **str** în șirul **varstr**. Dacă **str** nu este găsit, atunci este returneată valoarea specială **string::npos**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** s = "ala-bala-portocala";  **string** subsir = "ala";  **int** poz;  poz = s.find(subsir);  **if** (poz != **string**::npos)  cout << "Prima aparitie: " << poz << endl;  **return** 0;  }  Ieșire:  Prima aparitie: 0 |
| varstr.find(str, pos); | Returnează indicele primei apariții *la* sau *după* poziția **pos** în care **str** se găsește în **varstr**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** s = "ala-bala-portocala";  **string** s1 = "ala";  **int** poz;  poz = s.find(s1, 6);  **if** (poz != **string**::npos)  cout << "Urmatoarea aparitie: " << poz << endl;  **return** 0;  }  **Ieșire:**  Urmatoarea aparitie: 15 |
| varstr.find\_first\_of(str, pos) | Returnează indicele primei apariții a oricărui caracter a lui **varstr** în **str**. Căutarea începe de la poziția specificată **pos**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main ()  {  **string** s="Acesta este un sir de caractere";  **int** poz = s.find\_first\_of("aeiou");  **while** (poz!=**string**::npos)  {  s[poz]='\*';  poz=s.find\_first\_of("aeiou",poz+1);  }  cout << s << '\n';  **return** 0;  }  **Ieșire:**  Ac\*st\* \*st\* \*n s\*r d\* c\*r\*ct\*r\* |
| varstr.find\_first\_not\_of (str, pos) | Returnează indicele primei apariții a oricărui caracter a lui **str** care nu este în **varstr**. Căutarea începe de la poziția specificată **pos**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main ()  {  **string** s="Acesta este un sir de caractere";  **int** poz = s.find\_first\_not\_of("aeiou");  **while** (poz!=**string**::npos)  {  s[poz]='\*';  poz=s.find\_first\_not\_of("aeiou",poz+1);  }  cout << s << '\n';  **return** 0;  }  **Ieșire:**  \*\*e\*\*a\*e\*\*e\*u\*\*\*i\*\*\*e\*\*a\*a\*\*e\*e |
| varstr.insert(pos, n, ch); | Inserează **n** apariții ale caracterului **ch** în poziția **pos** din șirul **varstr**; pos și n sunt de tip **string::size\_type**; iar **ch** este un caracter. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** S="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";  cout << "Sirul initial: ";  cout << S << endl;  // Insereaza 5 caractere '$' in sirul S din pozitia 8  S.insert(8, 5, '$');  cout << "Dupa inserare: ";  cout << S;  **return** 0;  }  **Ieșire:**  Sirul initial: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Dupa inserare: \*\*\*\*\*\*\*\*$$$$$\*\*\*\*\*\*\*\* |
| varstr.insert(pos, str); | Inserează toate caracterele șirului **str** în poziția **pos** din șirul dat **varstr**. | **#include**<iostream>  **#include**<**string**>  **using namespace** std;  **int** main()  {  **string** S1="Imi place mult";  **string** S2=" Informatica";  cout << "Sirul original: ";  cout << S1 << endl;  // Inserarea sirului S2 in sirul S1  S1.insert(9, S2);  cout << "Dupa inserare: ";  cout << S1;  **return** 0;  }  **Iesire:**  Sirul original: Imi place mult  Dupa inserare: Imi place Informatica mult |
|  |  |  |
| varstr.length() | Returnează valoarea de tip **string::size\_type** care reprezintă numărul de caractere din **varstr**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** s = "Saint Elisabeth";  cout << s.size() << " ";  cout << s.length() << " ";  cout << strlen(s.c\_str()) << " ";  // cat timp nu am ajuns la sfarsit de sir (NOT NULL)  **int** i = 0;  **while** (s[i])  i++;  cout << i << " ";  **for** (i = 0; s[i]; i++)  ;  cout << i << endl;  **return** 0;  }  **Iesire:**  15 15 15 15 15 |
| varstr.replace(pos, n, str); | Începând de la indicele **pos**, înlocuiește următoarele **n** caractere ale **varstr** cu toate caracterele din **str**. Dacă **n>lungimea varstr**, atunci toate caracterele până la sfârșitul lui **varstr** sunt înlocuite. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** s1 = "Functia de inlocuire";  **string** s2 = "Demonstrare";  **string** s3 = " Ionela";  **string** s4 = "Noroc!";  *// inlocuire 10 caractere din s1 cu s2*  s1.replace(0, 10, s2);  cout << s1 << endl; *// Demonstrare inlocuire*  *// inlocuire 5 caractere din s1 cu "Salut"*  s4.replace(0, 5, "Salut");  cout << s4 << endl; *// Salut*  *// inlocuire 5 caractere din s4 incepand cu indicele 6*  *// cu 7 caractere din s3, incepand cu 0*  s4.replace(5, 6, s3, 0, 7);  cout << s4 << endl; *// Salut Ionela*  *// inlocuire 6 caractere din s4 de la indicele 6*  *// cu 4 caractere din sirul "Alexandru"*  s4.replace(6, 6, "Alexandru", 4);  cout << s4 << endl; *// Salut Alex*  *// inlocuire caracter din pozitia 9 cu 3 caractere '!'*  s4.replace(9, 1, 3, '!');  cout << s4 << endl; *// Salut Ale!!!*  **return** 0;  }  **Ieșire:**  Demonstrare inlocuire  Salut!  Salut Ionela  Salut Alex  Salut Ale!!! |
| varstr.substr(pos, len) | Returnează un șir care este un șir de caractere al **varstr** începând cu poziția specificată **pos**. Lungimea șirului este cel mult de **len** caractere. Dacă **len** este prea mare, înseamnă "până la capătul" șirului din **varstr**. | **#include** <iostream>  **#include** <**string**>  **using namespace** std;  **int** main()  {  **string** s ("Tipul string in limbajul C++");  cout << s.substr(6) << std::endl;  **string** s1 ("Abracadabra");  cout << s1.substr(3,4) << std::endl;  **return** 0;  }  **Ieșire:**  string in limbajul C++  acad |
| varstr.size() | Returnează valoarea de tip **string::size\_type** care reprezintă numărul de caractere din **varstr**. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  **string** S;  S = "Limbajul de programare C++";  cout << S.size() << endl;  **return** 0;  }  **Ieșire:**  26 |
| varstr.swap(str1); | Interschimbă conținutul lui **varstr** și **str1**. **str1** este o variabilă de tip șir de caractere. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  string s1 = "abracadabra";  string s2 = "ala-bala-portocala";  cout << "Sirul s1 inainte de interschimbare: ";  cout << s1 << endl;  cout << "Sirul s2 inainte de interschimbare: ";  cout << s2 << endl;  s1.swap(s2);  cout << "Sirul s1 dupa interschimbare: ";  cout << s1 << endl;  cout << "Sirul s2 dupa interschimbare: ";  cout << s2 << endl;  return 0;  }  **Ieșire:**  Sirul s1 inainte de interschimbare: abracadabra  Sirul s2 inainte de interschimbare: ala-bala-portocala  Sirul s1 dupa interschimbare: ala-bala-portocala  Sirul s2 dupa interschimbare: abracadabra |
| getline(istreamVar, varstr); | **istreamVar** este o variabilă al fluxului de intrare (de tip **istream** sau **ifstream**). Caracterele până la caracterul linie nouă sunt introduse din **istreamVar** și stocate în **varstr**. (Însăși caracterul linie nouă este citit, dar nu este stocat în **varstr**) Valoarea returnată de această funcție este de obicei ignorată. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  **string** S;  getline(cin, S);  cout << "Acesta este sirul citit: ";  cout << S << endl;  **return** 0;  }  **Ieșire:**  Abracadabra  Acesta este sirul citit: Abracadabra |
| varstr.c\_str() | Returnează un pointer (adresa de bază) la un șir de caractere în stil C cu terminator nul **'\0'**, care corespunde caracterelor din **varstr**. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  **string** s1 = "Tehnologii Informationale";  **if** (s1.size() == strlen(s1.c\_str()))  cout << "s1.size este egal cu strlen(s1.c\_str()) " << endl;  **else**  cout << "s1.size este diferit de strlen(s1.c\_str())" << endl;  cout<< s1.c\_str() << endl;  cout << "Caracterul cu indicele 11 din sirul " << s1  << " este " << s1.c\_str()[11] << endl;  **return** 0;  }  **Ieșire:**  s1.size este egal cu strlen(s1.c\_str())  Tehnologii Informationale  Caracterul cu indicele 11 din sirul Tehnologii Informationale este I |
| varstr.capacity() | Returnează dimensiunea zonei de memorie alocate pentru **varstr** în octeți. Această capacitate nu este neapărat egală cu lungimea șirului. Poate fi egal sau mai mare, cu spațiul suplimentar care permite obiectului să-și optimizeze operațiunile atunci când sunt adăugate caractere noi în șir. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  **string** sir = "Tehnologii Informationale";  cout << "Sirul initial este: ";  cout << sir << endl;  cout << "Capacitatea sirului este: ";  cout << sir.capacity() << endl;  cout << "Lungimea sirului este: "<< sir.length()<< endl;  **return** 0;  }  **Ieșire:**  Sirul initial este: Tehnologii Informationale  Capacitatea sirului este: 25  Lungimea sirului este: 25 |
| varstr.resize(n, ch); | Schimbă dimensiunea zonei de memorie allocate pentru **varstr** la **n**. Dacă **n** este mai mic decât dimensiunea curentă de stocare a **varstr**, dimensiunea de stocare a șirului este trunchiată la **n**. Dacă **n** este mai mare decât dimensiunea curentă de stocare, șirul este extins la dimensiunea **n** și spațiul suplimentar este umplut cu copii ale caracterului specificat de variabila char **ch**. | **#include** <iostream>  **#include** <**string**>  **int** main()  {  **string** sir = "Tehnologii Informationale";  cout << "Sirul initial este: ";  cout << sir << endl;  *// Modificam lungimea cu resize()*  sir.resize(15);  cout << "Capacitatea sirului este: ";  cout << sir.capacity() << endl;  cout << "Lungimea sirului este: "<< sir.length()<< endl;  cout << "Sirul este: ";  cout << sir << endl;  **return** 0;  }  **Ieșire:**  Sirul initial este: Tehnologii Informationale  Capacitatea sirului este: 25  Lungimea sirului este: 25  Sirul este: Tehnologii Info |
|  |  |  |
|  |  |  |