Capitolul 6 – Prelucrarea șirurilor de caractere în C++

În C++, șirurile de caractere pot fi manipulate în două moduri:

- ca tablouri de caractere (stil C),
- prin clasa string din biblioteca standard (#include <string>).
-

A. Stil C – tablouri de caractere

```
char nume[20] = "Ana";
```

Acest șir este de tipul char, adică un tablou de caractere, terminat cu caracterul NULL ('\0').

B. Stil modern – clasa string

```
#include <string>
using namespace std;

string prenume = "Ion";
```

Clasa string oferă metode utile pentru manipularea șirurilor.

---

2. Funcții utile pentru șiruri în stil C (biblioteca <cstring>)

a) strlen() – determină lungimea unui șir

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
   char text[] = "Programare";
   cout << "Lungimea sirului este: " << strlen(text);
   return 0;
}
```

Explicații:

Funcția strlen(text) returnează numărul de caractere din șir, fără a include caracterul '\0' de final.

---

b) strcpy() – copiază un șir într-altul

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char sursa[] = "C++";
    char destinatie[10];
    strcpy(destinatie, sursa);
    cout << "Sir copiat: " << destinatie;
    return 0;
}
```

Explicații:

Funcția strcpy(destinatie, sursa) copiază caracterele din sursa în destinatie. Asigură-te că destinatie are spațiu suficient!

---

c) strcmp() – compară două șiruri

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char a[] = "Ana";
    char b[] = "Ion";
```

```
    if (strcmp(a, b) == 0)
        cout << "Sirurile sunt egale.";
    else
        cout << "Sirurile sunt diferite.";

    return 0;
}
```

Explicații:

strcmp() returnează:

- 0 dacă șirurile sunt egale,
- valoare negativă dacă primul e "mai mic" lexicografic,
- valoare pozitivă dacă e "mai mare".

---

d) strcat() – concatenează două șiruri

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char mesaj[50] = "Bine ai venit, ";
    char nume[] = "Ion!";

    strcat(mesaj, nume);
    cout << mesaj;
    return 0;
}
```

Explicații:
strcat(mesaj, nume) adaugă șirul nume la finalul șirului mesaj. Spațiul alocat lui mesaj trebuie să fie suficient pentru a conține concatenarea.

---

3. Clasa string din C++ – metode esențiale

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string nume = "Popescu";
   cout << "Lungime: " << nume.length() << endl;
   cout << "Primul caracter: " << nume[0] << endl;
   nume += " Ion";
   cout << "Complet: " << nume;
   return 0;
}
```

Explicații:

- nume.length() returnează lungimea șirului.
- nume[0] accesează primul caracter.
- Operatorul += permite concatenarea ușoară a două șiruri.

---

4. Alte metode utile pentru string

Metodă	Descriere
s.length()	Lungimea șirului
s.substr(p, l)	Subșir de la poziția p, lungime l
s.find("text")	Caută textul și returnează poziția
s.erase(p, l)	Șterge l caractere de la poziția p
s.insert(p, "abc")	Inserează șirul "abc" la poziția p

5. substr(pos, len) – extrage un subșir

cpp

CopyEdit

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string mesaj = "Salutare, elev!";
    string sub = mesaj.substr(0, 8);
    cout << "Subsir extras: " << sub;
    return 0;
}
```

Explicații:

Funcția substr(0, 8) extrage 8 caractere începând de la poziția 0. În acest caz, rezultatul va fi "Salutare".

---

6. find() – caută prima apariție a unui text

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string text = "Programarea este frumoasa";
    size_t poz = text.find("este");

    if (poz != string::npos)
        cout << "'este' gasit la pozitia: " << poz;
    else
        cout << "Textul nu a fost gasit.";

    return 0;
}
```

Explicații:

find("este") returnează poziția primei apariții a cuvântului "este". Dacă nu este găsit, funcția returnează string::npos.

---

7. replace(pos, len, text) – înlocuiește o porțiune din șir

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string fraza = "Imi place matematica.";
   fraza.replace(8, 10, "informatica");
   cout << fraza;
   return 0;
}
```

Explicații:

Înlocuiește 10 caractere începând de la poziția 8 cu textul "informatica". Rezultatul: Imi place informatica.

---

8. erase(pos, len) – șterge o parte din șir

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string propozitie = "Ana are mere rosii";
   propozitie.erase(8, 5);
   cout << propozitie;
   return 0;
```

```
}
```

Explicații:

Funcția erase(8, 5) șterge 5 caractere începând de la poziția 8, rezultând "Ana are rosii".

---

9. insert(pos, text) – inserează un text la o anumită poziție

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string mesaj = "Salut!";
    mesaj.insert(5, " tuturor");
    cout << mesaj;
    return 0;
}
```

Explicații:

Funcția insert(5, " tuturor") inserează șirul " tuturor" după caracterul de pe poziția 5. Rezultat: "Salut tuturor!"

---

10. append(text) – adaugă text la finalul unui șir

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string nume = "Popescu";
    nume.append(" Ion");
    cout << "Numele complet: " << nume;
```

```
    return 0;
}
```

Explicații:

Metoda append() funcționează ca și operatorul += și adaugă textul la finalul șirului.

---

11. compare() – compară două șiruri (alfabetic)

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string a = "Ana";
    string b = "Ion";

    if (a.compare(b) == 0)
        cout << "Sirurile sunt egale.";
    else if (a.compare(b) < 0)
        cout << "Ana vine inainte de Ion.";
    else
        cout << "Ana vine dupa Ion.";

    return 0;
}
```

Explicații:

Metoda compare() returnează:

- 0 dacă șirurile sunt egale,
- <0 dacă a < b,
- 0 dacă a > b (lexicografic, adică ordine alfabetică).