**PROGRAMMING**

**Șiruri de caractere – abordarea cu clasa „string”**

Declarare variabilă:

**string nume\_variabilă;**

Constantele de tip string se scriu între ghilimele.

Ex:

**string a;**

**a="Ana are mere";** - i-am atribuit variabilei a un șir de caractere

Pentru a afla numărul de caractere dintr-o variabilă de tip string:

**variabila.size();**

Pentru a accesa UN anumit caracter din acel string:

**variabila[indice];** unde indicii încep de la 0!

Pentru a scrie o constantă care este UN singur caracter (de exemplu dacă vrem să schimbăm o literă din string sau să o comparăm) caracterul se scrie între apostrofuri.

Ex:

**s[0]='@';**

De exemplu, dacă vrem să schimbăm TOATE caracterele **'a'** (literă mică) cu **'@'** dintr-un string:

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

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

**s[i]='@';**

Concatenarea (alipirea) a două stringuri se face cu operatorul +.

Ex:

**string q="ele";**

**q=q+"fant";**

**cout<<q;**

**Citirea** unui string:

Se poate face și cu **cin>>**, însă va citi doar șirul de caractere până la primul spațiu.

Pentru a citit tot șirul de caractere care se introduce până se dă Enter:

**getline(cin,variabila\_string);**

**Pentru a căuta un subșir în string (inclusiv un subșir format dintr-un singur caracter)**

Se face prin

**string.find(subsir);**

funcția asta întorcând indicele PRIMEI apariții ale subsirului in string.

Dacă nu-l găsește, întoarce **-1**.

**Pentru a extrage(obține) un subșir dintr-un string**

**string.substr(indice,lungime);**

Ex:

**string s="Ana are mere";**

**cout<<s.substr(4,3);//afișează "are"**

**Pentru a șterge un subșir dintr-un string**

**string.erase(indice,lungime);**

Ex:

**string s="Ana are mere";**

**s.erase(4,4)**

**cout<<s;//afișează "Ana mere"**

**Conversii**

Prin conversie se înțelege transformarea (uneori aproximativă, cu pierderi) de la un tip de date la alt tip de date.

Știm deja:

**(int)val\_reală** convertește la int, cu pierderi - se pierd zecimale

**(double)val\_întreagă** converște la val. reală - ea capătă zecimale, care sunt .0

Exact la fel, putem avea conversii între string și tipurile de date numerice.

De la orice tip de dată numerică la string: folosim funcția

**to\_string(dată\_numerică)**

De la string la număr întreg:

**stoi(string);** - convertește cât poate. Dacă NU poate, dă eroare

De la string la număr real:

**stof(string);** - convertește cât poate. Dacă NU poate, dă eroare

**Tipul char**

Memorează UN SINGUR caracter.

Acest lucru se face de fapt printr-un număr - există un standard numit ASCII (American Standard Character Information Interchange) care asociază fiecărui caracter un număr.

Pentru început să știți că literele sunt în ordine alfabetică, cifrele în ordine numerică, codurile ASCII importante fiind:

**'A' = 65 ('B' = 66)**

**'a' = 97**

**'0' = 48 ('1' = 49)**

**' ' = 32** (caracterul spațiu)

Astfel, dacă vrem să transformăm un caracter cifră la valoarea care îi corespunde trebuie să scădem din codul său ASCII valoare 48.

La ACSL foarte adesea primim un șir cu multe numere și suntem nevoiți să le extragem, unul câte unul.

De exemplu, dacă șirul de caractere este

**s="132 5 -14 80"** vrem să extragem separat aceste numere, unul câte unul.

Schema pe care o propunem este una repetitivă, care la fiecare pas:

- determină indicele primului spațiu

- extrage cu substr numărul și îl convertește la int

- șterge numărul cu tot cu spațiul de după el.

Schema va face acest lucru până când șirul de caractere devine vid (putem verifica asta prin **s==""**).

Ca să NU tratăm separat ultimul număr din șir (care are particularitatea că după el NU se mai află niciun spațiu) punem noi un spațiu la sfârșitul stringului.

Schema va fi asta:

fie s=stringul cu șirul de numere

**s=s+" ";**

**while(s!="")**

**{**

**poz=s.find(" ");**

**numarul = stoi(s.substr(0,poz));**

**s.erase(0,poz+1);**

**}**

Dacă avem vreo suspiciune că între două numere din string s-au băgat mai multe spații, mai punem un if care verifică faptul că poz este diferit de 0:

**s=s+" ";**

**while(s!="")**

**{**

**poz=s.find(" ");**

**if(poz)**

**{**

**numarul = stoi(s.substr(0,poz));  
 }**

**s.erase(0,poz+1);**

**}**

**SHORTS**

**Funcții recursive**

Este un capitol mai ușor - rezolvarea lor se face aproximativ ca la matematică - plecăm de la valoarea care se cere și pe baza ei calculăm valorile funcției după reguli, până când ajungem la rezultat.

Există și cazuri în care poate fi mai simplu dacă facem o abordare de jos în sus: adică în loc să plecăm de la valoarea dată până ajungem la valori mici, plecăm de la valori mici până la cea dată.

Exerciții:



