**Șiruri de caractere (stringuri)**

Un șir de caractere se declara pur și simplu ca și vector de char, însă este dotat cu proprietăți suplimentare față de un vector cu elemente numerice.

Mai precis, o serie de instrucțiuni tratează stringul ca un "tot".

Spre exemplu, poate fi inițializat cu un șir particular de caractere "dintr-un foc" fără a face acest lucru caracter cu caracter.

**Tipul char**

Permite memorarea unui singur caracter. În memorie orice caracter este reprezentat prin un număr cuprins între 0..255 care se numește codul său ASCII (American Standard Character Information Interchange)

O variabilă de tip char este duală: poate fi precizată atât prin numărul care reprezintă codul ASCII fie prin constanta caracter respectivă, adică acel caracter delimitat de **apostrofuri**.(!!!)

Exemplu:

char c;

c='A'; //daca am scrie c=65 e fix acelashi lucru

cout<<c;//se afișează A (care btw are codul 65)

c++;

cout<<c;//se afișează B

cout<<(int)c;//se afișează 66

c=100;

cout<<c;//se afișează d - caracterul cu codul 100

Obs: Tipul char, d.p.d.v. numeric, memorează de fapt valori cuprinse între -128..127.

Tipul unsigned char, d.p.d.v. numeric memorează valori cuprinse între 0..255

Obs: Există câteva constante caracter speciale, numite "secvențe escape". Ele sunt cele care încep cu '\'. Cele mai uzuale sunt:

\n = rând nou

\t = tab

\\ = backslash

\' = apostrof

\" = ghilimele

Obs:

Dacă avem variabila c de tip char, atribuirile următoare:

c='A'; c=(char)'A'; c=(int)'A'; c=65; c=(char)65; c=(int)65;

fac ABSOLUT același lucru

Iată o schiță a tabelei ASCII:

|  |  |  |
| --- | --- | --- |
| Codul | Caracterele | Obs |
| 0 | Marca de sfârșit de string | se poate scrie și ca \0 |
| 1..31 | Coduri neafișabile (de fapt unele dintre ele sunt afișabile, sau se comportă diferit - la citire pot însemna ceva, la afișare altceva) | de exemplu \n = 10  \a = 7 (codul pt. beep)  codul 27 = la citire, tasta ESC |
| 32 | Spațiul: ' ' |  |
| 33..47 | !"#$%&'()\*+,-./ | Semne |
| 48..57 | 0123456789 | Cifrele |
| 58..64 | :;<=>?@ | Semne |
| 65..90 | ABCDEFGHIJKLMNOPQRSTUVWXYZ | Literele mari ale alfab. englez |
| 91..96 | [\]^\_` | Semne |
| 97..122 | abcdefghijklmnopqrstuvwxyz | Literele mici ale alfab. englez |
| 123..127 | {|}~⌂ | Semne |

Obs: Cele mai mari ca 127 fac parte din codul ASCII extins.

Dacă le memorăm pe tipul char, ele se iau de la cele negative (-128, -127, ...)

Codul ASCII extins depinde de setările device-ului care îl utilizează.

Compararea a două caractere se face prin operatorii obișnuiți (<, <=,...). La comparare se ia după codul ASCII. Astfel, literele mari sunt mai mici decât literele mici.

**Citirea unui caracter:**

1) Doar caractere nealbe:

cin>>caracterul;

Dacă la o astfel de citire îi dăm spațiu, Tab sau Enter, NU va citi, ci stă după noi până introducem un caracter.

2) Orice caracter (inclusiv \n):

caracterul=cin.get();

Deja putem concepe algoritmi care să citească șiruri de caractere fără a folosi efectiv structuri de tip string. Condiția este ca datele să poată fi prelucrate caracter cu caracter, în sensul că citirea unui nou caracter îl distruge pe cel anterior:

c=cin.get();

while(c!='\n')

{

..prelucrăm c..

c=cin.get();

}

**Obs:** În cazul citirii din fișier, dacă la sfârșitul liniei NU este prezent un Enter, citirea cu cin.get() va întoarce valoarea -1. Pentru că pe pbinfo citirea se face din fișiere, programele făcute în stilul de mai sus trebuie să aibă condiția de oprire **while(c!='\n' && c!=-1)**

**Operații cu caractere**

Orice operație aritmetică implicând date de tip char are REZULTAT NUMERIC.

Dacă avem nevoie de caracterul care are codul dat de respectivul rezultat, folosim operatorul (char)rezultat.

!! Dacă unei variabile de tip char îi atribui o valoare numerică, NU mai trebuie folosit operatorul (char) !!

Ex:

char c='A',d;

cout<<'A'+1;//afișează 66

cout<<(char)('A'+1);//afișează B

d='A'+1;

cout<<d;//afișează B

cout<<(int)d;//afișează 66

!!Atenție la caracterele cifră, al căror cod ASCII NU coincide cu cifra!!

Dacă dorim să convertim un caracter cifră la cifra corespunzătoare:

caracterul\_cifră – '0'

Invers, dacă dorim să convertim o cifră la caracterul cifră corespunzător:

(char)(cifra+'0')

**Funcții cu char-uri (pt. o parte dintre ele treb. inclus #include<cctype>)**

toupper(char) – dacă char este o literă mică, întoarce codul ASCII (!număr!) al său,

transformat la litera mare corespunzătoare. În rest, întoarce codul

ASCII al caracterului neschimbat.

Ex:

cout<<toupper('a'); //afișează 65

cout<<(char)toupper('a'); //afișează A

cout<<toupper('2'); //afișează 50

cout<<(char)toupper('2'); //afișează 2

char c;

c=toupper('2');

cout<<c; //afișează 2

tolower(char) – analog, de la literă mare la literă mică (dacă e cazul)

isdigit(char) – întoarce 1 dacă char este un caracter cifră, 0 în caz contrar.

isalpha(char) – întoarce 1 dacă char este literă mare, 2 dacă e literă mică, 0 în caz

contrar

isupper(char) – întoarce 1 dacă char este literă mare, 0 în caz contrar

islower(char) – întoarce 2 dacă char este literă mică, 0 în caz contrar

Să vă feriți în cazul celor de mai sus să folosiți comparații exacte, gen:

**~~if(isdigit(c)==0)~~**

Ci folosiți forme de genul:

**if(isdigit(c)) ⇔ if(c>='0' && c<='9')**

SAU

**if(!isdigit(c))**

Obs: ultimele 4 funcții de mai sus NU sunt în programa de bac.

Dacă le utilizați, scrieți lângă ele și un mic comentariu de genul:

//funcția ... folosește la verificarea caracterului ... dacă este ....