**Formarea unui șir caracter cu caracter**

După cum am văzut, în C++ un șir de caractere este format din caracterele care îl compun după urmează caracterul special - marca de sfârșit de string - caracterul nul (de cod ASCII 0).

De exemplu, dacă vrem să formăm șirul de caractere "Lia" putem face astfel:

char s[10]; //în el se pot pune maxim 9 caractere

**s[0]='L'; //care este același lucru cu s[0]=76;**

**s[1]='i'; //care este același lucru cu s[1]=105;**

**s[2]='a'; //care este același lucru cu s[2]=97;**

**s[3]='\0'; //care este același lucru cu s[3]=0;**

Dacă vrem să facem acest lucru automat, preluând caracterele din altă parte, folosim o variabilă (fie ea **nc** = nr.caractere) în care menținem numărul de caractere de la pasul curent din șir.

Astfel, formarea are următoarele etape:

1. Ințializare: **nc=0;**

2. Adăugarea repetată de caractere la variabila șir de caractere (fie ea "**s**"):

 **s[nc++]=caracterul dorit;**

3. De făcut obligatoriu la final: **s[nc]=0;** - adică punem marca de sfârșit de string

!! NEVER EVER NU AFIȘAȚI UN ȘIR DE CARACTERE PE FOR !!

**Funcții care folosesc șiruri de caractere**

Majoritatea se găsesc în biblioteca **<cstring>**. Iată-le pe cele mai uzuale:

\* **strchr(adr\_șir\_caractere,caracter)** - această funcție caută prima apariție a lui "caracter" în șirul de la **adr\_șir\_caractere** și, în cazul în care îl găsește, întoarce adresa de memorie a acestuia. În cazul în care NU îl găsește întoarce o constantă specială, NULL = adresă de memorie nulă. A NU SE CONFUNDA NULL cu caracterul nul.

!! Dacă un cout ajunge să primească această constantă, din acel moment afișările se invalidează !! (programul NU crapă, dar NU mai afișează nimic) !!

Ex:

**char s[10]="Teleferic";**

**cout<<strchr(s,'f'); //feric**

**char \*p=strchr(s,'f');///am dus pointerul peste litera 'f' din feric**

**cout<<p<<"\n";//feric**

**cout<<\*p;//e fix același lucru cu p[0] - deci afișează 'f'**

**cout<<\*(p+2);//e fix același lucru cu p[2] - deci afișează 'r'**

**cout<<s+4;///afișează 'feric' - deci s+4 este același lucru cu p**

**cout<<p-s;///afișează 4 de tip int (s+4=p => p-s=4)**

!! Deci, scăzând doi pointeri obținem diferența de adrese de memorie dintre ele !!

În cazul unui șir de caractere, prin expresia

**strchr(s,char)-s** se obține de fapt indicele (poziția) char-ului în cadrul string-ului, DOAR ÎN CAZUL ÎN CARE char apare în șir. Dacă nu apare, expresia este eronată !!

Putem folosi funcția strchr pentru a verifica, mai simplu, dacă un caracter aparține unui grupușor restrâns de caractere.

De exemplu, dacă folosim

**if(strchr("aeiouAEIOU",caracter)!=NULL)** - putem verifica dacă caracter este vocală. Putem renunța la !=NULL, acesta fiind subînțeles dacă îl omitem.

**if(strchr("aeiouAEIOU",caracter)==NULL && isalpha(caracter))** - putem verifica dacă "caracter" este consoană.

\* **strstr(șir\_în\_care\_căutăm,șir\_pe\_care\_căutăm)** - are aceeași ordine a parametrilor ca strchr, doar că face căutarea unui subșir de caractere în alt subșir de caractere.

Ex:

**cout<<strstr("lalelele mele","ele"); -> "elele mele"**

Ex:

**char s[100]="lalelele mele",\*p;**

**p=strstr("lalelele mele","ele");**

**cout<<\*p; // e <=> same shit cu p[0]**

**cout<<p-s; // 3**

**p=strstr(p+1,"ele");**

**cout<<p-s; // 5**

**cout<<p[4]; //m**

**\*strcpy(adr\_destinație,adr\_sursă)** - copiază caracterele de la adr\_destinație la adr\_sursă. Copierea se termină când la adr\_sursă dă de caracterul nul, pe care îl copiază de asemenea.

!!Această funcție a fost reimplementată folosind arhitecturile paralele de calcul (procesoare cu mai multe nuclee, astfel copierea caracterelor se face în paralel!!

De exemplu:

**char s[100];**

**strcpy(s,"telecomand");**

Dacă lucrează pe un sistem cu un procesor cu 2 nuclee stringul sursă se împarte în 2 ("telec" și "omand") și literele se copiază ÎN PARALEL! (în același timp) adică:

s[0]='t' se face în același timp cu s[5]='o'

s[1]='e' se face în același timp cu s[6]='m'

Din acest motiv (lucru pe care îl găsiți și în documentația de C++) "the size of the array pointed by destination shall be long enough to contain the same C string as source (including the terminating null character), and should not overlap in memory with source."

În manualele mai vechi (chiar și în subiectele de bac din 2009), în care funcția asta NU era optimizată pt. arhitecturi paralele, ci funcționa de fapt pe un for, veți găsi foarte adesea apeluri de genul:

**strcpy(s+i,s+i+1)** menite să șteargă caracterul de la indicele i.

**!!! ACEASTĂ FORMĂ A DEVENIT ÎNTRE TIMP ILEGALĂ !!!**

Dacă chiar vrem să ștergem caracterul de la indicele i:

- fie mutăm caracterele manual:

 **for(j=i;s[j];j++)s[j]=s[j+1];**

- fie folsim o formă asemănătoare celei devenite ilegale DAR trecând prin altă variabilă:

 **char s[100],aux[100];**

 **strcpy(aux,s+i+1);**

 **strcpy(s+i,aux);**

Ex:

**char s[100]="iepuri",q[100]="rechin";**

**strcpy(s+5,q+1);**

**cout<<s; ///iepurechin**

**\*strncpy(adr\_destinație,adr\_sursă,nr\_caractere)** - copiază primele nr caracterele de la adr\_sursă la adr\_destinație. După caracterele copiate NU pune marca de final de string!

Ex:

**char s[100]="farmacie",q[100]="un cintezoi";**

**strncpy(s+2,q+5,4);**

**cout<<s; //fantezie**

Pe când:

**char s[100]="farmacie",q[100]="ntez";**

**strcpy(s+2,q);**

**cout<<s; //fantez**

Obs: este SINGURA funcție care NU pune marca de final de string.

**\*strcat(adr\_destinație,adr\_sursă)** - concatenează (lipește la final) după stringul din adr\_destinație stringul din adr\_sursa.

Ex:

**char s[100]="pisica",q[100]="camila";**

**strcat(s,q+2);**

**cout<<s; ///pisicamila**

Ex:

**char s[100]="";**

**strcat(s,"Ana");**

**strcat(s," ");**

**strcat(s,"are");**

**strcat(s," ");**

**strcat(s,"porodici");**

A ține seama de această tehnică, de formare progresivă a unui șir de caractere (concatenăm rând pe rând toate elementele constitutive)

**\*strncat(adr\_destinație,adr\_sursă,nr\_caractere)** - concatenează (lipește la final) după caracterele din adr\_destinație doar primele nr\_caractere din adr\_sursă

Ex:

**char s[100]="cami",q[100]="lacusta verde";**

**strncat(s,q,7);**

**cout<<s;///camilacusta**

**\*strcmp(string1,string2);**

compară lexicografic (ca în dicționar) cele 2 stringuri.

Dacă string1 este în dicționar înainte de string2 întoarce -1,

dacă-s egale întoarce 0

dacă string1 e DUPĂ string2 întoarce 1.

Ex:

**strcmp("mama","bunica"); -> 1**

**strcmp("cor","condiment"); -> 1**

**strcmp("cor","coriandru"); -> -1**

**strcmp("cor","cor"); -> 0**

(ca să țineți minte: vă gândiți că dacă am scrie o comparație ipotetică de genul

**if(string1<string2)** aceasta ar fi echivalentă cu comparația "ilegala" sintactic

**if(string1-string2<0)** - deci dacă string1<string2 acest <0 ipotetic îi corespunde lui -1

la fel

**if(string1==string2)** aceasta ar fi echivalentă cu comparația "ilegala" sintactic

**if(string1-string2==0)** - deci dacă string1==string2 acest ==0 ipotetic îi corespunde lui 0 analog cu >)