Siruri de caractere – prezentare generală

Un șir de caractere este o structură de date care este formată dintr-o mulțime ordonată de caractere, în care fiecare caracter se identifică prin poziția sa în cadrul mulțimii. Un șir de caractere este, de fapt, o succesiune de caractere. În limbajul C++ șirurile de caractere pot fi implementate ca vectori de caractere. După cum știți, în general, vectorii au două lungimi: o lungime fizică și o lungime logică, ceea ce se aplică și vectorilor de caractere. Ceea ce deosebește un vector de caractere de alte tipuri de vectori este posibilitatea de a marca sfârșitul logic al vectorului prin folosirea caracterului **NULL** (care are codul ASCII 0).

1. Declararea șirurilor de caractere

O constantă de tip șir de caractere se declară între doua caractere “. În memoria internă, o constantă de acest tip este reținută sub forma unui vector de caractere. Fiecare componentă a șirului (începând cu cea de indice 0) reține codul ASCII al caracterului pe care îl memorează. Caracterul nul este memorat automat. Trebuie rezervate *lungimea șirului*+1 caractere char (+1 pentru caracterul nul).

Exemple

char s[6] = {’a’, ’b’, ’c’, ’d’, ’e’};
char sir[5] = {’a’, ’b’, ’c’, ’d’, ’\0’};
char s[5] = ”abcd”; char s[ ] =
”bac2015”; char sir1[15] =
”abracadabra”; char s[10];

2. Citirea și afișarea șirurilor de caractere

Fie următoarea declarație:

char s[256]; - s-a declarat un șir de caractere cu numele s ce poate memora maximum 255 de caractere.

Citirea sirului s se poate face utilizând operatorul uzual de citire **>>** : **cin>>s;**

În acest caz se vor citi în șirul s toate caracterele până la primul caracter alb (spațiu, tab, enter). De exemplu, dacă fluxul de intrare conține caracterele **“Buna ziua”,** după citire, sirul s va fi Buna. Pentru a elimina acest dezavantaj se pot folosi funcțiile get( ) sau getline( ) (diferența între ele este că getline() extrage din fluxul de intrare caracterul delimitator, în timp ce get() nu îl extrage). Dacă totuși doriți să folosiți numai funcția get, atunci după citirea fiecărui șir trebuie să scrieți funcția cin.get() fără parametri.

Sintaxa:

cin.get(nume_sir,lungimea_sirului);
cin.getline(nume_sir,lungimea_sirului,delimitator); - unde delimitatorul este optional (implicit este caracterul ' \ n’**.**

Exemple

```
#include <iostream>
using namespace std; int
main()
{
    //declararea unor siruri
    char s[256],vocale[]="aeiou",sir[12]="bacalaureat",s1[50],s2[35];
cout<<"Introduceti primul sir:";
    //citirea primului sir
cin.get(s,255);
    //afisarea primului sir citit
cout<<s<<endl;
    cin.get();//fara aceasta instructiune nu se poate citi urmatorul sir
cout<<"Introduceti al doilea sir:";     cin.getline(s1,49);
cout<<s1<<endl;
    cout<<"Introduceti al treilea sir:";
cin.get(s2,34);     cout<<s2; }
```

Afișarea unui șir de caractere în limbajul C++ se face cu ajutorul operatorului de scriere <<, cum se observă și în exemplul de mai sus, caracterele fiind afișate până la întâlnirea marcajului de sfârșit de șir (NULL). **cout<<s;** Șirurile de caractere **pot fi prelucrate la nivel de caracter** (pot fi parcurse caracter cu caracter, ca un vector de caractere), sau **pot fi prelucrate la nivel de structură** (cu ajutorul funcțiilor existente în bibliotecile limbajului, **cstring** sau **string.h**).

3. Variabilele de tip pointer și șirurile de caractere

Șirurile de caractere pot fi manipulate prin intermediul unei variabile de tip pointer către tipul **char**. Pointerul este o variabilă de memorie în care se memorează o adresă de memorie. **Declararea unui pointer char *sir=”albacazapada”;**

Exemplu:

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
  char s[60]="informatica",*sir,*p;
  sir=s;
  cout<<sir<<endl;//se afiseaza informatica
  p=s+2;
  cout<<p<<endl;//se afiseaza formatica
  p=strchr(sir,'m');
cout<<p;//se afiseaza matica
}
```

4. Funcții pentru prelucrarea șirurilor de caractere

Funcția **strlen** - **strlen**(nume_sir); – returnează lungimea efectivă a unui șir (fără a număra terminatorul de șir).

Exemplu: char a[50]=”ora de informatica”;

cout<<strlen(a);//se afișează 18 Funcția

strcpy

strcpy(sir_destinatie,sir_sursa); **–** copiază șirul sir_ sursa în sir_destinatie (se simulează atribuirea a=b).

Exemplu:

char a[50]=”primul sir”,b[40]=”al doilea sir”;
a=b; //eroare strcpy(a,b); a = ”al doilea sir”;
b=”al doilea sir”;

Funcția **strcat** strcat(dest,sursa); – adaugă șirului dest șirul sursa. Șirul sursa rămâne nemodificat. Operația se numește *concatenare* și nu este comutativă.

Exemplu: char *a=”vine ”,*b=”vacanta?”; strcat(a,b); a = ”vine vacanta?”; Funcția

strncat

strncat(dest,sursa,nr); – adaugă la dest primele nr caractere din șirul sursa. Șirul sursa rămâne nemodificat.

Exemplu:

char *a=”vine”,*b=”varanul?”; strncat(a,b,4); a = ”vine vara”;

Funcția **strchr**

strchr(sir,c); – are rolul de a căuta caracterul c în șirul sir. Căutarea se face de la stânga la dreapta, iar funcția întoarce adresa subșirului care începe cu prima apariție a caracterului c. Dacă nu este găsit

caracterul, funcția returnează 0. Diferența dintre adresa șirului inițial și cea a subșirului returnat reprezintă chiar poziția caracterului căutat în șirul dat. *Exemplu*:

char *a="acesta este un sir",b='t',c='x',d; cout<<strchr(a,b);//se tipărește "ta este un sir"; cout<<strchr(a,c);//nu se tipărește nimic

d= strchr(a,b); cout<<"Caracterul apare prima data la poziția "<<d-a;

Ex: Să se afișeze toate pozițiile unui caracter într-un șir.

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
{
   char a[100],*p,c;
cin.get(a,100);
cin>>c;
p=strchr(a,c);
while (p)
     {
        cout<<"Pozitia "<<p-a<<endl;
p++;
        p=strchr(p,c);
     } }
```

Funcția **strrchr** strrchr(sir,c); – are același rol cu strchr, cu deosebirea că returnează adresa ultimei apariții a caracterului (căutarea se face de la dreapta spre stânga)

Funcția **strcmp** strcmp(sir1,sir2); – are rolul de a compara două șiruri de caractere. Valoarea returnată este -1 (dacă sir1<sir2), 0 (dacă sir1=sir2) și 1 (dacă sir1>sir2).
Obs: Funcția **strcmp** returnează diferența dintre codurile ASCII ale primelor caractere care nu coincid.

Funcția **stricmp** stricmp(sir1,sir2); – are același rol cu strcmp, cu deosebirea că *nu face distincție* între literele mari și cele mici ale alfabetului.

Funcția **strstr** strstr(sir1,sir2); – are rolul de a identifica dacă șirul sir2 este subșir al șirului sir1. Dacă este, funcția returnează adresa de început a subșirului sir2 în șirul sir1, altfel returnează adresa 0. În cazul în care sir2 apare de mai multe ori în sir1, se returnează adresa de început a primei apariții. Căutarea se face de la stânga la dreapta.

Funcția **strtok** strtok(sir1,sir2); – are rolul de a separa șirul $_{sir1}$ în mai multe șiruri (cuvinte) separate între ele prin unul sau mai multe caractere cu rol de separator. Șirul sir2 este alcătuit din unul sau mai multe caractere cu rol de separator. Funcția **strtok** acționează în felul următor:

- Primul apel trebuie să fie de forma strtok(sir1,sir2); funcția întoarce adresa primului caracter al primei entități. După prima entitate, separatorul este înlocuit automat cu caracterul nul.
- Următoarele apeluri sunt de forma strtok(NULL,sir2); de fiecare dată, funcția întoarce adresa de început a următoarei entități, adăugând automat după ea caracterul nul.
- Când șirul nu mai conține entități, funcția returnează adresa nulă.

Exemplu: Să se separe cuvintele dintr-un text.

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
{
```

```
char text[100],*p,separator[]=",. !?";
int nr=0; cout<<"Dati sirul:";
cin.get(text,100);
p=strtok(text,separator); while (p)
   {
       cout<<p<<endl;
nr++;
       p=strtok(NULL,separator);
    }
    cout<<"Textul are "<<nr<<" cuvinte."; }
```

Funcția **strlwr** cu forma generală strlwr(sir); – are rolul de a converti toate literele mari din sir în litere mici. Restul caracterelor rămân neschimbate.

Funcția **strupr** cu forma generală strupr(sir); – are rolul de a converti toate literele mici din sir în litere mari. Restul caracterelor rămân neschimbate.

Funcția **tolower(ch)** - transformă caracterul ch din literă mare în literă mică, altfel îl lasă neschimbat.

Funcția **toupper(ch**) - transformă caracterul ch din literă mică în literă mare, altfel îl lasă neschimbat.

Funcția **strrev(sir) –** inversează conținutul unui șir de caractere.

Funcția **isalnum(ch);** - testează dacă un caracter este literă sau cifră.

Funcția **isalpha(ch); -** testează dacă un caracter este literă.

Funcția **isdigit(ch); -** testează dacă un caracter este cifră.

Funcția **islower(ch); -** testează dacă un caracter este literă mică.

Funcția **isupper(ch); -** testează dacă un caracter este literă mare.

Exemplul nr.1:

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
{
char sir[40]="Azi Am Luat 10 La Informatica?"; int
i;
for(i=0;i<strlen(sir);i++) {
    if(isalnum(sir[i])==0)
        cout<<sir[i];//se afiseaza spatiile si semnul intrebarii
}
}
```

Exemplul nr.2:

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
{
char sir[40]="Azi Am Luat 10 La Informatica?"; int
i;
for(i=0;i<strlen(sir);i++) {
    if(isalpha(sir[i])==0)
        cout<<sir[i];//se afiseaza spatiile,cifrele si semnul intrebarii
}
}
```

Exemplul nr.3:

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
{
char sir[40]="Azi Am Luat 10 La Informatica?"; int
i;
for(i=0;i<strlen(sir);i++)
{
    if(isdigit(sir[i]))
        cout<<sir[i];//se afiseaza cifrele
    //sau
    //if(sir[i]>='0' && sir[i]<='9')
        //cout<<sir[i];

}
}
```

Exemplul nr.4:

```
#include <iostream>
#include <cstring> using
namespace std; int
main()
```

```
{
char sir[40]="Azi Am Luat 10 La Informatica?"; int
i;
for(i=0;i<strlen(sir);i++)
{
    if(islower(sir[i]))
        cout<<sir[i];//se afiseaza numai literele mici
    //sau
    //if(sir[i]>='a' && sir[i]<='z')
        //cout<<sir[i];
} cout<<endl;
for(i=0;i<strlen(sir);i++)
{
    if(isupper(sir[i]))
        cout<<sir[i];//se afiseaza numai literele mari
    //sau
    //if(sir[i]>='A' && sir[i]<='Z')
        //cout<<sir[i];
}
}
```

Probleme – Șiruri de caractere

1. Ce se va afișa în urma executării secvenței de mai jos: `char s[10]="primavara"; int i; for(i=1;i<=3;i++) strcpy(s+1,s+2); cout<<s;`

2. Ce se va afișa în urma executării secvenței de mai jos.

```
char x[]="Mama",y[]="Macara";

if(strcmp(x,y)>0)
cout<<x;      else

if(strcmp(x,y)==0)
cout<<"Incorect";
else          cout<<y;
```

3. Ce se va afișa la finalul executării următoarei secvențe de instrucțiuni?

```
    char x[]="albacazapada",*p;
x[0]=x[0]-32;
p=strchr(x,'a');
    cout<<x[0]<<p[0]<<x[strlen(x)-1];
```

4. Fie următorul program:

```
    #include <iostream>
    #include <cstring>
    using namespace std;
    int main()
    {
       char s[10],x;    int i;
    strcpy(s,"clasa");
    for(i=0;i<strlen(s)-1;i++)
        {
            if(s[i]>s[i+1])
            {
    x=s[i];
```

```
s[i]=s[i+1];
s[i+1]=x;
          }
cout<<s<<" ";
     } }
```

Ce se va afișa pe ecran în urma execuției acestui program?

5. Se citeşte un şir de caractere. Să se contorizeze vocalele, spaţiile şi consoanele.

6. Se citesc două cuvinte. Să se afişeze mesajul DA sau NU, dacă cele două cuvinte rimează.

7. Se citeşte un şir format numai din litere şi spaţii. Să se afişeze şirul, după ce vocalele au fost şterse.

8. Se citeşte o frază. Se cere să se afişeze numărul cuvintelor, cel mai lung şi cel mai scurt cuvânt.

9. Se citeşte un text cu cel mult 255 de caractere. Scrieţi un program care să modifice textul citit astfel:

a) Transformaţi toate literele în majuscule;
b) Transformaţi toate literele în minuscule;
c) Transformaţi toate literele mari în litere mici şi literele mici în litere mari;
d) Transformaţi fiecare cuvânt text astfel încât să fie scris cu prima literă mare;
e) Transformaţi fiecare cuvânt text astfel încât să fie scris cu prima şi ultima literă mare.

10. Să se verifice dacă o frază este de tip palindrom.(Au o nava noua. Ele fac cafele.)

11. Se citeşte un text de cel mult 255 caractere şi o literă c. Ştergeţi din textul citit toate apariţiile literei c. Exemplu: **“abcd” şi c=‘b’** se obține șirul **“acd”.**

12. Se citeşte un text de cel mult 255 caractere. Înlocuiţi toate vocalele din text cu ‘*’. Exemplu: **“abedi”** se obține șirul **“*b*d*”.**

13. Se citeşte un text de cel mult 255 caractere şi două litere c1 şi c2. Înlocuiţi în textul citit toate apariţiile literei c1 cu litera c2. Exemplu: **“abada” , c1=‘a’ şi c2=‘e’** se obține șirul **“ebede” .**

14. Se citeşte un text de cel mult 255 caractere şi două litere c1 şi c2. Inversaţi în textul citit toate apariţiile literei c1 cu litera c2. Exemplu: **“abada” , c1=‘a’ şi c2=‘b’** se obține șirul **“babdb”.**