Tipuri structurate de date

Sumar

Competenţe generale

- *identificarea datelor care intervin într-o problemă şi a relaţiilor dintre acestea*
- *elaborarea algoritmilor de rezolvare a problemelor*
- *aplicarea algoritmilor fundamentali în prelucrarea datelor*
- *identificarea conexiunilor dintre informatică şi societate*

Competenţe specifice

- *evidenţierea necesităţii structurării datelor*
- *prelucrarea datelor structurate*
- *alegerea structurii de date adecvată rezolvării unei probleme*
- *elaborarea unui algoritm de rezolvare a unei probleme din aria currciculară a specialităţii*
- *alegerea unui algoritm eficient de rezolvare a unei probleme*
- *identificarea aplicaţiilor informaticii în viaţa socială*
- *elaborarea şi implementarea unor algoritmi de rezolvare a unor probleme cotidiene*

Prezentare generală

Un şir de caractere:
- este un tablou unidimensional (vector) de caractere;
- reprezintă o succesiune de caractere cuprins între ghilimele şi terminat cu caracterul **`NULL`** notat **`'\0'`**;
- conţine orice tip de caracter: literă mică, literă mare, cifră, caracter special, delimitator, fiecare caracter fiind reprezentat prin codul său ASCII.

Un *şir de caractere* este o structură de date care este formată dintr-o mulţime ordonată de caractere, în care fiecare caracter se identifică prin poziţia sa în cadrul mulţimii.

Exemplu

Şirul de caractere "**Limbajul C++**" se poate reprezenta astfel:

L	i	m	b	a	j	u	l		C	+	+	\0
0	1	2	3	4	5	6	7	8	9	10	11	12

Fiecare caracter din setul de caractere al limbajului C++ se caracterizează printr-un cod unic, numit codul ASCII al caracterului, un număr întreg între **0** şi **255**.

În cadrul setului ASCII, codurile caracterelor sunt după cum urmează:
- literele mari începând cu **65 (A – 65, B – 66, ..., Z – 90)**;
- literele mici începând cu **97 (a – 97, b – 98, ..., z – 122);**
- cifrele începând cu **48 (0 – 48, 1 – 49, ..., 9 – 57).**

Observaţie: Diferenţa între codul ASCII al oricărei litere mici şi codul ASCII al literei mari pereche este aceeaşi, **32**.

Exemplu

```
char c1,c2;
int x;
c1='A';
x=c1+32;     //conversie implicită
c2=x;        //conversie implicită
cout<<c1<<endl<<c2<<endl<<x;
```

Declararea şirurilor de caractere

- vectorul de caractere trebuie declarat cu un caracter mai mult (caracterul **NULL**) decât cel mai mare şir pe care îl poate conţine;
- un şir de caractere poate fi definit ca un vector de caractere, în două moduri:

```
char nume[dimensiune];
```

sau

```
char *nume;
```

unde **nume** este identificatorul variabilei de tip şir de caractere, iar **dimensiune** reprezintă numărul maxim de caractere ce pot fi memorate în şir (inclusiv caracterul **NULL**);

```
char nume[dimensiune];
```
// se declară un vector cu elemente de tip caracter;

```
char *nume;
```
// se declară un pointer către tipul caracter;

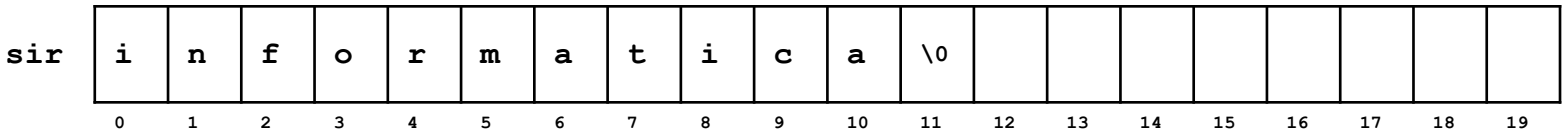
Observaţie: caracterele şirului vor ocupa poziţii consecutive în vector, începând cu poziţia 0

Exemplu 1

```
char sir[20];
```

- se declară un şir de caractere în care vor putea fi memorate maxim **19** caractere;

- dacă variabila **sir** reţine şirul "**informatica**", reprezentarea în memoria internă este următoarea:



unde: **sir[0]='i'**
sir[1]='n'
sir[2]='f'
.....
sir[10]='a'

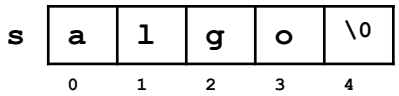
- caracterul **NULL** ('\0') este adăugat automat de către compilator;
- lungimea şirului de caractere este dată de numărul de caractere (în exemplu este de **11**);

Exemplu 2

```
char s[5];
```

- se declară un şir de caractere în care vor putea fi memorate maxim **4** caractere;

- dacă variabila **s** reţine şirul ”**algoritm**”, reprezentarea în memoria internă este următoarea:

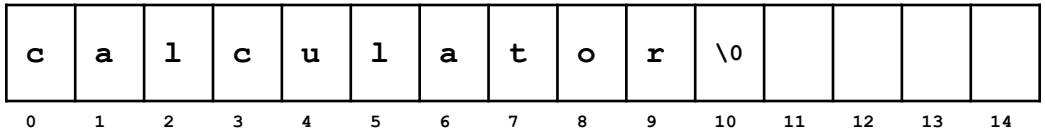


Exemplu 3

```
char a[15]="calculator";
```

- se declară un şir de caractere în care vor putea fi memorate maxim **14** caractere;

- şirul de caractere **a** a fost iniţializat la declararea lui, iar reprezentarea în memoria internă este următoarea:



- dacă se iniţializează şirul de caractere, nu mai este nevoie să se precizeze lungimea maximă a şirului, aceasta fiind calculată de către compilator:

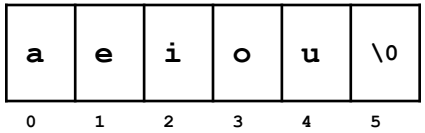
```
char a[]="calculator";
```

Exemplu 4

```
char s[]={'a','e','i','o','u'};
s[5]=NULL;
```

- se declară un şir de caractere care memorează vocalele litere mici;

- şirul de caractere **a** a fost iniţializat la declararea lui, iar reprezentarea în memoria internă este următoarea:



- utilizatorul trebuie să plaseze la sfârşitul şirului indicatorul de sfârşit de şir;

Citirea şi scrierea şirurilor de caractere

Citirea şirurilor de caractere

Citirea se face folosind instrucţiunea `cin` în două moduri:

- la nivel de caracter;
- la nivelul şirului de caractere.

a. Citirea la nivel de caracter (caracter cu caracter)

```
cout<<"numarul de caractere:";
cin>>n;
for(i=0;i<n;i++)
   cin>>sir[i];
sir[n]=NULL;
```

- dacă citirea şirului de caractere se face caracter cu caracter trebuie adăugat caracterul **NULL** la sfârşitul şirului de caractere.

b. Citirea la nivelul şirului de caractere

```
cin>>sir;
```

- în acest caz caracterul **NULL** este adăugat automat de către compilator;
- operaţia de citire de la tastatură se termină atunci când a fost introdus un caracter alb (***spaţiu*** sau ***Enter***);
- pentru a citi caracterul *spaţiu* se foloseşte funcţia ***get*** care are două forme.

Forma 1

```
cin.get(sir,nr,ch);

cin.getline(sir,nr,ch);
```

unde:

- **sir** este o variabilă de tip şir de caractere care se va citi;
- **nr** este o variabilă de tip întreg şi reprezintă numărul maxim de caractere care vor fi citite, inclusiv caracterul **NULL**;
- **ch** este o variabilă de tip caracter care reprezintă caracterul care încheie citirea şirului;

– dacă al treilea parametru lipseşte, se consideră implicit caracterul '**\n**' generat de tasta **Enter**;

– diferenţa dintre funcţiile **get** şi **getline** este aceea că functia **getline** preia din fluxul de date de intrare şi delimitatorul de sfârşit de şir, în timp ce funcţia **get** nu preia ultimul caracter citit;

Forma 2

```
cin.get();
```

- folosită după o fucţie ***`cin.get()`*** cu parametri pentru a descărca din fluxul de date ultimul caracter citit, care ar împiedica efectuarea unei a doua operaţii de citire de la tastatură.

- **Exemplu 1**

```
char s1[10];
cin>>s1;
```

//dacă introducem textul **informatica** şi tastăm **Enter** se memorează în variabila **s1** şirul **informati**;

- **Exemplu 2**

```
char s2[10];
cin>>s2;
```

//dacă introducem textul **informat** şi tastăm **Enter** se memorează în variabila **s2** şirul **informat**;

- **Exemplu 3**

```
char s3[10];
cin>>s3;
```

//dacă introducem textul **info arena** şi tastăm **Enter** se memorează în variabila **s3** şirul **info**;

Exemplu 4

```
char s4[10];
cin.get(s4,10);
```

//dacă introducem textul **informatica** şi tastăm **Enter** se memorează în variabila **s4** şirul **informati**;

Exemplu 5

```
char s5[10];
cin.get(s5,5);
```

//dacă introducem textul **informatica** şi tastăm **Enter** se memorează în variabila **s5** şirul **info**;

Exemplu 6

```
char s6[10];
cin.get(s6,10);
```

//dacă introducem textul **info arena** şi tastăm **Enter** se memorează în variabila **s6** şirul **info aren**;

Exemplu 7

```
char s7[10];
cin.get(s7,10,'$');
```

//dacă introducem textul **informat$** şi tastăm **Enter** se memorează în variabila **s7** şirul **informat**;

Exemplu 8

```
char s8[10];
cin.get(s8,10);
```

//dacă introducem textul **info$arena** şi tastăm **Enter** se memorează în variabila **s8** şirul **info$aren**;

Exemplu 9

```
char s9[10];
cin.get(s9,9,'\n');
```

//dacă introducem textul **info arena$** şi tastăm **Enter** se memorează în variabila **s9** şirul **info are**;

Exemplu 10

```
char a[10], b[10];
cin.get(a,10);
cin.get(b,10);
```

//dacă introducem textul **info** şi tastăm **Enter** se memorează în variabila **a** şirul **info** şi nu va putea fi citit al doilea şir;

Exemplu 11

```
char a[10], b[10];
cin.get(a,10);
cin.get();
cin.get(b,10);
```

//dacă introducem textul **info** şi tastăm **Enter** şi apoi introducem şirul **arena** şi tastăm **Enter** se memorează în variabila **a** şirul **info** şi în variabila **b** şirul **arena**;

Exemplu 12

```
char a[10], b[10];
cin.getline(a,10);
cin.getline(b,10);
```

//dacă introducem textul **info arena** şi tastăm **Enter**, iar apoi introducem textul **algoritmi** şi tastăm **Enter**, se memorează în variabila **a** şirul **info aren** şi în variabila **b** şirul **algoritmi**;

Scrierea şirurilor de caractere

Scrierea se face folosind instrucţiunea **`cout`** în două moduri:

- la nivel de caracter;
- la nivelul şirului de caractere.

a. Scrierea la nivel de caracter (caracter cu caracter)

```
cin>>sir;
i=0;
while(sir[i]!=NULL)
{
   cout<<sir[i];
   i++;
}
```

b. Scrierea la nivelul şirului de caractere

```
cout<<sir;
```

- **Exemplu 1**

```
char s1[10];
cin>>s1;
cout<<s1;
```

//dacă introducem textul **informat** şi tastăm **Enter** se afişează şirul **informat**;

- **Exemplu 2**

```
char s2[10];
cin>>s2;
cout<<s2+2;
```

//dacă introducem textul **informat** şi tastăm Enter se afişează şirul **format**;

- **Exemplu 3**

```
char s3[10];
cin>>s3;
cout<<s3;
```

//dacă introducem textul **info arena** şi tastăm **Enter** se afişează şirul **info**;

Prelucrarea şirurilor de caractere

Funcţii care lucrează cu şiruri de caractere

1. STRLEN
2. STRCPY
3. STRNCPY
4. STRCAT
5. STRNCAT
6. STRCMP
7. STRNCMP
8. STRICMP
9. STRSET
10. STRNSET
11. STRLWR
12. STRUPR
13. TOUPPER
14. TOLOWER
15. STRCHR
16. STRRCHR
17. STRSTR
18. STRREV

1. Funcţia `STRLEN`

Efect:

Returnează lungimea şirului de caractere.

Sintaxa:

```
strlen(sir)
```

unde **`sir`** este o variabilă de tip şir de caractere.

Exemplu

```
char s[21];
int n;
cin>>s;
n=strlen(s);
cout<<n;
```

Dacă se citeşte în variabila **s** şirul **informatica** se afişează valoarea **11**.

Exerciţiu

Care este efectul apelului:

```
cout<<strlen("programare");
```
?

2. Funcţia `STRCPY`

Efect:

Copie un şir de caractere în alt şir de caractere.

Sintaxa:

```
strcpy(sir1,sir2)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere.

Exemplu

```
char a[21],b[21];
cin>>a;
strcpy(b,a);
cout<<b;
```

Dacă se citeşte în variabila **a** şirul **informatica** se afişează şirul **informatica**.

Exerciţiu

Care este efectul apelului:

strcpy(b,a+2); ?

3. Funcţia `STRNCPY`

Efect:

Copie un număr de caractere specificat, dintr-un şir de caractere în alt şir de caractere.

Sintaxa:

```
strncpy(sir1,sir2,nr)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere, iar **`nr`** este o variabilă de tip întreg.

Observaţie:

Dacă **`strlen(sir2)<nr`**, atunci **`sir1`** se completează automat cu caracterul **`NULL;`** în caz contrar **`sir1`** nu se va termina cu caracterul **`NULL`**, acesta trebuind să fie adăugat de către utilizator.

Exemplu

```
char a[21],b[21];
cin>>a;
strncpy(b,a,4);
b[4]=NULL;
cout<<b;
```

Dacă se citeşte în variabila **a** şirul **informatica** se afişează şirul **info**.

Exerciţiu

Care este efectul apelului:

```
strncpy(b,a+7,3);
```
?

4. Funcţia `STRCAT`

Efect:

Concatenează două şiruri de caractere.

Sintaxa:

```
strcat(sir1,sir2)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere.

Exemplu

```
char a[21],b[21];
cin>>a>>b;
strcat(a,b);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **b** şirul **arena** se afişează şirul **infoarena**.

Exerciţiu

Care este efectul apelurilor:

```
strcat(a," ");
strcat(a,b);
```
?

5. Funcţia `STRNCAT`

Efect:

Concatenează la un șir de caractere un număr de caractere specificat din alt șir de caractere.

Sintaxa:

```
strncat(sir1,sir2,nr)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere, iar **`nr`** este o variabilă de tip întreg.

Exemplu

```
char a[21],b[21];
cin>>a>>b;
strncat(a,b,3);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **b** şirul **arena** se afişează şirul **infoare**.

Exerciţiu

Care este efectul apelului:

```
strncat(a,b+0,3);
```
?

6. Funcţia `STRCMP`

Efect:

Compară două şiruri de caractere.

Sintaxa:

```
strcmp(sir1,sir2)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere.

`strcmp(sir1,sir2)=`
- o valoare negativă, dacă **`sir1<sir2`**
- **`zero`**, dacă **`sir1=sir2`**
- o valoare pozitivă, dacă **`sir1>sir2`**

Exemplu

```
char a[21],b[21];
cin>>a>>b;
cout<<strcmp(a,b);
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **b** şirul **intro** se afişează valoarea **-1**.

Exerciţiu

Care este efectul apelului:

```
cout<<strcmp("intro",a);
```
?

7. Funcţia `STRNCMP`

Efect:

Compară un număr de caractere specificat din două şiruri de caractere.

Sintaxa:

```
strncmp(sir1,sir2,nr)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere, iar **`nr`** este o variabilă de tip întreg.

Exemplu

```
char a[21],b[21];
cin>>a>>b;
cout<<strncmp(a,b,3);
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **b** şirul **intro** se afişează valoarea **-14**.

Exerciţiu

Care este efectul apelului:

```
cout<<strncmp(a,"intro",2);
```
?

8. Funcţia `STRICMP`

Efect:

Compară două şiruri de caractere fără a face diferenţa între litere mici şi litere mari.

Sintaxa:

```
stricmp(sir1,sir2)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere.

Exemplu

```
char a[21],b[21];
cin>>a>>b;
cout<<stricmp(a,b);
```

Dacă se citeşte în variabila **a** şirul **arena** şi în variabila **b** şirul **ARE** se afişează valoarea **1**.

Exerciţiu

Care este efectul apelului:

```
cout<<stricmp("Are","are");
```
?

9. Funcţia `STRSET`

Efect:

Iniţializează un şir de caractere cu acelaşi caracter.

Sintaxa:

```
strset(sir,ch)
```

unde **`sir`** este o variabilă de tip şir de caractere, iar **`ch`** este o variabilă de tip caracter.

Exemplu

```
char a[21],x;
cin>>a>>x;
strset(a,x);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **x** caracterul **1** se afişează şirul **1111**.

Exerciţiu

Care este efectul apelului:

```
strset("#*#*","#");
```
?

10. Funcţia `STRNSET`

Efect:

Iniţializează într-un şir de caractere, primele **`nr`** caractere, cu caracterul **`ch`**.

Sintaxa:

```
strnset(sir,ch,nr)
```

unde **`sir`** este o variabilă de tip şir de caractere, **`ch`** este o variabilă de tip caracter, iar **`nr`** este o variabilă de tip întreg.

Exemplu

```
char a[21],x;
int n=2;
cin>>a>>x;
strnset(a,x,n);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **x** caracterul **1** se afişează şirul **11fo**.

Exerciţiu

Care este efectul apelului:

```
strnset(a,'&',1);
```
?

11. Funcţia `STRLWR`

Efect:

Transformă literele mari în litere mici.

Sintaxa:

```
strlwr(sir)
```

unde **`sir`** este o variabilă de tip şir de caractere.

Exemplu

```
char a[21];
cin>>a;
strlwr(a);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **iNFo** se afişează şirul **info**.

Exerciţiu

Care este efectul apelului:

```
strlwr("Arena");
```
?

12. Funcţia `STRUPR`

Efect:

Transformă literele mici în litere mari.

Sintaxa:

```
strupr(sir)
```

unde **`sir`** este o variabilă de tip şir de caractere.

Exemplu

```
char a[21];
cin>>a;
strupr(a);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** se afişează şirul **INFO**.

Exerciţiu

Care este efectul apelului:

```
strupr("ARENA");
```
?

13. Funcţia `TOLOWER`

Efect:

Transformă o literă mare în literă mică.

Sintaxa:

```
tolower(ch)
```

unde **`ch`** este o variabilă de tip caracter.

Exemplu

```
char c;
cin>>c;
c=tolower(c);
cout<<c;
```

Dacă se citeşte în variabila **c** caracterul **N** se afişează caracterul **n**.

Exerciţiu

Care este efectul apelului:

```
c=tolower('A');
cout<<c;
```
?

14. Funcţia `TOUPPER`

Efect:

Transformă o literă mică în literă mare.

Sintaxa:

```
toupper(ch)
```

unde **`ch`** este o variabilă de tip caracter.

Exemplu

```
char c;
cin>>c;
c=toupper(c);
cout<<c;
```

Dacă se citeşte în variabila **c** caracterul **b** se afişează caracterul **B**.

Exerciţiu

Care este efectul apelului:

```
c=toupper('h');
cout<<c;
```
?

15. Funcţia `STRCHR`

Efect:

Caută prima apariţie a ununi caracter într-un şir de caractere.

Sintaxa:

```
strchr(sir,ch)
```

unde **`sir`** este o variabilă de tip sir de caractere, iar **`ch`** este o variabilă de tip caracter.

Exemplu

```
char a[21],c;
cin>>a>>c;
cout<<strchr(a,c);
```

Dacă se citeşte în variabila **a** şirul **info** şi în variabila **c** caracterul **f** se afişează şirul **fo**.

Exerciţiu

Care este efectul apelului:

```
cout<<strchr("arena",'a');
```
?

16. Funcţia `STRRCHR`

Efect:

Caută ultima apariţie a ununi caracter într-un şir de caractere.

Sintaxa:

```
strrchr(sir,ch)
```

unde **`sir`** este o variabilă de tip sir de caractere, iar **`ch`** este o variabilă de tip caracter.

Exemplu

```
char a[21],c;
cin>>a>>c;
cout<<strrchr(a,c);
```

Dacă se citeşte în variabila **a** şirul **informatica** şi în variabila **c** caracterul **i** se afişează şirul **ica**.

Exerciţiu

Care este efectul apelului:

```
cout<<strrchr("info",'f');
```
?

17. Funcţia `STRSTR`

Efect:

Caută un subşir de caractere într-un şir de caractere.

Sintaxa:

```
strstr(sir1,sir2)
```

unde **`sir1`** şi **`sir2`** sunt două variabile de tip şir de caractere.

Exemplu

```
char a[21],b[21];
cin>>a>>b;
cout<<strstr(a,b);
```

Dacă se citeşte în variabila **a** şirul **informatica** şi în variabila **b** şirul **fo** se afişează şirul **formatica**.

Exerciţiu

Care este efectul apelului:

```
cout<<strstr("arena","a");
```
?

18. Funcţia `STRREV`

Efect:

Inversează ordinea caracterelor într-un şir de caractere.

Sintaxa:

```
strrev(sir)
```

unde **`sir`** este o variabillă de tip şir de caractere.

Exemplu

```
char a[21];
cin>>a;
strrev(a);
cout<<a;
```

Dacă se citeşte în variabila **a** şirul **info** se afişează şirul **ofni**.

Exerciţiu

Care este efectul apelului:

```
strrev("123321");
```
?

Fişe de lucru

- Întrebări şiruri de caractere
- Aplicaţii şiruri de caractere

7. Bibliografie şi webografie

1. Miloşescu M., *Informatică. Manual pentru clasa a X-a*, Editura Didactică şi Pedagogică, Bucureşti, 2005
2. Munteanu F., *Programarea calculatoarelor.* M*anual pentru licee de informatică clasele X-XII*, Editura Didactică şi Pedagogică, Bucureşti, 1994
3. Logofătu D.*, Bazele programării în C++*, Editura Polirom, Iaşi, 2006
4. Popescu C., *Culegere de probleme de informatică*, Editura Donaris-Info, Sibiu, 2002
5. Ministerul Educaţiei, Cercetării şi Tineretului, Centrul Naţional pentru Curriculum şi Evaluare în Învăţământul Preuniversitar, *Proba scrisă la informatică. Examenul de bacalaureat – Variante (1-100)* , Bucureşti 2008
6. http://www.tutorialeprogramare.ro/Tutorial%20C/Siruri%20de%20caractere.html
7. http://ro.wikipedia.org/wiki/Strlen
8. http://en.wikipedia.org/wiki/C_string_handling