

Programarea calculatoarelor

FMI

Secția Calculatoare și tehnologia informației, anul I

Cursul 9 / 27.11.2023

Programa cursului

□ Introducere

- Algoritmi
- Limbaje de programare.

□ Fundamentele limbajului C

- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: pointeri, tablouri, șiruri de caractere, structuri, uniuni, câmpuri de biți, enumerări
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

□ Fișiere text

- Funcții specifice de manipulare.

□ Funcții (1)

- Declarare și definire. Apel. Metode de transmitere a paramerilor. Pointeri la funcții.

□ Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetica pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

□ Șiruri de caractere

- **Funcții specifice de manipulare**

□ Fișiere binare

- Funcții specifice de manipulare

□ Structuri de date complexe și autoreferite

- Definire și utilizare

□ Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.



Cuprinsul cursului de azi

- 1. Șiruri de caractere**
2. Funcții specifice de manipulare a șirurilor de caractere
3. Manipularea blocurilor de memorie

ȘIRURI DE CARACTERE

- ❑ **un șir de caractere (string)** este un tablou unidimensional cu elemente de tip char terminat cu caracterul '\0' (NUL)
- ❑ o zonă de memorie ocupată cu caractere (un caracter ocupă un octet) terminată cu un octet de valoare 0 (caracterul '\0' are codul ASCII egal cu 0).
- ❑ o variabilă care reprezintă un șir de caractere este un pointer la primul octet. Se poate reprezenta ca:
 - ❑ tablou de caractere(pointer constant):
 - ❑ `char sir1[10];` //se alocă 10 octeti
 - ❑ `char sir2[10] = "exemplu";` //se alocă 10 octeti
 - ❑ `char sir3[] = "exemplu";` //se alocă 8 octeti
 - ❑ pointer la caractere:
 - ❑ `char *sir4;` //se alocă memorie numai pentru pointer
 - ❑ `char *sir5 = "exemplu";` //se alocă 8 octeti (se adaugă '\0' la final

CITIREA ȘI AFIȘAREA ȘIRURILOR DE CARACTERE

❑ citire:

- ❑ funcția scanf cu modelatorul de format %s:
 - ❑ atenție: dacă inputul este un șir de caractere cu spațiu citește până la spațiu
- ❑ funcția fgets (în loc de gets)
 - ❑ `char *fgets(char *s, int size, FILE *stream)`
 - ❑ `fgets(buffer, sizeof(buffer), stdin);`
 - ❑ citește și spațiile

❑ afișare:

- ❑ funcția printf cu modelatorul de format %s:
- ❑ funcția puts (trece pe linia următoare)

CITIREA ȘI AFIȘAREA ȘIRURILOR DE CARACTERE

□ exemplu

main.c

```
3
4 int main()
5 {
6     char sir1[] = {'r','a','t','o','n','\0'};
7     char sir2[] = "raton";
8     printf("%s %s \n", sir1, sir2);
9
10    char *sir3=sir1;
11    sir3[0] = 'b';
12    printf("%s %s %s\n", sir1, sir2, sir3);
13
14    char sir4[100] = "raton";
15    printf("%s \n",sir4);
16    int i;
17    for (i=0;i<10;i++)
18        printf("Caracterul %c = codul ASCII %d\n",sir4[i],sir4[i]);
19    sir4[5] = 'i';
20    printf("%s \n",sir4);
21
22    char sir5[10] = "raton";
23    sir5[4]=0;
24    printf("%s \n",sir5);
25    sir5[3]='\0';
26    printf("%s \n",sir5);
```

CITIREA ȘI AFIȘAREA ȘIRURILOR DE CARACTERE

□ exemplu

main.c

```
3
4 int main()
5 {
6     char sir1[] = {'r','a','t','o','n','\0'};
7     char sir2[] = "raton";
8     printf("%s %s \n", sir1, sir2);
9
10    char *sir3=sir1;
11    sir3[0] = 'b';
12    printf("%s %s %s\n", sir1, sir2, sir3);
13
14    char sir4[100] = "raton";
15    printf("%s \n",sir4);
16    int i;
17    for (i=0;i<10;i++)
18        printf("Caracterul %c = codul ASCII %d\n",sir4[i],sir4[i]);
19    sir4[5] = 'i';
20    printf("%s \n",sir4);
21
22    char sir5[10] = "raton";
23    sir5[4]=0;
24    printf("%s \n",sir5);
25    sir5[3]='\0';
26    printf("%s \n",sir5);
```

```
raton raton
baton raton baton
raton
Caracterul r = codul ASCII 114
Caracterul a = codul ASCII 97
Caracterul t = codul ASCII 116
Caracterul o = codul ASCII 111
Caracterul n = codul ASCII 110
Caracterul  = codul ASCII 0
Caracterul  = codul ASCII 0
Caracterul  = codul ASCII 0
Caracterul  = codul ASCII 0
Caracterul  = codul ASCII 0
ratoni
rato
rat
```

Cuprinsul cursului de azi

1. Șiruri de caractere
2. **Funcții specifice de manipulare a șirurilor de caractere**
3. Manipularea blocurilor de memorie

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ funcții de procesare a șirurilor de caractere specifice incluse în **fișierul string.h**:
 - ❑ **lungimea unui șir** – funcția **strlen**
 - ❑ antet: **int strlen(const char *sir)**

exempluStrlen.c

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char sir[100] = "test";
7      printf("%d \n", sizeof(sir));
8      printf("%d \n", strlen(sir));
9      return 0;
10 }
```

100
4

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- lungimea unui șir – funcția strlen
 - antet: `int strlen(const char *sir)`

exempluStrlen.c

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char sir[100] = "test";
7      printf("%d \n", sizeof(sir));
8      printf("%d \n", strlen(sir));
9      int i;
10     for(i=0; i<strlen(sir); i++)
11         printf("%s \n", sir+i);
12     return 0;
13 }
```

```
100
4
test
est
st
t
```

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

❑ copierea unui șir

- ❑ nu se poate copia conținutul unui șir în alt șir folosind operația de atribuire (se copiază pointerul și nu conținutul)

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char sir6[10] = "raton";
7      char *sir7="baton";
8      printf("Adresa lui sir6 este %d \n",sir6);
9      printf("Adresa lui sir7 este %d \n",sir7);
10
11     sir7=sir6;
12     puts(sir7);
13
14     printf("Adresa lui sir6 este %d \n",sir6);
15     printf("Adresa lui sir7 este %d \n",sir7);
16
17     return 0;
18 }
```

```
Adresa lui sir6 este 1606416720
Adresa lui sir7 este 3812
raton
Adresa lui sir6 este 1606416720
Adresa lui sir7 este 1606416720
```

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

❑ copierea unui șir

- ❑ nu se poate copia conținutul unui șir în alt șir folosind operația de atribuire (se copiază pointerul și nu conținutul).

main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char sir6[10] = "raton";
7      char sir7[20] = "baton";
8      printf("Adresa lui sir6 este %d \n", sir6);
9      printf("Adresa lui sir7 este %d \n", sir7);
10
11     sir7=sir6;
12     puts(sir7);
13
14     printf("Adresa lui sir6 este %d \n", sir6);
15     printf("Adresa lui sir7 este %d \n", sir7);
16
17     return 0;
18 }
```

11 error: incompatible types in assignment

sir7 este numele unui tablou (pointer constant). Instrucțiunea sir7=sir6 da eroare la compilare.

FUNCTȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ copierea unui șir – funcțiile **strcpy** și **strncpy**
 - ❑ antet: **char* strcpy(char *d, char* s);**
 - ❑ copiază șirul sursă **s** în șirul destinație **d**;
 - ❑ returnează adresa șirului destinație
 - ❑ șirul rezultat are un **'\0'** la final
 - ❑ antet: **char* strncpy(char *destinație, char* sursa, int n);**
 - ❑ copiază primele **n** caractere din șirul sursă **s** în șirul destinație **d**;
 - ❑ returnează adresa șirului destinație
 - ❑ șirul rezultatul NU are un **'\0'** la final

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ copierea unui șir – folosim funcțiile **strcpy** și **strncpy**

exempluStrncpy.c

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[10] = "exemplu";
7      char t[10] = "test";
8      strncpy(s,t,3);
9      printf("%s \n",s);
10
11     s[4] = 0;
12     printf("%s \n",s);
13
14     char p[100] = "nimic";
15     strcpy(p,s);
16     p[3] = '\\0';
17     printf("%s \n",p);
18
19     return 0;
20 }
```

```
tesmplu
tesm
tes
```

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ copierea unui șir – folosim funcțiile `strcpy` și `strncpy`
 - ❑ antet: `char* strcpy(char *d, char* s);`
 - ❑ presupune că șirurile destinație și sursa nu se suprapun
 - ❑ dacă cele două șiruri se suprapun funcția prezintă *undefined behaviour* (comportament nedefinit)

```
exempluStrncpy1.c x
1  #include<stdio.h>
2
3  int main()
4  {
5      char s[10] = "exemplu";
6      char t[10] = "test";
7      strcpy(t,t+1);
8      printf("%s \n",t);
9
10     strcpy(s+1,s);
11     printf("%s \n",s);
12
13     return 0;
14 }
```

est
eeeeeeeeeeeeeeeeeeee

Se folosesc functiile
`memcpy`, `memmove`

FUNCȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

Compararea șirurilor – funcțiile **strcmp** și **strncmp**

- ❑ antet: **int strcmp(const char *s1, const char* s2);**
 - ❑ compară lexicografic șirurile **s1** și **s2**;
 - ❑ returnează: <0 dacă $s1 <_L s2$
0 dacă $s1 =_L s2$
>0 dacă $s1 >_L s2$
- ❑ antet: **int strncmp(const char *s1, const char* s2, int n);**
 - ❑ compară lexicografic șirurile **s1** și **s2** trunchiate la lungimea **n**
- ❑ ambele funcții sunt case sensitive
 - ❑ strcmp("POPA","Popa") returnează un număr < 0 întrucât 'O' < 'o' (codurile ASCII 79 respectiv 111)
- ❑ unele implementări au funcția **stricmp** – case insensitive

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- compararea șirurilor – funcțiile **strcmp** și **strncmp**

```
main.c ×
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main()
6  {
7      char s1[20] = "Nor", *s2 = "Noiembrie", s3[10], s4[20];
8
9      int c = strcmp(s1,s2);
10     printf("c=%d\n",c);
11     c>0?printf("%s > %s",s1,s2):((c==0)?printf("%s = %s",s1,s2):printf("%s < %s",s1,s2));
12     printf("\n");
13     c = strncmp(s1,s2,2);
14     printf("c=%d\n\n",c);
15
16     char *s23=strcpy(s3,s2);
17     printf("Sirul s3 este = %s\n",s3);
18     printf("Sirul s23 este = %s\n",s23);
19     printf("Sirul s23 pointeaza catre adresa %d \n",s23);
20     printf("Adresa lui s3 este = %d \n",s3);
21
22     strncpy(s4,s2,4);
23     printf("Primele 4 litere in sirul copiat s4 = %s \n",s4);
24     return 0;
25 }
```

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

❑ compararea șirurilor – funcțiile **strcmp** și **strncmp**

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main()
6  {
7      char s1[20] = "Nor", *s2 = "Noiembrie", s3[10], s4[20];
8
9      int c = strcmp(s1,s2);
10     printf("c=%d\n",c);
11     c>0?printf("%s > %s",s1,s2):((c==0)?printf("%s = %s",s1,s2):printf("%s < %s",s1,s2));
12     printf("\n");
13     c = strncmp(s1,s2,2);
14     printf("c=%d\n\n",c);
15
16     char *s23=strcpy(s3,s2);
17     printf("Sirul s3 este = %s\n",s3);
18     printf("Sirul s23 este = %s\n",s23);
19     printf("Sirul s23 pointeaza catre adresa");
20     printf("Adresa lui s3 este = %d \n",s3);
21
22     strncpy(s4,s2,4);
23     printf("Primele 4 litere in sirul copiat s4 = %s \n",s4);
24     return 0;
25 }
```

c=9
Nor > Noiembrie
c=0

Sirul s3 este = Noiembrie
Sirul s23 este = Noiembrie
Sirul s23 pointeaza catre adresa 1606416720
Adresa lui s3 este = 1606416720
Primele 4 litere in sirul copiat s4 = Noie

FUNCȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ concatenarea șirurilor – funcțiile **strcat** și **strncat**
 - ❑ antet: **char* strcat(char *d, const char* s);**
 - ❑ concatenează șirul sursă **s** la șirul destinație **d**.
 - ❑ returnează adresa șirului destinație
 - ❑ șirul rezultat are un `'\0'` la final
 - ❑ condiție: șirurile destinație și sursă nu se suprapun, alfel funcția prezintă *undefined behaviour*; (strcat(s,s) =?)
 - ❑ antet: **char* strncat(char *d, const char* s, int n);**
 - ❑ concatenează primele **n** caractere din șirul sursă **s** la șirul destinație **d**
 - ❑ returnează adresa șirului destinație **d**
 - ❑ șirul rezultat NU are un `'\0'` la final

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- concatenarea șirurilor – funcțiile **strcat** și **strncat**

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[100] = "test";
7      char t[10] = "joi";
8
9      char* p = strncat(s,t,2);
10     printf("%s \n",s);
11     printf("%s \n", p);
12
13     strcat(s,t);
14     printf("%s \n",s);
15
16     return 0;
17 }
```

```
testjo
testjo
testjojoi
```

FUNCTȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ căutarea unui caracter într-un șir – funcțiile **strchr** și **strrchr**
 - ❑ antet: **char* strchr(const char *s, char c);**
 - ❑ caută caracterul **c** în șirul **s** și întoarce un pointer la prima sa apariție
 - ❑ căutare *de la stânga la dreapta*
 - ❑ dacă nu apare caracterul **c** în șirul **s** returnează NULL
 - ❑ antet: **char* strrchr(const char *s, char c);**
 - ❑ caută caracterul **c** în șirul **s** și întoarce un pointer la prima sa apariție
 - ❑ căutare *de la dreapta la stânga*
 - ❑ dacă nu apare caracterul **c** în șirul **s** returnează NULL

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ căutarea unui caracter într-un șir – `strchr` și `strrchr`

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[] = "exemplu";
7      char litere[] = {'e','f','g'};
8      char *p;
9      int i;
10     for(i=0;i<3;i++)
11         if(p=strchr(s,litere[i]))
12             {
13                 printf("%s \n",strchr(s,litere[i]));
14                 printf("%s \n",strrchr(s,litere[i]));
15             }
16         else
17             printf("litera %c nu se gaseste in sirul %s \n",litere[i],s);
18
19
20     return 0;
21 }
```

exemplu

emplu

litera f nu se gaseste in sirul exemplu

litera g nu se gaseste in sirul exemplu

FUNCȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ căutarea unui șir în alt șir – funcția **strstr**
 - ❑ antet: **char* strstr(const char *s, const char *t);**
 - ❑ caută șirul **t** în șirul **s** și întoarce un pointer la prima sa apariție
 - ❑ căutare de la stânga la dreapta
 - ❑ dacă nu apare șirul **t** în șirul **s** returnează **NULL**
- ❑ exemplu: să se numere de câte ori apare un șir **t** într-un șir **s**.

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ **exemplu**: să se numere de câte ori apare un șir t într-un șir s.

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[1000], t[1000];
7      printf("Sirul s este : ");scanf("%s",s);
8      printf("Sirul t este : ");scanf("%s",t);
9
10     int nrAparitii = 0;
11     char *p;
12
13     p = strstr(s,t);
14
15     while(p)
16     {
17         nrAparitii++;
18         p = strstr(p+1,t);
19     }
20
21     printf("nrAparitii = %d \n",nrAparitii);
22
23     return 0;
24 }
25
```

Sirul s este : abracadabra
Sirul t este : ab
nrAparitii = 2

Sirul s este : aaaaaa
Sirul t este : aa
nrAparitii = 4

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ **exemplu**: să se numere de câte ori apare un șir t într-un șir s.
Număr aparițiile disjuncte.

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[1000], t[1000];
7      printf("Sirul s este : ");scanf("%s",s);
8      printf("Sirul t este : ");scanf("%s",t);
9
10     int nrAparitii = 0;
11     char *p;
12
13     p = strstr(s,t);
14
15     while(p)
16     {
17         nrAparitii++;
18         p = strstr(p+strlen(t),t);
19     }
20
21     printf("nrAparitii = %d \n",nrAparitii);
22
23     return 0;
24 }
```

```
Sirul s este : aaaaaa
Sirul t este : aa
nrAparitii = 2
```

FUNCȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ împărțirea unui șir în subșiruri – funcția **strtok**
 - ❑ antet: **char* strtok(char *s, const char *sep);**
 - ❑ împarte șirul **s** în subșiruri conform separatorilor din șirul **sep**
 - ❑ **s = "Ana ; are . mere!!"**, **sep = " ,.!?"** -> **Ana are mere**
 - ❑ string-ul inițial se trimite doar la primul apel al funcției, obținându-se primul subșir
 - ❑ la următoarele apeluri, pentru obținerea celorlate subșiruri se trimite ca prim argument **NULL**
- ❑ exemplu: să se numere cuvintele dintr-o frază

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ împărțirea unui șir în subșiruri – funcția **strtok**
- ❑ exemplu: să se numere cuvintele dintr-o frază

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[1000];
7      printf("Sirul s este : ");fgets(s,1000,stdin);
8
9      int nrCuvinte = 0;
10     char *p;
11     char separatori[] = {'(', ')', ' ', '?', '.', ';', ':', ',', '\n'};
12
13     p = strtok(s,separatori);
14
15     while(p)
16     {
17         printf("%s \n",p);
18         nrCuvinte++;
19         p = strtok(NULL,separatori);
20     }
21
22     printf("nrCuvinte = %d \n",nrCuvinte);
23
24     return 0;
25 }
```

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ împărțirea unui șir în subșiruri – funcția **strtok**
- ❑ exemplu: să se numere cuvintele dintr-o frază

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[1000];
7      printf("Sirul s este : ");
8
9      int nrCuvinte = 0;
10     char *p;
11     char separatori[] = {' '};
12
13     p = strtok(s,separatori);
14
15     while(p)
16     {
17         printf("%s \n",p);
18         nrCuvinte++;
19         p = strtok(NULL,separatori);
20     }
21
22     printf("nrCuvinte = %d\n",nrCuvinte);
23
24     return 0;
25 }
```

Sirul s este : Ana are mere. Bogdan n-are. Cativa studenti de la seria 13 (oare cati?) au deja la restanta la algebra!!!

Ana
are
mere
Bogdan
n-are
Cativa
studenti
de
la
seria
13
oare
cati
au
deja
la
restanta
la
algebra!!!

nrCuvinte = 19

FUNCTII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ conversia de la un șir la un număr și invers – funcțiile **sscanf** și **sprintf**
- ❑ conversia de la șir la un număr poate fi făcută cu ajutorul funcției **sscanf** și descriptori de format potriviți
- ❑ exemplu:

```
char *string="-45.8614";  
double numar;  
sscanf(string, "%lf", &numar);  
printf("%f", numar);
```
- ❑ conversia de la un număr la șir poate fi făcută cu ajutorul funcției **sprintf** și descriptori de format potriviți
- ❑ exemplu:

```
char string[12];  
int numar=897645671;  
sprintf(string, "%d", numar);  
printf("%s", string);
```

FUNCȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ funcții de clasificare a caracterelor (*nu a șirurilor de caractere*)
- ❑ sunt în fișierul **ctype.h**

Prototip	Descriere
int isdigit(int c)	Returnează true dacă c este cifră și false altfel
int isalpha(int c)	Returnează true dacă c este literă și false altfel
int islower(int c)	Returnează true dacă c este literă mică și false altfel
int isupper(int c)	Returnează true dacă c este literă mare și false altfel
int tolower(int c)	Dacă c este literă mare, tolower returnează c ca și literă mică. Altfel, tolower returnează argumentul nemodificat
int toupper(int c)	Dacă c este literă mică, toupper returnează c ca și literă mare. Altfel, toupper returnează argumentul nemodificat
int isspace(int c)	Returnează true dacă c este un caracter <i>white-space</i> — <i>newline</i> ('\n'), <i>space</i> (' '), <i>form feed</i> ('\f'), <i>carriage return</i> ('\r'), <i>horizontal tab</i> ('\t'), <i>vertical tab</i> ('\v') — și false altfel

FUNCTȚII PREDEFINITE PENTRU MANIPULAREA ȘIRURILOR DE CARACTERE

- ❑ funcții de clasificare a caracterelor (nu a șirurilor de caractere)
- ❑ sunt în fișierul **ctype.h**

```
char *t="9 Portocale\n3 Pere";  
printf("%s\n",t);  
printf("%d\n",isdigit(t[0]));  
printf("%d\n",isalpha(t[1]));  
printf("%d\n",islower(t[2]));  
printf("%d\n",isupper(t[2]));  
printf("%d\n",isspace(t[11]));  
printf("%d\n",isspace(t[1]));  
printf("%c\n",tolower(t[2]));  
printf("%c\n",toupper(t[4]));  
puts(t);
```

Rezultate afișate:

9 Portocale

3 Pere

1

0

0

1

8

8

P

R

9 Portocale

3 Pere

Cuprinsul cursului de azi

1. Șiruri de caractere
2. Funcții specifice de manipulare a șirurilor de caractere
3. Manipularea blocurilor de memorie

FUNCTȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea elementelor unui tablou a într-un alt tablou b:
 - ❑ **nu** se poate face prin atribuire ($b=a$), întrucât a și b sunt pointeri constanți;
 - ❑ copierea se face element cu element folosind instrucțiuni repetitive (for, while);
- ❑ pentru stringuri (tablouri de caractere) avem funcțiile predefinite **strcpy** și **strncpy**:
 - ❑ **char* strcpy(char *d, char* s);**
 - ❑ copiază șirul sursă s în șirul destinație d ;
 - ❑ returnează adresa șirului destinație
 - ❑ șirul rezultat are un `'\0'` la final
 - ❑ **char* strncpy(char *d, char* s, int n);**
 - ❑ copiază primele n caractere șirul sursă s în șirul destinație d ;
 - ❑ returnează adresa șirului destinație
 - ❑ șirul rezultatul NU are un `'\0'` la final

FUNCȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea elementelor unui tablou a într-un alt tablou b:
 - ❑ nu se poate face prin atribuire ($b=a$), întrucât a și b sunt pointeri constanți;
 - ❑ copierea se face element cu element folosind instrucțiuni repetitive (for, while);
 - ❑ pe cazul general (a și b nu sunt neapărat tablouri de caractere) putem folosi funcții pentru manipularea blocurilor de memorie: **memcpy, memmove**;
 - ❑ lucrează la nivel de octet fără semn (unsigned char)
 - ❑ alte funcții pentru manipularea blocurilor de memorie: **memcmp, memset, memchr**

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea unui tablou – funcțiile **memcpy** și **memmove**
 - ❑ antet: **void* memcpy(void *d, const void* s, int n);**
 - ❑ copiază primii **n** octeți din sursa **s** în destinația **d**;
 - ❑ returnează un pointer la începutul zonei de memorie destinație **d**;
 - ❑ un fel de strcpy extins (merge și pe alte tipuri de date, nu numai pe char-uri)
 - ❑ nu se oprește la octeți = 0 (funcția strcpy se oprește la octeți ce au valoarea 0 = sfârșit de string);
 - ❑ presupune că șirurile destinație și sursa nu se suprapun
 - ❑ dacă cele două șiruri se suprapun funcția prezintă undefined behaviour (comportament nedefinit)

FUNCȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea unui tablou – funcțiile **memcpy** și **memmove**
 - ❑ antet: **void* memcpy(void *d, const void* s, int n);**

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  int main()
6  {
7      int a[] = {25, -36, 0, 91, 7415};
8      int *b = (int*) malloc(sizeof(a));
9      memcpy(b, a, sizeof(a));
10     int i;
11     for (i=0; i<sizeof(a)/sizeof(int); i++)
12         printf("%d ", b[i]);
13     printf("\n");
14
15     float a1[] = {2.0, 3.5, -1.2};
16     float b1[3];
17     memcpy(b1, a1, sizeof(a1));
18     for (i=0; i<sizeof(a1)/sizeof(float); i++)
19         printf("%f ", b1[i]);
20     printf("\n");
21
22     char c[50] = "Ana are mere";
23     memcpy(c+8, c, 12); puts(c);
24
25     return 0;
26 }
```

```
25 -36 0 91 7415
2.000000 3.500000 -1.200000
Ana are Ana are mere
```

FUNCȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea unui tablou – funcțiile **memcpy** și **memmove**
 - ❑ antet: **void* memmove(void *d, const void* s, int n);**
 - ❑ copiază primii **n** octeți din sursa **s** în destinația **d**;
 - ❑ returnează un pointer la începutul zonei de memorie destinație **d**;
 - ❑ identică cu funcția **memcpy** + tratează cazurile de suprapunere dintre **d** și **s**
 - ❑ nu contează că șirurile destinație **d** și sursă **s** se suprapun
 - ❑ folosește un buffer intern pentru copiere

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ copierea unui tablou – funcțiile **memcpy** și **memmove**
 - ❑ antet: **void* memmove(void *d, const void* s, int n);**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main ()
5  {
6      char t[] = "memmove este foarte folositor.....";
7      memmove(t + 20, t + 13, 16);
8      puts(t);
9      return 0;
10 }
```

memmove este foarte foarte folositor.....

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ funcție care elimină toate aparițiile unui șir **t** într-un șir **s**

```
int main()
{
    char s[100], t[100];
    strcpy(s, "abbbccca");
    strcpy(t, "bc");
    eliminaAparitii(s, t);
    printf("%s\n", s);
    return 0;
}
```

abbbccca

Trebuie sa obțin "abbcca"

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

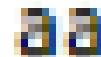
- ❑ funcție care elimină toate aparițiile unui șir **t** într-un șir **s**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void eliminaAparitii(char *s, char* t)
5  {
6      char *p = strstr(s,t);
7      while(p != NULL)
8      {
9          memmove(p,p+strlen(t),s + strlen(s)- (p + strlen(t)) + 1);
10         p = strstr(s,t);
11     }
12 }
13
14 int main()
15 {
16     char s[100],t[100];
17     strcpy(s,"abbbccca");
18     strcpy(t,"bc");
19     eliminaAparitii(s,t);
20     printf("%s\n",s);
21     return 0;
22 }
```


FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- funcție care elimină toate aparițiile unui șir **t** într-un șir **s**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void eliminaAparitii(char *s, char* t)
5  {
6      char *p = strstr(s,t);
7      while(p != NULL)
8      {
9          memmove(p,p+strlen(t),s + strlen(s)- (p + strlen(t)) + 1);
10         p = strstr(s,t);
11     }
12 }
13
14 int main()
15 {
16     char s[100],t[100];
17     strcpy(s,"abbbccca");
18     strcpy(t,"bc");
19     eliminaAparitii(s,t);
20     printf("%s\n",s);
21     return 0;
22 }
```



Nu obțin ceea ce
trebuie. Unde am greșit?

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- funcție care elimină toate aparițiile unui șir **t** într-un șir **s**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void eliminaAparitii(char *s, char* t)
5  {
6      char *p = strstr(s,t);
7      while(p != NULL)
8      {
9          memmove(p,p+strlen(t),s + strlen(s)- (p + strlen(t)) + 1);
10         p = strstr(s,t);
11     }
12 }
13
14 int main()
15 {
16     char s[100],t[100];
17     strcpy(s,"abbbccca");
18     strcpy(t,"bc");
19     eliminaAparitii(s,t);
20     printf("%s\n",s);
21     return 0;
22 }
```



abbcca

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- setarea unor octeți la o valoare – funcția **memset**
 - antet: **void* memset(void *d, char val, int n);**
 - în zona de memorie dată de pointerul **d**, sunt setate primele **n** poziții (octeți) la valoarea dată de **val**. Funcția returnează șirul **d**.

exempluMemset.c

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main ()
5  {
6      char t[] = "nu prea vrem sa vina vacanta!!!";
7      memset(t, '-', 8);
8      puts(t);
9      return 0;
10 }
```

-----vrem sa vina vacanta!!!

FUNCTII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ căutarea unui octet într-un tablou – funcția **memchr**
 - ❑ antet: **void* memchr(const void *d, char c, int n);**
 - ❑ determină prima apariție a octetului **c** în zona de memorie dată de pointerul **d** și care conține **n** octeți. Funcția returnează pointerul la prima apariție a lui **c** în **d** sau NULL, dacă **c** nu se găsește în **d**.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main ()
5  {
6      char t[] = "nu prea vrem sa vina vacanta!!!";
7      char *p = memchr(t, 'm', 25);
8      puts(p);
9
10     return 0;
11 }
```

m sa vina vacanta!!!


FUNCTȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ compararea a două tablouri pe octeți – funcția **memcmp**
 - ❑ antet: **int memcmp(const void *s1, const void * s2, int n);**
 - ❑ compară primii **n** octeți corespondenți începând de la adresele **s1** și **s2**.
 - ❑ Returnează:
 - ❑ 0 dacă octeții sunt identici
 - ❑ ceva mai mic decât 0 dacă **s1 <_L s2**
 - ❑ ceva mai mic decât 0 dacă **s1 >_L s2**

FUNCTȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

- ❑ compararea a două tablouri pe octeți – funcția **memcmp**
 - ❑ antet: `int memcmp(const void *s1, const void * s2, int n);`
 - ❑ compară primii **n** octeți corespondenți începând de la adresele **s1** și **s2**.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main ()
5  {
6      char t[] = "Ana are mere!!!";
7      char s[] = "Ana are pere!!!";
8      int i = memcmp(t,s,6);
9      printf("%d \n",i);
10     i = memcmp(t,s,strlen(t));
11     printf("%d \n",i);
12
13     int v[] = {1,2,4,5,6};
14     int w[] = {1,2,3,7,8};
15     i = memcmp(v,w,8);
16     printf("%d \n",i);
17     i = memcmp(v,w,sizeof(v));
18     printf("%d \n",i);
19
20     return 0;
21 }
```



0
-3
0
1

FUNCȚII PREDEFINITE PENTRU MANIPULAREA BLOCURILOR DE MEMORIE

Funcția	Descriere
<code>void* memcpy (void *dest, void *src, unsigned cnt)</code>	Funcția copiază cnt octeți din zona de memorie src în dest (src și dest trebuie să fie disjuncte) și returnează prin numele său adresa destinație dest.
<code>void* memmove (void *dest, void *src, unsigned cnt)</code>	Funcția copiază cnt octeți din zona de memorie src în dest (nu neapărat disjuncte) și returnează prin numele său adresa sursă src.
<code>void* memchr (void *src, int c, unsigned cnt)</code>	Funcția caută valoarea c în primii cnt octeți din zona de memorie src și returnează prin numele său adresa octetului c sau NULL dacă c nu a fost găsit.
<code>void* memset (void *dest, int c, unsigned cnt)</code>	Funcția scrie valoarea c în primii cnt octeți din zona de memorie dest și returnează prin numele său adresa destinație dest.
<code>int memcmp (void *src1, void *src2, unsigned cnt)</code>	Funcția compară în ordine cel mult cnt octeți din zonele de memorie src1 și src2. Funcția returnează prin numele său valoarea întreagă 0 dacă informația din src1 este identică cu cea din src2; valoarea întreagă -1 dacă primul octet diferit din src1 este mai mic decât octetul corespunzător din src2; valoarea întreagă 1 dacă primul octet diferit din src1 este mai mare decât octetul corespunzător din src2

Cursul 9

1. Șiruri de caractere
2. Funcții specifice de manipulare
3. Funcții pentru manipulare blocuri de memorie

Cursul 10

1. Fișiere binare