Limbaje de Programare
Curs 5 – Șiruri de caractere

Dr. Casandra Holotescu

Universitatea Politehnica Timișoara

Ce discutăm azi...

1. Șiruri de caractere
2. Tipul pointer
3. Funcții cu șiruri de caractere

Șiruri de caractere

În C, un șir de caractere este un tablou de caractere, încheiat în memorie cu caracterul '\0'.
$\Rightarrow$ dimensiunea unui șir = numărul de caractere + 1!

```
char cuvant[20]; // neinitializat

char msg1[] = "test";
// 5 octeti, terminat cu '\0'
char msg2[] = {'t','e','s','t','\0'};
// acelasi ca msg1, scris altfel

char nume[3] = { 'E', 'T', 'C' };
// nu are '\0' la sfarsit !
char sir[20] = "test";
// restul pana la 20 sunt '\0'
```

Șiruri de caractere

În C, un șir de caractere este un tablou de caractere, **încheiat în memorie cu caracterul '\0'**.

Atenție:

- '\0' apare în șir doar **la memorare**, nu apare explicit într-o constantă șir de caractere, nu se citește și nu se tipărește!
- toate funcțiile standard pentru șiruri (din **string.h**) d**epind de prezența lui '\0' la sfârșitul șirului** (astfel nu au nevoie să cunoască lungimea șirului!)
- când specificăm dimensiunea unui șir trebuie să alocăm **dimensiunea dorită + 1** pentru '\0'

Citirea unui șir caracter cu caracter

```
void readString(char a[], int n){
//citim un sir de maxim n caractere

  int i = 0;
  int c = getchar();

  while(c != EOF && i < n){
     a[i] = c;  //punem caract. citit in tablou
     i++;   //avansam indicele tabloului
     c = getchar();  //citim caract. urmator
  }

  a[i] = '\0';
 //marcam sfarsitul sirului
}
```

Citirea și tipărirea formatată

Tipărirea șirului cu **printf**:

```
char sir[20] = "un sir oarecare";
printf ("%s", sir);
```

Citirea șirului cu **scanf**:

```
char sir[20];
scanf ("%19s", &sir);   //adauga '\0' la sf.
scanf ("%19s", sir);    //echivalenta cu &sir
```

Nu se citesc șiruri fără limitare! Nu se citește cu **gets**!

```
scanf ("%s", &sir);   //NU!!
//f. periculos, citeste oricate caractere
//de la intrare, indiferent de dimensiunea
//sirului destinatie!!
```

Tipul pointer

Orice variabilă x are o **adresă**, &**x**, unde se memorează valoarea ei.

– fiind o expresie, &**x** are un tip: tipul **pointer** (adresă)

Pentru o variabilă declarată:

nume-tip x;

adresa **& x** are tipul **nume-tip ***

= pointer la nume-tip, adresa unui obiect de tip nume-tip

Tipul pointer

Numele unui tablou are tipul **pointer la tipul elementului**:

int a[4]; ⇒ *a* are tipul *int **

char s[8]; ⇒ *s* are tipul *char **

La declararea parametrilor funcției:

void f(tip a[ ]) înseamnă de fapt

*void f(tip *a)*

(de aceea dimensiunea: *void f(tip a[6])* nu contează)

Tipul unei constante șir de caractere "sir" este *char **
= adresa unde se găsește șirul în memorie

Valoarea specială **NULL** (0 de tip void * = adresă de tip neprecizat) e folosită pentru a indica o **adresă invalidă**.

Funcții din **string.h**

```
size_t strlen(const char *s);
// returneaza lungimea sirului s

char *strchr(const char *s, int c);
// cauta caract. c in sirul s, returneaza adresa
//unde l-a gasit sau NULL (0) daca nu-l gaseste

char *strcpy(char *dest, const char *src);
// copiaza src in dest

char *strncpy
   (char *dest, const char *src, size_t n);
// copiaza cel mult n caractere din src in dest
```

size_ t: tip întreg fără semn pentru dimensiuni
const: specificator de tip, $\Rightarrow$ obiectul respectiv nu e modificat

Funcții din **string.h**

```
int strcmp (const char *s1, const char *s2);
// compara 2 siruri
// returneaza intreg < 0 sau == 0 sau > 0
// dupa cum e s1 fata de s2

int strncmp
    (const char *s1, const char *s2, size_t n);
//compara sirurile pe lung. cel mult n caractere

char *strcat(char *dest, const char *src);
// concateneaza src la dest
// la dest trebuie sa fie loc suficient

char *strncat
    (char *dest, const char *src, size_t n);
// concat cel mult n caractere din src la dest
```

Lungimea unui șir

```
size_t strlen(const char *s) {
// lungimea sirului, pana la '\0'

// size_t (stddef.h): tip pt. dimensiuni pozitive
// (unsigned sau long unsigned)

  int i=0;
  while (s[i] != '\0')
      i++;  // avanseaza pana intalneste '\0'

   return i; // nr. de caract.; '\0' nu e numarat
}
```

Compararea a două șiruri

```
int strcmp (const char *s1, const char *s2) {
// compara 2 siruri
  int i=0;

  while (s1[i] == s2[i] && s1[i] != '\0')
     i++; // egale dar s1[i] nu e '\0'

  return s1[i] - s2[i];

  // < 0 pt. s1 < s2
  // > 0 pt. s1 > s2
  // == 0 daca s1, s2 egale
}
```

Copierea unui șir

```
char *strcpy(char *dest, const char *src) {
 // copiaza src in dest

  int i=0;
  while (src[i]!='\0'){
  // copiaza pana la '\0'
     dest[i] = src[i];
     i++;
   }
  // trebuie sa avem loc in sirul dest !!!

  dest[i] = '\0';

  return dest; // returneaza dest prin conventie
}
```

Concatenarea a două șiruri

```
char *strcat(char *dest, const char *src) {
 // concateneaza src la dest
 // trebuie sa avem loc in coada lui dest !!!

  int i=0, j=0;
  while (dest[i] != '\0')
    ++i;

  while (src[j] != '\0'){
    dest[i] = src[j];
    ++i; ++j;
  }
  dest[i] = '\0';
  return dest;
}
```

Găsirea unui caracter într-un sir

```
char *strchr(const char *s, char c) {
  //cauta caracterul c in sirul s
  int i=0;

  while (s[i] != '\0'){
    if (s[i] = = c)
       return &s[i];
       //returneaza adresa caract. in sir
    i++;
  }

  return NULL;
  //returneaza NULL daca nu a fost gasit
}
```

Exemple

```
int indiceMax(char *tab[], int n){
//max unui tablou de siruri de caractere

  int i=0, index=0;

  for(i=0;i<n;i++){
     if(strcmp(tab[i], tab[index]) > 0)
     //daca sirul tab[i] > sirul tab[index]
          index=i;
   }

   return index;
}
```

Exemple

```
void all(char *tab[], int n, char *rez, int rn){
//concateneaza elementele unui tablou de siruri

 int i=0;
 if(strlen(tab[0]) < rn)  //daca e loc in rez
     strcpy(rez, tab[0]); //copiem primul sir

 for(i=1;i<n;i++)
   if(strlen(rez) + strlen(tab[i]) + 2) < rn){
    //daca mai avem loc in sirul rez

      strcat(rez, ", "); //cu virgula intre
      strcat(rez, tab[i]);
   }
}
```