ŞIRURI DE CARACTERE

1. Conţinutul lucrării

În lucrare se prezintă modul de reprezentare în memorie a unui şir de caractere şi unele funcţii standard de prelucrare a şirurilor de caractere.

2. Consideraţii teoretice

2.1. Reprezentarea în memorie a unui şir de caractere

Un şir de caractere este păstrat într-un tablou unidimensional de tip char. Fiecare caracter se păstrează într-un octet prin codul ASCII al său. Ultimul caracter al şirului, deci terminatorul şirului, este caracterul NUL (‘\0’).

Numele tabloului care păstrează şirul de caractere este un pointer constant spre şirul de caractere.

Exemplu:

char sir []=”SIR DE CARACTERE”;

În memorie, reprezentarea sa va fi (în hexazecimal):

sir	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	53	49	52	20	44	45	20	43	41	52	41	43	54	45	52	45	00

sir are ca valoare adresa zonei de memorie care conţine şirul.

Avem următoarele relaţii:

- sir[i] , unde i $\in$ {0,…,16} – reprezintă codul ASCII al celui de al i-lea caracter din şirul de caractere;
- sir+i , unde i$\in${0,…,16}, reprezintă adresa celui de al i-lea caracter din şirul de caractere;
- *(sir+i) are acelaşi efect ca sir[i].

Tot ce s-a explicat mai sus, rămâne valabil şi în cazul declarării în felul următor:

char *const sir=”SIR DE CARACTERE”;

Declararea unui tablou de şiruri de caractere se poate face astfel:

char *tab[]={sir_0,sir_1,…,sir_n};

În acest caz, tab[i], pentru i $\in${0,…,n}, este un pointer spre şirul de caractere “sir_i”.

În cazul apelului

printf(“%s\n”, tab[i]);

se va afişa textul stocat în **sir_i.**

2.2. Funcții standard de prelucrare a șirurilor de caractere

Funcţiile standard de citire/scriere a şirurilor de caractere:

- gets/puts;
- scanf/printf;
- sscanf/sprintf

au fost prezentate în lucrarea de laborator nr. 2.

În continuare sunt prezentate câteva funcţii de prelucrare a şirurilor de caractere, al căror prototip se găseşte în fişierul **string.h**

2.2.1. Lungimea unui şir de caractere

Lungimea unui şir de caractere, fără a fi luat în considerare caracterul '\0', este returnat de funcţia **strlen**, care are prototipul:

unsigned strlen (const char *s);

Exemplu:

```
/* Programul L9Ex1 */

/* Programul exemplifica utilizarea functiei  strlen */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define alfa "Apasati o tasta!"

int main()
{
  char sir1[]="SIR DE CARACTERE";
  char  *sir2="SIR DE CARACTERE";
  int n1,n2,n3;
  n1=strlen(sir1);
  n2=strlen(sir2);
  n3=strlen("SIR DE CARACTERE");
  /* Atat n1,cat si n2 si n3 au ca valoare 16 */
  printf("n1=%d n2=%d n3=%d\n",n1,n2,n3);
  printf("%s\n",alfa);
  getch();
  return 0;
}
```

2.2.2. Copierea unui şir de caractere

Copierea unui şir de caractere dintr-o zonă de memorie de adresă **sursă** într-o altă zonă de memorie de adresă **dest** se face cu ajutorul funcţiei **strcpy**, al cărei prototip este:

char *strcpy (char *dest, const char *sursă);

Se menţionează că are loc copierea inclusiv a caracterului NUL (‘\0’). Funcţia returnează adresa unde a avut loc copierea, adică chiar destinaţia.

Pentru a copia cel mult n caractere, din zona de memorie de adresă **sursă** în zona de memorie de adresă **dest**, se va folosi funcţia **strncpy**, al cărei prototip este următorul:

char *strncpy (char *dest, const char *sursă, unsigned n);

După ultimul caracter transferat, trebuie pus caracterul ‘\0’. Evident că dacă n este mai mare decât lungimea şirului de la adresa sursă, atunci are loc transferarea întregului şir de caractere.

Exemplu:

```
/* Programul L9Ex2 */

/* Programul exemplifica utilizarea functiei strcpy */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define alfa "\nApasati o tasta!"

int main()
{
  char sir1[]="SIR DE CARACTERE";
  char  *sir2="SIR DE CARACTERE";
  char sir3[100],sir4[100],sir5[100];
  strcpy(sir3,sir1);
  printf("sir3 contine: %s\n",sir3);
  strcpy(sir4,"Functii standard de prelucrare siruri de caractere");
  printf("sir4 contine: %s\n",sir4);
  strncpy(sir5,sir2,6);          /* sir5 contine SIR DE */
  sir5[6]='\0';
  printf("sir5 contine: %s\n",sir5);
  printf(alfa);
  getch();
  return 0;
}
```

2.2.3. Concatenarea a două şiruri de caractere

Adăugarea şirului de caractere de la adresa **sursă** după ultimul caracter (cel care precede NUL) al şirului de caractere de la adresa **dest** se face cu ajutorul funcţiei **strcat**, care are prototipul:

char *strcat(char *dest, const char *sursă);

După şirul rezultat, se pune evident caracterul NUL (‘\0’). Funcţia retunează valoarea adresei destinaţie.

Există posibilitatea de a prelua din şirul de caractere de la adresa sursă numai n caractere, cu ajutorul funcţiei **strncat,** care are prototipul:

char *strncat (char *dest, const char *sursa, unsigned n);

La sfârşit se pune automat caracterul NUL (‘\0’). Dacă n este mai mare decât lungimea şirului de la adresa sursă, atunci funcţia **strncat** are acelaşi efect ca şi funcţia **strcat**.

Exemplu:

```
/* Programul L9Ex3 */

/* Programul exemplifica utilizarea functiei strcat */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define alfa "\nApasati o tasta!"

int main()
{
  char sir1[100]="SIR1 DE CARACTERE";
  char  *sir2="<SIR2 DE CARACTERE";
  char sir3[100];
  strcpy(sir3,sir1);
  strcat(sir1,sir2);
  printf("sir1 contine: %s\n",sir1);
  strncat(sir3,sir2,5);
  /* Dupa ultimul caracter din sir3 se pune implicit '\0' */
  for (int i=0;i<=strlen(sir3);++i) printf("%02d %02X %c\n",sir3[i], sir3[i], sir3[i]);
  printf("\nsir3 contine: %s\n",sir3);
  printf(alfa);
  getch();
  return 0;
}
```

2.2.4. Compararea a două şiruri de caractere

Compararea a două şiruri de caractere se face caracter cu caracter (pe baza codurilor ASCII) până când:

- s-a ajuns la un caracter i din primul şir care diferă de caracterul i din al doilea şir;
- s-a ajuns la sfârşitul unuia din şiruri sau a ambelor.

Compararea a două şiruri de caractere de la adresele **sir1** şi respectiv **sir2** se poate face cu funcţia de prototip:

int strcmp(const char *sir1,const char *sir2);

Funcţia returnează:

- o valoare negativă dacă şirul de caractere de la adresa **sir1** este mai mic decât cel de la adresa **sir2**;
- zero dacă şirurile sunt egale;
- o valoare pozitivă, dacă şirul de la adresa **sir1** este mai mare decât cel de la adresa **sir2**.

Dacă dorim să se compare numai primele n caractere din cele două şiruri se foloseşte funcţia de prototip:

int strncmp (const char *sir1, const char *sir2, unsigned n);

Dacă dorim să nu se facă distincţie între literele mici şi cele mari, atunci cele două funcţii au drept corespondenţe:

int stricmp (const char *sir1, const char *sir2);
int strnicmp (const char *sir1, const char *sir2, unsigned n);

Exemplu:

```
/* Programul L9Ex4 */

/* Programul exemplifica utilizarea functiei strcmp*/
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define alfa "\nApasati o tasta!"

int main()
{
  char sir1[100]="SIR DE CARACTERE";
  char  *sir2="SIR de caractere";
  int i,j,k,l;
  i=strcmp(sir1,sir2);/* i<0 , rezulta sir1<sir2 */
  printf("i=%d\n",i);
  j=strncmp(sir1,sir2,3);/*j=0 ,rezulta ca primele 3 caractere din sir1 si sir2 sunt egale */
  printf("j=%d\n",j);
  k=stricmp(sir1,sir2); /* k=0, rezulta ca cele 2 siruri sunt egale */
  printf("k=%d\n",k);
  l=strnicmp(sir1,"SIR de 10 caractere",6); /*l=0 */
  printf("l=%d\n",l);
  printf(alfa);
  getch();
  return 0;
}
```

3. Mersul lucrării

3.1. Se vor analiza şi executa programele din lucrare şi din materialul de curs.

3.2. Se va scrie o funcţie care să realizeze extragerea dintr-un şir de caractere sursă a unui subşir specificat prin poziţia în cadrul sursei şi a numărului de caractere extrase.

3.3. Se va scrie o funcţie pentru inserarea unui şir de caractere sursă într-un şir de caractere destinaţie, specificând poziţia din care începe inserarea.

3.4. Se va scrie o funcţie pentru ştergerea unui subşir dintr-un şir de caractere dat. Subşirul se va specifica prin poziţie şi număr de caractere.

3.5. Se va scrie o funcţie pentru a verifica dacă un şir dat este subşir al unui alt şir de caractere. În caz afirmativ, se va specifica poziţia pe care se regăseşte pentru prima dată.

3.6. Să se scrie două funcţii: una care converteşte un număr întreg sau real într-un şir de caractere, iar cealaltă care face operaţia inversă.

3.7. Să se scrie un program care citeşte n şiruri de caractere şi afişează şirul cel mai lung şi şirul cel mai mare alfanumeric.

3.8. Să se scrie un program care citeşte șiruri de caractere reprezentând numere întregi sau numere reale separate prin spații și care afișează suma valorilor întregi și suma valorilor reale, ignorând (dacă există) acele șiruri de caractere care nu reprezintă numere.