Laborator 8 – Exerciții rezolvate

Să se rezolve prin realizarea programului C următoarele probleme:

Problema 1. Să se scrie o funcție C care primind ca si parametru 2 șiruri de caractere returnează un număr negativ dacă primul sir este mai mic, 0 dacă cele 2 șiruri sunt egale si un număr pozitiv dacă primul sir este mai mare decât al 2-lea. Comparațiile sunt lexicografice. (implementare pentru funcția strcmp).

Rezolvare. Funcția mystrcmp(s, t) compara șirurile de caractere s și t și returnează un număr negativ, zero sau pozitiv în funcție de relația lexicografică dintre s și t; care poate fi: s<t, s=t sau s>t. Valoarea returnată este obținută prin scăderea caracterelor ASCII de pe prima poziție unde s diferă de t.

Varianta cu tablouri:

Metoda 1

```
#include <stdio.h>

// returnează <0 daca s<t,
// 0 daca s==t,
// >0 daca s>t
int mystrcmp(char s[], char t[])   {
  int i = 0; // ne poziționam pe primul caracter din ambele șiruri
  while (s[i] == t[i])
    if (s[i++] == '\0')// ne poziționam pe următoarea pereche de caractere
      return(0); //daca am ajuns la sfarsitul ambelor siruri si ele sunt egale
caracter cu caracter returnam 0
    return(s[i] - t[i]);//returnam diferența dintre codurile ASCII ale primei
perechi de caractere diferite
}

int main()
{
  int rez;
  char s[4] = "Ana"; // sau s[4] = {'A','n','a', '\0'};
  char t[9] = "Anamaria";

  rez = mystrcmp(s, t);

  if (rez > 0)
    printf("sirul %s e mai mare decat %s",s,t);
  else if (rez < 0)
    printf("sirul %s e mai mic decat %s",s,t);
  else
    printf("sirul %s e egal cu %s",s,t);
```

```
  return 0;
}
```

Rulare:

```
Sirul Ana e mai mic decat Anamaria
```

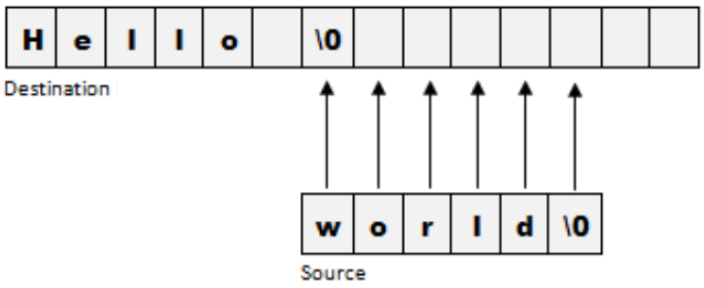
Metoda 2

```
int mystrcmp(char a[], char b[])
{ int i;
  for(i=0;a[i]&& b[i] && a[i]==b[i];i++);
  //ieșim din ciclu dacă unul din cele 2 șiruri a ajuns la sfârșit
  //prin întâlnirea octetului nul sau când elementele
  //corespunzătoare din cele 2 șiruri diferă
  return (a[i]-b[i]);
}
```

Problema 2. Să se scrie o funcție C care primind ca și parametru 2 șiruri de caractere, le concatenează cu obținerea noului sir in primul argument al funcției (implementare pentru funcția strcat).

Rezolvare. Funcția mystrcat concatenează șirul sursă la șirul destinație. Funcția returnează șirul destinație. Șirul destinație trebuie să aibă suficientă memorie alocată pentru a acomoda șirul rezultat. Marcatorul de sfârșit de sir ('\0') al șirului destinație este șters si in locul sau e copiat primul caracter din șirul sursa. Vom copia mai apoi toate celelalte caractere din șirul sursa inclusiv marcatorul de sfârșit al acestuia ('\0').

strcat



Varianta cu tablouri:

Metoda 1

```
#include <stdio.h>

const char* mystrcat(char dest[], char s[])
{ // il facem pe i sa refere marcatorul de sfarsit al lui dest
```

```
  int i = 0, j=0;
  while(dest[i])i++;//sau while(dest[i]!='\0') i++;
  // Adaugam caracterele lui s la finalul lui dest
  while((dest[i++]=s[j++])); // sau while(s[j]!='\0') dest[i++]=s[j++]; dest[i]
='\0';
  //strcat() intoarce pointerul inceputul sirului dest
  return dest;
}

int main () {
  char dest[100]= "Hello ";
  printf("%s",mystrcat(dest, "world"));
  return 0;
}
```

Rulare:

```
Hello world
```

Metoda 2

```
const char* mystrcat(char a[], char b[])
{ int i,j;
  for(i=0;a[i];i++);      //ajungem cu i la sfârșitul șirului a
  for(j=0;a[i]=b[j];i++,j++);
  //ieșim din acest ciclul for când ajungem la sfârșitul
  //șirului b adică vom atribui lui a[i] valoarea 0
  return a;
}
```

Problema 3. Se citește un text de la tastatură, terminat prin caracterul sfârșit de rând (enter). Să se scrie programul C care determină numărul de apariții ale fiecărei litere din șir – literele mici și mari se consideră împreună.

Rezolvare. Cream o tabela de frecvențe (histograma) cu 27 de poziții pentru toate literele din alfabet in care poziția 0 corespunde literelor A si a, poziția 1 literelor B si b, ..., poziția 26 literelor Z si z. Inițializăm tabela de frecvente cu 0 pentru toate literele. Parcurgem textul si de fiecare data când întâlnim o litera incrementam nr. de apariții din tabela corespunzător acesteia cu o unitate. Afișăm numărul de apariții al fiecărei litere doar daca acesta este mai mare decât 0.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main(void)
```

```
{
  char sir[80];
  int aparitii[27],i;
  //cream o tabela de frecventa (histograma) cu 27 de pozitii
  //pentru toate literele din alfabet
  //pozitia 0 corespunde literelor A si a iar pozitia 26 literelor Z si z
  puts("Intr sirul=");gets(sir);

  //initilizam tabela de frecvente cu 0 pentru toate literele
  for(i=0;i<27;i++) aparitii[i]=0;


  for(i=0;i<strlen(sir);i++)
    //de fiecare data cand intalnim o litera
    //incrementam nr. de apartii corepunzator din tabela cu 1
    if(isalpha(sir[i])) aparitii[toupper(sir[i])-'A']++;

  //afisam numarul de aparitii al fiecarei litere daca acesta este mai mare
decat 0
  for(i=0;i<27;i++)
    if(aparitii[i]) printf("Literele %c sau %c apar de %d
ori\n",'a'+i,'A'+i,aparitii[i]);

    return 0;
}
```

Rulare:

```
Introdu șirul:
Ana are mere.

Literele a sau A apar de 3 ori
Literele e sau E apar de 3 ori
Literele m sau M apar de 1 ori
Literele n sau N apar de 1 ori
Literele r sau R apar de 2 ori
```