Siruri de caractere

Siruri de caractere in C

- cum sunt reprezentate sirurile de caractere in C?
- declaratii si initializari
- prelucrari elementare
  - accesarea elementelor
  - parcurgerea sirurilor
  - compararea lexicografica a 2 siruri de caractere
  - citirea unui sir de caractere
- colectii de siruri
- sortarea unui sir de caractere vs. sortarea unei colectii de siruri
- functiile din biblioteca <string.h>
- greseli frecvente

remember?

- int fcmp (const void*a, const void*b)

- Ce inseamna?
  - const void*a
  - void const* a

- const void* a vs void* const a
- (points to constant data) vs (points to constant memory address)

- Atunci ce inseamna?
  - const void* const a
  - void const* const a

Constante de tip pointer

- câte erori de compilare avem în codul următor?

```
int main()
{
        int x=5;
        const int*p=&x;
        int* const q=&x;
        const int* const r=&x;
        printf(" x=%d *p=%d *q=%d *r=%d\n",x,*p,*q,*r);
        p++;
        (*p)++;
        q++;
        (*q)++;
        r++;
        (*r)++;
        return 0;
```

```c
int main()
{
        int x=5;
        const int*p=&x;
        int* const q=&x;
        const int* const r=&x;
        printf(" x=%d *p=%d *q=%d *r=%d\n",x,*p,*q,*r);
        p++;
        (*p)++;
        q++;
        (*q)++;
        r++;
        (*r)++;
        return 0;


}
```

```
const.c:9:2: error: increment of read-only location ‘*p’
  (*p)++;
  ^
const.c:10:2: error: increment of read-only variable ‘q’
  q++;
  ^
const.c:12:2: error: increment of read-only variable ‘r’
  r++;
  ^
const.c:13:2: error: increment of read-only location ‘*r’
  (*r)++;
  ^
```

```c
int main()
{
        int x=5;
        const int*p=&x;  //valoarea de la adresa p este constanta
        int* const q=&x;            //pointerul q nu poate fi modificat sa
        const int* const r=&x;    //puncteze spre o alta adresa
        printf(" x=%d *p=%d *q=%d *r=%d\n",x,*p,*q,*r);
        p++;
        (*p)++;    //nici valoarea de la adresa r nu poate fi
        q++;       //modificată și nici pointerul r nu poate fi
        (*q)++;    //făcut să puncteze către altă adresă
        r++;
        (*r)++;
        return 0;


}
```

Cum sunt reprezentate sirurile de caractere in C?

- sir de caractere = vector cu elemente de tip char
- delimitat de
  - adresa de inceput
  - un character special numit null terminator (‘\0’) marcheaza sfarsitul sirului
- sirul “abcde” va fi reprezentat in memorie pe 6 octeti

- sirul de caractere trebuie sa aiba rezervata o zona de memorie
- in zona de memorie rezervata trebuie sa poata fi incadrat tot sirul + null terminatorul.

declaratii si initializari

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

```
abcde 4
abcde 100
u<illegible> 50
abcde 6
2 4
```

```
printf( "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n",
        sir1,sizeof(sir1),
        sir2,sizeof(sir2),
        sir3,sizeof(sir3),
        sir4,sizeof(sir4),
        sir5,sizeof(sir5)
        );
```

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

```
abcde 4
abcde 100
u 50
abcde 6
2 4
```

- sir1 – pointer catre o zona de memorie de 6 caractere (vom reveni :P)
- sir2 – vector de dimensiune maxima 100 de octeti initializat cu abcde\0
- sir3 – vector de maxim 50 de octeti cu valori neinitializate
- sir4 – vector de maxim 6 octeti initializat cu abcde\0
- sir5 – pointer catre un caracter, neinitializat

accesarea elementelor

- char sir[]=“ana are mere”; char c;
- sir[index] acceseaza caracterul de la indicele index
- putem sa citim valoarea
  - c=sir[5];
- sau sa o modificam
  - sir[0]=‘i’;
- daca sir[index]==‘\0’ -> am ajuns la capatul sirului

accesarea elementelor - citire

```
printf("%c %c %c %c %c",sir1[0],sir2[1],sir3[2],sir4[3],sir5[4]);
```

```
a b  d
```

- sir3 si sir5 nu au valorile initializate. Accesarea valorii de la adresa sir5+4 ar putea da segmentation fault (undefined behavior)

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

accesarea elementelor - scriere

```
sir1[2]='\0';
sir2[2]='\0';
sir3[2]='\0';
sir4[2]='\0';
sir5[2]='\0';
printf( "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n",
        sir1,sizeof(sir1),
        sir2,sizeof(sir2),
        sir3,sizeof(sir3),
        sir4,sizeof(sir4),
        sir5,sizeof(sir5)
        );
```

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

ce afiseaza?

accesarea elementelor - scriere

```
sir1[2]='\0';
sir2[2]='\0';
sir3[2]='\0';
sir4[2]='\0';
sir5[2]='\0';
printf( "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n"
                "%s %d\n",
        sir1,sizeof(sir1),
        sir2,sizeof(sir2),
        sir3,sizeof(sir3),
        sir4,sizeof(sir4),
        sir5,sizeof(sir5)
        );
```

```
char *sir1="abcde";
char sir2[100]="abcde";
char sir3[50];
char sir4[]="abcde";
char *sir5;
```

```
Segmentation fault (core dumped)
```

- char*sir1=“abcde”;
- sir1 – pointer catre o zona de memorie de 6 caractere
- abcde = constanta, sir1 este un pointer catre o valoare constanta.
- atunci cand incercam sa modificam sirul =>segmentation fault
- corect este

```
const char *sir1="abcde";
```

```
sir1[2]='\0';
```

```
p1.c: In function ‘main’:
p1.c:23:9: error: assignment of read-only location ‘*(sir1 + 2u)’
  sir1[2]='\0';
         ^
```

```
//sir1[2]='\0';
sir2[2]='\0';
sir3[2]='\0';
sir4[2]='\0';
//sir5[2]='\0';
printf( "%s\n%s\n%s\n",sir2,sir3,sir4);
```

Parcurgerea sirurilor

```
int i; char*p;
for(i=0;sir2[i]!='\0';i++)
        printf("%c\n",sir2[i]);

for(p=sir2;*p!='\0';p++)
        (*p)++;

printf("%s\n",sir2);
```

```
a
b
c
d
e
bcdef
```

- se pot folosi atat indici cat si pointeri
- se parcurge sirul pana la intalnirea caracterului ‘\0’

Compararea a 2 siruri

- “Popa” vs. “Popescu”
- daca sortam alfabetic Popa < Popescu
- comparam caracter cu caracter
  - daca avem 2 caractere diferite intoarcem nr negativ daca cel din primul sir este mai mic
  - daca sunt egale continuam pana se termina unul din siruri
    - daca se termina doar unul acel sir este “mai mic”
    - daca se termina ambele in acelasi timp sunt egale
- sau putem folosi functia int strcmp(const char*a, const char*b);

citirea unui sir de caractere

- char * fgets (char * str, int num, FILE * stream);
- Reads characters from stream and stores them as a C string into str until (num-1) characters have been read or either a newline or the end-of-file is reached, whichever happens first.

- A newline character makes fgets stop reading, but it is considered a valid character by the function and included in the string copied to str.

- A terminating null character is automatically appended after the characters copied to str.

colectii de siruri

- char s[100][100] – defineste o colectie de 100 de siruri fiecare avand maxim 100 de caractere.
- char*s[100] – defineste un vector de 100 de pointeri care puncteaza catre zone de memorie neinitializate
  - inainte de a-l folosi trebuie sa-i alocam memorie

functii din <string.h>

Copying:

memcpy	Copy block of memory (function)
memmove	Move block of memory (function)
strcpy	Copy string (function)
strncpy	Copy characters from string (function)

Concatenation:

strcat	Concatenate strings (function)
strncat	Append characters from string (function)

Comparison:

memcmp	Compare two blocks of memory (function)
strcmp	Compare two strings (function)
strcoll	Compare two strings using locale (function)
strncmp	Compare characters of two strings (function)
strxfrm	Transform string using locale (function)

functii din <string.h>

Searching:

memchr	Locate character in block of memory (function)
strchr	Locate first occurrence of character in string (function)
strcspn	Get span until character in string (function)
strpbrk	Locate characters in string (function)
strrchr	Locate last occurrence of character in string (function)
strspn	Get span of character set in string (function)
strstr	Locate substring (function)
strtok	Split string into tokens (function)

Other:

memset	Fill block of memory (function)
strerror	Get pointer to error message string (function)
strlen	Get string length (function)

si un exemplu

strstr

```
const char * strstr ( const char * str1, const char * str2 );
      char * strstr (       char * str1, const char * str2 );
```

Locate substring

Returns a pointer to the first occurrence of *str2* in *str1*, or a null pointer if *str2* is not part of *str1*.

The matching process does not include the terminating null-characters, but it stops there.

Parameters

`str1`

C string to be scanned.

`str2`

C string containing the sequence of characters to match.

Return Value

A pointer to the first occurrence in *str1* of the entire sequence of characters specified in *str2*, or a null sequence is not present in *str1*.

de cate ori apare sirul 2 in sirul 1?

```
//de cate ori apare s2 in s1
int main()
{
        char s2[]="abc";
        char s1[]="abcdeabcaabc abacabc";
        char*p,*s;
        s=s1;
        int count=0;
        while(p=strstr(s,s2))
        {
                count++;
                s=p+strlen(s2);
        }
        printf("apare de %d ori\n", count);
        return 0;


}
```