Siruri de caractere

#include<cstring>

char s[1000], \*sir;

s = un sir de max 1000 de caractere

sir = sir de lungime variabila

Important

* ‘\0’ = caracterul de sfarsit de sir
* se initializeaza automat din pozitia 0
* char ocupa un octet

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | n | f | o | r | m | a | t | i | c | i | a | n |
| 100 | 101 | 102 | 103 | 104 | 105 |  |  |  |  |  |  |  |

s

s => “informatician”

s + 3 => “omatician”

* codul ASCII : litere mari ‘A’:65…’a’:97…diferenta este 32

char ch; litera mare

* litera mica : ch-‘A’ + ‘a’; ch + 32
* din litera mica in litera mare : ch – ‘a’ + ‘A’; ch – 32
* cout << ch => cout << (char)ch

Citire:

char s[1000];

cin >> s; citeste pana la spatiu sau enter

“Ana are mere” => s = “Ana”

cin.gets(a, dim)

cin.gets(a, 1000) citeste pana la Enter (fara ‘\n’)

cin.get() citeste un character (citeste ‘\n’)

Afisare:

cout << s;

Prelucrare:

1. Lungime sirului : strlen(sir)
2. Copiere

strcpy(dest, sursa)

strncpy(dest, sursa, lg)

strncpy(s, “informatica”, 4) => s = “info”

1. Concatenare:

strcat(dest, sursa)

strncat(dest, sursa, lg)

strcpy(s, “Ionescu”)

strcat(s, “ ”)

strcat(s, “Ion”) => s = “Ionescu Ion”

strncat(s, “Vasile”, 2) => s = “Ionescu IonVa”

1. Cautare in sir:

* character

strchr(sir, caracter)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | n | f | o | r | m | a | t | i | c | i | a | n |
| 100 | 101 | 102 | 103 | 104 | 105 | 106 |  |  |  |  |  |  |

s s+1 s+2 p

char \*p;

p = strchr(“informatician”, ‘a’)

if(p != NULL) => ‘a’ exista in sir

subsirul p => p = “atician”

pozitia literei cautate in sir

p – s (106 – 100 = 6 = pozitia pe care se gaseste prima aparitie a literei cautate)

cout << s[p-s] (‘a’)

Ex : sa se numere cate vocale sunt in sir

char s[100];

int k;

cin.gets(s, 1000);

k = 0;

for(int i = 0; I < strlen(s); i++){

if(strchr(“aeiouAEIOU”, s[i])){

k++;

}

}

* subsir

strstr(sir, subsir)

Functii pentru caractere

is…(ch) => raspund la interbare daca ch este

to…(ch) => fac conversie

|  |  |
| --- | --- |
| Functie | Effect |
| isalnum(ch) | Ch este litera sau cifra |
| isalpha(ch) | Ch este litera |
| isdigit(ch) | Ch este cifra in baza 10 |
| isupper(ch) | Litera mare |
| islower(ch) | Litera mica |
| isxdigit(ch) | Cifra in baza 16 |
| ch = toupper(ch) | Face conversia in litera mare |
| ch = tolower(ch) | Face conversia in litera mica |

strtok

char \*p;

folosind for

for(p = strtok(sir, separat); p!=0; p = strtok(NULL, separat))

p = subsirul care se formeaza intre separatori

p = strtok(sir, separat)

while(p){

prelucrare p;

p = strtok(NULL, separat)

}

Ex:

separ cuvintele din text

char s[10000], \*p, a[1000][30];

int n = 0;

cin.get(s, 10000);

for(p = strtok(s, “.,;:?!”); p != 0; p = strtpk(NULL, “.,;:?!”))

strcpy(a[n++], p);

Initializarea unui sir

* sirul vid = “”
* prin citire
* declaratie char v = “aeiou”

Functii de comparare

sirurile a, b;

a < b lexicografic(dictionar)

a < v ⬄ exista i astfel incat a[i] < b[i] si oricare j < i => a[j]==b[j]

strcmp(s1, s2) – compara sirurile folosind codurile ASCII

strncmp(s1, s2, lg) – compara sirurile pe primele lg caractere

stricmp(s1, s2) – nu este case sensitive

< 0 ⬄ s1 < s2 lexicografic

= 0 ⬄ s1 == s2

> 0 ⬄s1 > s2 lexicografic

strcmp(“abc”, “Abcd”) > 0 “abc” > “Abcd”

strncmp(“abc”, “Abcd”, 3) > 0

stricmp(“abc”, “Abc”) == 0

Stergerea caracterelor

1. se elimina din sir toate vocalele

char s[1000], a[1000], v[] = “aeiou”;

cin.gets(s, 1000)

for(i = 0; i < strlen(s); i++){

if(strchr(v, s[i])){

strcpy(a, s + 1 + i);

strcpy(s + i, a);

}

}