1. **ŞIRURI DE CARACTERE**
   1. **SCOPUL LUCRĂRII**

Scopul lucrării este exersarea multiplelor funcţii din bilioteca <string.h> pentru exploatarea şirurilor de caractere.

* 1. **BREVIAR TEORETIC**
     1. **Tipul char**

cel mai mic întreg: ocupă 8 biţi, numiţi şi un octet (un octet are opt biţi).

Intervalul de numere care se pot reprezenta pe 8 cifre binare este de la 0 la 28-1, adică 0..255, cu totul 256 de valori; deoarece avem şi numere negative, intervalul se translatează cu 128, ceea ce înseamnă că un char este un număr întreg din intervalul -128..127.

În acelaşi timp, ştim că el este un tip dual, poate fi văzut atît ca întreg pe 8 biţi cît şi ca caracter. Dacă vrem să îl tipărim, putem folosi în C formatul %d (dacă vrem să afişăm valoarea lui întreagă), respectiv %c (pentru a-l afişa ca un caracter).

În C++ putem folosi cout<<.

***Constantele caracter*** sunt incadrate intre apostrofuri.

***Exemplu:***

'a'  //tip char

O constantă caracter are ca valoare codul ASCII al caracterului pe care il reprezinta.

<http://www.asciitable.com/>

Acest set de caractere are urmatoarele proprietati:

* Fiecarui caracter ii corespunde o valoare intreaga distincta (ordinala);
* Valorile ordinale ale literelor mari sunt ordonate si consecutive ('A' are codul ASCII 65, 'B' - codul 66, 'C' - codul 67, etc.);
* Valorile ordinale ale literelor mici sunt ordonate si consecutive ('a' are codul ASCII 97, 'b' - codul 98, 'c' - codul 99, etc.);
* Valorile ordinale ale cifrelor sunt ordonate si consecutive ('0' are codul ASCII 48, '1' - codul 49, '2' - codul 50, etc.).

***Constante caracter corespunzătoare caracterelor imprimabile.***

O constantă caracter corespunzatoare unui caracter imprimabil se reprezinta prin caracterul respectiv inclus intre apostrofuri.

***Exemplu:***

**Constanta caracter  Valoare**   
  ‘A’   65   
  ‘a’   97   
  ‘0’   48   
  ‘\*’   42

Excepţii de la regula de mai sus le constituie caracterele imprimabile ***apostrof*** (') si ***backslash*** (\).

Caracterul backslash se reprezinta: '\\'.

Caracterul apostrof se reprezinta: '\''.

***Constante caracter corespunzătoare caracterelor neimprimabile***

Pentru caracterele neimprimabile, se folosesc ***secvenţe escape***.

O ***secvenţă escape*** furnizează un mecanism general şi extensibil pentru  reprezentarea caracterelor invizibile sau greu de obţinut.

În continuare, sunt prezentate câteva caractere escape utilizate frecvent:

‘\n’ 10 LF rand nou (Line Feed)

‘\t’ 9 HT tabulator orizontal

‘\r’ 13 CR pozitioneaza cursorul in coloana 1 din randul curent

‘\f’ 12 FF salt de pagina la imprimanta (Form Feed)

‘\a’ 7 BEL activare sunet

O constantă caracter pentru o secvenţă escape poate apare însă şi sub o formă în care se trece codul ASCII, în octal, al caracterului dorit:

  ’\ddd’  - unde d este o cifră octală

***Exemple:***

 ’\11’ (pentru ’\t’) - reprezintă constanta caracter ***backspace***, cu codul 9 în baza 10, deci codul 11 în baza *8.*

*’\15’ (pentru* ’\r’) - reprezintă constanta caracter ***CR***, cu codul 13 în baza 10, deci codul 15 în baza 8.

***Exerciţiu:***

Să se scrie urmatorul program şi să se urmarească rezultatele execuţiei acestuia.

*#include <iostream.h>*

*#include <conio.h>*

*void main(void){*

*clrscr();*

*cout<<"Un caracter este memorat pe "<<sizeof(char)<<" octet\n";*

*cout<<"Caracterul escape \\n este memorat pe ";*

*cout<<sizeof('\n')<<" octet\n";*

*cout<<"Caracterul escape '\\n\' este memorat pe "<<sizeof('\n');*

*cout<<" octet\n";*

*cout<<"Caracterul '9' este memorat pe "<<sizeof('9')<<" octet\n";*

*cout<<'B';cout<<' ';cout<<'c';cout<<'\t';*

*cout<<'\t';cout<<'9';cout<<'\b';cout<<'\t';*

*cout<<'L';cout<<'\t';cout<<'L';*

*cout<<'\'';cout<<'\t';cout<<'\"';cout<<'\\';getch();*

*}*

În fişierul **<ctype.h>** există o serie de funcţii (codificate ca macroinstrucţiuni) care primesc un parametru întreg (ce se converteşte în **unsigned char)**, şi întorc rezultatul diferit de 0 sau egal cu 0, după cum caracterul argument satisface sau nu condiţia specificată:

**islower(c) 1 dacă c∈ {‘a’..’z’}**

**isupper(c) 1 dacă c∈ {‘A’..’Z’}**

**isalpha(c) 1 dacă c∈ {‘A’..’Z’}∨{‘a’..’z’}**

**isdigit(c) 1 dacă c∈ {‘0’..’9’}**

**isxdigit(c) 1 dacă c∈ {‘0’..’9’}∨{‘A’..’F’}∨{a’..’f’}**

**isalnum(c) 1 dacă isalpha(c)||isdigit(c)**

**isspace(c) 1 dacă c∈ {‘ ‘,’\n’,’\t’,’\r’,’\f’,’\v’}**

**isgraph(c) 1 dacă c este afişabil, fără spaţiu**

**isprint(c) 1 dacă c este afişabil, cu spaţiu**

**iscntrl(c) 1 dacă c este caracter de control**

**ispunct(c) 1 dacă isgraph(c) && !isalnum(c)**

* + 1. **Şiruri de caractere**

În limbajul C/C++, şirurile de caractere sunt vectori având tipul de bază ***char***.

Un şir de caractere se termina prin marcatorul \0, sau caracterul ***nul***.

De exemplu, şirul "x2A" este memorat pe 4 caractere, ultimul fiind \0, numărul de elemente al şirului fiind, deci, 3, iar dimensiunea şirului 4.

Un caracter dintr-un şir (vector) de caractere "a" poate fi accesat folosind indexul şirului (a[i], de exemplu) sau folosind pointeri la caracter.

Iniţializarea (citirea) unui şir de caractere se poate face în mai multe moduri:

a. Iniţializarea fiecărui element cu câte un character (ca la orice vector):

a[0] = 'x';

a[1] = '2';

a[2] = 'A';

a[3] = '\0';

b. În C, folosind funcţia "***scanf( )***":

scanf("%s", a);

Formatul "%s" este folosit pentru citirea unui sir de caractere. Se efectuează trei paşi:

- poziţionarea pe primul caracter al şirului;

- citirea tuturor caracterelor diferite de <Enter> şi introducerea lor în vectorul "a";

- citirea se face până la întâlnirea EOF, după care se plasează '\0' la sfârşitul şirului.

Deoarece ***numele unui şir*** (ca la orice vector) ***este un pointer la adresa de bază a şirului,*** expresia "a" este echivalentă cu "&a[0]".

Dacă şirul citit are mai multe caractere decat cele rezervate, se va obţine o eroare.

c. În C++, folosind cin>> (dar citirea se opreşte la primul caracter blank)

d. Folosind macroul gets() – care se găseşte în headerul ***stdio.h***

Este important de reţinut că 'b' şi "b" sunt două lucruri diferite, prima fiind o constantă caracter, în timp ce a doua este o constantă şir de caractere, care conţine pe prima poziţie constanta caracter 'b' şi pe a doua poziţie caracterul '\0'

e. Şirurile se pot iniţializa la fel ca şi caracterele

char a[] = "x2A";

sau, echivalent

char a[] = {'x', '2', 'c', '\0'};

f. Putem folosi un pointer către un şir constant, dar interpretarea este diferită:

char \*p = "x2A";

Numele unui şir (ca la orice vector) poate fi tratat ca un pointer către adresa de bază a şirului din memorie.

O constantă - de exemplu, "x2A" - este memorată de către compilator. În acelaşi timp, aceasta este "un nume de şir". Diferenţa dintre un şir iniţializat cu o constantă şir şi un pointer iniţializat tot cu o constantă şir este că şirul conţine caractere individuale urmate de caracterul "\0", în timp ce pointerul este asignat cu adresa şirului constant din memorie.

**Constante şir de caractere**

***Constanta şir*** este o succesiune de zero sau mai multe caractere, încadrate de ghilimele.

În componenţa unui şir de caractere poate intra orice caracter, deci şi caracterele escape.

Lungimea unui şir este practic nelimitată.

Daca se doreşte continuarea unui şir pe randul următor, se foloseşte caracterul backslash.

Caracterele componente ale unui şir sunt memorate într-o zonă continuă de memorie (la adrese succesive).

Pentru fiecare caracter se memorează codul ASCII al acestuia.

După ultimul caracter al şirului, compilatorul plasează automat caracterul NULL (\0), caracter care reprezintă marcatorul sfârşitului de şir. Numărul de octeţi pe care este memorat un şir va fi, deci, mai mare cu 1 decât numărul de caractere din şir.

***Exemple:***

”Acesta este un sir de caractere” //constanta sir memorata pe 32 octeti

”Sir de caractere continuat\”pe randul urmator!”   //constantă şir memorată pe 45 octeţi

 ”Sir \t cu secvente escape\n”  //constantă şir memorată pe 26 octeţi

 ’\n’      //constantă caracter memorată pe un octet

 ”\n” //constantă şir memorată pe 2 octeţi (codul caracterului ***escape*** şi terminatorul de şir)

 ”a\a4”  /\*Sir memorat pe 4 octeti:

Pe  primul  octet: codul  ASCII al  caracterului a

Pe  al doilea  octet: codul  ASCII al  caracterului escape \a

Pe al treilea octet: codul ASCII al caracterului 4

Pe al  patrulea octet:  terminatorul de sir NULL, cod ASCII 0 \*/

 ”\\ASCII\\”  /\*Sir memorat pe 8 octeti:

Pe primul octet: codul  ASCII al  caracterului backslah

Pe  al doilea   octet:  codul   ASCII   al   caracterului A

Pe  al  treilea  octet:  codul  ASCII    al   caracterului S

Pe al   patrulea octet: codul ASCII al caracterului S

Pe al  6-lea  octet:  codul  ASCII  al  caracterului  I

Pe al  7-lea  octet:  codul  ASCII  al  caracterului  I

Pe  al  8-lea  octet:  codul   ASCII   al  caracterului backslah

Pe al 9-ea octet: terminatorul de sir NULL, de cod ASCII 0 \*/

 ”1\175a”  /\*Sir memorat pe 4 octeti:

Primul octet: Codul ASCII al caracterul 1

Al 2-lea  octet:  codul  ASCII 125 (175 in octal) al caracterului }

Al  3-lea  octet:  codul  ASCII  al caracterului a

Al 4-lea  octet: codul ASCII 0 pentru terminatorul sirului \*/

***Exercitiu:***

Să se scrie următorul program şi să se urmarească rezultatele execuţiei acestuia.

*#include <iostream.h>*

*#include <conio.h>*

*void main(){*

*clrscr();*

*cout<<"Sirul \"Ab9d\" este memorat pe:"<<sizeof("Ab9d")<<" octeti\n";*

*cout<<"Sirul \"Abcd\\t\" este memorat pe:"<<sizeof("Abcd\t")<<" octeti\n";*

*cout<<"Sirul \"\n\" este memorat pe "<<sizeof("\n")<<" octeti\n";*

*cout<<"Sirul \"\\n\" este memorat pe "<<sizeof("\n")<<" octeti\n";*

*cout<<"Sirul \"ABCDE\" se memoreaza pe "<<sizeof("ABCDE")<<" octeti\n";*

*getch();*

*}*

* + 1. **Funcţii pentru operaţii cu şiruri de caractere**

**strlen**

Calculeazǎ lungimea unui şir

size\_t strlen(const char \*s);

Returneazǎ numǎrul de caractere din ***s***, fǎrǎ caracterul null.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25];*

*cout<<"Introduceti sirul ";gets(sir);*

*cout<<"Sirul are lungimea "<<strlen(sir);*

*getch();*

*}*

**strcpy**

Copiazǎ şirul ***src*** în ***dest***

char \*strcpy(char \*dest, const char \*src);

Returneazǎ ***dest***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir1[25];*

*char sir2[25] = "Bacalureat";*

*strcpy(sir1, sir2);*

*cout<<sir1;*

*getch();*

*}*

**strncpy**

Copiazǎ cel mult maxlen caractere din src la

dest

char \*strncpy(char \*dest, const char \*src,

size\_t maxlen);

Returneazǎ dest.

**strcat**

Adaugǎ şirul ***src*** la sfârşitul şirului ***dest***

char \*strcat(char \*dest, const char \*src);

Returneazǎ ***dest***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <string.h>*

*#include <stdio.h>*

*void main(void)*

*{*

*clrscr();*

*char destinatie[25], blank[25] = " ", prenume[25] = "Dan", nume[25] = "Gogoncea";*

*strcpy(destinatie, prenume);*

*strcat(destinatie, blank);*

*strcat(destinatie, nume);*

*cout<<destinatie;*

*getch();*

*}*

**strncat**

Adaugǎ cel mult maxlen caractere din src la

dest

char \*strncat(char \*dest, const char \*src,

size\_t maxlen);

Returneazǎ dest.

**strcmp**

Comparǎ un şir cu altul

int strcmp(const char \*s1, const char \*s2);

Returneazǎ o valoare care este

< 0 dacǎ ***s1*** este mai mic decât ***s2***

== 0 dacǎ ***s1*** este acelaşi cu ***s2***

> 0 dacǎ ***s1*** este mai mare decât ***s2***

*#include <conio.h>*

*#include <iostream.h>*

*#include <string.h>*

*#include <stdio.h>*

*void main(void)*

*{*

*clrscr();*

*char sir1[25] = "aaaa", sir2[25] = "aab";*

*int rez;*

*rez = strcmp(sir2, sir1);*

*if (rez > 0) cout<<"sirul sir2 este mai mare decat sirul sir1"<<endl;*

*else cout<<"sirul sir2 este mai mic decat sirul sir1"<<endl;*

*getch();*

*}*

**stricmp**

Comparǎ un şir cu altul, ignorând case

int stricmp(const char \*s1, const char \*s2);

**strncmp**

Comparǎ cel mult ***maxlen*** caractere dintr-un şir cu cele din alt şir

int strncmp(const char \*s1, const char \*s2,

size\_t maxlen);

**strncmpi**

Comparǎ o porţiune dintr-un şir cu o porţiune dintr-un alt şir, fǎrǎ case sensitiv.

int strncmpi(const char \*s1,

const char \*s2, size\_tn);

**strchr**

Cautǎ caracterul ***c*** în şirul ***str***

char \*strchr(const char \*s, int c);

Returneazǎ un pointer la prima apariţie a caracterului ***c*** în şirul ***s***; dacǎ does nu apare în şirul s, strchr returneazǎ ***NULL***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <string.h>*

*#include <stdio.h>*

*void main(void)*

*{*

*clrscr();*

*char s[15], \*ptr, c;*

*cout<<"Introduceti sirul ";gets(s);*

*cout<<"Introduceti caracterul ";cin>>c;*

*ptr = strchr(s, c);*

*if (ptr)*

*cout<<"Caracterul "<<c<<" este pe pozitia "<<ptr-s;*

*else*

*cout<<"Caracterul "<<c<<" nu se gaseste in sir";*

*cout<<endl<<ptr ;*

*getch();*

*}*

**strrchr**

Gǎseşte ultima apariţie a caracterului c în s

char \*strrchr(const char \*s, int c);

Returneazǎ un pointer la ultima apariţie a caracterului c, sau NULL, dacǎ c nu apare în s

**strstr**

Gǎseşte prima apariţie a unui subşir în alt şir

char \*strstr(const char \*s1,

const char \*s2);

Returneazǎ un pointer la elemental din ***s1*** care conţine pe ***s2*** (pointeazǎ cǎtre ***s2*** în ***s1***), sau ***NULL*** dacǎ ***s2*** nu apare în ***s1***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir1[25] = "Bunica bate toba", sir2[25] = "at", \*ptr;*

*ptr = strstr(sir1, sir2);*

*cout<<"Am gasit subsirul "<<sir2<<" la "<<ptr<<endl;*

*cout<<"adica pe pozitia "<<ptr-sir1;*

*getch();*

*}*

**strspn**

Scaneazǎ un şir pentru gǎsirea unui segment care este o submulţime a unei mulţimi de caractere

size\_t strspn(const char \*s1,

const char \*s2);

Returneazǎ lungimea segmentului din s1 care constǎ numai din caractere din s2.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char s1[25] = "1234567890";*

*char s2[25] = "123DC8";*

*int poz;*

*poz = strspn(s1, s2);*

*cout<<"Sirurile difera la pozitia "<<poz<<" care este "<<s2[poz];*

*getch();*

*}*

**strcspn**

Scaneazǎ un şir

size\_t strcspn(const char \*s1,

const char \*s2);

Returneazǎ lungimea segmentului din ***s1*** care constǎ în întregime doar din caractere care ***NU*** se gǎsesc în ***s2***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char s1[25] = "Bunica bate toba";*

*char s2[25] = "Bunica bate tare";*

*int poz;*

*poz = strspn(s1, s2);*

*cout<<"Sirurile difera in pozitia "<<poz<<endl;*

*cout<<"(primul caracter din s2 care nu se gaseste in s1 este pe pozitia "<<poz<<endl;*

*cout<<"si este caracterul \'"<<s1[poz]<<"\' in sirul s1 si caracterul \'"<<s2[poz]<<"\' in sirul s2)";*

*getch();*

*}*

**strset**

Seteazǎ toate caracterele din s la ch

char \*strset(char \*s, int ch);

Returneazǎ un pointer la s.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25] = "Bunica bate toba";*

*char caracterul = '\*';*

*strset(sir, caracterul);*

*cout<<sir;*

*getch();*

*}*

**strnset**

Seteazǎ primele ***n*** caractere ale ***s*** lui la ***ch***

char \*strnset(int \*s, int ch, size\_t n);

Returneazǎ un pointer la ***s***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25] = "Bunica bate toba";*

*char caracterul = '\*';*

*strnset(sir, caracterul, 3);*

*cout<<sir;*

*getch();*

*}*

**strrev**

Inverseazǎ toate caracterele din ***s***

char \*strrev(char \*s);

Returneazǎ un pointer la şirul inversat.

*#include <conio.h>*

*#include <iostream.h>*

*#include <string.h>*

*#include <stdio.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25];*

*cout<<"Introduceti sirul ";gets(sir);*

*strrev(sir);*

*cout<<sir;*

*getch();*

*}*

**strlwr**

Converteşte totul la lowercase

char \*strlwr(char \*s);

Returneazǎ un pointer la s.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25] = "Cucu Bau";*

*strlwr(sir);*

*cout<<sir;*

*getch();*

*}*

**strupr**

Converteşte toate caracterele din ***s*** la uppercase

char \*strupr(char \*s);

Returneazǎ un pointer la ***s***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <stdio.h>*

*#include <string.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[25] = "Dan Gogoncea";*

*strlwr(sir);*

*cout<<sir;*

*getch();*

*}*

**strtok**

Scaneazǎ şirul ***s1*** pentru a găsi prima apariţie a unui caracter din şirul **s2**

char \*strtok(char \*s1, const char \*s2);

Şirul ***s2*** defineşte, de obicei, caractere separatoare.

Funcţia ***strtok*** interpreteazǎ şirul ***s1*** ca o serie de cuvinte separate de caractere din şirul ***s2***.

Dacă în şirul ***s1*** nu este găsit nici un caracter din şirul ***s2***, funcţia returnează ***NULL***.

Dacă este găsit un character, pe poziţia următoare din s1este pus un character ***NULL*** şi funcţia returnează un pointer.

Următoarele chemări ale funcţiei ***strtok*** se vor efectua cu valoarea NULL ca prim parametru al funcţiei ***strtok***.

*#include <conio.h>*

*#include <iostream.h>*

*#include <string.h>*

*#include <stdio.h>*

*void main(void)*

*{*

*clrscr();*

*char sir[255];*

*char separatori[20]=",.;:?!<>(){}[]' -";*

*char \*ptr;*

*// cout<<"Introduceti sirul : ";gets(sir);*

*strcpy(sir,"Vii?Nu!N-o sa vin,nici eu, niciea;e bine?");*

*ptr = strtok(sir, separatori);*

*if (ptr) cout<<ptr<<endl;*

*while(ptr){*

*ptr = strtok(NULL, separatori);*

*if (ptr) cout<<ptr<<endl;*

*}*

*getch();*

*}*

P. Funcţia ***atof***

double atof(sir); – converteşte un şir către tipul double. Dacă această conversie eşuează (se întâlneşte un caracter nenumeric), valoarea întoarsă este 0. Această funcţie (ca şi cele similare) necesită includerea bibliotecii ***stdlib.h***.

Q. Funcţia ***\_atold***

long double \_atold(sir); – converteşte un şir către tipul long double. Dacă această conversie eşuează, valoarea întoarsa este 0.

P. Funcţia ***atoi***

int atoi(sir); – converteşte un şir către tipul int. Dacă această conversie eşuează (se întâlneşte un caracter nenumeric), valoarea întoarsă este 0.

R. Funcţia ***atoll***

long atol(sir); – converteşte un şir către tipul long. Dacă această conversie eşuează, valoarea întoarsa este 0.

S. Functia ***itoa***

itoa(int valoare,sir,int baza); – converteşte o valoare de tip int în şir, care este memorat în variabila ***sir***. Baza reţine baza de numeraţie către care să se facă conversia. În cazul bazei 10, şirul reţine şi eventualul semn -.

T. Funcţia ***ltoa***

ltoa(long valoare,sir,int baza); – converteşte o valoare de tip long int în şir, care este memorat în variabila ***sir***.

U. Funcţia ***ultoa***

ultoa(unsigned long valoare,sir,int baza); – converteşte o valoare de tip unsigned long în şir, care este memorat în variabila ***sir***.

* 1. **EXEMPLE**

1. ***Afişarea tuturor poziţiilor unui caracter într-un şir***

*#include <iostream.h>*

*#include <string.h>*

*#include <conio.h>*

*void main( ){*

*clrscr();*

*char a[100],\*p,c;*

*cout<<”Introduceti sirul: ”; cin.get(a,100);*

*cout”Introduceti caracterul: ”; cin>>c;*

*p=strchr(a,c);*

*while (p){*

*cout<<"Pozitia "<<p-a<<endl;*

*p++;*

*p=strchr(p,c);*

*}*

*getch();*

*}*

1. ***Separarea cuvintelor dintr-un text***

*#include <iostream.h>*

*#include <conio.h>*

*#include <string.h>*

*void main(){*

*char text[100],cuv[10][10],\*p,\*r,separator[]=",. !?";int i=0,nr=0;*

*clrscr();*

*cout<<"Dati sirul:"; cin.get(text,100);*

*strcpy(p,text);*

*p=strtok(p,separator);*

*while (p)*

*{strcpy(cuv[++nr],p);*

*p=strtok(NULL,separator);}*

*cout<<"Sunt "<<nr<<" cuvinte:"<<endl;*

*for (i=1;i<=nr;i++) cout<<cuv[i]<<endl;*

*getch();*

*}*

1. ***Este şirul este alcătuit exclusiv din caractere numerice?***

*#include <iostream.h>*

*#include <conio.h>*

*#include <string.h>*

*void main(){*

*char text[100],cifre[]="0123456789";*

*clrscr();*

*cout<<"Dati sirul:"; cin.get(text,100);*

*if (strcspn(cifre,text)==strlen(text))*

*cout<<"exclusiv numeric";*

*else cout<<”nenumeric”;*

*getch();}*

* 1. **TEME DE LABORATOR**

1. Să se despartă un text în cuvinte cu ajutorul unui şir determinat de separatori şi să se afişeze cuvintele separate. Să se afişeze cuvântul de lungime maximă.

2. Pentru un text ce conţine cuvinte separate de câte un spaţiu, să se afişeze cuvintele care au cel puţin 3 consoane sau 3 vocale consecutive.

3. Se citeşte un şir de caractere. Să se afişeze şirul oglindit, din care lipsesc vocalele.

4. Să se afişeze vocalele unui cuvânt.

5. Să se afişeze vocalele dintr-un text, în ordinea în care apar ele, apoi, pe rândul următor, numărul de apariţii pentru fiecare din cele 5 vocale posibile, în ordinea lor alfabetică

6. Să se verifice dacă un text introdus de la tastatură este palindrom.

7. Pentru un text introdus de la tastatură, să se afişeze pe monitor textul în care toate vocalele sunt dublate, iar literele mari sunt înlocuite prin caracterul “\*”

8. Pentru un text introdus de la tastatură, să se înlocuiască literele mici cu litere mari si invers.

BONUS

9. Pentru un text introdus de la tastatură, să se afişeze pe monitor ultima consoană care apare în text

10. Se citeşte un text. Textul conţine cuvinte separate prin unul sau mai multe spaţii. Să se afişeze pe monitor textul în care cuvintele îşi păstrează poziţia iniţială, dar fiecare dintre ele este inversat