**LUCRAREA DE LABORATOR nr. 6**

**Prelucrarea caracterelor şi şirurilor caracteriale în C în baza propriilor subprograme**

*Obiectivele temei*

1. Aprofundarea cunoştinţelor în limbajul şi mediul TurboC şi perfecţionarea tehnicii de programare în prelucrarea caracterelor și şirurilor de caractere în C cu propriile subprograme.
2. Însuşirea procedeelor referitoare la prelucrarea caracterelor și şirurilor caracteriale numai cu funcțiile predefinite, incluse în indicații.
3. Analiza specificului de algoritmizare şi progamare a fragmentelor de texte prin comparaţia soluţiilor stereotipe şi celor eficiente de introducere, afişare şi diverse manipulări asupra textelor: parcurgeri, căutări, schimbări şi rearanjări etc.

*Subiectele temei şi ordinea executării*

1. Studierea principiilor prelucrării (descrierii, declarării, formării, etc.) şirurilor caracteriale în C.
2. Studierea metodelor şi tehnicilor de bază de prelucrare a textelor ca şiruri caracteriale, analizând exemplele din indicaţii şi de reflectat în raport cu comentariile de rigoare.
3. Elaborarea algoritmului şi programului de soluţionare a variantei respective (Anexa).
4. Depanarea programului şi verificarea corectitudine cu ajutorul testelor elaborate.

*Conţinutul raportului (vezi lucr. de laborator nr.1-6)*

Suplimentar: Evidenţierea specificului prelucrării şirurilor caracteriale în C şi analiza erorilor admise pe parcursul efectuării lucrării şi eficienţa algoritmilor elaboraţi.

***I. Noţiuni generale***

***1.1. Tipul char (tipul caracter)***

O variabilă de tip caracter se declară prin specificatorul de tip char. Zona de memorie alocată unei variabile de tip char este de un octet. Ea este suficient de mare pentru a putea memora orice caracter al setului de caractere implementate pe calculator.

Dacă un caracter din setul de caractere este memorat într-o variabilă de tip char, atunci valoarea sa este egală cu codul întreg al caracterului respectiv. Şi alte cantităţi pot fi memorate în variabile de tip char, dar implementarea este dependentă de sistemul de calcul.

Ordinul de mărime al variabilelor caracter este între -128 şi 127. Caracterele setului ASCII sunt toate pozitive, dar o constantă caracter specificată printr-o secvenţă de evitare poate fi şi negativă, de exemplu ‚\377’ are valoarea -1. Acest lucru se întâmplă atunci când această constantă apare într-o expresie, moment în care se converteşte la tipul int prin extensia bitului cel mai din stânga din octet (datorită modului de funcţionare a instrucţiunilor calculatorului).

O valoare de tip CHAR este un element al mulţimii finite şi ordonate de caractere recunoscute de translatorul limbajului. Datele de tip caracter sunt stocate în memorie pe un octet, putînd avea 256 de valori distincte. Aceste valori sunt în mod obişnuit caracterele setului de caractere ASCII extins (setul de caractere al limbajului TurboPascal fiind un subset al acestuia din urmă).

Constantele literale de tip caracter pot fi scrise în mai multe moduri:

- un caracter imprirnabil intre apostrofuri: '\*', '3', 'a';

- un nurnăr de la 0 la 255 precedat de caracterul #:

#27 (codul caracterului ESC), #65 (codul caracterului A); etc.

**1.2. *Tipul şir de caractere***

Limbajul C nu are incorporate facilităţi de prelucrare directă a şirurilor de caractere, mulţimilor, listelor, tablourilor. Din acest motiv, nu exista definit un tip de date pentru tipurile de caractere, folosindu-se pentru şiruri tablouri de caractere (**char[]**) sau pointeri la caracter (**char \***). Pentru aceste operaţii, cea mai mare parte a implementărilor în C oferă colecţii standard de funcţii de bibliotecă. Aceste funcţii oferă o varietate de opţiuni.

În plus, fiecare programator îşi poate construi propria sa biblioteca de funcţii care să înlocuiască sau să extindă colecţia de funcţii standard ale limbajului.

O valoare de tip **char** este un element al mulţimii finite şi ordonate de caractere recunoscute de translatorul limbajului. Datele de tip caracter sunt stocate în memorie pe un octet, putând avea 256 de valori distincte. Aceste valori sunt în mod obişnuit caracterele setului de caractere ASCII extins (setul de caractere al limbajului C, fiind un subset al acestuia din urmă).

Un **şir de caractere** este o succesiune de zero sau mai multe caractere ale setului de caractere ASCII extins, scrise pe o singură linie în program şi încadrate între apostrofuri. şirul fără nici un caracter între apostrofuri se numeşte şir nul -- vid. Două apostrofuri succesive intr-un string desemnează un singur apostrof. Relaţia de ordine dintre două şiruri de caractere rezultă din relaţia de ordine dintre valorile caracterelor din poziţiile corespunzatoare.

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

Marcatorul "sfârşit de şir de caractere" **\0.** Prin convenţie, un şir de caractere se termină prin marcatorul (santinela, delimitator) **\0**, sau caracterul nul. De exemplu, şirul "abc" este memorat pe 4 caractere, ultimul fiind \0. Deci numărul de elemente al şirului este 3, iar dimensiunea 4. Constanta "abc" este memorată de către compilator şi în acelaşi timp, aceasta este "un nume de şir".

O constantă caracter constă dintr-un singur caracter scris între apostrofuri, de exemplu ‚x’. Valoarea unei constante caracter este valoarea numerică a caracterului, în setul de caractere al calculatorului. De exemplu în setul de caractere ASCII caracterul zero sau ‚0’ are valoarea 48 în zecimal, total diferită de valoarea numerică zero.

a) **Constantele caracter** participă la operaţiile aritmetice ca şi oricare alte numere. De exemplu, dacă variabila c conţine valoarea ASCII a unei cifre, atunci prin instrucţiunea:

c = c – ‚0’ ; această valoare se transformă în valoarea efectivă a cifrei.

Deci un caracter poate apărea oriunde unde un întreg este admis. În toate cazurile valoarea caracterului este convertită automat într-un întreg. Deci într-o expresie aritmetică tipul char şi int pot apărea împreună. Aceasta permite o flexibilitate considerabilă în anumite tipuri de transformări de caractere. Un astfel de exemplu este funcţia **atoi** descrisă în Help-ul C care converteşte un şir de cifre în echivalentul lor numeric.

Atragem atenţia că atunci când o variabilă de tip char este convertită la tipul int, se poate produce un întreg negativ, dacă bitul cel mai din stânga al octetului conţine 1. Caracterele din setul de caractere ASCII nu devin niciodată negative, dar anumite configuraţii de biţi memorate în variabile de tip caracter pot apărea ca negative prin extensia la tipul int. Ordinul de mărime al variabilelor caracter este între -128 şi 127. Caracterele setului ASCII sunt toate pozitive, dar o constantă caracter specificată printr-o secvenţă de evitare poate fi şi negativă, de exemplu '\377' are valoarea -1. Acest lucru se întâmplă atunci când această constantă apare într-o expresie, moment în care se converteşte la tipul int prin extensia bitului cel mai din stânga din octet (datorită modului de funcţionare a instrucţiunilor calculatorului). Conversia tipului int în char se face cu pierderea biţilor de ordin superior.

Întregii de tip short sînt convertiţi automat la int. Conversia întregilor se face cu extensie de semn; întregii sînt totdeauna cantităţi cu semn.

Un întreg long este convertit la un întreg short sau char prin trunchiere la stânga; surplusul de biţi de ordin superior se pierde.

Anumite caractere negrafice şi caractere grafice ‘ (apostrof) şi \ (backslash) pot fi reprezentate ca şi constante caracter cu ajutorul aşa numitor secvenţe de evitare. Secvenţele de evitare oferă de altfel şi un mecanism general pentru reprezentarea caracterelor mai greu de introdus în calculator şi a oricăror configuraţii de biţi. Aceste secvenţe de evitare sunt:

\n new-line \r carriage return \\ backslash

\t tab orizontal \f form feed \’ apostrof

\b backspace \a semnal sonor \” ghilimele

\ddd configuraţie de biţi (*ddd*)

Aceste secvenţe, deşi sunt formate din mai multe caractere, ele reprezintă în realitate un singur caracter. Secvenţa ‚\ddd’ unde ddd este un şir de 1 până la 3 cifre octale, generează pe un octet valoarea caracterului dorit sau a configuraţiei de biţi dorite, date de şirul *ddd*.

*Exemplu*: secvenţa ‚\040’ va genera caracterul spaţiu.

Un caz special al acestei construcţii este secvenţa ‚\0’ care indică caracterul NULL, care este caracterul cu valoarea zero. ‚\0’ este scris deseori în locul lui 0 pentru a sublinia natura de caracter a unei anumite expresii.

Când caracterul care urmează după un backslash nu este unul dintre cele specificate, backslash-ul este ignorat. Atragem atenţia că toate caracterele setului ASCII sunt pozitive, dar o constantă caracter specificată printr-o secvenţă de evitare poate fi şi negativă, de exemplu ‚\377’ are valoarea -1.

b) **Constante simbolice.**  O constantă simbolică este un identificator cu valoare de constantă. Valoarea constantei poate fi orice şir de caractere introdus prin construcţia #define.

*Exemplu*: #define MAX 1000

După întîlnirea acestei construcţii compilatorul va înlocui toate apariţiile constantei simbolice MAX cu valoarea 1000.

Numele constantelor simbolice se scriu de obicei cu litere mari (fără a fi obligatoriu).

Deci un şir este o succesiune de caractere scrise între ghilimele, de exemplu „ABCD”.

*Ghilimelele* nu fac parte din şir; ele servesc numai pentru delimitarea şirului. Caracterul **“** (ghilimele) poate apărea într-un şir dacă se utilizează secvenţa de evitare \”. În interiorul unui şir pot fi folosite şi alte secvenţe de evitare pentru constante caracter, de asemenea poate fi folosit caracterul \ (backslash) la sfârşitul unui rând pentru a da posibilitatea continuării unui şir pe mai multe linii, situaţie în care caracterul \ însuşi va fi ignorat.

Pentru şirul de caractere se mai foloseşte denumirea constantă şir sau constantă de tip şir.

Când un şir apare într-un program C, compilatorul creează un masiv de caractere care conţine caracterele şirului şi plasează automat caracterul NULL (‚\0’) la sfârşitul şirului, astfel ca programele care operează asupra şirurilor să poată detecta sfârşitul acestora. Această reprezentare înseamnă că, teoretic, nu există o limită a lungimii unui şir, iar programele trebuie să parcurgă şirul, analizându-l pentru a-i determina lungimea. Se admit şi şiruri de lungime zero.

Tehnic, un şir este un masiv ale cărui elemente sunt caractere. El are tipul masiv de caractere şi clasa de memorie static. Un şir este iniţializat cu caracterele date.

La alocare, memoria fizică cerută este cu un octet mai mare decât numărul de caractere scrise între ghilimele, datorită adăugării automate a caracterului null la sfârşitul fiecărui şir.

**Constantele literale de tip caracter** pot fi scrise în mai multe moduri:

* un caracter imprimabil între apostrofuri: '\*', '3', 'a';
* un număr de la 0 la 255 precedat de caracterul #:
* #27 (codul caracterului ESC), #65 (codul caracterului A); etc.

Constantele de tip **char** sunt descrise potrivit urmatoarei sintaxe:

**# define nume constanta *constanta\_caracter*;**

**char** nume\_ variabila = *constanta caracter*;

unde constanta\_caracter este o constantă literală de tip caracter, iar nume\_variabila reprezintă identificatorul unei variabile iniţializate (constantă cu tip).

Limbajul C nu defineste, într-un mod evident ca string in Pascal, tipul de data *şir*, dar exista *doua* posibilitati de definire a şirurilor:

Un **şir de caractere** este un tablou de caractere, al cărui ultim element este caracterul NULL (“”).

Sintaxa declarării variabilelor de tip caracter este următoarea:

**tip\_de\_date nume\_şir[dimensiune];**

unde:

* -tip\_de\_date: este tipul datei (obligatoriu char) ;
* -nume şir: este numele pe care îl dăm şirului ;
* -dimensiune: reprezintă numărul de caractere pe care îl conţine şirul .

Exemple:

* ca tablou de caractere; *exemple:*

#define MAXCUVINT 100

*char name [15];* char w[MAXCUVINT]; char sir1[30]; char sir2[10]="exemplu";

* ca pointer la caractere; *exemple*:

**char \*sir3;** *// şir3 trebuie initializat cu adresa unui  şir sau a unui spatiu alocat pe heap*

sir3=sir1; *// sir3 ia adresa unui sir static*

*// sir3=&sir1; sir3=&sir1[0]; sunt echivalente cu instr de atribuire de mai sus*

**sir3=(char \*)malloc(100);** *// se aloca dinamic un spatiu pe heap.*

**char \*sir4="test";** *// sir4 este initializat cu adresa sirului constant*

**char s[10];** vom avea tabloul **s**, cu elemente de tip char, având 10 elemente.

În C nu exista instructiuni dedicate pentru citire / scriere, ci functii care se preiau din biblioteca limbajului. Pentru a utiliza aceste functii trebuie incluse fisierele <stdio.h> (StanDard Input/Output Header) sau <conio.h> (CONsole Input/Output);

* + **Functii predefinite de prelucrare a caracterelor și** **şirurilor**, (declarate in stdio.h )
* **char \* gets(char \* s);** citeste caracterele din intrare pina la intalnirea caracterului Enter, care nu se adauga la şirul s; plaseaza '\0' la sfarsitul lui s; returneaza adresa primului caracter din şir; daca se tasteaza CTRL/Z returneaza NULL; codul lui Enter e *scos* din buffer-ul de intrare
* **int puts(char \* s);** tipareste şirul s, trece apoi la rand nou
* **scanf("%s",s);** citeste caracterele din intrare pina la intalnirea primului blanc sau Enter, care nu se adauga la şirul s; plaseaza '\0' la sfarsitul lui s; daca se tasteaza CTRL/Z returneaza EOF; codul lui blanc sau Enter *raman* in buffer-ul de intrare

**getch( )** Citeste de la tastatura codul ASCII al caracterului tastat. Daca tastei nu-i corespunde un caracter ASCII getch( ) va trebui apelata de doua ori.

**getche( )** Citeste de la tastatura codul ASCII al caracterului si-l afiseaza pe ecran.

**putch( )**  - afiseaza un caracter pe ecran.

// program care afiseaza un caracter ce se citeste de la tastatura

#include<conio.h> /\* pentru ca se utilizeaza functii de citire \*/

void main(void)

**putch(getch());** /\* afiseaza ceea ce se citeste \*/

**putch('in');** /\* afiseaza un rand nou \*/

**getch();** /\* asteapta pana la apasarea unei taste \*/

**getchar( )** Citeste codul ASCII al caracterului introdus de la tastatura. Citirea nu se face direct de la tastatura, ci dintr-o zona de memorie atasata tastaturii de catre sistemul de operare. In aceasta zona se memoreaza toate caracterele tastate pana la <Enter> si apoi se pot prelua cu getchar. Citirea caracterelor prin intermediul zonei de memorie tampon a tastaturii permite corectarea erorilor de tastare (inainte de a tasta <Enter>).

**putchar (expresie)** -afiseaza un caracter corespunzator codului ASCII transmis.

/\* Program care citeste doua caractere si apoi asteapta o tasta \*/

#include <stdio.h>

#include <conio.h>

void main(void)

{**putchar(getchar());** /\* afiseaza caracterul citit \*/

putchar(getchar());

putchar('in');

getch(); /\* asteapta o tasta \*/ }

/\* Program care afiseaza tot ce se tasteaza pana la apasarea lui Ctrl z \*/

#include<stdio.h>

void main(void)

{int c; cagetchar();

while (c!aEOF){ /\* câtă vreme c nu este Ctrl z (EOF, End Of File): \*/

putchar(c); /\* afiseaza \*/

cagetchar(); /\* citeste \*/ }

sau o alta varianta a programului de mai sus:

#include<stdio.h>

void main(void)

{int c;

while((cagetchar()!aEOF) putchar(c); }

**gets** **( )** Citirea cu ecou a unui sir de caractere. Parametrul functiei este adresa de inceput a sirului, declarat ca un tablou de caractere

char numes20t;

gets(nume);

**puts ( )** Afiseaza un sir a carui adresa de inceput este specificata ca parametru. Sirul se termina cu caracterul NULL, care nu se afiseaza

char prenume[]="Maria";

puts(prenume);

/\* *program care citeste numele si prenumele, iar apoi le afiseaza in ordine inversa*

#include<stdio.h>

#include<conio.h>

void main(void)

{ char nume[20], prenume[15];

puts("in Introduceti numele: "); gets(nume);

puts("in Introduceti prenumele: "); gets(prenume);

puts("in"); puts(prenume); putchar(' ');

putchar(nume[0]); putchar ('.'); getch(); }

Modurile de afişare cu functia **printf ( )** Functia permite afisarea datelor sub controlul unui format de afisare. Ea returneaza numarul de caractere afisate sau -1 in caz de eroare. Sintaxa functiei este:

printf(sir\_control, param\_1,param\_2,....param\_n); unde sir\_control este un sir de caractere, iar param\_1, param\_2, ...., param\_n sunt datele de afisat.

Sirul de control defineste:

- caracterele de afisat propriu-zise

- specificatorii de format sf1, sf2,... sfn, care precizeaza modul de afisare al parametrilor param\_1, param\_2, ...., param\_n. Fiecare specificator de format incepe cu caracterul % si se termina cu una sau doua litere.

Alinierea datelor de afisat se face la stanga campului definit de specificator (implicit datele sunt aliniate la dreapta). Litera defineste tipul de conversie aplicat parametrului care se afiseaza. Intre % si litera pot sa apara in mod optional:

- un sir de cifre zecimale - arata dimensiunea campului pe care se va face afisarea

- punct (.) urmat de un sir de cifre zecimale - indica numarul de zecimale care se vor afisa pentru un numar real

- o combinatie a celor doua situatii anterioare.

Tabelul 7.1. Specificatorii de format ai functiei printf( )

|  |  |
| --- | --- |
| cod | semnificaTie |
| %c | Caracter |
| %d | Intreg cu semn |
| %i | Intreg cu semn |
| %e | Afisare numere reale in format cu exponent (e) |
| %E | Afisare numere reale in format cu exponent (E) |
| %f | Afisare numere reale simpla precizie cu virgula |
| %g | Foloseste cel mai scurt format dintre %e si %f |
| %G | Foloseste cel mai scurt format dintre %E si %f |
| %o | Afisarea unui numar in octal |
| %s | Afisarea unui sir de caractere |
| %u | Un intreg fara semn |
| %x | Hexazecimal fara semn (folosind litere mici) |
| %X | Hexazecimal fara semn (folosind litere mari) |
| %p | Afisarea unui pointer |
| %n | Pointer intreg spre numarul de caractere afisate anterior |
| %% | Afisarea semnului % |

Literele pot fi:

**c** - indica afisarea unui caracter a carui cod ASCII este dat de valoarea parametrului

char c;

c='c'; printf("caracterul este: %c",c); putchar(getchar()); *// printf("%c",getchar());*

putchar(getchar()); putchar('in'); // *printf("%c in",getchar());*

printf("\*%5c\*",'a'); //Efect: \* a\*

printf("\*%-5c\*",'a'); //Efect: \*a \*

**s** - indica afisarea unui sir de caractere a carui adresa de inceput este data ca parametru

printf("123abc"); // printf("%s","123abc");

**d** - indica afisarea datelor de tip intreg, ca valori zecimale (%d este inlocuit de valoarea parametrului corespunzator)

printf("\*%6d\*",i05 );

printf("Am afisat %d",i3 );

printf("Afisam %d din nou",i3 );

int ia5, ja3;

printf("in%d + %d a %d in ",i,j,i+j ); /\* 5 + 3 a 8 \*/

**o** - afiseaza parametrii in octal

int ka10;

printf("in zecimal ka%d it in octal ka%o",k,k); /\* in zecimal ka10 in octal ka11 \*/

**x** - afiseaza parametrii in hexazecimal

**u** - afisearea in zecimal a intregilor fara semn

**l,L** - se utilizeaza inaintea literelor specificatoare: d,o,x,u pentru a preciza ca datele sunt de tip long.

**f** - afisarea datelor de tip flotant sau double in forma: parte\_intreaga.parte \_fractionara

|  |  |  |
| --- | --- | --- |
| Valoare | Specificator de format | Ce se va afisa |
| 523.625  523.625  523.625 | %5f  %6.1f  %4.0f | 523.62500  523.6  524 |

**e,E** - permit scrierea cu exponent a numerelor reale:

523.625 %e 5.23625 e-2

523.625 %.1e 5.2 E-2

**g,G** - permit afisarea numerelor reale cu sau fara exponent pe o lungime minima de caractere

g) **scanf** **( )** Functia permite citirea datelor de la tastatura sub controlul unui format, asemanator cu cel de la printf. Citirea datelor se face din zona tampon a tastaturii, deci citirea e efectiva doar dupa ce s-a tastat <Enter>. Sintaxa functiei este:

**scanf(sir\_control, param\_1, param\_2,..., param\_n);**

Sirul de control contine specificatorii de format, iar *param\_k* sunt adresele zonelor de memorie in care vor fi pastrate datele. Adresa e precizata prin caracterul &. Functia returneaza numarul campurilor citite corect.

1)

char c;

scanf("%c",&c); /\* citirea unui caracter in variabila c \*/

printf("%c",c); /\* afisarea caracterului citit \*/

2)

char nume[20],prenume[20];

scanf("%s %s",nume,prenume); /\*citeste doua siruri : nume si prenume \*/

printf("in%s %c in", nume, prenumes0t);

/\* s-a scris numele complet si prima litera din prenume (initiala) \*/

3)

int l,L;

printf("in Introduceti latimea "); scanf("%d", &l); /\* se citeste un intreg \*/

printf("in Introduceti lungimea: "); scanf("%d", &L); /\* se citeste un intreg \*/

printf("in Aria este: %10d", l\*L);

4)

int i;

printf("in Introduceti un intreg: "); scanf("%d", &i);

printf("in Numarul:%din în octal :%din în hexazecimal: %d ",i,i,i);

Citirea unui şir de caractere de la tastatură: pentru a citi un şir de caractere, rapid, de la tastatură avem nevoie de funcţia **gets()** care se găseşte în librăria <stdio.h> . Exemplu:

* printf(“Dati şirul : ” );
* **gets(s);**

Exemple: *//ex.1*

*void main(void) {*

*int i;* ***char name [15];***

*printf ("Culege numele:");* ***gets(name);***

*printf ("Numele dumnevoastra este:");*

*for (i=0; i<15; i++) printf ("%****c****",* ***name[i]****); }*

Iniţializarea unui şir se poate face in mai multe moduri:

1. Iniţializarea fiecărui element cu cate un caracter:

w[0] = 'A';

w[1] = 'B';

w[2] = 'C';

w[3] = '\0';

2. Citirea, folosind funcţia "scanf()":

**scanf("%s", w);**

Formatul "%s" este folosit pentru citirea unui şir de caractere. Distingem trei paşi:

1. poziţionare pe primul caracter al şirului;
2. se citesc toate caracterele diferite de si se introduc in "w";
3. citirea se face pana când întâlnim EOF; acum se plasează la sfârşitul şirului '\0'.

Din moment ce numele unui şir este un pointer la adresa de baza a şirului, expresia "w" este echivalenta cu "&w[0]".

Daca şirul citit are mai multe caractere decât cele rezervate, atunci se va obţine o eroare.

Atenţie! **'a'** si **"a"** sunt diferite. Prima este o constanta caracter, iar a doua este o constanta şir de caractere.

Deci **"a" = | 'a' | '\0' |**

3. Şirurile se pot initializa la fel ca si caracterele:

**char s[] = "abc";**

sau echivalent

**char s[] = {'a', 'b', 'c', '\0'};**

4. Putem folosi si un pointer către un şir constant, dar interpretarea este diferita:

**char \*p = "abc";**

Va reamintim ca numele unui şir poate fi tratat ca un pointer către adresa de baza a şirului din memorie. Aşadar, diferenţa dintre un şir initializat cu o constanta şir si un pointer iniţializat tot cu o constanta şir este ca şirul conţine caractere individuale urmate de caracterul **"\0"**, in timp ce pointerul este asignat cu adresa şirului constant din memorie.

Exemplu: Utilizarea şirurilor de caractere (ca vectori). Citim o linie de caractere dintr-un şir, le afişăm in ordine inversa si adunam literele din şir.

#include<stdio.h>

#include<conio.h>

#include<string.h>

#define MAXSTRING 100

main()

{ char c, name[MAXSTRING];

int i, sum = 0;

printf("\nSalut! Care este numele tau? ");

for (i = 0; (c = getchar()) != '\n'; ++i)

{ name[i] = c;

if (isalpha(c)) sum += c;

}

name[i] = '\0';

printf("\n%s%s%s\n%s", "Ma bucur ca te-am intalnit ", name, ".", "Numele tau scris invers este ");

for (--i; i >= 0; --i) putchar(name[i]);

printf("\n%s%d%s\n\n%s\n", "si numele tau are ", sum," litere .", "La revedere. ");

}

**Funcţiile predefinite pentru operaţii cu şiruri se găsesc în header-ul <string.h>**.

**strlen** **(**nume\_sir**);** Returnează un numar intreg ce reprezinta lungimea unui şir de caractere, fara a numara terminatorul de şir. Adică, prototipul funcţiei, fiind: **unsigned strlen(const char \*s);** păstrează numărul de caractere inaintea lui '\0'.

*1.3* E X E M P L E de P R O G R A M-E M O D E L în C

*1. Condiţia problemei:* Din propoziţia dată să se elimene cuvintele situate pe poziţii impare şi să se inverseze cuvintele de pe poziţiile pare. Să se afişeze şirurile obţinute şi numărul de operaţii efectuate.

*Schema logică:*

**Start**

**t=1,i=0,k=0,q=0,con=0**

**n**

**i=0**

nu

**s[j]==' '**

da

**con++**

**j++**

da

**j<strlen(s)**

nu

**s[i]!='\0'&&c!='\0'**

nu

da

nu

**t%2=0**

da

**s[i]!=' '&&s[i]!='\0'**

nu

da

**s1[k]=s[i];k++;i++**

**A**

**B**

**C**

**D**

**A**

**C**

**D**

**B**

**k++**

**h=k**

**sr[q]=s1[h]**

**h--;q++**

**h>=0**

da

nu

**sr[q]=' ';q++;k=0**

**c=’1’**

**c!=' '&&c!='\0'**

nu

da

**c=s[i]; i++**

**t++**

**sr[q-1]='\0'**

**sr; con+1**

**Stop**

Listingul:

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{ clrscr();

char s[50],s1[20],sr[50],c;

int t=1,i=0,k=0,q=0,h,con=0,j;

gets(s);

for(j=0;j<strlen(s);j++) if(s[j]==' ') con++;

while(s[i]!='\0'&&c!='\0')

{ if (t%2==0)

{

while (s[i]!=' '&&s[i]!='\0')

{

s1[k]=s[i];

k++;

i++;

}

k--;

for (h=k;h>=0;h--,q++)

sr[q]=s1[h];

sr[q]=' ';

q++;

k=0;

}

c='1';

while(c!=' '&&c!='\0')

{

c=s[i];

i++;

}

t++;

}

sr[q-1]='\0';

puts(sr);

printf("\n%d",con+1);

/\*i=0;

while (sr[i]!='\0'){printf("%c",sr[i]);i++; } \*/

getch();

}

**2. E X E M P L U de P R O G R A M:** *Sarcina lucrarii:* Este dat un şir de caractere care conţine de la 5 până la 25 cuvinte si cuvintele sunt separate prin spaţiu; iar dupa ultimul cuvint urmeaza punct. Determinaţi în baza schemei logice a algoritmului şi programului de mai jos după ce principiu se transformă fiecare cuvint al şirului? Găsiţi în ce segment al programului se efectuează următoarele:

a) deplasarea primei litere la sfârşitul cuvintului;

b) inlaturarea penultimei litere a cuvintului;

c) verificarea daca lungimea cuvintului este impara şi se înlătură litera din mijlocul lui.

**Schema bloc:**



**Varianta 1. Listingul programului** :

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{ clrscr();

printf("\n\t\t\t------------------------------------");

printf("\n\t\t\t---------- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ј");

printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

char s[100],cv[100][100],sc; int i,j,n,k,l;

printf("Introduceti propozitia dorita formata cel putin din 5 cuvinte si maximum din 25 sa fie sfirşita de punct.\nIntroduceti:");

gets(s);

k=0;l=0;i=0;

while(s[i]!='.')

{

if(s[i]==' ') { cv[k][l]='\0'; k++; l=0; i++; }

cv[k][l]=s[i];

l++; i++;

if(s[i]=='.') { cv[k][l]='\0'; k++; }

}

for(i=0;i<k;i++)

{

cv[i][strlen(cv[i])-1]='\0';

}

for(i=0;i<k;i++)

{

j=0; sc=cv[i][j];

for(j=0;j<strlen(cv[i]);j++)

{

if((j+1)==(strlen(cv[i]))) cv[i][j]=sc;

else cv[i][j]=cv[i][j+1];

}

}

for(i=0;i<k;i++)

{

if(strlen(cv[i])%2!=0)

{

for(j=strlen(cv[i])/2;j<strlen(cv[i]);j++)

{

cv[i][j]=cv[i][j+1];

if((j+1)==(strlen(cv[i]))) {cv[i][j]='\0'; break;}

}

}

}

for(i=0;i<k;i++)

{

//printf(" ");

for(j=0;j<strlen(cv[i]);j++)

printf("%c",cv[i][j]);

printf(" ");

}

getch();

}

**Varianta 2. Listingul programului**

#include<conio.h>

#include<stdio.h>

#include<string.h>

#define NMAX 500

void main() {

char s[NMAX],cv[NMAX],cuv[NMAX][25],\*p,c;

int ncuv=0,l,i,j;

clrscr();

printf("Introduceti şirul de caractere:\n");

gets(s);

// Extragerea cuvintelor din şir

p=strtok(s," ");

while(p) {

strcpy(cuv[ncuv++],p);

p=strtok(NULL," ");

}

// Stergerea punctului din ultimul cuvant

l=strlen(cuv[ncuv-1]);

if(cuv[ncuv-1][l-1]=='.') cuv[ncuv-1][l-1]='\0';

// a)

printf("\n a) Deplasarea primei litere la sfarsitul cuvintelor:\n");

for(i=0;i<ncuv;i++) {

strcpy(cv,cuv[i]);

l=strlen(cv);

c=cv[0];

for(j=0;j<l-1;j++)

cv[j]=cv[j+1];

cv[l-1]=c;

printf("%s ",cv);

}

printf("\b.\n\n");

// b)

printf("b) Inlaturarea penultimei litere a cuvintelor:\n");

for(i=0;i<ncuv;i++) {

strcpy(cv,cuv[i]);

l=strlen(cv);

if(l>1) {

cv[l-2]=cv[l-1];

cv[l-1]='\0';

}

printf("%s ",cv);

}

printf("\b.\n\n");

// c)

printf("c) Inlaturarea literei din mijloc a cuvintelor de lungime impara:\n");

for(i=0;i<ncuv;i++) {

strcpy(cv,cuv[i]);

l=strlen(cv);

if(l%2) {

for(j=l/2;j<l-1;j++) cv[j]=cv[j+1];

cv[l-1]='\0';

}

printf("%s ",cv);

}

printf("\b.\n\n");

getch();

}

# **Întrebările de autocontrol**

1.1. Caracterizaţi şirurile caracteriale şi declaraţi în diferite moduri posibile în limbajul C.

1.2. În ce mod se citesc şi se afişează elementele şirurilor caracteriale?

1.3. Enumeraţi şi exemplificaţi apelurile funcţiilor predefinite pentru prelucrarea şirurilor caracteriale.

* 1. Modificaţi subprogramele din EXEMPLUL de PROGRAM-MODEL, utilizând funcţii recursive şi alte funcţii predifinite.
  2. Unde este eroarea ?

1.5.1 char s[14]; strcpy(s, "Ce mai faci ?\n");

1.5.2. char s[14]; scanf("%s", &s);

* 1. Pentru programul din exemplul utilizării şirurilor de caractere (ca vectori) de mai sus analizaţi şi prin schema logică a algoritmului.
  2. Apreciaţi ce face următoare funcţie:

int firstword(char \*S, char \*buf)

║ { int i=0;

║ while(S[i]!=0 && i<strlen(S))

║ { if(S[i]==' ' && i<strlen(S)) i++;

║ if(S[i]!=' '&& i<strlen(S))

║ { int j=0;

║ while(S[i]!=' ' && i<strlen(S))

║ { buf[j]=S[i];

║ j++; i++;

║ }

║ buf[j]=0;

║ return 0;

║ }

║ }

║return 0;

║}

* 1. Analizati corectitudinea si explicati descrierile, scopul si rezultatele ce se vor obţine:

int f(){const char \*pc=”abc”; pc[1]=’x’; pc=”123”; return;}

int g(){char\*const cp=”abc”;cp[1]=’x’; cp=”123”; return cp;}

int h(){const char\*const cpc=”abc”; cpc[1]=’x’; cpc=”123”; return cpc;}

int{\*arr[3] ) ()={f,g,h};

main(){int i;for(i=0;i<3;i++)printf (”%d\n”,(\*arr[i])(i+7));}

1.9. Lansati in executie mediul Turbo C++, deschizand meniurile si urmarind descrierea de mai sus. Editati, compilati si rulati urmatorul program C, care citeste initialele numelui vostru si afiseaza un salut ( liniile pot fi selectate si aduse in TC prin intermediul unui fisier text ):

#include <stdio.h>   
#include <conio.h>

void main(void){  //antet main   
 //corp main   
 char c1,c2; //variabile locale   
 clrscr(); //se sterge ecranul   
 puts("\*\*\* Buna! \*\*\*");   //1   
 printf("Initialele numelui tau:");  c1=getche();c2=getche(); // citirea initialelor numelui   
 printf("\nImi pare bine, %c.%c.!\n",c1,c2); //2   
 puts("\*\*\* Succes! \*\*\*");  getche();    
} // terminare corp main

1.10. Corectaţi, completaţi şi rulati urmatoarele doua secvente, observand conversiile implicite si explicite:

       char c='a',cc;      int i=4;      float f=5.95;   
     printf("%d %f\n",i,f);      i=f; // conversie implicita, trunchiere   
     printf("%d %f\n",i,f);      f=i+100000; // conversie implicita a rezultatului expresiei   
     printf("%d %f\n",i,f);      i=-99.001; // conversie implicita, trunchiere   
     f='a';   
     c=0x3239; cc=-i; // preluarea ultimului octet   
     printf("%d %f %c %c\n",i,f,c,cc);

     float r1=5/2,   
     r2=(float)5/2,   
     r3=(float)(5/2),   
     r4=5/(float)2,   
     r5=(float)5/(float)2;   
     printf("%f %f %f %f %f\n",r1,r2,r3,r4,r5);

**ANEXA Ll nr.6. Variantele L.l. “Prelucrarea şirurilor caracteriale”**

**De elaborat algoritmii şi programele în C:**

1 - Se citeşte de la tastatura un şir (s) şi Sa se determine lungimea cuvintelor şi şirului

2 - Se citeşte de la tastatura un şir (s) şi Sa se introduca in şir un alt şir citit de la tastatura in pozitia a 4-a

3 - Se citeşte de la tastatura un şir (s) şi Sa se stearga din şir 3 caractere, incepand din pozitia a 5-a

4 - Se citeşte de la tastatura un şir (s) şi Sa se determine de cate ori apare in şir caracterul ‘a’ / un caracter citit de la tastatura / ultimul caracter citit / al treilea caracter citit

5 - Se citeşte de la tastatura un şir (s) şi Sa se determine cate majuscule / litere mici / cifre conţine şirul

6 - Se citeşte de la tastatura un şir (s) şi Sa se determine cate cuvinte conţine şirul

7 - Se citeşte de la tastatura un şir (s) şi Sa se determine cate cuvinte de lungime 1 conţine şirul

8- Se citeşte de la tastatura un şir (s) şi Sa se transforme in majuscula fiecare prima litera a unui cuvant.

9 - Se citeşte de la tastatura un şir (s) şi Sa se dubleze vocalele

10 - Sa se citească de la tastatura caractere si, intr-un vector, sa se stocheze primele p cifre citite

11 - Se citeşte de la tastatura un şir (s) şi Sa se citească de la tastatura caractere si, intr-un vector, sa se stocheze primele p paranteze drepte citite. Sa se verifice daca şirul este corect, adica:

* 1/ numarul de paranteze deschise este egal cu numarul de paranteze inchise;
* 2/ orice paranteza inchisa are in stanga ei un numar mai mare de paranteze deschise decat inchise.

12 - Se citeşte de la tastatura un şir (s) şi Sa se afiseze şirul astfel:

EDIT, DITU, ITUR, TURA

13. Pentru propoziţia dată găsiţi cuvintele nesimetrice cu lungimea impară şi simetrice cu lungimea pară. Apoi înlocuiţile cu valoarea lor numerică, alcătuită din suma codurilor fiecărei litere din cuvînt.

14. Din propoziţia dată eliminaţi cuvintele care se întîlnesc în propoziţie de mai multe ori şi înlocuiţi-le cu numărul format din suma literelor în cuvînt şi codul lor. Calculaţi şi vizualizaţi frecvenţa repetării fiecărei cifre din numerele obţinute.

15. Se citeşte de la tastatura un şir (s) şi Sa se testeze algoritmii de descompunere în 3 părţi pentru şirul “UNIVERSITATEA”

16. Se citeşte de la tastatura un şir (s) şi Sa se determine şi să afişeze câţi muguri sunt in vector (un mugur e o paranteza deschisa urmata de una inchisa). Sa se afiseze şirul cu o culoare, iar mugurii sa fie afisati cu alta culoare.

17. Să se elaboreze algoritmul şi să se scrie un program care:

* citeste cuvintele tastate fiecare pe cate un rand nou, pana la CTRL/Z  ( varianta: pana la introducerea unui cuvant vid )
* afiseaza cuvantul cel mai lung

18. Organizaţi prelucrările pentru ca programul să deternine şi sa mai afiseze:

* cuvintele in ordine alfabetica - se va modifica [functia de ordonare](http://www.timsoft.ro/aux/module/modul6.html#ordonare) a unui tablou de intregi
* cuvântul ce conţine numarul maxim de vocale, respectiv consoane

19. Pornind de la un şir citit de la tastatura, sa se construiasca algoritmul şi sa se tipareasca ( ultimele trei prelucrari nu sunt implementate ):

* şirul care conţine cifrele din cel iniţial
* şirul care conţine minusculele vocalelor din cel iniţial
* şirul invers celui iniţial; sa se verifice daca şirul iniţial este palindrom

20. Se citesc trei şiruri s1, s2 si s3. Sa se construiasca algoritmul pentru afişarea şirului obtinut prin inlocuirea in s1 a tuturor aparitiilor lui s2 prin s3. (Observatie: Daca s3 este şirul vid, din s1 se vor sterge toate subşirurile s2 ).

**21**  Pentru propoziţia dată găsiţi cel mai lung cuvînt simetric de pe poziţii pare şi cel mai scurt cuvînt simetric de pe poziţii impare. Dacă astfel de cazuri nu se întîlnesc, atunci modificaţi cuvintele prin permutări de litere. Să se afişeze şirurile obţinute şi numărul de operaţii efectuate.

22. Să se elaboreze algoritmul şi să se scrie un program care:

* construieste si afiseaza cuvantul format din ultima litera din cuvintele introduse, transformata in minuscula ( nu vor fi prelucrate cuvintele ce nu se termina cu o litera )
* afiseaza cuvintele ce incep cu o vocala.

23. Organizaţi prelucrările pentru ca programul să determine şi sa mai afiseze:

* cuvintele ce sunt reprezentarea unor numere intregi sau reale
* cuvintele inversate
* cuvintele care sunt palindroame - un palindrom este un cuvânt identic cu inversatul sau.

24. Pornind de la un şir citit de la tastatura, sa se construiasca algoritmul şi sa se tipareasca ( ultimele trei prelucrari nu sunt implementate ):

* şirul care conţine majusculele literelor din cel iniţial
* şirul care conţine m caractere incepand de la pozitia p din şirul initial; m si p se citesc
* şirul care conţine caracterele celui iniţial cu exceptia celor m incepand de la pozitia p; m si p se citesc.

25. Este dat un şir de caractere care conţine de la 1 până la 22 cuvinte, unde fiecare din aceste cuvinte include de la 1 până la 5 litere şi cuvintele sunt separate prin virgulă; iar după ultimul cuvânt urmează punct. Afişaţi şi vizualizaţi:

* şirul în care cuvintele şirului iniţial sunt situate în ordinea inversă;
* cuvintele înaintea cărora în şirul dat se află numai cuvintele ce le precedează după alfabet, iar după aceea numai cuvintele ce le urmează;
* şirul iniţial fără cuvintele ce se repetă mai mult de 2 ori.

26. De introdus de la tastatură o propoziţie. Cuvintele cărora sunt separate prin spaţiu, iar după ultimul cuvânt urmează punct. De afişat din propoziţie şirurile care corespund cerinţelor următoare:

* şirul obţinut din propoziţia dată, eliminând cuvintele de pe poziţii impare;
* şirul obţinut din propoziţia dată, inversând cuvintele de pe poziţii pare;
* subşirurile egale din ambele şiruri: cel iniţial şi cele obţinute.

27. Este dat un şir de caractere care conţine de la 5 până la 27 cuvinte şi cuvintele sunt separate prin spaţiu; iar după ultimul cuvânt urmează punct. Transformaţi şi afişaţi fiecare cuvânt a şirului după regulile următoare:

* deplasaţi prima literă a cuvântului la sfârşitul lui;
* deplasaţi prima literă la sfârşitul cuvântului;
* înlăturaţi penultima literă a cuvântului;
* dacă lungimea cuvântului este impară, atunci înlăturaţi litera din mijlocul lui.

28. Se consideră dată o frază alcătuită din 2 cuvinte cu m şi respectiv n litere. Să se transforme primul cuvânt în cuvântul al doilea, utilizând trei operaţii:

* adaugarea unei litere;
* modificarea unei litere;
* ştergerea unei litere.

Transformarea să se facă prin număr minim de operaţii. Afişaţi numărul minim de operaţii precum şi şirurile intermediare şi final. Ex: costul transformării cuvintelor DANIA în IOANA este 6.

29. Pentru propoziţia dată căutaţi şi vizualizaţi perechi de cuvinte în care unul din cuvinte este inversarea altuia. Dacă astfel de cazuri nu se întâlnesc, atunci modificaţi unul din cuvinte prin permutări.

30. Pentru propoziţia dată calculaţi frecvenţa repetării fiecărui cuvânt, fiecărei rădăcini şi vocale. Se cere afişarea numărului de operaţii precum şi şirurile intermediare şi finale.

31. Pentru propoziţia dată găsiţi cel mai lung cuvânt simetric. Dacă astfel de cazuri nu se întâlnesc, atunci modificaţi cuvintele prin permutări.

32. Propoziţia dată de transformat conform definiţiei că distanţa dintre două cuvinte să fie egală cu diferenţa numărului de poziţii iniţiale a acestor cuvinte în propoziţia iniţială. Apoi găsiţi o pereche de cuvinte cu distanţa medie.

33. Pentru 2 propoziţii date găsiţi setul de cuvinte care se întâlnesc în fiecare din ele. Apoi calculaţi numărul de cuvinte în set şi lungimea fiecărui cuvânt.

34. Din propoziţia dată eliminaţi cuvintele situate pe poziţii impare şi inversaţi cuvintele de pe poziţiile pare. Afişaţi numărul de operaţii precum şi şirurile intermediare şi finale.

35. Găsiţi cel mai lung cuvânt comun pentru două propoziţii date şi frecvenţa repetării rădăcinii lui în alte cuvinte. Afişaţi numărul de operaţii precum şi şirurile intermediare şi finale.

36. Sunt date două propoziţii. Găsiţi cel mai scurt cuvânt din prima propoziţie care nu este în a doua propoziţie. Apoi găsiţi perechile de cuvinte cu lungimi egale.

37. Pentru propoziţia dată găsiţi cuvintele nesimetrice cu lungimea impară şi simetrice cu lungimea pară. Apoi înlocuiţile cu valoarea lor numerică, alcătuită din suma codurilor fiecărei litere din cuvânt.

38. Din propoziţia dată eliminaţi cuvintele care se întâlnesc în propoziţie de mai multe ori şi înlocuiţi-le cu numărul format din suma literelor în cuvânt şi codul lor. Calculaţi şi vizualizaţi frecvenţa repetării fiecărei cifre din numerele obţinute.

39. Este dată o propoziţie care conţine de la 5 până la 25 de cuvinte, unde cuvintele sunt separate prin spaţiu; iar după ultimul cuvânt urmează punct. Afişaţi cuvintele care corespund următoarelor proprietăţi:

a) primă literă a cuvântului ce se repetă de mai multe ori în fiecare cuvânt;

b) cuvântul este format din segmentul iniţial ale alfabetului (a- ab- abc- etc.);

c) cuvântul cu cea mai mare frecvenţă de repetări a fiecărui caracter.

40. Într-o expresie aritmetică care conţine peste 5 subexpresii să se genereze toate subşirurile, alcătuite din subexpresii, cuprinse între paranteze rotunde (). Să se verifice modurile de includere în cele n paranteze care trebuie să se închidă corect. Ex: n=4 sunt 2 soluţii: (()) şi ()().

1. Se dau doua şiruri de caractere. Să se construiască:

* un şir ce conţine caracterele comune din cele două şiruri, însă nu depăşesc 1/3 din lungimea şirului;
* un şir ce conţine caracterele care există în primul şir, dar nu există în al doilea, însă nu depăşesc 1/2 din lungimea primului şir;
* un şir ce conţine caracterele care există în al doilea şir, dar nu există in primul, însă nu depăşesc 1/2 din lungimea şirului al doilea.

42. Este dat un şir de caractere care conţine de la 3 până la 21 de cuvinte, unde fiecare din aceste cuvinte include de la 2 până la 5 litere şi cuvintele sunt separate prin virgulă; iar după ultimul cuvânt urmează asteriscul.

43. Afişaţi şi vizualizaţi:

a) cuvintele care se întâlnesc în şir numai o singură dată;

b) cuvintele ce se repetă, cu frecvenţa repetării lor în şir.

c) cuvintele ordonate după alfabet şi invers.

44. Pentru propoziţia dată să se caute perechile de cuvinte în care unul din cuvinte este inversul altuia. Dacă astfel de cazuri nu se întâlnesc, atunci să se modifice unul dintre cuvinte prin adăugări succesive a literelor absente pentru obţinerea inversului altuia. Calculaţi numărul de operatii elementare efectuate

45. Într-un text citit de pe mediul de intrare care conţine trei propoziţii să se înlocuiască toate apariţiile unui şir de caractere ”şir1”, citit înaintea textului, prin cel mai scurt cuvânt din prima propoziţie şi nu se conţine în a doua propoziţie. Apoi de găsit cuvintele cu lungimi egale.

46. Este dat un şir de caractere care conţine de la 10 până la 32 cuvinte, unde fiecare din aceste cuvinte include de la 5 până la 9 litere şi cuvintele sunt separate prin virgulă; iar după ultimul cuvânt urmează punct.

Să se afişeze următoarele fragmente:

1. cuvintele care sunt palindromuri;
2. cuvintele înaintea cărora în şirul dat se află numai cuvinte cu lungime impară, apoi urmează numai cuvinte mai scurte;
3. şirul fără cuvinte ce se repetă.

47. Este dat un şir de caractere care conţine de la 5 pina la 30 cuvinte si cuvintele sunt separate prin spatiu, iar dupa ultimul cuvint urmeaza punct. Transformati fiecare cuvint al şirului dupa regulile urmatoare:

1. deplasati prima litera la sfirşitul cuvintului
2. inlaturati litera din mijlocul cuvintului
3. virificati daca lungimea cuvintului este para, atunci inlaturati a treia litera.

48. Pentru propoziţia dată să se calculeze frecvenţa repetării fiecărui cuvânt, fiecărei rădăcini şi vocale. Să se afişeze şirurile obţinute şi numărul de operaţii efectuate.

1. Scrieţi un program care sa-si converteasca intrarea la litere mici , folosind o functie lower(c) care intoarce pe c daca c nu e litera si valoarea literei mici daca c e litera.
2. Scrieţi un program care sa tipareasca o histograma a lungimii cuvintelor de la intrarea sa . Histograma se va trasa orizontal.
3. Scrieţi un program care afiseaza cuvintele de la intarare cate unul pe o linie.
4. Scrieţi programul de tab care sa inlocuiasca tab-urile din intrare cu numarul potrivit de blank-uri pentru a se ajunge la urmatoarea coloana de tabulare. Se presupune un set fixat de coloane de tabulare de exemplu n.
5. Scrieţi un program entab care inlocuieste şirurile de blank-uri prin numrul minim de tab-uri si blank-uri care realizeaza aceeasi spatiere.
6. Scrieţi un program care elimina toate comentariile dintr-un program C.
7. Scrieţi un program care sa gaseasca erorile elementare de sintaxa dintr-un program C , de genul paranteze rotunde , patrate , acolade neinchise , ghilimele , apostroafe, comentarii.
8. Scrieţi o funcţie proprie reverse(s) care sa inverseze şirul de caractere s.
9. Elaborati un program cu o funcţie proprie care compara 2 şiruri de caractere astfel:

* daca şirurile sunt egale intoarce valoarea 0 ;
* daca şirul 1 >şirul 2 intoarce -1;
* daca şirul 2 >şirul 1 intoarce 1.

Şirul 1 şirul 2 are urmatoarea semnificatie:- un caracter din şirul 1 are codul ASCII mai mic decât caracterul corespunzator aceleasi pozitii din şirul 2.

## Bibliografie

1. O. Catrina, I. Cojocaru, Turbo C++, ed. Teora 1993
2. V. Petrovici, Florin Goicea, Programarea in limbajul C. Eed. Teora 1999
3. Liviu Negrescu, ,,Limbajul C” ,volumul I\_partea I-a si partea II-a16 16ditura MicroInformatica, Cluj-napoca 2001
4. Б.Керниган, Д.Ритчи. Язык программирования Си. Санкт-Петербург, 2001,,, Brian Kernighan, Dennis Ritchie *” The C Programming Language” este în l. română format electronic*
5. Vlad Caprariu ”Ghid de utilizare Turbo C” Cluj - Napoca 1993.
6. Cristea Valentin. Tehnici de programare. Ed.: Bucur., Teora, 1993. /681.3; T29/
7. Odagescu Ioan, Copos Cristina s.a. Metode si Tehnici de programare./enunturi, solutii, probleme propuse/ Ed.:Bucur.: INTACT, 1994 /681.3; O23/
8. Tudor Bălănescu. Corectudinea agoritmilor.Bucur.:Ed. Tehn.1995