Ll7 TEMA: ***Prelucrarea şirurilor caracteriale cu ajutorul funcţiilor standarde***

**Obiectivele temei**

1. *Insuşirea principiilor si perfecţionare tehnicii de programare cu subprograme standarde si definite pentru prelucrarea datelor şi şirurilor de caractere in TC.*
2. *Insuşirea procedeelor de elaborare si utilizarea efectiva a programelor cu proceduri si funcţii predefinite pentru prelucrarea şirurilor caracteriale.*
3. *Analiza elementelor specifice ale funcţiilor predefinite prin algoritmizare si programare pentru prelucrarea fragmentelor de texte prin comparaţia soluţiilor stereotipice cu cele eficiente pentru diverse moduri de manipulare ale textelor.*

Subiectele temei şi ordinea executării

1. Aprofundarea studierei principiilor prelucrării (descrierii, declarării, formării, etc.) şirurilor caracteriale în C.
2. Studierea metodelor şi tehnicilor de bază de prelucrare a textelor ca şiruri caracteriale, utilizînd subprograme definite şi predefinite.
3. Elaborarea algoritmului şi programului de soluţionare a variantei.
4. Depanarea programului şi verificarea corectitudinei cu ajutorul testelor elaborate.

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

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

***Noţiuni generale***

**Functii de prelucrare a sirurilor** Majoritatea compilatoarelor C/C++ au incorporate funcţii speciale pentru lucrul cu şirurile. Evident, că pentru aşa scopuri se poate crea funcţii proprii, dar este mai efectiv de folosit funcţiile din bibliotecile standarde. Prototipurile acestor funcţii sunt descrise în biblioteca *string.h* şi deci pentru folosirea lor, în program trebuie inclusă această bibliotecă folosind sintaxa: *#include <****string.h****>*.

Funcţii din string.h:

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

char \***strcpy**(char \*d, const char \*s);

char\* **strncpy**(char \*d, const char \*s, unsigned n);

char \***strdup**(const char \*s);

int **strlen**(const char \*s);

char \***strcat**(char \*d, const char \*s);

char \***strncat**(char \*d, const char \*s, unsigned n);

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

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

char \***strstr**(const char \*s, const char \*subsir);

Unele din aceste funcţii le vom analiza:

**1. Functia strlen.**  Prototip: **size\_t strlen(const char \*S);**

Efect: Determină lungimea şirului de caractere S. În majoritatea cazurilor lungimea şirului nu coincide cu lungimea masivului în care se află şirul. Adică lungimea masivului este mai mare. Funcţia strlen() determină lungimea reală a şirului şi întoarce o valoare de tip întreg egală cu cantitatea de caractere ce se află în şir.

**int strlen (char str[])**

**{ int len; for(len=0; str[len]; len++); return len; }**

Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void) { char name[20]; int k; puts (“culege numele”); gets (name);***

***k=strlen(name); printf (“Numele dvs are % d caractere”,K); getch(); }***

Are rolul de a returna lungimea unui sir(fara a lua in considerare caracterul nul).

Forma generala : strlen(nume sir);

Exemplu în C++: Se citeste un sir de caractere si se afiseaza numarul de caractere a sirului

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main() {

char s1[100];

cin.get(s1,100);

cin.get();

cout<<"Sirul citit are"<<" "<<strlen(s1)<<" "<<"caractere";

getch(); }

2. Funcţia **strcmp();** Prototip: int strcmp (const char \* S1, const char\* S2); Efect: Compară două şiruri.

Limbajul C nu permite compararea a două şiruri în forma: *if (S1==S2) ;* Aici compararea este făcută cu ajutorul funcţiei strcmp() care întoarce valoare nulă, în caz că şirurile sînt identice, sau o valoare diferită de zero, în caz că şirurile nu coincid. După executarea funcţiei strcmp(), va fi întoarsă o valoare întreagă care va fi: mai mică ca 0 dacă S1<S2; mai mare ca 0 dacă S1>S2; egală cu 0 daca S1==S2; Exemplu:

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

returneaza: <0, daca s1 < s2

         0, daca s1 = s2

       >0, daca s1 > s2

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void) { char S1 [30]; char S2 [30]; int k;***

***gets (S1); gets (S2); k=strcmp(S1,S2);***

***If (k==0) puts (“sirurile coincid”); else puts (“sirurile nu coincid”); getch();}***

* + int strncmp(char \*s1,char \*s2,int n); // comparare a doua siruri pe lungimea n

**3.** Funcţia **strcpy();** Prototip: char \*strcpy(char \*S1, const char \*S2);

* + Forma generala: strcpy(destinatie,sursa). Efect: Copie şirul S2 în şirul S1. După îndeplinirea funcţiei *strcpy (S1,S2);* şirul S1 îşi va pierde valoarea iniţială şi va avea valoarea nouă din S2. Iar S2 va rămîne neschimbat. Copiaza sirul sursa s in sirul destinatie d; returneaza  adresa sirului destinatie.

Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void) { char name [30]; char fam [30]; puts (“Culege numele”); gets (nume);***

***puts (“culege familia”); gets (fam); strcpy (nume, fam); puts(nume); puts(fam); getch(); }***

În rezultatul îndeplinirii acestui exemplu numele va fi identic cu familia.

Exemplu C++:

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main() {

char s1[20]="Limbalul Turbo C++",s2[20]="Limbajul C++";

strcpy(s1,s2); cout<<s1; getch(); }

**4. Functia strcat.** Prototip: **char \*strcat (char \*dest, const char \*sursa);**

Forma generala: **strcat(destinatie,sursa)**

Funtia are rolul de a adauga sirului de la adresa destinatie sirul de la adresa sursa, adică concateneaza cele doua siruri si returneaza adresa sirului rezultat.

Exemplu 1:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void) {***

***char dest [50]; char sursa [5];***

***dess=“Turbo ”; sursa =“C++”;***

***strcat (dest,sursa); puts(dest); getch(); }***Aici rezultatul va fi: *dest=Turbo C++*.

Exemplu 2 în C++:

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main()

{

char s1[20]="Limbajul Turbo C++", s2[20]=" si Limbajul C++";

strcat(s1,s2);

cout<<s1;

getch();

}

# 5. Functia strncat Prototip: char \*strncat (char \*dest, const char \*sursa, size\_t k);

Forma generala: strncat(destinatie,sursa,n)

Functia adauga sirului destinatie primii n octeti ai sirului sursa adică adaogă un număr egal cu *k* caratere de la începutul şirului sursa la sfărşitul şirului dest.

Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char a[20]=“Turbo”; b[10]=“Pascal”; strncat (a, b, 3); puts (a); }***

În rezultatul executării acestui exemplu primele 3 caractere din şirul *b[10]*, (adica subşirul “Pas”) vor fi adaogate la sfîrşitul şirului *a[20].* Rezultatul executării funcţiei strncat() va fi de tip şir de caractere şi se va conţine în vaiabila *a*. După executarea exemplului variabila a va conţine valoarea “*TurboPas*”.

Exemplu:

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main()

{

char s1[20]="Limbajul Turbo C++",s2[20]=" si Limbajul C++";

strncat(s1,s2,5);

cout<<s1;

getch();

}

# 6. Functia strchr Prototip: char \*strchr(const char \*S, int c);

Forma generala: strchr(nume sir, ' caracter ' )

Efect: Scanează şirul S în căutarea caracterului c. În caz de succes funcţia întoarce un indicator la caracterul din S care primul a fost găsit identic cu caracterul c. În caz de eroare (dacă aşa caracter nu există în şirul S) funcţia întoarce valoare nulă. Exemplu1:

*#include <string.h>*

*#include <conio.h>*

*#include <stdio.h>*

***void main(void) { clrscr(); char S[15]; char \*ptr, c = 'r'; strcpy(S, "TurboC++"); ptr = strchr(S, c);***

***if (ptr) {printf("Caracterul %c are pozitia %d in sirul %s\n", c, ptr-S, S); puts(ptr);}***

***elseprintf("Caracterul n-a fost gasit\n"); getch(); }***

În rezultatul îndeplinirii acestui exemplu la monitor va fi afişat mesajul:

*Caracterul r are poziţia 2 în şirul TurboC++* şi apoi datorită funcţiei *puts(ptr);* va fi afişat şirul S trunchiat de la caracterul *r* la care indică *ptr*: *„rboC++”*.

Functia cauta in sirul nume sir caracterul caracter si returneaza subsirul care incepe cu prima aparitie a caracterului citit

Exemplu2: se va lista Turbo C++

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main()

{

char s1[20]="Limbajul Turbo C++";

cout<<strchr(s1,'T');

getch();

}

Exemplul3: Se tipareste indicele primei aparitii a caracterului 'u'

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main()

{char s1[20]="Limbajul Turbo C++";

cout<<strchr(s1,'u')-s1;

getch();} *Returneaza valoarea 6*

8. Funcţia **strcspn();** Prototip: size\_t strcspn (const char \*S1, const char \*S2);

Efect: Determină poziţia caracterului din şirul S1, care primul a fost întîlnit în şirul S2. Întoarce o valoare de tip întreg egală cu numărul de ordine a acestui caracter. Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char name[20],fam[20]; int k; puts ("culege numele"); gets (name);***

***puts ("culege familia"); gets (fam); k=strcspn(name,fam);***

***printf ("Simbolul %c din %s primul a fost gasit in %s", name[k],name,fam); getch(); }***

Dacă în cazul nostru name=“Stepan” şi fam=“Ivanov”, atunci rezultatul funcţiei *k=strcspn(name,fam);* va fi k=4, din cauză că pe locul patru în numele “Step**a**n” se află caracterul “a” care primul a fost depistat în “Iv**a**nov” .

9. Funcţia **strspn();** Prototip: size\_t strspn(const char \*S1, const char \*S2);

Efect: Determină poziţia caracterului din şirul S1 începînd cu care S1 diferă de S2. Întoarce o valoare tip întreg egală cu poziţia acestui caracter. Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char name[20],fam[20]; int k; puts ("culege numele"); gets (name);***

***puts ("culege familia"); gets (fam); k=strspn(name,fam);***

***printf ("Începînd cu simbolul %c şirul %s diferă de %s", name[k],name,fam); getch(); }***

În caz că name=“Stepan”, iar fam=“Stoianov” lui k va fi atribuită valoarea 2 din cauză că anume începînd cu caracterul cu numărul de ordine 2 variabila *nume* diferă de variabila *fam*.

10. Funcţia **strdup();** Prototip: char \*strdup (const char \*S);

Efect: Dublează şirul de caractere S. În caz de succes funcţia strdup() întoarce ca valoare indicatorul adresei de memorie, ce conţine şirul dublat. Şi întoarce valoare nulă în caz de eroare. Funcţia strdup(S) face o copie a şirului S, obţinînd spaţiu prin apelul funcţiei malloc(). După folosirea şirului dublat programatorul trebuie să elibereze memoria alocată pentru el. Exemplu :

*#include < alloc.h>*

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char a []=”UTM”; char \*b; b=strdup(a); puts (b); free (b);}***

Aici *a* este un şir de caractere, iar *b* un indicator la adresa de mmorie unde se va înscrie şirul dublat. La executarea funcţiei *strdup()* valoarea şirului din *a* este copiat într-o locaţie de memorie, adresa căreia se află în *b*. Funcţia *puts()* afişază conţinutul şirului dublat la monitor. Funcţia *free()* elibereză memoria ocupată pînă acum de şirul dublat.

11. Funcţia **strlwr();** Prototip: **char \*strlwr (char \*S);**

Efect: Trece toate caracterele din şirul S în echivalentul lor minuscul. În calitate de parametru funcţia foloseşte o variabilă de tip şir de caractere. În rezultatul executării acestei funcţii, dacă în şir se vor conţine caractere majuscule, ele vor fi transformate în echivalentul lor minuscul, iar dacă în şir caractere majuscule nu se vor conţine - şirul va rămâne neschimbat. Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char a[10]= “Pascal”; strlwr(a); puts (a); getch(); }***

12. Funcţia **strupr();** Prototip: **char \*strupr(char \*S);**

Efect: transformă toate caracterele din şir în echivalentul lui majuscul. Exemplu: Echivalent cu funcţia strlwr();

**13. Functia strrchr** Prototip: **char \*strrchr(const char \*s, int c);**

Efect: Funcţia caută ultima apariţie a caracterului c în şirul S. În caz de succes funcţia întoarce un indicator la ultimul caracterdin S identic cu caracterul c. Exemplu1:

*#include <stdio.h>*

*#include <conio.h>*

*#include <string.h>*

***void main(void) { clrscr(); char S[15]; char \*ptr, c = 'r'; strcpy(S,"Programare"); ptr = strrchr(S, c);***

***if (ptr) {printf("Caracterul %c este pe pozitia: %d\n", c, ptr-S); puts(ptr);}***

***else printf("The character was not found\n"); getch();}*** În rezultatul îndeplinirii acestui exemplu la monitor va fi afişat mesajul:

*Caracterul r este pe poziţia 8* şi apoi datorită funcţiei *puts(ptr);* va fi afişat şirul S trunchiat de la ultimul caracter *r* la care indică *ptr*: *„re”*.

Exemplu1: Returneaza adresa ultimei aparitii a caracterului cautat strrchr(sir, 'caracter' )

char s1[20]="Limbajul Turbo C++";

cout<<strrchr(s1,'u')-s1;

Returneaza valoarea 10

14. Funcţia **strncpy();** Prototip: **char \*strncpy (char \*dest, const char \*sursa, size\_t n);**

Efect: Copie un numar dat de caractere dintr-un şir în altul; Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char a[40]=”turbo”; b [40]=”basic “; strncpy (a,b,2); puts (a); }***

Funcţia strncpy() înscrie un număr N dat de caractere din şirul sursă la începutul şirului destinaţie. Dacă şirul destinaţie va avea lungime mai mare ca N, atunci rezultatul va avea începutul egal cu caracterele copiate, iar sfîrşitul iniţial. Valoarea rezultatului se va conţine în şirul destinaţie. În exemplu de mai sus variabila a va avea valoare egală cu “*barbo*”.

15. Funcţia **strnset();** Prototip: **char \*strnset (char\* s, int ch, size\_t n) ;**

Efect: Funcţia strnset() copie caracterul *ch* pe primele n locuri din şirul \*S. În caz că n>strlen (s), atunci n va deveni egal cu strlen(s) Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char a[15] = “student” , b=W ; strnset (a,b,3); puts (a) ;}***

În urma executării funcţiei strnset() va fi întoarsă o valoare de tip şir care va fi înscrisă în şirul destinaţie . Din exemplu de mai sus *a =”WWWdent”.*

16. Funcţia **strrev();** Prototip **char \*strrev( char \*s);**

Efect: Funcţia strrev() inversează şirul de caractere S. După execuţia funcţiei strrev() primul caracter din şir va fi schimbat cu locul cu ultimul caracter, caracterul 2 cu penultimul, etc. fără a lua în consideraţie caracterul nul.

Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char s1[10]=”student”; printf(“Şirul iniţial - %s\n”,s1); strrev(s1);***

***printf(“Şirul final - %s\n”,s1);}*** După executarea programului va fi primit mesajul: Şirul final – tneduts.

17. Funcţia **strstr ();** Prototip: **char \*strstr(const char \*s1, const char \*s2);**

Efect: Funcţia strstr() determină dacă şirul S2 se conţine în şirul S1. Funcţia întoarce un indicator la caracterul din S1 începînd cu care a fost depistat şirul S2. Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char S1[20], S2[20], rez; S1 = “ international”; S2 = “nation”;***

***rez = strstr (S1,S2); printf (“subsirul este : %s”,rez); }***

Rezultatul: “subşirul este: national ;”. Dacă şirul S2 n-a fost depistat în S1, funcţia strstr() întoarce valoarea *“null”.*

18. Funcţia **strset();** Prototip: **char \*strset(char \*s, int ch);** Efect: Schimbă toate caracterele din şirul S în valoarea caracterului c. Rezultatul final se va păstra în acelaşi şir S. Exemplu:

*#include<string.h>*

*#include<stdio.h>*

*#include<conio.h>*

***void main(void){ char s[10]=”student”; printf(“Şirul iniţial - %s\n”,s); strset(s,’a’);***

***printf(“Şirul final - %s\n”,s); getch();}***

În rezultat va fi primit: *Şirul iniţial – student, Şirul final – aaaaaaa.*

Initializare in limbajele C si C++ [aut. Sergiu G. Istrati].pdf

19. Funcţia **strerror();** Prototip: **char \*strerror(int errnum);** Efect: Determină eroarea după numărul erorii şi returnează un indicator la şirul de caractere ce conţine descrierea erorii. Exemplu:

*#include <stdio.h>*

*#include <conio.h>*

*#include <errno.h>*

***void main(void) { clrscr(); char \*numerr; numerr = strerror(11);***

***printf("Eroarea este: %s\n", numerr); getch(); }***

În rezultaul executării acestui exemplu funcţia *strerror(11);* va determina eroarea după numărul ei, iar funcţia printf va afişa: *Eroarea este: Invalid format.*

20. Funcţia **strpbrk();** Prototip: **char \*strpbrk(const char \*s1, const char \*s2);**

Efect: Funcţia caută în şirul S1 primul caracter ce există şi în şirul S2; În caz de succes funcţia întoarce un indicator la primul caracter din S1 apărut în S2. Exemplu:

*#include <stdio.h>*

*#include <conio.h>*

*#include <string.h>*

***void main(void) { clrscr(); char \*S1="Universitate"; char \*S2="Moldova"; char \*ptr;***

***ptr=strpbrk(S1,S2); if (ptr) printf("Primul caracter din S1 gasit în S2 este: %c\n", \*ptr);***

***else printf("Caracter nu a fost gasit\n"); getch(); }*** În rezultat va fi găsit caracterul *„v”* .

Exemple (utile): Litere mari -> litere mici: transformarea literelor mari in mici folosind operatori de deplasare.

#include

#include

#include

void main()

{

clrscr();

int c;

while ((c = getchar()) != EOF)

{

if (isupper(c)) // sau (c>='A' && c<='Z')

putchar(c | (1 << 5));

else

putchar(c);

}

}

21. char **\*strdel**(char \*p, int n); // sterge n caractere din pozitia poz

22. char **\*strins** (char \*p, char \*s); // insereaza la p, sirul s

O funcţie care extrage un subşir de lungime dată dintr-o poziţie dată a unui şir:

char **\*substr**(char \*src, int n, char \*dest); // extrage de la pozitia src in dest n caractere

Funcţia are ca rezultat adresa şirului extras.

Pentru a extrage 5 caractere începând cu poziţia 3 (al 4-lea caracter) din şirul s în şirul p va trebui să apelaţi cu parametrii: substr(s + 3, 5, p).

O funcţie de comparare a două şiruri de cifre zecimale ce reprezintă numere naturale (fără semn), cu rezultat -1, 0 sau 1:

int **strdcmp**(char \*s1, char \*s2);

Funcţie care extrage următorul cuvânt format numai din litere mici începând de la o adresă dată. Funcţia are ca rezultat adresa imediat următoare cuvântului extras (în al doilea argument al funcţiei):

char **\*next** (char \*from, char \*word);

**strrev(char\* s)** Inversează şirul.

char\* strstr(char \*s,char \*ss); // returneaza pozitia primei aparitii a sirului ss in sirul s, respectiv NULL daca ss nu e in s

Exemplul: Declaraţii de variabile, initializari, expresie şi obţinerea valorilor prin *funcţii predefinite*

| Declaratii si initializari |

-----------------------------------------------------------

| **char s1[]** = "tara noastra frumoasa si bogata", |

| **s2[]** = "facultatea de informatica"; |

-----------------------------------------------------------

| Expresie | Valoare |

-----------------------------------------------------------

| **strlen(s1)**  | 31 |

| **strlen(s2 + 8)** | 17 |

| **strcmp(s1, s2)**  | număr pozitiv |

-----------------------------------------------------------

| Instructiune | Ce se va tipari ? |

-----------------------------------------------------------

| printf("%s", s1 + 13); | frumoasa si bogata |

| **strcpy(s2 + 11, s1 + 25);**| |

**| strcat(s2, "\n");**  | |

| printf("%s", s2); | facultatea bogata |

-----------------------------------------------------------

sa consideram funcţia squeeze(s,c) care elimina toate apariţiile lui c din şirul s:

* **squeeze (s,c) /\*** sterge toate aparitiile lui c din s **\*/**
* **char s[]; int c; {int i, j; for (i = j = 0; s[i] != '\0'; i++) if (s[i] != c) s[j++] = s[i]; s[j] = '\0';}**

De fiecare data când apare un caracter non-c el este copiat in pozitia j curenta si numai după aceea j este incrementat pentru a fi gata pentru urmatorul caracter. Aceasta construcţie este echivalenta cu urmatoarea: if (s[i] != c) {s[j] = s[i]; j++; }

Un alt exemplu de construcţie similara este luata din funcţia getline, in care putem inlocui

if (c == '\n' {s[i]=c; ++i; }

cu mult mai compacta construcţie: **if (c == '\n') s[i++] = c;**

Test55 Analizaţi construcţia şi rulaţi programul pentru determinarea rezultatelor

#include <stdio.h>

#include <string.h>

#include <conio.h>

#include <ctype.h>

#define LUNG 80

#define NR 15

char tab\_cuv[NR][LUNG];   
int nr\_cuv=0;  /\* numarul cuvintelor introduse \*/

void citire\_cuv(void){   
  printf("\*\*\* Se introduc maxim %d cuvinte, terminate cu CTRL/Z:\n",NR);   
  while(nr\_cuv<NR && gets(tab\_cuv[nr\_cuv]))nr\_cuv++;    
       /\* la CTRL/Z gets returneaza NULL (= 0) \*/   
  /\* citirea se poate face si cu scanf:   
  while(nr\_cuv<NR && scanf("%s",tab\_cuv[nr\_cuv])!=EOF)nr\_cuv++; \*/   
  /\* daca terminarea se face cu un cuvant vid:   
  while(nr\_cuv<NR && strcmp("",gets(tab\_cuv[nr\_cuv])))nr\_cuv++; \*/   
}

void constr\_cuv(void){   
  char c,cuv[NR+1]; /\* lungimea maxima este egala cu numarul maxim de cuvinte + 1 pentru terminator \*/   
  int i,j; /\* i parcurge cuvintele din tab\_cuv, j pozitiile din cuv \*/   
  for(i=j=0;i<nr\_cuv;i++)   
    if(isalpha(c=tolower(tab\_cuv[i][strlen(tab\_cuv[i])-1])))   
      cuv[j++]=c;   
  cuv[j]='\0';   
  printf("\*\*\* Cuvantul construit:|%s|\n",cuv);   
}

void cuv\_max(void){   
  int i,lung\_crt,lung\_max=0;    
  char \* p\_max; /\* pointerul spre cuvantul maxim \*/   
    /\* se poate memora indicele cuvantului maxim: int i\_max;   
       sau memora cuvantul maxim intr-un sir: char c\_max[LUNG]; \*/   
  for(i=0;i<nr\_cuv;i++)   
    if((lung\_crt=strlen(tab\_cuv[i]))>lung\_max){   
      p\_max=tab\_cuv[i];   
      lung\_max=lung\_crt;   
    }   
  printf("\*\*\* Cuvantul de lungime maxima %d este:|%s|\n",lung\_max,p\_max);   
}

void cuv\_vocale(void){   
  int i;   
  puts("\*\*\* Cuvintele ce incep cu vocale:");   
  for(i=0;i<nr\_cuv;i++)   
    switch(toupper(tab\_cuv[i][0])){   
     case 'A': case'E': case 'I': case 'O': case 'U':puts(tab\_cuv[i]);   
   }  /\* in loc de switch se putea folosi   
  char c; if(c=toupper(tab\_cuv[i][0]),c=='A' ||   
     c=='E' || ...)puts(tab\_cuv[i]); \*/   
}

void main(void){

  citire\_cuv();   
  cuv\_max();   
  constr\_cuv();   
  cuv\_vocale();

  getch();   
}

Test56 Analizaţi construcţia şi rulaţi programul pentru determinarea rezultatelor

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

#define LUNGS 80

void main(void){   
  char s1[LUNGS],s2[LUNGS],s3[LUNGS],rezultat[LUNGS];   
  char \*s1ptr=s1,\*s2pos, \*rezptr=rezultat;

  puts("\*\*\* sirul s1:"); gets(s1);   
  puts("\*\*\* subsirul s2:"); gets(s2);   
  puts("\*\*\* s3:"); gets(s3);

  while(s2pos=strstr(s1ptr,s2)){    
    while(s1ptr<s2pos)\*rezptr++=\*s1ptr++;   
    strcpy(rezptr,s3);   
    rezptr+=strlen(s3);   
    s1ptr+=strlen(s2);   
  }   
  strcpy(rezptr,s1ptr);   
  puts("\*\*\* sirul rezultat:"); puts(rezultat);

  getch();   
}

Test57 Analizaţi construcţia şi rulaţi programul pentru determinarea rezultatelor. FUNCTII UTILIZATE PENTRU CONVERSIA VALORILOR NUMERICE IN SIR. La utilizarea acestor functii se introduce directiva #include<stdlib.h>

a) *Functia atof* - converteste un sir catre tipul double

b) *Functia atold* - converteste un sir catre tipul long double

Exemplu:

#include <stdlib.h>

#include <iostream.h>

#include<conio.h>

void main() {

float f;

char \*str = "12345.67";

f = atof(str);

cout<<"string = "<<str<<endl<<"float = "<< f;

getch(); }

c) *Functia atoi* - converteste un sir catre tipul int

d) *Functia atol* - converteste un sir catre tipul long

e) *Functia ecvt* - converteste o valoare dubla catre un sir

f) *Functia itoa* - converteste o valoare de tip intreg catre un sir

g) *Functia ltoa* - converteste o valoare de tip long int catre un sir

ATOI(S) /\* CONVERT S TO INTEGER \*/

CHAR S[];

{

INT I, N, SIGN;

FOR(I=0;S[I]==' ' \!\! S[I]=='\N' \!\! S[I]=='\T';I++); /\* SKIP WHITE SPACE \*/

SIGN = 1;

IF(S[I] == '+' \!\! S[I] == '-') /\* SIGN \*/

SIGN = (S[I++]=='+') ? 1 : - 1;

FOR( N = 0; S[I] >= '0' && S[I] <= '9'; I++)

N = 10 \* N + S[I] - '0';

RETURN(SIGN \* N);

}

REVERSE(S) /\* REVERSE STRING S IN PLACE \*/

CHAR S[];

{

INT C, I, J;

FOR(I = 0, J = STRLEN(S) - 1; I < J; I++, J--) {

C = S[I];

S[I] = S[J];

S[J] = C; }

}

ITOA(N,S) /\*CONVERT N TO CHARACTERS IN S \*/

CHAR S[];

INT N;

{

INT I, SIGN;

IF ((SIGN = N) < 0) /\* RECORD SIGN \*/

N = -N; /\* MAKE N POSITIVE \*/

I = 0;

DO { /\* GENERATE DIGITS IN REVERSE ORDER \*/

S[I++] = N % 10 + '0';/\* GET NEXT DIGIT \*/

} WHILE ((N /=10) > 0); /\* DELETE IT \*/

IF (SIGN < 0)

S[I++] = '-'

S[I] = '\0';

REVERSE(S);

}

DOUBLE ATOF(S) /\* CONVERT STRING S TO DOUBLE \*/

CHAR S[];

{

DOUBLE VAL, POWER;

INT I, SIGN;

FOR(I=0; S[I]==' ' \!\! S[I]=='\N' \!\! S[I]=='\T'; I++)

; /\* SKIP WHITE SPACE \*/

SIGN = 1;

IF (S[I] == '+' \!\! S[I] == '-') /\* SIGN \*/

SIGN = (S[I++] == '+') ? 1 : -1;

FOR (VAL = 0; S[I] >= '0' && S[I] <= '9'; I++)

VAL = 10 \* VAL + S[I] - '0';

IF (S[I] == '.')

I++;

FOR (POWER = 1; S[I] >= '0' && S[I] <= '9'; I++) {

VAL = 10 \* VAL + S[I] - '0';

POWER \*= 10;

}

RETURN(SIGN \* VAL / POWER);

}

GETOP(S, LIM) /\* GET NEXT OPRERATOR OR OPERAND \*/

CHAR S[];

INT LIM;

{

INT I, C;

WHILE((C=GETCH())==' '\!\! C=='\T' \!\! C=='\N');

IF (C != '.' && (C < '0' \!\! C > '9'))

RETURN(C);

S[0] = C;

FOR(I=1; (C=GETCHAR()) >='0' && C <= '9'; I++)

IF (I < LIM)

S[I] = C;

IF (C == '.') { /\* COLLECT FRACTION \*/

IF (I < LIM)

S[I] = C;

FOR(I++;(C=GETCHAR()) >='0' && C<='9';I++)

IF (I < LIM)

S[I] =C;

}

IF (I < LIM) { /\* NUMBER IS OK \*/

UNGETCH(C);

S[I] = '\0';

RETURN (NUMBER);

} ELSE { /\* IT'S TOO BIG; SKIP REST OF LINE \*/

WHILE (C != '\N' && C != EOF)

C = GETCHAR();

S[LIM-1] = '\0';

RETURN (TOOBIG);

}

}

##### DIRECTIVE. MACROINSTRUCŢIUNI

Limbajul C asigură anumite facilităţi prin intermediul unui preprocesor (preprocesarea este o fază iniţială, separată a compilării). Comenzile către preprocesor sunt numite directive. Directivele încep cu simbolul #. Iată principalele directive de interes:

**a) Includerea în textul sursă a unui fişier (#include)**

Are două forme:

**#include "nume"** în care se consideră fişierul **nume** în acelaşi director cu fişierul în care apare directiva include şi

**#include <nume>** în care se consideră fişierul într-un director predefinit, de obicei borlandc\include sau tc\include. Directiva se foloseşte frecvent pentru includerea fişierelor header, cum ar fi: **stdio.h, string.h,** etc. (predefinite), dar şi a unor fişiere header utilizator, care pot conţine declaraţii de variabile şi funcţii externe, tipuri definite de utilizator, etc.

b) Substituţia unui text cu alt text (#define)

Are forma generală:

#define nume text efectul fiind că se va substitui peste tot **nume** cu **text,** de exemplu:

#define EOF -1

#define NULL 0

Directiva **#define** poate avea şi parametri, caz în care se definesc macroinstrucţiuni, similare cu MACRO din limbaje de asamblare, de exemplu:

#define max(A,B) ( ( (A) > (B) ) ? (A) : (B) )

Putem scrie acum x = max(a+b/a-b); care se va substitui în: **x = ( ( (a+b) > (a-b) ) ? (a+b) : (a-b) );**

Acesta este un mod avantajos de a defini unele operaţii, indiferent de tipul operanzilor. Practic, max(a,b) este o macroinstrucţiune; ea se poate folosi cu orice fel de operanzi (int, **float, double).** Dacă s-ar fi scris o funcţie, ar fi trebuit precizat, de fiecare dată tipul:

**int imax(int, int);**

**float fmax (float, float);**

Macroinstrucţiunile de acest gen trebuie folosite cu atenţie: în definirea lui max, se observă că sau a sau b se evaluează de două ori: o dată în condiţie, şi a doua oară în expresiile propriu-zise. O construcţie de forma:

**x = max(i++,j++) ;** va incrementa de două ori cea mai mare valoare dintre i şi j, datorită expandării de forma: **x = ( ( (i++) > (j++) ) ? (i++) : (j++) );**

Trebuie acordată atenţie şi definirii propriu-zise. Se recomandă să se pună paranteze la fiecare subexpresie. De exemplu, dacă se defineşte:

**#define squaze(x) x \* x** şi se consideră apelul de macro:

**t = squaze(a+1);** acesta se va expanda în:

**t = a+1\*a+1;** ceea ce este diferit de (a+1)\*(a+1). Corect este:

**#define square(x) ((x)\*(x))**

Directivele nu sunt instrucţiuni deci ele nu se termină cu ;.

Cu toate aceste limitări, macroinstrucţiunile sunt utile. Unele din funcţiile de bibliotecă sunt implementate ca macroinstrucţiuni **(#define** cu parametri). Când folosim aceste funcţii nu ştim dacă sunt funcţii propriu-zise sau macroinstrucţiuni.

Macrodefiniţiile pot schimba chiar cuvinte cheie ale limbajului dar acest lucru nu este recomandabil.

Iată două exemple foarte utile de macroinstrucţiuni:

**#define NREL(array) (sizeof(array)/(sizeof(array[0]))**

**#define SIZE(array) (sizeof(array[0]))**

Ele vor da numărul de elemente şi, respectiv, dimensiunea unui element al unui tablou, indiferent de tipul de bază al tabloului.

Există şi alte directive (de compilare condiţionată, etc.), multe din ele fiind specifice diverselor implementări (vezi anexa care conţine manualul de referinţă).

**Macro**-uri în C. În C se mai pune la dispozitie fisierul header care conţine o multime de macro-uri (definitii) folosite pentru testarea caracterelor si o multime de prototipuri de Funcţii ce sunt folosite pentru conversia caracterelor. In tabelul de mai jos prezentam o lista de macro-uri folosite la testarea caracterelor. Aceste macro-uri iau ca argument o variabila de tip int si returneaza o valoare de tip int (zero=false, diferit de zero=true).

### Biblioteca <ctype.h>

**int isprint(int c)** – verifică dacă caracterul c este printabil.

------------------------------------------------------------------------

| Macro | Se returneaza true (diferit de zero) daca |

------------------------------------------------------------------------

**isalpha(c)** c este litera

**isupper(c)**  c este litera majuscula

**islower(c)** c este litera mica

**isdigit(c)**  c este cifra

**isalnum(c)** c este litera sau cifra

**isxdigit(c)** c este cifra hexazecimala

**isspace(c)** c este caracter spatiu

**ispunct(c)**  c este semn de punctuatie

**isprint(c)**  c este caracter tiparibil

**isgraph(c)** c este tiparibil, dar diferit de spatiu

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

**isascii(c)** c este cod ASCII

--------------------------------------------------------------------

**toupper(c)** schimba c din litera mica in majuscula

**tolower(c)** schimba c din majuscula in litera mica

**toascii(c)** schimba c cu codul ASCII

Exemplu: Funcţia lower converteşte literele mari din setul de caractere ASCII în litere mici. Dacă lower primeşte un caracter care nu este o literă mare atunci îl returnează neschimbat.

lower(int c) {

if (c>='A' && c<='Z') return c + 'a' - 'A';

else return c;

}

**Exemple pentru analiză:**

În scopul scrierii acestui program vom scrie trei funcţii. Funcţia getline citeşte o linie din textul sursă şi returnează lungimea ei. Această funcţie are două argumente. Argumentul s indică numele masivului unde se va citi linia, iar argumentul lim reprezintă lungimea maximă a unei linii care poate fi citită.

Funcţia index caută configuraţia de caractere dorită în interiorul liniei şi furnizează poziţia sau indexul în şirul s, unde începe şirul t, sau 1 dacă linia nu conţine şirul t. Argumentele funcţiei sînt s care reprezintă adresa şirului de studiat şi t care reprezintă configuraţia căutată.

## *Funcţia care realizează afişarea este printf din bibliotecă.*

Programul care realizează structura de mai sus este:

#define MAXLINE 1000

#define EOF -1

getline(char s[], int lim) {

/\* citeşte o linie în s; returnează lungimea \*/

int c, i;

i = 0;

while (--lim>0 && (c=getchar())!= EOF &&

c!='\n')

s[i++]=c;

if (c=='\n') s[i++]=c;

s[i]='\0';

return i;

}

**index(char s[], char t[]) {**

/\* returnează poziţia din şirul s unde începe şirul t, sau 1 \*/

int i,j,k;

for (i=0; s[i]!='\0'; i++) {

for (j=i, k=0; t[k]!='\0' &&

s[j]==t[k]; j++, k++)

;

if (t[k]=='\0')

return i;

}

return -1;

}

main() {

/\* imprimă toate liniile care conţin cuvîntul „*the*” \*/

char line [MAXLINE];

while (getline(line, MAXLINE)>0)

if (index(line,"the")>=0)

printf("%s",line);

}

2. Funcţia atof converteşte un şir de cifre din formatul ASCII într-un număr flotant în precizie dublă.

double atof(char s[]) { /\* converteşte şirul s în dublă precizie \*/

double val, power;

int i, sign;

for (i=0; s[i]==' ' || s[i]=='\n' ||

s[i]=='\t'; i++) ; /\* sare peste spaţii albe \*/

sign = 1;

if (s[i]=='+' || s[i]=='-')

sign = (s[i++]=='+') ? 1 : -1;

for (val=0; s[i]>='0' && s[i]<='9'; i++)

val = 10 \* val + s[i] - '0';

if (s[i]== '.') /\* punct zecimal \*/

i++;

for (power=1; s[i]>='0' && s[i]<='9'; i++)

{ val = 10 \* val +s[i] - '0':

power \*= 10;

}

return sign \* val / power;

}

3. Funcţia atoi converteşte un şir de cifre în echivalentul lor numeric întreg.

atoi(char s[]) { /\* converteşte şirul s la un întreg \*/

int i,n;

n = 0;

for (i=0; s[i]>='0' && s[i]<='9'; i++)

n = 10 \* n +s[i] - '0';

return n;

}

Urmatoarea functie converteste literele majuscule ale codului ASCI extins , in litere minuscule :

char lower ( char c)

{ if ( c > = `A` && c < ` Z ` ) return c l 0 x 20 ;

if ( c = = 161 ll c = = 164 ll c = = 166 ll c = = 171 )

return c + 1 ; / \* Majusculele T , A, A si I \* /

if ( c = = 163) / \* Majuscula S \* /

return 160 ;

return c ; }

*Funcţie care converteşte un şir de caractere reprezentând un număr întreg, într-o valoare întreagă. Numărul poate avea semn şi poate fi precedat de spaţii albe.*

**#include <ctype.h>**

**int atoi(char \*s)**

**{ int i, nr, semn;**

**for(i=0; isspace(s[i]); i++); /\*ignora spatii albe\*/**

**semn=(s[i]==-1)?-1:1; /\*stabilire semn\*/**

**if(s[i]==’+’||s[i]==’-‘) i++; /\*se sare semnul\*/**

**for(nr=0;isdigit(s[i]); i++) /\*conversie in cifra\*/**

**nr=10\*nr+(s[i]-‘0’); /\*si alipire la numar\*/**

**return semn\*nr; }**

*Funcţie care converteşte un întreg într-un şir de caractere în baza 10.*

Algoritmul cuprinde următorii paşi:

**{ se extrage semnul numărului;**

**se extrag cifrele numărului, începând cu cmps;**

**se transformă în caractere şi se depun într-un tablou;**

**se adaugă semnul şi terminatorul de şir;**

**se inversează şirul; }**

**void inversare(char[]);**

**void itoa(int n, char s[]){**

**int j, semn;**

**if((semn=n)<0) n=-n; j=0;**

**do s[j++]=n%10+’0’; while ((n/=10)>0);**

**if(semn<0) s[j++]=’-‘; s[j]=’\0’; inversare(s); }**

**void inversare(char s[]) { int i,j; char c;**

**for(i=0,j=strlen(s)-1;i<j;i++,j--) c=s[i], s[i]=s[j], s[j]=c; }**

**char str1[5];**

**char str2[5];**

**char str3[5];**

**itoa(12, str1, 10); //str1=”12”**

**itoa(12, str1, 16); //str1=”C”**

**itoa(12, str1, 2); //str1=”1100”**

*Funcţie care converteşte un întreg fără semn într-un şir de caractere în baza 16.*

Pentru a trece cu uşurinţă de la valorile cifrelor hexazecimale **0,1,…15** la caracterele corespunzătoare; **‘0’,’1’,…,’a’,…,’f’**, vom utiliza un tablou iniţializat de caractere.

**static char hexa[]=”0123456789abcdef”;**

**void itoh(int n, char s[]) { j=0;**

**do { s[j++]=hexa[n%16]; while ((n/=16)>0); s[j]=’\0’; inversare(s); }**

Este scris 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
* 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.

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

#define LUNG 81  // lungime maxima cuvant   
#define NR 15 // nr max de cuvinte citite

char tab\_cuv[NR][LUNG];   
int nr\_cuv=0;  // numarul cuvintelor introduse

void citire\_cuv(void){   
  printf("\*\*\* Se introduc maxim %d cuvinte, terminate cu CTRL/Z:\n",NR);   
  while(nr\_cuv<NR && gets(tab\_cuv[nr\_cuv]))nr\_cuv++;    
      /\* la CTRL/Z gets returneaza NULL (= 0) \*/   
  /\* citirea se poate face si cu scanf:   
  while(nr\_cuv<NR && scanf("%s",tab\_cuv[nr\_cuv])!=EOF)nr\_cuv++; \*/   
  /\* daca terminarea se face cu un cuvant vid:   
  while(nr\_cuv<NR && strcmp("",gets(tab\_cuv[nr\_cuv])))nr\_cuv++; \*/   
}

/\* Varianta: tab\_cuv se declara ca tablou de pointeri, se modifica declaratia lui tab\_cuv si functia citire\_cuv, restul functiilor raman nemodificate;

#include <alloc.h>   
char\* tab\_cuv[NR];   
void citire\_cuv(void){   
  char cuv\_crt[LUNG];   
  printf("\*\*\* Se introduc maxim %d cuvinte, terminate cu CTRL/Z:\n",NR);   
  while(nr\_cuv<NR && gets(cuv\_crt)){   
    if((tab\_cuv[nr\_cuv]=(char \*)malloc(strlen(cuv\_crt)+1))==NULL){   
      puts("\*\*\* Spatiu insuficient ! \*\*\*");   
      exit(1);   
    }   
    strcpy(tab\_cuv[nr\_cuv],cuv\_crt);   
    nr\_cuv++;    
  }   
}   
\*/

void constr\_cuv(void){   
  char c,cuv[NR+1]; /\* lungimea maxima este egala cu numarul maxim de cuvinte + 1 pentru terminator \*/   
  int i,j,lung; /\* i parcurge cuvintele din tab\_cuv, j pozitiile din cuv, lung e lungimea cuv curent \*/   
  for(i=j=0;i<nr\_cuv;i++)   
    if((lung=strlen(tab\_cuv[i]))>0 && isalpha(c=tolower(tab\_cuv[i][lung-1])))   
      cuv[j++]=c;   
  cuv[j]='\0';   
  printf("\*\*\* Cuvantul construit:|%s|\n",cuv);   
}

void cuv\_max(void){   
  int i,lung\_crt,lung\_max=0;    
  char \* p\_max; /\* pointerul spre cuvantul maxim \*/   
    /\* se poate memora indicele cuvantului maxim: int i\_max;   
       sau memora cuvantul maxim intr-un sir: char c\_max[LUNG]; \*/   
  for(i=0;i<nr\_cuv;i++)   
    if((lung\_crt=strlen(tab\_cuv[i]))>lung\_max){   
      p\_max=tab\_cuv[i];   
      lung\_max=lung\_crt;   
    }   
  printf("\*\*\* Cuvantul de lungime maxima %d este:|%s|\n",lung\_max,p\_max);   
}

void cuv\_vocale(void){   
  int i;   
  puts("\*\*\* Cuvintele ce incep cu vocale:");   
  for(i=0;i<nr\_cuv;i++)   
    switch(toupper(tab\_cuv[i][0])){   
     case 'A': case'E': case 'I': case 'O': case 'U':puts(tab\_cuv[i]);   
   }  /\* in loc de switch se putea folosi   
  char c; if(c=toupper(tab\_cuv[i][0]),c=='A' ||   
     c=='E' || ...)puts(tab\_cuv[i]); \*/   
}

void main(void){

  citire\_cuv();   
  cuv\_max();   
  constr\_cuv();   
  cuv\_vocale();

  getch();   
}

// 2

#include <stdio.h>

#include <string.h>

int strlen2 (char s[]);

main() { int c, i; char s[100];

i = 0;

while ((c = getchar()) != '**\n**') { s[i] = c; ++i; }

  i = strlen(s); printf("%d**\n**", i);

i = strlen2(s);

printf("%d**\n**", i); }

 int strlen2 (char s[]) { int i;

  i = 0;

while (s[i] != '**\0**') ++i; return i; }

// 3

# include <stdio.h>

# include <string.h>

char\* strdel(char\* p, int n) {   //caz 1: nu exista mai mult de n caractere incepand de la pozitia p

        if (strlen(p) <= n) {      \*p = '\0';                return p;           }

        //in caz contrar, facem stergerea copiind caracterele de la p+n peste cele de la p  
        strcpy(p, p + n);        return p; }

char\* strins(char\* p, char \*s)

{        //pas 1: copiem sirul din p intr-o alta locatie

        char \*tail\_p = strdup(p);        //pas 2: copiem la pozitia p noul sir  
        strcpy(p, s);        //pas 3: copiem la sfarsit caracterele care existau inainte in p

        strcat(p, tail\_p);

        return p; }

int main() {         char s[100],replacethis[100],withthis[100];

        gets(s);         scanf("%s %s", replacethis, withthis);

        char \*p;        // pornim cu p fiind prima aparitie a sirului <replacethis> in sirul <s>

        // atata timp cat mai exista aparitii, facem inlocuirea si setam "p" ca fiind adresa urmatoarei

        // aparitii a sirului <replacethis> in <s>

        //

        // ATENTIE! este foarte important sa dam p=strstr(p+strlen(withthis),...) pentru ca in caz contrar programul ar cicla la infinit daca sirul <replacethis> se regaseste in <withthis>

        // Cu alte cuvinte, am inlocui in inlocuire, in locuire, in inlocuire...

        for (p = strstr(s, replacethis); p != NULL; p = strstr(p + strlen(withthis), replacethis)){

                strdel(p, strlen(replacethis));      strins(p, withthis);         }

        printf("%s\n", s);              return 0; }

// 4

**void main(void) { char s1[80],s2[80]; gets(s1);gets(s2);**

**printf('lungimea:%d%d',strlen(s1),strlen(s2));**

**if(!strcmp(s1,s2)) printf('Sirurile sunt egale'); strcat(s1,s2);printf('%d',s1);**

**strcpy(s1,'Acesta este un test.');printf(s1);**

**if(strchr('salut','u')) printf('u este in salut')**

**strstr('ce mai faci','ce') printf('am gasit ce');**

// 4 In mod probabil, cel mai comun tip de tablouri in limbajul C este tabloul de caractere. Pentru a ilustra folosirea tablourilor de caractere si a functiilor care le manipuleaza, vom scrie un program care citeste un set de linii si o tipareste pe cea mai lunga. Schita lui este destul de simpla:

while (mai exista o alta linie)  
if (este mai lunga decit linia anterioara) salveaza-o pe ea si lungimea ei tipareste linia cea mai lunga

Aceasta schita ne arata clar ca programul se imparte in bucati. O bucata citeste o linie noua, o alta bucata o testeaza, o alta o salveaza iar restul controleaza procesul. Deoarece lucrurile se impart aşa de frumos, ar fi mai bine sa le scriem la fel. Pentru aceeasta, vom scrie la inceput o functie getline care va citi urmatoarea linie de la intrare; ea este generalizare a functiei getchar. Pentru a face functia utila si in alte contexte, vom incerca sa o scriem cit mai flexibil. In mod minim, getline va trebui sa returneze un semnal despre posibilul sfirsit de fisier; proiectind-o mai general,ea va trebui sa returneze lungimea liniei sau zero daca se intilneste sfirsitul de fisier. Zero nu este niciodata o lungime valida de linie, deoarece orice linie are cel putin un caracter, chiar si o linie ce contine numai caracterul "linie noua" are lungimea 1.

Cind gasim o linie care este mai lunga decit linia cea mai lunga gasita anterior, trebuie sa o salvam undeva. ceasta sugereaza o a doua functie, copy, pentru a salva noua linie intr-un loc sigur. In final, avem nevoie de un program principal care sa controleze functiile getline si copy. Iata rezulatul:

#define MAXLINE 1000 /\* lungimea maxima a liniei \*/  
main() /\* gaseste linia cea mai lunga \*/

{ int len; /\* lungimea liniei curente \*/ int max; /\* lungimea maxima gasita pina acum \*/  
char line[MAXLINE]; /\* linia curenta introdusa \*/ char save[MAXLINE]; /\* cea mai lunga linie salvata \*/  
max = 0; while ((len = getline(line, MAXLINE)) > 0) if (len > max) {max = len; copy(line, save); }

if (max > 0) /\* s-a citit cel putin o linie \*/ printf("%s", save); }

getline (s, lim) /\* citeste linia in s, returneaza lungimea \*/  
char s[]; int lim;

{ int c, i; for(i = 0; i < lim - 1 && (c=getchar())!=EOF && c!='\n';++i) s[i] = c;

if (c == '\n') { s[i] = c; ++i; } s[i] = '\0'; return(i); }

copy(s1, s2) /\* copiaza pe s1 in s2; s2 suficient de mare \*/  
char s1[], s2[];  
 { int i; i = 0; while ((s2[i] = s1[i]) != '\0') ++i; }

main si getline comunica intre ele printr-o pereche de argumente si o valoare returnata. In getline, argumumentele sint declarate prin liniile:

char s[];  
int lim;

care spun ca primul argument este un tablou iar al doilea un intreg. Lungimea tabloului s nu este specificata in getline deoarece ea este determinata in main. "getline" foloseste instructiunea return pentru a trimite o valoare inapoi apelantului, la fel cum facea si functia power. Unele functii returneaza o valoare utila; altele, de exemplu copy, sint folosite numai pentru efectul lor si nu returneaza nici o valoare.

getline pune caracterul \0 (caracterul nul, a carui valoare este zero) la sfirsitul tabloului pe care il creaza, pentru a marca sfirsitul sirului de caractere. Aceasta conventie este folosita de asemenea si de catre compilatorul C; cind o constanta sir de tipul "hello\n" este scrisa intr-un program C, compilatorul isi creaza un tablou de caractere continind carcterele sirului si terminat cu \0, astfel incit o functie, de exemplu printf, poate sa-i determine sfirsitul.

-------------------------------  
| h | e | l | l | o | \n | \0 |  
-------------------------------

Specificatorul de format %s din printf se asteapta la un sir reprezentat tocmai in aceasta forma. Daca examinati functia copy, veti descoperi ca si ea se bizuie de fapt pe terminarea argumentului sau de intrare s1 cu un \0 si ea copiaza acest caracter in argumentul de iesire s2. (Toate acestea presupun ca \0 nu este parte a unui text normal).

Este demn de mentionat in trecere ca, un program, chiar si atit de mic ca acesta, prezinta unele probleme delicate de proiectare. De exemplu, ce ar face main daca ar intilni o linie mai mare decit limita sa? getline lucreaza bine, adica se va opri atunci cind tabloul este plin chiar daca nu a intilnit nici un caracter "linie noua". Testind lungimea si ultimul caracter returnat, main poate determina cind a fost linia prea lunga si apoi sa actioneze cum vrea. Pentru a scurta programul, am ignorat acest aspect.

Nu exista vre-o cale pentru utilizatorul lui getchar de a sti inainte cit va fi de lunga o linie de intrare, aşa ca getline verifica daca nu s-a produs o depasire. Pe de alta parte, utilizatorul lui copy stie intodeauna (sau poate descoperi) cit este de mare sirul, aşa ca nu trebuie sa adaugam la functie o verificare de erori.

**Exercitiul 1.14.** Revizuiti rutina main din programul precedent astfel incit ea sa tipareasca corect lungimea unei linii de intrare de o lungime arbitrara, si atita text cit este posibil de tiparit.

**Exercitiul 1.15.** Scrieti un program care sa tipareasca toate liniile mai lungi de 80 de caractere.

Dam de asemenea, o noua versiune getline;

#define MAXLINE 1000  
main() /\* gasiti toate liniile ce contin un model dat \*/  
{ char line[MAXLINE];  
while (getline(line, MAXLINE) > 0) if (index(line, "the") >= 0) printf("%s", line); }

getline(s, lim) /\* citeste linia in s, returneaza lungimea ei \*/

char s[]; int lim;  
{int c, i;  
i = 0; while (--lim > 0 && (c = getchar()) != EOF && c!= '\n') s[i++] = c;

if (c == '\n') s[i++] = c; s[i] = '\0'; return(i); }

index(s, t) /\* returneaza indexul lui t in s, -1 in lipsa \*/  
char s[], t[];  
{int i, j, k;  
for (i = 0; s[i] != '0'; i++) { for (j = i, k = 0; t[k] != '\0' && s[j] == t[k]; j++, k++);

if (t[k] == '\0') return(i); } return(-1); }

Fiecare functie are forma   
nume (lista de argumente, daca exista)  
declaratii de argumente, daca exista  
{declaratii si instructiuni, daca exista }

Asa cum am sugerat, anumite parti pot sa lipseasca; functia minima este: dummy() {} care nu face nimic. (O functie care nu face nimic este utila uneori ca loc pastrat pentru dezvoltari ulterioare in program). Numele functiei poate fi deasemenea precedat de un tip daca functia returneaza altceva decit o valoare intreaga; acesta este subiectul urmatorului capitol.

Un program este tocmai un set de definitii de functii individuale. Comunicarea intre functii este (in acest caz) facuta prin argumente si valori returnate de functii; ea poate fi facuta deasemenea, prin variabile externe. Functiile pot apare in orice ordine in fisierul sursa, si programul sursa poate fi spart in mai multe fisiere, pe cind o functie nu este sparta. Instructiunea return este mecanismul de returnare a unei valori din functia apelata in apelant. Orice expresie poate urma dupa instructiunea return: return (expresie)

Functia apelanta este libera sa ignore valoarea returnata daca doreste. Mai mult, nu e necesar sa existe nici o expresie dupa return; in acest caz, nici o valoare nu este returnata apelantului. Controlul este deasemenea returnat apelantului fara nici o valoare atunci cind executia "se continua dupa sfirsitul" functiei, atingind cea mai din dreapta paranteza. Nu este ilegal ci probabil un semn de necaz (deranj), daca o functie returneaza o valoar dintr-un loc si nici o valoare din altul. In orice caz "valoarea" unei functii care nu returneaza nici una, este sigur un gunoi. Verificatorul "lint" cauta si dupa astfel de erori.

Mecanismul prin care se compileaza si se incarca un program care rezida in mai multe fisiere sursa variaza de la un sistem la altul . Pe sistemul UNIX, de exemplu, comanda CC, mentionata in Capitolul 1, face lucrul acesta. Sa presupunem ca cele trei functii se gasesc in trei fisiere numite main.c, getline.c si index.c. Atunci comanda:

CC main,c,getline,c,index,c

compileaza cele trei fisiere, plaseaza codul obiect relocabil rezultat in fisierele main.o, getline.o si index.o si le incarca pe toate intr-un fisier executabil numit a.out.

Daca exista vreo eroare, sa spunem in main.c, fisierul poate fi recompilat singur si rezultatul incarcat cu fisierele obiect anterioare, cu comanda:  
CC main.c getline.o index.o  
Comanda CC foloseste conventia de notare ".c" spre deosebire de ".o" pentru a distinge fisierele sursa de fisierul obiect.

Determinaţi ce efectuează următorul program:

#include <stdio.h>

#include <string.h>

#define MAXLINES 5000 /\* \*/

char \*lineptr[MAXLINES]; /\* \*/

int readlines(char \*lineptr[], int nlines);

void writelines(char \*lineptr[], int nlines);

void qsort(char \*lineptr[], int left, int right); /\* : \*/

main() { int nlines; /\* \*/

if ((nlines = readlines(lineptr, MAXLINES)) >= 0) { qsort(lineptr, 0, nlines-1);

writelines(lineptr, nlines); return 0; } else { printf("\n"); return 1; } }

#define MAXLEN 1000 /\* \*/

int getline(char \*, int); char \*alloc(int); /\* readlines: citirea sirurilor: \*/

int readlines(char \*lineptr[], int maxlines) { int len, nlines; char \*p, line[MAXLEN]; nlines = 0;

while ((len = getline(line, MAXLEN)) > 0) if (nlines >= maxlines || (p = alloc(len)) == NULL)

return -1; else { line[len-1] = '\0'; /\* eliminăm cimbolul \n \*/

strcpy(p, line); lineptr[nlines++] = p; } return nlines; }

void writelines(char \*lineptr[], int nlines) /\* writelines: \*/ { int i;

for (i = 0; i < nlines; i++) printf("%s\n", lineptr[i]); }

**Exercitii propuse spre implementare**

* + - 1. Folosind functiile "getchar()" si "putchar()", sa se scrie un program C care transforma literele mici in litere mari. Incercati si o varianta de program care foloseste functiile "islower()" si "toupper()".
      2. Utilizand functiile "getchar()" si "putchar()" creati un program C care sa copie un fisier in alt fisier (comanda voastra proprie de copiere). Utilizati redirectarea ! De asemeni, precizati si cazul cand dorim sa copiem un fisier la sfarsitul unui fisier existent.
      3. Scrieti in C un analizor lexical care sa recunoasca cat mai multi atomi lexicali din C. De exemplu, cuvintele rezervate (while, do, for, ...), identificatori, operatori (relationali, logici, artimetici, ...) si eventual alte structuri. Apoi, tipariti acelasi fisier de intrare cu exceptia spatiilor multiple si a comentariilor.
      4. Scrieti un program C care citeste caractere de la tastatura si le scrie la ecran. Scrieti toate vocalele cu litere majuscule si consoanele cu litere mici. Apoi, scrieti un program C care citeste caractere de la tastatura si sterge vocalele din ele (afisand doar consoanele). (Acest mod de scriere era folosit in scrisul hieroglific al Greciei Antice).

1. Pentru propoziţia dată calculaţi numărul de vocale şi frecvenţa repetării lor.
2. Pentru propoziţia dată căutaţi şi vizualizaţi perechi de cuvinte în care unul dintre cuvinte este inversarea altuia.
3. Pentru propoziţia dată calculaţi frecvenţa repetării fiecărui cuvînt.
4. Pentru propoziţia dată găsiţi cel mai lung cuvînt simetric.
5. Este determinată distanţa dintre două cuvinte cu un număr egal de poziţii în care diferă aceste cuvinte. Pentru propoziţia dată găsiţi o pereche de cuvinte cu distanţa maximă.
6. Găsiţi setul de cuvinte care se aseamănă în fiecare din două propoziţii date.
7. Din propoziţia dată eliminaţi cuvintele situate pe poziţii impare şi inversaţi cuvintele de pe poziţiile pare.
8. Găsiţi cel mai lung cuvînt comun pentru două propoziţii date.
9. Sunt date două propoziţii. Găsiţi cel mai scurt cuvînt din prima propoziţie care nu este în a doua propoziţie.
10. Pentru propoziţia dată găsiţi cuvintele nesimetrice cu lungimea impară.
11. Din propoziţia dată eliminaţi cuvintele care se întîlnesc în propoziţie de un număr dat de ori.
12. Este dat un şir de caractere. Calculaţi şi vizualizaţi frecvenţa repetării fiecărui caracter.
13. Este dat un şir de caractere, care conţine de la 2 pînă la 20 de cuvinte; cuvintele sunt separate prin spaţiu; după ultimul cuvînt urmează punct. Afişaţi cuvintele, care corespund următoarelor proprietăţi:
14. cuvîntul este simetric;
15. prima literă a cuvîntului se repetă de mai multe ori în acest cuvînt;
16. Este dat un şir de caractere, care conţine de la 2 pînă la 20 de cuvinte; cuvintele sunt separate prin spaţiu; după ultimul cuvînt urmează punct. Afişaţi cuvintele, care corespund următoarelor proprietăţi:

fiecare literă se repetă de două sau mai multe ori;

1. Program pentru citirea unei linii ce conţine cuvinte formate din litere mici şi alte şiruri şi afişarea cuvintelor formate din litere mici (separate între ele prin orice alte caractere).

Exemplu:

|  |  |
| --- | --- |
| Intrare | Ieşire |
| ( unu, 1 doi DOI trei; "patru" ) | unu doi trei patru |

1. Pornind de la un sir citit de la tastatura, sa se construiasca si sa se tipareasca:

* sirul care contine cifrele din cel initial
* sirul care contine minusculele vocalelor din cel initial
* sirul invers celui initial; sa se verifice daca sirul initial este palindrom

1. Sa se citeasca si sa se tipareasca o matrice de siruri. Sa se gaseasca sirul din matrice care are numarul maxim de consoane.
2. Pentru propoziţia dată să se gasească cuvintele asimetrice cu o lungime impară şi simetrice cu o lungime pară.Apoi ele să se înlocuiască cu valoarea lor numerică, alcătuită din suma codurilor fiecărei litere din cuvînt.
3. Scrieti 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.
4. Scrieti un program care sa tipareasca o histograma a lungimii cuvintelor de la intrarea sa . Histograma se va trasa orizontal.
5. Scrieti un program care afiseaza cuvintele de la intarare cate unul pe o linie.
6. Scrieti 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.
7. Scrieti un program entab care inlocuieste sirurile de blank-uri prin numrul minim de tab-uri si blank-uri care realizeaza aceeasi spatiere.
8. Scrieti un program care elimina toate comentariile dintr-un program C.
9. Scrieti un program care sa gaseasca erorile elementare de sintaxa dintr-un program C , de genul paranteze rotunde , patrate , acolade neinchise , ghilimele , apostroafe, comentarii.
10. Scrieti o functie reverse(s) care sa inverseze sirul de caractere s.
11. Scrieţi o funcţie reverse(s) care să inverseze şirul de cractere s.
12. M
13. Elaborati un program care sa scrie la terminal codurile ASCII ale literelor mari in zecimal si hexazecimal.
14. Elaborati un program cu o functie care compara 2 siruri de caractere astfel:

* daca sirurile sunt egale intoarce valoarea 0 ;
* daca sirul 1 >sirul 2 intoarce -1;

1. Pentru propoziţia dată să se calculeze frecvenţa repetării fiecărui cuvânt, fiecărei rădăcini şi vocale. Să se afişeze şirurile obţinute şi numărul de operaţii efectuate.
2. Este dat un şir de caractere- care conţine de la 1 pînă la 22 cuvinte, unde fiecare din aceste cuvinte include de la 1 pînă la 5 litere şi cuvintele sunt separate prin virgulă; iar după ultimul cuvînt urmează punct.

Afişaţi şi vizualizaţi:

a) şirul- în care cuvintele şirului iniţial sunt situate în ordinea inversă;

b) cuvintele înaintea cărora în şirul dat se află numai cuvintele ce le precedează după alfabet, iar după aceea numai cuvintele ce le urmează;

c) şirul iniţial fără cuvintele ce se repetă mai mult de 2 ori.

37. De introdus de la tastatură o propoziţie. Cuvintele cărora sunt separate prin spatiu, iar după ultimul cuvînt urmează punct. De afişat din propoziţie şirurile care corespund cerinţelor urmatoare:

1. şirul obtinut din propoziţia dată, eliminînd cuvintele de pe pozitii impare;
2. şirul obţinut din propoziţia dată, inversînd cuvintele de pe poziţii pare;
3. subşirurile egale din ambele şiruri: cel iniţial şi cele obţinute.

38. Este dat un şir de caractere- care conţine de la 4 pînă la 25 cuvinte şi cuvintele sunt separate prin spaţiu; iar după ultimul cuvînt urmează punct.

Transformaţi şi afişaţi fiecare cuvînt a şirului după regulile următoare:

a) deplasaţi prima literă a cuvîntului la sfîrşitul lui;

b) deplasaţi prima literă la sfîrşitul cuvîntului;

c) înlăturaţi penultima literă a cuvîntului;

d)dacă lungimea cuvîntului este impară- atunci înlăturaţi litera din mijlocul lui.

39. Se consideră dată o frază alcătuită din 2 cuvinte cu m şi respectiv n litere. Să se transforme primul cuvînt în cuvîntul al doilea, utilizînd 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.

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

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

42. Pentru propoziţia dată găsiţi cel mai lung cuvînt simetric. Dacă astfel de cazuri nu se întîlnesc, atunci modificaţi cuvintele prin permutări.

43. 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.

44. Pentru 2 propoziţii date găsiţi setul de cuvinte care se întîlnesc în fiecare din ele. Apoi calculaţi numărul de cuvinte în set şi lungimea fiecărui cuvînt.

45. 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.

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

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

48. 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.

49. 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.

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

a) primă literă a cuvîntului ce se repetă de mai multe ori în fiecare cuvînt;

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

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

51. 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.

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

Afişaţi şi vizualizaţi:

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

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

c) cuvintele ordonate după alfabet şi invers.

54. Anagrame. Vom citi cuvantul in sirul de caractere s. Vom nota cu n lungimea sirului s. La citire contorizam si numarul de aparitii ale fiecarei litere in cuvant (nr[i]=numarul de aparitii in cuvant ale celei de a i-a litere din alfabet). Numarul de anagrame distincte (permutari cu repetitie) este: n!/(nr[0]!nr[1]!...[nr[25]!)

Nu este necesar sa calculam factorialele, calculam doar descompunerea in factori primi a factorialelor. In vectorul m retinem multiplicitatile factoprilor primi. m[i]=multiplicitatea lui i in descompunerea in factori primi a rezultatului. Calculam mai intai in m descompunerea lui n!.

Apoi "simplificam" rezultatul, scazand multiplicitatile din m cu cele corespunzatoare descompunerilor factorialelor de la numitor.

55. Sa se scrie un program care:

- citeste cuvintele tastate fiecare pe cate un rand nou, pana la CTRL/Z   ( varianta: pana la introducerea un cuvant vid )

* afiseaza cuvantul cel mai lung
* - 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.

56 Se citesc trei siruri s1, s2 si s3. Sa se afiseze sirul obtinut prin inlocuirea in s1 a tuturor aparitiilor lui s2 prin s3. ( Observatie: Daca s3 este sirul vid, din s1 se vor sterge toate subsirurile s2 ).

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.

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

Afişaţi şi vizualizaţi:

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

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

c) cuvintele ordonate după alfabet şi invers.

59. Pornind de la un sir citit de la tastatura, sa se construiasca si sa se tipareasca ( ultimele trei prelucrari nu sunt implementate ):

* sirul care contine cifrele din cel initial
* sirul care contine minusculele vocalelor din cel initial
* sirul invers celui initial; sa se verifice daca sirul initial este palindrom
* sirul care contine majusculele literelor din cel initial
* sirul care contine m caractere incepand de la pozitia p din sirul initial; m si p se citesc
* sirul care contine caracterele celui initial cu exceptia celor m incepand de la pozitia p; m si p se citesc.

## 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-a22 22ditura 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