LUCRARE PENTRU OBȚINEREA ATESTATULUI DE COMPETENȚE PROFESIONALE

ȘIRURI DE CARACTERE

Candidat: Prof. îndrumător:

Apetrei Denis Pintea Eugen

Clasa a XII-a B

Matematică-Informatică

Intensiv Engleza

Anul școlar 2020-2021

Colegiul Național “Mihai Eminescu” Suceava

Cuprins

[I. Argument 3](#_Toc62069858)

[II. Introducere 4](#_Toc62069859)

[III. Pointeri, tipul char\* 7](#_Toc62069860)

[IV. Funcții specifice pentru caractere 8](#_Toc62069861)

[V. Funcții pentru prelucrarea șirurilor de caractere 9](#_Toc62069862)

[VI. Probleme cu șiruri de caractere 18](#_Toc62069863)

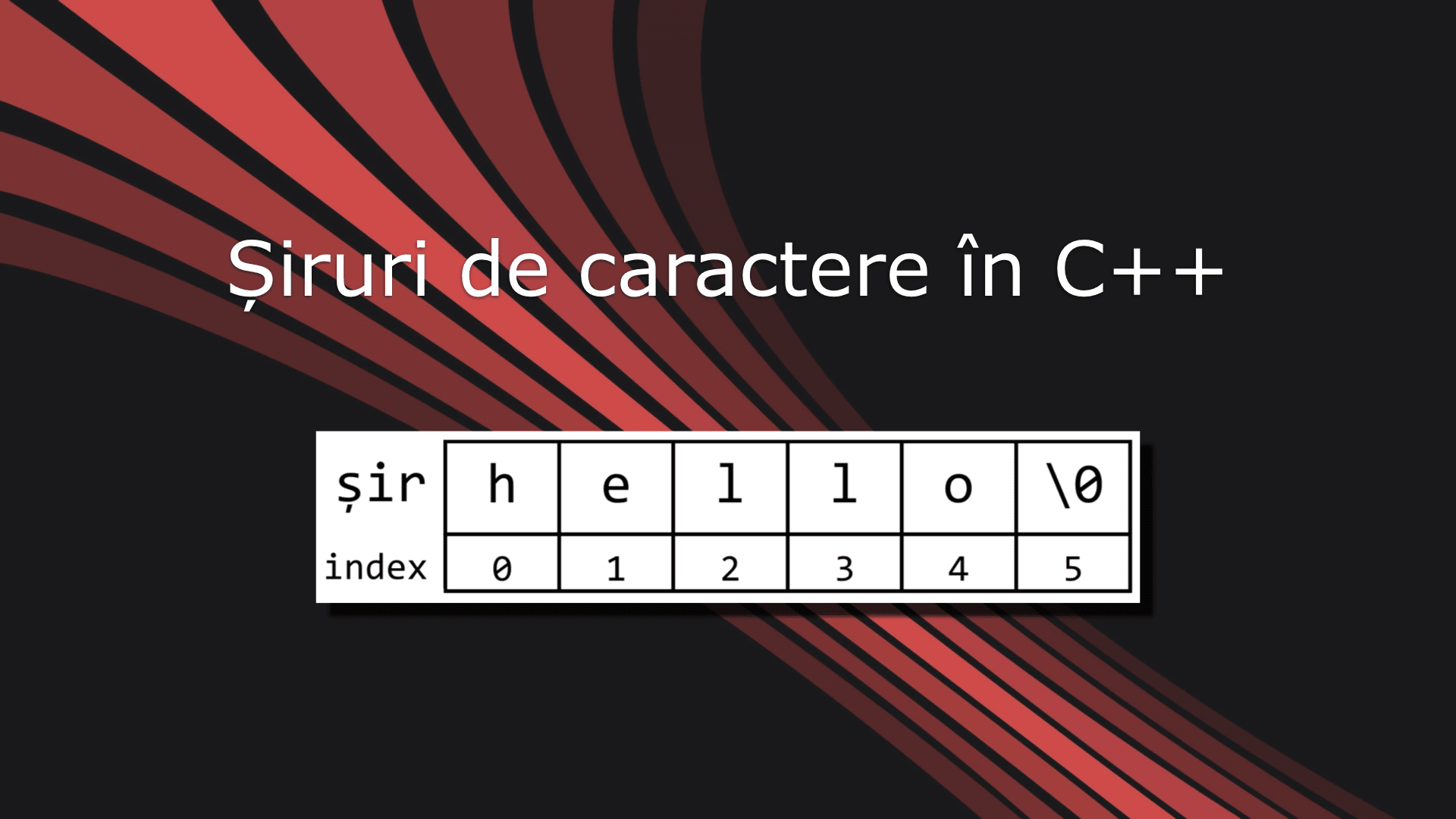
[VII. Bibliografie 21](#_Toc62069864)

## I. Argument

În era calculatoarelor și a vitezei suntem conștienți de faptul că trăim într-o relație de interdependență cu tot ceea ce înseamnă tehnologie.  Fără să ne dăm seama folosim zilnic aparatura ce are la bază un algoritm extrem de bine structurat, organizat și eficient.

Alegerea temei pentru atestatul meu la această disciplină extrem de căutată astăzi a avut la temelie dorința de a putea face, câtuși de puțin, curios pe oricine ar avea o oarecare îndoială de importanța acestui subiect în zilele în care trăim.

Fie ca vorbim de fabrici, corporații sau afaceri de familie, informatica este mașina care pune totul în mișcare.  Se formează astfel, nu doar un circuit virtual, dar și unul în timp real. Un mecanism ce are la rădăcină funcții, șiruri, matrici de numere, litere și oameni. Devenim una cu tehnologia și de aceea sunt de părere că pătrunderea, fie ea cât de sumară, în bula modernității, informaticii ne va face mai vigilenți de primordialitatea acesteia în viața pe care o ducem astăzi atât în societate cât și în viața privată.



# II. Introducere

Definiție

Un șir de caractere este o structură de date formată dintr-o succesiune de caractere care se termină cu caracterul NULL (’ \0’).

Implementarea șirurilor de caractere se face sub forma unui tablou unidimensional (vector) ale cărui elemente sunt de tip caracter, fiecare caracter fiind reprezentat prin codul său ASCII. Șirul de caractere se deosebește de vectorii cu alte tipuri de elemente prin marcatorul logic utilizat la sfârsitul șirului - NULL (care are codul ASCII 0). Limbajul C/C++ permite inițializarea unui tablou de caractere printr-o constantă șir, care include automat caracterul null.

Declararea șirurilor de caractere

Dimensiunea vectorului de caractere trebuie declarată cu un caracter mai mult decât cel mai mare șir de caractere pe care îl poate conține, pentru a exista loc și pentru terminatorul șirului (caracterul ‘\0’).

„char s[15];” // am declarat șirul s care va putea conține maxim 5 caractere

O constantă șir de caractere este o succesiune de caractere delimitată de ghilimele.

Exemplu; char s[256]=”Lucrare Atestat”;

De asemenea, șirurile de caractere pot fi declarate folosind pointeri. Unui șir astfel inițializat nu i se poate modifica valoarea. El este reținut în memorie ca un șir constant.

Citirea șirurilor de caractere

Limbaul C deţine numeroase funcţii pentru prelucrarea şirurilor de caractere. Funcţiile cin şi cout suportă şirurile de caractere terminate cu caracterul nul, astfel pot fi citite respectiv afişate şirurile de caractere.

* cin >> s;

În acest mod, datorită specificului operatorului >> nu se pot citi șiruri care conțîn spații – se vor citi caracterele până la primul spațiu, fără acesta. Pentru a citi șiruri care conțin spații, putem folosi metoda getline a obiectului cin. Se vor citi în șirul s caracterele din stream-ul de intrare (de la tastatură) până la apariția caracterului sfârșit de linie '\n', dar nu mai mult de n-1 caractere. Caracterul '\n' nu va fi adăugat la șirul s, dar va fi extras din stream:

* cin.getline(s , n);

Există încă o implementare a funcţiei cin.get() fară parametri. În acest caz funcţia citeşte un singur caracter alb sau nu.

În cazul citirii a două şiruri de caractere, în buffer-ul de citire se depune după citirea primului şir, caracterul ‘\n’, datorită faptului că s-a tastat. Citirea celui de-al doilea şir se opreşte pentru că se preia ‘\n’.din buffer. Apelând funcţia cin.get() aceasta va prelua caracterul ‘\n’, eliminăm acest inconvenient, iar al doilea şir se va citi în mod normal.

Un alt mod de citire a șirurilor, pe care nu îl vei folosii foarte des este citirea caracter cu caracter. Pentru a implementa acest tip de citire trebuie mai întăi să cunoaștem lungimea totala a șirului.

char s[15];

int n;

cin >> n;

for(int i = 0; i < n; i++)

cin >> s[i];

cout << s;

Parcurgerea șirurilor de caractere

În numeroase situații este necesară analizarea fiecărui caracter din șir. Pentru aceasta este necesară o parcurgere a șirului; aceasta se face similar cu parcurgerea unui tablou oarecare. Diferența constă în faptul că, pentru șirul de caractere nu se cunoaște explicit lungimea. Ea poate fi determinată cu funcția strlen, dar se poate controla parcurgerea șirului știind că după ultimul caracter valid din șir apare caracterul nul '\0'.

char s[11];

cin >> s; // se citeste un cuvant , fara spatii

for(int i = 0 ; s[i] ; i ++)

cout << s[i] << " ";

Sau dacă apare caracterul nul dupa ultimul caracter din șir:

char s[11];

cin >> s; // se citeste un cuvant , fara spatii

int i = 0;

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

{

cout << s[i] << " ";

i ++;

}

# III. Pointeri, tipul char\*

Tipul de dată char\* poartă numele de pointer la character. O variabilă de tip pointer la caracter, este capabiăl să reţină adresa de memorie a unui caracter. Numele unei variabile de tip şir de caractere (sau vector în general), reprezintă adresa de memorie a primului octet al şirului (vectorului).

char \* p , s[31] = "Atestat";

Spunem că este un șir de caractere, dar practic s este tot un pointer. Valoarea sa este adresa primului element din șir, adică adresa lui s[0]. Observăm că de fapt, variabilele p și s sunt de același tip, pointer la char. Diferența dintre cele două variabile este că s memorează o adresă de memorie unde începe un șir de caractere (la acea adresă există o dată de tip char ) în timp ce p memorează o adresă aleatorie.

Pointerul p poate fi inițializat cu adresa unei date de tip char. O asemenea dată este orice element al unui șir de caractere, de exemplu orice element din s. Dacă p reprezintă adresa unui caracter dintr-un șir, atunci cu p se pot face toate operațiile care se pot face cu acel șir.

Ex:

#include <iostream>

using namespace std;

int main(){

char \* p , s[]="atestat";

cout << s << endl; // atestat

p = s;

cout << p << endl; // atestat

p ++;

cout << p << endl; // testat

return 0;

}

Pointerii rețin informația de la o anumită adresă. Astfel, știind că funcțiile strtok, strchr și strstr returnează adresa unor elemente, pentru a accesa informația de la acea adresă este recomandat să folosim un char \*

# IV. Funcții specifice pentru caractere

Tipul predefinit, destinat în C++ lucrului cu caractere, este tipul char. Spațiul de memorie asociat variabilelor acestui tip este 1 octet; un caracter se memorează ca numărul întreg corespunzător codului ASCII al caracterului respectiv. Prin urmare, în C++ un caracter poate apărea în operații numerice. Valorile acestui tip sunt cuprinse în intervalul [-128...127] în cazul caracterelor cu semn, respectiv [0...255] în cazul celor fară semn.

Următoarele funcții au ca parametri valori numerice, reprezentând codul ASCII al unor caractere. Prototipul lor se află în header-ul ‘cctype’.

ISALNUM

int isalnum( int ch );

Verifică dacă un caracter este alfanumeric (cifră, literă mare, literă mică). Returnează o valoare diferită de zero dacă parametrul este alfanumeric, 0 în caz contrar.

ISALPHA

int isalpha( int ch );

Verifică dacă un caracter este alfabetic (literă mare, literă mică). Returnează o valoare diferită de zero dacă parametrul este alfabetic, 0 în caz contrar.

ISUPPER

int isupper( int ch );

Verifică dacă un caracter este literă mare. Returnează o valoare diferită de zero dacă parametrul este literă mare, 0 în caz contrar.

ISLOWER

int islower( int ch );

Verifică dacă un caracter este literă mică. Returnează o valoare diferită de zero dacă parametrul este literă mică, 0 în caz contrar.

ISDIGIT

int isdigit( int ch );

Verifică dacă un caracter este cifră. Returnează o valoare diferită de zero dacă parametrul este cifră, 0 în caz contrar.

TOLOWER

int tolower( int ch );

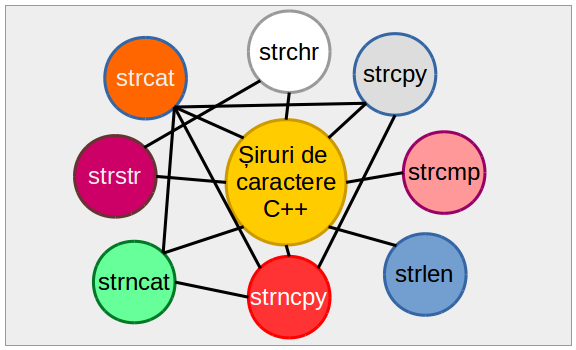
Convertește parametrul la literă mică. Dacă parametrul este literă mare, returnează valoarea convertită, în caz contrar returnează valoarea inițială a parametrului.

TOUPPER

int toupper( int ch );

Convertește parametrul la literă mare. Dacă parametrul este literă mică, returnează valoarea convertită, în caz contrar returnează valoarea inițială a parametrului.

# V. Funcții pentru prelucrarea șirurilor de caractere



Şirurile de caractere fiind în realitate tablouri unidimensionale nu sunt permise atribuiri de forma sir1=sir2 şi comparări între sir1 şi sir2. Pentru efectuarea acestora şi a altor operaţii sunt implementate funcţii care lucrează cu şiruri de caractere. Dacă nu se precizează altfel, prototipul lor se află în header-ul cstring.

STRLEN

Strlen(sir) returnează un număr întreg ce reprezintă lungimea unui şir de caractere, fără a număra terminatorul de şir.

int main{

cout << strlen("atestat"); // 7

char s[10]="elevi";

cout << strlen(s); // 5

cout << strlen(s + 2); //3

return 0;

}

STRCHR

strchr(sir,c); – are rolul de a căuta caracterul c în șirul sir. Căutarea se face de la stânga la dreapta, iar funcția întoarce adresa subșirului care începe cu prima apariție a caracterului c. Dacă nu este găsit caracterul, funcția returnează 0. Diferența dintre adresa șirului inițial și cea a subșirului returnat reprezintă chiar poziția caracterului căutat în șirul dat.

int main(){

char s[21]="atestat";

char \* p = strchr(s , 'a');

cout << p; // atestat

return 0;}

Ex: Să se afișeze toate pozițiile unui caracter într-un șir.

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

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

cin.get(s,100);

cin>>c;

p=strchr(s,c);

while (p)

{

cout<<”Pozitia”<<p-a<<”\n”;

p++;

p=strchr(p,c);

}

Return 0;

}

STRRCHR

Caută de la dreapta la stânga, caracterul car în şirul de caractere sir. Dacă este găsit, funcţia întoarce adresa subşirului care începe cu ultima apariţie a caracterului citit şi se termină cu carcterul nul. Dacă nu este găsit intoarce o expresie de tip char\* cu valoarea 0.

Exemplu:

char a[100]=“crocodil”;

cout<<strrchr(a,’o’); //odil

STRCPY

Copiază şirul de la adresa sursă la adresa destinaţie. Copierea se termină la întâlnirea caracterului nul. Funcţia returnează adresa şirului destinaţie. Simulează operaţia de atribuire a=b.

Exemplu:

char a[100]=“lucrare”,b[100]=“atestat”;

strcpy(a,b);

cout<<“sirul a: ”<<a<<endl; //atestat

cout<<“sirul b: ”<<b<<endl; //atestat

STRNCPY

Copiază primii n octeţi din şirul de la adresa sursă la adresa destinaţie, fără a adăuga caracterul nul. Funcţia returnează adresa şirului destinaţie. Șirul sursă rămâne nemodificat.

Exemplu:

char a[100]=“lucrare”,b[100]=“atestat”;

strncpy(a,b,4);

cout<<“sirul a: ”<<a<<endl; // ates

cout<<“sirul b: ”<<b<<endl; // atestat

STRCAT

Adaugă şirului de la adresa destinaţie, înaintea caracterului nul şirul de la adresa sursă. Şirul de la adresa sursă rămâne nemodificat. Operaţia se numeşte concatenare. La adresa destinaţie vom avea şirul destinaţie urmat de şirul sursă. Şirul destinaţie are lungimea egală cu suma lungimilor şirurilor.

Exemplu:

char a[100]=“lucrare”,b[100]=“atestat e”;

strcat(a,b);

cout<<“sirul a: ”<<a<<endl; //lucrareatestat

cout<<“sirul b: ”<<b<<endl; //atestat

STRNCAT

Adaugă şirului de la adresa destinaţie, înaintea caracterului nul primii n octeţi ai şirul de la adresa sursă. Şirul de la adresa sursă rămâne nemodificat. Funcţia returnează adresa de început a şirului destinaţie.

Exemplu:

char a[100]=“lucrare ”,b[100]=“atestat”;

strncat(a,b,3);

cout<<“sirul a: ”<<a<<endl; //lucrare ate

cout<<“sirul b: ”<<b<<endl; //atestat

STRSTR

Identifică dacă şirul sir2 este subşir(caractere succesive) al şirului sir1. dacă este găsit, funcţia returnează adresa sa de început în cadrul şirului s1, altfel returnează 0. Căutarea se face de la stânga la dreapta. Dacă sir2 apare de mai multe ori, returnează adresa primei sale apariţii.

Exemplu:

char a[100]=“azi ele fac cafele”,b[20]= “ele”;

cout<<strstr(a,b); //ele fac cafele

STRTOK

strtok(sir1,sir2);

Are rolul de a separa șirul sir1 în mai multe șiruri (cuvinte) separate între ele prin unul sau mai multe caractere cu rol de separator. Șirul sir2 este alcătuit din unul sau mai multe caractere cu rol de separator. Funcția strtok acționează în felul următor:

Primul apel trebuie să fie de forma strtok(sir1,sir2); funcția întoarce adresa primului caracter al primei entități. După prima entitate, separatorul este înlocuit automat cu caracterul nul.

Următoarele apeluri sunt de forma strtok(NULL,sir2); de fiecare dată, funcția întoarce adresa de început a următoarei entități, adăugând automat după ea caracterul nul.

Când șirul nu mai conține entități, funcția returnează adresa nulă.

Exemplu:

char a[100],sep[]=“, ;.?!”,\*p;

cin.get(a,100);

**p=strtok(a,sep);**

while (p)

{ cout<<p<<endl;

**p=strtok(NULL,sep);**

}

STRCMP

int strcmp(char \*sir1,char \*sir2);

Compară cele două şiruri de caractere. Valoarea returnată este:

<0 dacă sir1<sir2

=0 dacă sir1=sir2

>0 dacă sir1>sir2

Funcţia face distincţie între literele mari și literele mici. Compararea şirurilor se realizează comparând de la stânga la dreapta caracter cu caracter. Un şir este mai mic dacât altul dacă figurează în dicţionar înaintea lui.

Exemplu:

char a[20]=“adriana”,b[20]= “ana”, c[20]= “Ana”;

cout<<strcmp(a,b); //<0 deoarece ‘a’=’a’ si ‘d’<’n’ => “adriana”<“ana”

cout<<strcmp(a,c); //>0 deoarece ‘a’>’A’

cout<<strcmp(b,c); //>0 deoarece ‘a’>’A’

STRICMP

int stricmp(char \*sir1,char \*sir2);

Are acelaşi efect ca şi strcmp dar nu face diferenţă între literele mari şi literele mici.

char b[20]= “ana”, c[20]= “Ana”;

cout<<stricmp(b,c); //==0

STRNCMP

int strncmp(char \*sir1,char \*sir2,int n);

Are acelaşi efect ca şi strcmp dar compara doar primele n caractere din cele doua siruri.

Exemplu:

char b[20]= “adriana”, c[20]= “adina”;

cout<<strncmp(b,c,2); //==0

STRNCMPI

int strncmpi(char \*sir1,char \*sir2,int n);

Are acelaşi efect ca şi strncmp dar nu face diferenţă între literele mari şi literele mici.

Exemplu:

char b[20]= “adriana”, c[20]= “ADina”;

cout<<strncmpi(b,c,2); //==0

STRSPN

int strspn(sir1,sir2);

Are rolul de a returna numărul de caractere ale șirului sir1 (caractere consecutive care încep obligatoriu cu primul caracter) care se găsesc în șirul sir2 .

Exemplu:

strspn(“AB2def”,”1B3AQW”); //returneaza 2, pentru ca primele 2 caractere „A‟ si „B‟ din sir1 se gasesc in sir2 .

strspn(“FAB2def”,”16A32BF”); // returneaza 0, deoarece caracterul „F‟ cu care incepe sir1 nu se gaseste in sir2.

STRUPR

char \*strupr(char \*s)

Transformă un şir de caractere din litere mici în litere mari. Restul caracterelor rămân nemodificate.

Exemplu :

char a[100]=“1 atestat”;

cout<<strupr(a); //1 ATESTAT

STRLWR

char \*strlwr(char \*s);

Transformă un şir de caractere din litere mari în litere mici. Restul caracterelor rămân nemodificate.

Exemplu :

char a[100]=“1 ATESTAT”;

cout<<strlwr(a); //1 atestat

STRBRK

strbrk(sir1,sir2);

Caută primul caracter al șirului sir1 în sir2 . Dacă este găsit, returnează adresa sa din cadrul șirului sir1 și execuția se termină. Altfel, se trece la pasul următor.

Caută al doilea caracter al șirului sir1 în sir2 . Dacă este găsit, returnează adresa sa din cadrul șirului sir1 și execuția se termină. Altfel, se trece la pasul următor.

…

Dacă nici un caracter al șirului sir1 nu aparține șirului sir2 , funcția returnează adresa nulă.

Functii din libraria <stdlib.h>

**ATOF**

double atof(sir); Convertește un sir către tipul double. Dacă această conversie eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.

**ATOLD**

**long double \_atold(sir); Convertește un sir către tipul long double. Dacă această conversie eșuează, valoarea întoarsă este 0.**

**ATOI**

**int atoi(sir); Convertește un sir către tipul int. Dacă această conversie eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.**

**ATOL**

**long atol(sir); Convertește un sir către tipul long. Dacă această conversie eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.**

**ITOA**

**itoa(int valoare,sir,int baza); Convertește o valoare de tip int în sir, care este memorat în variabila sir. Baza reține baza de numerație către care să se facă conversia. În cazul bazei 10, șirul retine și eventualul semn -.**

**LTOA**

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

**ULTOA**

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

#include<iostream>

#include<cstring>

#include<stdlib.h>

int main()

{char sir[100]; int n;

cout<<"dați șirul ";cin.get(sir,100);

n=atoi(sir);

cout<<"numărul după conversie "<<n<<"\n";

itoa(n,sir,10);

cout<<n<<" în baza 10 "<<sir<<"\n";

itoa(n,sir,8);

cout<<n<<" în baza 8 "<<sir<<"\n";

itoa(n,sir,16);

cout<<n<<" în baza 16 "<<sir<<"\n";

itoa(n,sir,2);

cout<<n<<" în baza 2 "<<sir<<"\n";

return 0;

}

# VI. Probleme cu șiruri de caractere

1.Se consideră un text cu cel mult 100 de caractere (litere mici ale alfabetului englez şi spaţii), care conţine cel puţin o vocală. Scrieţi un program C/C++ care citeşte de la tastatură un text de tipul menţionat mai sus şi determină transformarea acestuia în memorie prin duplicarea tuturor vocalelor din text, că în exemplu. Programul afişează pe ecran textul obţinut. Se consideră vocale literele a, e, i, o, u.

Exemplu: pentru textul : problema aceea frumoasa

se afişează: proobleemaa aaceeeeaa fruumooaasaa.

Rezolvare:

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char s[202],aux[201;

cin.get(s,101);

for(int i=0;i<strlen(s)-1;i++)

if(strchr("aeiou",s[i]))

{

strcpy(aux,s+i);

strcpy(s+i+1,aux);

i++;

}

cout<<s;

return 0;

}

2.Se citește un număr n, și n versuri dintr-o poezie. Fiecare vers ocupă un rând și conține litere mari și mici ale alfabetului englez, spații, și semne de punctuație . , ! ?. Să se elimine din fiecare vers toate semnele de punctuație, mai puțin cel aflat la final.

Rezolvare:

#include <iostream>

#include <cstring>

using namespace std;

char s2[256];

int main()

{

char s[256],\*p;

int n,m,i,j,k;

cin>>n;

cin.get();

for(i=1;i<=n;i++)

{

cin.get(s,101);

cin.get();

m=strlen(s);

for( j=0;j<m;j++)

{

if(strchr(",.!?",s[j])&&j!=m-1)

{for(k=j;k<m;k++)

s[k]=s[k+1];

m--;

j--;

}

}

strcat(s2,s);

strcat(s2,"\n");

}

cout<<s2;

return 0;

}

3.Un șir de caractere s este șablon pentru un șir de caractere x dacă are aceeași lungime cu x și este alcătuit numai din caractere ale mulțimii {\*,#,?}, iar pe fiecare poziție din s în care apare \* în x apare vocală, pe fiecare poziție din s în care apare # în x apare consoană, iar pe fiecare poziție din s în care apare ? în x putem avea orice caracter.

Rezolvare:

#include <iostream>

#include <cstring>

#include <string.h>

using namespace std;

char cons[20];

int main()

{

char s1[101],s2[101],voc[]="aeiou";

cin>>s1>>s2;

for(int i=0;i<strlen(s1);i++)

{

if(strchr(voc,s1[i])==0&&strchr(voc,s2[i])==0)

{

cout<<"#";

}

else

if(strchr(voc,s1[i])!=0&&strchr(voc,s2[i])!=0)

cout<<"\*";

else

cout<<"?";}

return 0;

}

# VII. Bibliografie

<http://www.cnchogastecuci.ro/portofoliu/elevi/Siruri_probleme.pdf>

<https://tutoriale-pe.net/siruri-de-caractere-in-c-declararea-citirea/>

<http://info.tm.edu.ro:8080/~junea/cls%2010/siruri%20de%20caractere/teorie%20siruri%20de%20caractere.pdf>

<http://www.cnchogastecuci.ro/portofoliu/elevi/Siruri_probleme.pdf>

<http://pacosv.ro/2019/siruri%20caractere%20teorie.pdf>

<https://docs.microsoft.com/en-us/cpp/cpp/string-and-character-literals-cpp?view=msvc-160>

<https://www.pbinfo.ro/>

Bazele programarii in limbajul C++ de Liviana Tudor

Programarea in limbajul C/C++ pentru liceu. Volumul I de Marinel Serban, Emanuela Cerchez