ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №14

Sətirlər

Mündəricat

1. C dilində sətirlərlə iş

Sətir massivlərinin təyin edilmə sintaksisi və onların qiymətləndirilməsi

Keçən dərslərimizdə siz massivlərin müxtəlif növləri ilə tanış oldunuz. İndi isə massivlərin daha bir növü olan sətir massivlərini daha dərindən araşdıracağıq. Sətirlər simvol tipli informasiyaların girişi, çıxışı və emalı üçün təyin olunmuşdurlar.

Sətir sabiti - dırnaq işarəsi içərisində yazılmış sıfır və ya daha çox simvollardan ibarət ardıcıllıqdır. Dırnaq işarələri özləri isə sətir sabitinin bir hissəsi hesab olunmurlar, yalnız sabiti məhdudlaşdırmaq üçün istifadə olunurlar.

Sətirlər char tipli massivin elementləri şəklində verilirlər. Bu o deməkdir ki, sətrin simvollarını hər xanada bir simvol olmaq şərtilə yaddaşın qonşu xanalarında yerləşdirmək olar. Amma simvollardan ibarət massiv həmişə sətir deyil!

Nümunə üçün aşağıdakı sətirə baxaq: «Simvol sətri». Dırnaqlar sətrin hissəsi sayılmır. Onlar sətrin başlanğıcını və sonunu qeyd etmək üçün daxil edilmişdirlər (hər xana - 1 baytdır).

S	i	m	v	o	l		s	ə	t	r	i	\0

Qeyd etmək lazımdır ki, şəkildə massivin sonuncu elementi '\0' simvoludur. Bu sıfır-simvoldur: C dilində o, sətrin axırını bildirmək üçün istifadə olunur. Sıfır-simvol - 0 ədədi deyildir; o çap edilmir və ASCII kodlar cədvəlində 0 nömrəsinə malikdir. Sıfır-simvolun varlığı bildirir ki, yaddaşın xanalarının sayı burada yerləşdiriləcək simvollar sayından heç olmasa bir dənə artıq olmalıdır.

Simvol sabitini bir simvoldan ibarət sətir ilə qarışdırmaq olmaz: 'X' ilə "X" bir-birindən fərqlənirlər. Birinci halda - bu ayrıca simvoldur. İkinci halda isə - bir simvoldan (X hərfi) və '\0' sətrin sonu simvolundan ibarət sətirdir.

Sətir massivlərinin qiymətləndirilməsi qaydaları

Sətir massivinin qiymətləndirilməsi nümunəsinə baxaq:

```
#include <iostream>
using namespace std;
int n=5;
// Sətir massivinin qiymətləndirilməsi.

char line[5] = { 'C','a','t','!','\0' };
void main ()
{
    cout << "Word: ";
    for (int i=0; i<n; i++)
        cout << line[i];
}
```

Yuxarıda göstərilən nümunə uzun sətirlərin yaradılması üçün çox da rahat deyil.

Bundan əlavə, sətrin dovr daxilində tək-tək hərflərlə verilməsi qəribə görünür, elə deyilmi? Simvol massivləri üçün xüsusi qiymətləndirmə metodu vardır. Fiqurlu mötərizə və vergüllərin əvəzinə dırnaq içində yazılmış simvollar sətrindən istifadə etmək olar. Belə təsvir zamanı massivin ölçüsünü göstərmək məcburi deyil, çünki kompilyator başlanğıc qiymətləri hesablayaraq özü onun uzunluğunu təyin edir. Bu vaxt cout əməliyyatı elə şəkildə tənzimlənmişdir ki, sətir massivinin yalnız adını göstərmək kifayətdir ki, o, ekranda təsvir olunsun.

```
#include  <iostream>
using namespace std;
int n=5;
    char line[] = "Cat!";//Sətir massivinin
                         // qiymətləndirilməsi

void main ()
{
    cout<< "Word: ";
    // Sətir massivinin ekranda göstərilməsi.
    cout<<line;
}
```

2. Sətir və göstəricilərin qarşılıqlı əlaqəsi

Dərslərimizin birində biz artıq göstəricilər mövzusunu müzakirə etmişik və yəqin ki, sizin yadınızdadır ki, onlar massivlərlə sıx əlaqədədir. Ona görə də biz göstəricilərdən yan keçə bilmərik. Bu proqramda sətirə müraciət simvol göstəricinin köməyi ilə həyata keçirilir. Əgər message dəyişənini belə təsvir etsək:

```
char *message;
```

onda operatorun yerinə yetirilməsi nəticəsində

```
message = "and bye!";
```

message dəyişəni sətir göstəricisi olar. Diqqət yetirin ki, cin operatorunu belə göstəriciyə tətbiq etmək olmayacaq.

```
#include  <iostream>
using namespace std;
char *message;
char privet[] = "and bye!";
char *pr = privet;
void main ()
{
    message = "Hello";
    cout << " " << message << " " << pr << "\n";
    int i = 0;
    while  (*(pr+i)!='\0')
```

```
    {
        cout<< *(pr+i++) << " ";
    }
}
```

Göstəricilərdən istifadə çox vaxt sətir massivləri ilə iş zamanı tətbiq olunur. Bu halda hər bir sətrə onun birinci simvoluna göstəricinin köməyi ilə müraciət etmək olar. Belə ki, düzgün olmayan nizamla yerləşmiş iki sıranın yerdəyişməsi üçün sıraların yerini deyil, göstəricilər massivində göstəricilərin yerini dəyişmək kifayətdir.

Göstəricini n-ci ayın adı yazılan sətrə qaytaran ***month_name()*** funksiyasına baxaq. Bu, sətir massivindən istifadə etmək üçün tipik məsələdir.

month_name() funksiyasında lokal sətir massivi vardır və bu funksiyaya müraciət etdikdə o, göstəricini lazım olan sətirə qaytarır.

name[] simvoluna olan göstəricilərdən ibarət massivin təsvirində başlanğıc qiymət kimi sətirlərin siyahısı götürülür. i-ci sətrin simvolları yaddaşın müəyyən yerinə yerləşdirilir, bu sətrin əvvəlinə olan göstərici isə ***name[i]*** elementində saxlanılır. ***name*** massivinin ölçüsü göstərilmədiyinə görə, kompilyator özü başlanğıc qiymətlərin sayını hesablayır və müvafiq olaraq düzgün ədədi müəyyənləşdirir.

```
#include <iostream>
using namespace std;
const int n=15;
```

```
void main ()
{
     char *month_name(int);
     /*                  ---------- */
     for (int i=0; i < n; i++)
                        cout<<"Month number " << i << " - "
                        <<
                        month_name(i) <<"\n";
}

/* ---------------------------------------                    */
char *month_name (int k)  /*  k-cı ayın adı */
{
     static char *name[] = {
                        "none","January",
                        "February","March","April",
                        "May","June","July","August",
                        "September","October","November",
                        "December"
     };
     return (k<1||k>12)?name[0]:name[k];
}
```

Proqramın işinin nəticəsi:

```
Month number 0 - none
Month number 1 - January
Month number 2 - February
Month number 3 - March
Month number 4 - April
Month number 5 - May
Month number 6 - June
Month number 7 - July
Month number 8 - August
Month number 9 - September
Month number 10 - October
Month number 11 - November
```

```
Month number 12 - December
Month number 13 - none
Month number 14 - none
Press any key to continue
```

3. Sətirlərin emalı kitabxanasında olan sətirlərlə iş funksiyaları

Burada biz sətirlər üzərində iş üçün təyin olunmuş əsas funksiyaları sadalayacağıq. Bu funksiyaların bir çox prototipləri ***string.h*** başlıq faylında yerləşir.

- ***int getchar();*** — İstifadəçinin klaviaturada yığdığı əvvəlki qiymətini simvola qaytarır (əgər varsa). Simvolun daxil edilməsindən sonra ***Enter*** düyməsini basmaq lazımdır. Başlıq faylı — ***stdio.h***
- ***int getch();*** — Əvvəlki ilə analojidir, lakin simvol ekranda əks olunmur. Tez-tez proqramın yerinə yetirilməsini ləngitmək üçün istifadə olunur. Başlıq faylı — ***conio.h***
- ***int putchar(int c);*** — c simvolunu ekrana çıxarır. Uğurla yerinə yetirilərsə, c simvolunun özünü qaytarır, əks halda — ***EOF***. Başlıq faylı — ***stdio.h***
- ***char *gets(char *s);*** — Boşluqlar və tabulyasiyalar da daxil olmaqla simvolları, yeni sətir simvoluna rast gələnə qədər (hansı ki, sıfır simvolu ilə əvəz olunur) oxuyur. Oxunmuş simvollar ardıcıllığı yaddaşın s arqumentinin ünvanlanmış hissəsində yadda saxlanılır. Uğurlu halda s arqumentini qaytarır, xəta baş verdikdə isə — sıfır yazılır. Başlıq fayl — ***stdio.h***

- ***int puts(const char *s);*** — ***const char *s*** arqumenti ilə verilmiş sətri çıxarır. Başlıq faylı — ***stdio.h***
- ***char *strcat(char *dest, const char *scr);*** — ***scr*** başlanğıc sətrini ***dest*** nəticə sətrinə qoşaraq birəşdirir. Qaytarır — ***dest***.
- ***char *strncat(char *dest, const char *scr, int maxlen);*** — ***scr*** başlanğıc sətrinin ***maxlen*** simvolunu ***dest*** nəticə sətrinə birləşdirir. Qaytarır — ***dest***.
- ***char *strchr(const char *s, int c);*** — c simvolunun ilk girişini s sətirinin əvvəlindən başlayaraq axtarır. Uğurlu olduğu halda göstəricini tapdığı simvola qaytarır, əks halda sıfır qaytarır.
- ***char *strrchr(const char *s, int c);*** — Bundan əvvəlki ilə analojidir, yalnız axtarış sətrin sonundan aparılır.
- ***int strcmp(const char *s1, const char *s2);*** — İki sətri müqayisə edir. ***s1<s2*** olduğu halda məfi qiyməti, ***s1==s2*** olduqda sıfrı, ***s1>s2*** olduqda isə müsbət qiyməti qaytarır. Parametrlər — müqayisə olunan sətirlərin göstəriciləridir.
- ***int stricmp(const char *s1, const char *s2);*** — Əvvəlki ilə analojidir, yalnız müqayisə simvollar registrini nəzərə almadan aparılır.
- ***int strncmp(const char *s1, const char *s2, int maxlen);*** — Əvvəlki ilə analojidir, yalnız ilk ***maxlen*** simvollar müqayisə olunur.
- ***int strnicmp(const char *s1, const char *s2, int maxlen);*** — Əvvəlki ilə analojidir, yalnız ilk ***maxlen*** simvollar registri hesaba alınmadan müqayisə olunur.

■■***int strcspn(const char *s1, const char *s2);*** — **s1** sətrinin maksimal başlanğıc altsətrin **s2** sətrinin simvollarını özündə saxlamayan uzunluğunu qaytarır.

■■***int strlen(const char *s);*** — ***s*** sətrinin uzunluğunu — sıfır simvolundan əvvəlki simvolların sayını qaytarır.

■■***char *strlwr(char *s);*** — ***s*** sətrində bütün böyük hərfləri kiçik hərflərə çevirir.

■■***char *strupr(char *s);*** — ***s*** sətrində bütün kiçik hərfləri böyük hərflərə çevirir.

■■***char *strnset(char *s, int c, int n);*** — ***s*** sətrini ***c*** simvolları ilə doldurur. ***n*** parametri sətirdə yerləşən simvolların sayını göstərir.

■■ ***char *strpbrk(const char *s1, const char *s2);*** — ***s1*** sətrində **s2** sətrindən olan istənilən simvolun birinci girişini axtarır. Əgər belə bir simvol yoxdursa sıfır qaytarır.

■■***char *strrev(char *s);*** — sətirdə simvolların ardıcıllığı sırasını tərsinə (sonuncu sıfır simvolundan başqa) dəyişir. Funksiya **s** sətrinə qaytarır.

■■***char *strset(char *s, int c);*** — **s** sətrinin bütün simvollarını verilmiş **c** simvolu ilə əvəz edir.

■■***int strspn(const char *s1, const char *s2);*** — **s1** sətrinin maksimal başlanğıc altsətrinin yalnız **s2** sətrinin simvollarını özündə saxlayan uzunluğunu qaytarır.

в ***char *strstr(const char *s1, const char *s2);*** — **s1** sətrində **s2** sətrini axtarır. **S2** sətrinə girişin birinci simvolunun ünvanını qaytarır. Əgər belə bir sətir yoxdursa, sıfır qaytarır.

■■***char *strtok(char *s1, const char *s2);*** — **s1** başlanğıc sətrini **s2** sətrinin bir və ya bir neçə simvolları ilə leksemlere (altsətirlərə) bölür.

В ***double atof(const char *s);*** — **s** sətrini **double** tipli sürüşən nöqtəli ədədə çevirir.

Başlıq fayl — *math.h*

■■***int atoi(const char *s);*** — **s** sətrini **int** tipli ədədəd çevirir. Əgər sətiri çevirmək mümkün deyilsə, qiymətı və ya sıfır qaytarır. Başlıq faylı — ***stdlib.h***

■■***long atol(const char *s);*** — **s** sətrini **long** tipli ədədə çevirir. Əgər sətiri çevirmək mümkün deyilsə, qiyməti və ya sıfır qaytarır. Başlıq faylı — ***stdlib.h***

■■***char *itoa(int value, char *s, int radix);*** — **value** tipli tam ədədi **s** sətrinə çevirir. Göstəricini nəticə sətrinə qaytarır. **radix**— çevrilmələr (2-dən 36-ya qədər) vaxtı istifadə olunan hesab sisteminin əsasıdır. Başlıq faylı — ***stdlib.h***

4. C dilində sətirlərlə iş. Nümunələr

Bundan əvvəlki mövzu sətir funksiyalarına həsr olunmuşdu, indi isə onlar üzərində iş haqqıda söhbət açaq.

Sətirlərin uzunluğunun təyini

Sətrin uzunluğunu təyin etmək asandır. Bunun üçün ***strlen()*** funksiyasına sətir göstəricisini ötürmək lazımdır, hansı ki, öz növbəsində simvollarla ifadə olunmuş sətir uzunluğunu qaytaracaq. Aşağıdakı halda

```
char *c = "Any old string....";
int len;
```

növbəti operator ***len*** dəyişənini ***c*** göstəricisinin ünvanladığı sətrin uzunluğuna bərabər təyin edəcək:

```
len = strlen(c);
```

strlen() funksiyasının istifadəsinə aid nümunə

```
#include <stdio.h>
#include <string.h>
#include    <iostream>
using  namespace  std;
const  int  MAXLEN=256;
void main()
{
```

```
    char string[MAXLEN]; /* 255 simvol üçün yer.
    */ cout << "Input string:: ";
    gets(string);
    cout <<"\n"; /* yeni sətri başlamaq. */
    cout << "String: " << string << "\n";
    cout << "Length = " << strlen(string);
}
```

Burada ***gets()*** funsiyasından daxil etməni qəbul etmək üçün ***string*** adlanan sətir dəyişəni təyin olunur. Sətri daxil edən kimi proqram ***string*** dəyişənini simvollarla ifadə olunmuş sətir uzunlugunu hesablayan ***strlen()*** funksiyasına ötürülür.

strlen() funksiyasına sətirlərin başqa növlərini də ötürmək olar. Məsələn, simvol buferini aşağıdakı şəkildə təyin və qiymətləndirmək olar:

```
char buffer[128] = "Copy in buffer";
```

Sonra isə buferə köçürülmüş hərfi sətirdə olan simvolların sayına bərabər olan len dəyişənini təyin etmək üçün strlen() funksiyasından istifadə edilir:

```
int len; /* tam dəyişəni təyin etmək . */
len = strlen(buffer); /* Sətrin uzunluğunu
hesablamaq. */
```

Sətirlərin köçürülməsi

Sətirlər üçün mənimsətmə operatoru təyin edilməmişdir. Əgər c1 və c2 — simvol massivləridirsə, onlardan birini digərinə aşağıdakı şəkildə köçürə bilməzsiniz:

```
c1 = c2; //???
```

Əgər ***c1*** və ***c2*** ***char*** *******, tipi kimi elan olunarsa, kompilyator bu operator ilə razılaşacaq, lakin çətin ki, gözlədiyiniz nəticəni alasınız. Simvolları bir sətirdən digərinə köçürmək əvəzinə ***c1 = c2*** operatoru ***c2*** göstəricisini ***c1*** göstəricisinə köçürəcək, bu halda o, göstəricinin ünvanladığı informasiyanı itirərək ünvanı **c1**-ə yazacaq**.**

```
char*c1 = new char [10];
char*c2 = new char [10];
c1=c2; // c1-in altında ayrılmış yaddaş itdi.
```

Bir sətiri digərinə köçürmək üçün mənimsətmə operatoru əvəzinə ***strcpy()*** sətirləri köçürmə funksiyasından istifadə edin. ***char **** tipli ***c1*** və ***c2*** göstəriciləri üçün

```
strcpy(c1, c2);
```

operatoru **c2** göstəricisinin ünvanladığı simvolları sonuncu sıfırlar da daxil olmaqla **c1** göstəricisinin ünvanladığı yaddaşa köçürür. Qəbul edən sətrin surətinin saxlanılması üçün kifayət qədər yerə malik olmasına görə siz məsuliyyət daşıyırsınız.

strncpy() analoji funksiyası köçürülən simvolların sayini məhdudlaşdırır. Əgər ***(source)*** mənbəyi və ***(destination)*** qəbul edicisi **char** tipli göstərici və ya simvol massivləridirlərsə, onda

```
strncpy(destination, source, 10);
```

operatoru ***source*** göstəricisinin ünvanladığı sətirdən **destination** göstəricisinin ünvanladığı yaddaşa 10-adək simvol köçürəcək. Əgər **source** sətiri 10-dan çox simvola malikdirsə, onda nəticə kəsilər. Əgər azdırsa — nəticənin istifadə olunmayan baytları sıfır kimi təyin olunur.

> ***Qeyd:*** *Adında əlavə n hərfi olan sətir funksiyaları onun işini məhdudlaşdıran ədədi parametr verirlər. Bu funksiyalar təhlukəsizdirlər, lakin, adında n hərfi olmayan analoqlarına nisbətən ləngdirlər. Proqram nümunələri aşağıdakı funksiya cütlərinə malikdirlər:* ***strcpy() u strncpy(), strcat() u strncat(), strcmp() u strncmp().***

Sətirlərin birləşdirilməsi

İki sətrin birləşməsi, birinin digərinin sonuna əlavə edilməsi deməkdir, bu vaxt yeni, daha uzun sətir yaranır. Sətir verilərkərən

```
char original[128] = "Test ";
```

bu operator

```
strcat(original, " one, two, three!");
```

başlanğıc *original* sətrininin qiymətini «Test one, two, three!» sətrinə çevirəcək.

strcat() funksiyasına müraciət edəndə əmin olun ki, ***char **** tipli birinci arqument qiymətləndirilmişdir və nəticəni yadda saxlamaq üçün kifayət qədər yerə malikdir. Əgər ***c1*** artıq dolu olan sətri, ***c2*** isə sıfır olmayan sətri ünvanlayırsa, ***strcat(c1, c2)***; operatoru ciddi xəta əmələ gətirərək sətrin sonunu yenidən yazacaq.

strcat() funksiyası nəticə sətrin (onun birinci parametri ilə üst-üstə düşən) ünvanını qaytarır və bir neçə funksiyaya ardicil müraciət kimi istifadə oluna bilər:

```
strcat(strcat(c1,c2),c3)
```

Növbəti nümunə bir sətirdə məsələn, verilənlər bazasının sahəsi şəklində ayrıca saxlanılan ad, soyad, atasının adını birləşdirmək üçün ***strcat()*** funksiyasından necə istifadə etmək lazım olduğunu göstərir. Soyad, ad və atasının adını daxil edin. Proqram sizin daxil etdiyiniz sətirləri bir-birinə birləşdirəcək və ayrıca sətir kimi göstərəcək.

```
#include <iostream>
#include  <string.h>
using namespace std;
void main()
{
    // üç sətiri daxil etmək üçün yaddaşda yerin
    ayrilması.
    char *fam = new char[128];
    char *im = new char[128];
    char *otch = new char[128];
    //verilənlərin daxil edilməsi.
    cout << "Enter" << "\n";
    cout << "\tSurname: ";
    cin >> fam;
    cout << "\tName: ";
```

```
    cin >> im;
    cout << "\tLastname: ";
    cin >> otch;
    //Nəticə üçün yaddaşda yerin ayrılması.
    //Iki boşluq və sıfır simvolunu nəzərə almaq
lazımdır.
    char *rez=new char[strlen(fam) + strlen(im) +
         strlen(otch)+3];
    //Nəticənin yığılması
    strcat(strcat(strcpy(rez,fam)," "),im);
    strcat(strcat(rez," "),otch);
    //Yaddaşın topluya qaytarılması.
    delete [] fam;
    delete [] im;
    delete [] otch;
    //Nəticənin verilməsi.
    cout << "\nResult: " << rez;
    delete [] rez;
}
```

Nümunə kimi verilmiş proqram sətirlərin birləşdirilməsinin vacib prinsipini nümayiş etdirir: həmişə sətrin birinci arqumentini qiymətləndirin. Verilmiş halda ***rez*** simvol massivi ***fam***-ı ***rez***-ə daxil edən ***strcpy()*** funksiyasına müraciət ilə qiymətləndirilir. Bundan sonra proqram boşluqları və iki ***im*** və ***otch*** sətirlərini əlavə edir. Birinci arqumenti qiymətləndirilmiş olan ***strcat()*** funksiyasına heç vaxt müraciət etməyin.

Əgər altsətirlərin birləşdirilməsi üçün sətirdə kifayət qədər yerin olmasından əmin deyilsinizsə, onda ***strncat()*** funksiyasına müraciət edin. Bu funksiya ***strcat()*** funksiyasına analojidir, lakin köçürülən simvolların sayını müəyyən edən rəqəm tipli arqumenti tələb edir. ***char ** *** tipli göstərici, ya da simvol massivləri olan ***s1*** və ***s2***, sətirləri üçün,

```
strncat(s1, s2, 4);
```

operatoru ***s2***-dən maksimum dörd simvolu ***s1***-in sonuna birləşdirir. Nəticə mütləq sıfır simvolu ilə bitir.

Təhlükəsiz birləşdirmə üçün ***strncat()*** funksiyasından istifadənin bir üsulu vardır. Bu üsul ***strncat()*** fuksiyasına üçüncü arqument kimi, qəbul edən sətrin boş olan yaddaşının ölçüsünün ötürülməsindən ibarətdir. Aşağıdakı nümunəyə baxaq:

```
const int MAXLEN=128 char
s1[MAXLEN] = "Cat"; char
s2[] = "in hat";
```

Siz ***s2***-ni ***s1***-ə birləşdirərək ***strcat()*** funksiyasının köməyi ilə ***«Cat in hat»*** sətrini düzəldə bilərsiniz:

```
strcat(s1, s2);
```

Əgər siz nəticəni yadda saxlamaq üçün ***s1***-in kifayət qədər yerə malik olmasından əmin deyilsinizsə, onda alternativ operatordan istifadə edin:

```
strncat(s1, s2, (MAXLEN-1)-strlen(s1));
```

Bu üsul zəmanət verir ki, ***s2***-ni uyğun ölçüyə qədər kəssək belə ***s1*** həddi aşmayacaq. Əgər ***s1*** - sıfır sətirdirsə, bu operator yaxşı işləyər.

Adətən proqramlar ayrıca simvollar üzrə və yaxud alt sətirlər üzrə axtarış aparıllar.

Əsasən də faylın adını verilmiş genişlənmə üzrə axtararkən bu baş verir. Məsələn, istifadəçi faylın adını daxil etdikdən sonra yoxlanılır ki, o, .TXT genişlənməsini daxil etmişdir ya yox. Əgər bu belədirsə, .EXE genişlənməsi üçün yerinə yetirilən əməliyyatdan fərqli iş aparılacaq.

Ehtimal ki, siz arzuolunmayan tipdə olan verilənlər faylının yüklənməsi ilə əlaqədar olaraq əmələ gələn xətanın qarşısını almaq üçün müəyyən etdiyiniz genişlənmədən başqa bütün digər genişlənmələri rədd etmək istəyərsiz.

Simvolların axtarışı

strchr() funksiyasının istifadəsinə aid nümunə

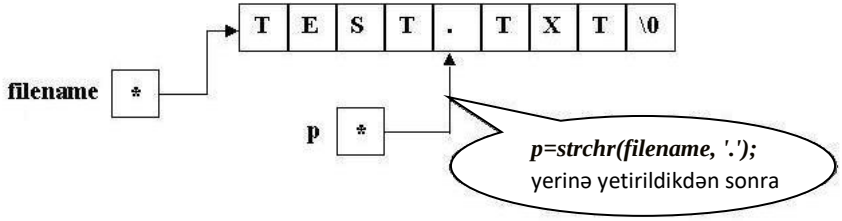
```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;
void main()
{
    char *filename = new char[128];
    cout << "Enter name of file: ";
    gets(filename);
    cout << "\nName of file: " << filename <<
    "\n"; if (strchr (filename,'.'))
        cout << "Name has extension" << "\n";
    else
        strcat (filename,".TXT");
    cout << "Name of file: " << filename <<
    "\n"; delete [] filename;
}
```

Verilmiş proqram daxil olunan sətrin simvolları arasında nöqtəni axtarmaqla faylın adındakı genişlənməni tapır. (Faylın adında (əgər nöqtə varsa) genişlənmədən əvvəl gələn nöqtə

ola bilər). Bu proqramda aşağıdakı operaotr əsasdır.

```
if (strchr (filename,'.'))
    cout << "Name has extension" << "\n";
else
    strcat (filename,".TXT");
```

strchr (filename,'.') ifadəsi ***filename*** göstəricisinin ünvanladığı sətirdəki nöqtə simvoluna qaytarır. Əgər belə simvol tapılmasa, ***strchr()*** funksiyası sıfır qaytarır. Sıfır olmayan ifadələr «doğru»nu bildirdiyindən, siz ***strchr()*** funksiyasından «doğru»/«yanlış» qiymətinin qaytarıcısı kimi istifadə edə bilərsiniz. Siz həm də ***strchr()*** funksiyasını göstəricinin verilmiş simvoldan başlayan altsətrə mənimsədilməsi üçün də istifadə edə bilərsiz. Məsələn, əgər ***p char **** kimi verilmiş göstəricidirsə, və ***filename*** gösətiricisi ***TEST.TXT*** sətrini ünvanlayırsa, onda ***p=strchr(filename, '.');*** operatorunun nəticəsi şəkildəki kimi olar.



Şəkil göstəricinin tam olmayan sətrə deyil, onun hissəsinə - altsətrə ünvanlanmasını nümayiş edir. Belə göstəricilər ilə ehtiyatla davranmaq lazımdır. Şəkildə yalnız

Sıfır baytı ilə qurtaran bir sətir, ***TEST.TXT***, və ***filename*** və ***p*** göstəriciləri verilmişdir. ***filename*** göstəricisi tam sətri ünvanlayır. p göstəricisi isə həmin simvol yığımının daxilindəki altsətri ünvanlayır. Sətir funksiyaları öz ilk simvollarından əvvəl gələn baytların qayğısına qalmır. Ona görə də

```
puts(p);
```

operatoru .***TXT*** altsətrini başqa bir sətrin hissəsi kimi deyil, sanki tam sətir dəyişəni kimi əks etdirir.

C dilində proqramlaşdırmada eyni tam sətrin altsətirlərinə ünvanlanan bir çox göstəricilərin istifadəsində qeyri adi heç nə yoxdur. Amma şəkildə göstərilən sətir, toplu daxilində yerləşdiyindən,

```
delete [] p;
```

operatoru p gösəricisinin ünvanladığı alsətri boşaltmağa cəhd edərkən, çətin aşkar olunan dərəcələrə aid xəta yaranır və toplunun dağılmasına gətirib çıxardır.

strchr() funksiyası sətirdəki ilk simvolu axtarır.

```
char *p;
char s[]="Abracadabra";

p = strchr(s,'a');
```

Operatorlarının verilməsi p göstəricisinə «Abracadabra» sətrindəki 'a' birinci yazı hərfinə olan ünvanı mənimsədir.

strchr funksiyası sətrin sonuncu sıfırına qiyməti olan simvol kimi baxır. Bu faktı nəzərə alaraq, sətrin sonunun ünvanını öyrənmək olar. Əvvəlki verilmələri nəzərə alaraq

```
p = strchr(s,0);
```

operatoru p göstəricisini «Abracadabra» sətrinin sonundakı «bra» altsərinə eyni olaraq ünvanlayır.

Altsətirlərin axtarışı

Sətirdəki simvollardan başqa, siz həm də altsətirlərin axtarışı ilə də məşğul ola bilərsiniz. Aşağıdakı nümunə bu metodu nümayiş etdirir. Bu proqram əvvəlki ilə analojidir, lakin faylın ***.TXT***. genişlənməsini təyin edir.

```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;
void main()
{
    char *filename = new
    char[128],*p; cout << "Enter name
    of file: "; gets(filename);
    cout << "\nName of file: " << filename <<
    "\n"; strupr(filename);
    p = strstr
    (filename,".TXT"); if (p)
        cout << "Name has extension" << "\n";
    else
    {      p      =      strchr
        (filename,'.'); if (p)
    *p=NULL; // İstənilən başqa genişlənməni silmək.
```

```
        strcat (filename,".TXT");
    }
    cout << "Name of file: " << filename <<
    "\n"; delete [] filename;
}
```

Bu proqram mütləq .TXT. genişlənməsi ilə bitən faylın adını yaradır. Faylın adında bu genişlənmənin olduğunu müəyyən etmək üçün aşağıdakı operator yerinə yetirilir.

```
p = strstr (filename,".TXT");
```

strchr() funksiyası kimi, ***strstr()*** funksiyası da başlanğıc sətir tapılmadıqda altsətrin ünvanını və yaxud sıfrı qaytarır. Əgər hədəf aşkar olunmayıbsa, ***p*** göstəricisi onun ünvanı ilə eyni təyin olunur, verilmiş nümunədə bu, ***.TXT***. altsətrindəki nöqtənin ünvanıdır. Genişlənmə yazı hərfləri ilə də verilə bildiyindən proqram aşağıdakı operatoru yerinə yetirir ki,

```
strupr(filename);
```

strstr() funksiyasına müraciət etməzdən əvvəl orijinal səitr yazı hərflərinə çevirsin.

Nümunə həm də sətirin verilmiş simvolun və ya altsətrin mövqeyində kəsilmə üsulunu nümayiş etdirir. Burada ***strstr()*** funksiyasına müraciət olunur ki, p göstəricisini ***filename*** sətrindəki birinci nöqtənin ünvanına yerləşdirsin. Əgər bu axtarışın nəticəsi sıfır deyilsə, onda nöqtənin əvəzinə sıfır baytını yazan operator yerinə yetiriləcək:

```
*p = NULL;
```

Bununla da yeni sətirsonu əvvəllər faylın genişlənməsinin olduğu yerə birləşdiriləcək. İndi artıq sətir, ***strcat()*** funksiyasına müraciət etməklə yeni genişlənmənin əlavə edilməsinə hazırdır**.**

5. Yeni dərsə aid məsələ nümunələri

Verilmiş bölmədə, biz göstəricilərin istifadəsinə aid bir neçə nümunələr hazırlamışıq.

x sözündə bütün «a» hərflərinin «ky» birləşməsi ilə əvəz olunmasına aid proqram.

```
#include <string.h>
#include <stdio.h>
void main ()
{
    /* k - Original massiv üzrə gəzən dəyişən
     i - Nəticə massivi üzrə gəzən dəyişən
       n - orijinal massivin uzunluğu
     */
    int k=0,i=0, n;
    /*
       x1 - Original massiv
       x2 - nəticə massiv (orijinal massiv tam olaraq
       'a' hərfləri ilə dolduğu halda, 2 dəfə çox)
       px1 - original massiv üzrə yerdəyişmə üçün
       göstərici
       px2 - nəticə massivi üzrə yerdəyişmə üçün
       göstərici
     */
    char x1[40],x2[80],*px1,*px2;

    //Orijinal massivin daxil olunması üçün sorğu

     puts( "Enter word (max 39 letters) ");
```

```
    gets(x1);

    /*
Original massivin əvvəlinin və nəticə massivin
ünvanlarını göstəriciyə yazırıq */
    px1 = x1;

    px2 = x2;

    /*
      Orijinal massivin real uzunluğunu
      hesablayırıq
     */
    n = strlen(x1)+1;

    // Dövr tək-tək elementlər üzrə original
       massivi seçmələyir
    while  (k<n)
    {
        // Əgər cari elementin qiyməti 'a' ilə
           üst-üstə düşmürsə //
        if (*(px1+k)!='a')
        {
            // Cari elementi nəticə massivinə
               köçürürük //
            *(px2+i) = *(px1+k);
            // Növbəti elementə keçirik  //
             i++;
            k++;
        }
        // Əgər cari elementin qiyməti 'a' ilə
           üst-üstə düşürsə //
        else
        {
            // 'k'  simvolunu  nəticə  massivin
               cari mövqeyinə yazırıq //
```

```
            *(px2+i) = 'k';
            // 'y' simvolunu nəticə massivin növbəti
               mövqeyinə yazırıq //
            *(px2+i+1) = 'y';
            // Orijinal massivin növbəti elementinə
               keçirik //
            k++;
            // Nəticə massivin bir elementinin
               üstündən atılırıq //
            i += 2;
        }
    }
    // nəticə massivi nümayiş edirik //
    puts(x2);
}
```

x sözündə bütün «ky» birləşmələrinin «a» hərfi ilə əvəz olunmasına aid proqram.

```
#include <string.h>

#include <stdio.h>

void main ()
{
    /*
       k – Original massiv üzrə gəzən dəyişən
       i – Nəticə massiv üzrə gəzən dəyişən
       n – Original massivin uzunluğu
    */

    int k=0,i=0, n;
    /*
       x1 – Original massiv
       x2 – Nəticə massiv
```

```
        px1 - original massiv üzrə
         yerdəyişmə üçün göstərici
          px2 - nəticə massiv üzrə
         yerdəyişmə üçün göstərici
 */

char x1[40],x2[40],*px1,*px2;

// Original massivin daxil olunması üçün
sorğu
 puts( "Enter word (max 39 letters)
"); gets(x1);

// Original massivin əvvəlinin və nəticə massivin
   ünvanlarını göstəriciyə yazırıq
 */
px1 = x1;

px2 = x2;

/*
  Original massivin real uzunluğunu hesablayırıq
 */
n = strlen(x1)+1;

// Dövr tək-tək elementlər üzrə original massivi
seçmələyir while (k<n)
{
    // Yoxlayırıq, əgər original massivin cari
    mövqeyində "ky" hərf birləşmısində iki simvol
    üst-üstə düşməsə
     if (strncmp((px1+k),"ky",2)!=0)
    {
        // Sadəcə cari mövqedən nəticə massivinə
           bir simvol köçürürük və bir simvol
           irəli hərəkət edirik
        *(px2+i++) = *(px1+k++);
    }
```

```
            // əgər original massivin cari mövqeyində
               iki simvol "ky" hərf birləşməsi ilə üst
               üstə düşürsə
        else
        {
            //nəticə massivinə 'a' simvolunu yazırıq
            və bir simvol irəli keçirik. Original
            massivdə isə iki simvol irəli keçirik.
            *(px2+i++) = 'a';
            k += 2;
         }
    }
    // Nəticə massivini nümayiş edirir
    puts(x2);}
```

x sözünün hər hərifini təkrarladan proqram.

```
#include <string.h>
#include <stdio.h>
void main ()
{
    // n - Original massivin uzunluğu *2
    int n;
    /*
      x1 - Original massiv
      x2 - Nəticə massiv (iki dəfə böyük)
         px1 - original massiv üzrə yerdəyişmə üçün
        göstərici
       px2 - nəticə massivi üzrə yerdəyişmə üçün
      göstərici
     */
    char x1[40],x2[80],*px1,*px2;
```

```
   //Original massivin daxil olunması üçün sorğu
   puts( "Enter word (max 39 letters) ");
   gets(x1); /*
        Original massivin əvvəlinin və nəticə
        massivin ünvanlarını göstəriciyə yazırıq
     */
     px1 = x1;
     px2 = x2;
     /*
       Original massivin ikiqat uzunluğunu
       hesablayırıq
      */
     n = 2*strlen(x1);

     //Nəticə massivinin axirinci elementinə '\0'
yazırıq
     *(px2+n) = '\0';

     //Dövr tək-tək elementlər üzrə orijinal massivi
seçmələyir
     while  ((*px1)!='\0')
     {
     /*Qiyməti, original massivin cari movqeyindən
nəticə massivin cari mövqeyinə yazırıq, axırda isə
bir element irəli keçirik*/
         *px2++ = *px1;
      /*Qiyməti, original massivin cari movqeyindən
nəticə massivin cari mövqeyinə yazırıq və hər iki
massivi bir element irəli keçiririk*/
         *px2++ = *px1++;}
     // Nəticə massivini nümayiş ediririk
     puts(x2);
 }
```

6. Ev tapşırığı

1. İstifadəçi sətiri klaviaturadan təyin edilmiş massivə daxil edir. Massivdə neçə elementinin dolu və boş olduğunu yoxlamaq lazımdır.
2. İstifadəçinin daxil etdiyi sətirin m-dən n-ə qədər elementlərini ekranda göstərmək və verilmiş parçanı başqa massivə yazmaq. (m və n də istifadəçi tərəfindən daxil olur)
3. m-dən n-ə qədər olan simvolları silmək, sətiri yenidən yazmaq və onu ekranda göstərmək.
4. İstifadəçi ayrıca sətir və simvol daxil edir. Bütün üst-üstə düşmələrin nömrələrini sıra ilə ekrana çıxardmaq. (nömrələmə birdən başlayır).
5. İstifadəçi ayrıca sətir və simvol daxil edir. Axırıncı üst-üstə düşmənin nömrəsini ekrana vermək lazımdır (nömrələmə birdən başlayır).