19-MA'RUZA. BELGILAR MASSIVI. CHAR TURIDAGI SATRLAR

1. char turidagi satrlar

Oldingi mavzularda deyarli har bir dasturimizda satrlar bilan ishladik. Aniqrog'i, satrli konstantalarni ekranda chop etdik. Bu esa " " belgilari orasida joylashgan belgilar ketma-ketligi. Biz tez-tez ekranda u yoki bu ma'lumotlarni ko'rsatishga majbur bo'ldik. Masalan:

```
cout<<"Bu oddiy matn";
```

Ushbu berilgan satr konstanta hisoblandi. Qo'shtirnoqlar satr konstantasining boshi va oxirini aniqlash uchun ishlatiladi va uning tarkibiy qismi emas.

Ko'pincha, dastur davomida ba'zi bir qisqa xabarlarni chop etishgina emas, balki ma'lum bir matn bilan ishlash, uni biron bir joyda saqlash, unga murojaat qilish va kerak bo'lsa tahrirlash zarur. Dasturda, masalan, yuqorida yozilgan satrga doimiy kira olmaymiz (masalan, uning nomini ham, xotiradagi manzilini ham bilmaymiz). Endi siz C++ da satrlar bilan ishlash usullaridan biri haqida bilib olasiz. Keyinchalik yana bir usul - string sinfidan foydalanish bilan tanishamiz.

Shunday qilib, birinchisi haqida: C++ da satrlarni saqlash uchun **belgilar massivi** ishlatiladi. Bular bir xil massivlardir, chunki biz C++ dagi massivlar haqida maqolada ko'rib chiqdik, ammo ular raqamli ma'lumotlarni emas, balki belgilar ma'lumotlarini saqlaydi. Siz qo'shni xotira yacheykalarida ketma-ket joylashtirilgan bunday massivning belgilarini tasavvur qilishingiz mumkin - har bir yacheyka bitta belgini saqlaydi va bitta baytni egallaydi. Bir bayt joy egallashining sababi belgilar massivining har bir elementi char tipida ekanligidadir. Har bir bunday satrning oxirgi belgisi '\0' belgisi (nol belgi). Masalan:

```
char satr[11]={'D', 'a', 's', 't', 'u', 'r', 'l', 'a', 's' , 'h', '\0'};
```

Matnning 10 ta belgidan iborat. Agar oxirgi yacheykada, '\0' nol belgi emas, balki . (nuqta) bo'lsa, - bu kompilyator uchun endi satr bo'lmay qoladi.

Oddiy massivlar kabi belgilar to'plami bilan ishlash kerak bo'ladi. Har bir yacheykaga ma'lumotlarni alohida yozish va uni belgi bo'yicha (sikl yordamida) namoyish etish mumkin:

```
#include <iostream>
using namespace std;
int main()
{
```

```
  char satr[11] ={'D', 'a', 's', 't', 'u', 'r', 'l', 'a', 's' , 'h', '\0'};

  for (int i =0; i < 11; i++)
  {
        cout << satr[i];
  }
  cout << endl;
  return 0;
}
```

C++ da simvollar massivi uchun satrlarni ishga tushirish va ularga kirishning juda qulay usul mavjud. Buning uchun massivning oxirgi belgisi nol belgisi bo‘lishi kerak.

Satr shu tarzda e'lon qilinadi - biz char tipidagi massiv yaratamiz, kvadrat qavsdagi o‘lcham ixtiyoriy (kompilyator uni hisoblab chiqadi), =operatori va ikkita tirnoq bilan kerakli belgini yozamiz. Ya’ni, biz massivni satrli konstanta bilan initsializatsiyalaymiz:

```
#include <iostream>
using namespace std;
int main()
{
  char str[] ="Dasturlash"; //'\0' bevosita mavjud bo‘ladi
  cout << str << endl;
  return 0;
}
```

Nol belgini kiritish shart emas. U to‘g‘ridan-to‘g‘ri mavjud va har bir satrli konstantaga avtomatik ravishda qo‘shiladi. Shunday qilib, biz qatorda 10 ta belgini ko‘rganimizga qaramay, massivning hajmi 11 ga teng bo‘ladi, chunki \0 ham belgi bo‘lib, xotiraning bir baytini egallaydi. U ushbu belgilar qatorining so‘nggi katagini egallaydi.

Ko‘rib turganingizdek, ekranda satrni chop etish uchun uning nomiga murojaat qilish kifoya:

```
cout << str << endl;
```

cout simvollar massivini chop etib, massiv katakchalarining birida satr oxiridagi belgi \0 ga duch kelguncha va chiqishni to‘xtatguncha bosib chiqaradi. Bunga oddiy belgilar massivi uchun ruxsat berilmaydi (\0 bo‘lmagan qator).

Endi quyida simvollar konstantasi (bitta tirnoqlarda - 'f', '@') va satrli konstanta ("f", "@" juft tirnoqlarda) o‘rtasidagi farqni qarab chiqamiz. Birinchisi uchun C++ kompilyatori tomonidan bir bayt xotira ajratadi. Qo‘shtirnoq bilan yozilgan belgi uchun ikkita bayt xotira ajratiladi - belgining o‘zi va nol uchun (kompilyator qo‘shadi).

Agar foydalanuvchi satrni klaviaturadan kiritishi kerak bo‘lsa-chi? Bunday holda, siz kiruvchi belgilarni saqlash uchun yetarli bo‘lgan hajmini ko‘rsatadigan **char** tipidagi satrni e'lon qilishingiz kerak, shu jumladan \0 uchun joy ajratishingiz kerak. Ushbu bo‘sh belgi haqida unutmang. Agar massivda 3 ta belgini saqlash kerak bo‘lsa, uning kattaligi yana bittaga oshirish kerak bo‘ladi, ya’ni 4 ta.

```
#include <iostream>
using namespace std;

int main()
{

  char saytNomi[15] ="";
  cout << "Sayt nomini kiriting: ";
  cin >> saytNomi;
  cout << saytNomi << endl;
  return 0;
}
```

Initsializatsiya paytida bo‘sh tirnoqlardan foydalanib, biz massivdagi har bir elementni \0 ga belgilaymiz. Shunday qilib, satr boshqa dasturlardan "qoldiq" dan tozalanadi. Agar foydalanuvchi kamroq belgilar bilan nom kiritgan bo‘lsa ham, to‘ldirilmagan yacheykalar \0 bilan to‘ladi. Bu yuzaga kelishi mumkin bo‘lgan xatoliklardan xalos qiladi. Xotirada ushbu satr quyidagicha bo‘ladi:

Masalan, klaviaturadan saytNomi o‘zgaruvchisiga www.samdu.uz matnini kiritsak, xotirada u quyidagicha joylashadi:

Yacheyka indeksi	0	1	2	3	4	5	6	7	8
Simvollar	w	w	w	.	s	a	m	d	u

Agar simvollar massivining 9-indeksining qiymatini quyidagicha o‘zgartirsak,

```
saytNomi[9]='\0';
```

ekranda quyidagi satrni olish mumkin bo‘ladi:

www.samdu

Nolinchi belgi ekranda ko‘rsatilganda asosiy rolni o‘ynaydi va undan keyin turgan birorta simvol ekranda chop etilmaydi.

Klaviaturadan satrlarni kiritish haqida batafsilroq gaplashamiz. Aslida, biz bilishimiz kerak bo‘lgan va qanday qilib hal qilishni o‘rganishimiz kerak bo‘lgan ba’zi muammolar mavjud. Quyida ko‘rib chiqadigan dasturimizda hammasi biz istaganimizchalik osonlik bilan hal bo‘lmasligini ko‘rishimiz mumkin:

```
#include <iostream>
using namespace std;

int main()
{
  char S[128] ="";

  cout << "Siz istagan matningizni kiriting: ";
  cin >> S;
  cout << S << endl;

  return 0;
}
```

“S” o‘zgaruvchisiga quyidagi satr kiritaylik:

Bu oddiy matn

Bu yerda biz istalgan satrni kiritishimiz mumkin, lekin kiritilgan matnda bo‘sh(probel) belgisini kirita olmaymiz. Buning natijasida kiritgan satrni ekranda ko‘rsatganda faqat birinchi so‘zni ko‘ramiz. Buning sababi, klaviaturadan bo‘sh belgini kirita olmaymiz va cin bo‘sh joy, yangi satr va yorliqni satrning oxiri deb hisoblaydi. Ya’ni, bizning holatlarimizda, cin faqat birinchi so‘zni o‘qiydi va avtomatik ravishda satr oxiri belgisini qo‘shadi. Natijada ekranda faqat **Bu** satri chiqadi. Kiritilgan ma’lumotlarning qolgan qismi kirish navbatiga joylashtiriladi.

Ushbu muammo osongina hal qilinadi. C++da get() va getline() funksiyalari mavjud, ulardan biz cin bilan birga foydalanishimiz mumkin. Ular o‘xshash, lekin getline() tez-tez ishlatiladi. Dasturimizga getline() yozuvini qo‘shaylik:

```
#include <iostream>
using namespace std;

int main()
{
  char quote[128] ="";

  cout <<"Istalgan matnni kiriting:\n";
  cin.getline(quote, 128); // satr nomini va uning hajmini funksiyaga o‘tkazing
  cout <<quote <<endl;

  return 0;
}
```

Qavslar ichida biz funksiya uchun ikkita argumentni ko‘rsatdik - qaysi satrda belgilarni (satr nomi) o‘qish kerakligi va ushbu massivning kattaligi kiritiladi. cin.getline() funksiyasi <Enter> tugmasi bosilguncha yoki massiv kattaligidan oshguncha butun qatorni bo‘shliqlar va yorliqlarni o‘z ichiga olgan qatorga o‘qiydi. Yangi satr belgisi massivda saqlanmaydi, lekin uning o‘rniga nol belgisi qo‘yiladi.

2. char turidagi satrlar bilan ishlovchi standart funksiyalar

C++ tilidagi satrlar va belgilar massivlari bilan tanishib chiqqanimizdan so‘ng, ular bilan ishlashning eng keng tarqalgan funksiyalarini ko‘rib chiqamiz. Satrlarni qayta ishlash uchun analog dasturlarimizni yozamiz va parallel ravishda **cstring** kutubxonasining standart funksiyalaridan foydalanamiz (eski versiyalarida string.h). Shunday qilib, ularning qanday ishlashini taxminan tasavvur qilasiz. **cstring** kutubxonasining ayrim standart funksiyalarini misollar orqali ko‘rib chiqamiz:

strlen() –satr uzunligini hisoblaydi (\0 dan tashqari belgilar soni);
strcat() - satrlarni birlashtiradi;
strcpy() - satrlarni bir qatordan ikkinchisiga nusxalaydi;
strcmp() - ikkita massivni taqqoslaydi.

1) **strlen()**. Satrdagi simvollar sonini hisoblaydigan dastur matni quyidagicha bo‘ladi:

```
#include <iostream>
using namespace std;
```

```
int main()
{
  char S[128] ="";  //satrni saqlash uchun
  cout <<"Istalgan matnni kiriting:\n";
  cin.getline(S, 128);
  int satrM=0;      //simvollar soni
  while (S[satrM] !='\0')
  {
        satrM++;
  }
  cout <<"Satr "<<satrM<<" ta belgidan iborat\n";
  return 0;
}
```

Kod qismini strlen() funksiyasi bilan almashtirish bilan quyidagicha yozishimiz mumkin:

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
  char S[128] ="";

  cout <<"Istalgan matnni kiriting:\n";
  cin.getline(S, 128);

  cout <<"Satr "<<strlen(S)<<" ta belgidan iborat\n";

  return 0;
}
```

Ko‘rib turganingizdek, ushbu kod qisqaroq. Buning uchun qo‘shimcha o‘zgaruvchilarni e'lon qilish va sikldan foydalanish shart emas. cout chiqish oqimida biz funksiyaga satrni uzatdik - **strlen(ourStr)**. U ushbu satrning uzunligini hisoblab chiqdi va qiymatni dasturga qaytarib berdi. Oldingi analog kodda bo‘lgani kabi, \0 belgisi umumiy belgilar soniga kiritilmagan.

2) strcat(). Bitta satr oxiridan ikkinchi satrni qo‘shadigan dastur. Boshqacha qilib aytganda, u ikkita satrni birlashtiradi.

```
#include <iostream>
using namespace std;

int main()
{
  char S1[20] ="www.";
  char S2[] ="samdu.uz";
  cout <<"Birinchi satr: " <<S1 <<endl;
  cout <<"Ikkinchi satr:" <<S2 <<endl;

  int satrM =0; //birinchi satrning '\0' saqlanadigan yacheyka indekslari
uchun
  while (S1[satrM] !=0)
  {
        satrM++; //birinchi satrning oxirini izlash
  }

  int satrM2 =0; //0-yacheykadan boshlab ikkinchi satr belgilaridan o‘tish
  while (S2[satrM2] !=0)
  {     //birinchi satr oxiriga ikkinchi satr belgilarini qo‘shish
        S1[satrM] =S2[satrM2];
        satrM++;
        satrM2++;
  }

  cout <<"Birlashtirilgan satr:" <<S1 <<endl;

  return 0;
}
```

Quyida esa xuddi shu dasturni strcat() funksiyasi orqali bajaramiz.

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
  setlocale(LC_ALL, "rus");
```

```
  char S1[20] ="www.";
  char S2[] ="samdu.uz";
  cout <<"Birinchi satr: " <<S1 <<endl;
  cout <<"Ikkinchi satr:" <<S2 <<endl;

  strcat(S1 , S2);

  cout <<"Birlashtirilgan satr:" <<S1 <<endl;

  return 0;
}
```

Birinchisida ham, ikkinchi kodda ham nimaga e'tibor berish kerak −birinchi belgilar satri hajmi ikkinchi satr belgilarini joylashtirish uchun yetarli bo‘lishi kerak. Agar o‘lcham yetarli bo‘lmasa, dastur g‘ayritabiiy tarzda tugashi mumkin, chunki satr yozilishi birinchi massiv egallagan xotiradan chiqib ketadi.

```
char S1[6] ="www.";
strcat(S1, "samdu.uz");
```

Bunday holda, "C ++ ni biz bilan o‘rganing!" S1 qatoriga yozib bo‘lmaydi. Bunday operatsiya uchun unda bo‘sh joy etarli emas.

Agar siz Microsoft Visual Studio ishlab chiqish muhitining so‘nggi versiyalaridan birini ishlatayotgan bo‘lsangiz, quyidagi xatoga yo‘l qo‘yishingiz mumkin: “Error C4996: ‘strcat’: This function or variable may be unsafe. Consider using strcat_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.” Buning sababi shundaki, strcat funksiyasining yangi, yanada xavfsiz versiyasi allaqachon ishlab chiqilgan −strcat_s.

Buferni to‘ldirish yo‘qligiga ishonch hosil qiladi (ikkinchi qator yozilgan belgilar massivi). Muhit eskirgan funksiya o‘rniga yangi funksiyadan foydalanishni talab qiladi.

3) **strcpy()**. Bitta satrni ko‘chirib, boshqa satr o‘rniga joylashtiramiz.

```
#include <iostream>
using namespace std;
int main()
{
  char S1[64] ="Matn1";
```

```
  char S2[] = "Matn2";
  cout << "Birinchi satr: " << S1 << "\n";
  cout << "Ikkinchi satr: " << S2 << "\n";
  int satrM = 0;
  while (true) //Cheksiz sikl
  {
        S1[satrM] = S2[satrM]; //Simvol bo‘yicha ko‘chirish

        if (S2[satrM] == '\0') //agar ikkinchi satrda \0 topsak
        {
              break; //sikldan chiqish
        }
        satrM++;
  }
  cout << "Nusxalangan satr: " << S1 << "\n";
  return 0;
}
```

cstring kutubxonasining standart funksiyasini qo‘llaymiz:

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
  char S1[20] = "Matn1";
  char S2[] = "Matn2";
  cout << "Birinchi satr: " << S1 << "\n";
  cout << "Ikkinchi satr: " << S2 << "\n";
  strcpy(S1, S2);
  cout << "Nusxalangan satr: " << S1 << "\n";
  return 0;
}
```

4) strcmp() - ushbu funksiya shunday tuzilgan: ikkita satrni simvollar bo‘yicha taqqoslaydi. Agar satrlar bir xil bo‘lsa (ikkala satrda ham, ularning sonida ham), funksiya 0 raqamini dasturga qaytaradi, agar birinchi satr ikkinchisidan uzunroq bo‘lsa, u 1 raqamini dasturga qaytaradi, agar kamroq bo‘lsa, u holda -1 qaytaradi.

Satr uzunligi teng bo‘lganda -1 raqami ham qaytariladi, ammo satrlarning simvollari to‘g‘ri kelmaydi.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
  char S1[] = "Satr1";
  char S2[] = "Satr2";
  cout << "Birinchi matn: " << S1 << "\n";
  cout << "Ikkinchi matn: " << S2 << "\n";
  int taqqos = 0; //satrlar uzunligini taqqoslash uchun
  int satrM = 0;
  while (true)
  {
        if (strlen(S1) < strlen(S2))
        {
              cout << "Satrlar teng emas: " << --taqqos << endl;
              break;
        }
        else if (strlen(S1) > strlen(S2))
        {
              cout << "Satrlar teng emas: " << ++taqqos << endl;
              break;
        }
        else //agar satrlar belgilar soni bo‘yicha ham bir xil bo‘lsa
        {
              if (S1[satrM] == S2[satrM]) //\0 ni o‘z ichiga olgan simvollar
bilan taqqoslash
              {
                    satrM++;
                    if (S1[satrM] == '\0' && S2[satrM] == '\0')
                    {
                          cout << "Satrlar teng " << taqqos << endl;
                          break;
                    }
              }
              else //\0 ni o‘z ichiga olgan belgilar bilan taqqoslash
              {
```

```
                    cout <<"Satrlar teng emas: " << --taqqos << endl;
                    break;
                }
            }
        }
        return 0;
}
```

Ushbu dasturni strcmp() funksiya yordamida bajaramiz:

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
  char S1[] ="Satr1";
  char S2[] ="Satr2";

  cout <<"Birinchi matn: " << S1 <<"\n";
  cout <<"Ikkinchi matn: " << S2 <<"\n";


  cout << strcmp(S1, S2) << endl << endl;

  return 0;
}
```

Satrlar bilan ishlovchi qolgan funksiyalar bilan quyidagi 13-jadval orqali tanishib chiqish mumkin:

13-jadval.

Belgili massivlar bilan ishlovchi funksiyalar

Funksiya	Tavsif
strlen (s1)	nol belgini hisobga olmaganda, belgilangan satr uzunligini aniqlaydi
Satrlarni nusxalash funksiyalari	
strcpy(s1,s2)	s2 satrdan s1 satrga belgilarning bayt bo‘yicha nusxasini oladi

Strncpy(s1,s2,n)	s2 satrdan s1 satrga n ta belgining nusxasi oladi va s1 satrni qaytaradi
Satrlarni ulash funksiyalari	
Strcat(s1,s2)	s2 satrni s1 satr bilan birlashtiradi. s1 satrni qaytaradi
Strncat(s1,s2)	s2 satrning n ta belgisini s1 satr bilan birlashtiradi. s1 satr natija sifatida qaytadi
Satrlarni taqqoslash funksiyalari	
Strcmp(s1,s2)	s1 satrini s2 satr bilan taqqoslaydi va int tipidagi natijani qaytaradi: 0 - agar satrlar teng bo‘lsa, >0 - agar s1<s2, <0 - agar s1>s2. Harflar registrlari ham farqlanadi
Strncmp(s1,s2)	s1 satrining n ta belgisini s2 satri bilan taqqoslaydi va int tipidagi natijani qaytaradi: 0 - agar satrlar teng bo‘lsa, >0 - agar s1<s2, <0 - agar s1>s2 bo‘lsa. Harflar registrlari ham farqlanadi
stricmp(**s1,s2**)	s1 satrini s2 satr bilan taqqoslaydi va int tipidagi natijani qaytaradi: 0 - agar satrlar teng bo‘lsa, >0 - agar s1<s2, <0 - agar s1> s2. Harflar registrlari farqlanmaydi
strnicmp(s1,s2,n**)**	s1 satrining n ta belgisini s2 satri bilan taqqoslaydi va int tipidagi natijani qaytaradi: 0 - agar satrlar teng bo‘lsa, >0 - agar s1<s2, <0 - agar s1>s2 bo‘lsa. Harflar registrlari farqlanmaydi
Simvollarni qayta ishlash funksiyalari	
isalnum(c**)**	agar c harf yoki raqam bo‘lsa, true qiymatini qaytaradi, aks holda false qaytadi
isalpha(c**)**	Agar c harf bo‘lsa true, aks holda false qiymatini qaytaradi
isdigit(c**)**	Agar c raqam bo‘lsa true, aks holda false qiymatini qaytaradi
islower(c**)**	Agar c kichik harf bo‘lsa true, aks holda false qiymatini qaytaradi
isupper(c**)**	Agar c katta harf bo‘lsa true, aks holda false qiymatini qaytaradi
i**sspace(**c**)**	Agar c probellardan iborat bo‘lsa, true, aks holda false qiymatini qaytaradi
toupper(c**)**	Agar c satri quyi registrda bo‘lsa, u satrni yuqori registrga o‘tkazadi. Aks holda c satrini o‘zi qaytadi.
Izlash funksiyalari	

strchr(s,c**)**	s satridan c belgisini izlaydi, va c belgisining birinchi uchragan indeksini qaytaradi. Agar simvol topilmasa 0 qaytadi
strcspn(s1,s2**)**	s1 va s2 satrlarni solishtiradi va s1 satrining s2 satriga kirgan birinchi belgini indeksini qaytaradi.
Almashtirish funksiyalari	
atof(s1**)**	s1 satrini double tipiga o‘tkazadi
atoi(s1**)**	s1 satrini int tipiga o‘tkazadi
atol(s1**)**	S1 satrini long int tipiga o‘tkazadi
Standart kiritish-chiqarish kutubxonasi <stdio> funksiyalari	
getchar(c**)**	standart kirishdan c simvolini o‘qiydi, belgini int tipida qaytaradi
gets(s**)**	<ENTER> tugmachasi bosilgunga qadar s satriga standart kirishdan belgilar oqimini o‘qiydi