20-MA’RUZA. STRING SINFI. STRING TURIDAGI SATRLAR

C++ tilida standart satr turiga qo‘shimcha sifatida string turi kiritilgan va string sinfi ko‘rinishida amalga oshirilgan. Bu turdagi satr uchun ‘\0’ belgisi tugash belgisi hisoblanmaydi va u oddiygina belgilar massivi sifatida qaraladi. string turida satrlar uzunligining bajariladigan amallar natijasida dinamik ravishda o‘zgarib turishi, uning tarkibida bir qator funksiyalar aniqlanganganligi bu tur bilan ishlashda ma’lum bir qulayliklar yaratadi.

Xabarlarni qabul qilish va yuborish uchun ma’lumotlar bazalari, fayllar bilan ishlashda matnli ma’lumotlardan foydalanish har bir dasturchi ishining ajralmas qismidir.

Satrlar ichki maqsadlarda ham (sozlash va testlash) va umumiy muammolarni hal qilish, ma’lumotlarni mijozga uzatish va grafik interfeysni loyihalash uchun ishlatiladi. Shuni hisobga olgan holda, C++ tilida satrlarning qanday joylashtirilganligiga, ularning asosiy amallari bilan qanday ishlashiga va string tipidagi obyektlarni qayta ishlash usullarining qaysi birida ma’lum bir vaziyatda tezroq yoki maqsadga muvofiqroq bo‘lishiga alohida e'tibor berishingiz kerak.

C++ satrlari C massivida ishlatilgan char massivlari yoki C++ standart kutubxonasiga kiritilgan string sinfining nusxasi bo‘lishi mumkin. Shuni ta’kidlash kerakki, ushbu turdagi obyektni yaratishda biz deyarli dinamik ravishda o‘zgaruvchan satrni olamiz, ya’ni boshlang‘ich o‘lchamlarini o‘rnatishga hojat yo‘q, chunki ular obyekt bilan ishlash jarayonida o‘zgarishi mumkin.

Birinchi holda, satr xotirada massiv sifatida ajratiladi. Bunga char ko‘rsatkichi orqali kirish mumkin. Ushbu yondashuv xatoga yo‘l qo‘yadigan va juda qiyin, chunki u juda past darajada amalga oshiriladi.

String sinfining obyektlari ishlashni osonlashtiradi, ular orqali siz satrlar bo‘yicha standart operatsiyalarga kirishingiz mumkin. Ular, shuningdek, STL (Standart Template Library – standart shablonlar kutubxonasi) nom maydonining bir qismidir. Standart C kutubxonalaridan yanada ko‘proq foydalanish uchun sarlavha faylini kiritishingiz kerak:

```
#include <cstring>
```

Satrli o‘zgaruvchini ishlatishning eng oddiy misoli uni ekranda ko‘rsatishdir. Vazifani murakkablashtiramiz va birinchi navbatda, satrning qiymatini o‘qib, keyin uni ishlatishga harakat qilaylik.

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

```
int main()
{
  string myString; //Stringni e'lon qilish

  cout << "Ismingiz nima? ";
  getline(cin, myString);                    // satrni kiritish

  cout << "Assalomu alaykum, " << myString << "! \n";
}
```

getline() metodi cin oqimidagi ma’lumotlarni o‘qish va verguldan keyin ko‘rsatilgan o‘zgaruvchiga yozish imkonini beradi.

O‘zgaruvchini yaratish paytida uni initsializatsiyalash zarur bo‘lganda, quyidagi sintaksisdan foydalanish kerak:

```
string oneMoreString = "Bu oddiy matn";
```

String turidagi o‘zgaruvchilarni qanday e'lon qilish odatiy o‘zgaruvchiga o‘xshash tarzda amalga oshiriladi.

```
string s1;
string s2 = “Bu oddiy satr”;
s1 = s2;
s2 = “Bu yangi satr”;
```

string sinfi va char tipida satrlarning asosiy farqlari. Satrning yangi turini yaratilishi char turidagi satrlar bilan ishlashning kamchiliklari bilan bog‘liq edi. char turiga nisbatan string turi quyidagi asosiy afzalliklarga ega:

- standart C++ operatorlari (=, +, = =, <> va boshqalar) yordamida satrlarni qayta ishlash qobiliyati. Ma’lumki, char turidan foydalanganda, hatto eng oddiy satr amallari ham murakkab ko‘rinardi va ortiqcha dasturlash kodini yozishni talab qildi;
- dastur kodining ishonchliligini (xavfsizligini) ta’minlash. Masalan, satrlarni nusxalashda, string turi manba satrlari belgilangan satrdan kattaroq bo‘lsa sodir bo‘lishi mumkin bo‘lgan tegishli harakatlarni ta’minlaydi;
- satrni mustaqil ma’lumotlar turi sifatida taqdim etish. Satr turini **string** sifatida e'lon qilish dasturdagi barcha o‘zgaruvchilar uchun bir xildir, bu ma’lumotlarning izchilligini ta’minlaydi.

String turining **char** turiga nisbatan asosiy kamchiligi - ma’lumotlarni qayta ishlash tezligining sustligi. Buning sababi shundaki, **string** turi, aslida, konteyner sinfidir va sinf bilan ishlash dastur kodining qo‘shimcha bajarilishini talab qiladi, bu esa o‘z navbatida qo‘shimcha vaqt talab etadi.

string sinfi qulaydir, chunki u standart operatorlar yordamida satrlarni qulay tarzda boshqarishga imkon beradi.

string sinfidagi obyektlar bilan quyidagi operatorlardan foydalanish mumkin:

= – o‘zlashtirish
+ – birlashtirish (satrlarni birlashtirish)
+= – birlashtirish va ta’minlash
== – aynan tenglik
! = – teng emaslik
< – kichik
<= – kichik yoki teng
> – katta
> = - katta yoki teng
[] - indeksatsiya

Misol. Asosiy amallarning qo‘llanilishi doir misollar

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

int main()
{
   string s1 = "s-1";
   string s2 = "s-2";
   string s3;
   bool b;

   //Ta’minlash
   s3 = s1; // s3 = "s-1"

   //Ikkita satrni birlashtirish
   s3 = s3 + s2; // s3 = "s-1s-2"

   // Ta’minlash va birlashtirish
   s3 = "s-3";
   s3 += "abc"; // s3 = "s-3abc"
```

```
    // amal '==' - Satrlarni taqqoslash
    b = s2==s1; // b = false
    b = s2=="s-2"; // b = true

    // Satrlarni taqqoslash
    s1 = "s1";
    s2 = "s2";
    b = s1 != s2; // b = true
    s1 = "abcd";
    s2 = "de";
    b = s1 > s2; // b = false
    b = s1 < s2; // b = true
    s1 = "abcd";
    s2 = "ab";
    b = s1 >= s2; // b = true
    b = s1 <= s2; // b = false
    b = s2 >= "ab"; // b = true

    // Indeksatsiya
    char c;
    s1 = "abcd";
    c = s1[2]; // c = 'c'
    c = s1[0]; // c = 'a’
  return 0;
}
```

4. string sinfidagi asosiy standart funksiyalar

push_back(char) funksiyasi joriy satr oxiriga istalgan char belgisini qo‘shishga imkon beradi. Shu bilan birga, biz satrning yangi qiymatini qayta saqlashimiz shart emas, eski o‘zgaruvchi shunchaki o‘zgartiriladi.

Satr uzunligini aniqlash. Satr uzunligini olish uchun ikkita usuldan foydalanish mumkin. Ulardan biri length(), ikkinchisi size(). Ikkala usul ham raqamlarning sonini qaytaradi. Ammo shuni ta’kidlash kerakki, satrlar massividagi elementlarning raqamlanishi 0 dan boshlanadi.

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

```
  string myString;
  cout << "Ismingiz nima? ";
  getline(cin, myString);
  cout << "Satr, " << myString << "! \n";
  cout << "Satr uzunligi: "   << myString.size()   << "\n";
  cout << "Satr uzunligi: " << myString.length() << "\n";
}
```

string o‘zgaruvchisida ma’lumotlar mavjudligini tekshirish uchun mantiqiy qiymatni qaytaradigan **empty()** funksiyasidan foydalanish mumkin (agar matn mavjud bo‘lmasa 1 aks holda 0 qaytadi). Ushbu amal ma’lumotlar yo‘qotilishidan himoya qilish kerak bo‘lganda foydali hisoblanadi.

string o‘zgaruvchisini tozalash uchun **clear()** metodi qo‘llaniladi. Natijada, uning bajarilishidan keyin **myString.size()** 0 qiymatini qaytaradi va **myString.empty()** qiymati true qiymatini qaytaradi, ya’ni 1.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string myString = "Dasturlash asoslari";
  cout << "Matn bo‘shmi? " << myString.empty() << "\n";
  cout << "Matn o‘lchami: "   << myString.size()  << "\n";
  cout << "Matnni tozalash! \n";
  myString.clear(); // ma’lumot o‘chirildi
  cout << "Hozir matn bo‘shmi? " << myString.empty() << "\n";
  cout << "Matn o‘lchami:  " << myString.size() << "\n";
}
```

1) Satrlarni o‘zlashtirish metodlari. assign() funksiyasi.

Bir satrni boshqasiga tayinlash uchun ikkita usuldan birini qo‘llashingiz mumkin:

- Ta’minlash operatoridan foydalanish '=';
- string sinfidan **assign()** funksiyasidan foydalanish.

assign() funksiyasi bir nechta qayta yuklanish realizatsiyaga ega. Birinchi variant - funksiyani parametrlarsiz chaqirish:

```
string assign (void)
```

Bunday holda, bitta satrni boshqasiga oddiy tayinlash mavjud.
Ikkinchi variant sizga satr belgilarini satrdan nusxalashga imkon beradi:

string assign(const string S, size_type st, size_type num)

Bu yerda:
S – mavjud satr;
st – nusxalash boshlanadigan satrdagi belgi indeksi
num – nusxalanuvchi belgilar soni
size_type – ma’lumot turi

Misol. assign funksiyasi ishlatishga oid misol

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

int main()
{
   string s1 = "Dasturlash";
   string s2;
   string s3;

   s3 = s1; // 1-usul
   s2.assign(s1); // 2-usul
   s2.assign(s1, 0, 4); //
   return 0;
}
```

2) Satrlarni birlashtirish. append() funksiyasi

Satrlarni birlashtirish uchun **append()** funksiyasi ishlatiladi. Satrlarni qo‘shish uchun "+" amalidan ham foydalanishingiz mumkin.

Biroq, satrning ayrim qismini qo‘shishingiz kerak bo‘lganda, **append()** funksiyasi yaxshiroq. Funksiyani quyidagicha realizatsiya qilish mumkin:

string append(const string s, size_type start);

3) Satrga simvollarni joylashtirish. insert() funksiyasi

string sinfi bir vaqtning o‘zida ikkita satrni manipulyatsiya qilishga, qiymatlarni boshqasiga yozishga imkon beradi. Buning uchun satr obyektiga **insert**(int k, string from, int k1, int a). metodini chaqiring. k - bu yangi ma’lumotlar yozilgan satrda katakning boshlang‘ich pozitsiyasi, ya’ni ushbu indeksdan boshlab yangi qiymatlar joylashtiriladi.

string o‘zgaruvchisi from - bu ma’lumotlar olingan satr, keyin barcha parametrlar unga tegishli. Ma’lumotlarni nusxalashni boshlash indeksining boshlang‘ich qiymati k1 bilan belgilanadi. from satridan ko‘chiriladigan belgilarning umumiy soni boshlang‘ich indeksidan kelib chiqqan holda a bilan belgilanadi.

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

int main()
{
  string myString1 = "Programmalash";
  string myString2 = " asoslari";
  cout << myString1 << "\n";
  cout << myString2 << "\n";

  myString1.insert(13, myString2, 0, 9);
  cout << myString1;
}
```

4) Satrlar simvollarini almashtirish. replace() funksiyasi

replace() funksiyasi yordamida berilgan satrning simvollarini almashtirish mumkin:

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
   string s1 = "abcdef";
   string s2 = "1234567890";

   s2.replace(2, 4, s1); // s2 = "12abcdef7890"
```

```
    s2 = "1234567890";
    s2.replace(3, 2, s1); // s2 = "123abcdef67890"
    s2 = "1234567890";
    s2.replace(5, 1, s1); // s2 = "12345abcdef7890"


     s1 = "abcdef";
     s2 = "1234567890";

    s2.replace(2, 4, s1); // s2 = "12abcdef7890"
    s2 = "1234567890";
    s2.replace(3, 2, s1); // s2 = "123abcdef67890"

    s2 = "1234567890";
    s2.replace(5, 1, s1); // s2 = "12345abcdef7890"
    s2 = "1234567890";
    s2.replace(5, 1, s1, 2, 3); // s2 = "12345cde7890"
    s2 = "1234567890";
    s2.replace(4, 2, s1, 0, 4); // s2 = "1234abcd7890"
  return 0;
}
```

5) Satrdan belgilangan miqdordagi belgilarni olib tashlash. erase() funksiyasi.

Berilgan satrdan ma’lum bir miqdordagi belgilarni olib tashlash uchun **erase()** funksiyasidan foydalaniladi:

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    string s = "01234567890";
    s.erase(3, 5); // s = "012890"
    s = "01234567890";
    s.erase(); // s = ""
  return 0;
}
```

6) Satrdan simvollarni izlash. find() va rfind() funksiyalari

String sinfida ichki satrni qidirish ikki xil usulda amalga oshirilishi mumkin, ular qidirish yo‘nalishi bo‘yicha farqlanadi:

- find () funksiyasi yordamida satrni boshidan oxiriga qadar ko‘rib chiqiladi;
- rfind() funksiyasi yordamida esa satrni oxiridan boshiga qadar ko‘rib chiqiladi.

Misol. find() funksiyasini ishlatishga oid misollar.

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
   string s1 = "01234567890";
   string s2 = "345";
   string s3 = "abcd";
   int pos;

   pos = s1.find(s2); // pos = 3
   pos = s1.find(s2, 1); // pos = 3
   pos = s1.find("jklmn", 0); // pos = -1
   pos = s1.find(s3); // pos = -1
   pos = s2.find(s1); // pos = -1
  return 0;
}
```

Misol. rfind() funksiyasini ishlatishga oid misollar

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
  // find() va rfind()
   string s1 = "01234567890";
```

```
    string s2 = "345";
    string s3 = "abcd";
    string s4 = "abcd---abcd";
    int pos;

    pos = s1.rfind(s2); // pos = 3
    pos = s1.rfind(s2, 12); // pos = 3
    pos = s1.rfind(s2, 3); // pos = 3
    pos = s1.rfind(s2, 2); // pos = -1
    pos = s2.rfind(s1); // pos = -1
    pos = s1.rfind(s3, 0); // pos = -1

    // find() va rfind() funksiyalari orasidagi farq
    pos = s4.rfind(s3); // pos = 7
    pos = s4.find(s3); // pos = 0
  return 0;
}
```

7) Satrlarning qismlarini taqqoslash. compare() funksiyasi

string turi sinf bo‘lganligi sababli, == amalidan foydalanib, ikkita satrni bir-biri bilan taqqoslash mumkin. Agar ikkita satr bir xil bo‘lsa, unda taqqoslash natijasi **true** qiymatini qaytaradi. Aks holda, taqqoslash natijasi **false** bo‘ladi.

Agar bitta satrning bir qismi boshqasi bilan taqqoslash kerak bo‘lsa, u holda compare() funksiyasini ishlatish kerak.

Funksiya quyidagicha tartibda ishlaydi. Agar chaqiruvchi satr s satrdan kichik bo‘lsa, u holda funksiya -1 (manfiy qiymat) qaytaradi. Agar chaqiruvchi satr s satrdan katta bo‘lsa, funksiya 1 (musbat qiymat) qaytaradi. Agar ikkita satr teng bo‘lsa, funksiya 0 qaytaradi.

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    string s1 = "012345";
    string s2 = "0123456789";
    int res;

    res = s1.compare(s2); // res = -1
```

```
    res = s1.compare("33333"); // res = -1
    res = s1.compare("012345"); // res = 0
    res = s1.compare("345"); // res = -1
    res = s1.compare(0, 5, s2); // res = -1
    res = s2.compare(0, 5, s1); // res = -1
    res = s1.compare(0, 5, "012345"); // res = -1
    res = s2.compare(s1); // res = 1
    res = s2.compare("456"); // res = -1
    res = s2.compare("000000"); // res = 1
  return 0;
}
```

8) string turidagi satrdan char turidagi satrga o‘tish. c_str () funksiyasi.
c_str () funksiyasi '\0' belgisi bilan tugaydigan satrni olish uchun ishlatiladi.

```
string s = "abcdef";
const char * ps;
ps = s.c_str();
```

string sinfining afzalligi shundaki, ulardan foydalanish oson va ko‘plab usullarni qo‘llab-quvvatlaydi. Biroq, satrga mos keladigan obyektlar sekin ishlov beriladi. Ba’zi hollarda ma’lumotlar yo‘qolishi yoki satrning yaxlitligi buzilishi mumkin, keyin ularni ko‘rish va tekshirish imkoniyati bo‘lmaydi.