**№3-ma’ruza. Klass va funksiyalar shablonlari.** Klass va funksiyalar shablonlarini dasturlash. Virtual funksiyalar. Shablonlarning standart kutubxonasi(STL). STL –komponentalari

**Reja:**

1. Shablonlarning umumiy harakteristikasi

2. Sinf shablonini yaratish

3. Argument qabul qiluvchi sinf shabloni

4.Virtual funksiyalar

5. Shablonlarning standart kutubxonasi(STL).

***Kalit so‘zlari:****Sinf, sinf shabloni, funksiya shabloni, template, class, typename, argumentli shablon, argumentsiz shablon.*

**1. Shablonlarning umumiy xarakteristikasi**

Shablonlar ma’lumotlar turini aniqlashda ishlatiladigan kontruktsiya (funksiya, sinf) larni aniqlash uchun qo‘llaniladi, lekin shablon kodini yozishda bu ma’lumot turi qanday bo‘lishi nama’lum bo‘ladi. Boshqacha so‘z bilan aytganda, shablonlar aniq bir ma’lumot turiga bog‘liq bo‘lmagan konstruktsiyalarni ishlab chiqishda qo‘llaniladi.

Shablonlarni yozish qoidalari:

Funksiya shablonlari funksiya hisoblanmaydi, shuning uchun ham ular virtual bo‘lishi mumkin emas.

Sinf shablonlari statik elementlar, do‘stona funksiyalar va sinflarni o‘z ichiga olishi mumkin.

Shablonlar meros qoldiradi. Ular shablonlar va odatiy sinflardan olinishi, shuningdek, shablonlar va oddiy sinflar uchun tayanch shablon bo‘lishi mumkin.

Barcha sinf yoki funksiya shablonlari «template» so‘zi bilan boshlanadi. Ushbu kalit so‘zdan keyin shablon parametrlarini tavsiflash uchun burchakli < > qavslar qo‘yiladi. Har bir parametrdan oldin «class» yoki «typename» kalit so‘zlari ishtirok etadi. Bu kalit so‘zlarning yo‘qligi kompiliyator tomonidan sintaksis xato sifatida qaraladi.

C++ algoritmik tilida shablonlarni e’lon qilishga misollar:

*template <class T>*

*template <typename T>*

*template <typename T1, typename T2>*

sinf yoki funksiya shablonlarida int, double, float, char kabi ma’lumotlar turini qo‘llashda ham typename kalit so‘zi ishlatiladi. «class» kalit so‘zi funksiya shablonida parametr sifatida sinf qo‘llanilganini xabar qiladi.

C++ algoritmik tilida argumentsiz sinf shablonini e’lon qilishning umumiy shakli:

*template <class T>*

*class ClassName {*

*//sinf tanasi*

*//...*

*};*

*T – tur nomi;*

*ClassName* – sinf nomi.

Sinf shabloni obyektining C++ algoritmik tilida e’lon qilinishining umumiy shakli

*ClassName <type> ObjName;*

*ClassName – shablonli sinf nomi;*

*type – dasturdagi ma’lumotning aniq turi;*

*objName – sinf obyekti (nusxasi) nomi.*

Ma’lumot turi butun yoki haqiqiy son bo‘lgan sinf shablonining e’lon qilinishiga misol:

*template <class T>*

*class MyNumber {*

*public:*

*MyNumber(void) { } // jimlik holatidagi konstruktor*

*void Mult2(T\* t); // sonni 2 ga ko‘paytiruvchi funksiya*

*T MySquare(T); // T turidagi sonning kvadratini qaytaruvchi funksiya*

*T DivNumbers(T, T);  // T turidagi ikkitasonni bo‘lish va T turidagi natijasini qaytarish funksiyasi*

*};*

*// sonni 2 ko‘paytirish funkuiyasining tatbiqi*

*template <class T>*

*void MyNumber<T>::Mult2(T\* t) {*

*\*t = (\*t) \* 2;*

*}*

*// sonning kvadratini qaytaruvchi funksiya tatbiqi*

*template <class T>*

*T MyNumber<T>::MySquare(T number){*

*return (T)(number \* number);*

*}*

*// ikkita sonni bo‘lish va bo‘linma natijasini qaytarish funksiyasi*

*template <class T>*

*T MyNumber<T>::DivNumbers(T t1, T t2) {*

*return (T)(t1 / t2);*

*};*

**2. Sinf shablonini yaratish**

Argumentlarni almashtirish orqali shablondan aniq sinf yaratish jarayoni shablonni yaratish bosqichi deb ataladi. Sinf shabloni ma’lumotlar turi ham, sinf obyekti ham hisoblanmaydi. Faqat shablonlar tavsiflangan kirish faylidan hech qanday kod yaratilmaydi. Dasturni kompilyatsiya qilish uchun shalonga aniq ma’lumot uzatilishi shart.

***C++ algoritmik tilida MyNumber sinf shablonining ishlatilishiga misol:***

*MyNumber <int> mi; // sinfning mi obekti int turi bilan ishlaydi*

*MyNumber <float> mf; // sinfning mf obyekti float turi bilan ishlaydi*

*int d = 8;*

*float x = 9.3f;*

*// sonni 2 ga ko‘paytirish*

*mi.Mult2(&d); // d = 16*

*mf.Mult2(&x); // x = 18.6*

*// sonni kvadratga ko‘tarish*

*int dd;*

*dd = mi.MySquare(9); // dd = 81 – butun son*

*double z;*

*z = mf.MySquare(1.1); // z = 1.21000... – haqiqiy son*

*// sonni bo‘lish*

*long int t;*

*float f;*

*t = mi.DivNumbers(5, 2); // t = 2 – butun sonni bo‘lish*

*f = mf.DivNumbers(5, 2); // f = 2.5 – haqiqiy sonni bo‘lish*

**3. Argument qabul qiluvchi sinf shabloni**

Sinf shabloni ba’zi argumentlarni qo‘llaydigan holatlar ham uchrab turadi. Bu argumentlar sinf shablonida tavsiflanadigan funksiyalarda ishlatilishi mumkin.

Argument ishlatilgan sinf shaloniga misol:

*template <class T, type1 var1, type2 var2, ..., typeN varN>*

*class ClassName {*

*// sinf shabloni tanasi*

*// ...*

*};*

T – ma’lumotlarning umumiy turi;

*type1, type2, …, typeN* – sinf shablonida ishlatiladigan *var1, var1, var2, …, varN* nomli argumentlarning aniq turlari.

Bitta argument ishlatiladigan sinf shabloniga misol:

*ClassName <type, arg> ObjName;*

*ClassName* – shablonli sinf nomi;

*type* – real sinfni shakllantirishda ishlatiladigan ma’lumot turi;

*arg* – sinf shablonida ishlatiladigan argument qiymati;

*objName* – sinf shabloni obyektining nomi.

Bitta argument ishlatiladigan sinf shaloniga misol:

*template < int ArrayLength, typename SomeValueType >*

*class SomeClass {*

*SomeValueType SomeValue;*

*SomeValueType SomeArray[ArrayLength];*

*...*

*};*

*SomeClass < 20, int > SomeVariable; //sinfning birinchi obyekti SomeClass – turi int, 20 ta elementli*

*SomeClass < 30, double> SomeVariable2; //sinfning ikkinchi obyekti SomeClass - turi double, 30 ta elementli*

**4. Virtual funksiyalar**

*Virtual funksiyalar* mexanizmiga har bir hosila sinf uchun ma'lum bir komponent funksiyasining o‘z versiyasini talab qiladigan hollarda murojaat qilinadi. Bunday funksiyalarni o‘z [ichiga olgan sinflar](https://hozir.org/reja-mathcad-dasturi-imkoniyatlari-va-uning-interfeysi.html)*polimorfik*deb nomlanadi va OYDda alohida rol o‘ynaydi. Virtual funksiyalar kech (kechiktirilgan) yoki dinamik bog‘lanish mexanizmini ta'minlaydi. Asosiy sinfning har qanday statik bo‘lmagan [funksiyasini](https://hozir.org/mavzu-sintaksis.html)*virtual* kalit so‘z yordamida virtual qilish mumkin.

[Shunday qilib](https://hozir.org/1--limitlar-nazariyasi-namunaviy-variantning-yechilishi-1-tops.html), har bir virtual funksiya chaqiruvining tayanch sinfiga ko‘rsatgich orqali talqini ushbu ko‘rsatgichning [qiymatiga](https://hozir.org/15-mavzu-15--maruza.html), ya'ni chaqiruv qilingan obyekt turiga bog‘liq.

Qaysi virtual funksiyani chaqirishni tanlash ko‘rsatgich [turiga emas](https://hozir.org/2-mavzu-fundamental-ozaro-tasirlar-buyuk-birlashuv-nazariyasin.html), balki ko‘rsatgich aslida yo‘naltirilgan obyekt turiga (dasturni bajarish vaqtida) bog‘liq bo‘ladi.

Faqat statik bo‘lmagan a'zo funksiyalari virtual bo‘lishi mumkin. Virtuallik meros qilib olingan. Funksiya virtual deb aniqlangandan so‘ng, uni hosila sinfda (bir xil prototip bilan) qayta aniqlash ushbu sinfda yangi virtual [funksiyani yaratadi va](https://hozir.org/16-mavzu-elementar-funksiyalar-monotonligi-eng-katta-va-eng-ki.html)***virtual*** kalit so‘zi ishlatilmasligi mumkin.

Konstruktorlar destruktorlardan farqli ravishda virtual bo‘lishi mumkin emas. Amaliyotda virtual funksiyaga ega bo‘lgan har bir sinf virtual destruktorga ega bo‘lishi shart.

*class First{*

***public****:*

[*virtual void print*](https://hozir.org/virtual-reallikning-ijtimoiy-jihatlari.html)*{*

*cout << "First";*

*}*

*};*

*class Second : public First*

*{*

*void print()*[*override*](https://hozir.org/axborot-xavfsizligi-fakulteti-v7.html)

*{*

*cout << "Second";*

*}*

*};*

*int main() {*

*Second s;*

*First f;*

*First\* p1 = &s;*

*First\* p2 = &f;*

*p1->print();//Second metodi chaqiriladi (ta’sir qiladi)*

*p->print();//First metodi chaqiriladi (ta’sir qiladi)*

*}*

**5. Shablonlarning standart kutubxonasi(STL)**

***STL tarkibi.***  Biblioteka yadrosi uchta elementdan iborat: konteynerlar, algoritmlar va iteratorlar.

* ***Konteynerlar*** *(containers)* – bu boshqa elementlarni saqlovchi obyektlar. Masalan, vektor, chiziqli ro‘yxat, to‘plam.
* ***Assotsiativ konteynerlar*** *(associative containers)*-kalitlar yordamida ularda saqlanadigan qiymatlarni tezkor olish imkonini yaratadi.

Har bir sinf – konteynerida ular bilan ishlash uchun mo‘ljallangan funksiyalar to‘plami aniqlangan. Masalan, ruyxat elementlarni kiritish, chiqarish, va qo‘shish funksiyalarni o‘z ichiga oladi.

* ***Algoritmlar*** *(algorithms)-* konteyner ichidagilar ustidan operatsiyalar bajaradi. Konteyner ichidagilarni initsializatsiyalash, qidirish, saralash va almashtirish uchun algoritmlar mavjud. Ko‘p algoritmlar konteyner ichidagi elementlarni chiziqi ro‘yxatini ifodalaydovchi ketma-ketlik (sequence) bilan ishlash uchun mo‘ljallangan.
* ***Iteratorlar*** *(iterators)* – bu konteynerga nisbatan ko‘rsatkich sifatida bo‘lgan obyektlar. Ular massiv elementlariga ruxsat oluvchi ko‘rsatkichlar kabi, konteyner ichidagiga ruxsat olish imkoni beradi.

***Sinf-konteynerlar.*** STL da quyidagi sinf-konteynerlar aniqlangan:

Asosiy konteynerlar:

* vector *<vector.h>* dinamk massiv;
* list *<list.h*> chiziqli ro‘yxat;
* deque *<deque.h>* ikki tarafli (двусторонная) tartib
* set *<set.h>* to‘plam
* multiset <set.h> har bir elementi noyob bo‘lishi shart emas to‘plam
* map  *<map.h>* kalit/ qiymat juftlikni saqlash uchun assotsiativ ro‘yxat. Bunda har bir kalit bitta qiymat bilan bog‘langan.
* multimap *<map.h>* har bir kalit bilan ikkita yoki ko‘proq qiymatlar bog‘langan

Hosila konteynerlar:

* *stack <stack.h> stek*
* *queue <queue.h> tartib*
* *priority\_queue <queue.h>* birinchi o‘rindagi tartib

***Konstruktorlar.*** Ixtiyoriy sinf-konteyner ko‘rsatilmagan holda konstruktor va destruktorni nusxalovchi konstruktorga ega.

Masalan, *vektor* sinf-konteynerning konstruktori va destruktori:

| *vector<elem> c* | bitta ham elementga ega bo‘lmagan bo‘sh vektorni yaratadi; |
| --- | --- |
| *vector<elem> c1(c2)* | ko‘rsatilgan tipdagi boshqa vektorning nusxasini yaratadi (barcha elementlarni nusxasini oladi); |
| *vector<elem> c(n)* | konstruktor orqali ko‘rsatilmagan holda yaratilgan n elementli vektorni yaratadi; |
| *vector<elem> c(n,x)* | x elementning n nusxalari yordamida initsializatsiya etilgan vektorni yaratadi; |
| *~vector<elem>()* | barcha elementlarni o‘chiradi va xotirani bo‘shatadi. |

Ixtiyoriy obyekt uchun ko‘rsatilmagan holda konteynerda saqlanuvchi konstruktor mavjud bo‘lishi shart. Undan tashqari, obyekt uchun < va == operatorlar aniqlanish lozim.

***Iteratorlar.*** Itaratorlar bilan ko‘rsatkichlar kabi ishlash mumkin. Ularga \*, inkrement, dekrement operatorlarni qo‘llash mumkin. Iterator tipi sifatida har xil konteynerlarda aniqlangan iterator tip elon qilinadi.

Itoratorlarning beshta tipi mavjud:

**1**. *Kiritish iteratorlar* (input\_iterator) tenglik, nomini o‘zgartirish va inkrementa operatsiyalarni qo‘llaydi.

*==, !=, \*i, ++i, i++, \*i++*

Kiritish iteratsiyasining maxsus holati istream\_iterator iborat.

2. *Chiqarish iteratorlar* (output\_iterator) o‘zlashtirish operatorning chap tarafidan imkon bo‘lgan isimning o‘zgartirish va inkrementa operatsiyalar qo‘llanadi.

*++i, i++, \*i=t, \*i++=t*

Chiqarish iteratsiyasining maxsus holati *ostream\_iterator*.

3. *Bitta yo‘nalishdagi iteratorlar* *(forward\_iterator)* kiritish/chiqarish operatsiyalarning barchasini qo‘llaydi, bundan tashqari chegarasiz o‘zlashtirishning imkonini beradi.

*==, !=, =, \*i, ++i, i++, \*i++*

4. *Ikki yo‘nalishdagi iteratorlar* (biderectional\_iterator) forward-iteratorlarning barcha xususiyatlariga ega, bundan tashqari, konteynerni ikkita yo‘nalishi bo‘yicha o‘tish imkonini beradigan qo‘shimcha dekrementa (--i, i--, \*i--) operatsiyasiga ega.

5. *Ixtiyoriy ruxsatga ega bo‘lgan iteratorlar*(random\_access\_iterator) biderectional-iteratorlarning barcha xususiyatlariga ega, bundan tashqari solishtirish va manzil arifmetikasi operatsiyalarni qo‘llaydi.

*i+=n, i+n, i-=n, i-n, i1-i2, i[n], i1<i2, i1<=i2, i1>i2, i1>=i2*

Shuningdek, STLda teskari iteratorlar (reverse iterators) qo‘llaniladi. Ketma-ketlikni teskari yo‘nalishda o‘tuvchi ikki yo‘nalishli yoki ixtiyoriy ruxsatga ega bo‘lgan iteratorlar teskari iteratoralar bo‘lishi mumkin.

***Xotirani taqsimlovchilar, predikatlar va solishtirish funksiyalari.*** Konteynerlarga, algoritmlarga va STLdagi iteratorlarga qo‘shimcha bir nechta standart komponentalar xam qo‘llaniladi. Ulardan asoslari esa xotira taqsimlovchilar, predikatlar,va solishtirish funksiyalaridir.

Har bir konteynerda uning uchun aniqlangan va konteyner uchun xotirani belgilash jarayonini boshqaradigan xotira taqsimlovchisi (allocator) mavjud.

Ko‘rsatilmagan holda esa xotira taqsimlovchisi allocator sinf obyektidir. Xususiy taqsimlovchini tavsiflash mumkin.

Ba’zi bir algoritmlar va konteynerlarda muxim tipdagi predikat ataluvchi funksiyalar ishlatiladi. Predikatlar unar va binar bo‘lishi mumkin. U yoki bu qiymatni olish aniq shartlari dasturchi orqali aniqlanadi. Unar predikatlarning tipi – UnPred, binar predikatlarning esa - BinPred. Argumentlar tipi konteynerda saqlanuvchi obyektlar tipiga mos.

Ikkita elementlarni solishtirish uchun binar predikatlarning maxsus tipi aniqlangan. U solishtirish funksiya (comparison function) deyiladi. Agarda birinchi element ikinchidan kichik bo‘lsa, unda funksiya rost qiymatni qaytaradi. Comp tip funksiya tipidir. STL da obyekt-funksiyalar o‘ziga xos axamiyatga ega.

Obyekt-funksiyalar – bu sinfda «kichik qavslar» () operatsiyasi aniqlangan sinf nusxalari. Ba’zi bir xollarda funksiyalarni obyekt-funksiyalarga almashtirish qulay deb hisoblanadi. Obyekt-funksiya funksiya sifatida ishlatilsa, unda uni chaqirish uchun operator () operator ishlatiladi.

*Vector-vektor konteynerlari*

STL da vector vektor dinamik massiv sifatida aniqlanadi. Massiv elementlariga indeks orqali ruxsat beriladi.

*Vector* sinfida quyidagi konstruktorlar aniqlangan:

* Birinchi shakl bo‘sh vektor konstruktorini tavsiflaydi.
* Konstruktor vektorning ikkinchi shaklida elementlar soni – bu son, har bir elementi esa qiymat qiymatiga teng. Qiymat parametri ko‘rsatilmagan holdagi qiymat bo‘lishi mumkin.
* Konstruktor vektorning uchinchi shakli – bu nusxalash konstruktori.
* To‘rtinchi shakli – bosh va oxirgi iteratorlar orqali elementlar diapazonini o‘z ichiga olgan konstruktor vektor.

Vektorda saqlanadigan ixtiyoriy obyekt uchun ko‘rsatilmagan holda konstruktor aniqlash zarur. Bundan tashqari, obyekt uchun < va == operatorlar aniqlanishi lozim.

Vektor sinfi uchun quyidagi solishtirish operatorlari mavjud:

*==, <, <=, !=, >, >=.*

Bundan tashqari, vector sinf uchun [] indeks operatori aniqlangan.

*Ikki yo‘nalishli tartib (Deque)*

*Deque* – vektor kabi, ixtiyoriy ruxsat iteratorlarni qo‘llovchi ketma-ketlik ko‘rinishi. Bundan tashqari, u o‘zgarmas vaqtda boshida yoki oxirida kiritish va ochirish operatsiyalarni qo‘llaydi. O‘rtada kiritish va o‘chirish chiziqli vaqtni egallaydi. Xotirani boshqarishiga ishlov berish esa vektorlar kabi avtomatik ravishda bajariladi.

*Ruyxat(List).* Ro‘yxat – ikki yo‘nalishli iteratorlarni qo‘llaydigan xamda kiritish va o‘chirish operatsiyalarni o‘zgarmas vaqtda ketma-ketlikni ixtiyoriy joyida bajaradigan, shuningdek, xotirani boshqarishiga avtomatik ravishda ishlov beruvchi ketma-ketlik ko‘rinishi. Vektorlar va ikkitarafli tartiblardan farqi shundaki elementlar ro‘yxatiga tez va ixtiyoriy ro‘xsat qo‘llanmaydi, lekin ko‘pgina algoritmlarga esa ketma-ketlik ruxsat zarur.

***Assotsiativ konteynerlar (massivlar).***  Assotsiativ massiv juft qiymatlardan iborat. (key) kalit deb atalgan bitta qiymatni bilib (mapped value) aks etuvchi qiymat deb atalgan ikkinchi qiymatga ruxsat olishimiz mumkin.

Assotsiativ massivni massiv indekslari butun tiplardan iborat bo‘lmagan massiv sifatida tavsiflash mumkin:

V& operator[](const K&) K ga mos keluvchi V ga ilovani qaytaradi.

Assotsiativ konteynerlar – bu assotsiativ massivning umumiy tushunchasi.

***Map*** *assotsiativ konteyner* − bu kalit yordamida qiymatga tez ega bo‘lish imkonini yaratadigan juftlik (kalit, qiymat) ketma-ketligi. map konteyneri ikki yo‘nalishli iteratorni tavsif etadi.

map assotsiativ konteyneri kalit tiplari uchun “<” operatsiyasi mavjudligini talab qiladi. U kalit bo‘yicha saralangan o‘z elementlarini saqlaydi. Saralash almashuvi esa tartib bo‘yicha bajariladi.

map sinfida quyidagi knstruktorlar aniqlangan:

birinchi shakli bo‘sh assotsiativ konteynerning konstruktorini tavsiflaydi. Ikkinchi shakli – konstruktor nusxasi, uchinchisi – elementlar diapazonini kamrab olgan assotsiativ konteynerning konstruktori.

O‘zgartirish operatsiyasi aniqlangan:

* *map& operator=(const map&);*
* quyidagi operatsiyalar aniqlangan: *==, <, <=, !=, >, >=.*
* mapda kalit/qiymat juftliklar pair tiplagi obyektlar ko‘rinishida saqlanadi.

Kalit/qiymat juftliklarni faqatgina pair sinf konstruktorlari yordamida, balki pair tipdagi obyektlarni yaratuvchi va ma’lumotlar tiplaridan parametrlar sifatida foydalanuvchi make\_pair funksiya yordamida yaratish mumkin.

Assotsiativ konteyner uchun o‘ziga xos operatsiya – bu ([]) indeksatsiyalash operatsiyasi yordamida assotsiativ qidiruv.

* *mapped\_type& operator[](const key\_type& K);*
* set to‘plamini assotsiativ massiv sifatida ko‘rish mumkin. Unda qiymatlar axamiyatga ega emas, shuning uchun faqat kalitlarni ko‘zatish mumkin.
* to‘plamlar assotsiativ massiv kabi T tip uchun (<) “kichik” operatsiyani mavjudligini talab etadi. U o‘z elementlarini saralangan holda saqlaydi. saralash almashuvi esa tartibda bajariladi.

***Konteyner usullari.*** Iteratorlarni olish usullari:

* *begin()* - birinchi elementga ko‘rsatadi;
* *end()* - oxiridan keyingi elementga ko‘rsatadi;
* *rbegin()* - teskari ketma-ketlikdagi birinchi elementni ko‘rsatadi;
* rend() teskari ketma-ketlikdagi oxirgidan keyingi elementni ko‘rsatadi

***Elementlarga ruxsat:***

* *front()* - birinchi elementga ilova;
* *back()* - oxiri elementga ilova;
* *operator[](i)* - tekshirishsiz indeks bo‘yicha ruxsat;
* *at(i)* - tekshirish bilan indeks bo‘yicha ruxsat.
* *front()* - birinchi elementga ilova;

***Elementlarni kiritish usullari:***

* *insert(p,x)* - r ko‘rsatgan elementdan oldin x ni qo‘shish
* *insert(p,n,x)* – r dan oldin xning n nusxalarini qo‘shish
* *insert(p,first,last)* – r dan oldin [first:last]dagi elementlarni qo‘shish
* *push\_back(x)* - oxiriga xlarni qo‘shish
* *push\_front(x)*- yangi birinchi elementni qo‘shish (ikta uchga ega bo‘lgan tartiblar va ro‘yxatlar uchun)

***Elementlarni o‘chirish usullari:***

* *erase(p)* - r pozitsiyadagi elementni o‘chirish;
* *erase(first,last)* - [first:last]dan elementlarni o‘chirish;
* *pop\_back()* - oxirgi elementni o‘chirish;
* *pop\_front()* - birinchi elementni o‘chirish (ikta uchga ega bo‘lgan tartiblar va ro‘yxatlar uchun).

***O‘zlashtirish usullari:***

* *operator=(x)* konteynerga x konteynerni elementlari o‘zlashtiriladi;
* *assign(n,x)* - konteynerga x elementning n nusxasi o‘zlashtiriladi (assotsiativ bo‘lmagan konteynerlar uchun);
* *assign(first,last)* - [first:last] diapazondagi elementlarni o‘zlashtirish

Assotsiativ usullari:

* *find(elem)* - elem qiymatga ega bo‘lgan birinchi elementni pzitsiyasi topadi
* *lower\_bound(elem)* - element qo‘yish mumkin bo‘lgan birinchi pozitsiyani to‘padi
* *upper\_bound(elem)* - element qo‘yish mumkin bo‘lgan oxirgi pozitsiyani to‘padi
* *equal\_range(elem)* - element qo‘yish mumkin bo‘lgan birinchi va oxirgi pozitsiyalarni to‘padi.

***Boshqa usullar:***

* *size()* - elementlar soni;
* *empty()* - konteyner bo‘shmi?
* *capacity()* - vektor uchun ajratilgan xotira (faqat vektorlar uchun);
* *reserve(n)* - n elementdan iborat bo‘lgan konteyner uchun xotira ajratadi;

**Nazorat savollari**

1. Yarimstatik ma’lumotlar tuzilmasi nima va unga nimalar kiradi?

2. Stek va uning xususiyatlari?

3. Steklarni dasturda e’lon qilinishi?

4. Navbat nima va dasturda qanday ifodalanadi?

5. Dek nima va stek , navbatdaqn farqi nima? Dasturda ifodalanishi qanday?

6. Bu tuzilmalar statik va dinamik tuzilmalardan nimasi bilan farq qiladi?