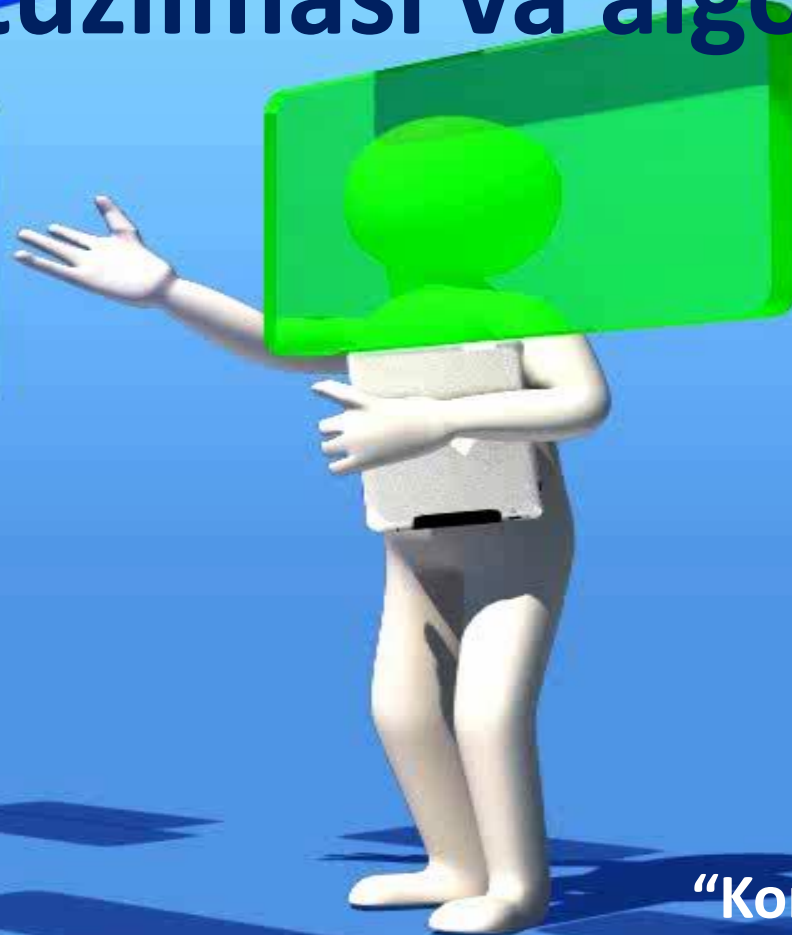




TOSHKENT AMALIY FANLAR UNIVERSITETI

# Ma'lumotlar tuzilmasi va algoritmlar fani



“Kompyuter injiniring” kafedrası

Katta o'qituvchi Kendjayeva Dildora Xudayberganovna

## 13 - Ma'ruza

Xesh jadvallar. Xesh jadvallar va ularni tashkil etish, C++ dasturlash tilida xesh jadvallarni realizatsiya qilish





## ***Asosiy adabiyotlar:***

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.
2. O. R. Yusupov, I. Q. Ximmatov, E. Sh. Eshonqulov. Algoritmlar va berilganlar strukturalari. Oliy o'quv yurtlari uchun o'quv qo'llanma. – Samarqand: SamDU nashri. 2021-yil, 204 bet.
3. Xayitmatov O'.T., Inogomjonov E.E., Sharipov B.A., Ruzmetova N., Ma'lumotlar tuzilmasi va algoritmlari fanidan o'quv qo'llanma
4. Rahimboboeva D. "Ma'lumotlar tuzilmasi va algoritmlari" fanidan o'quv qo'llanma – T.: TDIU, 2011.-135 bet.

# MA'RUZA REJASI



*Xesh jadvallar va ularni tashkil etish*



*Polinomial xeshlash.*



*Xesh funksiyasi xususiyatlari*



*C++ dasturlash tilida xesh jadvallarni realizatsiya qilish*

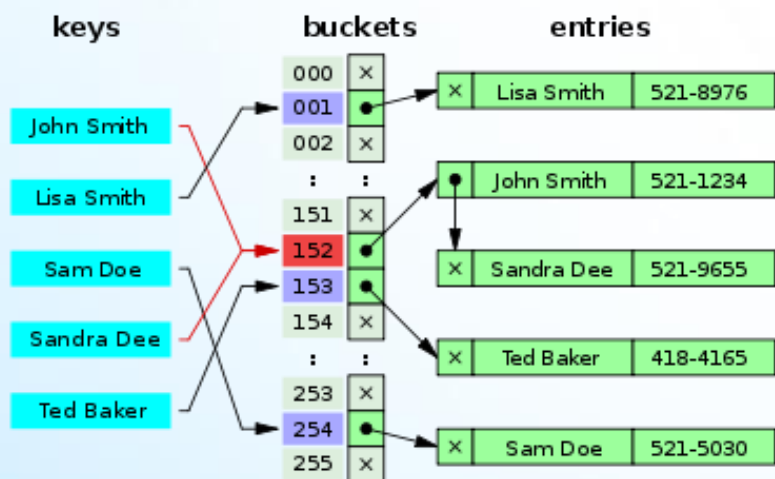


*Map bilan bog'liq ba'zi asosiy funksiyalar*

# Xesh jadvallar va ularni tashkil etish

**Xesh jadvali** - bu assotsiativ massiv interfeysini amalga oshiruvchi ma'lumotlar tuzilmasi, ya'ni juftlarni saqlashga (*kalit, qiymat*) va uchta amalni bajarishga imkon beradi: yangi juftlikni qo'shish, qidirish amali va juftlikni kalit bilan o'chirish.

Xesh jadvallarining ikkita asosiy varianti mavjud: **zanjirli** va **ochiq adreslash**. Xesh jadvali ba'zi bir massivini o'z ichiga oladi, ularning elementlari juftliklar (*ochiq adreslash bilan xesh jadvali*) yoki juftliklar ro'yxati (*zanjir bilan xesh jadvali*) bo'ladi.

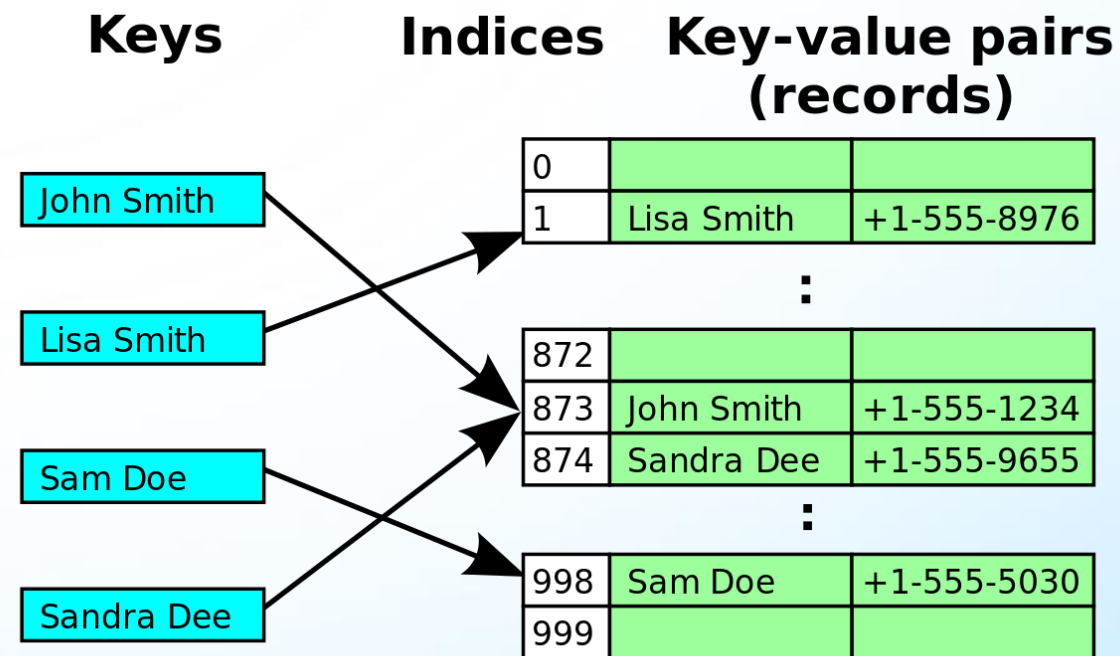




# Xeshlash

**Xeshlash** - bu ixtiyoriy uzunlikdagi kirish ma'lumotlari majmuasini ma'lum bir algoritm tomonidan bajarilgan, belgilangan o'lchamdagi chiqish massiviga aylantirish jarayoni.

Bunday algoritmni amalga oshiruvchi funksiya xesh funksiya, transformatsiya natijasi **xesh** yoki **xesh yig'indisi** deyiladi.



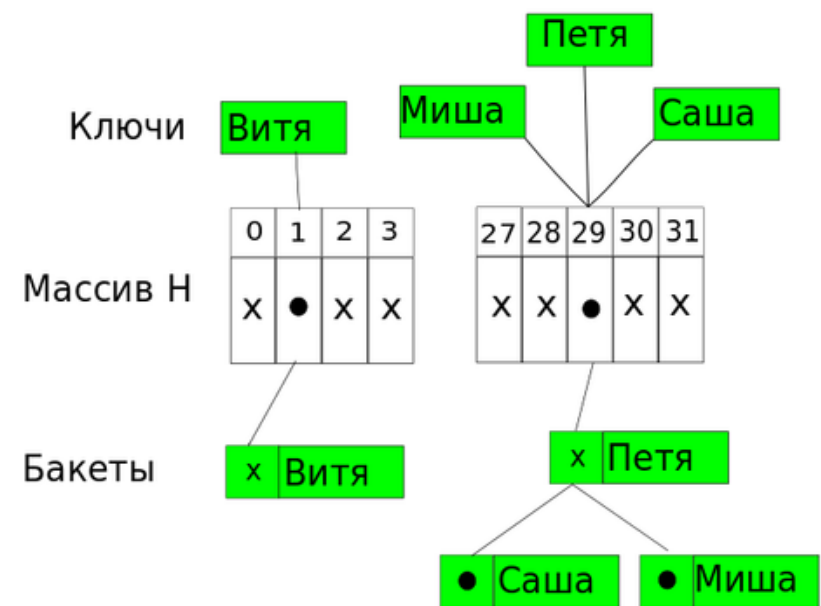
# Xesh funksiyasi xususiyatlari

*Xesh funksiyasi* quyidagi *xususiyatlarga* ega:

- bir xil ma'lumotlar bir xil xeshni beradi;
- "deyarli har doim" turli xil ma'lumotlar boshqacha xesh beradi.

Ikkinchi xususiyatdagi "deyarli har doim" izohi xeshlarning aniq o'lchamiga ega bo'lishidan kelib chiqadi, shu bilan birga kirish ma'lumotlari bu bilan cheklanmaydi.

Natijada, biz xesh funktsiyasi kirish ma'lumotlari to'plamidan xeshlar to'plamiga xaritalashni amalga oshiramiz, bu esa ularning kardinalligi ancha past bo'ladi.

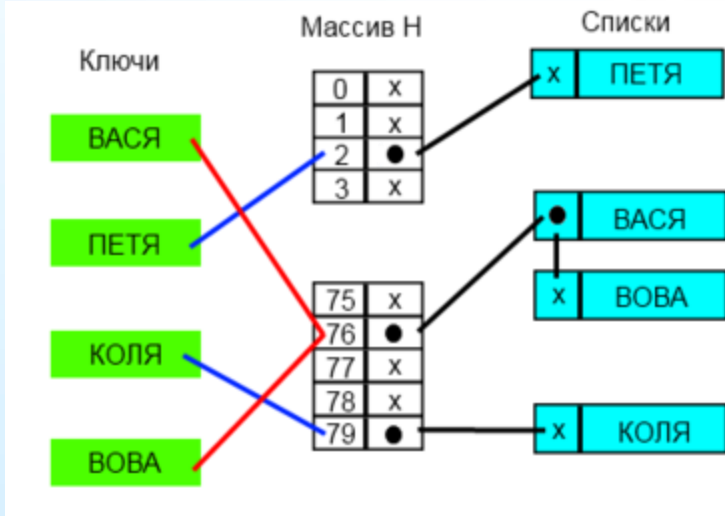


Dirixle prinsipiga ko'ra, har bir xesh uchun bir nechta turli xil ma'lumotlar to'plamlari bo'ladi. Bunday moslik ***kolliziya*** deb ataladi.

Agar biron bir muammoni hal qilishda kirish ma'lumotlari cheklangan bo'lsa, siz bunday xeshlar to'plamini tanlashingiz mumkin, shunda uning aniqligi kirish ma'lumotlari to'plamining muhimligidan oshib ketadi.

Bunday holda, biz ***inyeksion xaritalashni*** aniqlaydigan xesh funksiyasini qurishimiz mumkin (*mukammal xeshlash*).

Biroq, umuman olganda, ***kolliziya muqarrar***. Kolliziya ehtimoli xesh funksiyasi sifatini baholash uchun ishlatiladi.

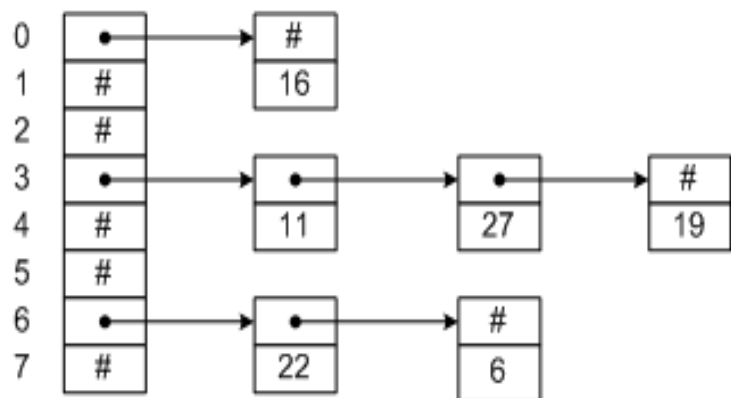




## ***Yaxshi xesh funksiyasi quyidagicha ishlaydi:***

- mavjud bo'lgan barcha xesh oralig'i maksimal darajada ishlatiladi;
- kirish ma'lumotlarining ozgina o'zgarishi ham mutlaqo boshqacha xeshni berishi kerak, to'qnashuvlar faqat butunlay boshqacha ma'lumotlar uchun ro'y berishi kerak.

hashTable



Xeshlash o'zi obyektga tasodifiy o'zgaruvchini xaritalashga o'xshaydi. Birinchi xususiyat natijasida xeshlar o'zlarini bir tekis taqsimlangan tasodifiy o'zgaruvchilar kabi tutishi kerak, bu butun diapazondan foydalanishni ta'minlaydi, bu foydali bo'lishi mumkin, masalan, xesh jadvalini tuzishda.

# Polynomial xeshlash

Quyida oddiy, ammo samarali xeshlash algoritmini ko'rib chiqamiz. Xesh funksiyamizni quyidagicha aniqlaylik:

$$h(s) = \sum_{i=0}^N b^{N-i} \cdot \text{code}(s_i) \quad (1)$$

yoki

$$h(\text{pref}_i) = b \cdot h(\text{pref}_{i-1}) + \text{code}(s_{i-1}), \quad (2)$$

bu yerda  $N = \text{str } l e n (s) - 1$ ,  $p r e f_i$  — uzunlik prefiksi  $i$ ,  $b$ -baza, asos.  $c o d e (s_i)$  — simvol kodi.

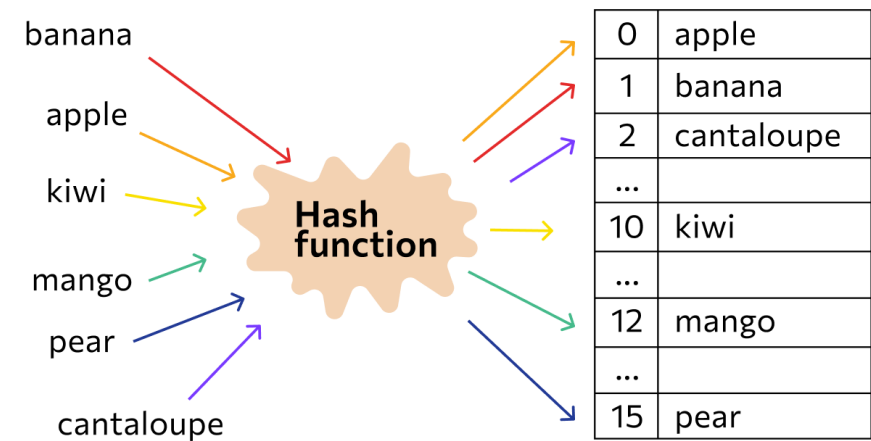
Agar (1) formulani kengaytirsak, biz  $N$  tartibli polinomni olamiz. (2) formula xeshni rekursiv shaklda o'rnatadi va kod yozishda foydalaniladi.

Belgilar kodiga va asosga e'tibor qaratish lozim, chunki bazani ***tanlash kodlarga bog'liq*** bo'ladi. ***Kod ASCII*** jadvalidagi belgilar kodi yoki alfavitdagi tartib raqam bo'lishi mumkin.

Masalan, agar muammo har qanday satr ingliz alifbosining faqat kichik harflaridan iborat bo'lishiga kafolat beradigan bo'lsa, unda tartib raqami belgilar kodlari uchun yaxshi imkoniyatdir.

Shuni ta'kidlash joizki, biz xeshni hech qanday cheklamaymiz, bu xeshlash ta'rifiga ziddir. Bunday holda, ikkita chiqish usuli mavjud: ***modul bo'yicha bo'lish*** amalidan yoki ***uzun arifmetikadan*** foydalanish.

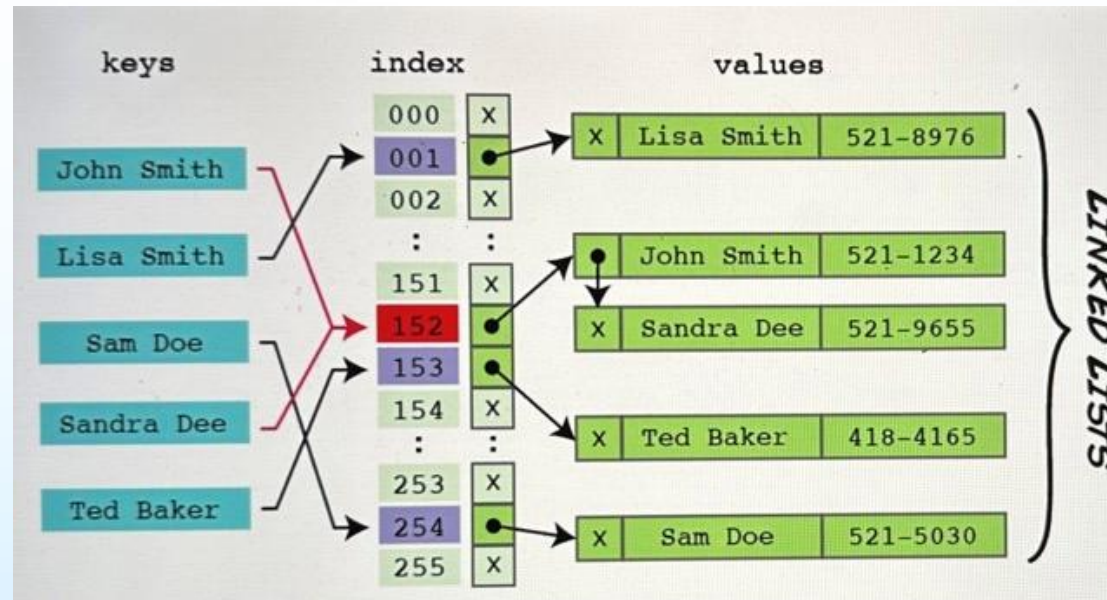
### Hash Tables






**Birinchi usul** uzun arifmetikaga ega bo'lmagan tillarda keng qo'llaniladi. Bundan tashqari, xesh saqlanadigan butun sonli ma'lumotlar turi bu bo'linishni avtomatik ravishda amalga oshiradi (*turlarning ko'payishi natijasida qo'shimcha bitlar avtomatik ravishda yo'qoladi*).

Natijada biz cheklangan xeshlar to'plamini olamiz, ammo yana **kolliziya** xavfi mavjud. Bundan tashqari, ko'p polinomli xeshni "buzish" ehtimoli mavjud.





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

```
long long Heshlash(char s[])
{
    long long h = 0;
    int base = 37;
    for(int i=0; i<=strlen(s); i++)
    {
        h = h* base + s[i] - 61 +1;
    }
    return h;
}
```

```
int main()
{
    char s[100];

    for(int i=1; i<10; i++)
    {
        cin.getline(s,100);
        cout<<s<<" "<<Heshlash(s);
        cout<<endl;
    }
}
```

Ikkinchi variantda kolliziya ehtimoli pastroq. Biroq, kattaroq xeshlar to'plamini qo'llab-quvvatlash, qo'shimcha xotira va ikkita xeshni taqqoslash uchun zarur bo'lgan vaqtni talab qiladi, bu oddiy ma'lumotlarni taqqoslashdan ko'ra tezroq.

Masalan, satrni inglizcha kichik harflardan iborat deb taxmin qilamiz. Quyida 37 raqamini asos qilib olamiz.



# Xesh jadvallardan foydalanish samaradorligi

Barcha ma'lumotlar yaxshi bajarilgan konteynerlarni, yaxshi tanlangan xesh funksiyalarini taqdim etdi.

Ushbu jadvaldan nega xesh jadvallardan foydalanish kerakligi juda aniq ko'rinib turibdi. Ammo keyin qarama-qarshi savol tug'iladi: nega ular doimo ishlatilmaydi?

Javob juda sodda: har doimgidek, birdaniga hamma narsani olish mumkin emas, ya'ni: ham tezlikdan, ham xotiradan yutib bo'lmaydi. Xesh jadvallari noqulay va ular operatsion jarayonning asosiy savollariga tezda javob berishlari bilan birga, ulardan foydalanish har doim juda qimmatga tushadi.


Konteyner / Amal	Insert (qo'shish)	Remove (O'chirish)	Izlash (find)
Massiv	$O(N)$	$O(N)$	$O(N)$
Ro'yxat	$O(1)$	$O(1)$	$O(N)$
Saralangan massiv	$O(N)$	$O(N)$	$O(\log N)$
Ikkilik qidiruv daraxti	$O(\log N)$	$O(\log N)$	$O(\log N)$
Xesh-jadval	$O(1)$	$O(1)$	$O(1)$

# *C++ dasturlash tilida xesh jadvallarni realizatsiya qilish*

C++ dasturlash tilida xesh jadvallarni hosil qilish uchun map konteyneri aniqlangan. map konteyner vector, list, deque kabi boshqa konteynerlarga juda o'xshaydi, lekin ozgina farqi mavjud. Bu konteynerga birdaniga ikkita qiymat qo'yish mumkin. Shunday qilib, bu map misolni batafsil ko'rib chiqaylik:

```
#include <iostream>
#include <map>    //map bilan ishlash uchun kutubxonani ulash
using namespace std;
int main()
{
    ///map oshkor initsializatsiyalash
    map <string,int> myFirstMap = {{ "Mother", 37 },
                                   { "Father", 40 },
                                   { "Brother", 15 },
                                   { "Sister", 20 }};

    /// initsializatsiyalangan mapni ekranga chiqarish
    for (auto it = myFirstMap.begin(); it != myFirstMap.end(); ++it)
    {
        cout << it->first << " : " << it->second << endl;
    }
}
```



```
char c;
map <char,int> mySecondMap;
for (int i = 0,c = 'a'; i < 5; ++i,++c)
{
    mySecondMap.insert ( pair<char,int>(c,i) );
}

/// initsializatsiyalangan mapni ekranga chiqarish
for (auto it = mySecondMap.begin(); it != mySecondMap.end(); ++it)
{
    cout << (*it).first << " : " << (*it).second << endl;
}
return 0;
}
```

# *Map bilan bog'liq ba'zi asosiy funksiyalar*

Map bilan bog'liq ba'zi asosiy funksiyalar quyida keltirilgan:

- **begin()** - iteratorni mapdagi birinchi elementga qaytaradi
- **end()** - iteratorni mapdagi oxirgi elementdan keyingi nazariy elementga qaytaradi
- **size()** - mapdagi elementlar sonini qaytaradi
- **max\_size()** - mapda saqlanishi mumkin bo'lgan elementlarning maksimal sonini qaytaradi
- **empty()** - mapning bo'shligini tekshiradi
- **pair\_insert(*keyvalue*, *mapvalue*)** - mapga yangi element qo'shiladi
- **erase(*iterator position*)** - elementni iterator ko'rsatgan joydan olib tashlaydi
- **erase(*const g*)** - mapdan "g" kalit qiymatini olib tashlaydi
- **clear()** - mapdagi barcha elementlarni olib tashlaydi



## ***Kolliziya muammosi.***

Tabiiyki, savol tugʻiladi, nega biz bir qator katakchaga ikki marta kirib olishimiz mumkin emas, chunki har bir elementga mutlaqo boshqacha natural sonlarni taqqoslaydigan funktsiyani taqdim etish shunchaki mumkin emas. Kolliziya muammosi xesh funktsiyasi turli elementlar uchun bir xil natural sonni hosil qilganda paydo boʻladigan muammo.

Ushbu muammoning bir nechta yechimlari mavjud: zanjirlash usuli va ikki marta xeshlash usuli.





## ***Mavzu yuzasidan savollar:***

1. Xesh jadval nima?
2. Xesh jadvallardan foydalanish samaradorligini taqqoslang
3. Xesh funksiyasiga misol keltiring
4. Satrlar uchun xesh funksiyasini qo'llang
5. Xesh funksiya ma'lumotlar strukturasi qo'llaniladigan sohalarga qaysilar kiradi?

***Do you have  
any questions?***

