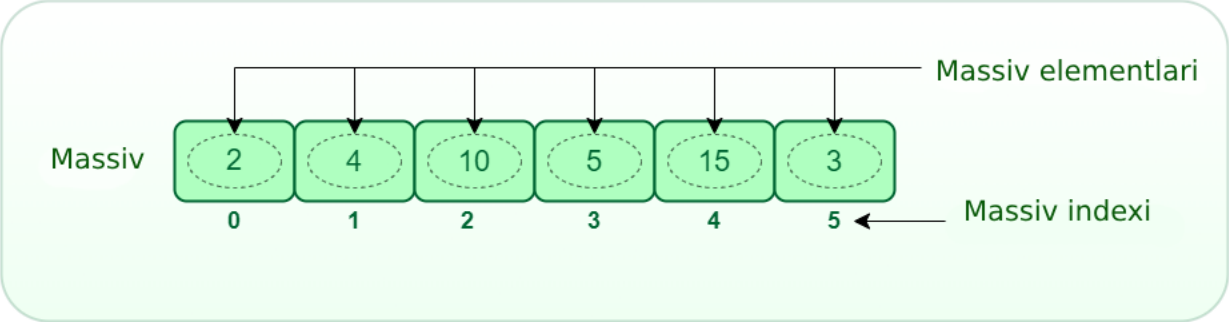
Massiv chiziqli ma’lumotlar tuzilmasi

Massiv-bu qo‘shni xotirada saqlanadigan bir xil turdagi o‘zgaruvchilardan tashkil topgan elementlari to‘plami. Bu eng mashhur va sodda ma’lumotlar tuzilmalaridan biri bo‘lib, ko‘pincha boshqa ma’lumotlar tuzilmalarini amalga oshirish uchun ishlatiladi.

Massivning asosiy shartlari

- Massiv indeksi
- Massiv elementi
- Massiv uzunligi



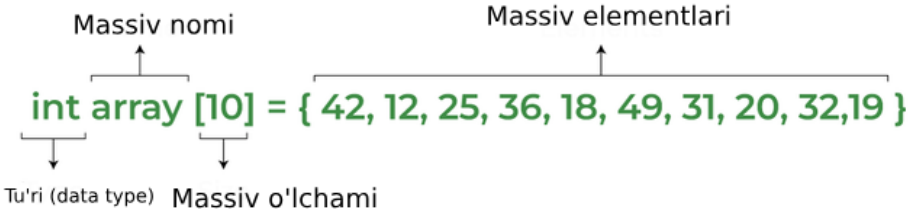
Rasm- 1 Massivning asosiy shartlari

Massivlar turli dasturlash tillarida turli yo‘llar bilan e’lon qilinishi mumkin. Yaxshiroq tasvirlash uchun quyida ikkita dasturlash tiliga xos deklaratsiyalari keltirilgan.

```
    //C++
    int arr[5];     // 5 ta elementga ega bo‘lgan butun sonlar massivi
    char arr[10]; // 10 ta elementga ega bo‘lgan simbollar massivi
    float arr[20]; // 20 ta elementga ega bo‘lgan haqiqiy sonlar massivi
```

```
    # Python
```

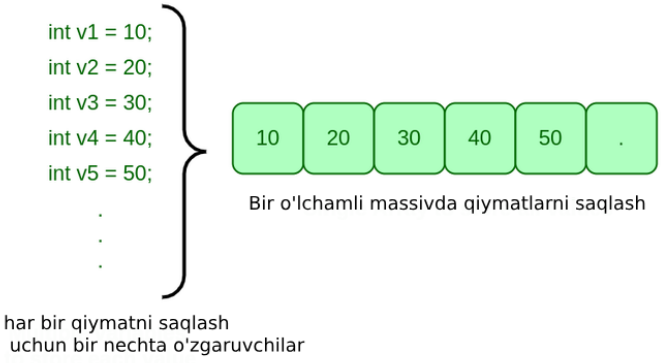
```
arr = [10, 20, 30] # Butun sonlar massivi
arr2 = ['c', 'd', 'e’] # simbollar massivi
arr3 = [28.5, 36.5, 40.2] # Haqiqy sonlar massiv
```



Rasm- 2 Massivning C++ dasturlash tilida e’lon qilinishi

Nima uchun massiv ma’lumotlar tuzilmalari kerak? faraz qilaylik, beshta o‘quvchidan iborat sinf bor va agar biz ularning imtihonlarda baholarini qayd etishimiz kerak bo‘lsa, biz buni beshta o‘zgaruvchini individual deb e’lon qilish va yozuvlarni kuzatish orqali amalga oshirishimiz mumkin, ammo agar talabalar soni juda katta bo‘lsa, ma’lumotlarni boshqarish va saqlash qiyin bo‘ladi.

Bu shuni anglatadiki, biz oddiy o‘zgaruvchilardan foydalanishimiz mumkin (v1, v2, v3, ..), bizda oz sonli ob’ektlar mavjud bo‘lganda. Ammo agar biz ko‘p sonli misollarni saqlamoqchi bo‘lsak, ularni oddiy o‘zgaruvchilar bilan boshqarish qiyin bo‘ladi. Massivning g’oyasi bitta o‘zgaruvchida bir nechta misollarni ifodalashdir.

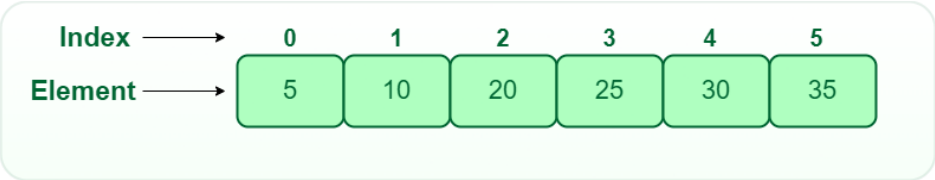


Rasm- 3 Bir o’lchamli massivda qiymatlarni saqlash

1.1 Massiv turlari.

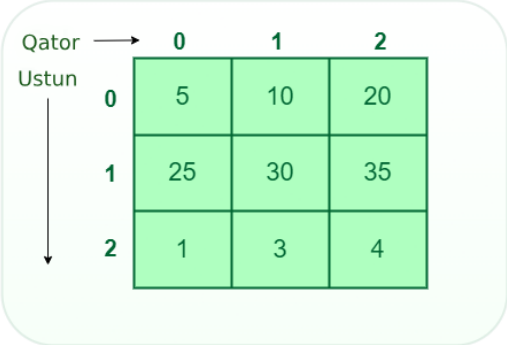
Massivlarning asosan ikki turi mavjud:bir o‘lchovli (1-D array) va ko‘p (n-D arrays) o‘lchovli.

Massiv bir o‘lchovli (1-D array) deyiladi, agar uning elementiga bir indeks orqali murojaat qilish mumkin bo‘lsa.



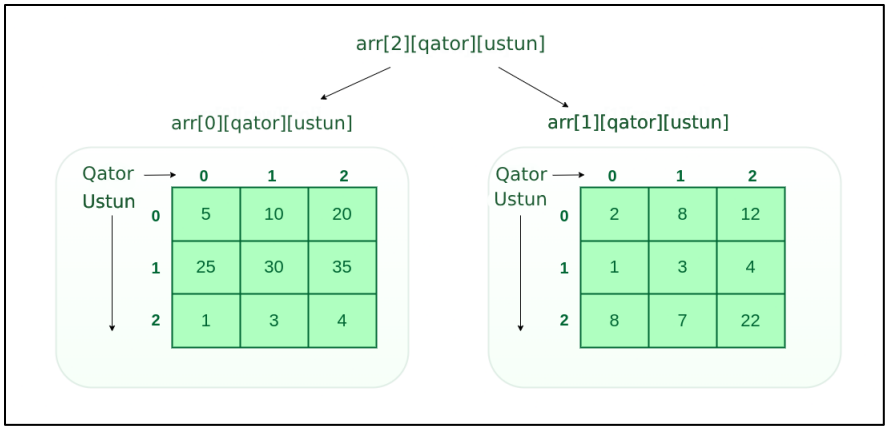
Rasm- 4 Bir o‘lchovli massiv

Ikki o‘lchovli massivlarni (2-D arrays) qatorlar va ustunlardan iborat matritsa sifatida ko‘rib chiqish mumkin.



Rasm- 5 Ikki o‘lchovli massiv

Uch o‘lchovli (3-D arrays)ko‘p o‘lchovli massiv uchta o‘lchovni o‘z ichiga oladi, shuning uchun uni ikki o‘lchovli massiv sifatida ko‘rish mumkin.



Rasm- 6 Uch o‘lchovli massiv

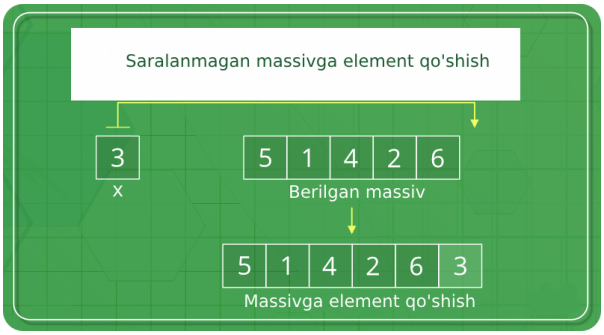
1.2 Massivlarda bajariladigan amallar:

- Massivda elementlarni qidirish(Searching).
- Massivda elementlarni saralash(Sorting).
- Massivga elementlarni qo‘shish (Insertion).
- Massivdan elementlarni o‘chirish(Deletion).

Massivga elementlarni qidirish va saralashni II bobta to‘liq ma’lumot berip o‘tamiz.

Massivga elementlarni qo‘shish(Insertion):

1.Massivning oxiriga elementlarni kiritish: saralanmagan massivda element kiritish operatsiyasi saralangan massivga nisbatan tezroq bo‘ladi, chunki biz element joylashtiriladigan pozitsiya haqida qayg’urmaymiz.



Rasm- 7 Saralanmagan massivga element qo‘shish

```
#include<iostream>
```

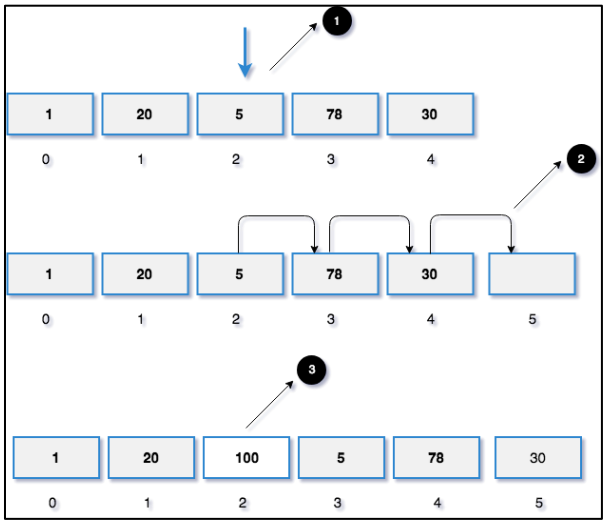
```
    // Kiritish chiqarish kutubxonasi

    using namespace std;
    //Standart nomlar fazosi

    // Funktsiya
    int insertSorted(int arr[], int n, int key, int capacity) {

    // Massiv elementlari to‘liq ekanligini tekshirish
    if (n >= capacity)
         return n;

    // massiv oxiriga kalit qo‘shish
    arr[n] = key;

    // O‘zgargan massivni bosh funktsiyaga qaytarish
         return (n + 1);
    }
    int main() {
    int arr[20] = {12, 16, 20, 40, 50, 70};

    int capacity = sizeof(arr) / sizeof(arr[0]);

    int n = 6;
    int i, key = 26;

    cout << "\nBerilgan massiv: ";
    for (i = 0; i < n; i++)
         cout << arr[i] << " "; // Inserting the key into the
  array

    n = insertSorted(arr, n, key, capacity);

    cout << "\nBerilgan massivga element qo‘shish: ";
    for (i = 0; i < n; i++)
         cout << arr[i] << " ";

    return 0;

    }}
    Dastur natijasi:
```

```
/tmp/E55Yg5xIN1.o
Berilgan massiv: 12 16 20 40 50 70
Berilgan massivga element qo'shish: 12 16 20 40 50 70 26
```

Vaqt murakkabligi: O (n)

Xotira murakkabligi: O (1)

2. Massivning istalgan joyga elementlarni kiritish: Agar biz 100 ni 2-indexka kiritishimiz kerak bo‘lsa, ijro massivning oxiridan bo‘cha katak ajratib har bir elementi o‘nga silzitishimiz kerak bo‘ladi.



Rasm- 8Massivning istalgan joyga elementlarni kiritish

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

// element qo‘shish funktsiyasi
// aniq bir pozitsiyaga
void insertElement(int arr[], int n, int x, int pos)
{
      // elementlarni o‘nga silzitish
      // silzitishtan so‘ng element qo‘shish
      for (int i = n - 1; i >= pos; i--)
            arr[i + 1] = arr[i];

      arr[pos] = x;
}
```

```
// Bosh funktsiya
int main()
{
	int arr[15] = { 2, 4, 1, 8, 5 };
	int n = 5;

	cout << "Berilgan massiv : ";
	for (int i = 0; i < n; i++)
		cout << arr[i] << " ";

	cout << endl;

	int x = 10, pos = 2;

	// Funktsiyani chaqirish
	insertElement(arr, n, x, pos);
	n++;

	cout << "Element qo‘shilgandan so‘ng massiv : ";
	for (int i = 0; i < n; i++)
		cout << arr[i] << " ";

	return 0;
}
```

Dastur natijasi:

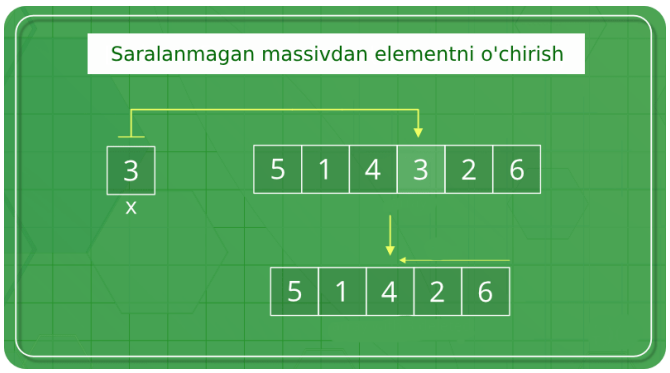
```
/tmp/6ySdlOixEk.o
Berilgan massiv : 2 4 1 8 5
Element qo'shilgandan so'ng massiv : 2 4 10 1 8 5
```

Vaqt murakkabligi: O(n)

Xotira murakkabligi: O(1)

Massivda elementlarni o‘chirish(Deletion).

1.Saralanmagan massivda elementlarni o‘chirish. O‘chirish operatsiyasida o‘chiriladigan element chiziqli qidiruv yordamida qidiriladi va keyin o‘chirish operatsiyasi amalga oshiriladi, so‘ngra elementlarni siljitish bajariladi.



Rasm- 9 Saralanmagan massivda elementni o‘chirish

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

// o‘chiriladigan elementni qidirish
int findElement(int arr[], int n, int key);

// Elementni massivdan o‘chirish
int deleteElement(int arr[], int n, int key)
{
     // ochirilgadigan elementning pozitsiyasini topish
     int pos = findElement(arr, n, key);

     if (pos == -1) {
          cout << "Element topilmadi";
          return n;
     }

     // Elementni o‘chirish
     int i;
     for (i = pos; i < n - 1; i++)
          arr[i] = arr[i + 1];

     return n - 1;
}

// Qidirish funktsiyasi
int findElement(int arr[], int n, int key)
{
     int i;
     for (i = 0; i < n; i++)
```

```
            if (arr[i] == key)
                return i;

    return -1;
}

// Bosh funktsiya
int main()
{
    int i;
    int arr[] = { 10, 50, 30, 40, 20 };

    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 30;

    cout << "Berilgan massiv \n";
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";

    // Function call
    n = deleteElement(arr, n, key);

    cout << "\nMassiv element o‘chirilgandan kiyin \n";
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

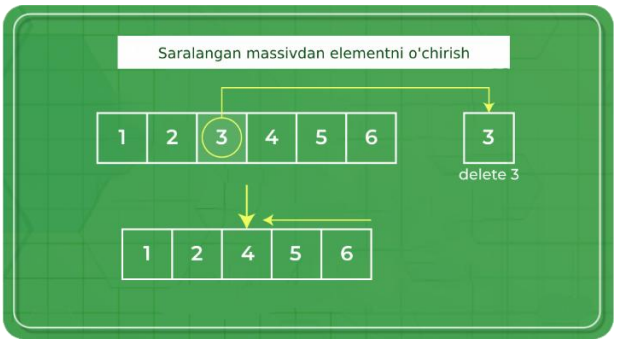
Dastur natijasi:

```
/tmp/6ySdlOixEk.o
Berilgan massiv
10 50 30 40 20
Massiv element o'chirilgandan kiyin
10 50 40 20
```

Vaqt murakkabligi: O(n)

Xotira murakkabligi: O(1)

Saralangan massivda elementlarni o‘chirish. O‘chirish operatsiyasida o‘chiriladigan element ikkilik qidiruv (binary search) yordamida qidiriladi va keyin o‘chirish operatsiyasi amalga oshiriladi, so‘ngra elementlarni siljitish bajariladi.



Rasm- 10 Saralangan massivdan elementni o‘chirish

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

// o‘chiriladigan elementni qidirish funktsiyasi
int findElement(int arr[], int n, int key);

// Elementni o‘chirish funktsiyasi
int deleteElement(int arr[], int n, int key)
{
     // element pozitsiyasini topish
     int pos = findElement(arr, n, key);

     if (pos == -1) {
          cout << "Element topilmadi";
          return n;
     }

     // Elementni o‘chirish
     int i;
     for (i = pos; i < n - 1; i++)
          arr[i] = arr[i + 1];

     return n - 1;
}
```

```
// o‘chiriladigan elementni qidirish funktsiyasi
int findElement(int arr[], int n, int key)
{
     int i;
     for (i = 0; i < n; i++)
          if (arr[i] == key)
               return i;

     return -1;
}

// Bosh funktsiya
int main()
{
     int i;
     int arr[] = { 10, 50, 30, 40, 20 };

     int n = sizeof(arr) / sizeof(arr[0]);
     int key = 30;

     cout << "Berilgan massiv\n";
     for (i = 0; i < n; i++)
          cout << arr[i] << " ";

     // funktsiyani chaqirish
     n = deleteElement(arr, n, key);

     cout << "\nMassiv elementni o‘chirgandan kiyin\n";
     for (i = 0; i < n; i++)
          cout << arr[i] << " ";

     return 0;
}
```

Dastur natijasi:

```
/tmp/6ySdlOixEk.o
Berilgan massiv
10 50 30 40 20
Massiv elementni o'chirgandan kiyin
10 50 40 20
```

1.3 Massivlardan foydalanishning afzalliklari:

- Massivlar elementlarga tasodifiy kirishga imkon beradi. Bu elementlarga pozitsiya bo‘yicha kirishni tezlashtiradi.
- Massivlar keshining xotirada yaxshi joylashuviga ega, bu esa ishlashda juda katta farq qiladi.
- Massivlar bitta nom yordamida bir xil turdagi bir nechta ma’lumotlar elementlarini ifodalaydi.
- Massivlar bir xil nomdagi o‘xshash turdagi bir nechta ma’lumotlarni saqlaydi.
- Massiv ma’lumotlar tuzilmalari bog’langan ro‘yxatlar, stacks, navbatlar, daraxtlar, Graflar va boshqalar kabi boshqa ma’lumotlar tuzilmalarini ijro qilish uchun ishlatiladi.

Massivning kamchiliklari:

- Massivlar belgilangan hajmga ega bo‘lgani uchun, ularga xotira ajratilgandan so‘ng, uni ko‘paytirish yoki kamaytirish mumkin emas, agar kerak bo‘lsa, qo‘shimcha ma’lumotlarni saqlash imkonsiz bo‘ladi. Ruxsat etilgan o‘lchamdagi massiv statik massiv deb ataladi.
- Massivga talab qilinganidan kamroq xotira ajratish ma’lumotlarning yo‘qolishiga olib keladi.
- Massiv bir hil turdagi ma’lumotlarni saqlaydi, shuning uchun bitta massiv turli xil ma’lumotlar turlarining qiymatlarini saqlay olmaydi.
- Massivlar ma’lumotlarni qo‘shni xotira joylarida saqlaydi, bu esa o‘chirish va kiritishni amalga oshirishni juda qiyinlashtiradi. Ushbu muammo elementlarga ketma-ket kirishga imkon beradigan bog’langan ro‘yxatlarni amalga oshirish orqali bartaraf etiladi.

Massivlarning qo‘llanishi:

- Ular massiv ro‘yxatlari, uyumlar, Xesh jadvallar, vektorlar va matritsalar kabi boshqa ma’lumotlar tuzilmalarini amalga oshirishda ishlatiladi.
- Ma’lumotlar bazasi yozuvlari odatda massiv sifatida amalga oshiriladi.

- Massivlar kompyuter orqali qidirish jadvallarida ishlatiladi.
- Bu kabi turli xil saralash algoritmlari uchun ishlatiladi qabariq saralash kiritish saralash, birlashtirish saralash va tezsaralash.

Mavzu yuzasidan savollar:

1. Dasturlashda massiv nima?
2. C/C++ dasturlash tilida massivni eleon qilish.
3. Massivlarning ba’zi afzalliklari va kamchiliklarini eslatib o‘ting.
4. Massivda asosiy operatsiyalarni bajarish uchun vaqt murakkabligi qanday?
5. Massivlarni bog’langan ro‘yxatlar bilan Solishtiring.