**Stack**

**Stack** – Stack inglizchadan bogʻlam degan ma’noni anglatadi.

**Stack** - bu LIFO (last in – first out; oxirgi kelgan – birinchi ketadi) prinsipi boʻyicha ishlaydigan ma’lumotlar strukturasi.

**Stackda** massivdagi kabi indekslar mavjud emas, ma’lum bir elementga murojaat qila olmaysiz. **Stackning** massifdan farqi **Stackda** ma’lumotlar qo’shish va o’chirish mumkin.

**Stackdan** foydalanish uchun **<stack>** kutubxonasini chaqirish kerak. Stack yaratish uchun biz quyidagi sxema bilan ishlashimiz kerak:

**stack<tipi><Stack nomi>;**

**push(**n**)** - Stackka element qoʻshadi

**empty()** - Stack boʻshligini tekshirish

**top()** – yuqori elementi ko’rish

**pop()** – Stack yuqori elementini o’chirish

**peek()** – Steckni indexsiga murojaat qiladi

**Dasturi**

#include <iostream>

#include <stack>

using namespace std;

int main()

{

    stack<int> s; // Stack yaratish

    // empty() - Stack bo'shligini tekshirish

    if (s.empty())

    {

        cout << "Stack bo'sh" << endl;

    }

    else

    {

        cout << "Stack bo'sh emas" << endl;

    }

    // push() - Element qo'shish

    s.push(1);

    s.push(2);

    s.push(3);

    // size() - Stackning o'lchamini olish

    cout << "Stackning o'lchami: " << s.size() << endl;

    // top() - Eng yuqori elementni o'qish

    cout << "Eng yuqori element: " << s.top() << endl;

    // pop() - Eng yuqori elementni o'chirish

    s.pop();

    // top() - Yangi eng yuqori elementni o'qish

    cout << "Eng yuqori element: " << s.top() << endl;

    // empty() - Stack bo'shligini tekshirish

    if (s.empty())

    {

        cout << "Stack bo'sh" << endl;

    }

    else

    {

        cout << "Stack bo'sh emas" << endl;

    }

    return 0;

}

**Navbat**

**Navbat** - bu FIFO (First In - First Out - "birinchi kelgan – birinchi ketadi") prinsipi boʻyicha qurilgan ma’lumotlar strukturasi.

**Navbat** bilan ishlash uchun avval kutubxonasini kiritish <**queue**> kerak.

**queue<tipi><navbat nomi>;**

**push(**n**) –** Element qoshish

**pop() –** Birinchi elemmentni o’chirish

**front() – N**avbatni birinchi elemetga murojad qilish

**back() –** Oxirgi elemetga murjad qilish

**emty() –** Bo’shligini tekshirish

**top()** – yuqori elementi ko’rish

**Dasturi**

#include <iostream>

#include <queue>

using namespace std;

int main()

{

    queue<int> q; // Queue yaratish

    // empty() - Queue bo'shligini tekshirish

    if (q.empty())

    {

        cout << "Queue bo'sh" << endl;

    }

    else

    {

        cout << "Queue bo'sh emas" << endl;

    }

    // size() - Queuening o'lchamini olish

    cout << "Queuening o'lchami: " << q.size() << endl;

    // push() - Elementni ohiriga qo'shish

    q.push(1);

    q.push(2);

    q.push(3);

    // size() - Queuening yangi o'lchamini olish

    cout << "Queuening yangi o'lchami: " << q.size() << endl;

    // front() - Eng oldagi elementni o'qish

    cout << "Eng oldagi element: " << q.front() << endl;

    // back() - Eng ohiridagi elementni o'qish

    cout << "Eng ohiridagi element: " << q.back() << endl;

    // pop() - Eng oldagi elementni o'chirish

    q.pop();

    // front() - Yangi eng oldagi elementni o'qish

    cout << "Yangi eng oldagi element: " << q.front() << endl;

    // back() - Yangi eng ohiridagi elementni o'qish

    cout << "Yangi eng ohiridagi element: " << q.back() << endl;

    // empty() - Queue bo'shligini tekshirish

    if (q.empty())

    {

        cout << "Queue bo'sh" << endl;

    }

    else

    {

        cout << "Queue bo'sh emas" << endl;

    }

    return 0;

}