STACK MA’LUMOTLAR TUZILMASI VA UNING METODLARI BILAN ISHLASH

РАБОТА СО СТРУКТУРОЙ ДАННЫХ СТЕК И ЕЁ МЕТОДАМИ

WORKING WITH THE STACK DATA STRUCTURE AND ITS METHODS

Abdullayev Shaxboz Solijon o‘g‘li
FarDU Axborot texnologiyalari kafedrasi katta o'qituvchisi
shaxbozfardu2023@gmail.com
ORCID ID 0000-0001-9382-732X
Qurbonaliyev Pirimboy Oybek o’g’li
FarDU Axborot tizimlari va texnologiyalari yo’nalishi 1-kurs talabasi
qurbonaliyevpirimboy@gmail.com

Annotatsiya. *Stack — bu **LIFO (Last In, First Out — oxirgi kirgan birinchi chiqadi)** printsipiga asoslangan ma’lumotlar tuzilmasidir. Stack dasturlashning muhim elementlaridan biri bo‘lib, arifmetik ifodalarni hisoblash, funksiyalar chaqiruvini boshqarish, qavslar muvozanatini tekshirish, fayl tizimlarida navigatsiya kabi jarayonlarda keng qo‘llaniladi. Uning asosiy metodlari — **push** (element qo‘shish), **pop** (olib tashlash), **peek/top** (yuqoridagi elementni ko‘rish) va **isEmpty** (bo‘shligini tekshirish) — barcha asosiy dasturlash tillarida oddiy shaklda amalga oshiriladi.*

Stackning soddaligi, tushunarli ishlash printsipi va real hayotdagi ko‘plab masalalarga mos kelishi uni o‘rganish va dasturga tatbiq etishda juda qulay qiladi. Ushbu maqolada stack ma’lumotlar tuzilmasining nazariy jihatlari, amaliy qo‘llanilishi, asosiy metodlari va turli dasturlash tillarida implementatsiyasi misollar bilan yoritiladi.

Аннотация. ***Стек** — это структура данных, основанная на принципе **LIFO (последним пришёл — первым вышел)**. Он является одним из ключевых элементов программирования и широко используется при решении задач, таких как вычисление арифметических выражений, управление вызовами функций, проверка баланса скобок и навигация в файловых системах. Основные операции стека — **push** (добавление элемента), **pop** (удаление верхнего элемента), **peek/top** (просмотр верхнего элемента) и **isEmpty** (проверка на*

пустоту) — реализуются просто на всех популярных языках программирования.

Благодаря своей простоте, понятной логике и практической применимости в реальных задачах стек является удобной структурой для изучения и использования. В данной статье рассматриваются теоретические основы стека, его практическое применение, основные методы и реализация на различных языках программирования с примерами.

Abstract. *A **stack** is a data structure based on the **LIFO (Last In, First Out)** principle. It is one of the essential components in programming and is widely used in tasks such as evaluating arithmetic expressions, managing function calls, checking the balance of parentheses, and navigating file systems. Its main operations — **push** (add an element), **pop** (remove the top element), **peek/top** (view the top element), and **isEmpty** (check if the stack is empty) — are simple to implement in all major programming languages.*

Due to its simplicity, intuitive logic, and practical use in real-world problems, the stack is an ideal structure to learn and apply in various programming tasks. This article explores the theoretical background of the stack, its practical applications, fundamental methods, and implementation in different programming languages with relevant examples.

Kalit so‘zlar: *stack, ma’lumotlar tuzilmasi, LIFO printsipi, push metodi, pop metodi, peek funksiyasi, bo‘shlikni tekshirish, funksiya chaqiruvlari, arifmetik ifodalar, dasturlash tillari, nazariy asoslar, amaliy qo‘llanilish*

Ключевые слова: *stack, data structure, LIFO principle, push method, pop method, peek function, checking for emptiness, function calls, arithmetic expressions, programming languages, theoretical foundations, practical applications*

Keywords: *stack, структура данных, принцип LIFO, метод push, метод pop, функция peek, проверка на пустоту, вызовы функций, арифметические выражения, языки программирования, теоретические основы, практическое применение*

KIRISH

Dasturlashda ma’lumotlar tuzilmalari dastur samaradorligi va unumdorligini ta'minlashda asosiy rol o‘ynaydi. Dasturchilar ma’lumotlarni samarali saqlash, tezkor ishlov berish va ularga oson kirishni ta’minlash uchun turli xil ma’lumotlar tuzilmalaridan foydalanadilar. Ularning har biri ma’lum bir turdagi vazifalar uchun moslashtirilgan bo‘lib, o‘zining afzalliklari va kamchiliklariga ega. Shu ma'noda, **stack** ma’lumotlar tuzilmasi juda muhim o‘rin tutadi. Stack, **LIFO (Last In, First**

Out), ya'ni oxirgi kirgan, birinchi chiqadigan prinsipiga asoslanadi. Bu prinsip orqali stackda ma’lumotlar kiritiladi va olingan ma’lumotlar faqat oxirgi kiritilgan elementdan boshlanadi.

Stack ma’lumotlar tuzilmasining ishlash printsipi juda soddadir, ammo uning amaliy qo‘llanilishi juda keng. Stackning asosiy xususiyati shundaki, unga faqat oxirgi kirgan elementga kirish mumkin, bu esa uni juda samarali qiladi, ayniqsa, **rekursiya** va **funksiya chaqiruvlari** bilan ishlashda. Stackni turli xil dasturlash tillarida ishlatish imkoniyati mavjud va u turli masalalarni hal qilishda, jumladan, **arifmetik ifodalar**, **skobka muvozanatini tekshirish**, va **kecha/kunduz navigatsiyasi** kabi jarayonlarni boshqarishda keng qo‘llaniladi.

Stackning eng muhim metodlari: **push** (element qo‘shish), **pop** (elementni olib tashlash), **peek** (yuqoridagi elementni ko‘rish), va **isEmpty** (bo‘shligini tekshirish). Ushbu metodlar stackni samarali va oson ishlatishni ta’minlaydi. Masalan, **push** metodi bilan yangi element qo‘shilganda, u stackning yuqori qismiga joylashadi. **Pop** metodi esa stackning yuqorisidagi elementni olib tashlaydi. **Peek** metodi esa stackning yuqorisidagi elementni ko‘rishga imkon beradi, ammo uni olib tashlamaydi.

Stackning amaliy qo‘llanilishi juda keng bo‘lib, ularni hisoblash, optimallashtirish, muvozanatni tekshirish kabi ko‘plab sohalarda uchratish mumkin. Dastur yozishda stackni qo‘llash dasturchiga ma’lumotlarni boshqarish va tizimli tarzda ishlash imkonini beradi. Ushbu maqolada stack ma’lumotlar tuzilmasining nazariy asoslari, uning metodlari va amaliy qo‘llanilishi haqida batafsil ma’lumotlar beriladi. Stackni dasturlashda qanday ishlatish, qanday metodlardan foydalanish va qanday vazifalarni hal qilish mumkinligi ko‘rsatiladi. Shuningdek, stackning turli dasturlash tillarida qanday implementatsiya qilinishi haqida misollar keltiriladi.

Dasturning kodi :

```
#include <iostream>
#include <stack>
using namespace std;
class Stack {
private:
   stack<int> st;
public:
   void push(int item) {
      st.push(item); }
   void pop() {
      if (!st.empty()) {
```

```
            cout << "Pop: " << st.top() << endl;
        st.pop();
        } else {
            cout << "Stack bo'sh!" << endl;  }  }
        void peek() {
        if (!st.empty()) {
            cout << "Peek: " << st.top() << endl;
        } else {
            cout << "Stack bo'sh!" << endl; }  }
        bool isEmpty() {
        return st.empty();   }
        void display() {
        if (!st.empty()) {
            cout << "Stack: ";
            stack<int> temp = st;
            while (!temp.empty()) {
                cout << temp.top() << " ";
                temp.pop(); }
            cout << endl;
        } else {
            cout << "Stack bo'sh!" << endl;}}} ;
    int main() {
        Stack stack;
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.display();
        stack.peek();
        stack.pop();
        stack.display();
        cout << "Is stack empty? " << (stack.isEmpty() ? "Yes" : "No") << endl;
        return 0;}
```

Dastur tahlili

#include <iostream> - Bu kutubxona fayli konsolga ma'lumot chiqarish (cout) va konsoldan ma'lumot olish (cin) imkoniyatlarini beradi. Bu satr dasturda ishlatilgan barcha konsol bilan aloqador funktsiyalarni chaqirishi uchun zarur.

#include <stack> - Bu satr stack ma'lumotlar tuzilmasi uchun kerakli kutubxonani dasturga qo‘shadi. stack — bu C++ ning STL (Standard Template Library) kutubxonasidan birining komponenti bo‘lib, u stack (o‘zgaruvchan) ma'lumotlar tuzilmasi bilan ishlashni osonlashtiradi.

using namespace std; - Bu satr std nomli kutubxonani dasturga kiritadi. std namespace ichidagi barcha elementlardan bevosita foydalanish imkonini beradi. Masalan, cout, cin, string kabi elementlar to'g'ridan-to'g'ri chaqiriladi va "std::" prefiksini qo'llash kerak emas.

int main() - Dastur bajarilishi shu funksiyadan boshlanadi.

class Stack { ... } - Bu satr **Stack** nomli **klass** ni yaratadi. Klass — bu ma'lumotlar tuzilmasini va u bilan bog'liq metodlarni bir joyga jamlash imkonini beradi. Bu yerda biz stack (yig'ish) ma'lumotlar tuzilmasini yaratish uchun **Stack** klassini yaratmoqdamiz.

private: - bu bo'limda, klass ichidagi ma'lumotlarga **private** (maxfiy) bo‘lishi kerak bo‘lgan elementlar e'lon qilinadi. **private** ma'lumotlar faqatgina klassning o'z metodlari orqali kirishish mumkin.

stack<int> st; - bu satrda st nomli **stack** obyekti yaratiladi. Stack — bu ma'lumotlar tuzilmasi bo‘lib, u o‘zining asosiy operatsiyalaridan biri bo‘lgan **LIFO (Last In, First Out)** tamoyili asosida ishlaydi, ya'ni oxirgi qo‘shilgan element birinchi bo‘lib chiqariladi. Bu yerda **int** turidagi elementlarni saqlash uchun stack yaratilgan.

public: - bu bo'limda **public** (ommaviy) metodlar e'lon qilinadi. **public** metodlar klassdan tashqaridan, ya'ni boshqa funksiyalar yoki obyektlar tomonidan chaqirilishi mumkin.

void push(int item) { st.push(item); } - bu metod **stackga element qo‘shish** uchun ishlatiladi. **push()** metodiga berilgan **item** argumenti stackning oxiriga qo‘shiladi. Bu yerda st.push(item); kodi stackning oxiriga **item** elementini qo‘shadi.

void pop() { ... } - bu metod stackdan element olib tashlash uchun ishlatiladi. pop() metodidan foydalanish stackning oxiridagi elementni olib tashlaydi. Agar stack bo‘sh bo‘lsa, foydalanuvchiga "Stack bo'sh!" xabari chiqariladi.

if (!st.empty()) { ... } — bu satr stackning bo‘sh emasligini tekshiradi. Agar stack bo‘sh bo‘lmasa, st.top() kodi stackning yuqori (oxirgi) elementini chiqarib, **pop()** operatsiyasini amalga oshiradi.

void peek() { ... } - bu metod stackning yuqori (oxirgi) elementini ko‘rish uchun ishlatiladi, ammo **pop** operatsiyasini bajarilmaydi. st.top() kodi stackning yuqori elementini ko‘rsatadi, lekin uni olib tashlamaydi. Agar stack bo‘sh bo‘lsa, "Stack bo'sh!" xabari chiqadi**.**

bool isEmpty() { return st.empty(); } - bu metod stackning bo‘sh yoki bo‘sh emasligini tekshiradi. Agar stack bo‘sh bo‘lsa, st.empty() true qaytaradi, aks holda false qaytaradi.

void display() { ... } - bu metod stackning barcha elementlarini ekranga chiqaradi. Stackdagi elementlarni ko‘rsatish uchun vaqtincha stack<int> temp = st; kodi bilan stackning nusxasi olinadi, chunki C++ stack konteynerining ichidagi elementlarni to‘g‘ridan-to‘g‘ri ko‘rish mumkin emas. temp.top() kodi stackning yuqori elementini ko‘rsatadi va keyin **pop()** metodidan foydalanib olib tashlanadi.

13. int main() { ... } - bu yerda **main()** funksiyasi dastur ishga tushiriladigan asosiy funksiya.

stack.push(10); — 10 raqami stackga qo‘shiladi.

stack.push(20); — 20 raqami stackga qo‘shiladi.

stack.push(30); — 30 raqami stackga qo‘shiladi.

stack.display(); — stackdagi barcha elementlarni ekranga chiqaradi.

stack.peek(); — stackning yuqori elementini ko‘rsatadi.

stack.pop(); — stackdan element olib tashlaydi (yuqoridan).

stack.display(); — pop operatsiyasidan keyingi stackni ko‘rsatadi.

cout << "Is stack empty? " << (stack.isEmpty() ? "Yes" : "No") << endl; — stack bo‘shmi yoki yo‘qligini tekshiradi va ekranga chiqaradi.

return 0; - return 0 satri dasturning to'liq yakunlangani va xatolik yo'qligini bildiradi.

Dasturning ishlash prinsipi:

```
Stack: 30 20 10
Peek: 30
Pop: 30
Stack: 20 10
Is stack empty? No

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

Dastur natijasi haqida qisqacha izoh:

Stack: 30 20 10 — stackda hozircha 3 element bor.

Peek: 30 — stackning yuqori elementi 30.

Pop: 30 — stackdan yuqori element (30) olib tashlandi.

Stack: 20 10 — stackda 30 olib tashlanganidan keyin faqat 20 va 10 qoladi.

Is stack empty? No — stack bo'sh emas, 2 ta element qolgan.

XULOSA

Ushbu dastur **stack** ma'lumotlar tuzilmasining asosiy operatsiyalarini amalga oshiradi. Dasturda **push**, **pop**, **peek**, **isEmpty**, va **display** metodlari yordamida stackning ishlash jarayonlari namoyish etildi. Dasturning ishlash jarayonida stackning yuqori elementi qo‘shish, o'chirish, ko‘rish va boshqarish kabi operatsiyalarni bajarish imkonini beradigan metodlar muvaffaqiyatli ishladi. Stackning LIFO (Last In, First Out) tamoyiliga asosan, oxirgi qo‘shilgan element birinchi olib tashlanadi.

Stack ma'lumotlar tuzilmasi asosan ma'lumotlarni saqlash va boshqarish uchun qo‘llaniladi, ayniqsa uning tezkor ishlash xususiyatlari va resurslar tejash imkoniyatlari tufayli turli dasturlarda, masalan, dasturlash tillarida, rekursiv funksiyalarni bajarishda yoki brauzer tarixini saqlashda keng foydalaniladi. Bu dastur stackning ishlash tamoyillarini tushunish va uning qanday ishlashini o‘rganish uchun qulay namunadir.

FOYDALANILGAN ADABIYOTLAR:

1. **Stroustrup, B. (2013).** *The C++ Programming Language* (4th ed.). Addison-Wesley.
2. **Meyers, S. (2005).** *Effective C++: 55 Specific Ways to Improve Your Programs and Designs* (3rd ed.). Addison-Wesley.
3. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009).** *Introduction to Algorithms* (3rd ed.). MIT Press.
4. **Gaddis, T. (2015).** *Starting Out with C++: From Control Structures through Objects* (8th ed.). Pearson.
5. **ISO/IEC. (2011).** *ISO/IEC 14882:2011 Information technology — Programming languages — C++*. International Organization for Standardization.
6. **GeeksforGeeks.** (n.d.). *C++ Stack Data Structure*. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/stack-data-structure-and-algorithms/