

**DASTURLASH TILLARIDA BERILGANLARNI CHIZIQLI
STRUKTURASINI IFODALASH**

**ПРЕДСТАВЛЕНИЕ ЛИНЕЙНЫХ СТРУКТУР ДАННЫХ В ЯЗЫКАХ
ПРОГРАММИРОВАНИЯ**

**REPRESENTATION OF LINEAR DATA STRUCTURES IN
PROGRAMMING LANGUAGES**

Abdullayev Shaxboz Solijon o‘g‘li

Farg‘ona davlat universiteti Axborot texnologiyalari kafedrasi katta o‘qituvchisi
shaxbozfaridu2023@gmail.com

Isoqjonov Barkamol Shuhratjonovich

Farg‘ona davlat universiteti Axborot tizimlari va texnologiyalar yo‘nalishi
1-kurs talabasi
ziyatovbarkamol@gmail.com

Annotatsiya: Ushbu ilmiy maqola zamonaviy C++ (C++11, C++14, C++17, C++20 standartlari) da chiziqli ma'lumotlar strukturalarining ilg'or usullari bilan ishlashga bag'ishlangan. Maqola nafaqat klassik STL konteynerlarini, balki ularning ichki ishlash mexanizmlari, xotira boshqaruvi, iteratorlar va parallel ishlash usullarini chuqur o'rganadi. Har bir bo'limda amaliy loyihalarda uchraydigan muammolar va ularning yechimlari bilan bog'liq detalli misollar keltirilgan. Maqola professional dasturchilar, ma'lumotlar strukturalari bo'yicha tadqiqot olib borayotgan talabalar hamda algoritmlar muxlislari uchun qimmatli resurs hisoblanadi.

Аннотация: Данная научная статья посвящена работе с линейными структурами данных с использованием современных стандартов C++ (C++11, C++14, C++17, C++20). В статье рассматриваются не только классические контейнеры STL, но и внутренние механизмы их работы, управление памятью, итераторы и методы параллельной обработки. В каждом разделе приводятся подробные примеры, связанные с проблемами, возникающими в прикладных проектах, и способами их решения. Статья представляет собой ценный ресурс

для профессиональных программистов, студентов, исследующих структуры данных, а также любителей алгоритмов.

Abstract: *This scientific article is dedicated to working with linear data structures using modern C++ standards (C++11, C++14, C++17, and C++20). It explores not only the classic STL containers but also delves into their internal mechanisms, memory management, iterators, and methods of parallel processing. Each section includes detailed examples related to real-world problems encountered in practical projects, along with their solutions. The article serves as a valuable resource for professional developers, students researching data structures, and algorithm enthusiasts.*

Kalit so‘zlar: *chiziqli ma’lumotlar strukturasi, massiv (Array), bog‘langan ro‘yxat (Linked List), to‘plam (Stack) – LIFO (Last In, First Out), navbat (Queue) – FIFO (First In, First Out), dasturlash tillari (Python, Java, C++), ma’lumotlar tuzilmasi samaradorligi.*

Ключевые слова: *линейная структура данных, массив (Array), связанный список (Linked List), стек (Stack) – LIFO (Last In, First Out), очередь (Queue) – FIFO (First In, First Out), языки программирования (Python, Java, C++), эффективность структуры данных.*

Keywords: *linear data structure, array, linked list, stack – LIFO (Last In, First Out), queue – FIFO (First In, First Out), programming languages (Python, Java, C++), data structure efficiency.*

Kirish

Dasturlashning fundamental tamoyillaridan biri - ma'lumotlarni samarali tashkil etish va boshqarishdir. C++ dasturlash tili, o'zining past darajadagi xotira boshqaruvi va yuqori darajadagi abstraksiyalarni uyg'unlashtirish qobiliyati tufayli, ma'lumotlar strukturalarini amalga oshirishda ayniqsa kuchli vosita hisoblanadi. Ushbu maqolada biz chiziqli ma'lumotlar strukturalarining C++ dagi ilg'or amaliyotlarini chuqur o'rganamiz.

Chiziqli struktura — bu elementlar ketma-ket joylashgan ma'lumotlar tuzilmasi bo'lib, har bir element aniq oldingi va keyingi elementga bog'langan bo'ladi. Ular birinchi elementdan boshlab oxirgi elementgacha tartibda ko'riladi.

Tarixiy Kontekst

Chiziqli ma'lumotlar strukturalarining rivojlanishi kompyuter fanlarining dastlabki kunlariga borib taqaladi. 1950-yillarda yaratilgan birinchi dasturlash tillaridan boshlab, bugungi kundagi zamonaviy C++20 standartigacha bo'lgan yo'lda ushbu strukturalar sezilarli evolyutsiyadan o'tdi:

Fortran davri (1957) - Statik massivlar

C davri (1972) - Ko'rsatkichlar va structlar yordamida dinamik strukturalar

C++98/03 - STL (Standard Template Library) ning paydo bo'lishi

C++11/14 - Move semantikasi va smart pointerlar

C++17/20 - Parallel algoritmlar va lock-free strukturalar

Massivlar: Advanced Usullar

```
#include <array>
```

```
#include <iostream>
```

```
template<typename T, size_t N>
```

```
void print_array(const std::array<T, N>& arr) {
```

```
    // CTAD (C++17 Class Template Argument Deduction)
```

```
    for(const auto& elem : arr) {
```

```
        std::cout << elem << " "; }
```

```
    std::cout << "\n";}
```

```
int main() { // constexpr massiv (C++11)
```

```
constexpr std::array<int, 5> arr = {1, 2, 3, 4, 5};
```

```
    // Structured binding (C++17)
```

```
    auto [a, b, c, d, e] = arr;
```

```
    // Compile-time o'lcham tekshiruvi
```

```
    static_assert(arr.size() == 5, "Massiv o'lchami noto'g'ri");
```

```
    print_array(arr);
```

```
    return 0;}
```

Bog'langan Ro'yxatlar

```
#include <iostream>
```

```
#include <list>
```

```
using namespace std;
```

```
int main() {
```

```
    list<int> myList = {1, 2, 3, 4, 5};
```

```
    // Element qo'shish
```

```
myList.push_back(6);
myList.push_front(0);
// Ro'yxatni chiqarish
cout << "Ro'yxat elementlari: ";
for(int x : myList) {
    cout << x << " "; }
cout << endl;
return 0;}
```

To'plamlar (Steklar)

```
#include <iostream>
#include <stack>
using namespace std;
int main() { stack<int> myStack;
// Elementlar qo'shish
myStack.push(10);
myStack.push(20);
myStack.push(30);
// Element olish
cout << "Eng ustki element: " << myStack.top() << endl;
myStack.pop();
cout << "Yangi ustki element: " << myStack.top() << endl;
return 0;}
```

Navbatlar

```
#include <iostream>
#include <queue>
using namespace std;
int main() { queue<string> myQueue;
// Elementlar qo'shish
myQueue.push("Bir");
myQueue.push("Ikki");
myQueue.push("Uch");
// Element olish
cout << "Navbatdagi birinchi element: " << myQueue.front() << endl;
```

```
myQueue.pop();
```

```
cout << "Yangi birinchi element: " << myQueue.front() << endl;
```

```
return 0;}
```

Qayerda ishlatiladi ? (Amaliy misollar)

Soha	Misol
Bank dasturlari	Navbatni boshqarish (Queue)
Brauzerlar	Orqaga qaytish (Stack)
O‘yinlar	O‘yinchi harakatlarini saqlash (Linked List)
Matn tahrirlovchilar	Belgilar ketma-ketligi (String)
E-commerce saytlar	Mahsulot ro‘yxatlari (Vector yoki Array)

C++ tilida chiziqli strukturaga misollar:

Strukturasi	Tavsifi	C++ da qanday ifodalanadi
Massiv (Array)	Fiksirlangan o‘lchamdagi ketma-ket elementlar to‘plami	int arr[5];
Vector	Dinamik o‘sadigan massiv	std::vector<int> v;
String	Belgilar ketma-ketligi	std::string s;
Stack (Stek)	Oxirgi kiritilgan element birinchi olinadi (LIFO)	std::stack<int> s;
Queue (Navbat)	Birinchi kiritilgan element birinchi olinadi (FIFO)	std::queue<int> q;
Linked List	Har bir element keyingisiga ko‘rsatkich saqlaydi	struct Node { int data; Node* next; };

Xulosa

C++ da chiziqli ma'lumotlar strukturalarini ishlatish nafaqat standart kutubxona funktsionalligi bilan chegaralanmaydi. Professional dasturlashda quyidagi jihatlarni hisobga olish muhim:

Xotira boshqaruvi: Modern C++ da smart pointerlar (`unique_ptr`, `shared_ptr`) va custom allocatorlar yordamida xotira xavfsizligini ta'minlash.

Parallel ishlash: Mutexlar, atomic operatsiyalar va lock-free algoritmlar yordamida multithread muhitda ishlash.

Move semantics: Ma'lumotlarni nusxalash o'rniga ko'chirish orqali samaradorlikni oshirish.

Compile-time optimizatsiyalar: `constexpr`, `static_assert` va template metaprogramming usullari.

Kelajakda chiziqli strukturalarni rivojlantirishda quyidagi yo'nalishlar muhim bo'ladi:

Heterogen strukturalar: `std::variant` (C++17) va `std::any` (C++17) yordamida turli tipdagi ma'lumotlarni saqlash.

Coroutine lar: C++20 da kiritilgan coroutinelar yordamida asinxron operatsiyalarni boshqarish.

SIMD optimizatsiyalar: Vectorized operatsiyalar yordamida massivlar ustida amallarni tezlashtirish.

Dasturchi sifatida nafaqat ma'lumotlar strukturasini tanlash, balki uning ichki implementatsiyasini tushunish va platformaga xos optimizatsiyalarni qo'llash - bu sizning dasturlaringizni raqobatbardosh qilishning kalitidir.

Foydalanilgan adabiyotlar:

1. Abelson, H., & Sussman, G. J. Structure and Interpretation of Computer Programs MIT Press, 2nd Edition, 1996.
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms MIT Press, 3rd Edition, 2009.
3. Sedgewick, R., & Wayne, K. Algorithms (4th Edition) Addison-Wesley, 2011.
4. Lafore, R. Data Structures and Algorithms in Java Sams Publishing, 2002.
5. Python Software Foundation The Python Standard Library – Collections Module (Deque, List, etc.)
<https://docs.python.org/3/library/collections.html>
6. Forouzan, B. A. Data Structures: A Pseudocode Approach with C Cengage Learning, 2nd Edition, 2003.
7. Tanenbaum, A. S. Structured Computer Organization Pearson, 6th Edition, 2013.