C++ DASTURLASH TILIDA MASSIVLARDAN FOYDALANISH

ИСПОЛЬЗОВАНИЕ МАССИВОВ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

USING ARRAYS IN THE C++ PROGRAMMING LANGUAGE

Shermatova Hilola Mirzayevna
Farg‘ona davlat universiteti Axborot texnologiyalari kafedrasi dotsenti
shermatovahilola1978@gmail.com
Bo’taboyeva Marjona Nurali qizi
Farg‘ona davlat universiteti Axborot tizimlari va texnologiyalar yo‘nalishi 1-kurs talabasi

Annotatsiya: *C++ dasturlash tilida massivlar ma’lumotlarni tartibli saqlash va ularga tezkor murojaat qilish imkonini beradi. Ushbu mavzu massivlarning turlari, ular bilan ishlash usullari va amaliy qo‘llanilishi haqida tushuncha beradi.*

Аннотация: *Массивы в языке программирования C++ позволяют упорядоченно хранить данные и быстро к ним обращаться. Данная тема дает представление о типах массивов, методах работы с ними и их практическом применении.*

Abstract: *Arrays in the C++ programming language allow for structured data storage and quick access. This topic provides an understanding of array types, methods of working with them, and their practical applications.*

Kalit so’zlar: *massiv, C++, ma'lumotlar tuzilmasi, indeks, dinamik massiv, statik massiv, tezkor murojaat.*

Ключевые слова: *массив, C++, структура данных, индекс, динамический массив, статический массив, быстрый доступ.*

Keywords: *array, C++, data structure, index, dynamic array, static array, fast access.*

KIRISH

Zamonaviy dasturlash tillari, jumladan, C++ tilida ma’lumotlarni saqlash va ulardan samarali foydalanish dastur kodining asosiy jihatlaridan biridir. Dasturlash jarayonida ba’zan bir xil turdagi ma’lumotlarni guruhlash va ulardan to‘plam sifatida foydalanish zarurati paydo bo‘ladi. Bu kabi holatlarda massivlar muhim rol o‘ynaydi.

Massivlar — bir xil turdagi ma’lumotlarni ketma-ket tartibda saqlashga imkon beruvchi ma’lumot tuzilmasi bo‘lib, ularning yordamida ma’lumotlarga tezkor va qulay tarzda ishlov berish mumkin.

C++ tilida massivlar statik va dinamik shaklda e’lon qilinadi. Statik massivlarning o‘lchami dastur bajarilishidan oldin aniqlanadi va o‘zgarmaydi. Dinamik massivlar esa dastur ishlash jarayonida foydalanuvchi tomonidan ajratilgan xotirada joylashadi va zarur bo‘lganda o‘z o‘lchamini o‘zgartirishi mumkin. Bu xususiyat C++ dasturlarining moslashuvchanligini oshiradi va xotira resurslaridan samarali foydalanishga imkon beradi.

Massivlarning dasturlashda tutgan o‘rni juda katta. Ular matematik hisob-kitoblardan tortib, real hayotdagi ma’lumotlarni saqlash va qayta ishlash jarayonlarigacha keng qo‘llaniladi. Masalan, talabalar baholari, harorat ko‘rsatkichlari, o‘yinlardagi xarakterlar holati kabi ko‘plab ma’lumotlar massivlar yordamida boshqariladi. Bundan tashqari, ikki o‘lchovli massivlar matritsalar bilan ishlashda, algoritmlar va ma’lumotlarni strukturalashda keng qo‘llanadi.

Ushbu maqolada C++ dasturlash tilida massivlardan foydalanish asoslari, ularning turlari, xotira bilan ishlash usullari hamda turli amaliy misollar ko‘rib chiqiladi. Massivlar bilan ishlash bo‘yicha asosiy tushunchalar, massiv elementlariga murojaat qilish, ularni to‘ldirish va o‘zgartirish jarayonlari batafsil tahlil qilinadi. Bundan tashqari, massivlardan samarali foydalanishning amaliy jihatlari ham ko‘rib chiqilib, ularning dastur tezligi va samaradorligiga ta’siri haqida tushuncha beriladi.

C++ tilida massivlar quyidagi turlarga bo‘linadi:

1. Statik massivlar – oldindan belgilangan o‘lchamga ega bo‘lgan massivlar. Ular xotira joylashuvi jihatidan samarador bo‘lib, kompilyatsiya vaqtida belgilangan o‘lcham bilan ishlaydi.

2. Dinamik massivlar – dastur ishlash jarayonida hajmi o‘zgarishi mumkin bo‘lgan massivlar. Ular xotira boshqaruvi jihatidan moslashuvchan bo‘lib, kerakli vaqtda o‘lchamini o‘zgartirish imkonini beradi

3.Ko‘p o‘lchamli massivlar – ikki yoki undan ortiq o‘lchamga ega bo‘lgan massivlar bo‘lib, ular matritsa va boshqa murakkab ma’lumotlar tuzilmalarini saqlash uchun ishlatiladi.

C++ tilida massivlarni e’lon qilish quyidagi sintaksis orqali amalga oshiriladi:

```
#include <iostream>
using namespace std;
int main() {
int sonlar[5] = {1, 2, 3, 4, 5};
```

```
 cout << "Oddiy massiv elementlari: ";
  for (int i = 0; i < 5; i++) {
     cout << sonlar[i] << " "; }
  cout << endl;
  int  d_massiv = new int[10];
  for (int i = 0; i < 10; i++) {
     d_massiv[i] = i * 2; }
  cout << "Dinamik massiv elementlari: ";
  for (int i = 0; i < 10; i++) {
     cout << d_massiv[i] << " ";   }
  cout << endl;
  delete[] d_massiv;
  int matritsa[3][3] =
{ {1, 2, 3},
     {4, 5, 6},
     {7, 8, 9}   };
  cout << "Ikki o'lchamli massiv (3x3):" << endl;
  for (int i = 0; i < 3; i++) {
     for (int j = 0; j < 3; j++) {
        cout << matritsa[i][j] << " ";   }
     cout << endl;  }
return 0;}
```

#include <iostream> – iostream kutubxonasi **C++ standarti** bo‘lib, ekranga chiqish (cout) va foydalanuvchidan kiritish (cin) uchun ishlatiladi.

using namespace std; – std::cout, std::cin deb yozmaslik uchun kerak. Agar bu bo‘lmasa, har bir chiqish va kiritish operatori oldidan **std::** yozish kerak bo‘ladi.

int main() – Har qanday C++ dasturining boshlang‘ich nuqtasi. Dastur shu yerdan ishga tushadi.

int sonlar[5] = {1, 2, 3, 4, 5}; – Bu qatorda sonlar nomli 5 ta butun sondan iborat statik massiv e'lon qilinmoqda. Uning qiymatlari {1, 2, 3, 4, 5} sifatida berilgan.

cout << "Oddiy massiv elementlari: "; – bu satr ekranga "Oddiy massiv elementlari: " degan yozuvni chiqaradi.

for (int i = 0; i < 5; i++) {

cout << sonlar[i] << " "; } – Bu tsikl sonlar massivining barcha elementlarini ekranga chiqaradi.

cout << endl; – Bu kod yangi qatorga o‘tish uchun ishlatiladi.

int d_massiv = new int[10]; – Bu qatorda 10 ta elementga ega dinamik massiv (d_massiv) yaratilyapti. Unga new operatori orqali joy ajratiladi.

for (int i = 0; i < 10; i++) { d_massiv[i] = i * 2;} – Bu tsikl yordamida massivning har bir elementi i * 2 formulasi orqali to‘ldirilmoqda. Natijada massiv {0, 2, 4, 6, 8, 10, 12, 14, 16, 18} qiymatlariga ega bo‘ladi.

cout << "Dinamik massiv elementlari: "; – Bu ekranga "Dinamik massiv elementlari: " degan matnni chiqaradi.

for (int i = 0; i < 10; i++) {cout << d_massiv[i] << " "; } – Bu tsikl d_massiv elementlarini ekranga chiqaradi.

cout << endl; – Bu yangi qatorga o‘tish uchun ishlatiladi.

delete[] d_massiv; – Bu kod d_massiv uchun ajratilgan xotirani bo‘shatadi. Dinamik massiv ishlatilgandan keyin delete[] operatori orqali xotirani tozalash kerak.

int matritsa[3][3] =
{ {1, 2, 3},
{4, 5, 6},
{7, 8, 9} }; – Bu qatorda matritsa nomli 3x3 o‘lchamdagi ikki o‘lchovli massiv e’lon qilinmoqda.

U quyidagi shaklda tuzilgan:
1 2 3
4 5 6
7 8 9

cout << "Ikki o'lchamli massiv (3x3):" << endl; – Bu ekranga "Ikki o'lchamli massiv (3x3):" degan matnni chiqaradi.

for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
cout << matritsa[i][j] << " "; }
cout << endl; } – Bu kod ikki ichki tsikl yordamida matritsaning har bir elementini ekranga chiqaradi.

i – qator indeksini,
j – ustun indeksini bildiradi**.**

return 0; – dastur to‘g‘ri ishlaganligini bildiradi (0 qaytaradi).

Dasturning ishlash prinsipi

Oddiy massiv qatori:

Oddiy massiv elementlari: 1 2 3 4 5
Bu qatorda oddiy statik massivning qiymatlari ekranga chiqarilgan.

Dinamik massiv qatori:

Dinamik massiv elementlari : 0 2 4 6 8 10 12 14 16 18
Bu qator dinamik ravishda ajratilgan massiv elementlarini ko‘rsatadi. Har bir qiymat indeksning 2 baravari hisoblanadi (i * 2).

Ikki o'lchamli massiv (3x3):

1 2 3
4 5 6
7 8 9

Bu qismda 3x3 matritsa shaklida ikki o‘lchovli massiv ekranga chiqarilgan. Barcha elementlar qatorlar bo‘ylab joylashtirilgan va ekranda matritsa ko‘rinishida chiqarilgan.

Xulosa: C++ dasturlash tilida massivlar ma’lumotlarni tartibli saqlash va samarali boshqarish imkonini beradi. Ular statik, dinamik va ko‘p o‘lchamli shaklda e’lon qilinishi mumkin va dastur samaradorligini oshirishda muhim rol o‘ynaydi. Massivlardan foydalanish ma’lumotlarni optimallashtirish va algoritmlarni samarali bajarishda juda muhim hisoblanadi. Dinamik massivlardan foydalanishda xotira oqimlarini oldini olish uchun ehtiyotkorlik talab etiladi.

FOYDALANILGAN ADABIYOTLAR:

1. Bjarne Stroustrup. (2013). The C++ Programming Language (4th Edition). Addison-Wesley.
2. Herbert Schildt. (2003). C++: The Complete Reference (4th Edition). McGraw-Hill.
3. Scott Meyers. (2005). Effective C++ (3rd Edition). Addison-Wesley.
4. Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. (2012). C++ Primer (5th Edition). Addison-Wesley.
5. Nicolai M. Josuttis. (2012). The C++ Standard Library: A Tutorial and Reference. Addison-Wesley.
6. Stephen Prata. (2011). C++ Primer Plus (6th Edition). Pearson Education.
7. Bruce Eckel. (2000). Thinking in C++. Prentice Hall.