МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский Государственный Технологический Университет

**РЕФЕРАТ**

по дисциплине: «Основы Алгоритмизации и Программирования»

на тему: “преобразование двумерного массива в одномерный и обратно”

Выполнил студент группы

ФИТ, ПИ, 8 группы

Лужецкий Владислав Константинович

Оглавление

[Что такое массивы и зачем они нужны? 3](#_Toc152725338)

[Одномерные массивы 4](#_Toc152725339)

[Одномерные динамические массивы 4](#_Toc152725340)

[Двумерные массивы 5](#_Toc152725341)

[Многомерные динамические массивы 5](#_Toc152725342)

[Вывод 7](#_Toc152725343)

[Задача 7](#_Toc152725344)

# Что такое массивы и зачем они нужны?

**Массив - это структура данных, представляющая собой упорядоченный набор элементов одного типа, которые хранятся в памяти компьютера**. **Каждый элемент массива имеет свой уникальный индекс или номер, начиная с нуля.** Массивы являются фундаментальным инструментом в программировании и используются для хранения и обработки больших объемов данных.

**Преимущества Массивa:**

1. **Эффективность поиска по индексу:**

Массивы обеспечивают константное время доступа к элементам по индексу. Это означает, что **для поиска или изменения элемента массива не требуется проход по всему массиву. Достаточно знать индекс**, чтобы быстро получить доступ к нужному элементу.

1. **Удобство хранения последовательных данных:**

Массивы упорядочивают данные в последовательность, обеспечивая легкость доступа и управления данными. Это особенно важно, когда данные имеют логическую структуру, например, при работе с временными рядами или изображениями.

1. **Эффективное использование памяти:**

Память для массива выделяется непрерывным блоком, что обеспечивает компактное распределение данных. Это улучшает производительность, так как уменьшается количество кэш-промахов.

1. **Простота в использовании:**

Массивы предоставляют простой и понятный механизм для хранения и доступа к данным. Их легко использовать в коде, что упрощает написание и понимание программ.

1. **Эффективность в передаче аргументов**:

Передача массива в функцию обычно происходит по ссылке (или указателю), что позволяет избежать копирования больших объемов данных.

1. **Поддержка различных типов данных:**

Массивы в большинстве языков программирования могут содержать элементы любого типа данных. Это позволяет универсально использовать массивы для различных задач.

1. **Возможность создания многомерных структур данных**:

Массивы могут быть многомерными, что позволяет организовать данные в более сложные структуры, такие как матрицы или кубы.

**НО**:

Массивы имеют фиксированный размер после их создания.

Один из недостатков массивов - ограничение по размеру и неудобство изменения размера после создания.

# Одномерные массивы

**Одномерные массивы являются наиболее простым типом массивов. Они представляют собой линейную последовательность элементов, доступ к которым осуществляется по индексу.**

//Пример кода для объявления и использования одномерного массива

#include <iostream>

int main() {

int arr[5] = { 1, 2, 3, 4, 5 };

std::cout << "Первый элемент: " << arr[0] << std::endl;

std::cout << "Второй элемент: " << arr[1] << std::endl;

return 0;

}

# Одномерные динамические массивы

**Отличие одномерных динамических массивов от обычных заключается в изменении их размера во время выполнения программы. Это полезно, когда неизвестно заранее, сколько элементов будет содержать массив.**

//Пример кода для объявления и использования одномерного динамического массива

#include <iostream>

using namespace std;

int main() {

int size;

cout << "Введите размер массива: ";

cin >> size;

int\* numbers = new int[size];

//различные действия с массивом

delete[] numbers;

return 0;

}

# Двумерные массивы

**Двумерные массивы представляют собой таблицу элементов, упорядоченных в виде строк и столбцов. Они часто используются для представления матриц и таблиц данных.**

//Пример кода для объявления и использования двумерного массива

#include <iostream>

using namespace std;

int main() {

int arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

cout << arr[0][0] << endl;

return 0;

}

# Многомерные динамические массивы

**это массивы с более чем двумя измерениями, которые создаются во время выполнения программы и имеют изменяемый размер. Они представляют собой структуру данных, состоящую из нескольких уровней вложенности.**

Многомерные динамические массивы используются для представления и работы с данными, которые имеют более сложную структуру, требующую нескольких измерений для их организации. Примеры использования многомерных динамических массивов включают:

**Матрицы**: Многомерные массивы часто используются для представления матриц, которые являются таблицами элементов, упорядоченных в виде строк и столбцов. Матрицы используются в различных областях, таких как математика, физика, компьютерная графика и анализ данных.

**Изображения**: Изображения могут быть представлены в виде многомерных массивов пикселей. Каждый пиксель может иметь несколько компонентов, таких как красный, зеленый и синий цвета (для RGB-изображений) или яркость и насыщенность (для HSV-изображений).

**Таблицы данных**: Многомерные массивы могут использоваться для представления таблиц данных с несколькими измерениями, где каждое измерение соответствует определенным характеристикам или атрибутам данных.

**Игровые сетки**: В разработке компьютерных игр многомерные массивы могут использоваться для представления игровых сеток, таких как игровые уровни, карты мира или сетки коллизий.

Использование многомерных динамических массивов позволяет эффективно организовывать и обрабатывать сложные данные с более высоким уровнем структуры. Они предоставляют удобный способ доступа к элементам массива и манипуляций с данными, обеспечивая гибкость и эффективность в управлении памятью.

//пример кода для создания и использования многомерных динамических массивов

#include <iostream>

using namespace std;

int main() {

int rows, cols;

cout << "Введите количество строк: ";

cin >> rows;

cout << "Введите количество столбцов: ";

cin >> cols;

int\*\* arr = new int\* [rows];

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

arr[i] = new int[cols];

}

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

for (int j = 0; j < cols; j++) {

cout << "Введите элемент [" << i << "][" << j << "]: ";

cin >> arr[i][j];

}

}

cout << "Массив:\n";

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

for (int j = 0; j < cols; j++) {

cout << arr[i][j] << " ";

}

cout << endl;

}

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

delete[] arr[i];

}

delete[] arr;

return 0;

}

# Вывод

Таким образом массивы обеспечивают удобный способ хранения и доступа к данным. Элементы массива располагаются последовательно в памяти, что упрощает доступ к ним с помощью индексов. Это особенно полезно при работе с большим объемом данных, таких как списки, матрицы или изображения. Массивы обеспечивают упорядоченное размещение элементов, что позволяет легко получать доступ к элементам по их индексам. Индексы начинаются с 0 и последовательно увеличиваются до размера массива минус один. В целом, использование массивов является фундаментальной частью программирования и предоставляет мощный инструмент для хранения, доступа и обработки данных. Они широко применяются в различных областях программирования, включая алгоритмы, структуры данных, анализ данных, компьютерную графику, игровую разработку и многое другое.

# Задача

//Преобразование двумерного массива в одномерный и обратно:

//Вариант через индексы

#include <iostream>

using namespace std;

void main() {

setlocale(0, "");

int rows;

int cols;

cout << "Введите кол-во строк:";

cin >> rows;

cout << "Введите кол-во столбцов:";

cin >> cols;

int\*\* arr = new int\* [rows];

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

arr[i] = new int[cols];

}

// заполняем массив

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

for (int j = 0; j < cols; j++) {

arr[i][j] = rand() % 10;

}

}

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

for (int j = 0; j < cols; j++) {

cout << arr[i][j] << '\t';

}

cout << '\n';

}

// преобразуем двумерный массив в одномерный

int\* arr\_1 = new int[rows \* cols];

int k = 0;

cout << "\n Одномерный массив: ";

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

for (int j = 0; j < cols; j++) {

arr\_1[k] = arr[i][j];

cout << arr\_1[k] << ' ';

k++;

}

}

// очищаем память двумерного массива

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

delete[] arr[i];

}

delete[] arr;

// преобразуем одномерный массив обратно в двумерный

int\*\* new\_arr = new int\* [rows];

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

new\_arr[i] = new int[cols];

}

k = 0;

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

for (int j = 0; j < cols; j++) {

new\_arr[i][j] = arr\_1[i \* cols + j];

}

}

// выводим преобразованный двумерный массив

cout << "\nПреобразованный двумерный массив:\n";

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

for (int j = 0; j < cols; j++) {

cout << new\_arr[i][j] << '\t';

}

cout << '\n';

}

// очищаем память одномерного массива и преобразованного двумерного массива

delete[] arr\_1;

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

delete[] new\_arr[i];

}

delete[] new\_arr;

}

//Вариант через указатели

#include <iostream>

using namespace std;

int main() {

setlocale(0, "");

int rows;

int cols;

cout << "Введите кол-во строк:";

cin >> rows;

cout << "Введите кол-во столбцов:";

cin >> cols;

int\*\* arr = new int\* [rows];

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

arr[i] = new int[cols];

}

// заполняем массив

int\*\* p = arr;

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

int\* q = \*p;

for (int j = 0; j < cols; j++) {

\*q = rand() % 10;

q++;

}

p++;

}

p = arr;

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

int\* q = \*p;

for (int j = 0; j < cols; j++) {

cout << \*q << '\t';

q++;

}

cout << '\n';

p++;

}

// преобразуем двумерный массив в одномерный

int\* arr\_1 = new int[rows \* cols];

int\* p1 = arr\_1;

cout << "\n Одномерный массив: ";

p = arr;

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

int\* q = \*p;

for (int j = 0; j < cols; j++) {

\*p1 = \*q;

cout << \*p1 << ' ';

p1++;

q++;

}

p++;

}

// очищаем память двумерного массива

p = arr;

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

delete[] \* p;

p++;

}

delete[] arr;

// преобразуем одномерный массив обратно в двумерный

int\*\* new\_arr = new int\* [rows];

p1 = arr\_1;

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

new\_arr[i] = new int[cols];

int\* q = new\_arr[i];

for (int j = 0; j < cols; j++) {

\*q = \*p1;

q++;

p1++;

}

}

// выводим преобразованный двумерный массив

cout << "\nПреобразованный двумерный массив:\n";

p = new\_arr;

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

int\* q = \*p;

for (int j = 0; j < cols; j++) {

cout << \*q << '\t';

q++;

}

cout << '\n';

p++;

}

// очищаем память одномерного массива и преобразованного двумерного массива

delete[] arr\_1;

p = new\_arr;

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

delete[] \* p;

p++;

}

delete[] new\_arr;

return 0;

}

