new c++

Оператор **new** в C++ используется для динамического выделения памяти во время выполнения программы. Это означает, что программист может запросить определенное количество памяти во время выполнения программы, вместо того, чтобы использовать статический размер памяти, который определяется при компиляции.

```
int* arr = new int[size];
```

В этом конкретном примере, оператор **new** используется для выделения памяти для массива целых чисел, размер которого задается пользователем. Таким образом, программа может выделить достаточное количество памяти для массива во время выполнения, даже если размер массива не известен при компиляции.

Использование оператора **new** для выделения памяти для массива происходит в несколько этапов:

1. Оператор **new** запрашивает определенное количество памяти, которое нужно выделить под массив. В данном случае, это количество равно **size * sizeof(int)**, то есть размеру массива, умноженному на размер типа **int**.
2. Если память успешно выделена, оператор **new** возвращает указатель на первый элемент массива. В данном случае, это указатель **int* arr**.
3. После выделения памяти, программист может использовать указатель для доступа к элементам массива. Например, в данном примере мы используем указатель **arr** для заполнения массива случайными значениями и для вычисления суммы элементов массива.
4. После использования массива, программа должна освободить выделенную для него память с помощью оператора **delete[]**. В данном примере мы используем оператор **delete[] arr** для освобождения памяти, выделенной для массива.

Важно отметить, что при использовании оператора **new** для выделения памяти, программист самостоятельно отвечает за управление этой памятью. Ошибки в управлении динамической памятью могут привести к утечкам памяти или краху программы, поэтому необходимо тщательно следить за использованием операторов **new** и **delete[]**.

```
for (int i = 0; i < size; ++i) {

        // Получение значения элемента массива через указатель

        int value = *(arr + i);

}
```

В этом коде мы используем указатель **arr** для доступа к элементам массива, не используя операцию индексации **[]**. Вместо этого мы используем арифметику указателей, чтобы получить адрес каждого элемента в памяти.

Конкретно, в каждой итерации цикла мы сначала получаем адрес текущего элемента массива, используя указатель **arr** и операцию **+** для добавления смещения **i** к адресу начала массива. Затем мы используем оператор разыменования ***** для получения значения, хранящегося по этому адресу.

Таким образом, строка **int value = *(arr + i);** выполняет ту же функцию, что и **int value = arr[i];**, но без использования операции индексации.

Этот подход может быть полезен в случаях, когда необходимо обойти массив или выполнить какие-то операции с каждым элементом, но использование операции индексации невозможно или неэффективно. Например, в некоторых алгоритмах сортировки или поиска может потребоваться доступ к элементам массива в произвольном порядке, а не по порядку индексов.

Арифметика указателей.

Арифметика указателей - это возможность выполнять математические операции над указателями в C++. Она используется для перемещения указателей по памяти, например, при обходе массивов, или для вычисления адресов внутри структур данных.

Основные операции, которые можно выполнять над указателями в C++, включают:

1. **+** и **-**: сложение и вычитание числа из указателя для получения нового адреса. Например, если **ptr** указывает на начало массива **arr**, то **ptr + 1** будет указывать на следующий элемент массива, а **ptr - 1** - на предыдущий.
2. **++** и **--**: инкремент и декремент указателя для перехода к следующему или предыдущему элементу. Например, **++ptr** увеличит адрес, на который указывает **ptr**, на размер элемента, на который он указывает.
3. **[]**: обращение к элементу массива через указатель. Например, если **ptr** указывает на начало массива **arr**, то **ptr[i]** будет обращаться к элементу массива с индексом **i**.
4. *****: разыменование указателя для получения значения, на которое он указывает. Например, если **ptr** указывает на адрес переменной **x**, то ***ptr** вернет значение этой переменной.

Арифметика указателей может быть очень полезной в C++, но она также может быть опасной, если не использовать ее правильно. Например, использование неправильных адресов или выход за пределы массива может привести к непредсказуемым результатам и ошибкам в работе программы. Поэтому необходимо быть внимательным и аккуратным при работе с указателями и арифметикой указателей в C++.

Выход за пределы массива (т.е. обращение к элементу массива с индексом, выходящим за пределы его допустимых значений) может привести к непредсказуемым результатам и ошибкам в работе программы в C++. Некоторые примеры таких ошибок:

1. Ошибка сегментации: это тип ошибки, который возникает при обращении к памяти, которой не существует или которая не может быть доступна. Если программа попытается обратиться к памяти, которая находится за пределами выделенного для массива блока памяти, это может привести к ошибке сегментации и аварийному завершению программы.
2. Неправильные значения: обращение к памяти, которая не относится к массиву, может привести к тому, что в переменные будут записываться неправильные значения, что может привести к непредсказуемому поведению программы.
3. Непредсказуемые результаты: если программа обращается к памяти за пределами массива, то результат может быть непредсказуемым. Например, если в массиве хранятся значения, используемые для вычисления, а программа обратится к несуществующему элементу массива, то результат вычислений будет неправильным и может привести к непредсказуемому поведению программы.

В целом, выход за пределы массива может привести к непредсказуемым последствиям в зависимости от контекста и кода программы, и может быть трудно определить, что именно привело к ошибке. Поэтому необходимо быть внимательным при работе с массивами и проверять, что индексы элементов массива находятся в допустимых пределах.