Технологии индустриального программирования

Направление подготовки	09.03.02 "Информационные системы и технологии"
Профиль	Фуллстек разработка
Курс, семестр	1-2, 1-3 семестры

Ссылка на репозиторий:

https://github.com/astafiev-rustam/industrial-programming-technologies/tree/practice-1-5

Практическое занятие №5: Массивы

Конечно! Вот подробный конспект по статическим массивам в C++, написанный в форме связного объяснения.

Если до этого мы работали с отдельными переменными (одно число, один символ), то массив позволяет нам хранить и обрабатывать целую группу однотипных данных под одним именем. Представьте себе, что вам нужно хранить оценки 30 студентов в группе. Создавать 30 переменных `int grade1, grade2, ..., grade30` было бы крайне неудобно. Массивы решают именно эту проблему.

Часть 1: Что такое статический массив?

Статический массив в C++ — это упорядоченная коллекция элементов одного и того же типа, размер которой фиксируется на этапе компиляции и не может быть изменен во время выполнения программы. Можно провести аналогию с рядом пронумерованных почтовых ящиков в подъезде. Все ящики одинакового размера (один тип данных), у каждого есть свой номер (индекс), и количество ящиков в ряду заранее известно и неизменно (статический размер).

Часть 2: Объявление и инициализация массива

2.1. Объявление массива

Чтобы объявить статический массив, нужно указать тип его элементов, имя массива и его размер в квадратных скобках.

```
тип_данных имя_массива[размер];
```

Например, создадим массив для хранения пяти целых чисел:

```
int numbers[5];
```

После этой команды в памяти компьютера будет выделен непрерывный блок, достаточный для хранения пяти целых чисел. Важно помнить, что на этом этапе, если мы не провели инициализацию, массив может содержать "мусор" — случайные значения, которые остались в выделенной памяти.

2.2. Инициализация массива

Инициализировать массив, то есть заполнить его начальными значениями, можно несколькими способами.

- **Полная инициализация при объявлении:** Мы можем перечислить все значения элементов в фигурных скобках.

  ```
  int numbers[5] = {10, 20, 30, 40, 50};
  ```

  В результате `numbers[0]` будет равен 10, `numbers[1]` — 20, и так далее.

- **Неполная инициализация:** Если значений в фигурных скобках меньше, чем размер массива, то остальные элементы автоматически заполняются нулями.

  ```
  int numbers[5] = {10, 20}; // numbers[0]=10, numbers[1]=20, numbers[2]=0, numbers[3]=0, numbers[4]=0
  ```

  Этот способ очень удобен для быстрого обнуления всего массива.

- **Инициализация без указания размеса:** Компилятор может сам подсчитать размер массива, если мы предоставим все начльные значения.

  ```
  int numbers[] = {1, 2, 3, 4, 5}; // Компилятор создаст массив размером 5
  ```

Часть 3: Работа с элементами массива. Концепция индекса.

Чтобы получить доступ к конкретному элементу массива, мы используем его имя и индекс. Индекс — это целое число, указывающее позицию элемента в массиве. Здесь кроется самый важный и часто упускаемый новичками момент: **нумерация элементов массива в C++ начинается с нуля**.

Это значит, что в массиве `int numbers[5] = {10, 20, 30, 40, 50};`:

- Первый элемент — это `numbers[0]`, и он равен 10.
- Второй элемент — это `numbers[1]`, и он равен 20.
- Пятый (последний) элемент — это `numbers[4]`, и он равен 50.

Попытка обратиться к несуществующему элементу, например, `numbers[5]` или `numbers[-1]`, приведет к неопределенному поведению — программа может аварийно завершиться, выдать мусорное значение или повредить другие данные. Это одна из самых распространенных ошибок, называемая "выходом за границы массива".

Мы можем читать значение элемента и записывать в него новое значение, используя оператор присваивания.

```
int numbers[5] = {10, 20, 30, 40, 50};

cout << numbers[0] << endl; // Выведет 10
numbers[0] = 100;           // Теперь первый элемент массива равен 100
cout << numbers[0] << endl; // Выведет 100

int firstElement = numbers[0]; // Скопировали значение элемента в переменную
```

Часть 4: Взаимосвязь массивов и циклов

Истинная сила массивов раскрывается при их использовании вместе с циклами. Циклы позволяют нам всего несколькими строками кода обработать все элементы массива, независимо от его размера.

Рассмотрим классический пример — заполнение массива и вывод его содержимого на экран с помощью цикла `for`.

```
#include <iostream>
using namespace std;

int main() {
    const int size = 5; // Хорошим тоном является использование константы для
размера
    int arr[size];

    // Заполнение массива значениями, введенными пользователем
    cout << "Введите " << size << " целых чисел:" << endl;
    for (int i = 0; i < size; i++) {
        cin >> arr[i]; // Для каждого i от 0 до 4 запрашиваем число
    }

    // Вывод содержимого массива на экран
    cout << "Содержимое массива: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Обратите внимание, как условие продолжения цикла `i < size` идеально соответствует диапазону допустимых индексов от `0` до `size - 1`. Использование константы `size` делает код гибким: чтобы изменить размер массива, нам нужно поправить значение всего в одном месте.

Часть 5: Практические примеры и алгоритмы

Давайте решим несколько классических задач для закрепления материала.

5.1. Поиск максимального элемента в массиве

Алгоритм прост: мы предполагаем, что первый элемент массива является максимальным, а затем последовательно сравниваем это предположение со всеми остальными элементами.

```
int numbers[] = {5, 12, -3, 25, 7};
int max = numbers[0]; // Предполагаем, что максимум — это первый элемент

for (int i = 1; i < 5; i++) { // Начинаем цикл со второго элемента (i=1)
    if (numbers[i] > max) {
        max = numbers[i]; // Если нашли элемент больше, обновляем максимум
    }
}

cout << "Максимальный элемент в массиве: " << max << endl; // Выведет 25
```

5.2. Вычисление суммы и среднего арифметического элементов

Это задача на накопление результата в переменной-аккумуляторе.

```
int values[] = {10, 20, 30, 40, 50};
int sum = 0; // Аккумулятор для суммы

for (int i = 0; i < 5; i++) {
    sum += values[i]; // На каждой итерации прибавляем текущий элемент к сумме
}

double average = static_cast<double>(sum) / 5; // Важно привести к double для
точного результата

cout << "Сумма элементов: " << sum << endl;     // Выведет 150
cout << "Среднее арифметическое: " << average << endl; // Выведет 30
```

5.3. Поиск элемента в массиве (линейный поиск)

Иногда нужно проверить, присутствует ли определенное значение в массиве.

```
int secretCode[] = {42, 15, 78, 91, 3};
int key = 78;
bool found = false; // Флаг, который сообщит, нашли ли мы элемент

for (int i = 0; i < 5; i++) {
    if (secretCode[i] == key) {
        found = true;
        cout << "Элемент " << key << " найден на позиции " << i << endl;
        break; // Элемент найден, дальнейший поиск не нужен
```

```
    }
}

if (!found) {
    cout << "Элемент " << key << " в массиве не найден." << endl;
}
```

Часть 6: Двумерные статические массивы

Массивы могут быть многомерными. Самый распространенный случай — двумерный массив, который можно представить как таблицу со строками и столбцами.

Объявление и инициализация двумерного массива:

```
тип_данных имя[количество_строк][количество_столбцов];
```

Например, создадим массив, представляющий собой поле для игры в "Крестики-нолики" 3x3.

```
char gameBoard[3][3] = {
    {'-', '-', '-'}, // Первая строка
    {'-', '-', '-'}, // Вторая строка
    {'-', '-', '-'}  // Третья строка
};
```

Для работы с двумерными массивами используются вложенные циклы. Внешний цикл обычно перебирает строки, а внутренний — столбцы.

```
// Вывод игрового поля на экран
for (int row = 0; row < 3; row++) {
    for (int col = 0; col < 3; col++) {
        cout << gameBoard[row][col] << " ";
    }
    cout << endl; // Переход на новую строку после вывода всех элементов строки
}
```

Результат:

```
- - -
- - -
- - -
```

Статические массивы — это основа для понимания более сложных структур данных. Их главное преимущество — скорость доступа к элементам и предсказуемость. Главный недостаток —

невозможность изменить размер после объявления. Но для огромного количества задач, где размер данных известен заранее, статические массивы являются идеальным инструментом.

Задачи и самостоятельная работа

1. Один русскоязычный рэпер, который учился в Лодноне, выбирает, кто выиграет в розыгрыше автографсессию. В том числе, решающим фактором был выбран возраст, `X`. Если `X>16`, то участник автоматически проигрывает. Пусть задано `N` - количество участников розыгрыша, далее через пробел вводятся `N` чисел - возрастов. Необходимо вывести те `X`, которые не проходят по правилу `X>16`.

2. Пациентам специального заведения по борьбе с игровой зависимостью бывает крайне нелегко, поэтому, чтобы было хоть что-то привычное для них, проворачивают следующее в столовой:

   - у каждого пациента есть личный номер для удобства работы (распределение номеров случайное - **деп**);
   - в столовую всегда очередь и она всегда состоит из N человек (формируется случайным образом - **додеп**);
   - сотрудница столовой Зинаида ходит в халате, на котором 1000 пуговиц и расстегивает его при работе (на случайное количество пуговиц - **последний додеп**);
   - если на Зинаиде расстёгнуто K пуговиц, то вся очередь смещается на K человек циклическим сдвигом (так как всё из-за случайных факторов - **супер мега последний додеп**).

   Например, если сейчас очередь: `2 3 4 1 5`, а количество пуговиц: `11`, то очередь превращается в `5 2 3 4 1`.

   Необходимо написать программу, которая по `N`, порядку в очереди и `K` найдет текущее состояние очереди. Нельзя использовать дополнительные массивы и наборы данных (сложность `O(1)` по памяти).

3. Кирилл очень любит грибы. У него есть друг Александр, который периодически присылает ему координаты грибов (ячейки сетки на карте по вертикали и горизонтали), но есть проблема: У Кирилла 2 по географии.

   Необходимо помочь Кириллу и составить таблицу по точкам, в которой местоположение грибов обозначается *, а остальные ячейки содержат количество грибов вокруг данной точки (как в сапёре).

   Формат ввода

   ```
   3 2
   2
   1 1
   2 2
   ```

   Формат вывода

```
* 2
2 *
1 1
```

Примечания Даны числа N и M (целые, положительные, не превышают 32) – количество строк и столбцов в поле карты соответственно, далее число W (целое, неотрицательное, не больше 1000) – количество грибов на поле, далее следует W пар чисел, координаты грибов на поле (первое число – строка, второе число – столбец).

4. Сон Ги Хун играет против толпы грудных младенцев, которые лежат на платформе в клетках квадрата размером NxN, при этом количество детей в клетках, лежащих под побочной диагональю - 2, на побочной диагонали лежат по 1 ребёнку, а над главной диагональю не осталось никого. Необходимо вывести карту расположения количества детей по клеткам при заданном N и их количество (в новой строке).

   Например, при N = 3:

   ```
   001
   012
   122
   9
   ```

5. Вы находитесь в помещении с большим количеством человек (для простоты, помещение - прямоугольник, в котором по высоте и ширине помещается по 1 человеку, по длине - все имеющиеся). Неожиданно на лбу у каждого человека появилось число (невыдуманная история, например Misfits S04E06).

   Так вышло, что все числа встречаются по два раза, кроме одного. Необходимо найти это число за `O(n)` по времени и за `O(1)` по памяти.

   Например:

   ```
   15
   1 2 3 2 3 4 6 7 8 9 7 6 4 9 1
   ```

   Ответ:

   ```
   8
   ```