Многомерные массивы. Строки. Файлы

Многомерные массивы

Многомерные массивы в C++ представляют собой одномерные, записанные построчно. На практике обычно используются двумерные массивы. Синтаксис:

```
тип имя[размер1][размер2]={инициализаторы};
```

Пример объявления обычного массива:

```
int a[3][4]={1, -3,  5, 0,
             6, -2, -4, 1,
             2,  7,  9, 8};
```

В функцию многомерные массивы передаются как одномерные. Если массив имеет размеры N x M, то к элементу a[i][j] массива внутри функции следует обращаться как a[i*M+j].

Динамические многомерные массивы

Динамические двумерные массивы часто удобнее обычных. При работе с динамическими двумерными массивами необходимо правильно выделять память под массив перед началом работы с ним и правильно ее очищать после завершения работы.

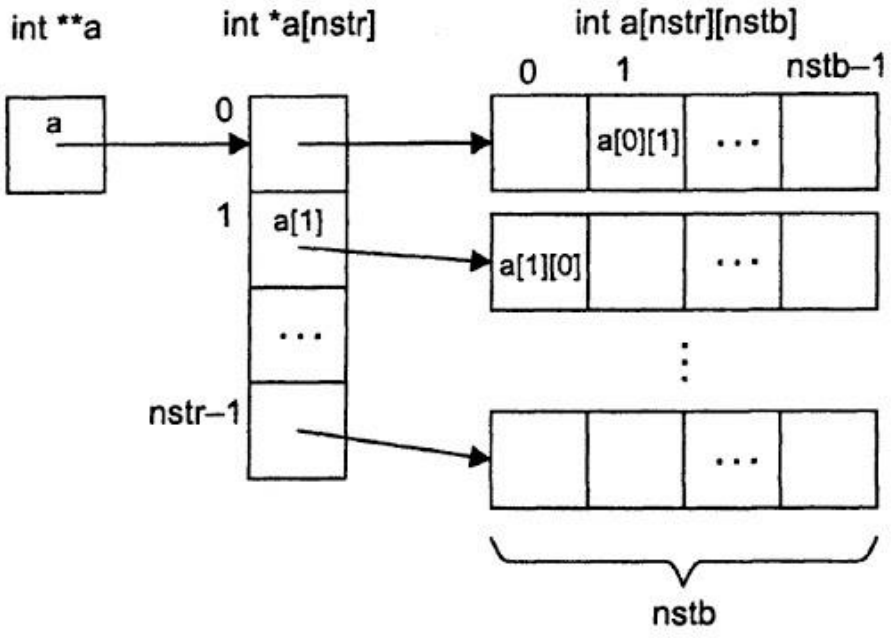
Синтаксис выделения памяти:

```
тип **имя = new тип*[кол_во_строк];
for (int i=0; i<кол_во_строк; i++)
     имя[i]=new тип[кол_во_столбцов];
```

Синтаксис очистки памяти:

```
for (int i=0; i<кол_во_строк; i++)
     delete []имя[i];
delete []имя;
```

В функцию обязательно передаются массив через двойной указатель и обе размерности. Тогда внутри функции массив можно использовать так же, как и в функции, в которой он был объявлен.



Выделение памяти под двумерный массив

Пример: см. выделение памяти и очистку памяти 5_1_2D_Arrays.cpp

Алгоритмы работы с многомерными массивами

Классификация:

1. Алгоритмы поэлементного обхода.
2. Работающие со строками или столбцами.
3. Матричные алгоритмы.
4. Алгоритмы на графах.
5. Прочие алгоритмы.

Пример: см. задачи 1, 2, 3 5_1_2D_Arrays.cpp

Возможности работы со строками в C++

В C++ существуют два вида строк:

1. Строки C (строки старого стиля).
2. Строки C++ (строки нового стиля).

Строки C

Строка C — это массив символов, завершающийся специальным нулевым символом '\0' (нуль-символом). Для подключения функций для работы со строками C необходимо добавить в программу

```
#include <cstring>
```

Длина строки определяется по положению нуль-символа. Для этого используется функция strlen().

При выделении памяти для хранения строки необходимо учитывать место для нуль-символа, т.е. если планируется строка длины n, то необходимо выделить n+1 байт. Если строка инициализируется при объявлении, размер можно не указывать.

Пример: см. пример 1 5_2_Cstring.cpp

Функции для работы со строками

Список функций можно посмотреть:

- в учебнике Павловской, Приложение 6-7, стр. 409 и далее.
- в интерактивном справочнике cplusplus.com/reference/cstring/ (можно включить автоматический перевод в браузере).

Например, strcmp используется для сравнения строк. Строки сравниваются в алфавитном порядке. Если первая строка стоит раньше второй, то strcmp возвратит отрицательное значение, если больше — положительное. Если строки совпадают, strcmp возвращает 0.

Пример: см. пример 2 5_2_Cstring.cpp

Строки C++

Строка C++ — это класс string. Классы будут изучаться далее, однако строками C++ можно пользоваться и без их подробного рассмотрения, считая string типом данных.

Для подключения функций для работы со строками C++ необходимо добавить в программу

```
#inclide <string>
```

Работа со строками C++ намного проще, чем со строками C. Например, для сравнения строк C++ можно использовать обычные операции сравнения. Склеить строки можно при помощи обычной операции

сложения. Полный список функций можно посмотреть в справочнике http://cplusplus.com/reference/string/.

Пример: см. 5_3_String_part1.cpp

Особенности чтения строк с клавиатуры

Если читать значения в строку с помощью cin, то чтение будет производиться до первого разделителя (пробел, табуляция, конец строки). Для чтения строки с пробелами и символами табуляции используется getline. Если getline был вызван после использования cin, то в потоке ввода может остаться "лишний" конец строки. Для избежания проблем необходимо использовать cin.ignore().

Пример: см. 5_4_String_part2.cpp

Управление вводом-выводом

Для управления потоками ввода и вывода можно использовать различные манипуляторы. Один из них уже знакомый нам endl, помещающий в поток символ перехода на новую строку. Для работы манипуляторов необходимо подключить

```
#include <iomanip>
```

Некоторые манипуляторы:

- setfill — заполнитель (по умолчанию заполнитель пробел).
- setw — указывает количество символов, которое должна занимать печать следующего числового значения. Если число содержит меньше символов, печатается заполнитель.
- setprecision — указывает количество знаков после точки, которые будут выводиться на экран. Если число короче, то нули не выводятся.
- fixed — принудительно заставляет выводить ровно столько знаков числа, сколько указано, даже если это нули.

Пример: 1_3_IOmanip.cpp

Работа с файлами

В C++ есть два способа работы с файлами, способ C и способ C++. Мы рассматриваем только способ C++, а именно работу с файлами с использованием потоков. Подробности работы с потоками будут рассмотрены в одной из следующих лекций.

Для работы с файлами необходимо подключить

```
#include <fstream>
```

В отличие от стандартных потоков, файловые потоки перед использованием нужно создать или открыть (при создании поток открывается автоматически). Файловые потоки можно открывать:

- на чтение с помощью команды ifstream
- на запись с помощью команды ofstream

Для файловых потоков можно использовать те же опции, что и для стандартных (endl, манипуляторы, <<, >> и т.д.). При работе с getline следует выполнять те же действия, что и при использовании стандартного ввода-вывода.

.eof() — функция, возвращающая true или false в зависимости от того, достигнут конец файла или нет.

Пример (не работает онлайн, нужно скопировать в среду разработки): 5_5_Files.cpp