//Как передать в функцию статический двумерный массив в C++ ?

//Ответ :

//Если бы массив был динамическим, то в функцию достаточно было бы передать указатель на указатель.Примерно так :

//#include <iostream>

//#include <cstdlib>

//#include <ctime>

//using namespace std;

//

//void arrprint(int\*\* arr, int m, int n) {

// for (int i = 0; i < m; i++) {

// for (int j = 0; j < n; j++) {

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

// }

// cout << endl;

// }

//}

//

//int main() {

// int m = 3, n = 5;

// int\*\* p = new int\*[m];

// for (int i = 0; i < m; i++) {

// p[i] = new int[n];

// for (int j = 0; j < n; j++) {

// p[i][j] = rand() % 10;

// }

// }

//

// arrprint(p, m, n);

//

// for (int i = 0; i < m; i++) delete[] p[i];

// delete[] p;

//}

//Обратите внимание на адреса, они будут примерно такими :

//1 0x760ff8 7 0x760ffc 4 0x761000 0 0x761004 9 0x761008

//4 0x761018 8 0x76101c 8 0x761020 2 0x761024 4 0x761028

//5 0x761038 5 0x76103c 1 0x761040 7 0x761044 1 0x761048

//Из адресов видно, что каждый одномерный динамический массив(являющийся условной «строкой» двумерного) располагаться в памяти может где угодно.Хотя элементы внутри каждой строки лежат последовательно друг за другом(через 4 байта для типа int).

//Создадим теперь двумерный статический массив и посмотрим на его адреса :

//#include <iostream>

//#include <cstdlib>

//#include <ctime>

//using namespace std;

//

//int main() {

// const int m = 3, n = 5;

// int a[m][n];

// for (int i = 0; i < m; i++) {

// for (int j = 0; j < n; j++) {

// a[i][j] = rand() % 10;

// cout << a[i][j] << ' ' << &a[i][j] << '\t';

// }

// cout << endl;

// }

//}

//Адреса будут такими :

//1 0x22fed4 7 0x22fed8 4 0x22fedc 0 0x22fee0 9 0x22fee4

//4 0x22fee8 8 0x22feec 8 0x22fef0 2 0x22fef4 4 0x22fef8

//5 0x22fefc 5 0x22ff00 1 0x22ff04 7 0x22ff08 1 0x22ff0c

//Из адресов видно, что «строки» статического двумерного массива в памяти располагаются строго друг за другом.То есть, по сути, двумерный массив размерности m на n в памяти размещается ровно также, как и одномерный массив размерности m\*n.

//А из этого, в частности, следует, что работать со статическим двумерным массивом мы можем через указатель первого уровня(в данном случае, через int\*).

//Значит передать в функцию двумерный статический массив можно, например, так:

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void arrprint(int\* arr, int m, int n) {

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

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

cout << \*(arr + i\*n + j) << ' ';

}

cout << endl;

}

}

int main() {

int a[3][6] = {

{ 3,7,8,0,1,4 },

{ 2,9,1,4,7,6 },

{ 5,2,7,6,1,9 }

};

arrprint(&a[0][0], 3, 6);

}

//Традиционное допущение «массив есть указатель» справедливо только для одномерного массива, да и то за счёт того, что при выполнении следующего кода происходит автоприведение :

//int a[10];

//int\* p = a;

//Но вот следующий пример уже является ошибкой :

//int a[7][3];

//int\*\* p = a;

//Попытавшись его выполнить, получим примерно такое сообщение от отладчика : hello.cpp : 8 : 12 : cannot convert int(\*)[3] to int\*\* in initialization.

//«Массив массивов» и «указатель на указатель» — это разные сущности.И разница между ними видна, например, на таком коде :

//#include <iostream>

//#include <cstdlib>

//#include <ctime>

//using namespace std;

//

//int main() {

// int a[3][6] = {

// { 3,7,8,0,1,4 },

// { 2,9,1,4,7,6 },

// { 5,2,7,6,1,9 }

// };

// int\* p = &a[0][0];

// int\*\* q = &p;

// cout << a[0][0] << endl; // 3

// cout << q[0][0] << endl; // 3

// cout << a[0] << endl; // 0x22fec0

// cout << q[0] << endl; // 0x22fec0

// cout << a[0][2] << endl; // 8

// cout << q[0][2] << endl; // 8

// cout << a[1] << endl; // 0x22fec0

// cout << q[1] << endl; // 0x22fed8

// cout << a[1][2] << endl; // 1

// cout << q[1][2] << endl; // 246351

//}

////Если в рамках задачи функция будет использоваться для обработки массивов с заведомо одинаковой длиной строк, то функцию возможно описать и потом вызывать вот так(пожертвовав её универсальностью, зато получив привычную возможность внутри функции работать не по смещению указателя, а по двум индексам) :

//

//#include <iostream>

//#include <cstdlib>

//#include <ctime>

// using namespace std;

//

//void arrprint(int arr[][6], int m) {

// for (int i = 0; i < m; i++) {

// for (int j = 0; j < 6; j++) {

// cout << arr[i][j] << ' ';

// }

// cout << endl;

// }

//}

//

//int main() {

// int a[3][6] = {

// { 3,7,8,0,1,4 },

// { 2,9,1,4,7,6 },

// { 5,2,7,6,1,9 } };

// arrprint(a, 3);}