**06.03.2024 года Динамические двумерные массивы (матрицы)**

**Частично динамические двумерные массивы. Указатель на указатель. Динамические двумерные массивы Передача двумерных динамических массивов в функцию. Примеры программ.**

**Частично динамические двумерные массивы**

Это массивы, количество строк у которых фиксируется в виде константы и не меняется, а количество элементов в строках может быть переменной (более того, оно не обязательно должно быть одинаковым). То есть можно сказать, что частично динамический двумерный массив **–** это статический массив динамических одномерных массивов. Для его создания используется **массив указателей** на строки массива.

// **Пример 1.** Составить функциюдля нахождения суммы элементов

// в **одномерном** массиве.

// Создать**/**вывести **частично динамический** двумерный массив целых чисел,

// используя **массив указателей** на строки. При этом количество строк задать

// как **const,** а строки создать как одномерные динамические массивы с разным

// количеством элементов, Создать массив сумм элементов каждой строки

// массива, используя составленную функцию.

#include<iostream>

#include <ctime>

using namespace std;

int summa(int\*, int);

int **main()**{

const int D = 20;

int n;

cout << " Enter n --- number rows: "; cin >> n ;

int\* ptr[D]; // массив указателей на строки

int m[D], sum[D];

srand(time(NULL));

// Выделение динамической памяти под строки матрицы

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

cout << " Enter number of elements in row "<<i<<"---";

cin >> m[i];

ptr[i] = new int[m[i]]; // выделение памяти под **i-ую** строку

for (int j = 0; j < m[i]; j++) // заполнение **i-ой** строки массива

ptr[i][j] = rand() % 25 - 9;

}

cout << " **Source array**: " << endl; // вывод массива

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

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

cout.width(5); cout << \*(ptr[i] + j);

}

cout << endl;

}

// Нахождение суммы элементов в каждой строке матрицы

for (int i = 0; i < n; i++)

sum[i] = summa(ptr[i], m[i]);

for (int i = 0; i < n; i++) // вывод массива сумм

cout << "sum["<<i<<"] = " << sum[i] << endl;

// Освобождение памяти, занимаемой строками массива

for (int i = 0; i < n; i++)

delete [] ptr[i];

}

int summa(int\* p, int n){

int sum = 0;

for (int i = 0; i < n; i++)

sum = sum + p[i];

return sum;

}

**Результат:**

Enter n --- number rows: **3**

Enter number of elements in row 0---3

Enter number of elements in row 1---4

Enter number of elements in row 2---5

Source array:

-9 13 -5

-8 15 -2 14

2 3 3 4 5

sum[0] = -1 sum[1] = 19 sum[2] = 17

**Указатель на указатель**

**Указатель на указатель** – это *переменная*, содержащая адрес другого *указателя*.

Для объявления указателя на указатель в программе, нужно просто удвоить число звёздочек в объявлении, например:

int \*\* ptr, \*qtr;

int x = 10;

qtr = &x; ptr = &qtr;

cout<<x<<" "<<\*qtr<<' '<<\*\*ptr<<endl; // **10 10 10**

Чтобы объявить в программе на **С++** *указатель на указатель на указатель и так далее* следует при объявлении перед указателем поставить нужное количество звездочек.

// **Пример 2**. Указатель на указатель.

#include <iostream>

using namespace std;

int **main()**{

int x(5), \*q, \*\*p; // **x(5) равносильно x = 5;**

q = &x; p = &q;

cout << x << ' ' << \*q << ' ' << \*\*p << endl; // **5 5 5**

int y(7); \*\*p = y;

cout << x << ' ' << \*q << ' ' << \*\*p << endl; // **7 7 7**

\*q = 9;

cout << x << ' ' << \*q << ' ' << \*\*p << endl; // **9 9 9**

int z = 3; q = &z;

cout << x << ' ' << \*q << ' ' << \*\*p << endl; // **9 3 3**

}

**Динамические двумерные массивы (матрицы)**

Основной способ работы с динамическими двумерными массивами базируется на использовании понятий **массив указателей и указатель на указатель**.

Пусть, например, требуется создать **динамический двумерный массив** целых чисел из **n** строк и **m** столбцов. Для этого сначала необходимо создать ***одномерный*** ***массив***из **n** ***указателей,*** и его адрес запомнить в переменной **ptr**, объявленной как ***указатель на указатель***. Далее необходимо ***в цикле*** выделить память под **m**элементов *каждой строки* массива.

**Доступ** к элементам двумерного динамического массива с индексами **i** и**j** возможен только с помощью *указателя на указатель* **ptr** следующими способами:

ptr[i][j] \*(ptr[i] + j) \*(\*(ptr + i) +j)

Соответственно их адреса:

&ptr[i][j] (ptr[i] + j) (\*(ptr + i) +j)

**Под обычный** **двумерный массив** из **n** строк и **m** столбцов при объявлении выделяется ***сплошной*** участок байтов памяти. Для такого же **динамического массива** выделенная память не представляет собой сплошной участок, так как выделяется с помощью нескольких *операций* **new**.

// **\*Пример 3**. Создание и удаление **двумерного** динамического массива,

// размерности **(n, m**) – **n** строк**, m** столбцов. **Вывод адресов**.

#include<iostream>

using namespace std;

int **main()**{

int n, m;

cout << "Enter n, m: "; cin >> n >> m; // ввод размеров массива **3 3**

int\*\* ptr; // **ptr** – указатель на указатель

ptr = new int\* [n]; // массив указателей на строки

for (int i = 0; i < n; i++) // память для каждой строки

ptr[i] = new int[m];

ptr[0][0] = 55; // **55**

// \*\*ptr = 55;

cout<<" ptr[0][0] = "<<ptr[0][0] <<" \*\*ptr = "<<\*\*ptr << '\n'; // **55 55**

cout<<" ptr = "<<ptr<< '\n'; // 0317D8**20**

cout << " &ptr[0] = " << &ptr[0] <<'\n'; // 0317D820

cout << " &ptr[1] = " << &ptr[1] << '\n'; // 0317D824

cout << " &ptr[2] = " << &ptr[2] << '\n'; // 0317D828

cout<<" \*ptr = "<<\*ptr<< '\n'; // 0317**D740**

cout << " ptr[0] = " << ptr[0] << '\n'; // 0317**D740**

cout << " &ptr[0][0] = " << &ptr[0][0] <<'\n'; // 0317**D740**

cout << " ptr[1] = " << ptr[1] << '\n'; // 0317**D3C0**

cout << " &ptr[1][0] = " << &ptr[1][0] << '\n'; // 0317**D3C0**

// или (ptr[1] +0) или (\*(ptr +1)+0)

cout << " ptr[2] = " << ptr[2] << '\n'; // 0317**D1C8**

cout << " &ptr[2][0] = " << &ptr[2][0] << '\n'; // 0317**D1C8**

for (int i = 0; i < n; i++)

delete [] ptr[i]; // удаление строк

delete [] ptr; // удаление массива указателей

}

// **Пример 4.** Создать/вывести динамический двумерный массив размерности **(m, n).**

// Память для массива выделить в функции **main().** В массиве найти индекс строки

// с максимальной суммой элементов в строке. Создать динамический массив

// сумм элементов в строках.

// Составить функцию нахождения суммы элементов в одномерном массиве,

// и использовать её для нахождения суммы элементов в строках массива **matr**

// передавая ей адреса строк массива.

#include<iostream>

using namespace std;

int **summa**(int\*, int);

int **main()**{

int n, m;

cout << "Enter n, m:"; cin >> n >> m;

int\* sum = new int[n]; // динамический массив сумм

srand(n + m);

int\*\* ptr = new int\* [n]; // создание массива

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

ptr[i] = new int[m];

for (int j = 0; j < m; j++)

\*(ptr[i] + j) = rand() % 25 - 9;

}

cout << " Source matrix: \n"; // вывод массива

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

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

cout.width(5); cout << ptr[i][j];

}

cout << endl;

}

sum[0] = summa(ptr[0], m); int ind = 0;

int maxSum = sum[0];

for (int i = 1; i < n; i++) {

sum[i] = **summa**(ptr[i], m);

if (sum[i] > maxSum) {maxSum = sum[i]; ind = i;}

}

cout << "Summa elements of rows: " << endl; // вывод сумм

for (int i = 0; i < n; i++)

cout << '\t' << sum[i] << endl;

cout << " maxSum = " <<maxSum<<" ind = "<<ind<<endl;

delete [] sum; // удаление массива сумм

for (int i = 0; i < n; i++)

delete [] ptr[i]; // удаление строк

delete [] ptr; // удаление массива указателей

}

int **summa**(int\* p, int m){ // функция **summa()**

int s = 0;

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

s = s + p[i];

return s;

}

**Результат:**

Enter n, m:3 4

Source matrix:

2 13 6 6

12 7 -7 10

-7 10 9 0

Summa elements of rows: 27 22 12

maxSum = 27 ind = 0

**Передача адреса двумерного динамического массива в функцию**

// **Пример 5.** Память для динамического двумерного массива (матрицы)

// размерности **(m, n)** выделить в **main().** Создать/вывести массив, выполнить

// перестановку **0-**ой **и 1-**ой строк массива, освободить память.

// Составить и использовать для указанных действий функции **init(), print(),**

**// obmen(), delMemory(),** передавая им адрес массива.

#include<iostream>

using namespace std;

void **init**(int\*\*, int, int); // прототип **init()**

void **print**(int\*\*, int, int); // прототип **print()**

void **obmen**(int\*\*, int, int); // прототип **obnen()**

void **delMemory**(int\*\*, int); // прототип **delMemory()**

int **main()**{

int m, n;

cout << "Enter n -- number rom: "; cin >> n;

cout << "Enter m -- number elements in row: "; cin >> m;

int\*\* ptr = new int\* [n]; // выделение памяти

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

ptr[i] = new int[m];

}

**init**(ptr, n, m); **//** вызов **init()**

cout << " Source matrix: " << endl;

**print**(ptr, n, m); **//** вызов **print()**

**obmen**(ptr, n, m); **//** вызов **obmen()**

cout << " Matrix-rezult: " << endl;

**print**(ptr, n, m); **//** вызов **print()**

**delMemory**(ptr, n); **//** вызов **delMemory()**

}

void **init**(int\*\* p, int n, int m) { // функция **init()**

srand(n + m);

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

p[i][j] = rand() % 25 - 10;

}

void **print**(int\*\* p, int n, int m) { // функция **print()**

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

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

cout.width(5); cout << p[i][j];

}

cout << endl;

}

}

void **obmen**(int\*\* p, int n, int m) { // функция **obmen()**

int\* q = p[0]; // перестановка строк

p[0] = p[1]; p[1] = q;

}

void **delMemory**(int\*\* p, int n) { // функция **delMemory()**

for (int i = 0; i < n; i++) // удаление строк

delete [] p[i];

delete [] p; // удаление массива указателей

}

**Результат:**

Enter n -- number rom: **3**

Enter m -- number elements in row: **4**

Source matrix:

1 12 5 5

11 6 -8 9

-8 9 8 -1

Matrix-rezult:

11 6 -8 9

1 12 5 5

-8 9 8 -1

// **Пример** **6**. Память для динамического двумерного массива (матрицы)

// размерности **(m, n)** выделить в функции.

// Создать и использовать функции **memory(), init(), print(), delMemory().**

// Для выбора функций использовать **меню**.

// Иметь возможность при выполнении программы изменить **n, m**.

#include <iostream>

using namespace std;

int\*\* memory(int, int); // прототип **memory()**

void init(int\*\*, int, int); // прототип **init()**

void print(int\*\*, int, int); // прототип **print()**

void delMemory(int\*\*, int); // прототип **delMemory()**

int **main()** {

int n, m, select, flag = 1;

iint \*\* ptr = 0;

while (flag){

cout << "Enter the label number:\n";

cout <<"memory-------->1\n"

<< "init---------->2\n"

<< "print--------->3\n"

<< "delMemory----->4\n"

<< "end programm-->5\n";

cin >> select;

switch (select) {

case 1: cout << "Enter n, m: "; cin >> n>>m;

ptr = memory(n, m); break;

case 2: init(ptr, n, m); break;

case 3: print(ptr, n, m); break;

case 4: delMemory(ptr, n); break;

case 5: flag = 0; break;

default: cout<<"Error\n";

}

}

cout << "End programm" << endl;

}

int\*\* **memory**(int n, int m){ // функция **memory()**

int\*\* p = new int\* [n];

for (int i = 0; i < n; i++)

p[i] = new int[m];

return p;

}

void **init**(int\*\* p, int n, int m) { // функция **init()**

srand(n + m);

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

p[i][j] = rand() % 25 - 10;

}

void **print(**int\*\* p, int n, int m) { // функция **print()**

cout << " Matrix: " << endl;

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

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

cout.width(5); cout << p[i][j];

}

cout << endl;

}

}

void **delMemory**(int\*\* p, int n) { // функция **delMemory()**

for (int i = 0; i < n; i++)

delete [] p[i];

delete [] p;

}

**Не забывать** освобождать (вызов функции **delMmemory())** перед изменением **n, m** и перед выходом из программы.

**Результат:**

Enter the label number:

memory-------->1

init------------->2 delMemory----->4

print----------->3 end programm-->5

**1**

Enter n, m: **2 3**

Enter the label number:

memory-------->1 delMemory----->4

init------------->2 end programm-->5

print----------->3

**2**

Enter the label number:

memory-------->1 delMemory----->4

init------------->2 end programm-->5

print----------->3

**3**

**Matrix**:

-6 8 -5

14 0 -5

Enter the label number:

memory-------->1 delMemory----->4

init------------->2 end programm-->5

print----------->3

**4**

Enter the label number:

memory-------->1 delMemory----->4

init------------->2 end programm-->5

print----------->3

**5** End programm

**Д. З. на 13 03.2024**

**Постановка задания.**

**!!!** В программах использовать **только** учебный материал .

!!! В тексте программы **обязательно** должны быть **условие задачи** и **тестовый пример**, использованный при отладке программы.

**Ограничения:**

* **не** создавать для хранения цифр числа **массив;**
* **не** задавать числа как **симводьный массив**;
* **не** использовать объекты класса **string**;
* **не** использовать стандартные функции.

**Требования к программе:**

* массивы должны быть созданы в **динамической области памяти**;
* в программе обязательно д.б.  **функция**, указанная в условии задачи;
* для задач со звёздочками **составить и использовать** и другие функции --- создание/удаление, ввод/вывод массива, обработки всего массива, строк массива, элементов массива (какие и сколько ---- определяетпрограммист**)**;
* использовать **меню** (для задач без звёздочки **по желанию**) для выбора функций при выполнении программы;
* для перестановки, удаления, вставки строк использовать их адреса. Не использовать дополнительный массив.
* результат выводить в доступном и понятном виде.

**!!!** Если в условии задачи сказано **Удалить строки**…**Оставшиеся отсортировать**…, то надо удалить строки (**в том же массиве**), вывести массив, а затем сортировать оставшиеся строки. Можно использовать **две** функции или **одну** ---- по желанию.

**Задачи**:

1. Составить функцию для определения --- элементы одномерного массива расположены по возрастанию их значений или нет.

В двумерном массиве размера **(n, m)** найти максимальный элемент (**первый**) среди строк массива, элементы которых **не** **расположены** по возрастанию их значений, и заменить его (найденный максимальный элемент) на сумму его цифр. Использовать составленную функцию для строк данного двумерного массива. **Указатели на строки.**

**исходный массив** **массив-результат**

13 34 22 52 43 13 34 22 52 43

20 21 22 95 96 20 21 22 95 96

52 41 **91** 72 9152 41 **10** 72 91

11 13 22 27 37 11 13 22 27 37

( максимальный элемент **91)**

1. \*Составить функцию для нахождения в одномерном массиве количества элементов, цифры которых расположены в порядке убывания (например, **54321**).

В двумерном массиве размера **(n, m) строки-соседи** (строки с индексами 0 и 1, 2 и 3, 4 и 5), содержащие одинаковое количество элементов, цифры которых расположены в порядке убывания, поменять местами и в каждой строке удалить первый такой элемент.

**исходный массив** **массив-результат**

123 234 345 631 567 678 123 234 345 631 567 678

432 456 654 678 345 567 432 456 654 678 345 567

456 531 976 34 321 765 432 456 654 876 345

321 432 456 654 876 345 456 976 34 321 765

12 13 14 15 654 764 13 14 15 16 876

765 13 14 15 16 876 12 13 14 15 764

1. \*\*В двумерном массиве **matr(n, m)** после каждой строки, количество максимальных элементов (поиск за один просмотр) у которых чётное, вставить нулевую строку (все элементы равны нулю). Дополнительный массив не использовать**. Указатели на строки**.
2. \*\*\*Составить функцию для нахождения в одномерном массиве количествагрупп стоящих подряд элементов, цифры которых расположены в порядке убывания. Например, в массиве **456 531 87 34 321 986 23** две группы

В двумерном массиве размера **(n, m) удалить** строки (не создавать дополнительный массив), не содержащие таких групп элементов. Оставшиеся строки расположить в порядке возрастания количества таких групп, используя эффективную сортировку методом "**пузырька**" (т.е. без лишних просмотров элементов массива). Можно создать дополнительный одномерный массив количеств таких групп в строках. Использовать составленную функцию для строк заданного двумерного массива. **Указатели на строки**.

**исходный массив** **массив-результат**

321 234 345 456 765 678 789 321 12 13 14 15 654 764 18 345

321 432 456 654 876 345 541 543 456 531 876 34 321 986 23 234

345 567 876 12 13 543 123 432 321 432 456 654 876 345 541 543

12 13 14 15 654 764 18 345

456 531 876 34 321 986 23 234

!!!Не забывать удалять память для i-строки**!!!**

**delete [] p[i];**  // освобождение памяти **i-ой** строки

for (int k = i; k < n - 1; k++) // затираем строку

p[k] = p[k + 1];