**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По лабораторной работе №10 на тему

«Динамические массивы»

Вариант №11

Выполнил студент группы РИС-20-1б

Шумилов Лев Сергеевич

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –научиться работать с динамическими массивами, организация динамических массивов.

**Постановка задачи**

Задача – реализовать на языке C++ программу, которая:

1. Создаёт динамический массив
2. Выполняет обработку динамического массива
3. Реализует вывод элементов массива на экран

Исходные данные для **варианта №11**:

Задачи:

1. Сформировать двумерный массив;
2. Добавить в него строку и столбец с заданным номером (индексом)

**Анализ задачи**

Для решения задачи были использованы следующие средства:

1. Язык программирования C++ (Microsoft Visual C++)
2. Текстовый редактор Microsoft Visual Studio Code
3. Система контроля версий Git

В математике двумерные массивы чисел представляются в виде таблицы и называют матрицей (англ. *Matrix*). Вывод чисел также будет организован в виде таблицы.

Для того, чтобы работал ввод-вывод в программе, необходимо подключить заголовочный файл iostream и воспользоваться пространством имён std.

#include <iostream>

using namespace std;

Поскольку будет происходить работа с динамическими массивами, будут использоваться соответствующие типы данных, которые объявляются в куче при помощи ключевого слова new. Для того, чтобы объявить одномерный массив чисел, используется тип int\*, который указывает на область памяти, в который будет записан массив чисел типа int. По данной логике, для того чтобы объявить двумерный массив, необходимо использовать тип данных int\*\*, который указывает на массив данных типа int\*, который указывает на массив чисел; получается **массив массивов, что, по смыслу, является двумерным массивом**.

Для удобства, была реализована функция, которая автоматически инициализирует новый двумерный массив, заполняя его случайными числами. Для того, чтобы заполнить массив случайными числами, используется функция rand(), которая изначально присутствует в пространстве имён std, который входит в библиотеку STL языка C++.

int\*\* NewMatrix(int rows, int columns)

{

int\*\* matrix = new int\*[rows];

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

{

matrix[i] = new int[columns];

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

matrix[i][j] = rand();

}

\_rowCount = rows;

\_columnCount = columns;

return matrix;

}

Также, для удобства, были объявлены две глобальные переменные типа int, которые хранят размерность данной матрицы, а именно:

* int \_rowCount; – количество строк;
* int \_columnCount; – количество столбцов;

Для того, чтобы вывести все элементы матрицы на экран, реализована функция PrintMatrix, которая структурированно выводит элементы матрицы благодаря служебному символу \t. Функция просто проходит по всем строкам и столбцам, извлекает значение каждого элемента и сразу же выводит его на экран.

void PrintMatrix(int\*\* matrix)

{

for (int i = 0; i < \_rowCount; i++)

{

for (int j = 0; j < \_columnCount; j++)

cout << matrix[i][j] << '\t';

cout << endl;

}

}

Ещё по заданию необходимо добавить строку и столбец. Для упрощения задачи немного условимся: новая строка будет содержать только единицы (1), а новый столбец только двойки (2), сначала будет добавляться новая строка, затем новый столбец; поскольку строка и столбец в одной клетке пересекаются, то её значения будет зависеть от того, в каком порядке будет добавлены строка и столбец: поскольку столбец заполняется последним, то на пересекающейся клетке можно будет увидеть двойку (2).

Функция, добавляющая новую строку, работает в следующем порядке:

1. Начиная с первого элемента и по номер строки, в который собираемся вставлять новую строку (не включая сам индекс новой строки), переносятся элементы из старой матрицы в новую без изменений;
2. Добавляется новая строка в новую матрицу;
3. Переносятся оставшиеся старой матрицы в новую;
4. Значение старой матрицы переприсваивается на значение новой (с новой добавленной строкой;

Поскольку программа работает с динамическими массивами в куче, то следует в конце функции удалить образованный мусор (в данном случае, это новая матрица). Вдобавок, в быту и в математике номера строк и столбцов считают с единицы, а не с нуля: поэтому в начале выполнения функции значения аргумента уменьшается на единицу.

void AddNewRow(int\*\*& baseMatrix, int rowNumber)

{

rowNumber--;

\_rowCount++;

int\*\* newMatrix = new int\*[\_rowCount];

int i = 0;

while ((i != rowNumber) && (i < \_rowCount))

{

newMatrix[i] = new int[\_columnCount];

newMatrix[i] = baseMatrix[i];

i++;

}

newMatrix[i] = new int[\_columnCount];

for (int j = 0; j < \_columnCount; j++)

newMatrix[i][j] = 1;

for (i = rowNumber + 1; i < \_rowCount; i++)

{

newMatrix[i] = new int[\_columnCount];

newMatrix[i] = baseMatrix[i - 1];

}

baseMatrix = newMatrix;

delete[] newMatrix;

}

Функция, добавляющая новый столбец, работает точно по такому же принципу, что предыдущая, только вместо строк – столбцы.

void AddNewColumn(int\*\*& baseMatrix, int columnNumber)

{

columnNumber--;

\_columnCount++;

int\*\* newMatrix = new int\*[\_rowCount];

for (int i = 0; i < \_rowCount; i++)

{

newMatrix[i] = new int[\_columnCount];

int j = 0;

while ((j != columnNumber) && (j < \_columnCount))

{

newMatrix[i][j] = baseMatrix[i][j];

j++;

}

newMatrix[i][j] = 2;

for (j = columnNumber + 1; j < \_columnCount; j++)

newMatrix[i][j] = baseMatrix[i][j - 1];

}

baseMatrix = newMatrix;

delete[] newMatrix;

}

Для того, чтобы удобно вводить натуральные числа, была реализована функция ReadNaturalNum, которая возвращает число, которое было введено с клавиатуры (однако, если было введено отрицательное число, то функция попросит повторить ввод).

int ReadNaturalNum()

{

int x;

do cin >> x; while (x <= 0);

return x;

}

Все необходимые функции для работы с динамическими массивами реализованы, теперь можно приступать к реализации основного метода main().

Далее, основная программа должна выполнить следующие действия:

1. Принять на ввод два параметра:
   1. Количество строк;
   2. Количество столбцов;
2. Инициализировать новый динамический массив;
3. Вывести полученный массив на экран;
4. Принять на ввод номер строки, которую нужно добавить в массив;
5. Добавить новую строку в массив;
6. Принять на ввод номер столбца, который нужно добавить в массив;
7. Добавить новый столбец в массив;
8. Вывести обработанный массив;
9. Удалить массив (в конце выполнения программы память следует освобождать);

Ввод количество строк и столбцов:

cout << "Enter the row's count and column's count of new matrix:"

<< endl;

int rowCount = ReadNaturalNum(), columnCount = ReadNaturalNum();

Инициализация и вывод нового двумерного массива:

int\*\* matrix = NewMatrix(rowCount, columnCount);

PrintMatrix(matrix);

Ввод номера строки, добавление строки в массив и вывод обработанного массива:

cout << "Enter row's index where you'll add new row:" << endl;

int index = ReadNaturalNum();

AddNewRow(matrix, index);

PrintMatrix(matrix);

Ввод номера столбца, добавление столбца в массив и вывод обработанного массива:

cout << "Enter column's index where you'll add new column:" << endl;

index = ReadNaturalNum();

AddNewColumn(matrix, index);

PrintMatrix(matrix);

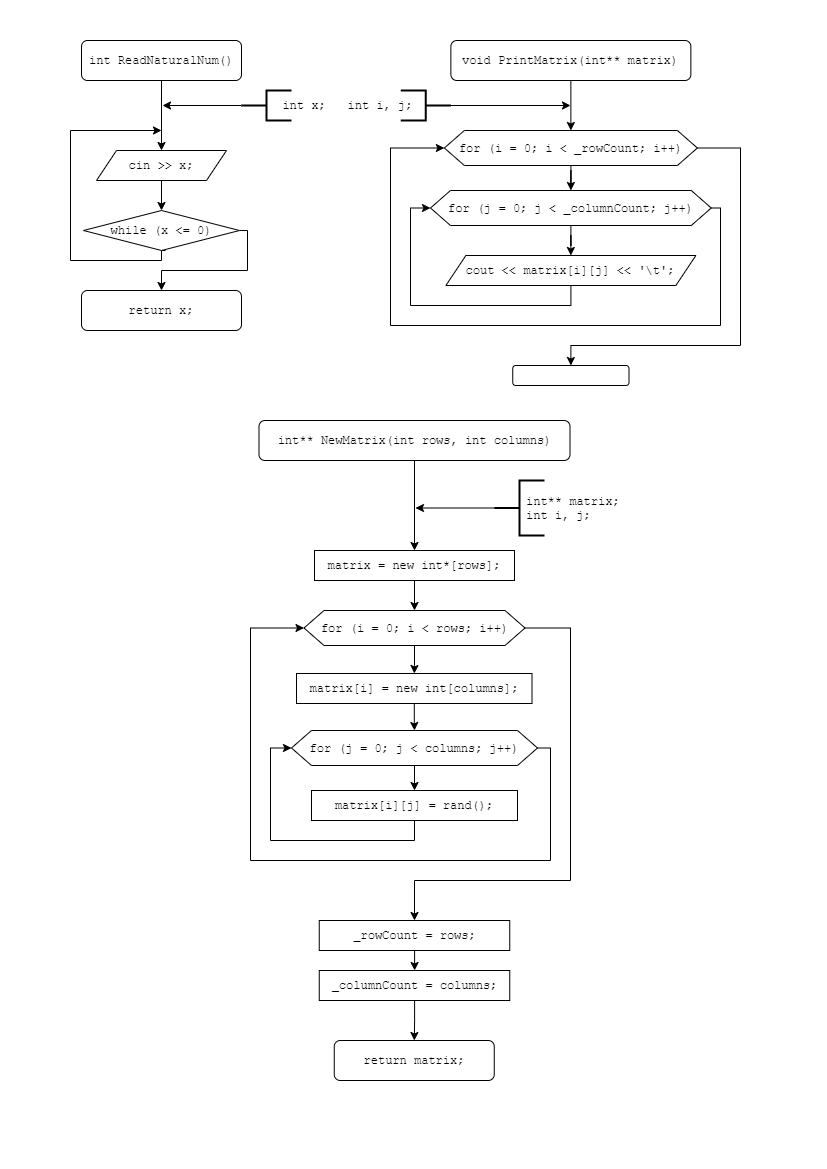
Удаление массива и завершение программы:

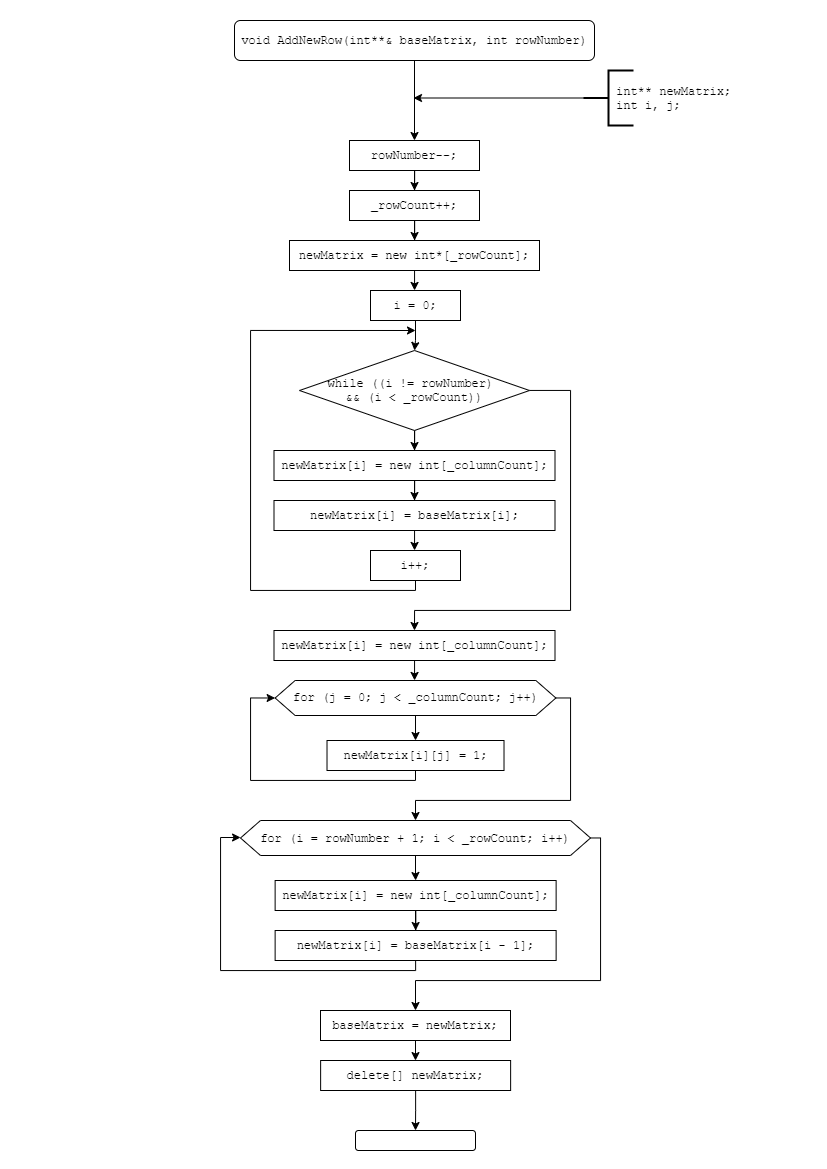
delete[] matrix;

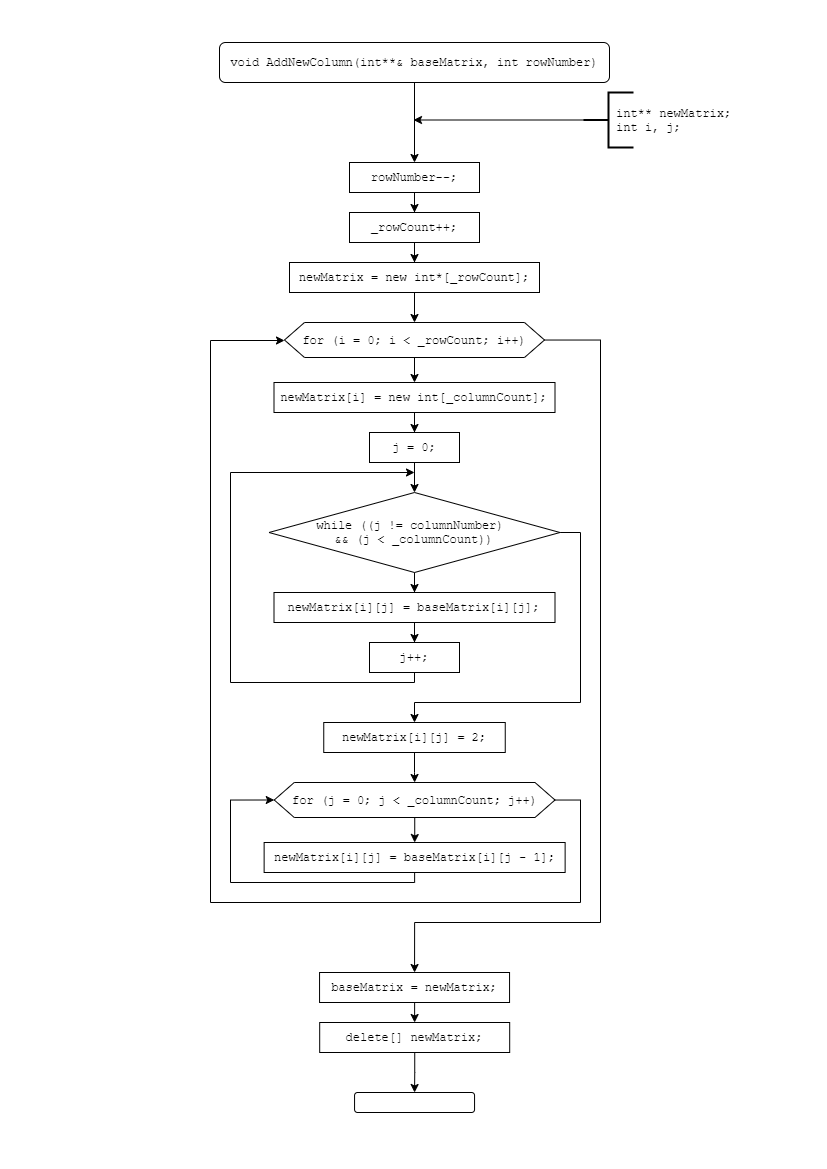
return 0;

Блок-схема решённой задачи:









Полный исходный код программы на языке C++:

#include <iostream>

using namespace std;

int

\_rowCount,

\_columnCount;

int ReadNaturalNum()

{

int x;

do cin >> x; while (x <= 0);

return x;

}

int\*\* NewMatrix(int rows, int columns)

{

int\*\* matrix = new int\*[rows];

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

{

matrix[i] = new int[columns];

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

matrix[i][j] = rand();

}

\_rowCount = rows;

\_columnCount = columns;

return matrix;

}

void PrintMatrix(int\*\* matrix)

{

for (int i = 0; i < \_rowCount; i++)

{

for (int j = 0; j < \_columnCount; j++)

cout << matrix[i][j] << '\t';

cout << endl;

}

}

void AddNewRow(int\*\*& baseMatrix, int rowNumber)

{

rowNumber--;

\_rowCount++;

int\*\* newMatrix = new int\*[\_rowCount];

int i = 0;

while ((i != rowNumber) && (i < \_rowCount))

{

newMatrix[i] = new int[\_columnCount];

newMatrix[i] = baseMatrix[i];

i++;

}

newMatrix[i] = new int[\_columnCount];

for (int j = 0; j < \_columnCount; j++)

newMatrix[i][j] = 1;

for (i = rowNumber + 1; i < \_rowCount; i++)

{

newMatrix[i] = new int[\_columnCount];

newMatrix[i] = baseMatrix[i - 1];

}

baseMatrix = newMatrix;

delete[] newMatrix;

}

void AddNewColumn(int\*\*& baseMatrix, int columnNumber)

{

columnNumber--;

\_columnCount++;

int\*\* newMatrix = new int\*[\_rowCount];

for (int i = 0; i < \_rowCount; i++)

{

newMatrix[i] = new int[\_columnCount];

int j = 0;

while ((j != columnNumber) && (j < \_columnCount))

{

newMatrix[i][j] = baseMatrix[i][j];

j++;

}

newMatrix[i][j] = 2;

for (j = columnNumber + 1; j < \_columnCount; j++)

newMatrix[i][j] = baseMatrix[i][j - 1];

}

baseMatrix = newMatrix;

delete[] newMatrix;

}

int main()

{

cout << "Enter the row's count and column's count of new matrix:"

<< endl;

int rowCount = ReadNaturalNum(), columnCount = ReadNaturalNum();

int\*\* matrix = NewMatrix(rowCount, columnCount);

PrintMatrix(matrix);

cout << "Enter row's index where you'll add new row:" << endl;

int index = ReadNaturalNum();

AddNewRow(matrix, index);

PrintMatrix(matrix);

cout << "Enter column's index where you'll add new column:"

<< endl;

index = ReadNaturalNum();

AddNewColumn(matrix, index);

PrintMatrix(matrix);

delete[] matrix;

return 0;

}

Скриншоты выполненной программы:

