Государственный Университет Молдовы
Факультет Математики и Информатики
Департамент Информатики

Лабораторная работа №4

по курсу “Основы программирования”
тема: Функции

Выполнил студент группы I2302:
Ciobanu
Stanislav.,
Проверил преподаватель:
G.Sturza

Кишинэу, 2023

Лабораторная работа №3

Задание

Вариант 3 : Заданы три матрицы различных размерностей. Написать программу, вычисляющую сумму положительных элементов каждой из матриц. Сумму элементов вычислять при помощи функции. Для вывода матриц использовать функцию

Программа

```
#include <iostream>

//Matrix as a struct

struct Matrix
{
public:
	int** _matrixValues;
	int* _sizeX = (int*)malloc(sizeof(int));
	int* _sizeY = (int*)malloc(sizeof(int));
};

Matrix* CreateMatrix()
{
	std::cout << " Enter size X of matrix\n ";
	int x;
	std::cin >> x;
	std::cout << "\n\n";

	std::cout << " Enter size Y of matrix\n ";
	int y;
	std::cin >> y;
	std::cout << "\n\n";

	Matrix* matrix = new Matrix();
	matrix->_matrixValues = (int**)calloc(y, sizeof(int*));

	*matrix->_sizeY = y;
	*matrix->_sizeX = x;

	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		matrix->_matrixValues[i] = (int*)calloc(*matrix->_sizeX, sizeof(int));
	}

	std::cout << "\n";
	return matrix;
}

Matrix* CreateMatrix(int x, int y)
{
	Matrix* matrix = new Matrix();
	matrix->_matrixValues = (int**)calloc(y, sizeof(int*));

	*matrix->_sizeY = y;
	*matrix->_sizeX = x;
```

```
	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		matrix->_matrixValues[i] = (int*)calloc(*matrix->_sizeX, sizeof(int));
	}

	std::cout << "\n";
	return matrix;
}

void SetMatrix(Matrix* matrix)
{
	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		for (int j = 0; j < *matrix->_sizeX; j++)
		{
			std::cout << " Enter matrix value (" << i + 1 << "," << j + 1 <<
")\n ";
			int inp;

			std::cin >> inp;

			matrix->_matrixValues[i][j] = inp;
		}
	}
}

void PrintMatrixValues(Matrix* matrix)
{
	std::cout << " Matrix is\n";

	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		std::cout << " ";
		for (int j = 0; j < *matrix->_sizeX; j++)
		{
			std::cout << matrix->_matrixValues[i][j] << " ";
		}
		std::cout << "\n";
	}

	std::cout << "\n";
}

void CountSumOfPosEl(Matrix* matrix)
{
	std::cout << " ";
	int sum = 0;
	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		for (int j = 0; j < *matrix->_sizeX; j++)
		{
			if (matrix->_matrixValues[i][j] > 0)
			{
				sum += matrix->_matrixValues[i][j];
			}
		}
	}
	std::cout << "Sum of positive elements = " << sum << "\n\n";
}

void DeleteMatrixContent(Matrix* matrix)
{
	for (int i = 0; i < *matrix->_sizeY; i++)
	{
		free(matrix->_matrixValues[i]);
```

```
	}

	free(matrix->_matrixValues);
	delete(matrix->_sizeX);
	delete(matrix->_sizeY);
}

void DeleteMatrix(Matrix* matrix)
{
	/*matrix->DeleteMatrixContent();*/
	DeleteMatrixContent(matrix);
	free(matrix);
}

//Matrix as a dynamic array

int* CountSizeOfMatrix(int** matrix)
{
	int* size = (int*)calloc(2, sizeof(int));

	size[0] = _msize(matrix) / sizeof(int*);
	size[1] = _msize(matrix[0]) / sizeof(int);

	return size;
}

int** CreateSimpleMatrix()
{
	std::cout << " Enter size X of matrix\n ";
	int x;
	std::cin >> x;
	std::cout << "\n\n";

	std::cout << " Enter size Y of matrix\n ";
	int y;
	std::cin >> y;
	std::cout << "\n\n";

	int** matrix = (int**)calloc(y, sizeof(int*));

	for (int i = 0; i < y; i++)
	{
		matrix[i] = (int*)calloc(x, sizeof(int));
	}

	std::cout << "\n";
	return matrix;
}

void SetMatrix(int** matrix)
{
	int* size = CountSizeOfMatrix(matrix);
	for (int i = 0; i < size[0]; i++)
	{
		for (int j = 0; j < size[1]; j++)
		{
			std::cout << " Enter matrix value (" << i + 1 << "," << j + 1 <<
")\n ";
			int inp;

			std::cin >> inp;

			matrix[i][j] = inp;
		}
	}
```

```
}

void PrintMatrixValues(int** matrix)
{
	std::cout << " Matrix is\n";

	int* size = CountSizeOfMatrix(matrix);

	for (int i = 0; i < size[0]; i++)
	{
		std::cout << " ";
		for (int j = 0; j < size[1]; j++)
		{
			std::cout << matrix[i][j] << " ";
		}
		std::cout << "\n";
	}

	std::cout << "\n";
}

void CountSumOfPosEl(int** matrix)
{
	int* size = CountSizeOfMatrix(matrix);

	std::cout << " ";
	int sum = 0;
	for (int i = 0; i < size[0]; i++)
	{
		for (int j = 0; j < size[1]; j++)
		{
			if (matrix[i][j] > 0)
			{
				sum += matrix[i][j];
			}
		}
	}
	std::cout << "Sum of positive elements = " << sum << "\n\n";
}

void DeleteMatrix(int** matrix)
{
	int* size = CountSizeOfMatrix(matrix);

	for (int i = 0; i < size[0]; i++)
	{
		free(matrix[i]);
	}
	free(matrix);
}

int main()
{
	//Prefabricated matrixes
	std::cout << "\n Matrix is 2x2\n ------------\n";
	Matrix* matrix1 = CreateMatrix(2, 2);
	SetMatrix(matrix1);
	PrintMatrixValues(matrix1);
	CountSumOfPosEl(matrix1);

	std::cout << "\n Matrix is 3x1\n ------------\n";
	Matrix* matrix2 = CreateMatrix(1, 3);
	SetMatrix(matrix2);
	PrintMatrixValues(matrix2);
	CountSumOfPosEl(matrix2);
```

```
	std::cout << "\n Matrix is 1x5\n -------------\n";
	Matrix* matrix3 = CreateMatrix(5, 1);
	SetMatrix(matrix3);
	PrintMatrixValues(matrix3);
	CountSumOfPosEl(matrix3);

	std::cout << "\n Want to create your own matrix?\n ---------------------------
----\n";
	while (1)
	{
		/*Matrix* matrix = CreateMatrix();  // Create a struct matrix

		SetMatrix(matrix);
		PrintMatrixValues(matrix);
		CountSumOfPosEl(matrix);
		DeleteMatrix(matrix);*/

		int** matrix = CreateSimpleMatrix();   // Create an array matrix

		SetMatrix(matrix);
		PrintMatrixValues(matrix);
		CountSumOfPosEl(matrix);
		DeleteMatrix(matrix);
	}
}
```

Используемые библиотеки

<iostream > – стандартный заголовочный файл, дающий доступ к вводу и выводу.

Результат выполнения программы:

```
Sum of positive elements = 21

Enter size X of matrix
3


Enter size Y of matrix
2



Enter matrix value (1,1)
1
Enter matrix value (1,2)
2
Enter matrix value (1,3)
-2
Enter matrix value (2,1)
3
Enter matrix value (2,2)
4
Enter matrix value (2,3)
-1
Matrix is
1 2 -2
3 4 -1

Sum of positive elements = 10
```