Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа №1

Вариант 4.

Передача значения по ссылке.

Выполнил студент группы № M3111
Соловьев Михаил Александрович.

Санкт-Петербург
2023

В данном задании необходимо объявить в отдельном заголовочном файле и реализовать процедуры согласно варианту. Все процедуры должны быть написаны в двух вариантах – один вариант использует указатели, второй вариант – ссылки. Написать программу, проверяющую и демонстрирующую правильность работы
процедур
Реализуемые процедуры:
- отбрасывает от вещественного числа его дробную часть
- меняют знак переменной (вещественной, целой)
- сдвигает окружность на заданный вектор
- умножает матрицу 3x3 на вещественное число

Код:

functions.cpp

```
#include <iostream>
#include "functions.h"
#include <cmath>

void GetFractPartPointer(float *real) {
   (*real) = *real - std::floor(*real);
}

void GetFractPartReference(float &real) {
   real = real - std::floor(real);
}

void ReverseRealPointer(float *real) {
   (*real) *= -1;
}

void ReverseRealReference(float &real) {
   real *= -1;
}

void MoveSquarePointer(Rectangle *rect, Vector2D *to_move) {
   rect->position.x += to_move->x;
   rect->position.y += to_move->y;
}

void MoveSquareReference(Rectangle &rect, Vector2D &to_move) {
   rect.position.x += to_move.x;
   rect.position.y += to_move.y;
}

void ChangeRawsPointer(int **matrix, int rows, int cols, int r_1, int r_2) {
   r_1 -= 1;
   r_2 -= 1;

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

```
        if (i == r_1) {
            for (int j = 0; j < cols; j++) {
                std::swap(matrix[i][j], matrix[r_2][j]);
            }
        }
    }
}

void ChangeRawsReference(int (&matrix)[3][3], int r_1, int r_2) {
    r_1 -= 1;
    r_2 -= 1;

    for (int i = 0; i < 3; i++) {
        if (i == r_1) {
            for (int j = 0; j < 3; j++) {
                std::swap(matrix[i][j], matrix[r_2][j]);
            }
        }
    }
}
```

functions.h

```
#ifndef COURSE_C___FUNCTIONS_H
#define COURSE_C___FUNCTIONS_H

struct Vector2D {
    float x;
    float y;
};

struct Rectangle {
    Vector2D position;
    Vector2D scale;
};

void GetFractPartPointer(float *real);
void GetFractPartReference(float &real);

void ReverseRealPointer(float *real);
void ReverseRealReference(float &real);

void MoveSquarePointer(Rectangle *rect, Vector2D *to_move);
void MoveSquareReference(Rectangle &rect, Vector2D &to_move);

void ChangeRawsPointer(int **matrix, int rows, int cols, int r_1, int r_2);
void ChangeRawsReference(int (&matrix)[3][3], int r_1, int r_2);

#endif
```

main.cpp

```
#include <iostream>
#include "functions.h"

int main() {
    float real, real_1;
    float rev_real, rev_real_1;

    Rectangle rectangle = {{0.0f, 0.0f}, 1.0f};
    Rectangle res_rectangle = rectangle;
    Vector2D to_move = {1.0f, 0.0f};
    Vector2D res_to_move = to_move;

    int matrix[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int rows = 3;
    int cols = 3;

    int **r_matrix = new int *[rows];

    for (int i = 0; i < rows; i++) {
        r_matrix[i] = new int[cols];
    }

    r_matrix[0][0] = 1;
    r_matrix[0][1] = 2;
    r_matrix[0][2] = 3;
    r_matrix[1][0] = 4;
    r_matrix[1][1] = 5;
    r_matrix[1][2] = 6;
    r_matrix[2][0] = 7;
    r_matrix[2][1] = 8;
    r_matrix[2][2] = 9;

    std::cout << "Input your real number:" << "\n";
    std::cin >> real;

    real_1 = real;
    rev_real = real;
    rev_real_1 = real;

    GetFractPartPointer(&real);
    GetFractPartReference(real_1);
    std::cout << "Fractional part of number (pointer): " << real << "\n";
    std::cout << "Fractional part of number (reference): " << real_1 << "\n";

    ReverseRealPointer(&rev_real);
    ReverseRealReference(rev_real_1);
```

```
    std::cout << "Reverse real number (pointer): " << rev_real << "\n";
    std::cout << "Reverse real number (reference): " << rev_real_1 << "\n";

    std::cout << "Rectangle coordinates: (" << rectangle.position.x << ", " <<
                                        rectangle.position.y << ")\n";
    MoveSquarePointer(&rectangle, &to_move);
    std::cout << "Rectangle coordinates after moving by pointer: (" << rectangle.position.x <<
              ", " << rectangle.position.y << ")\n";
    MoveSquareReference(res_rectangle, to_move);
    std::cout << "Rectangle coordinates after moving by pointer: (" << res_rectangle.position.x <<
             ", " << res_rectangle.position.y << ")\n";

    ChangeRawsPointer(r_matrix, rows, cols, 2, 3);
    ChangeRawsReference(matrix, 2, 3);
    std::cout << "The Matrix \n";
    std::cout << "1 2 3 \n4 5 6 \n7 8 9 \n";
    std::cout << "----------" << "\n";
    std::cout << "Matrix after changing rows 2 and 3 (reference):" << "\n";

    for (auto & i : matrix) {
        for (int j : i) {
            std::cout << j << " ";
        }
        std::cout << "\n";
    }

    std::cout << "Matrix after changing rows 2 and 3 (pointer):" << "\n";

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            std::cout << r_matrix[i][j] << " ";
        }
        std::cout << "\n";
    }

    for (int i = 0; i < rows; i++) {
        delete[] r_matrix[i];
    }
    delete[] r_matrix;

    return 0;
}
```