Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Программирование на C++

Работа:

Лабораторная работа №1, Передача значения по ссылке

Вариант 5

Выполнил: Гаджиев С. И.,
M3115

Санкт-Петербург

2023 г.

ТЗ:
1 Объявите (в отельном заголовочном файле) и реализуйте (в другом файле) процедуры (они не возвращают значений!) согласно варианту.
2 Все процедуры должны быть написаны в двух вариантах – один вариант использует указатели, второй вариант – ссылки.
3 Напишите программу, проверяющую и демонстрирующую правильность работы процедур.

Задания:
№5 Отбрасывает от вещественного числа его целую часть.

№8 Изменяют вещественную переменную на обратное к ней число.

№13 Передвигает квадрат на заданный вектор.

№16 Меняет в матрице местами две указанные строчки.

File “function.cpp”:

1.

```
#include <iostream>
#include <cmath>
#include "func.h"
using namespace std;

// TASK #5
void part_pointer(double *x) {
    *x = fmod(*x, 1);
}

void part_link(double &x) {
    x = fmod(x, 1);
}
// end
```

Реализовал 2 процедуры: с использованием указателей и ссылок (part_pointer и part_link). В обеих процедурах реализовано отбрасывания от вещественного числа его целую часть, с одним лишь отличием в способе указания передаваемых аргументов: в 1-й функции сначала берём указатель, чтобы получить доступ к желаемому числу и только потом производим операцию отбрасывания целой части, во второй функции просто отбрасываем целое число. Для осуществления описанного выше действия воспользовались функцией “fmod”.

2.

```
// TASK #8
void change_pointer(double *x) {
    if (*x != 0) {
        *x = 1 / *x;
    }
}

void change_link(double &x) {
    if (x != 0) {
        x = 1 / x;
    }
}
// end
```

Реализовал 2 процедуры изменяющие вещественную переменную на обратную к ней число (change_pointer и change_link). Отличия те же, что и в первых двух процедурах.

3.

```
// TASK #13
Square::Square(double x1, double x2, double x3, double x4, double y1, double
y2, double y3, double y4) {
    this->x1 = x1;
    this->x2 = x2;
    this->x3 = x3;
    this->x4 = x4;
    this->y1 = y1;
    this->y2 = y2;
    this->y3 = y3;
    this->y4 = y4;
}

void Square::move_square_pointer(double *x, double *y) {
    x1 += *x;
    x2 += *x;
    x3 += *x;
    x4 += *x;
    y1 += *y;
    y2 += *y;
    y3 += *y;
    y4 += *y;
}

void Square::move_square_link(double &x, double &y) {
    x1 += x;
    x2 += x;
    x3 += x;
    x4 += x;
    y1 += y;
    y2 += y;
    y3 += y;
    y4 += y;
}

void Square::print() {
    cout << "_______________" << "\n";
    cout << "|" << "(" << x1 << ", " << y1 << ")" << " " << "(" << x2 << ", "
<< y2 << ")" << "|\n";
    cout << "|" << "               " << "|\n";
    cout << "|" << "(" << x3 << ", " << y3 << ")" << " " << "(" << x4 << ", "
<< y4 << ")" << "|\n";
    cout << "---------------" << "\n";
}
// end
```

Реализовал 2 процедуры для передвижения квадрата на заданный вектор: move_square_pointer и move_square_link.

В качестве входных параметров процедуры принимаю координаты x и y на которые нужно сместить наш квадрат. Перемещение реализовано путём прибавления к координатам углов фигуры.

4.

```
void matrix_swap_link(Matrix &matrix, int &first, int &second) {
    for (int i = 0; i < matrix.value[0].size(); i++) {
        swap(matrix.value[first][i], matrix.value[second][i]);
    }
}

void matrix_swap_pointer(Matrix *matrix, int *first, int *second) {
    for (int i = 0; i < matrix->value[0].size(); i++) {
        swap(matrix->value[*first][i], matrix->value[*second][i]);
    }
}

void matrix_print_link(Matrix &matrix) {
    for (int i = 0; i < matrix.value.size(); i++) {
        for (int j = 0; j < matrix.value[i].size(); j++) {
            cout << matrix.value[i][j] << ' ';
        }
        cout << '\n';
    }
}

void matrix_print_pointer(Matrix *matrix) {
    for (int i = 0; i < matrix->value.size(); i++) {
        for (int j = 0; j < matrix->value[i].size(); j++) {
            cout << matrix->value[i][j] << ' ';
        }
        cout << '\n';
    }
}
```

Реализовано по 2 функции для смены в матрице указанных строк (matrix_swap_link и matrix_swap_pointer) который принимаю в качестве аргументов строки которые мы хотим свапнуть. И 2 функции для вывода матрицы (matrix_print_link и matrix_swap_pointer), чтобы мы смогли убедиться, что свапаются нужные строки.