МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

# Лабораторные работы №1-3

**По дисциплине «Программирование»**

Выполнил студент группы №M3106

*Казаков Никита Андреевич*

Проверил

*Повышев Владислав Вячеславович*

Санкт-Петербург
2020

***Лабораторная работа №1***
*Передача значения по ссылке.*

*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.*

*main.cpp*

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

using namespace std;

int main() {
    //first task
    int first_value = 1;
    int second_value = 3;
    //second task
    double double_value_1 = 1.22222222;
    double double_value_2 = 1.33333333;
    //third task
    complex first;
    complex second;
    //fourth task
    circle krug;
    int x = 1;
    int y = 2;
    cout << '\n' << "Task 1 (swap): " << '\n';
    cout << "   first_value: " << first_value << " second_value: " <<
second_value << '\n' << "swap &" << ":" << '\n';
    custom_swap(first_value, second_value);
    cout << "   first_value: " << first_value << " second_value: " <<
second_value << '\n' << "swap *:" << '\n';
    custom_swap(&first_value, &second_value);
    cout << "   first_value: " << first_value << " second_value: " <<
second_value << '\n';
    cout << '\n' << "Task 4 (int_part): " << '\n';
    cout << "Int_part &:" << '\n';
    cout << "   Value: " << double_value_1 << '\n';
    custom_int_part(double_value_1);
    cout << "   Value: " << double_value_1 << '\n';
    cout << "Int_part *:" << '\n';
    cout << "   Value_2: " << double_value_2 << '\n';
    custom_int_part(&double_value_2);
    cout << "   Value_2: " << double_value_2 << '\n';
    cout << '\n' << "Task 9 (complex_multiplier): " << '\n';
    cout << "Multiplier_func &:" << '\n';
    cout << "   Value: ";
    print(first);
    cout << "   Multiplier: ";
```

```
    print(second);
    multiplier(first, second);
    cout << "   Value: ";
    print(first);
    cout << "Multiplier_func *:" << '\n';
    cout << "   Value: ";
    print(first);
    cout << "   Multiplier: ";
    print(second);
    multiplier(&first, &second);
    cout << "   Value: ";
    print(first);
    cout << '\n' << "Task 12 (transport): " << '\n';
    cout << "Circle position ";
    print(krug);
    cout << '\n';
    cout << "Transport(1, 2) &:" << '\n';
    transport(krug, x, y);
    cout << "   Circle position ";
    print(krug);
    cout << '\n';
    cout << "Transport(2, 1) *:" << '\n';
    transport(&krug, &y, &x);
    cout << "   Circle position ";
    print(krug);
    cout << '\n';
    return 0;
}
```

*functions.cpp*

```
#include <iostream>
#include <cmath>
#include "functions.h"
using namespace std;
void multiplier (complex &first, const complex &second) {
    first.a = first.a * second.a - first.b * second.b;
    first.b = first.a * second.b + first.b * second.a;
}
void multiplier (complex *first, const complex *second) {
    first->a = first->a * second->a - first->b * second->b;
    first->b = first->a * second->b + first->b * second->a;
}
void print (const complex &value) {
    cout << value.a << " + " << value.b << "i" << '\n';
}
void transport (circle &src, const int &x, const int &y) {
    src.position_x += x;
    src.position_y += y;
}
void transport (circle *src, const int *x, const int *y) {
    src->position_x += *x;
    src->position_y += *y;
}
void print(const circle &value) {
    cout << "X: " << value.position_x << " Y: " << value.position_y;
```

```
}
void custom_swap (int &first, int &second) {
    int mem;
    mem = first;
    first = second;
    second = mem;
}
void custom_swap (int *first, int *second) {
    int mem;
    mem = *first;
    *first = *second;
    *second = mem;
}
void custom_round (double &value, const int &sign) {
    value *= pow(10, sign);
    value = (int)value;
    value /= pow(10, sign);
}
void custom_round (double *value, const int &sign) {
    *value *= pow(10, sign);
    *value = (int)*value;
    *value /= pow(10, sign);
}
void custom_increase (int &src_variable, int multiplier) {
    src_variable += multiplier;
}
void custom_increase (int *src_variable, int multiplier) {
    *src_variable += multiplier;
}
void custom_biggest_increase (int &first, int &second) {
    if (first > second) {
        first += first % second;
    } else {
        second += second % first;
    }
}
void custom_biggest_increase (int *first, int *second) {
    if (*first > *second) {
        *first += *first % *second;
    } else {
        *second += *second % *first;
    }
}
void custom_int_part (double &value) {
    value = (int)value;
}
void custom_int_part (double *value) {
    *value = (int)*value;
}
void custom_fractional_part (double &value) {
    value -= (int)value;
}
void custom_fractional_part (double *value) {
    *value -= (int)*value;
}
void custom_inverse (double &value) {
    value *= -1;
```

```
}
void custom_inverse (double *value) {
    *value *= -1;
}
void custom_inverse (int &value) {
    value *= -1;
}
void custom_inverse (int *value) {
    *value *= -1;
}
void custom_reverse (double &value) {
    value = 1. / value;
}
void custom_reverse (double *value) {
    *value = 1. / *value;
}
```

*functions.h*

```
#pragma once
struct complex {
    double a = 1;
    double b = 2;
};
struct circle {
    double radius = 0;
    int position_x = 2;
    int position_y = 0;
};
void print (const complex &);
void multiplier (complex *, const complex *);
void multiplier (complex &, const complex &);
void print(const circle &);
void transport (circle *, const int *, const int *);
void transport (circle &, const int &, const int &);
void custom_round (double, const int &);
void custom_round (double &, const int &);
void custom_swap (int &, int &);
void custom_swap (int *, int *);
void custom_increase (int &, int);
void custom_increase (int *, int);
void custom_biggest_increase (int &, int &);
void custom_biggest_increase (int *, int *);
void custom_int_part (double &);
void custom_int_part (double *);
void custom_fractional_part (double &);
void custom_fractional_part (double *);
void custom_inverse (double &);
void custom_inverse (double *);
void custom_inverse (int &);
void custom_inverse (int *);
void custom_reverse (double &);
void custom_reverse (double *);
```

Санкт-Петербург
2020