**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»**

**(МТУСИ)**

Кафедра «Информационная безопасность»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

по дисциплине

**«Разработка безопасного ПО»**

на тему

«Модули. Линейные, разветвляющиеся и итерационные алгоритмы»

**Вариант №10**

Выполнила:

студентка группы БПЗ1902

Клевцова В.А.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва 2022

**Цель работы**

Овладеть навыками создания модулей в языке C++ и научится создавать линейные, разветвляющиеся и итерационные программы на языке C++.

**Задание к работе**

* *Задание 1*

Экспортировать функции , , , . Само пространство имен НЕ экспортировать.

При необходимости подключить заголовочные файлы, используйте фрагмент глобального модуля (Global Module Fragment).

* *Задание 2*

При реализации функции использовать инструкцию выбора if.

Функции и реализовать с помощью цикла while.

Объявления функций , , , поместить в единицу трансляции, описывающую интерфейс модуля (Module Interface Unit). Объявление функции помещать не нужно.

Экспортировать пространство имен RBPO::Lab3::Task2 целиком.

Определения функций , f , , поместить в одну единицу трансляции, описывающую реализацию модуля (Module Implementation Unit).

При необходимости подключить заголовочные файлы, используйте используйте фрагмент глобального модуля (Global Module Fragment).

* *Задание 3*

Функции и реализовать с помощью цикла do … while. Реализации функций должны возвращать те же результаты, что и в предыдущих заданиях.

Объявления функций , , , , поместить в единицу трансляции, описывающую интерфейс модуля (Module Interface Unit).

Экспортировать функции , , , . Само пространство имен НЕ экспортировать.

Каждое определение функций , f , , поместить в отдельную единицу трансляции, описывающую реализацию модуля (Module Implementation Unit).

Итоговый модуль будет содержать одну единицу трансляции, описывающую интерфейс модуля и 5 единиц трансляции, описывающих реализацию модуля. В каждой единице трансляции будет размещено одно определение функции.

При необходимости подключить заголовочные файлы, используйте фрагмент глобального модуля (Global Module Fragment).

* *Задание 4*

Модуль разделить на 5 разделов (module partition).

Объявление каждой из функций , , , , поместить в отдельную единицу трансляции, описывающую интерфейс раздела модуля (Module Partition Interface Unit).

Экспортировать функции , , , в соответствующих единицах трасляции.

В единице трансляции, описывающей интерфейс модуля (Module Interface Unit) импортировать разделы, содержащие экспорт фукнций , , , и экспортировать их.

Каждое определение функций , f , , поместить в отдельную единицу трансляции, описывающую реализацию раздела модуля (Module Partition Implementation Unit). В единицах трансляции, содержащих реализации и потребуется импортировать раздел с объявлением функции .

Итоговый модуль будет содержать одну единицу трансляции, описывающую интерфейс модуля и 5 разделов. Каждый из разделов будет включать единицу трансляции, описывающую интерфейс раздела модуля (в которой будет содержаться прототип функции), и единицу трансляции, описывающую реализацию раздела модуля (в которой будет содержаться реализация фукнции).

При необходимости подключить заголовочные файлы, используйте фрагмент глобального модуля (Global Module Fragment).

* *Задание 5*

Объявления и определения функций , , , , поместить в одну единицу трансляции, описывающую интерфейс модуля (Module Interface Unit – файл с расширением \*.ixx).

При этом определение (реализацию) функций поместить в приватный фрагмент модуля (Module Private Fragment).

Экспортировать функции , , , . Само пространство имен НЕ экспортировать.

При необходимости подключить заголовочные файлы, используйте фрагмент глобального модуля (Global Module Fragment).

* *Задание 6*

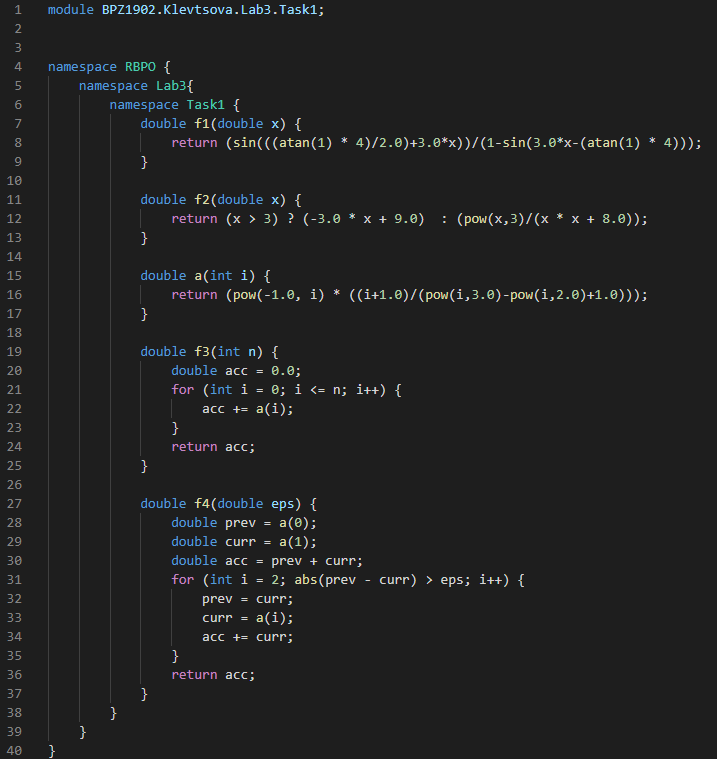
Разработать функцию main, демонстрирующую работу всех разработанных функций в заданиях 1-5.

Функция должна в цикле показывать меню и давать возможность выбрать задание и продемонстрировать работу всех функций из этого задания.

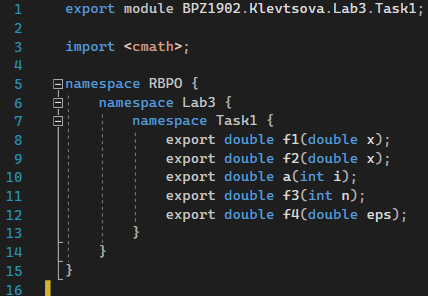
**Выполнение работы**

**Задание 1.**

Код BPZ1902.Klevtsova.Lab3.task1.cpp:

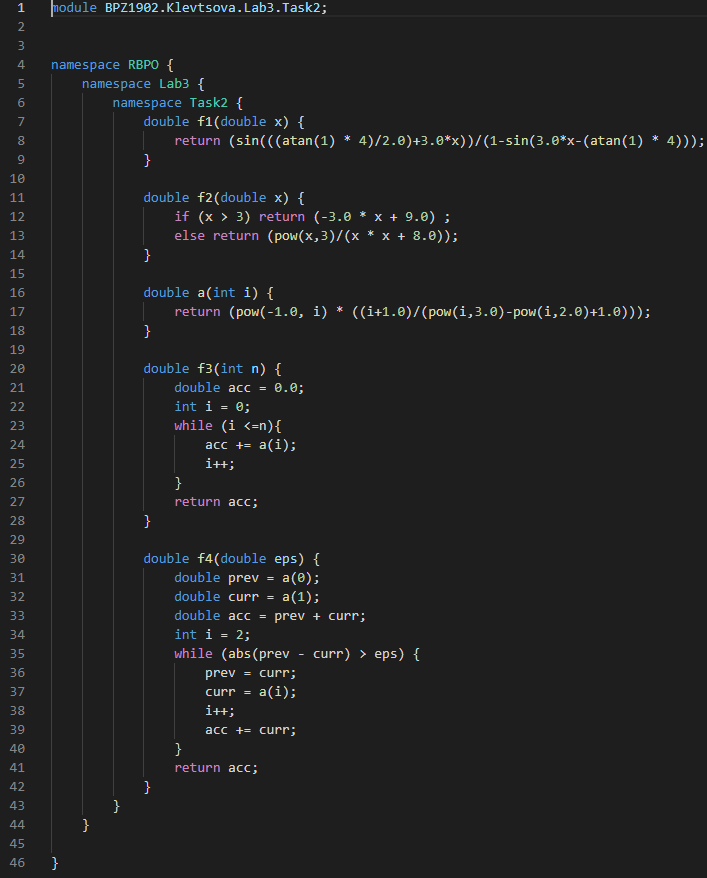


Код BPZ1902.Klevtsova.Lab3.task1.ixx:

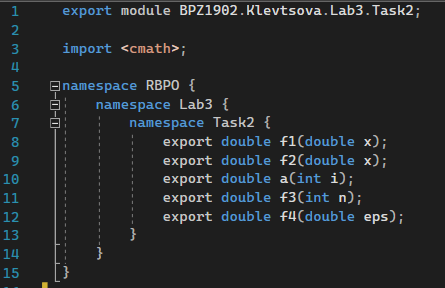


**Задание 2.**

Код BPZ1902.Klevtsova.Lab3.task2.cpp:

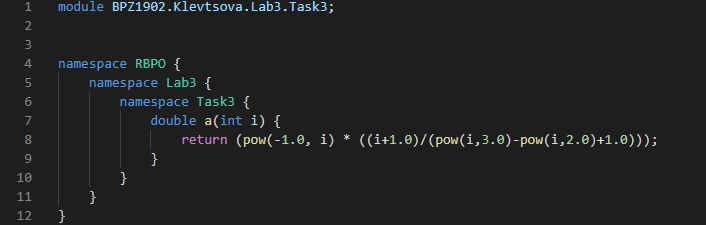


Код BPZ1902.Klevtsova.Lab3.task2.ixx:

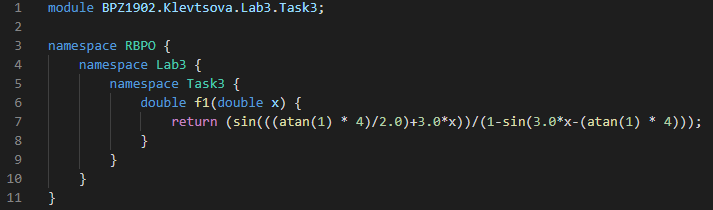


**Задание 3.**

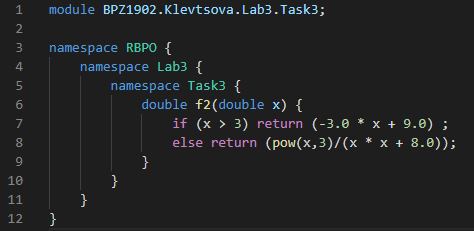
Код BPZ1902.Klevtsova.Lab3.task3.a.cpp:



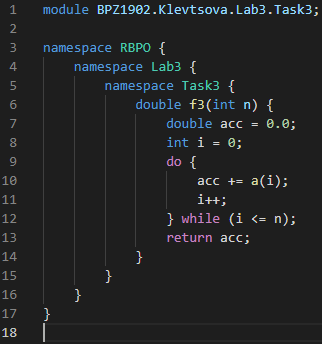
Код BPZ1902.Klevtsova.Lab3.task3.f1.cpp:



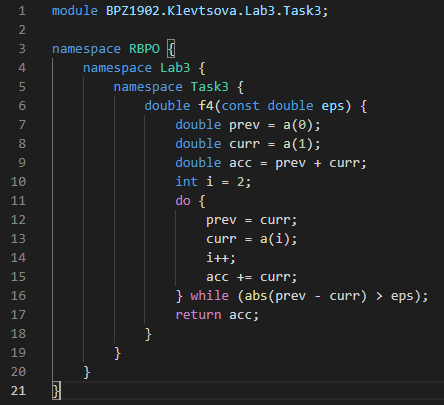
Код BPZ1902.Klevtsova.Lab3.task3.f2.cpp:



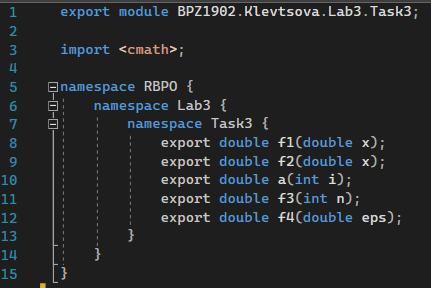
Код BPZ1902.Klevtsova.Lab3.task3.f3.cpp:



Код BPZ1902.Klevtsova.Lab3.task3.f4.cpp:

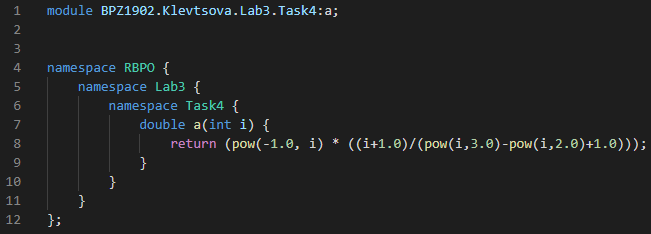


Код BPZ1902.Klevtsova.Lab3.task3.ixx:

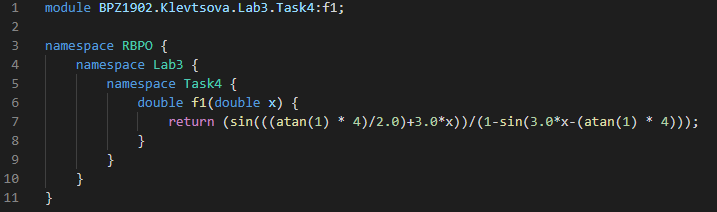


**Задание 4.**

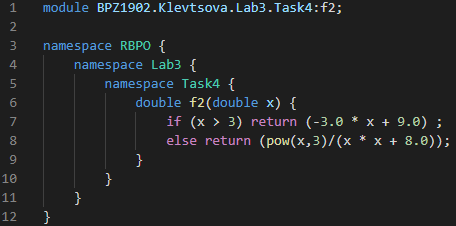
Код BPZ1902.Klevtsova.Lab3.task4.a.cpp:



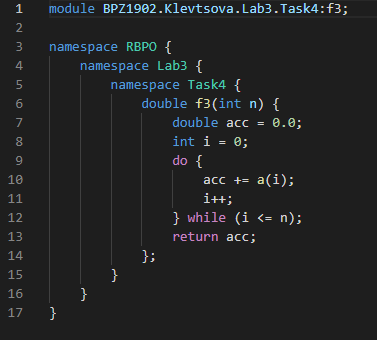
Код BPZ1902.Klevtsova.Lab3.task4.f1.cpp:



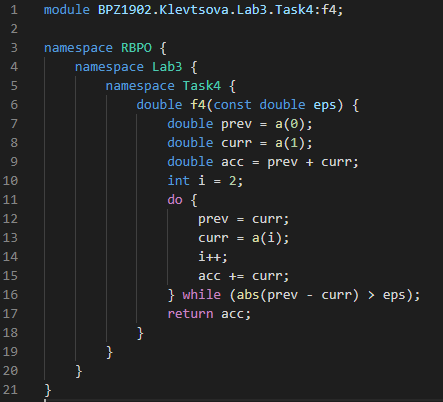
Код BPZ1902.Klevtsova.Lab3.task4.f2.cpp:



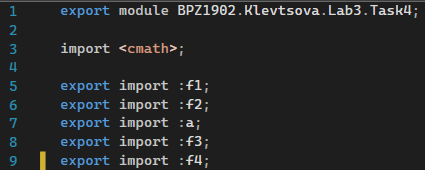
Код BPZ1902.Klevtsova.Lab3.task4.f3.cpp:



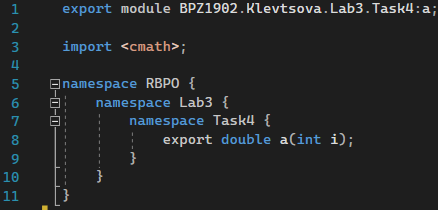
Код BPZ1902.Klevtsova.Lab3.task4.f4.cpp:



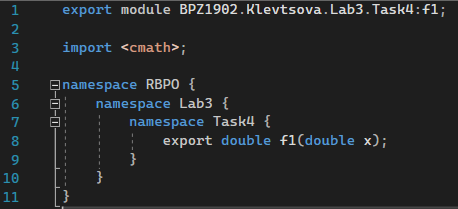
Код BPZ1902.Klevtsova.Lab3.task4.ixx:



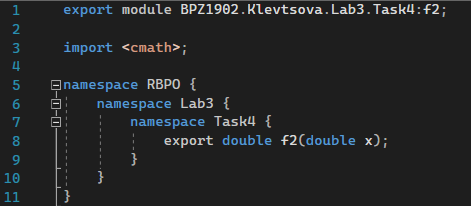
Код BPZ1902.Klevtsova.Lab3.task4.a.ixx:



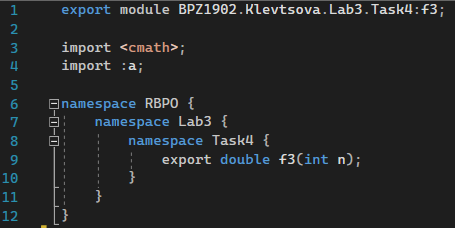
Код BPZ1902.Klevtsova.Lab3.task4.f1.ixx:



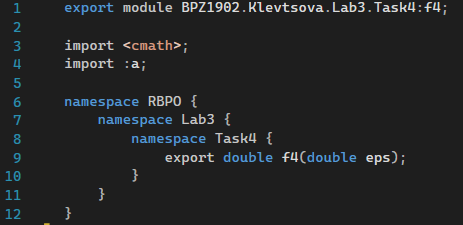
Код BPZ1902.Klevtsova.Lab3.task4.f2.ixx:



Код BPZ1902.Klevtsova.Lab3.task4.f3.ixx:

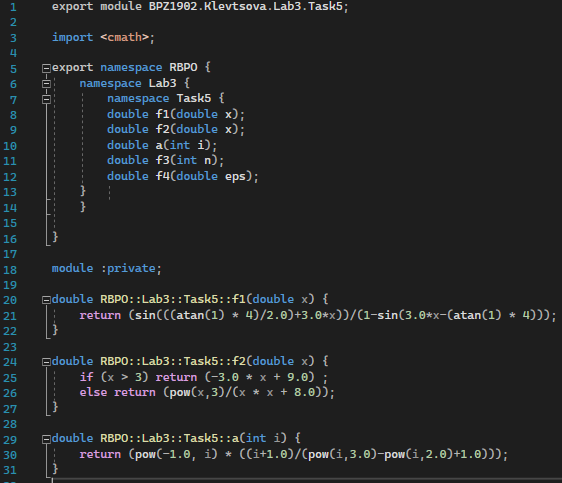


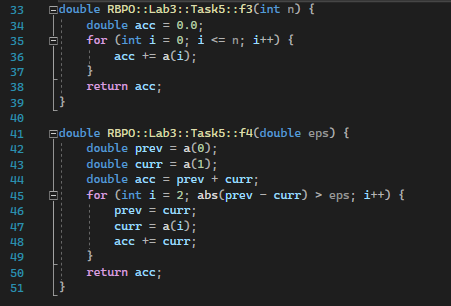
Код BPZ1902.Klevtsova.Lab3.task4.f4.ixx:



**Задание 5.**

Код BPZ1902.Klevtsova.Lab3.Task5.ixx:





**Задание 6.**

Код main.cpp:

#include <iostream>

import BPZ1902.Klevtsova.Lab3.Task1;

import BPZ1902.Klevtsova.Lab3.Task2;

import BPZ1902.Klevtsova.Lab3.Task3;

import BPZ1902.Klevtsova.Lab3.Task4;

import BPZ1902.Klevtsova.Lab3.Task5;

using namespace std;

void task1();

void task2();

void task3();

void task4();

void task5();

int main() {

    setlocale(LC\_ALL, "Russian");

    while (true){

        int choose = 0;

        printf("Выберите задание\n\t1 - Task1\n\t2 - Task2\n\t3 - Task3\n\t4 - Task4\n\t5 - Task5\n\t6 - Выход\nПункт: ");

        cin >> choose;

        printf("\n");

        switch (choose){

            case 1:

                task1();

                printf("\n");

                break;

            case 2:

                task2();

                printf("\n");

                break;

            case 3:

                task3();

                printf("\n");

                break;

            case 4:

                task4();

                printf("\n");

                break;

            case 5:

                task5();

                printf("\n");

                break;

            case 6:

                printf("Завершение программы");

                return 0;

            default:

                printf("Введите корректные данные!\n");

        }

    }

    return 0;

}

void task1() {

    double x;

    double eps;

    int n;

    cout << "Введите значение переменной x: ";

    cin >> x;

    cout << "Введите значение погрешности eps: ";

    cin >> eps;

    cout << "Введите количество итераций n: ";

    cin >> n;

    cout << "f1(" << x << ") : " << RBPO::Lab3::Task1::f1(x) << endl;

    cout << "f2(" << x << ") : " << RBPO::Lab3::Task1::f2(x) << endl;

    cout << "f3(" << n << ") : " << RBPO::Lab3::Task1::f3(n) << endl;

    cout << "f4(" << eps << ") : " << RBPO::Lab3::Task1::f4(eps) << endl;

}

void task2() {

    double x;

    double eps;

    int n;

    cout << "Введите значение переменной x: ";

    cin >> x;

    cout << "Введите значение погрешности eps: ";

    cin >> eps;

    cout << "Введите количество итераций n: ";

    cin >> n;

    cout << "f1(" << x << ") : " << RBPO::Lab3::Task2::f1(x) << endl;

    cout << "f2(" << x << ") : " << RBPO::Lab3::Task2::f2(x) << endl;

    cout << "f3(" << n << ") : " << RBPO::Lab3::Task2::f3(n) << endl;

    cout << "f4(" << eps << ") : " << RBPO::Lab3::Task2::f4(eps) << endl;

}

void task3() {

    double x;

    double eps;

    int n;

    cout << "Введите значение переменной x: ";

    cin >> x;

    cout << "Введите значение погрешности eps: ";

    cin >> eps;

    cout << "Введите количество итераций n: ";

    cin >> n;

    cout << "f1(" << x << ") : " << RBPO::Lab3::Task3::f1(x) << endl;

    cout << "f2(" << x << ") : " << RBPO::Lab3::Task3::f2(x) << endl;

    cout << "f3(" << n << ") : " << RBPO::Lab3::Task3::f3(n) << endl;

    cout << "f4(" << eps << ") : " << RBPO::Lab3::Task3::f4(eps) << endl;

}

void task4() {

    double x;

    double eps;

    int n;

    cout << "Введите значение переменной x: ";

    cin >> x;

    cout << "Введите значение погрешности eps: ";

    cin >> eps;

    cout << "Введите количество итераций n: ";

    cin >> n;

    cout << "f1(" << x << ") : " << RBPO::Lab3::Task4::f1(x) << endl;

    cout << "f2(" << x << ") : " << RBPO::Lab3::Task4::f2(x) << endl;

    cout << "f3(" << n << ") : " << RBPO::Lab3::Task4::f3(n) << endl;

    cout << "f4(" << eps << ") : " << RBPO::Lab3::Task4::f4(eps) << endl;

}

void task5() {

    double x;

    double eps;

    int n;

    cout << "Введите значение переменной x: ";

    cin >> x;

    cout << "Введите значение погрешности eps: ";

    cin >> eps;

    cout << "Введите количество итераций n: ";

    cin >> n;

    cout << "f1(" << x << ") : " << RBPO::Lab3::Task5::f1(x) << endl;

    cout << "f2(" << x << ") : " << RBPO::Lab3::Task5::f2(x) << endl;

    cout << "f3(" << n << ") : " << RBPO::Lab3::Task5::f3(n) << endl;

    cout << "f4(" << eps << ") : " << RBPO::Lab3::Task5::f4(eps) << endl;

}

**Результат работы программы**

