**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 №14**

Выполнил:

студент группы БПЗ1902

Оглоблин Д.И.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва, 2021

# Цель работы

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

# Задание

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.

# Индивидуальный вариант задания

Рис.1 Функция f1(x)

Рис.2 Функция f2(x)

Рис.3 Функция a(n)

# Выполнение домашнего задания

*Ссылка на гитхаб:* https://github.com/n3rfy/labs\_cpp/tree/main/lab3

**Задание 1**

**Код BPZ1902.Ogloblin.Lab3.task1.cpp:**

module BPZ1902.Ogloblin.Lab3.Task1;

namespace RBPO {

namespace Lab3{

namespace Task1 {

double f1(double x) {

return (cos(x) + sin(x)) / (cos(x) - sin(x));

}

double f2(double x) {

return (x < 2) ? abs(1.0-sin(2.0\*x)) : (1.0 + x\*x) / sqrt (2.0\*x);

}

double a(int i) {

return pow(-1.0, i) \* (i+1.0)/(pow(i, 3.0)+1.0);

}

double f3(int n) {

double result = 0.0;

for (int i = 0; i <= n; i++) {

result += a(i);

}

return result;

}

double f4(double eps) {

double prev = a(0);

double curr = a(1);

double result = prev + curr;

for (int i = 2; abs(prev - curr) > eps; i++) {

prev = curr;

curr = a(i);

result += curr;

}

return result;

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task1.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task1;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task1 {

export double f1(double x);

export double f2(double x);

export double a(int i);

export double f3(int n);

export double f4(double eps);

}

}

}

**Задание 2**

**Код BPZ1902.Ogloblin.Lab3.task2.cpp:**

module BPZ1902.Ogloblin.Lab3.Task2;

namespace RBPO {

namespace Lab3 {

namespace Task2 {

double f1(double x) {

return (cos(x) + sin(x)) / (cos(x) - sin(x));

}

double f2(double x) {

return (x < 2) ? abs(1.0-sin(2.0\*x)) : (1.0 + x\*x) / sqrt (2.0\*x);

}

double a(int i) {

return pow(-1.0, i) \* (i+1.0)/(pow(i, 3.0)+1.0);

}

double f3(int n) {

double result = 0.0;

int i = 0;

while (i <=n){

result += a(i);

i++;

}

return result;

}

double f4(double eps) {

double prev = a(0);

double curr = a(1);

double result = prev + curr;

int i = 2;

while (abs(prev - curr) > eps) {

prev = curr;

curr = a(i);

i++;

result += curr;

}

return result;

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task2.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task2;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task2 {

export double f1(double x);

export double f2(double x);

export double a(int i);

export double f3(int n);

export double f4(double eps);

}

}

}

**Задание 3**

**Код BPZ1902.Ogloblin.Lab3.task3.a.cpp:**

module BPZ1902.Ogloblin.Lab3.Task3;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

double a(int i) {

return pow(-1.0, i) \* (i+1.0)/(pow(i, 3.0)+1.0);

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task3.f1.cpp:**

module BPZ1902.Ogloblin.Lab3.Task3;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

double f1(double x) {

return (cos(x) + sin(x)) / (cos(x) - sin(x));

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task3.f2.cpp:**

module BPZ1902.Ogloblin.Lab3.Task3;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

double f2(double x) {

return (x < 2) ? abs(1.0-sin(2.0\*x)) : (1.0 + x\*x) / sqrt (2.0\*x);

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task3.f3.cpp:**

module BPZ1902.Ogloblin.Lab3.Task3;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

double f3(int n) {

double result = 0.0;

int i = 0;

do {

result += a(i);

i++;

} while (i <= n);

return result;

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task3.f4.cpp:**

module BPZ1902.Ogloblin.Lab3.Task3;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

double f4(const double eps) {

double prev = a(0);

double curr = a(1);

double result = prev + curr;

int i = 2;

do {

prev = curr;

curr = a(i);

i++;

result += curr;

} while (abs(prev - curr) > eps);

return result;

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task3.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task3;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task3 {

export double f1(double x);

export double f2(double x);

export double a(int i);

export double f3(int n);

export double f4(double eps);

}

}

}

**Задание 4**

**Код BPZ1902.Ogloblin.Lab3.task4.a.cpp:**

module BPZ1902.Ogloblin.Lab3.Task4:a;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

double a(int i) {

return pow(-1.0, i) \* (i+1.0)/(pow(i, 3.0)+1.0);

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.a.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4:a;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

export double a(int i);

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f1.cpp:**

module BPZ1902.Ogloblin.Lab3.Task4:f1;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

double f1(double x) {

return (cos(x) + sin(x)) / (cos(x) - sin(x));

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f1.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4:f1;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

export double f1(double x);

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f2.cpp:**

module BPZ1902.Ogloblin.Lab3.Task4:f2;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

double f2(double x) {

return (x < 2) ? abs(1.0-sin(2.0\*x)) : (1.0 + x\*x) / sqrt (2.0\*x);

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f2.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4:f2;

import <cmath>;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

export double f2(double x);

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f3.cpp:**

module BPZ1902.Ogloblin.Lab3.Task4:f3;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

double f3(int n) {

double result = 0.0;

int i = 0;

do {

result += a(i);

i++;

} while (i <= n);

return result;

};

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f3.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4:f3;

import <cmath>;

import :a;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

export double f3(int n);

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f4.cpp:**

module BPZ1902.Ogloblin.Lab3.Task4:f4;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

double f4(const double eps) {

double prev = a(0);

double curr = a(1);

double result = prev + curr;

int i = 2;

do {

prev = curr;

curr = a(i);

i++;

result += curr;

} while (abs(prev - curr) > eps);

return result;

}

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.f4.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4:f4;

import <cmath>;

import :a;

namespace RBPO {

namespace Lab3 {

namespace Task4 {

export double f4(double eps);

}

}

}

**Код BPZ1902.Ogloblin.Lab3.task4.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task4;

import <cmath>;

export import :f1;

export import :f2;

export import :a;

export import :f3;

export import :f4;

**Задание 5**

**Код BPZ1902.Ogloblin.Lab3.Task5.ixx:**

export module BPZ1902.Ogloblin.Lab3.Task5;

import <cmath>;

export namespace RBPO {

namespace Lab3 {

namespace Task5 {

double f1(double x);

double f2(double x);

double a(int i);

double f3(int n);

double f4(double eps);

}

}

}

module :private;

double RBPO::Lab3::Task5::f1(double x) {

return (cos(x) + sin(x)) / (cos(x) - sin(x));

}

double RBPO::Lab3::Task5::f2(double x) {

return (x < 2) ? abs(1.0-sin(2.0\*x)) : (1.0 + x\*x) / sqrt (2.0\*x);

}

double RBPO::Lab3::Task5::a(int i) {

return pow(-1.0, i) \* (i+1.0)/(pow(i, 3.0)+1.0);

}

double RBPO::Lab3::Task5::f3(int n) {

double result = 0.0;

for (int i = 0; i <= n; i++) {

result += a(i);

}

return result;

}

double RBPO::Lab3::Task5::f4(double eps) {

double prev = a(0);

double curr = a(1);

double result = prev + curr;

for (int i = 2; abs(prev - curr) > eps; i++) {

prev = curr;

curr = a(i);

result += curr;

}

return result;

}

**Задание 6**

**Код main.cpp:**

#include <iostream>

import BPZ1902.Ogloblin.Lab3.Task1;

import BPZ1902.Ogloblin.Lab3.Task2;

import BPZ1902.Ogloblin.Lab3.Task3;

import BPZ1902.Ogloblin.Lab3.Task4;

import BPZ1902.Ogloblin.Lab3.Task5;

using namespace std;

void task1();

void task2();

void task3();

void task4();

void task5();

double x;

double eps;

int n;

int main() {

setlocale(LC\_ALL, "Russian");

while (true){

cout << "Введите значение x: ";

cin >> x;

cout << "Введите нужную точность eps: ";

cin >> eps;

cout << "Введите значение n: ";

cin >> n;

int choose = 0;

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

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:

return 0;

default:

printf("Выбирете число!\n");

}

}

return 0;

}

void task1() {

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() {

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() {

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() {

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() {

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;

}

**Выполнение:**



