ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ.

Цель работы: ознакомится с основополагающими концепциями объектно-ориентированного программирования.

Решение каждой лабораторной работы, должно находиться в «П*устом бланке решений, не содержащем проектов»*.

Создание пустого бланка решений

Запустить среду выполнения ***Microsoft Visual Studio***

1.1. В появившемся окне, в пункте меню ***File*** выбрать ***-> New-> Project***

Файл выбрать ***-> Создать-> Проект***

1.2.В появившемся окне «***New Project***» или «***Создать проект***»

Выбрать ***Other projects types -> Visual Studio Solutions ->Blank Solution***

Другие типы проектов -> Решения Visual Studio ->Новое решение

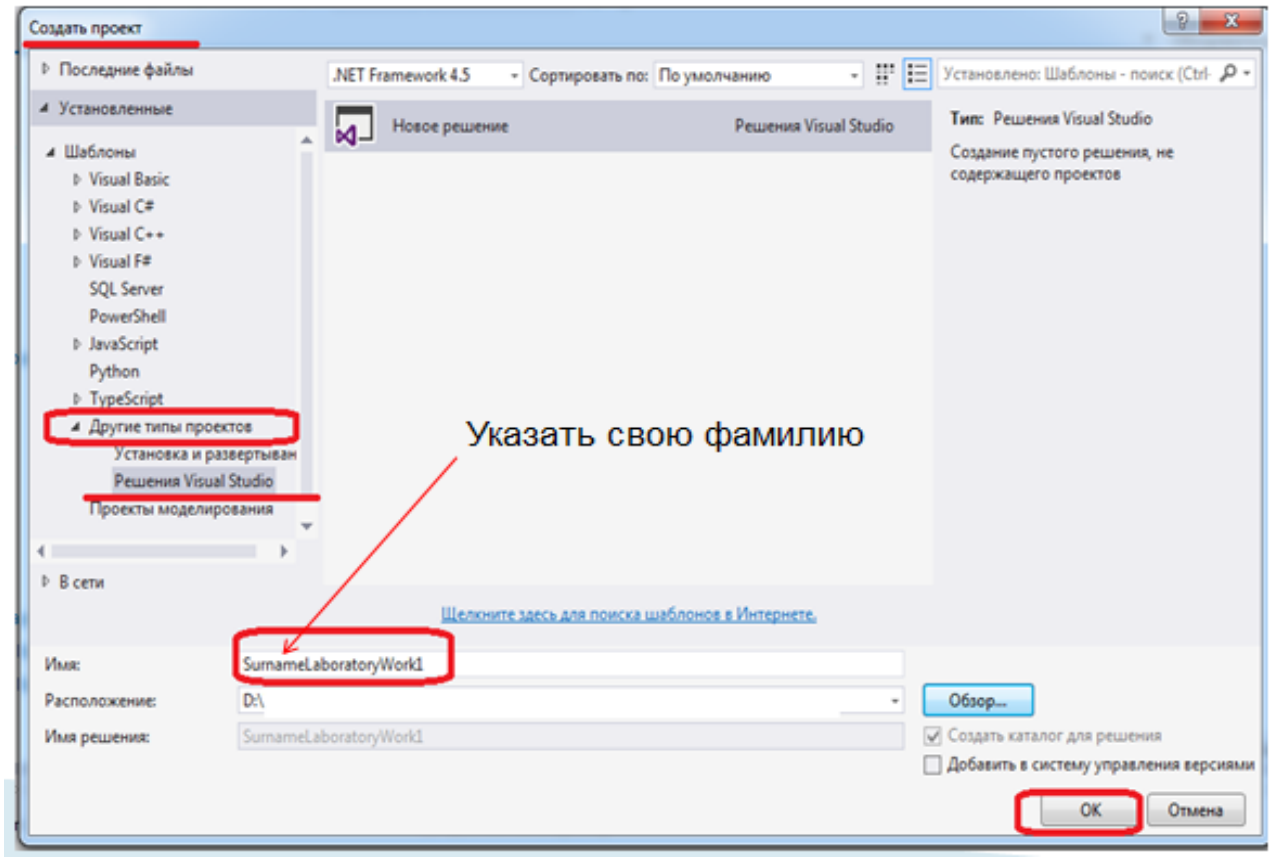
Задать имя проекта

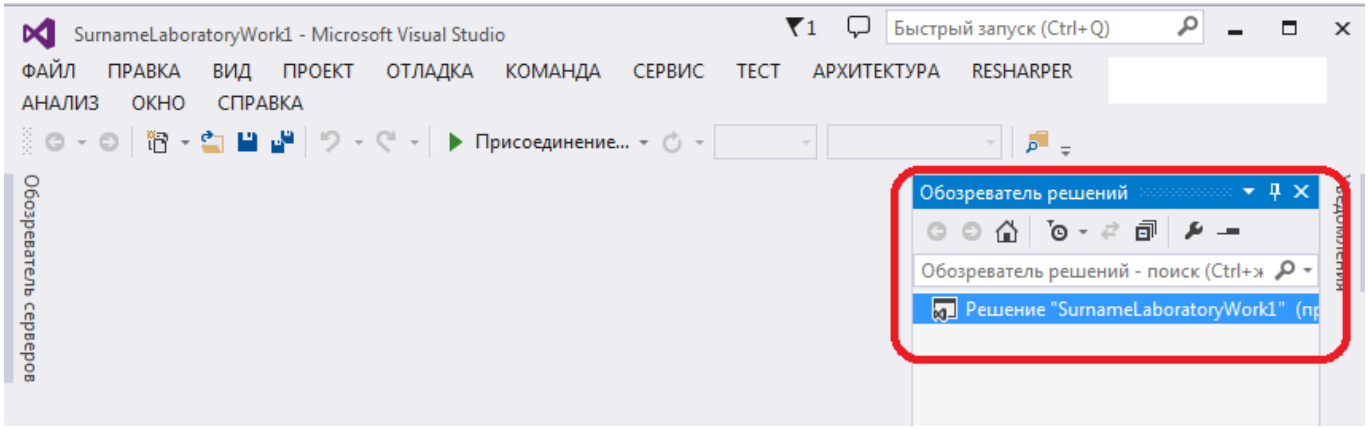
Имя: ЛабораторнаяРабота№

Name: LaboratoryWork№

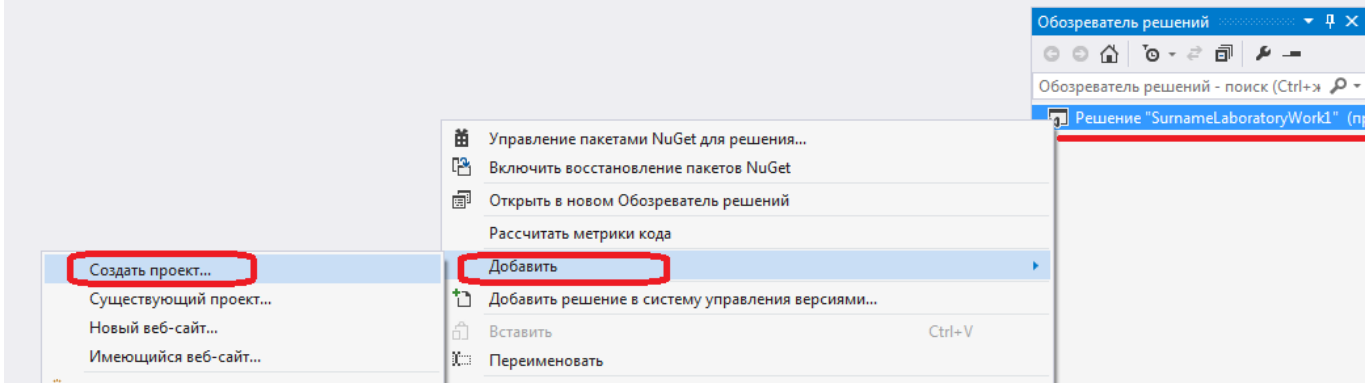
И место расположения проекта, для этого необходимо заранее на диске D или в любом другом месте

Location: D

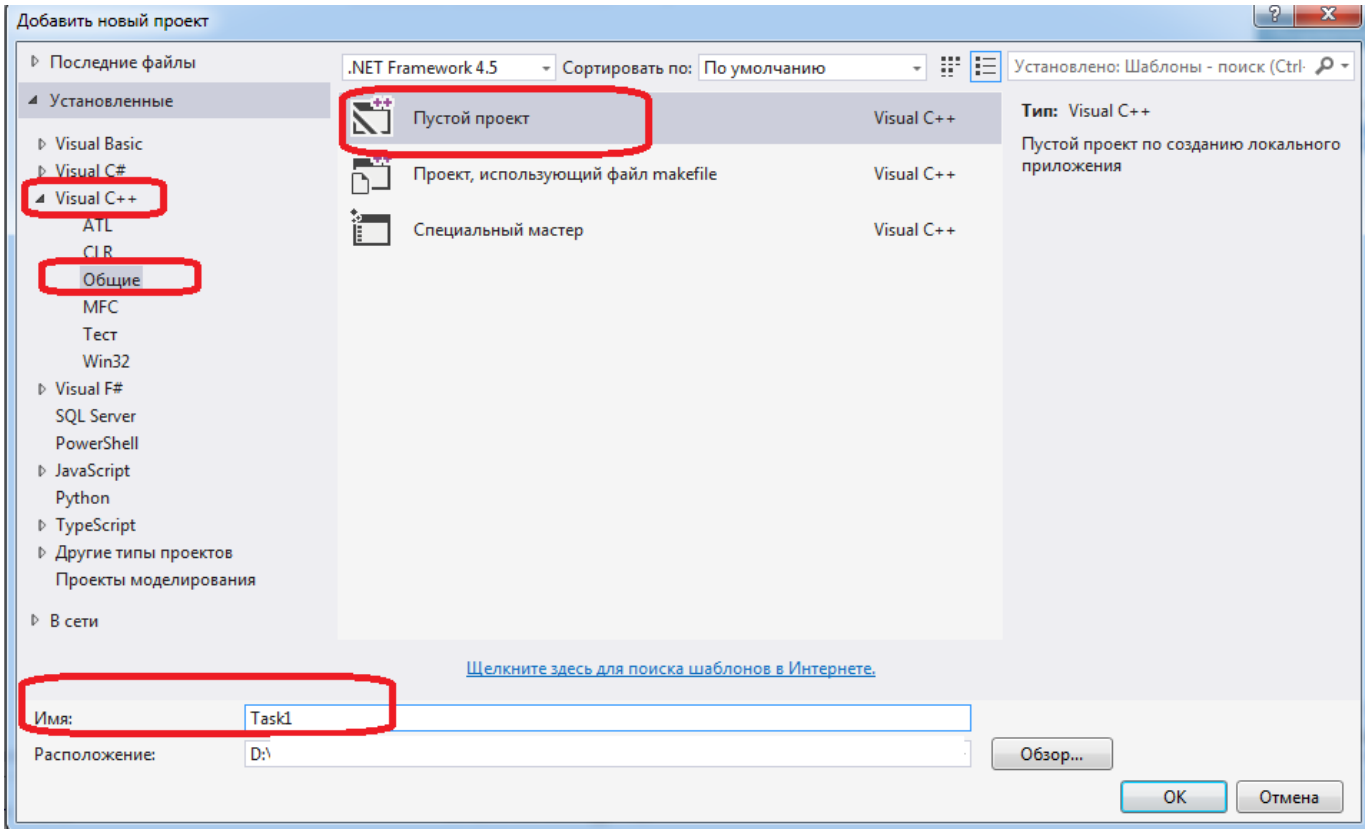


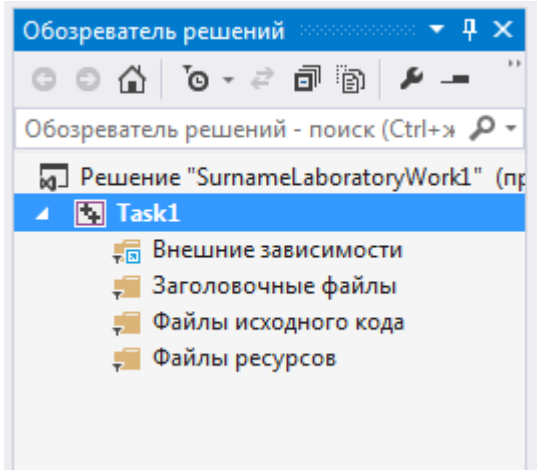


Каждое задание лабораторной работы – это отдельный проект, содержащийся в Бланке решений:
Необходимо вызвать контекстное меню ***Решение "SurnameLaboratoryWork1"***



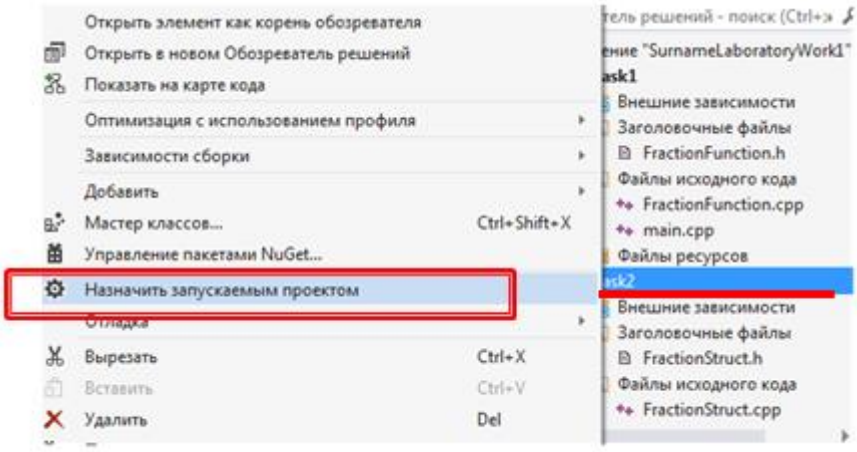
1.3.В появившемся окне «New Project» или «Добавить новый проект»
Выбрать ***Visual C++ -> General-> Empty Project***
Visual C++ -> Общие-> Пустой проект
Задать имя проекта
Name: Task№
Имя: Задание№





При этом каждый проект должен содержать минимум один заголовочный файл (т.е. с расширением .h) и два файла исходного модуля (т.е. с расширением .cpp).

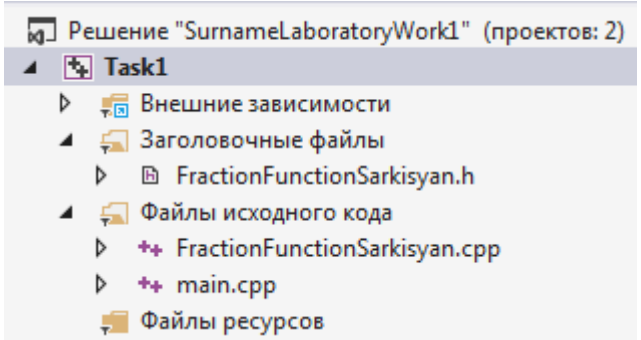
Бланк решений будет содержать столько проектов, сколько заданий в данной лабораторной работе и следовательно, чтобы запустить определенный проект, его необходимо сделать ***запускаемым***, для этого необходимо вызвать контекстное меню данного приложения, т.е. правой клавишей мыши и выбрать пункт "***Назначить запускаемым проектом***"



ЗАДАЧИ ДЛЯ ВЫПОЛНЕНИЯ

ПРОГРАММА, ВЫПОЛНЯЮЩАЯ ДЕЙСТВИЯ НАД ДРОБЯМИ.

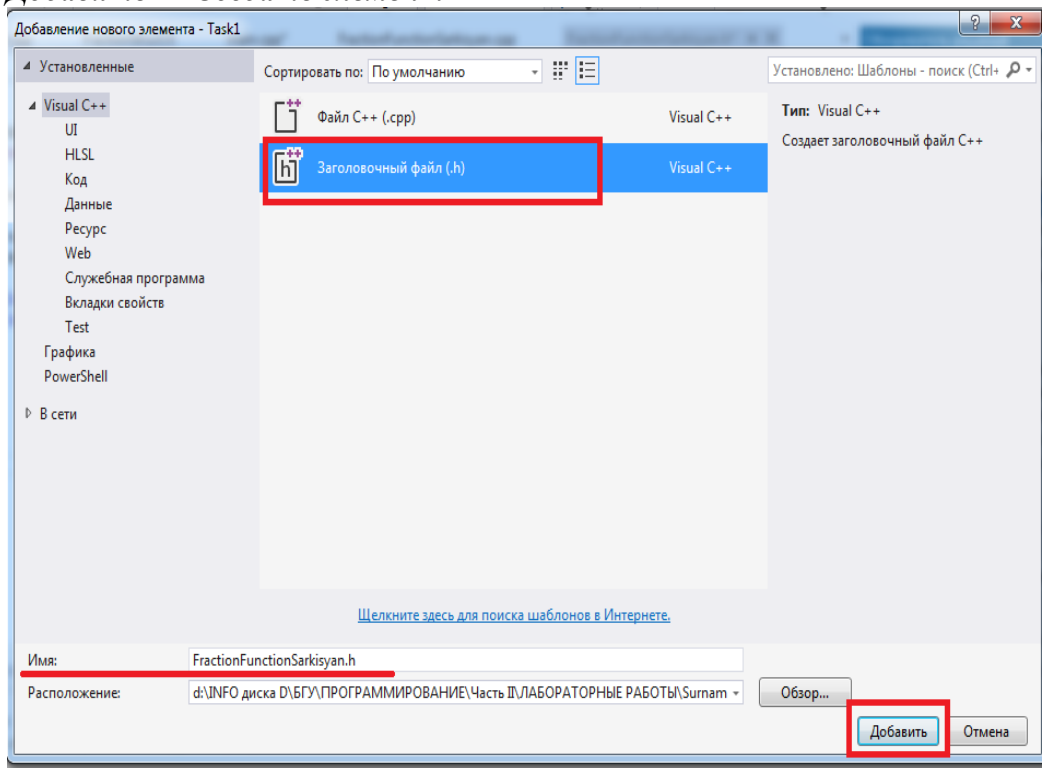
ЗАДАНИЕ1. *с помощью функций*



1. Создание заголовочного файла с расширением h.

Из контекстного меню папки

Header Files
Заголовочные файлы
выбрать
Add -> New Item…
Добавить -> Создать элемент

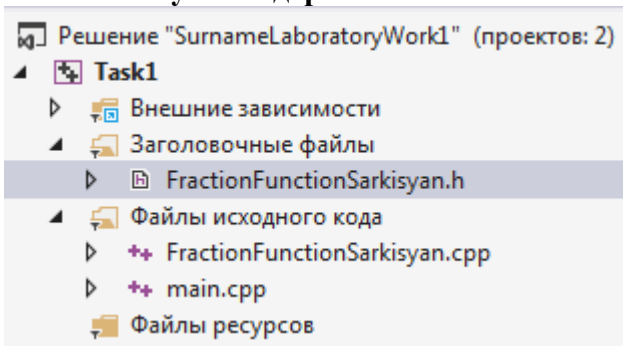


Одна из проблем связанная с заголовочным файлом (который содержит объявление типа, функций, констант и т.д.) – это дублирование самого заголовочного файла, т.к. в коде программы его приходится подключать неоднократно.
Чтобы избежать дублирования необходимо все определения, содержащиеся в заголовочном файле, записывать внутри директивы, которая называется *макроблокировкой*.

```
#ifndef ИмяHeaderFile_H //имя (заглавными буквами),
//которое Вы задали Header файлу, нижнее подчеркивание и H
#define ИмяHeaderFile_H //имя (заглавными буквами), которое
//Вы задали Header файлу, нижнее подчеркивание и H

Содержимое заголовочного файла

#endif
```

Операторы находяшиеся между строками `#ifndef ИмяHeaderFile_H` и `#endif` будут скомпилированы, только при первом обнаружении компилятором подключения заголовочного файла.

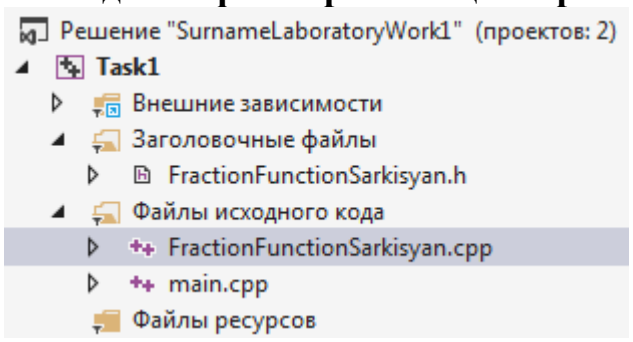
В нашем случае содержимое заголовочного файла выглядит так



Объявления прототипов функции:

1. **сумма (*`add`*);**
2. **вычитания (*`subtract`*);**
3. **умножения (*`multiplication`*);**
4. **деления (*`divide`*) дробей;**
5. **сокращения дроби.**
6. **распечатать результат оба операнда, операция и результат вычисления.**

2. Создание файла реализации с расширением .cpp для заголовочного файла



После создания заголовочному файлу, необходимо создавать и **файл реализации (для данного класса) с расширением cpp**

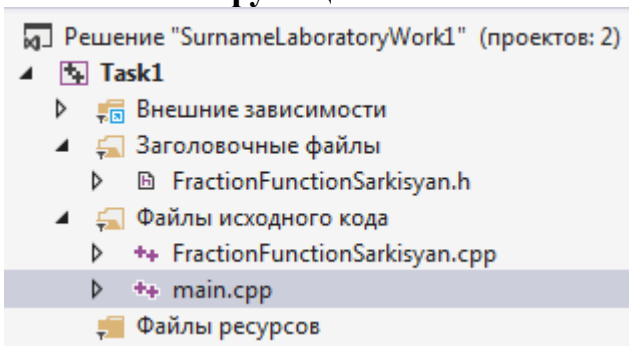
Файл реализации – это описание функций, прототипы которых находятся в заголовочном файле. Он должен в себя включать:

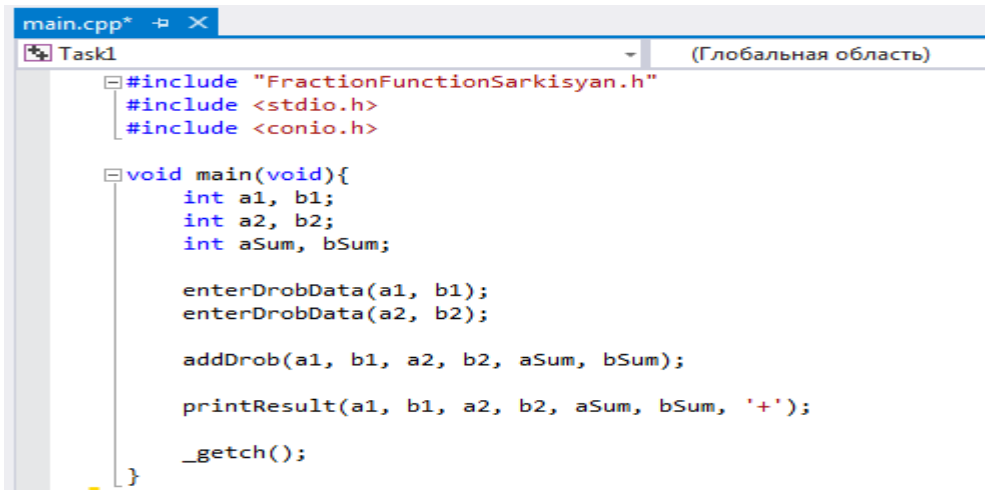
1. подключение всех необходимых библиотек и дерективу и обязательно тот заголовочный файл для котрого и создается данный файл реализации

```
#include “ИмяHeaderFile.h”;
```

2. описание всех функций описанных в заголовочном файле

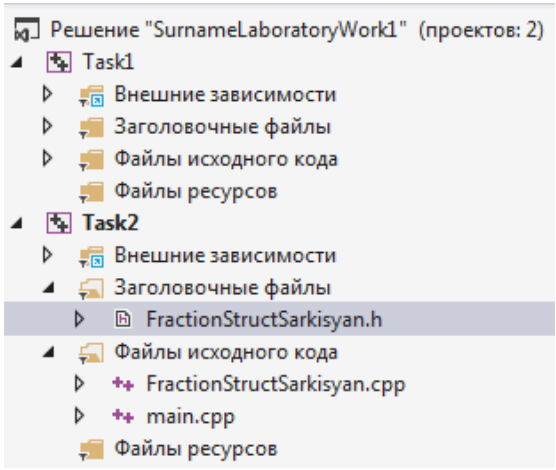
3. Создание файла реализации с расширением .cpp для использовании написанных функций.





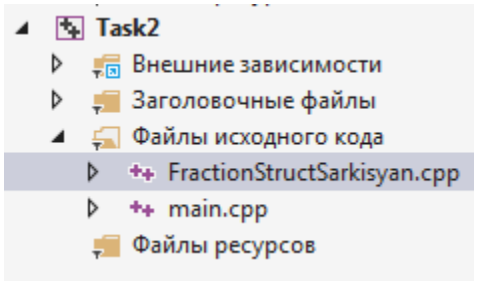
```
#include "FractionFunctionSarkisyan.h"
#include <stdio.h>
#include <conio.h>

void main(void){
    int a1, b1;
    int a2, b2;
    int aSum, bSum;

    enterDrobData(a1, b1);
    enterDrobData(a2, b2);

    addDrob(a1, b1, a2, b2, aSum, bSum);

    printResult(a1, b1, a2, b2, aSum, bSum, '+');

    _getch();
}
```

ЗАДАНИЕ 2. *с помощью структуры*

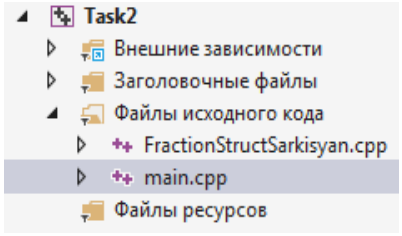


Объявление стркутуры и прототипы следующих функций

1. **сумма (*`add`*);**
2. **вычитания (*`subtract`*);**
3. **умножения (*`multiplication`*);**
4. **деления (*`divide`*) дробей;**
5. **сокращения дроби.**
6. **распечатать результат оба операнда, операция и результат вычисления.**

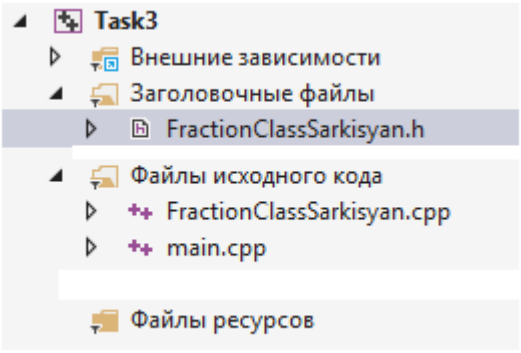


Файл реализации – это описание функций, прототипы которых находятся в заголовочном файле.



Продемонстрировать работу всех функций и создание объектов структур.

ЗАДАНИЕ 3. *с помощью классов*



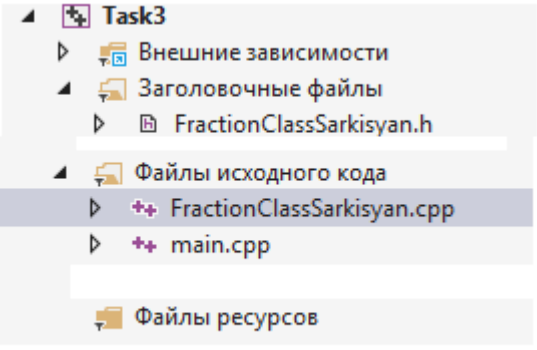
FractionClassSarkisyan.h*

Task3 (Глобальная область)

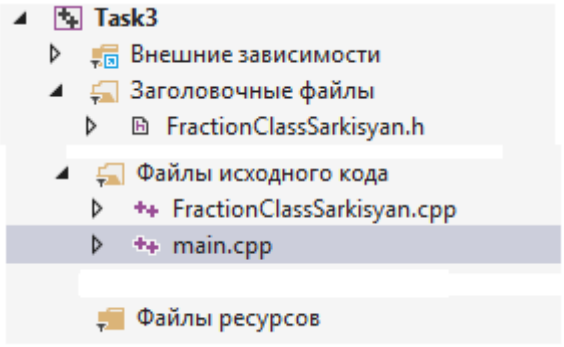
```
#ifndef FRACTIONCLASSSARKISYAN_H
#define FRACTIONCLASSSARKISYAN_H

class CFraction
{
private:
    int numerator;
    int denominator;

public:

    void setNumerator(int numerator);
    int getNumerator()const;

    void setDenominator(int denominator);
    int getDenominator() const;

    CFraction add(const CFraction& fraction);
};

#endif
```

Дописать следующие методы:

1. **вычитания (*`subtract`*);**
2. **умножения (*`multiplication`*);**
3. **деления (*`divide`*) дробей;**
4. **сокращения дроби.**
5. **распечатать результат оба операнда, операция и результат вычисления.**



Файл реализации – это описание методов описанных в классе.



main.cpp*

Task3 (Глобальная область)

```cpp
#include "FractionClassSarkisyan.h"
#include "PrintFractionClassSarkisyan.h"

#include <iostream>

using namespace std;

void main(void)
{
    int numerator, denominator;
    CFraction faction1, faction2, faction;

    cout<<"Enter the numerator of fractions: ";
    cin>>numerator;

    cout<<"Enter the denominator of the fraction:";
    cin>>denominator;

    faction1.setNumerator(numerator);
    faction1.setDenominator(denominator);

    cout << "Enter the numerator of fractions: ";
    cin >> numerator;

    cout << "Enter the denominator of the fraction:";
    cin >> denominator;

    faction2.setNumerator(numerator);
    faction2.setDenominator(denominator);

    faction = faction1.add(faction2);
```