Практическая работа №5

Составление простых алгоритмов

1 Цель работы

1.1 Научиться составлять программы, содержащие массивы;

2 Литература

2.1 Ашарина, И.В. Объектно-ориентированное программирование в C++: лекции и упражнения: учебное пособие для вузов. - Москва: Горячая линия - Телеком, 2014.

3 Основное оборудование

3.1 Персональный компьютер.

4 Подготовка к работе

4.1 Повторить основные алгоритмические конструкции, принципы алгоритмизации;

4.2 Подготовить бланк отчета.

5 Задание

5.1 Составить алгоритмы для предложенных заданий в Visio или LibreOffice Draw.

5.2 Составить отчет в электронном виде с помощью LibreOffice Writer.

5.3 Сохранить работу по пути C:\Temp\КСК-31\Практическая работа 5

6 Порядок выполнения работы

6.1 Пользователь инициализирует две переменные, значения переменных необходимо поменять местами и вывести их на экран.

6.2 Пользователь вводит две стороны прямоугольника (высоту и ширину), найти площадь и периметр прямоугольника.

6.3 Цифры 1, 2, 3 и 4 обозначают операции сложение, умножение, вычитание и деление. Предложить пользователю ввести два числа и выбрать действие. Выполнить операцию и вывести результат на экран

7 Содержание отчета

7.1 Титульный лист;

7.2 Цель работы;

7.3 Текст программ (скриншоты);

7.4 Ответы на контрольные вопросы;

7.5 Вывод по проделанной работе.

8 Контрольные вопросы

8.1 Что такое C++ и где используется?

8.2 Как создать новый проект в VisualStudio?

8.3 Какие базовые арифметические операции есть в C++?

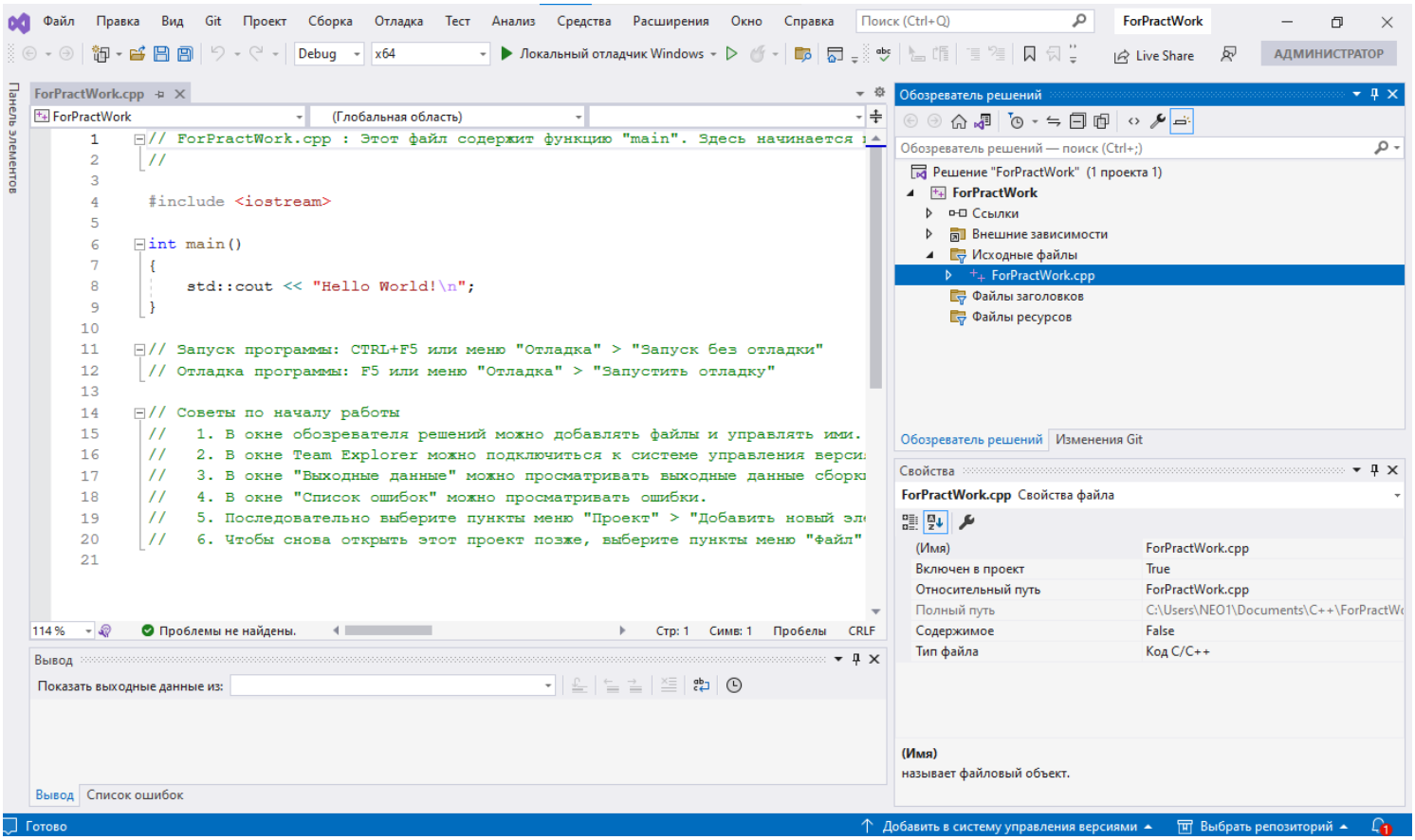
.4 Как подключить библиотеку?

9 Приложение

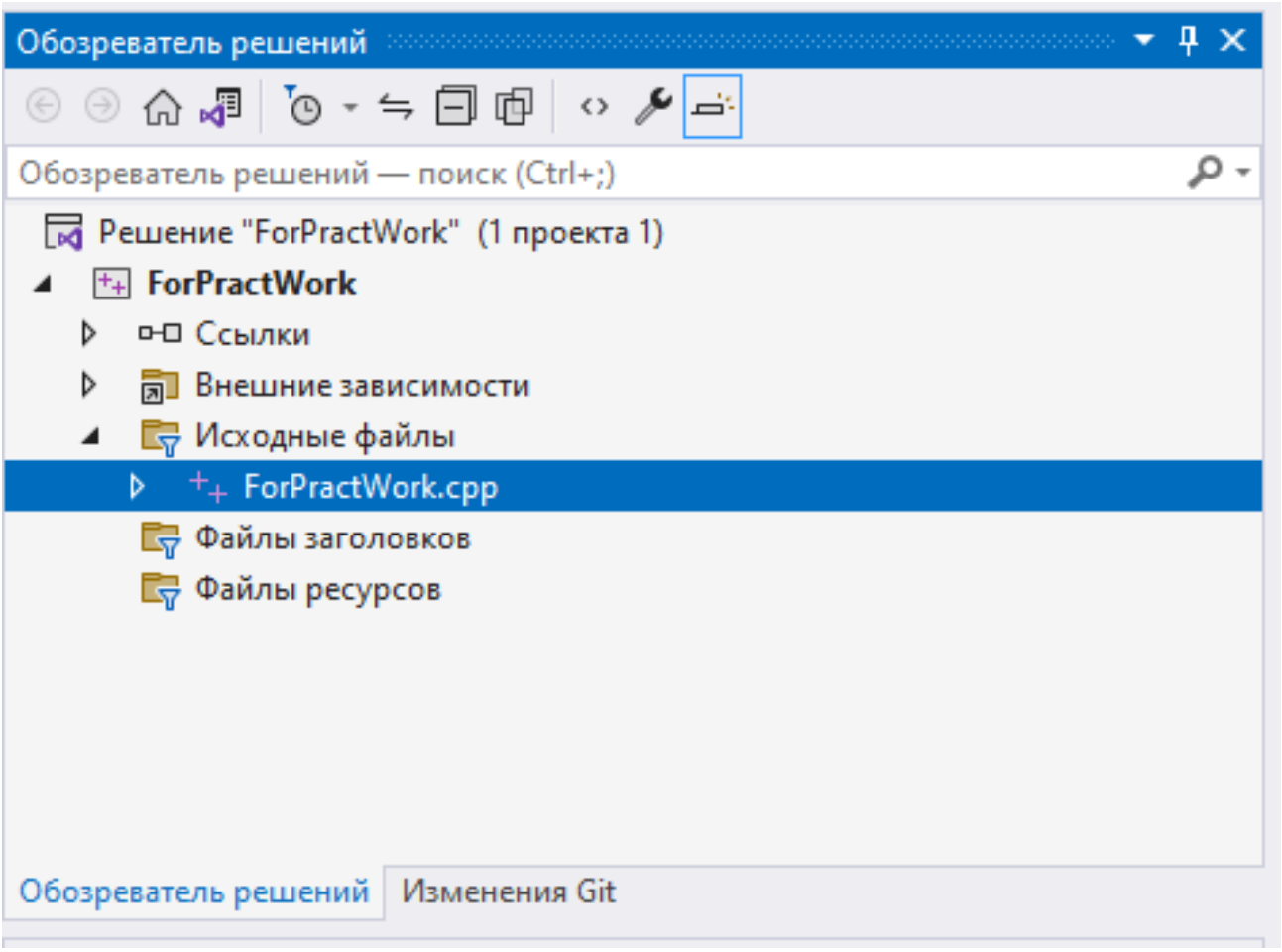
Создание нескольких проектов внутри одного решения

Именно так в нашем случае нужно создавать проекты т.к. появляется возможность быстро переключаться между проектами. Для этого:

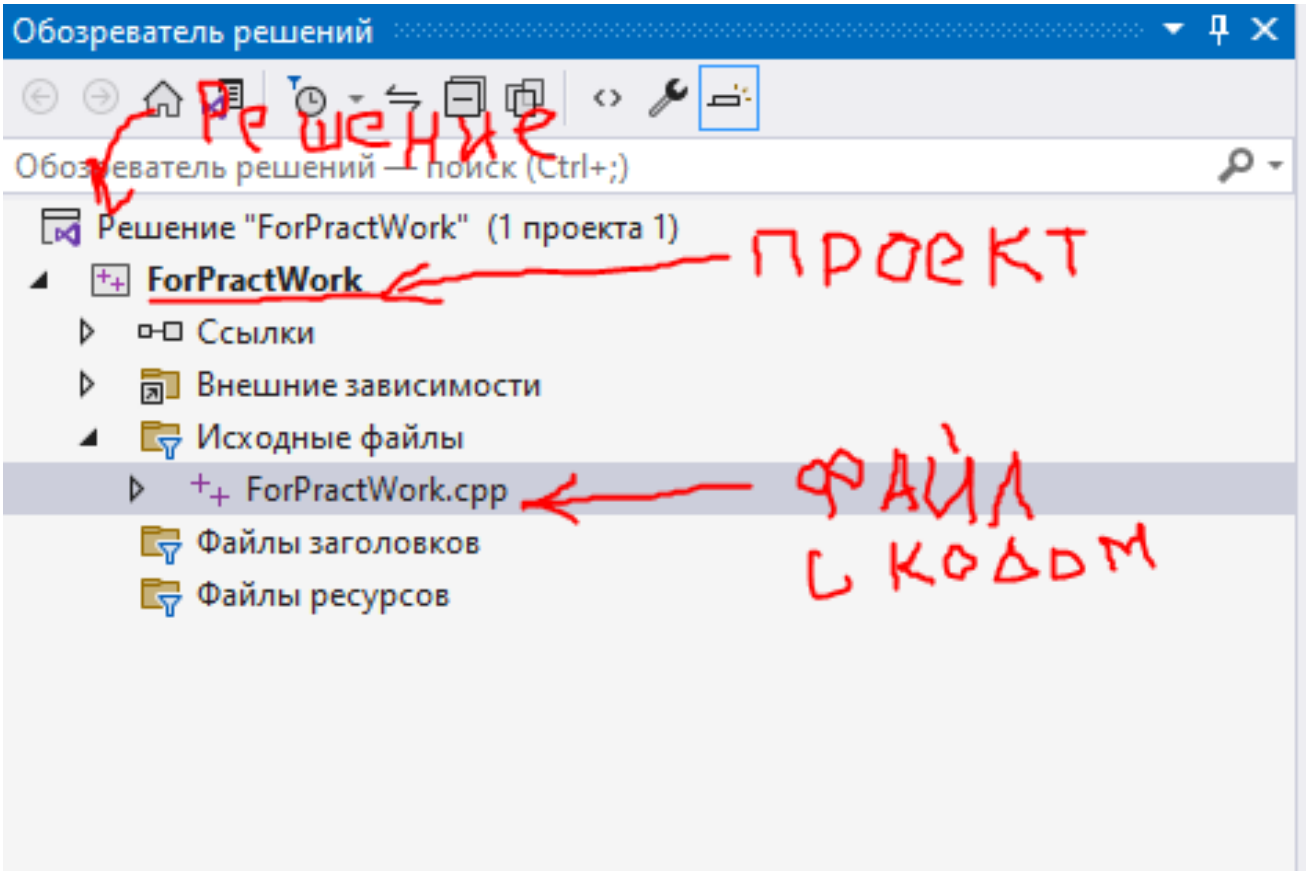
Создаете проект как обычно: Файл -> Создать -> Проект, тип проекта Win 32 Console Application. В качестве папки для сохранения выбирайте папку C:\Temp\КСК-31\ иначе рискуете потерять проект (Если что код сохраняется автоматически, так что ничего сохранять в cpp файлы не надо). Результат выглядит примерно так:



Справа вы можете наблюдать Обозреватель решений (Solution Explorer). Именно там отображается структура проекта_ов:

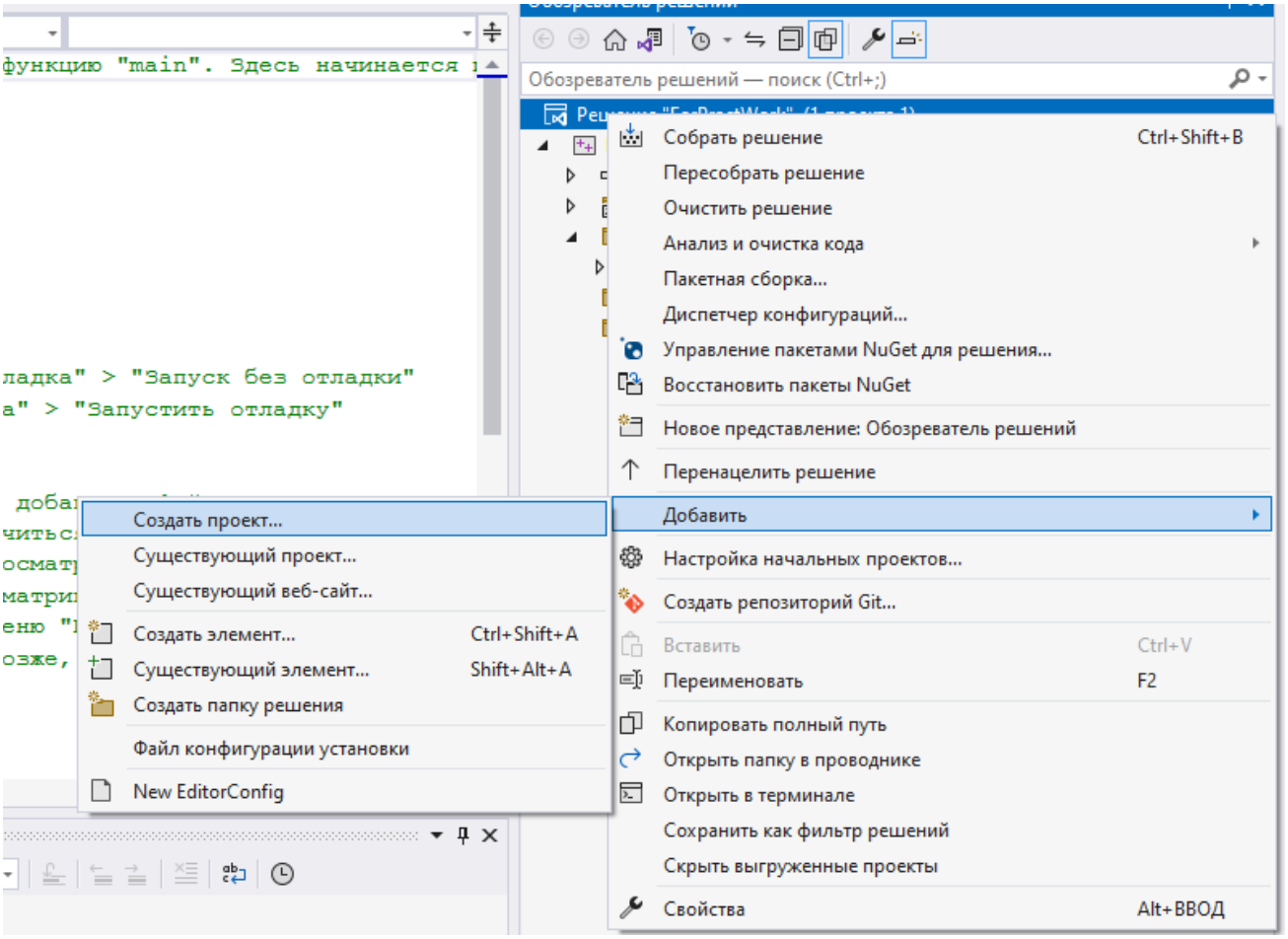


Внутри решения размешены все проекты и их файлы. Внутри проекта размещены файлы проекта, самый важный из них находится в папке «Исходные файлы» (Source files), называется так же, как и проект и имеет расширение .cpp. Именно внутри него вы пишете код. Создавать дополнительные файлы не нужно!!

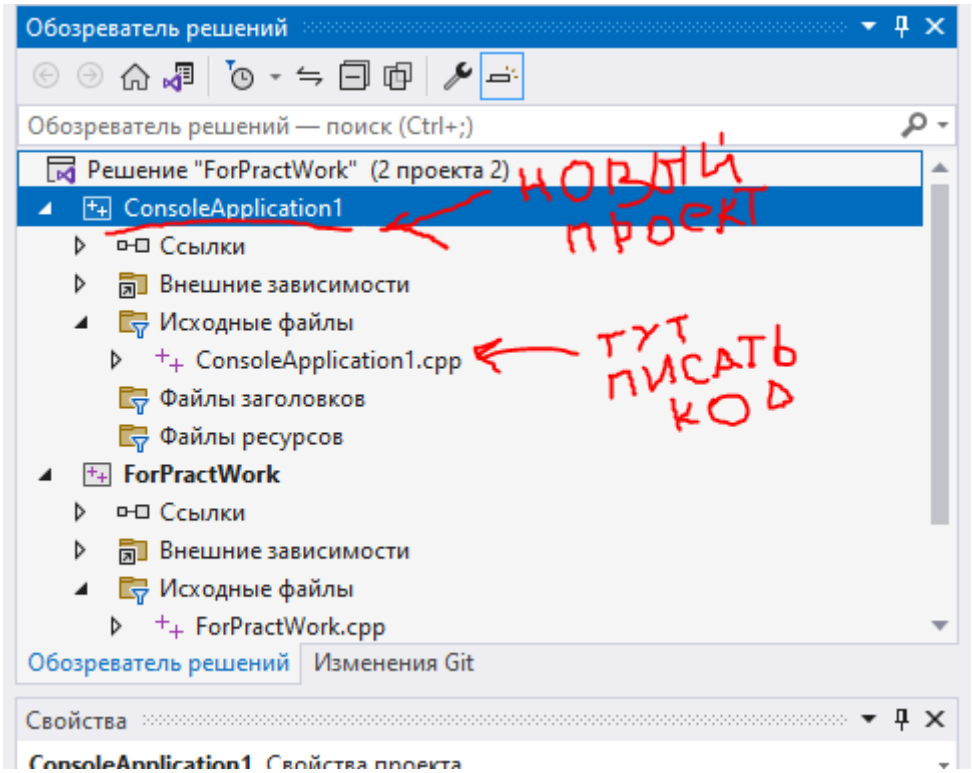


Предположим вы решаете лабу, в таком случае может быть полезным под каждое задание создавать новый проект. Чтобы не писать весь код вместе. Под первое задание 1 проект, под второе 2 проект и т.д.

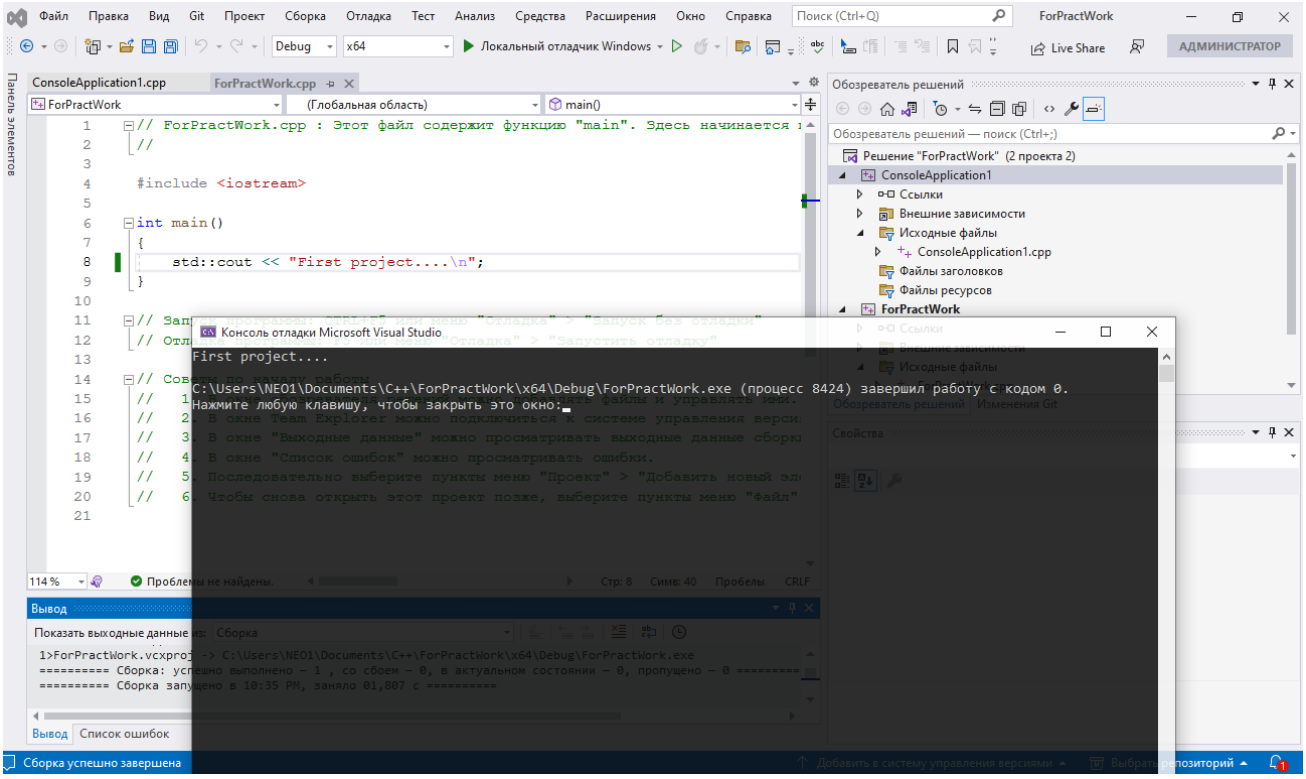
Для добавления нового проекта кликнете правой кнопкой мыши (ПКМ) по имени решения, выберите пункт Add, затем New Project:



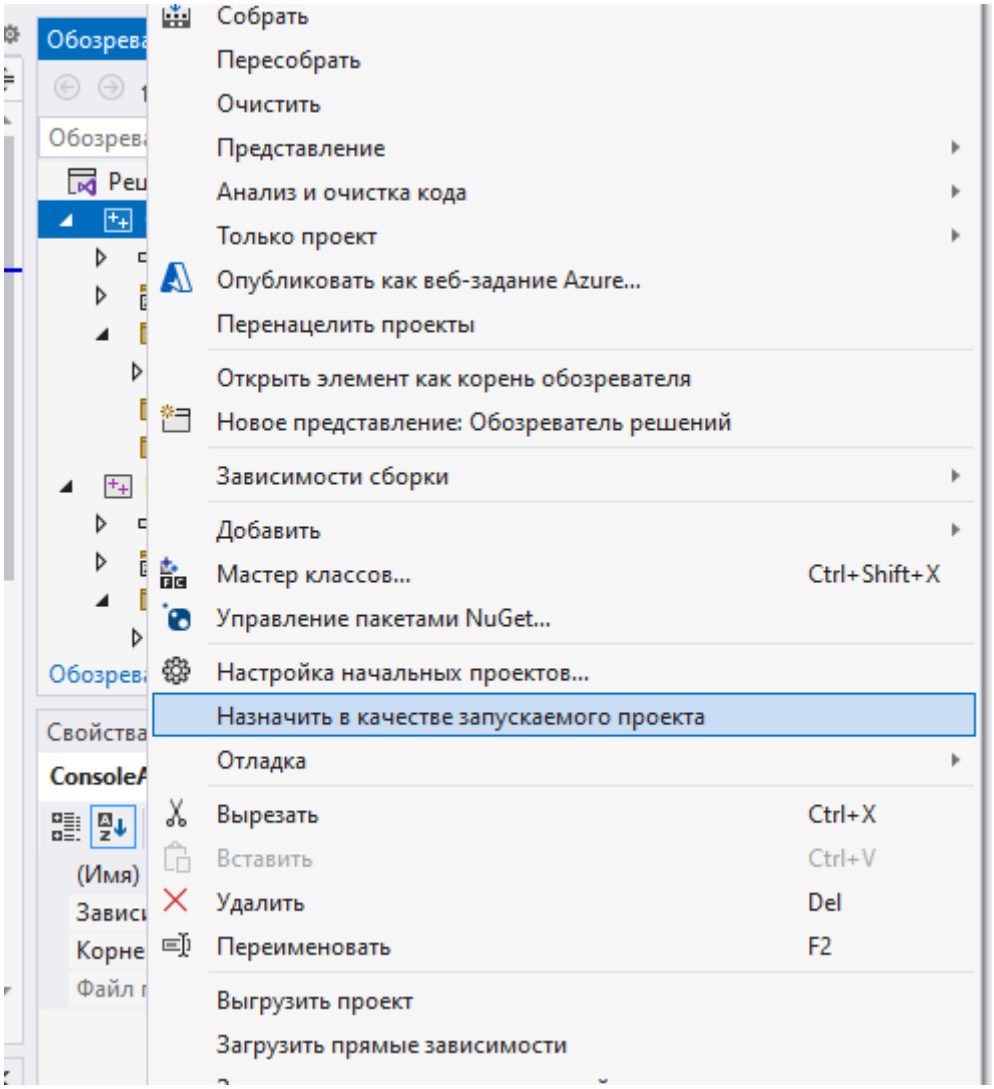
Выберите тип проекта и дайте ему название. Новый проект также отобразится в обозревателе решений. В моем случае это проект ConsoleApplication1 с файлом исходного кода ConsoleApplication1.cpp, где я и буду писать код. Предыдущий проект никуда не делся и при желании я могу снова к нему вернуться, открыв его .cpp файлик.



Но теперь при запуске моей программы (Ctrl+F5) у меня все еще запускается первый проект.



Все потому что теперь мне нужно назначить запускаемым мой второй проект. Для этого кликаю по проекту и выбираю пункт Set as setup project:



Теперь при запуске, будет выполнятся второй проект. Если я захочу вновь запустить первый, я просто назначу его запускаемым (Set as setup project).

Массивы

Массив представляет набор однотипных данных. Формальное определение массива выглядит следующим образом:

```
тип_переменной название_массива [длина_массива];
```

После типа переменной идет название массива, а затем в квадратных скобках его размер. Например, определим массив из 4 чисел:

```
int numbers[4];
```

Мы можем указать конкретные значения для всех элементов массива:

```
int numbers[4] = {1, 2, 3, 4};
```

В данном случае в памяти выделяется некоторая область из четырех ячеек по 4 байта (размер типа int), где каждая ячейка содержит определенный элемент массива:

numbers[0]	numbers[1]	numbers[2]	numbers[3]
1	2	3	4

Если значений в инициализаторе меньше, чем элементов в массиве, то значения передаются первым элементам, а остальные получают нулевые значения:

```
int numbers[4] = {1, 2};   //  {1, 2, 0, 0}
```

Если значений в инициализаторе больше, чем элементов в массиве, то при компиляции возникнет ошибка:

```
int numbers[4] = {1, 2, 3, 4, 5, 6};   // ! Ошибка
```

Здесь массив имеет размер 4, однако ему передается 6 значений.

Если размер массива не указан явно, то он выводится из количества переданных значений:

```
int numbers[] {1, 2, 3, 4, 5, 6};
```

Индексы. Получение и изменение элементов массива

После определения массива мы можем обратиться к его отдельным элементам по индексу. Индексы начинаются с нуля, поэтому для обращения к первому элементу необходимо использовать индекс 0. Обратившись к элементу по индексу, мы можем получить его значение, либо изменить его. Например, получим второй элемент (индекс 1):

```
int n = numbers[1];
```

Изменение значения второго элемента:

```
numbers[1] = 123;
```

Например, получим и изменим значения элементов:

```
int numbers[4] = {1,2,3,4};
    int first = numbers[0];     // получаем первый элемент
    cout << first << endl;    // 1
    numbers[0] = 34;                    / изменяем значение элемента
    cout << numbers[0] << endl; // 34
```

При обращении по индексу следует учитывать, что мы не можем обратиться по несуществующему индексу. Так, если в массиве 4 элемента, то мы можем использовать индексы с 0 до 3 для обращения к его элементам. Использование любого другого индекса приведет к ошибке:

```
int numbers[4] = {1,2,3,4};
int forth = numbers[4];     // !Ошибка - в массиве только 4 элемента
```

Перебор массивов

Используя циклы, можно пробежаться по всему массиву и через индексы обратиться к его элементам:

```
    const int n = 4;
    int numbers[n] = {11, 12, 13, 14};
    for(int i=0; i < n; i++)
    {
        cout << numbers[i] << endl;
    }
```

Чтобы пройтись по массиву в цикле, надо знать его длину. Здесь длина задана константой n.вначале надо найти длину массива. И в цикле for перебираем все элементы, пока счетчик i не станет равным длине массива. В итоге на консоль будут выведены все элементы массива:

```
11
12
13
14
```

Другой пример - вычислим сумму элементов массива:

```
int numbers[] = {1, 2, 3, 4};
int sum = 0;
for (int i = 0; i < size(numbers); i++){
    sum += numbers
}
cout << "Sum: " << sum << endl;   // Sum: 10
```

Ввод значений массива с консоли

Подобно тому, как вводятся данные для отдельных переменных, можно вводить значения для отдельных элементов массива. Например, пусть пользователь вводит значения числового массива:

```
int numbers[5];
for (int i = 0; i < 5; i++){
    cout<<”Введите элемент массива: ”;
    cin>>numbers[i];
}
```