Занятие 1. Настройка рабочего окружения. «Hello world!»

Цели занятия:

- Установка и настройка программы Visual Studio Community
- Знакомство с ключевыми особенностями языка и областью применения
- Знакомство с средой разработки
- Создание первой программы «Hello world!»

Результат: настроили рабочее окружение, познакомились с средой разработки (IDE), обсудили цели и задачи программирования

Введение

На первом занятии необходимо настроить рабочее окружение.

Что такое рабочее окружение программиста? Рабочее окружение программиста состоит из всех необходимых программ и технологий, которые используются в повседневной жизни и конкретных проектах.

В нашем курсе будет использоваться рабочее окружение (далее - стенд), состоящий из таких программ:

- Microsoft Visual Studio 2022
- Microsoft Windows Forms
- Git в связке с GitHub

Visual Studio – мощная среда разработки, которая поддерживает множество языков. Основное преимущество перед другими IDE – русификация и простота настройки.

WinForms – технология в составе Visual Studio, которая помогает создавать графический дизайн приложений и интегрировать код в это приложение.

Git – система контроля версий, о ней позже.

Установка и настройка MS Visual Studio

Если программа уже скачана и настроена, можно пропустить этот пункт, просто рассказав о программах.

Для скачивания установщика переходим по ссылке:

https://visualstudio.microsoft.com/ru/free-developer-offers/

На сайте листаем вниз и выбираем Visual Studio Community (рис. 1), выбираем версию для соответствующей операционной системы

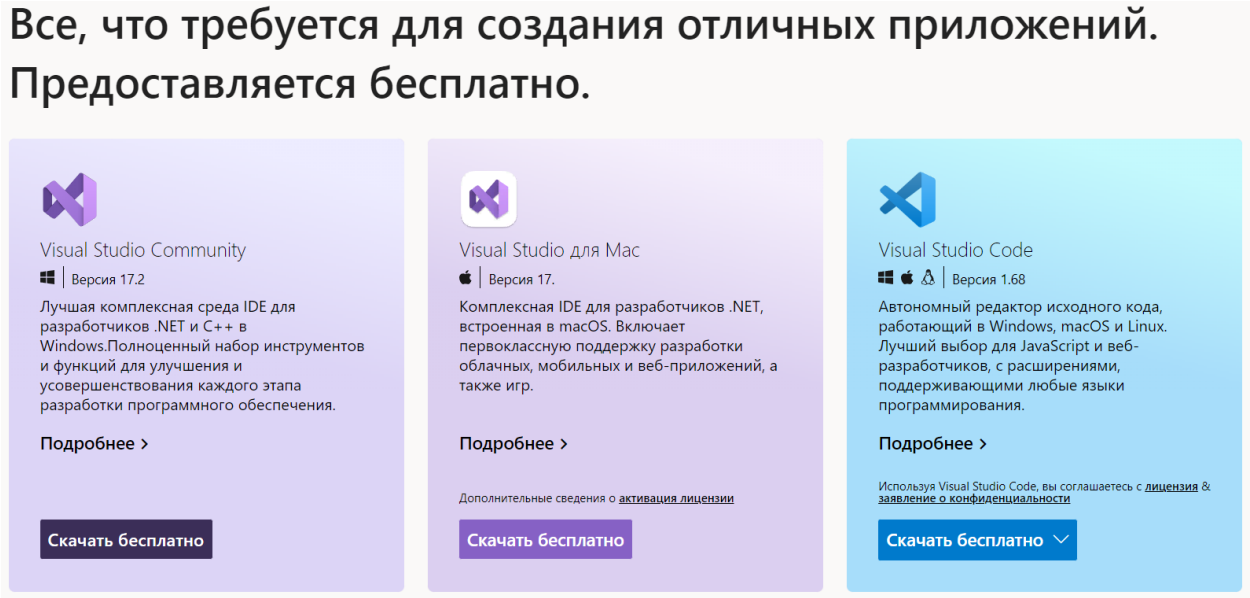

рис. 1

В окне доступно выбираем Visual Studio Community, и начинаем выбирать пакеты для разработки классических приложений на C++, как на рис.2:

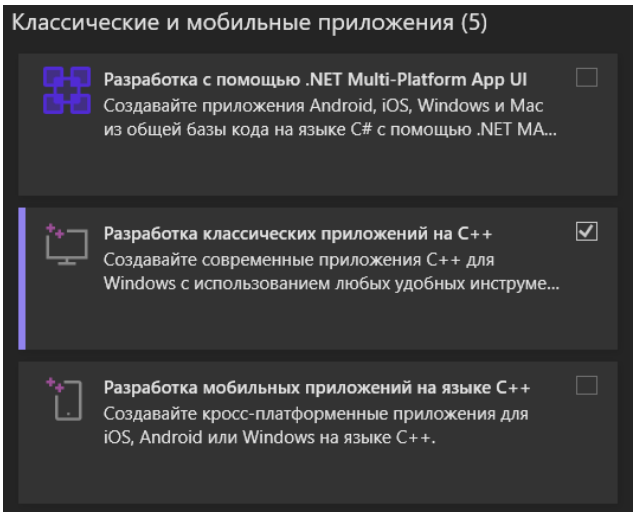

рис. 2

После того, как выбрали всё необходимое, нажимаем кнопку «Установка при скачивании», затем «Установить» (рис. 3).

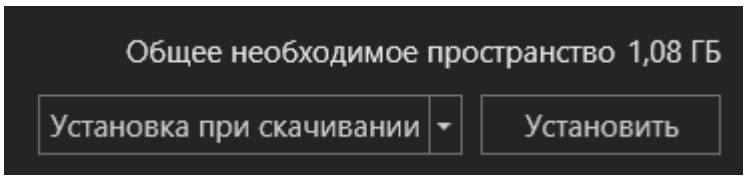

рис. 3

Дожидаемся окончания установки.

Интерфейс IDE

После того, как установка завершена, можем открыть IDE через встроенное средство или из меню «Пуск».

Когда мы отрываем среду разработки, нас встречает интерфейс-хаб (рис. 4).

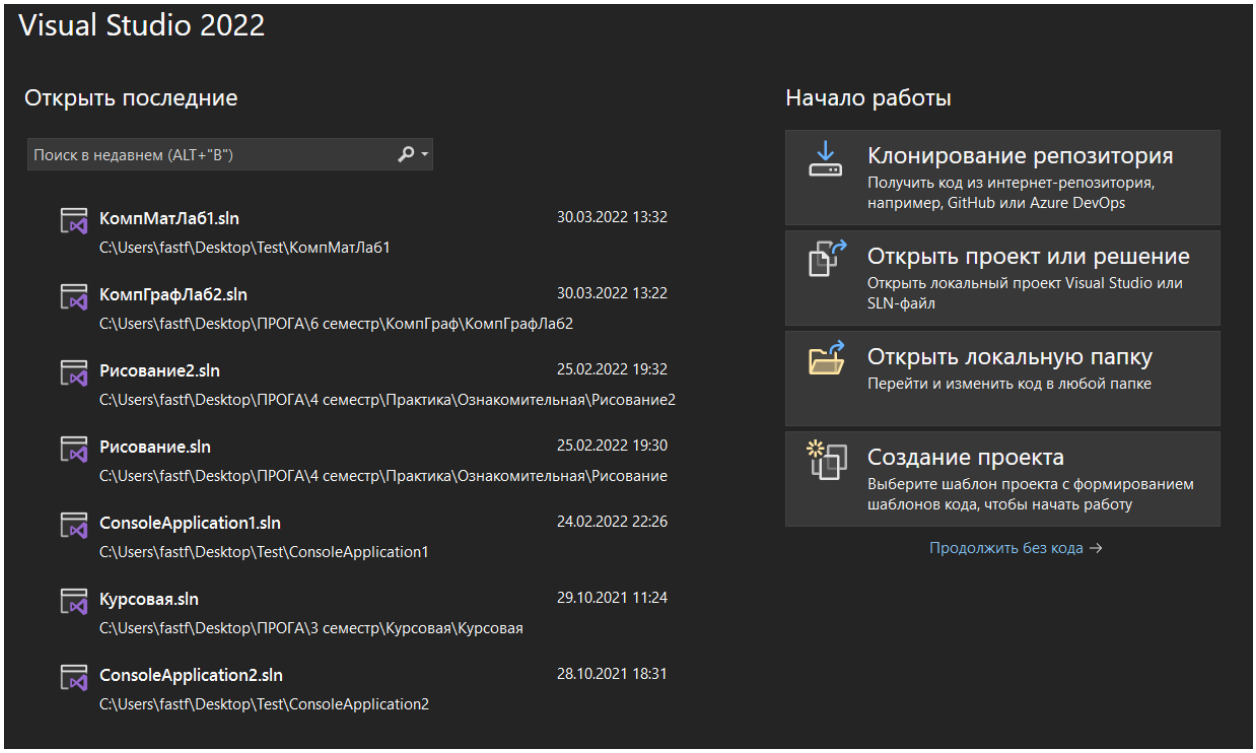

рис. 4

Слева мы можем увидеть созданные проекты, а справа опции «Начало работы». Среди этих опций нам больше всего интересны:

- «Открыть проект или решение», для открытия уже готовых проектов
- «Создание проекта», для создания собственного проекта

При нажатии на кнопку «Создание проекта», откроется интерфейс выбора проекта (рис. 5).

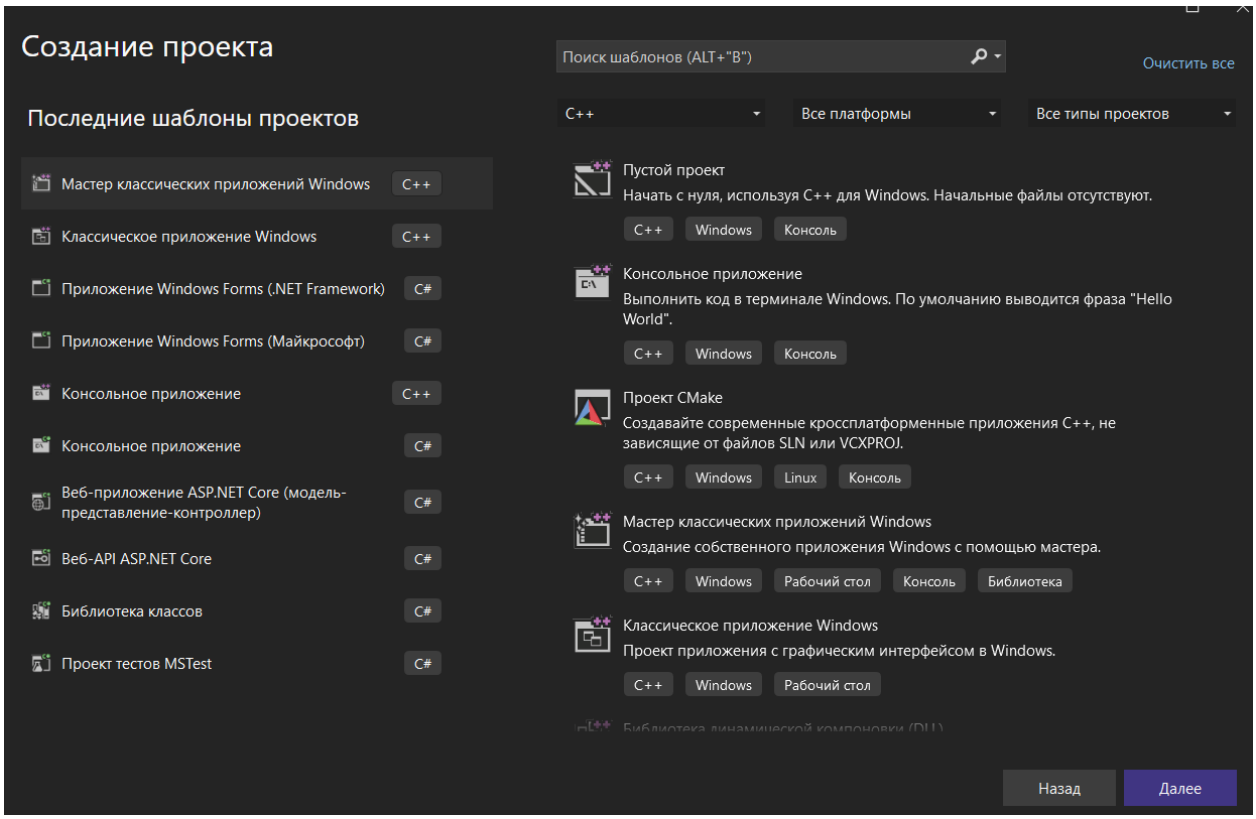

рис. 5

В этом интерфейсе мы можем выбрать язык, технологии и другие опции под конкретную необходимость. В нашем курсе будет в основном использоваться шаблон «Консольное приложение».

При нажатии на выбранный шаблон мы оказываемся в окне настройки проекта (рис. 6). Нас интересует имя проекта, которое должно быть уникальным и понятным для ученика, учителя и любого другого программиста.

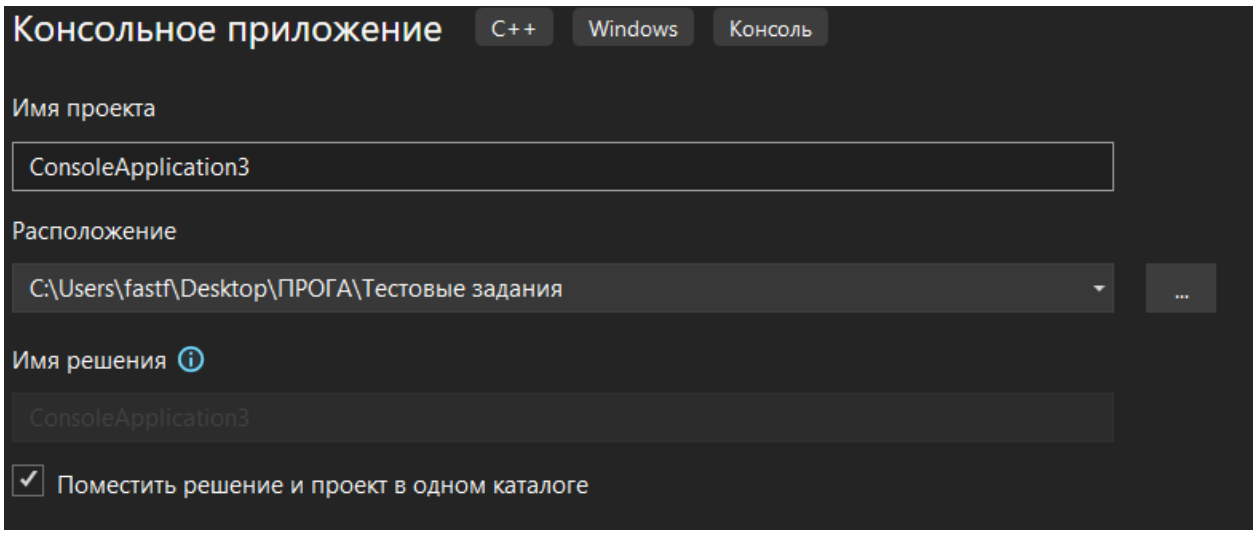

рис. 6

Здесь необходимо помочь правильно выбрать расположение и название проекта.

Нажимаем далее и оказываемся в основной рабочей области среды.

Слева у нас находится «Обозреватель решений» (рис. 7), в котором мы видим структуру подключенных к проекту данных. Всё, что мы создадим и напишем, можно найти в обозревателе.

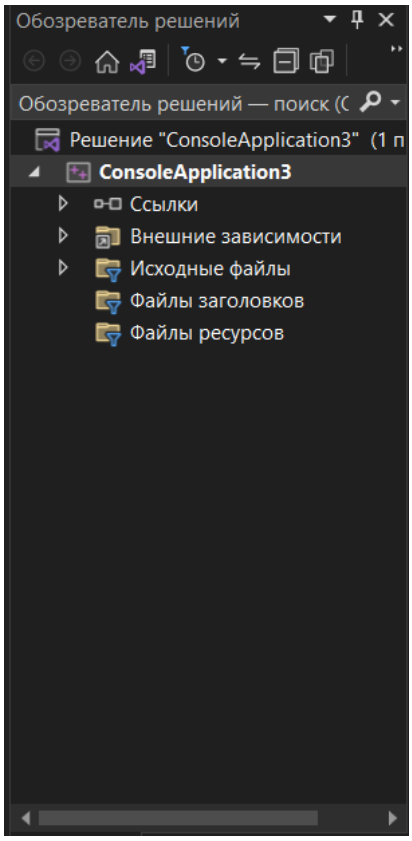

рис. 7

В центре экрана (рис. 8) расположен текстовый редактор кода, который позволяет писать новый код. Обратим внимание, что окно редактирования содержит вкладки (сверху). В этих вкладках мы можем параллельно писать код в разных файлах, а также смотреть результаты работы кода (если выводим результаты в файл).

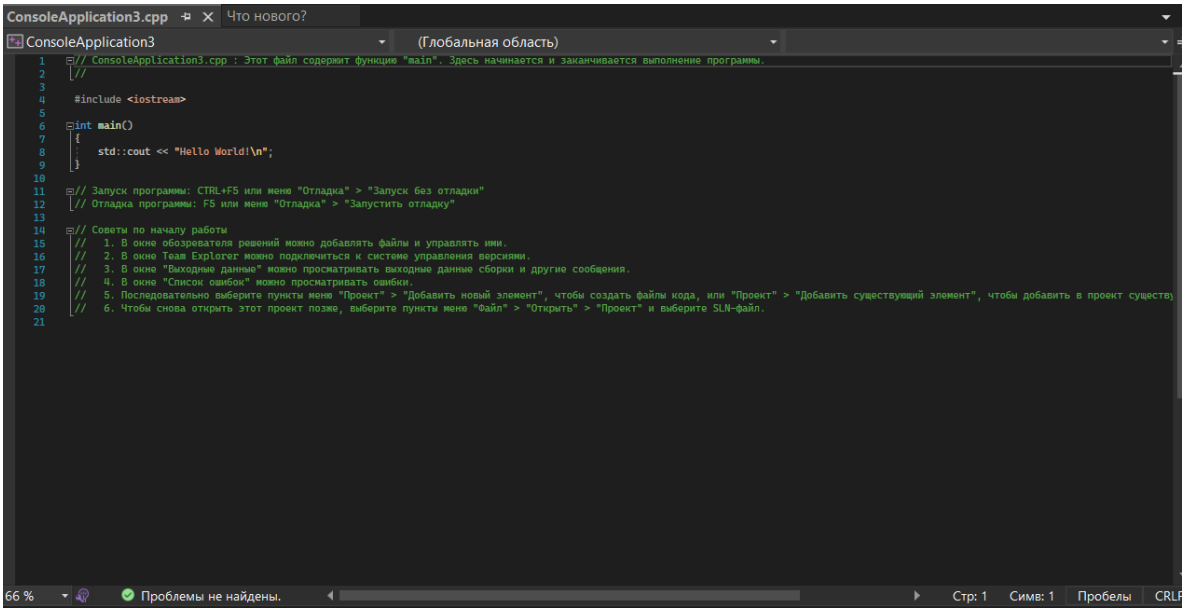

рис. 8

Внизу экрана мы видим окно вывода (рис. 9). Здесь мы видим техническую информацию, ошибки кода, которые среда отследила в реальном времени (здесь и сейчас), а также ошибки найденные в другие этапы работы программы.

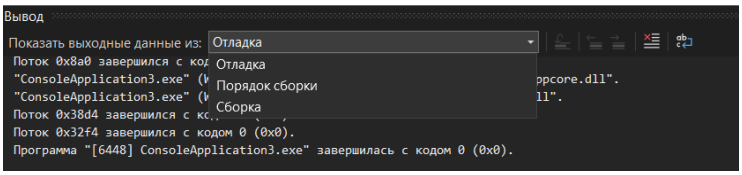

рис. 9

Сверху, ровно над обозревателем решений и рабочим окном, находится панель (рис. 10), в которой мы можем выбрать режим запуска кода (дебаг и релиз), сохранить проект или отдельные открытые файлы, запустить проект и изучить результаты.

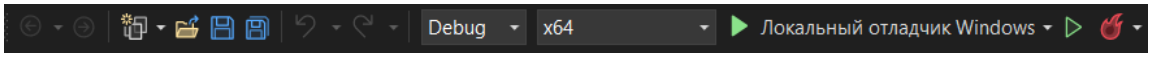

рис. 10

В панели вкладок, нас интересуют вкладки «Файл», «Правки», «Вид», «Сборка» и «Отладка» (рис. 11 - 16).

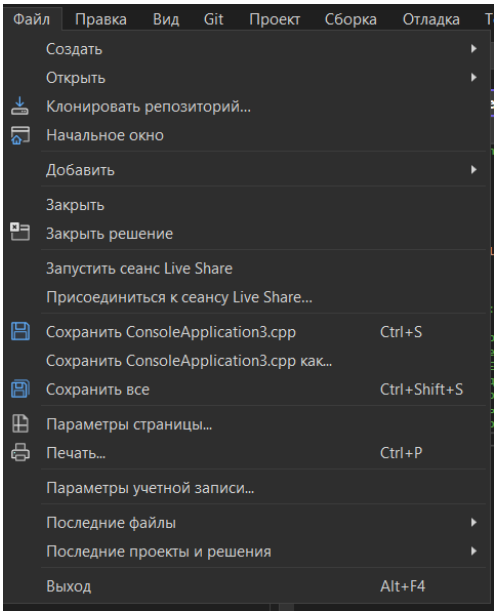

рис. 11

Во вкладке «Файлы» мы можем создать новый проект, открыть файл или проект, сохранить файлы и много других полезных функций. Нас интересует в основном «Создать», «Открыть», «Сохранить всё».

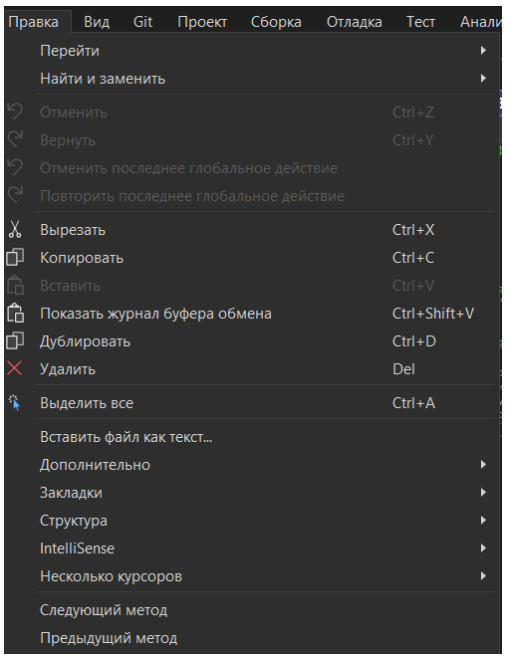

рис. 12

Во вкладке «Правка» мы можем отменять изменения, манипулировать информацией (копировать. Вставить, вырезать и т.д.). Остальные пункты нам не так интересны.

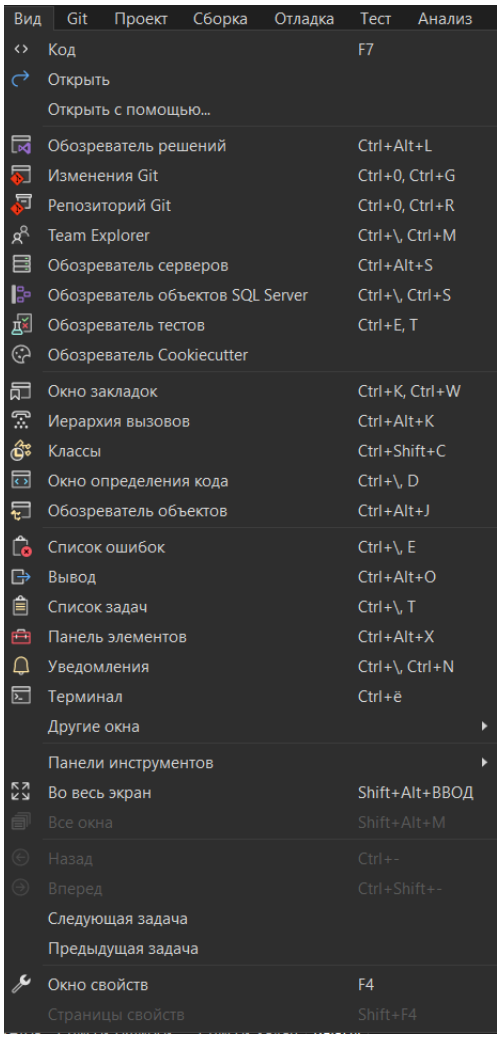

рис. 13

Во вкладке «Вид» мы можем открыть дополнительные окна, или открыть случайно закрытые окна, которые использовали ранее.

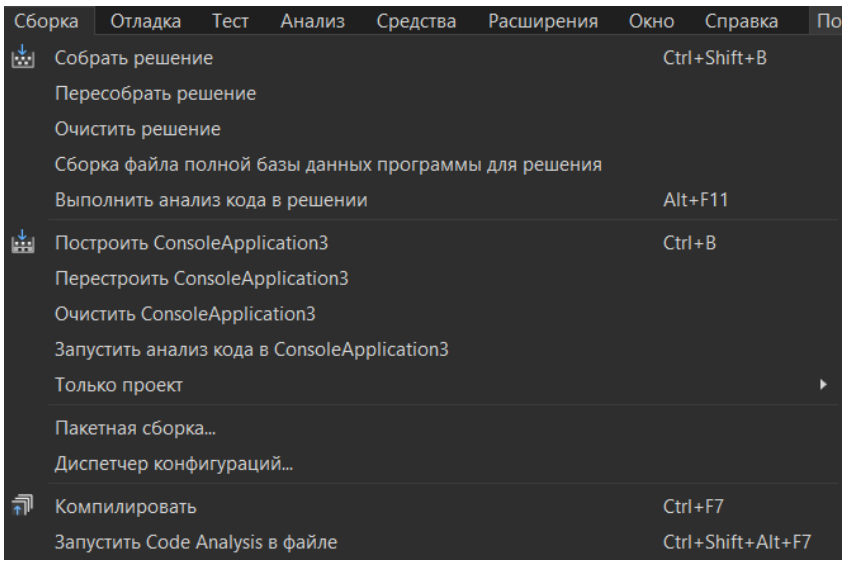

рис. 14

Во вкладке «Сборка» нам интересны только пункты «Собрать решение» и «Пересобрать решение». С помощью этих кнопок мы можем обновить сторонние данные, осуществить сборку приложения с новым кодом или с новыми подключенными библиотеками или технологиями.

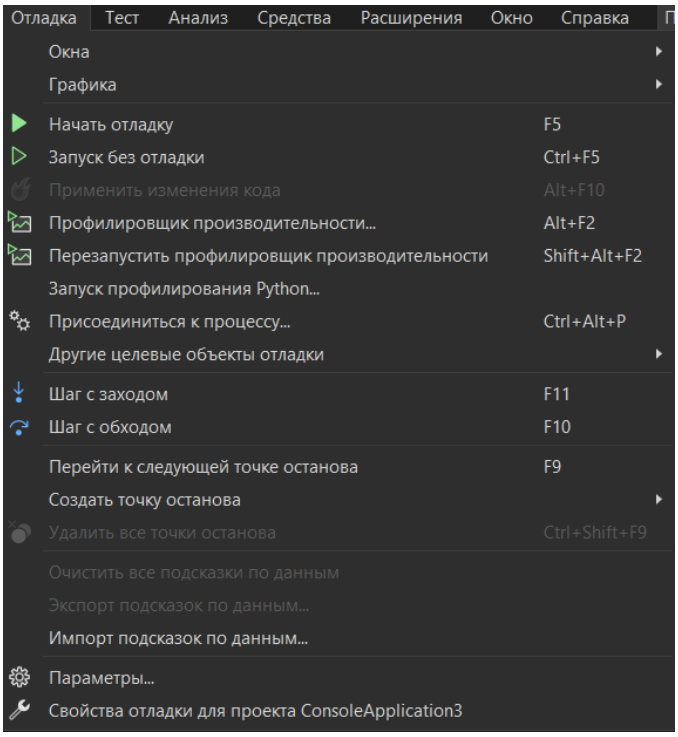

рис. 15

Вкладка «Отладка» является одной из важнейших в жизни программиста, ведь она позволяет запустить одноимённый процесс отладки, с помощью которого можно вручную протестировать приложение, посмотреть внутренние данные программы, изучить ошибки и многое другое. Про отладку подробнее мы будем говорить на протяжении всего курса.

Первое приложение, ввод и вывод информации

Для создания первого приложения необходимо познакомиться с операторами cout и cin. Дословно, cout – console out (вывод в консоль), cin – console in (ввод из консоли).

Для демонстрации работы приложения можно запустить уже созданный автоматически код (вывод «Hello world!»).

Также можно запустить самый типичный код:

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World!\n";
    return 0;
}
```

Для демонстрации ввода данных предлагаем поработать с таким кодом:

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    //это когда-нибудь потом
    setlocale(LC_ALL, "");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    string name;
    cout << "Введи своё имя: ";
    cin >> name;
    cout << "Привет " << name << "!\n";
    return 0;
}
```

Как объяснить, какой оператор поставить << или >> ?

Проще всего объяснить cout и cin как потоки информации в консоли. Поток больше всего напоминает реку, по которой течёт вода. Если мы хотим взять воду из реки, или получить данные из консоли, нам необходимо с cin использовать >> , который направлен от потока.

Аналогично с cout, мы ходим добавить воды в реки, или вывести информацию в консоль, мы будем использовать оператор <<, который направлен к потоку.

Можно просто запомнить, что с чем использовать, но в любом случае механику потоков придётся объяснять для работы с вводом/выводом в файл.

Домашнее задание

1. Установить Visual Studio на свой компьютер
2. Изучить в домашней обстановке интерфейс
3. Попробовать модифицировать код из задания. Например, написать диалог с помощью последовательностей cin, cout. Можно отправить на проверку преподавателю или принести на флешке код.