Федеральное государственное бюджетное образовательное учреждение высшего образования «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича».

**Факультет Информационных систем и технологий**

**Кафедра Информатики и компьютерного дизайна**

**Лабораторная работа №6**

Разработка макета программного проекта

****

**Автор работы:**

Сухоруков А.А.

**Научный руководитель:**

Федоров Д.Ю.

Санкт-Петербург, 2020

1. **Постановка задачи.**

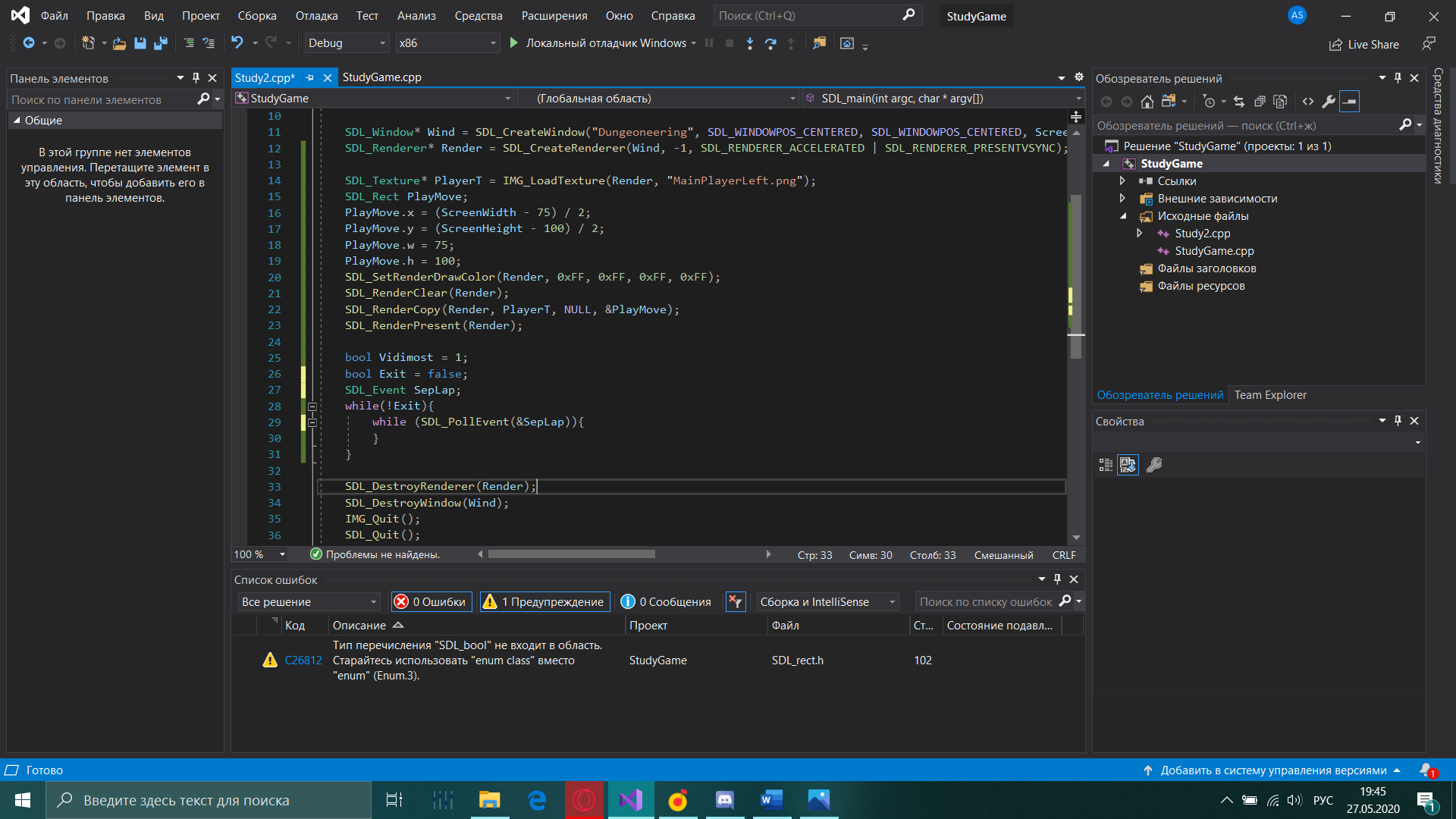
**1.** Написать алгоритм программы работы с событиями **1.2** События с компьютерной мышки  
 **1.3** События с клавиатуры  
**2.** Работая с событиями, научить героя игры перемещаться

**2. Формализация задачи.**

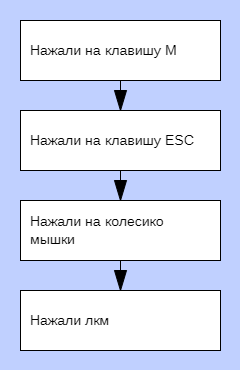
Главной целью будет изучить событийно-ориентированное программирование, отсюда и появляются наши задачи.  
**1.** Написать алгоритм программы работы с событиями **1.2** События с компьютерной мышки  
 Нам нужно будет воспользоваться новым для нас классом, а также   
 прописать непрерывный цикл проверки на события нажатия клавиш  
 мышки  
 **1.3** События с клавиатуры  
 Далее нам нужно будет дополнить наш цикл проверками на нажатия  
 клавиш на клавиатуре, но для этого нам понадобится уже другой  
 класс  
**2.** Работая с событиями, научить героя игры перемещаться  
 И главной целью этой работы будет научить главного героя  
 перемещаться по экрану при нажатии клавиатуры, а также для  
 взаимодействия с мышкой мы введем появление/исчезновение   
 героя при нажатии лкм и пкм

1. Написать алгоритм программы работы с событиями

Для начала нам понадобится основа основ, а именно переменная **Exit** класса **boolean**, которая будет повторять цикл до тех пор, пока она не станет равна true.   
Далее мы создаем переменную **SepLap** (ЦепкиеЛапы) класса **SDL\_Event**, данная переменная как раз и будет содержать в себе событие.  
Задем цикл (Пока Exit не равен false) и цикл с **SDL\_PollEvent** в условии.   
**SDL\_PollEvent** возвращает 0, если события не было и 1, если таковое было. И так как мы указали &SepLap, то при наличии события он будет записывать его в нашу переменную событий.



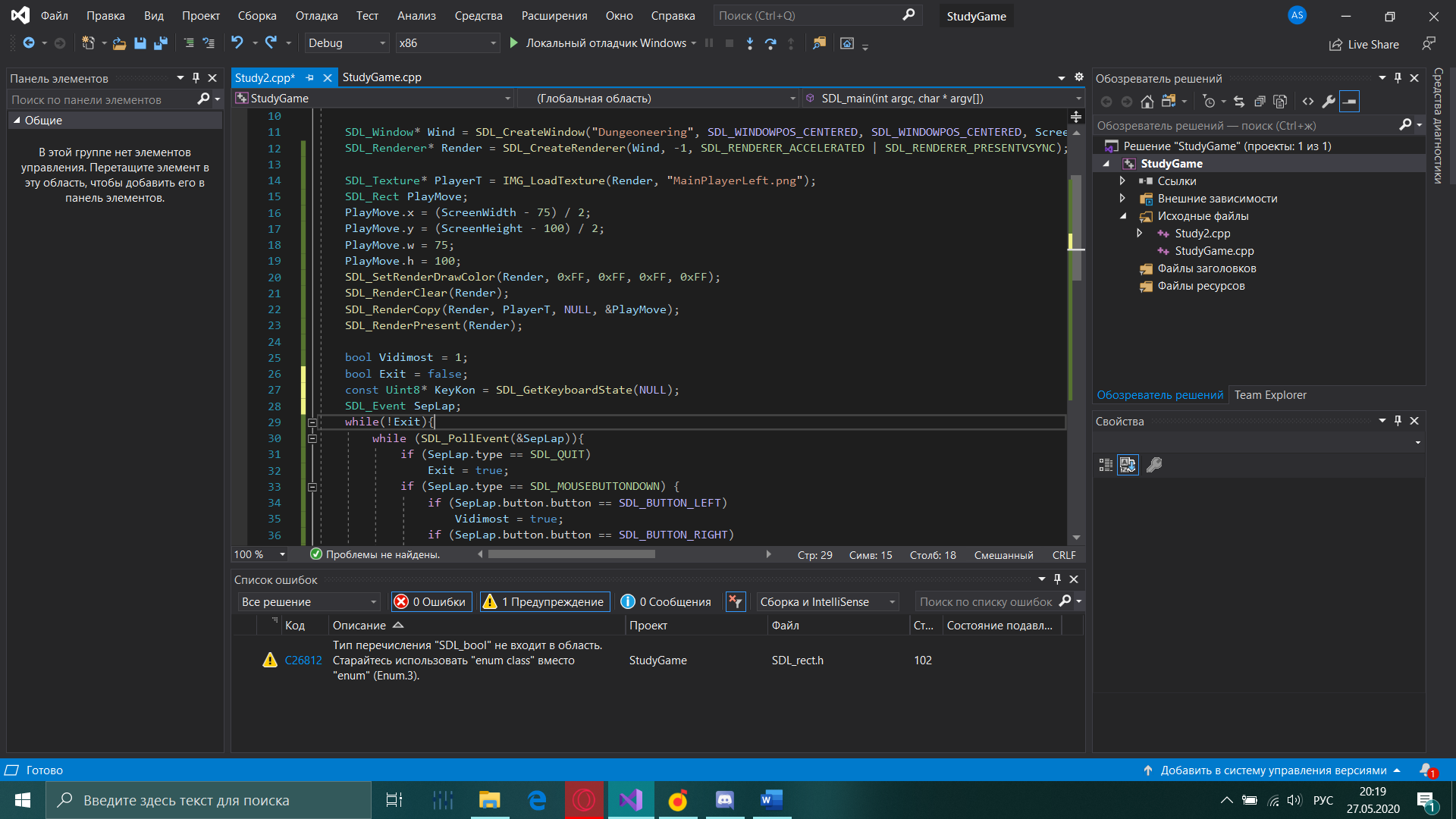
Важно понимать как устроены события в SDL

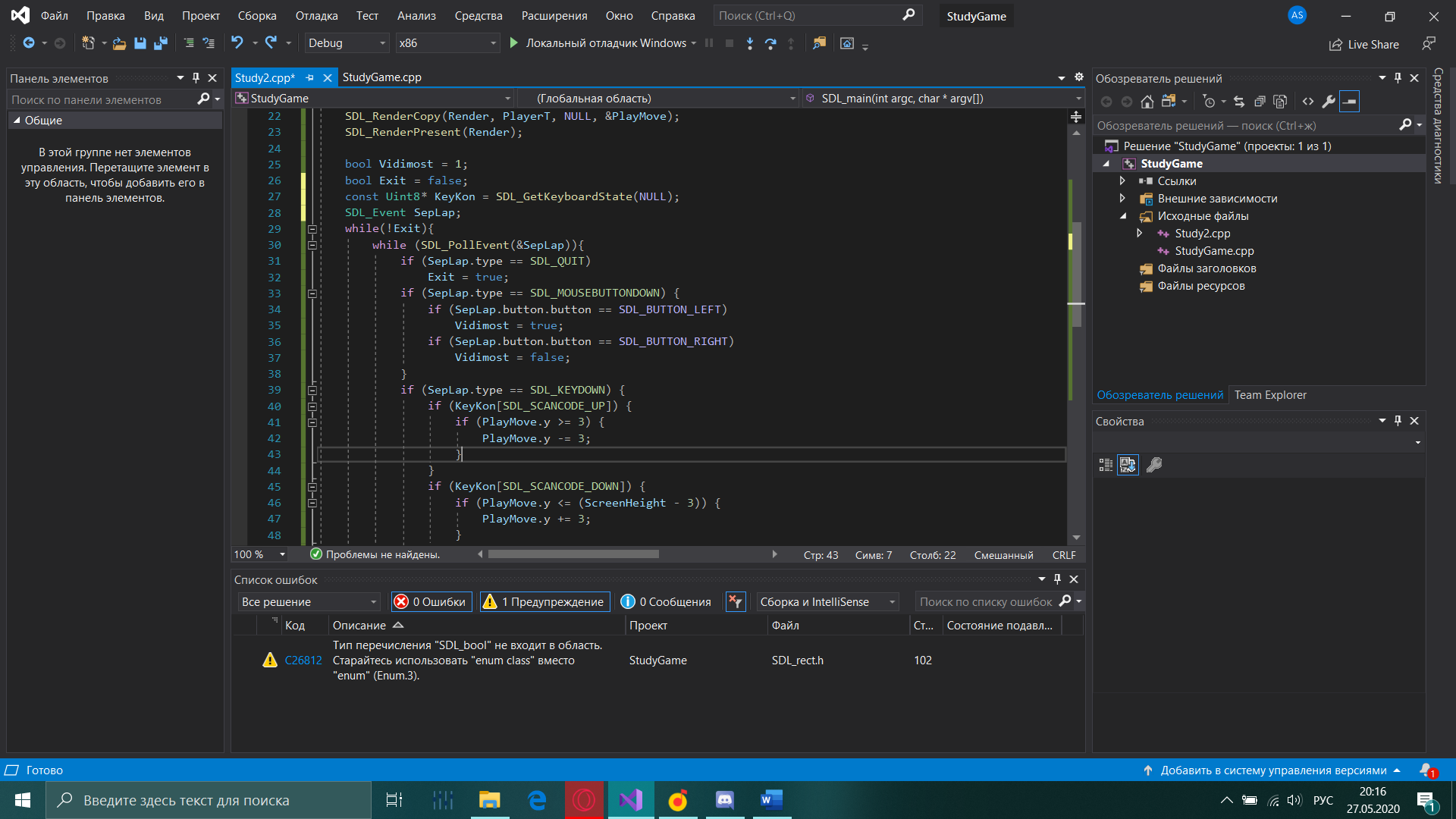
В начале мы нажали на клавишу M, потом на Escape, после на колесико мышки и последним мы нажали лкм. Историю событий я изображу графиком  


**SDL\_PollEvent** будет брать вначале те события, что мы делали первым при открытом окне приложения. То есть если мы как-то успели нажать на все до получения результата в программе, то мы будем видеть как программа отвечает на все события от самого старого к самому новому в истории событий. Также **SDL\_PollEven** удаляет событие после взаимодействия с ним, потому второе по очереди событие становится первым (самым старшим)

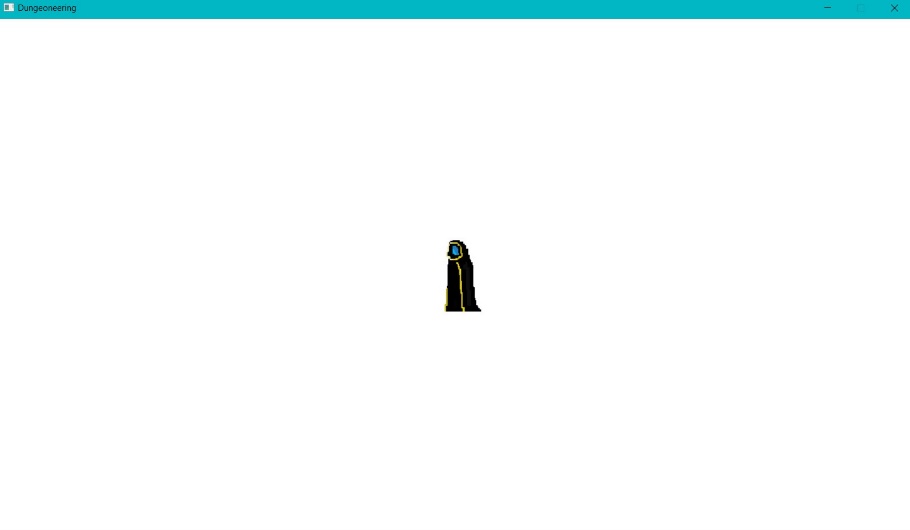
* 1. События с компьютерной мышки

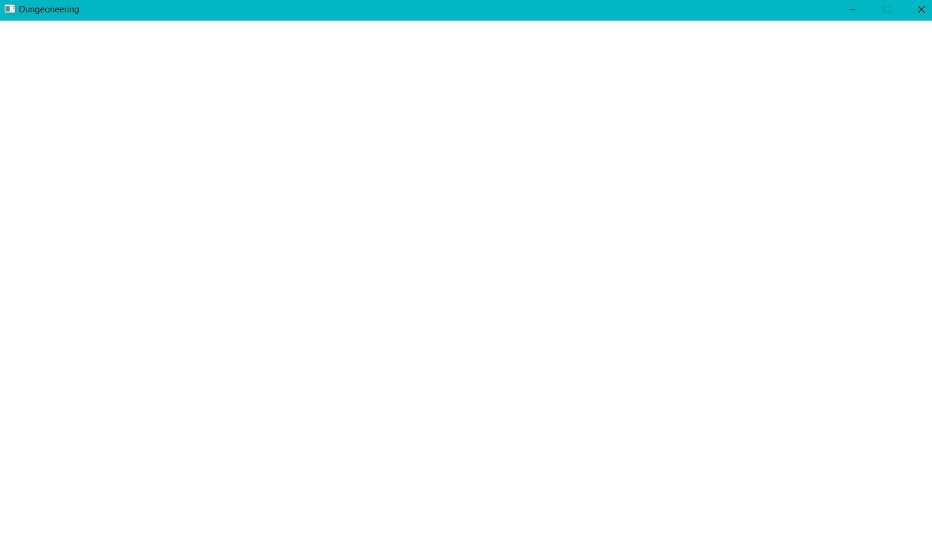
Для начала перед циклом введем переменную **Vidimost** также типа **bool**, она нам будет нужна для нашего примера взаимодействия с компьютерной мышкой. Прежде всего пишем проверку на то, что было нажато что-то именно на мышке, это **SDL\_MOUSEBUTTONDOWN**. Далее мы внутри цикла вводим две новые проверки: на нажатие **лкм** и **пкм**, первая будет делать нашего героя (нарисованного вручную) видимым, вторая невидимым

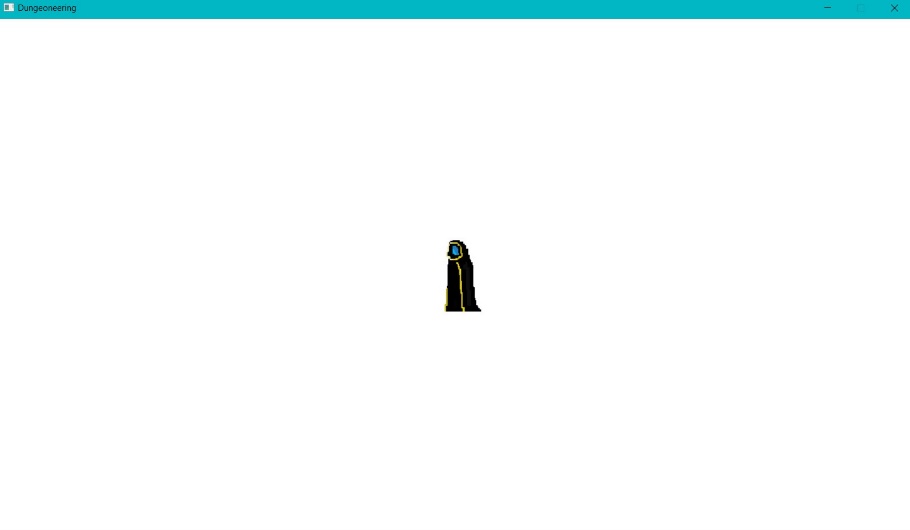




Также в конце цикла событий пишем очистку и повторную прорисовку рендера с уже новыми параметрами, но это мы рассмотрим немного позже. Сейчас же взглянем на то, что у нас получилось с действиями мышки

Открытие программы  


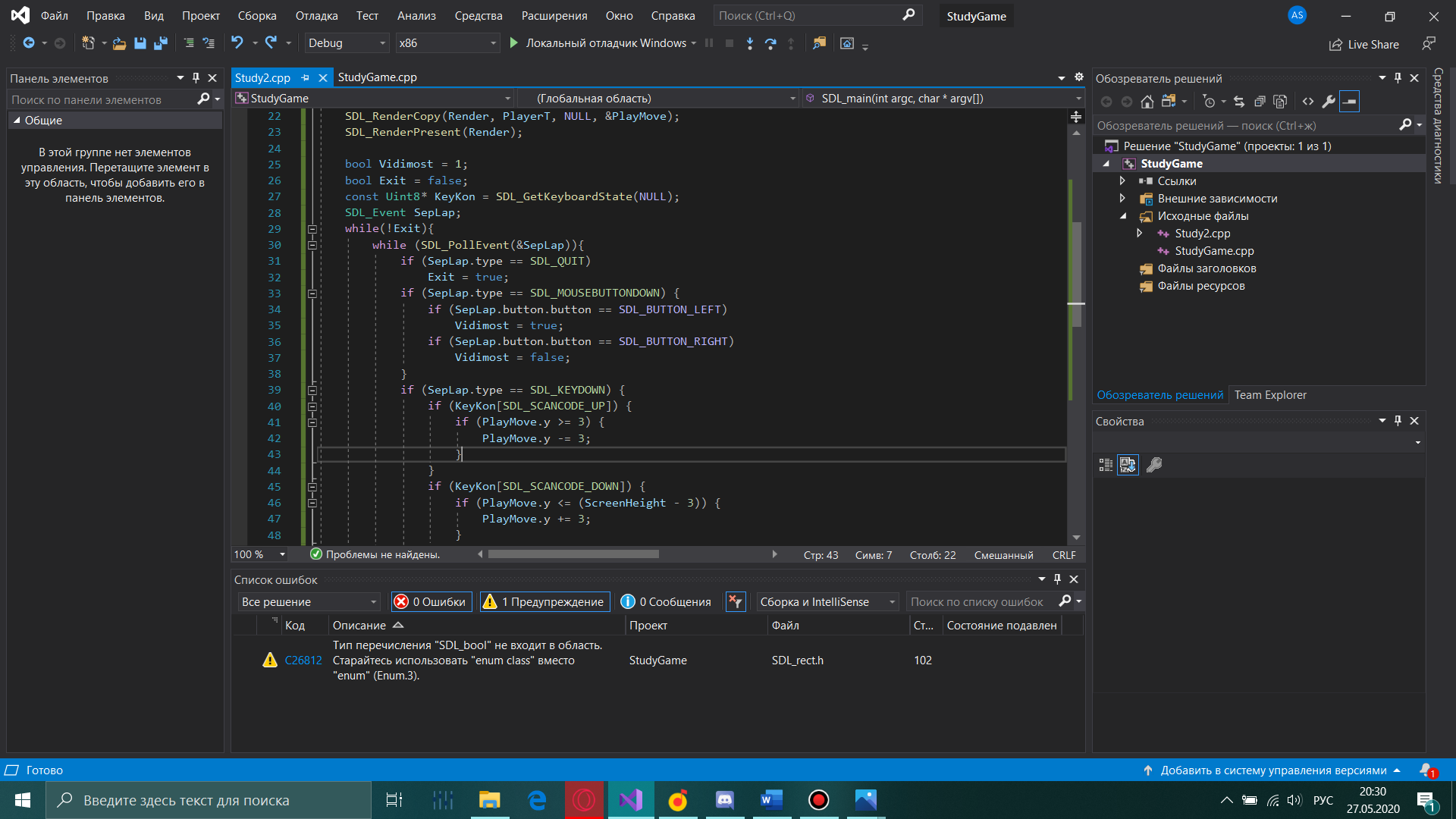
Нажатие пкм  


Нажатие лкм  


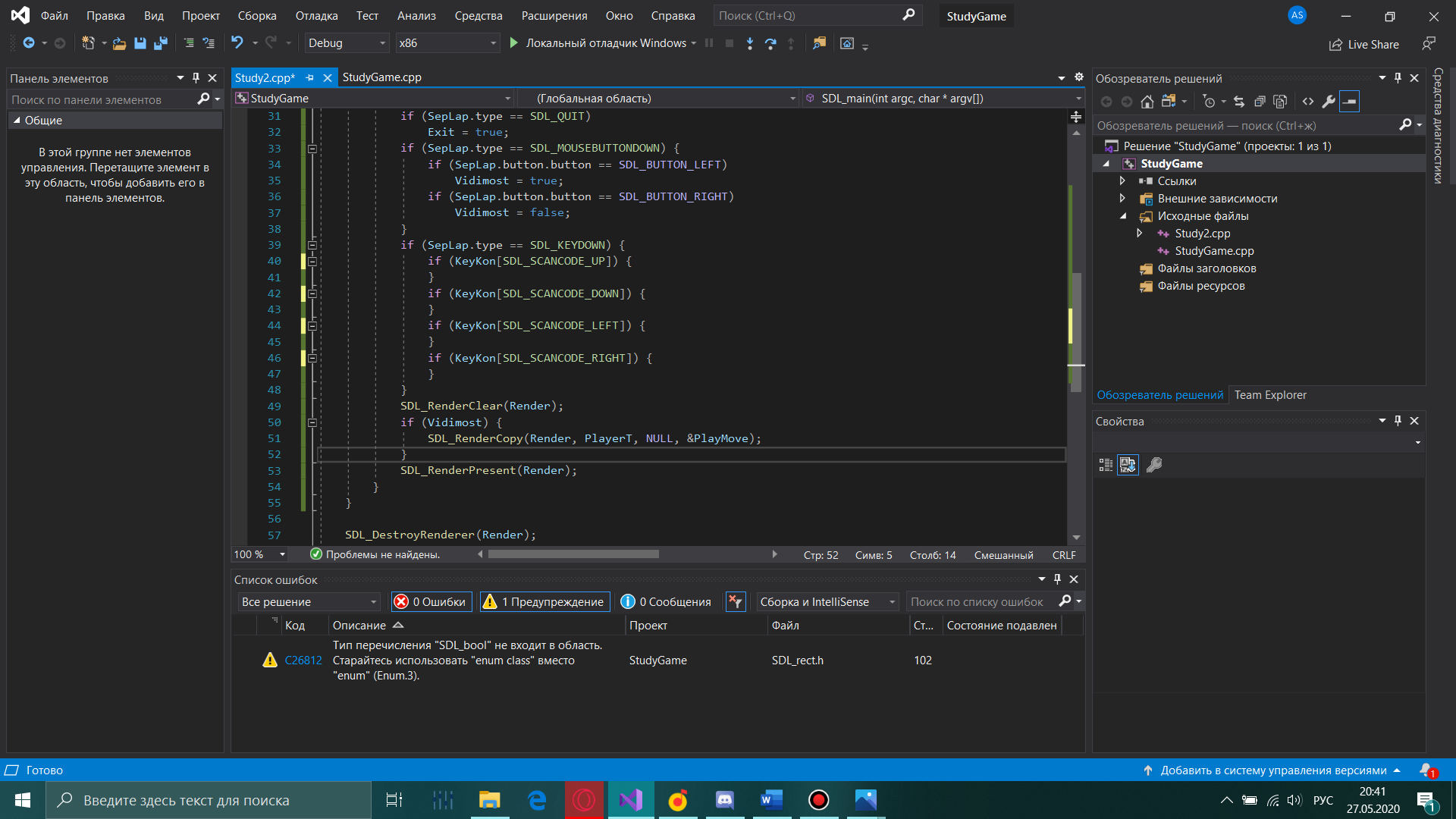
Как видим, наше начальное взаимодействие работает просто отлично

* 1. События с клавиатуры

Для начала нам понадобится новая переменная **KeyKon** класса **const Uint8\***, которая будет хранить нажатую клавишу по той же схеме событий. Для ее инициализации нам понадобится функция **SDL\_GetKeyboardState**, которая как раз и помещает в нее ссылку на самое старшое событие (при NULL)



Теперь мы внутри нашего цикла проверяем на то, что клавиша была нажата, а это тип событий **SDL\_KEYDOWN**. Далее мы проверяем уже нашу переменную **KeyKon**, которая позволит нам узнать какая клавиша была нажата.   
**UP, DOWN, LEFT, RIGHT** – это стрелки на вашей клавиатуре

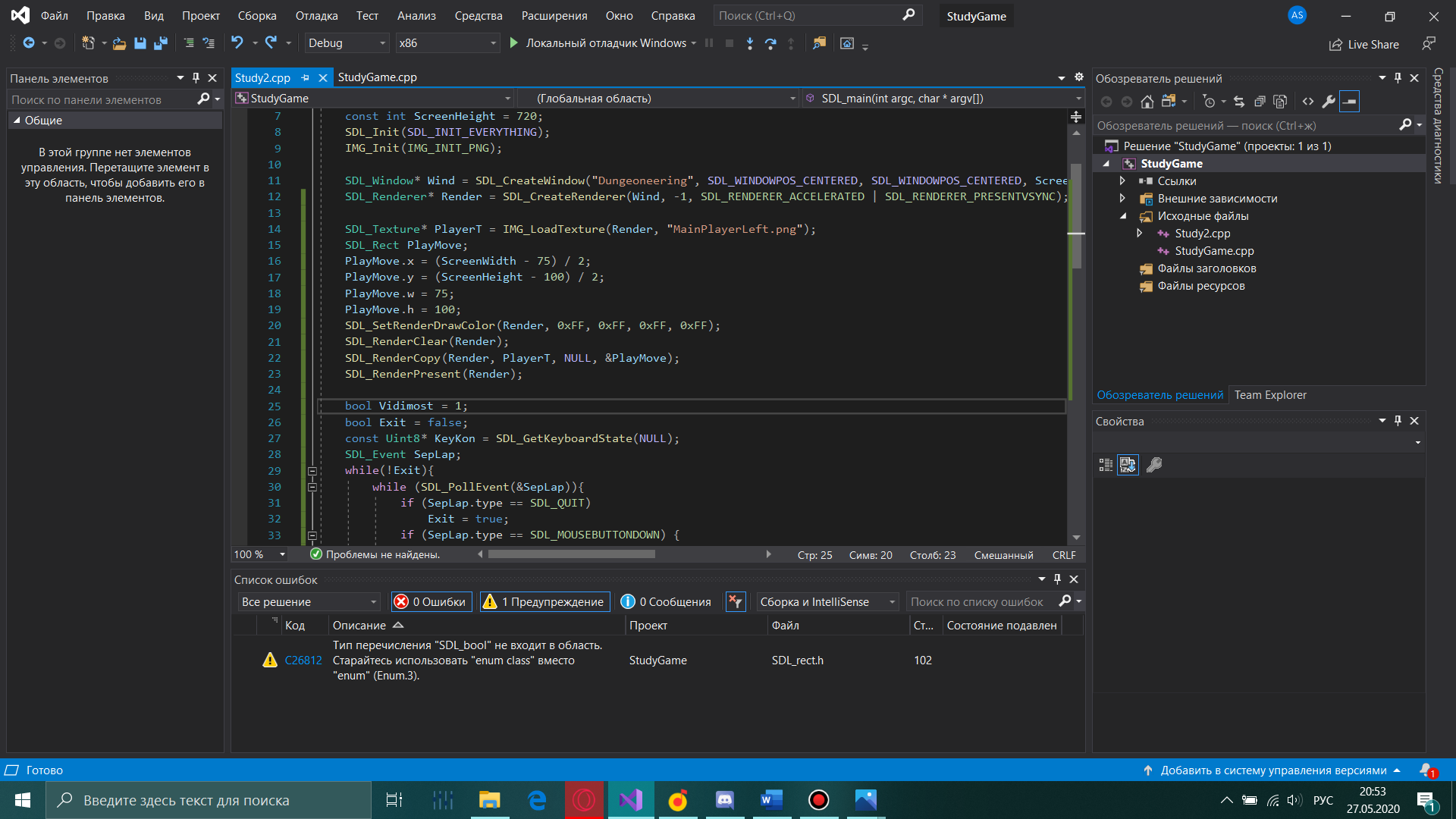


1. Работая с событиями, научить героя игры перемещаться

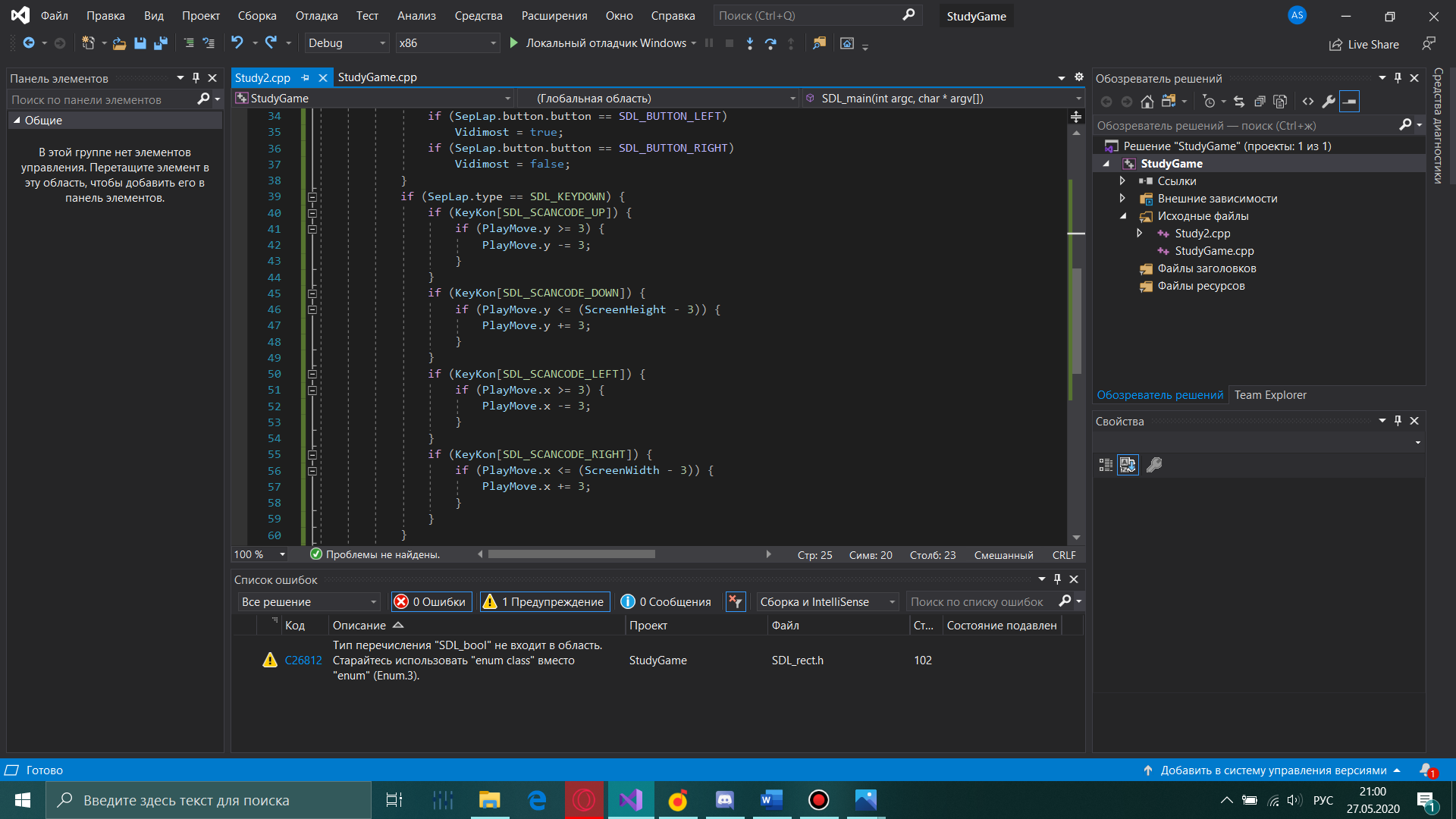
Он уже умеет исчезать и появляться, но на этом же игру не сделаешь? Потому нам надо обучить его передвигаться по нашему экрану, а для этого нам нужно хранить **4 вещи**:

1. Координата героя по X
2. Координата героя по Y
3. Ширина текстуры героя
4. Высота текстуры героя

И с этим нам поможет такой класс библиотеки SDL как **SDL\_Rect**  
С ним можно очень удобно взаимодействовать для записи в него тех 4 элементов, так еще и передавать его прямиком в наши функции  
x,y - координаты, в данном случае я помещаю героя в центр экрана  
w, h - ширина и высота текстуры героя соответственно



Также нам надо дополнить наши ранее созданные проверки нажатия клавиш. Теперь при нажатии нужной клавишы герой будет двигаться в соответствующую сторону (зажатие клавишы будет восприниматься как множество нажатий)



И последним шагом продемонстрирую как работает видимость и как мы передаем новые координаты в рендер, стирая старое местоположение текстуры

