Федеральное агентство связи  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
 «Сибирский государственный университет  
телекоммуникаций и информатики»

Кафедра вычислительных систем

Курсовая работа

по дисциплине «Технологии разработки программного обеспечения»

на тему «Todo-лист»

Выполнил:

ст. гр. ИВ-223

Децюк Д. К.

Проверил:

ст. преподаватель Токмашева Е. И.

Содержание

[Введение и постановка задачи 3](#__RefHeading___Toc132_1766323042)

[Техническое задание 4](#__RefHeading___Toc134_1766323042)

[Описание выполненного проекта 5](#__RefHeading___Toc136_1766323042)

[Приложение. Текст программы 11](#__RefHeading___Toc140_1766323042)

# Введение и постановка задачи

**Цель работы:**

Целью данной работы является разработка программы для создания todo-листа, которая позволит пользователям удобно и эффективно планировать свои задачи и управлять ими. В рамках работы будет проведён анализ существующих решений, определены требования к программе и спроектирован её функционал. Кроме того, будут рассмотрены современные технологии разработки программного обеспечения и реализовано тестирование программы. В результате работы будет создана функциональная программа todo-лист, которая сможет помочь пользователям в повседневных задачах.

**Задание курсовой работы:**

* Написать техническое задание к проекту
* Составить план задач
* Создать функциональную программу todo-лист
* Покрыть тестами написанную программу
* Использовать CI
* Написать отчёт о проделанной работе

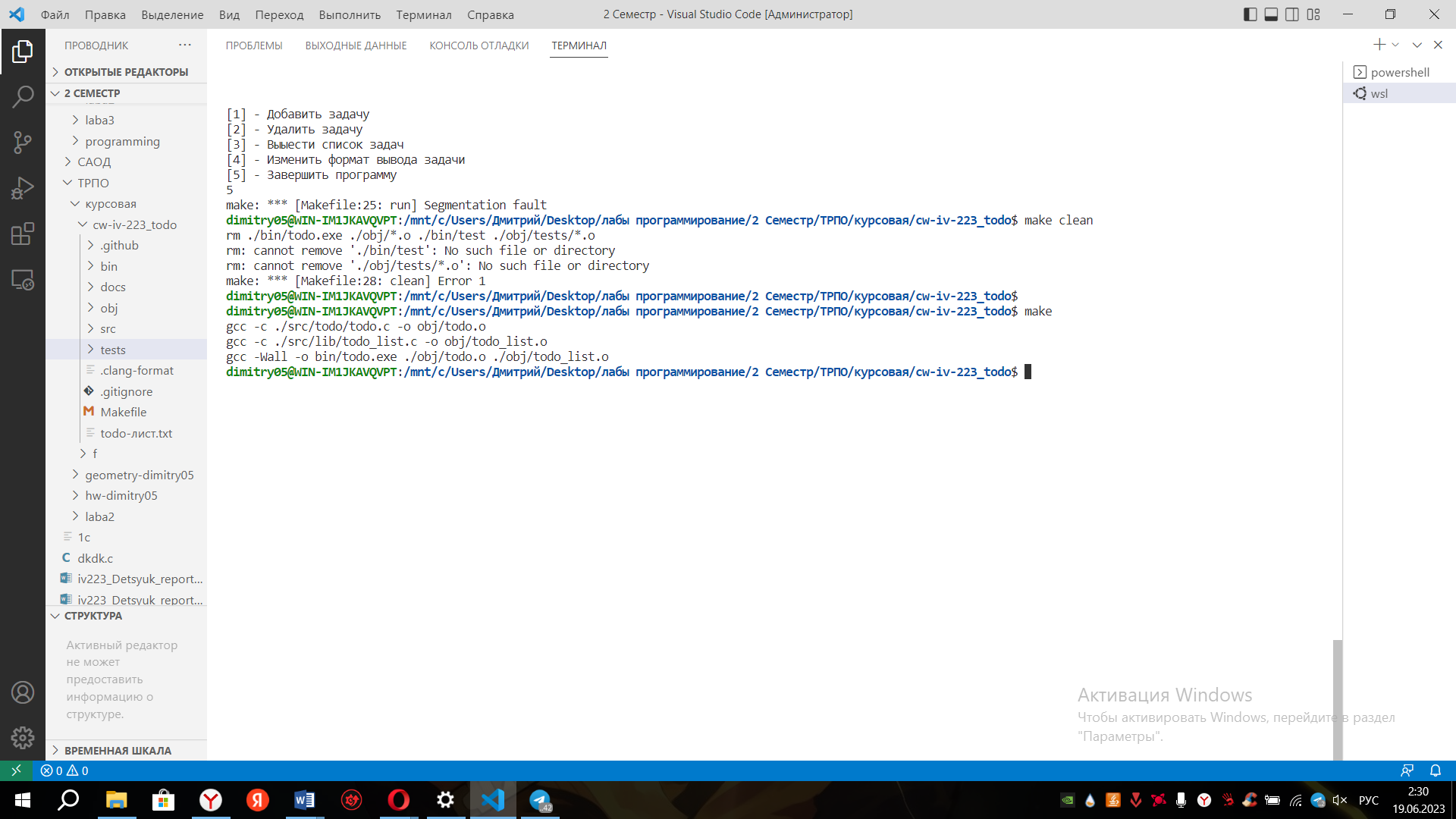
# Техническое задание

1. **Функциональность.**
   1. Приложение разрабатывается на языке программирования C.
   2. Todo-лист - это простой список дел или список задач, в котором можно записать задачи, которые необходимо выполнить, так как порой их так много, что некоторые можно забыть. Пользователь может изменить формат задачи: изменить цвет, добавить подчёркивание, для выделения задачи.
   3. Цель списка задач в том, чтобы записать всё необходимое, для того чтобы в последующем открыть данный список и просмотреть всё что могло забыться.
   4. Пользование списком задач будет осуществляться с помощью меню, которое будет выводиться в терминал.
2. **Входные данные.**
   1. В процессе выполнения программы пользователь вводит в командную строку все необходимые данные в строковом или числовом виде.
3. **Интерфейс.**
   1. Интерфейс программы - командная строка среды программирования системы Linux.
   2. Интерфейс позволит управлять действиями путём ввода команд в командную строку.
4. **Аргументы командной строки.**
   1. Программа в процессе выполнения принимает на вход команды, которые соответствуют предлагаемым меню действиям, если же это ввод задачи, то программа на вход принимает символьную строку, которая будет записана как задача, в иных случаях происходит переход по меню путём ввода числа, которое соответствует пронумерованной команде, выведенной в командную строку в виде меню.
5. **Чтение и запись данных.**
   1. Сохранение задач реализовано путём записи последовательного выполнения команд меню, которые записываются в текстовый файл при завершении выполнения программы, и считываются при повторном запуске программы.

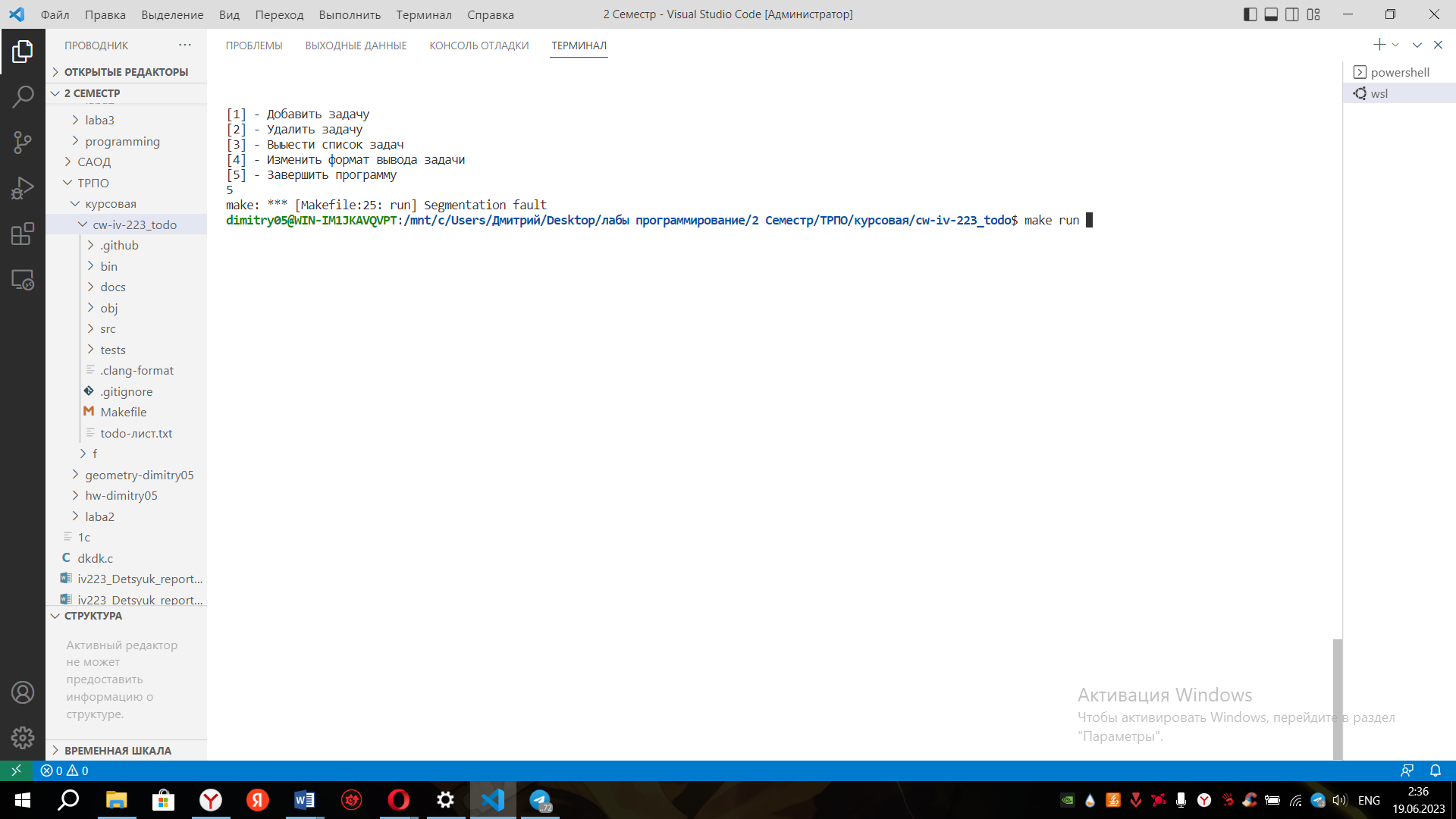
# Описание выполненного проекта

**Функционал работы кода:**

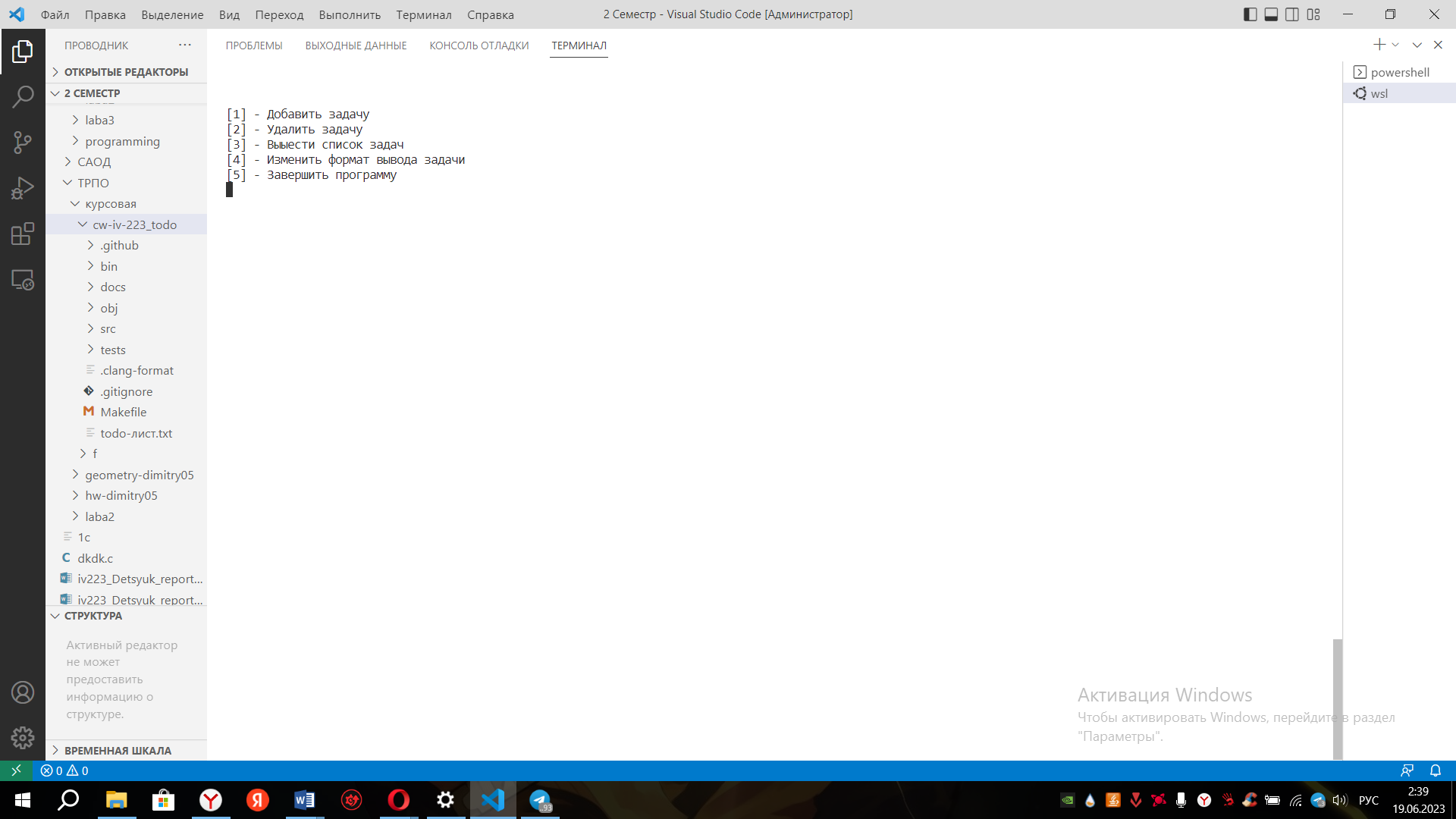
Сборка приложения.



Запуск приложения.

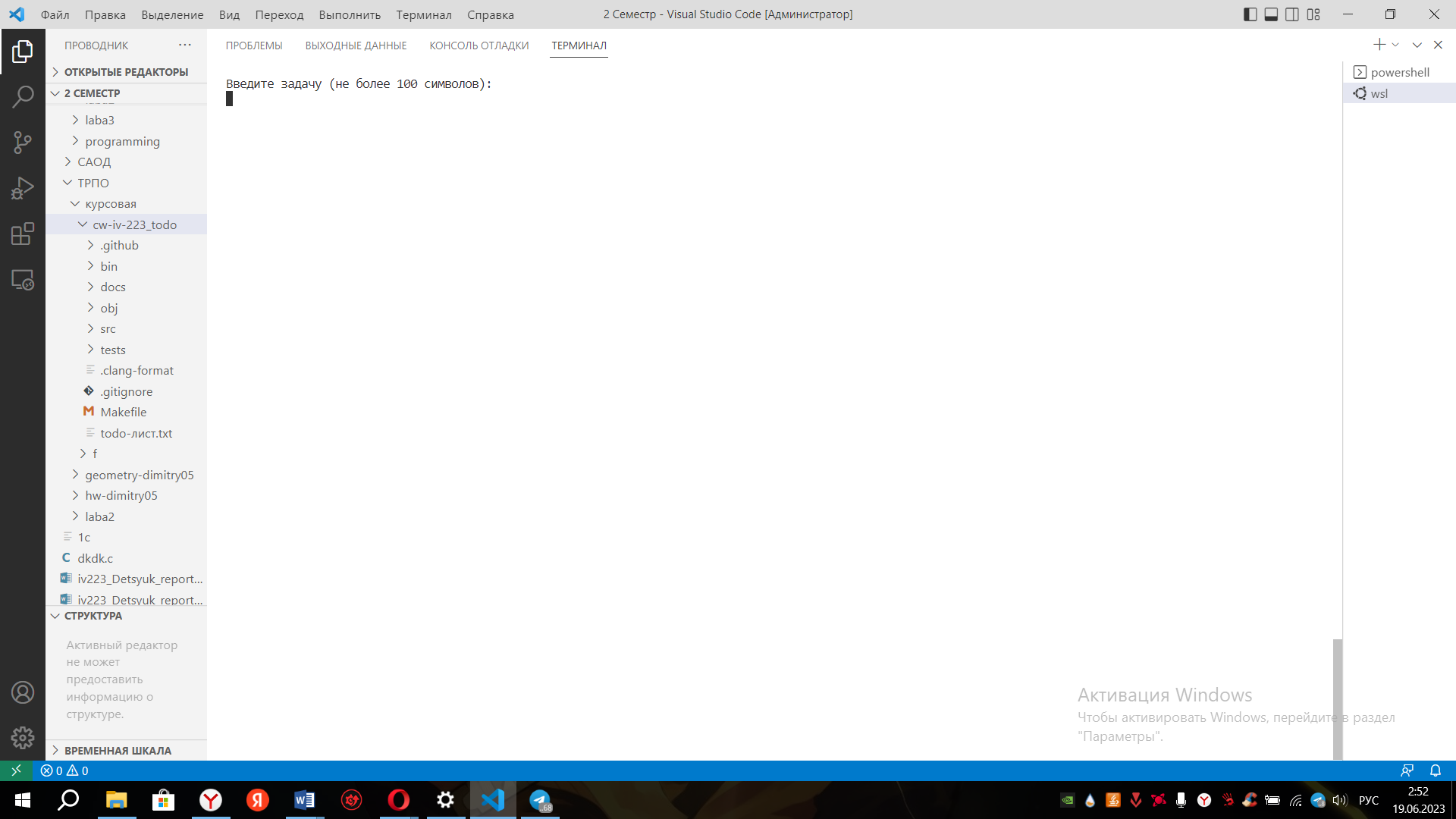


При запуске приложения нас встречает меню.

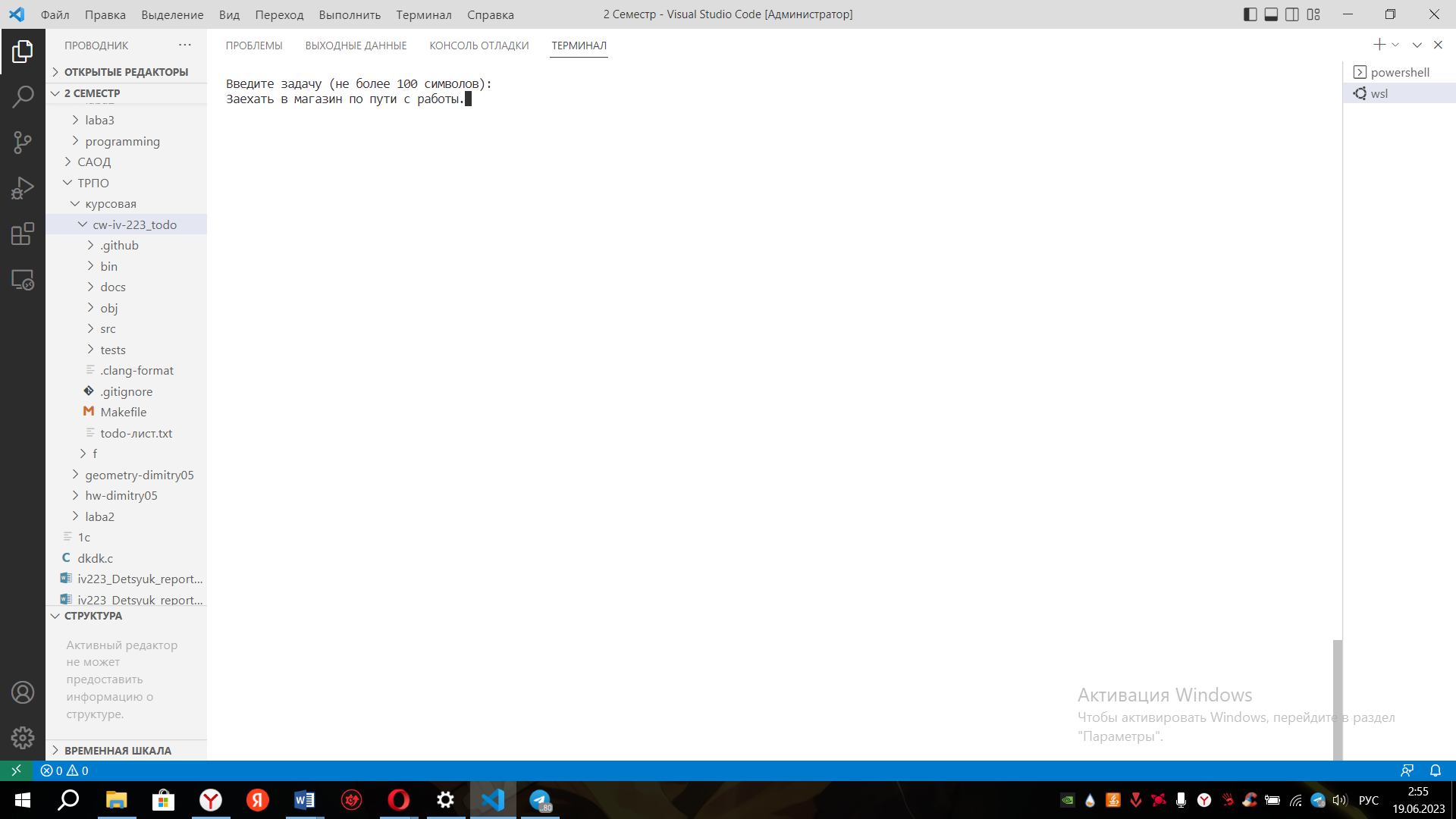


Меню имеет 5 команд: [1] – добавить задачу (добавление задачи в список); [2] – удалить задачу (удаление задачи из списка); [3] – вывести список задач (в консоль выводится весь список задач с установленным форматированием); [4] – изменить формат вывода задачи (происходит переход к выбору формата задачи, который требуется поменять); [5] – завершить программу (записывается весь список задач в файл и завершается работа программы).

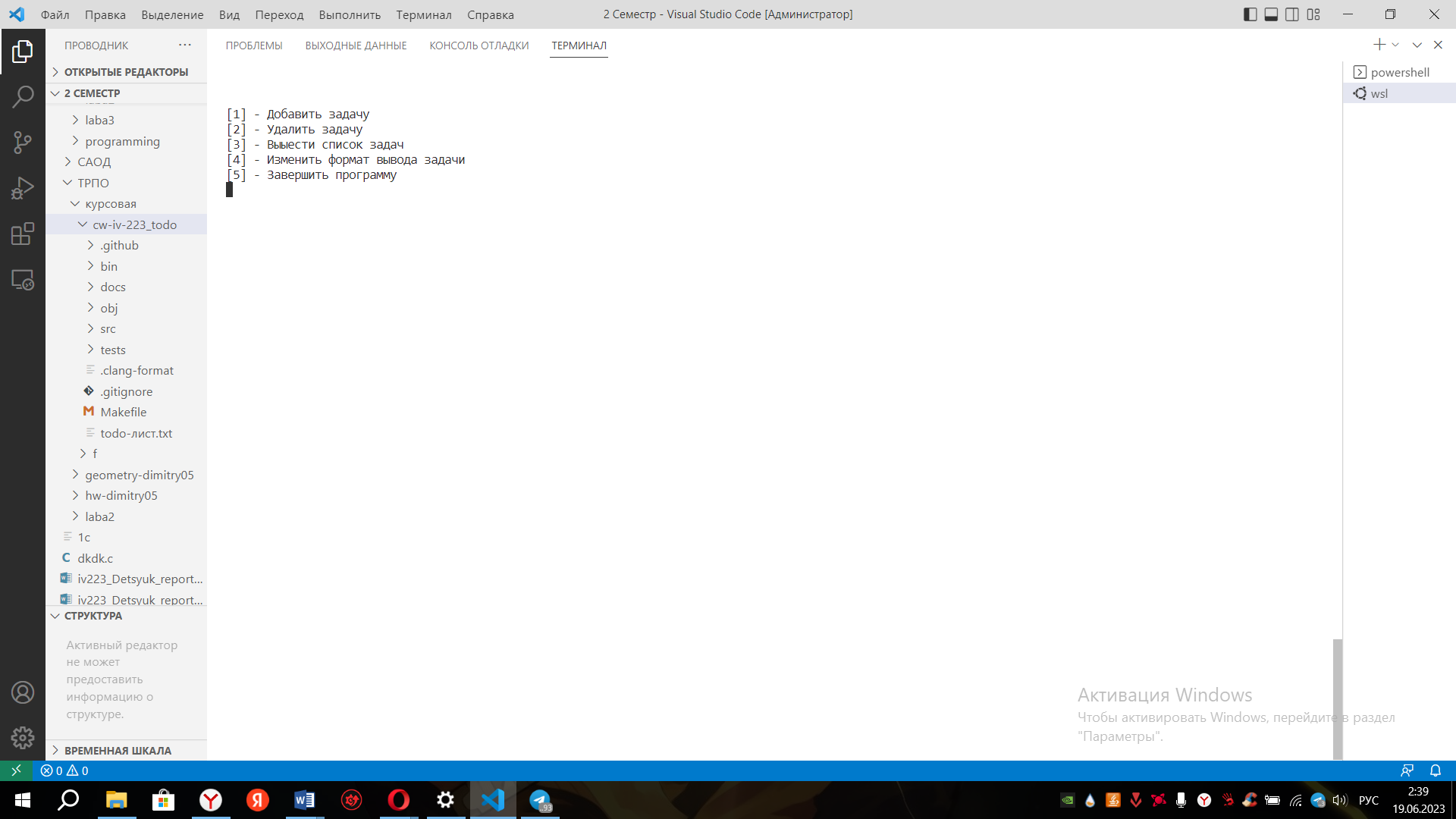
Выберем команду добавления задачи введя 1.



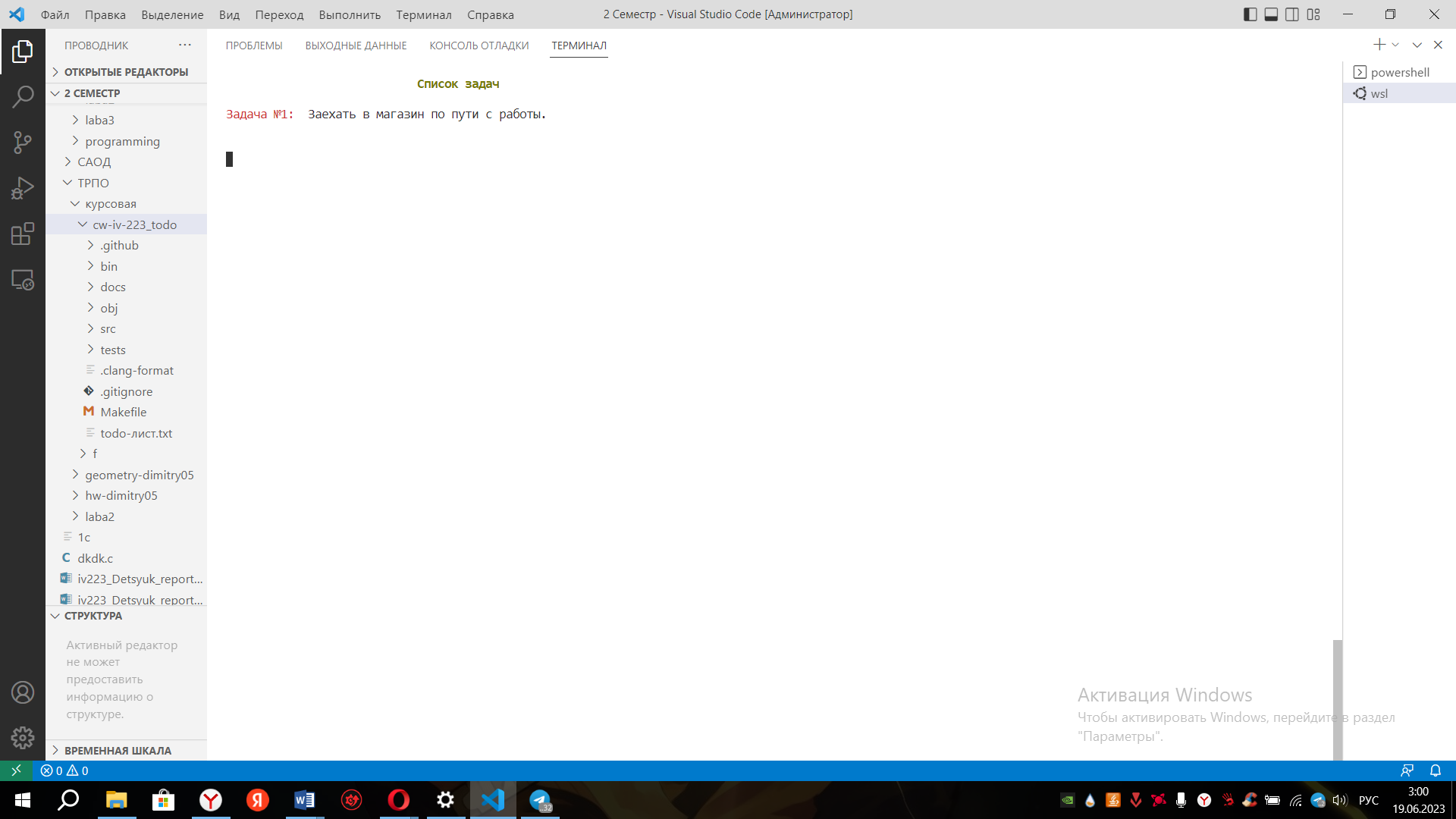
Здесь мы можем ввести задачу не более 100 символов. Введём задачу.



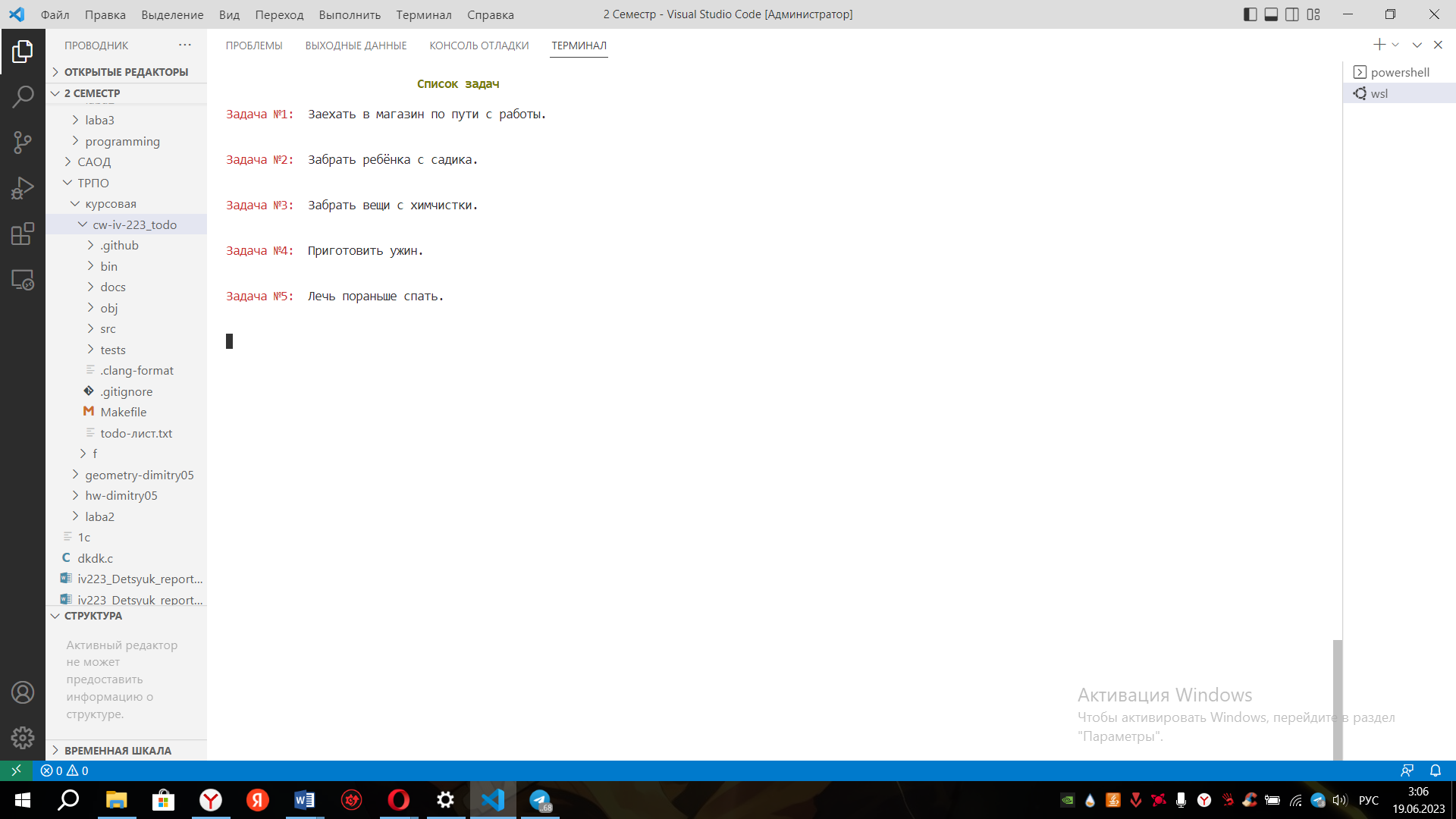
Нажмём Enter чтобы продолжить



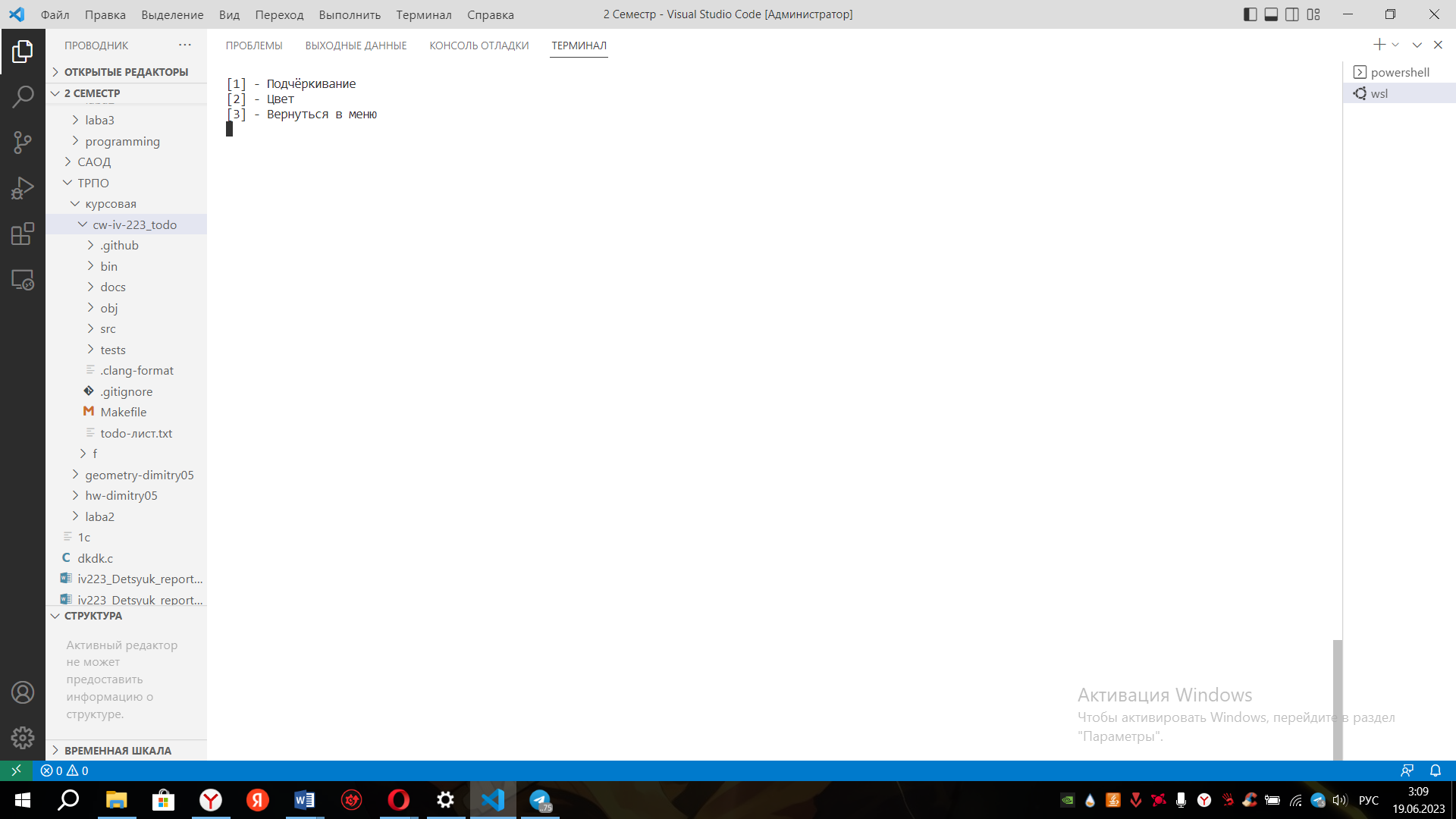
Далее мы снова возвращаемся в меню. Просмотрим список задач.



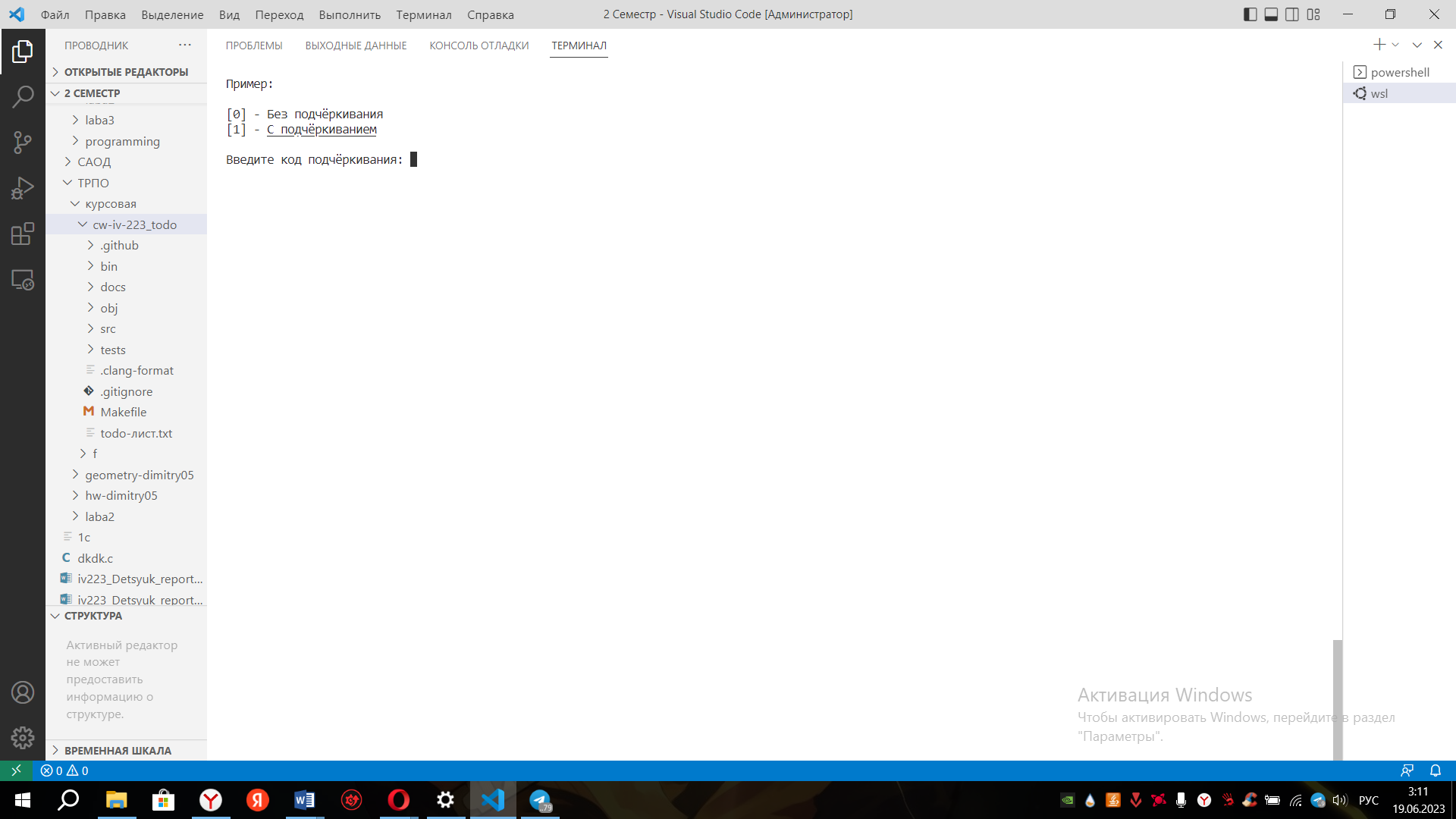
Вернёмся в меню нажав Enter и добавим ещё 4 задачи. Снова выведем список задач.



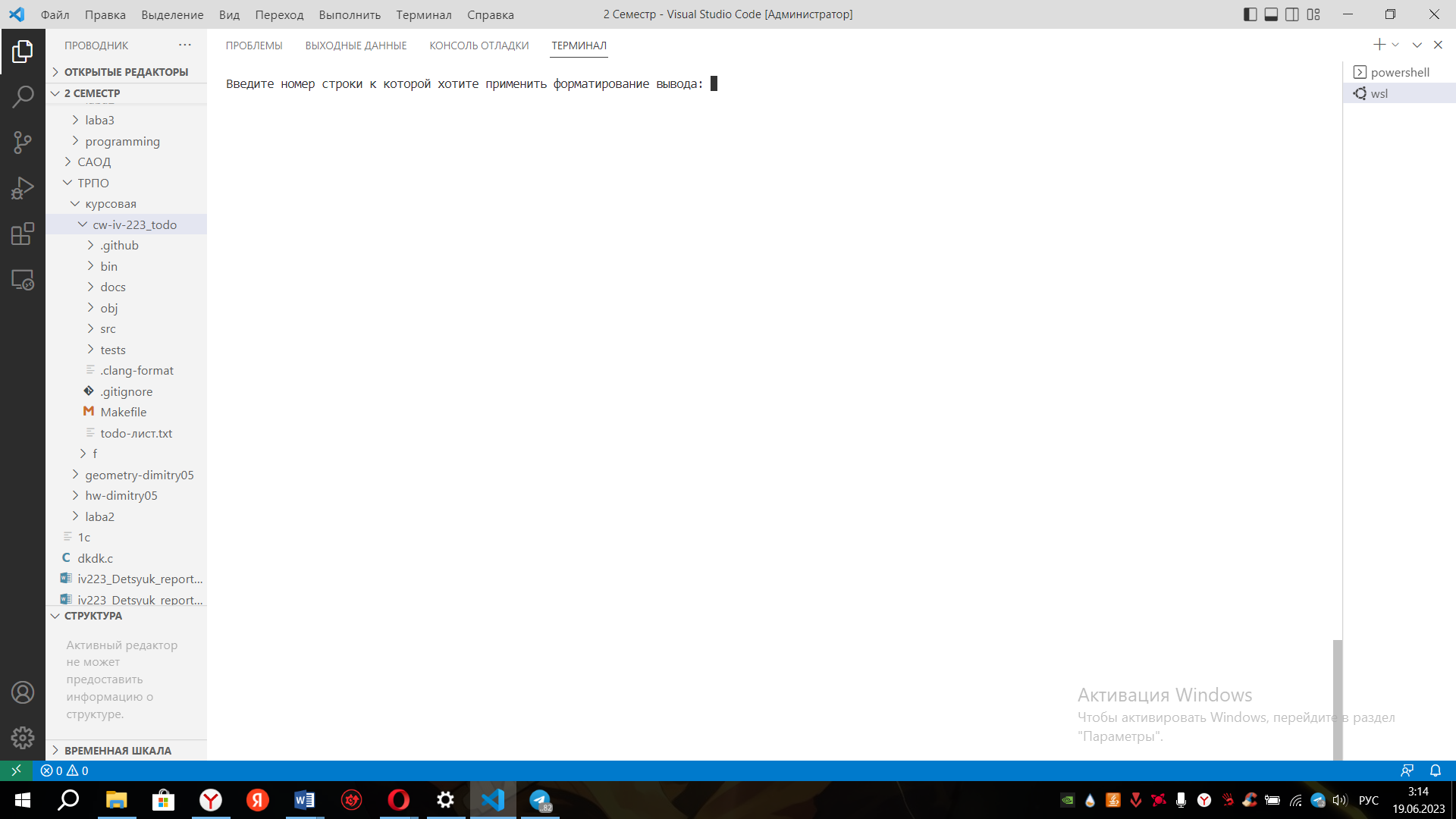
Список задач благополучно вывелся. Вернёмся в меню и попробуем изменить формат вывода задач, для этого введём 4.



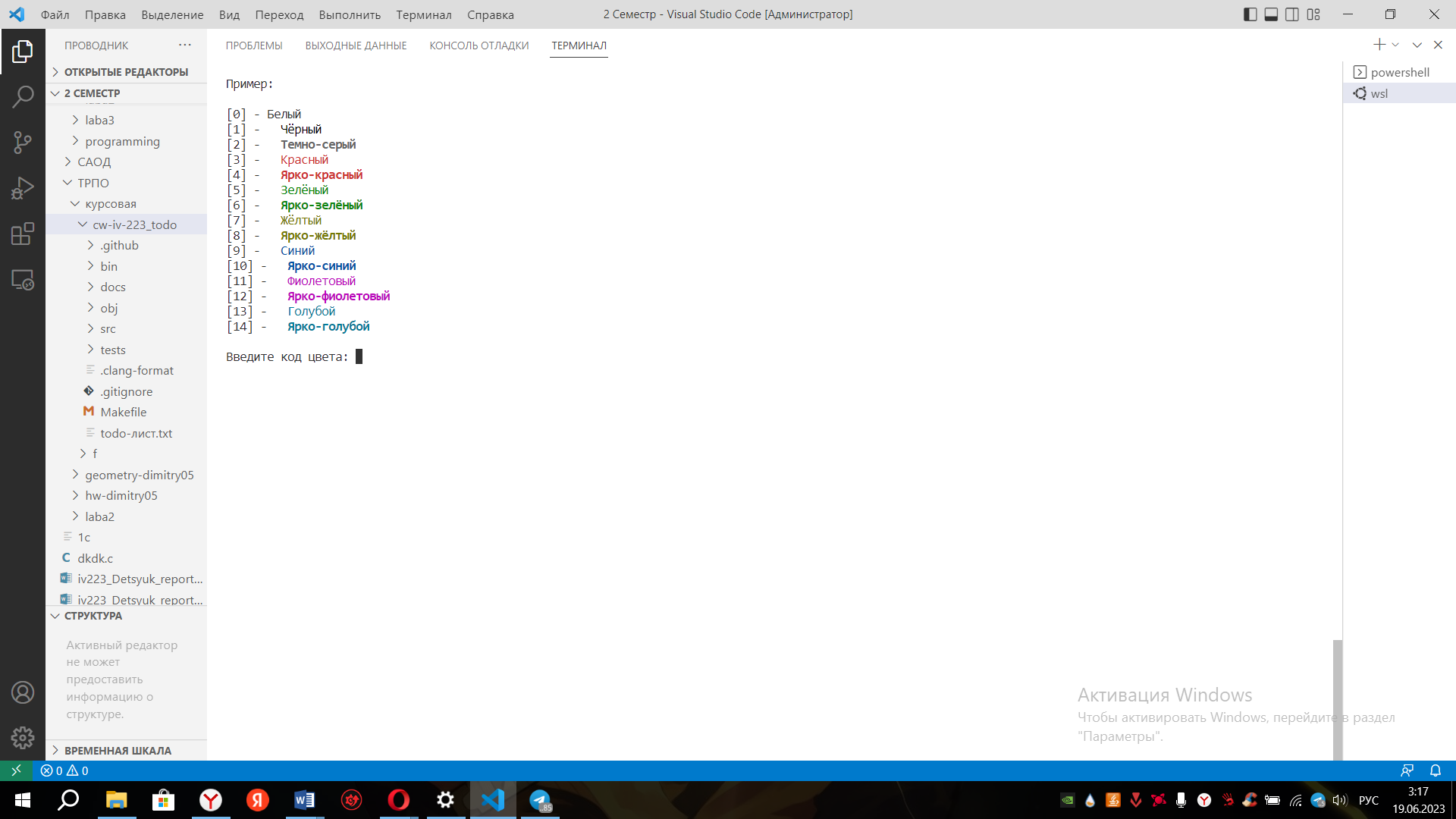
Здесь нам предлагают добавить подчёркивание, изменить цвет или вернуться в меню. Выберем подчёркивание.



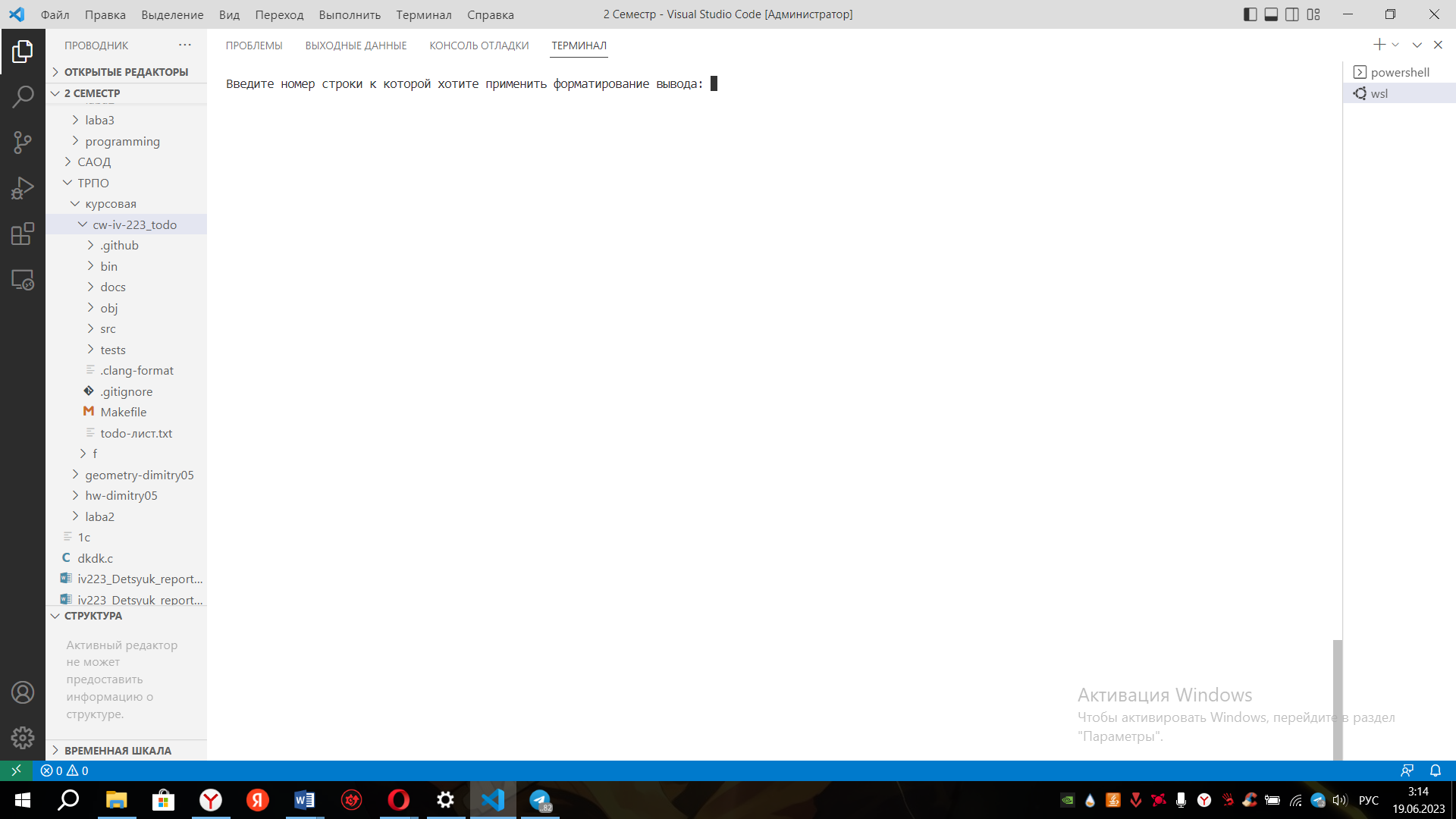
Здесь нам выводится пример и предлагается ввести номер, который мы выберем. Выберем с подчёркиванием.



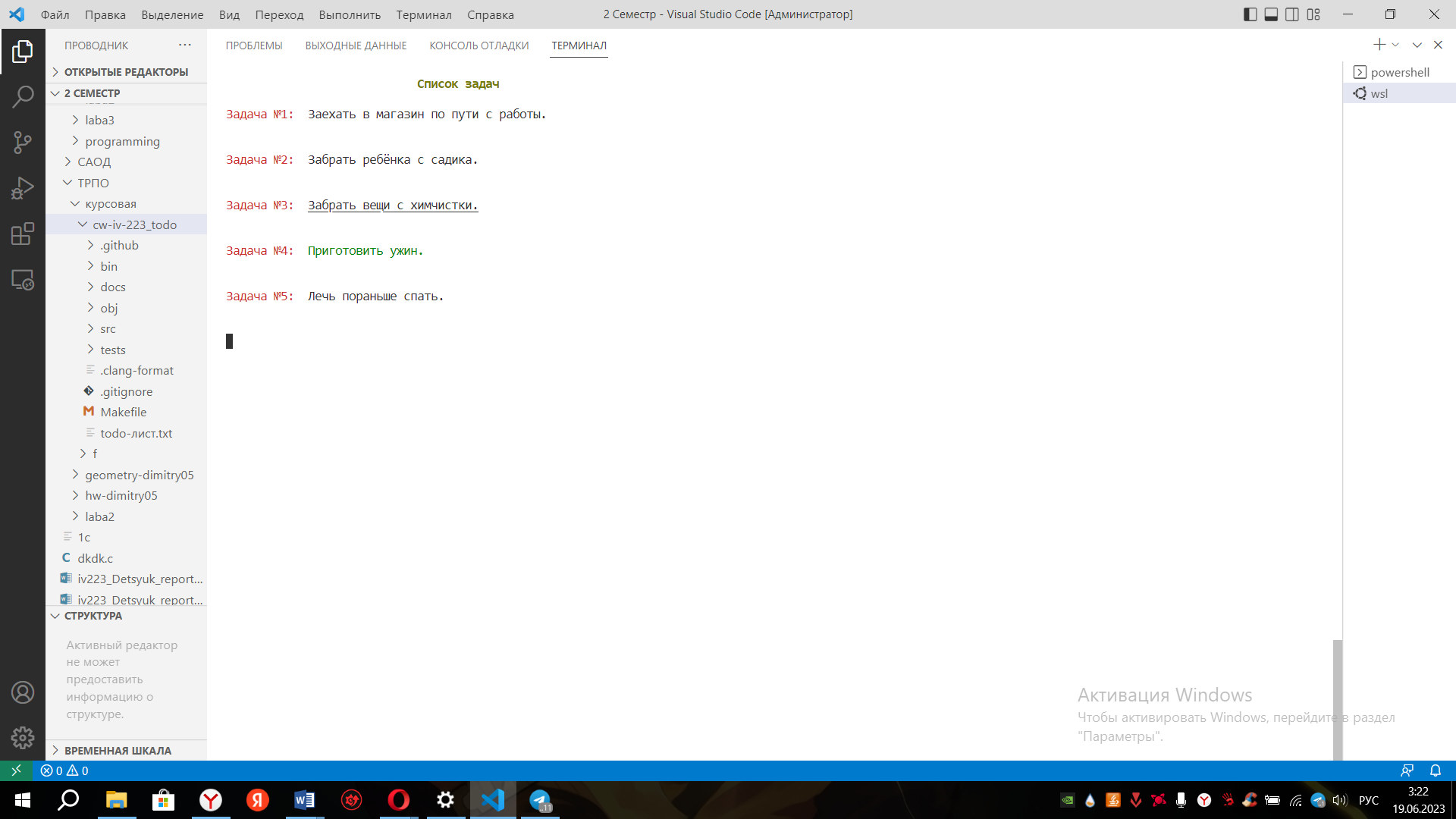
Введём номер 3 и попробуем теперь изменить цвет. Выберем также изменить формат, только теперь цвет.



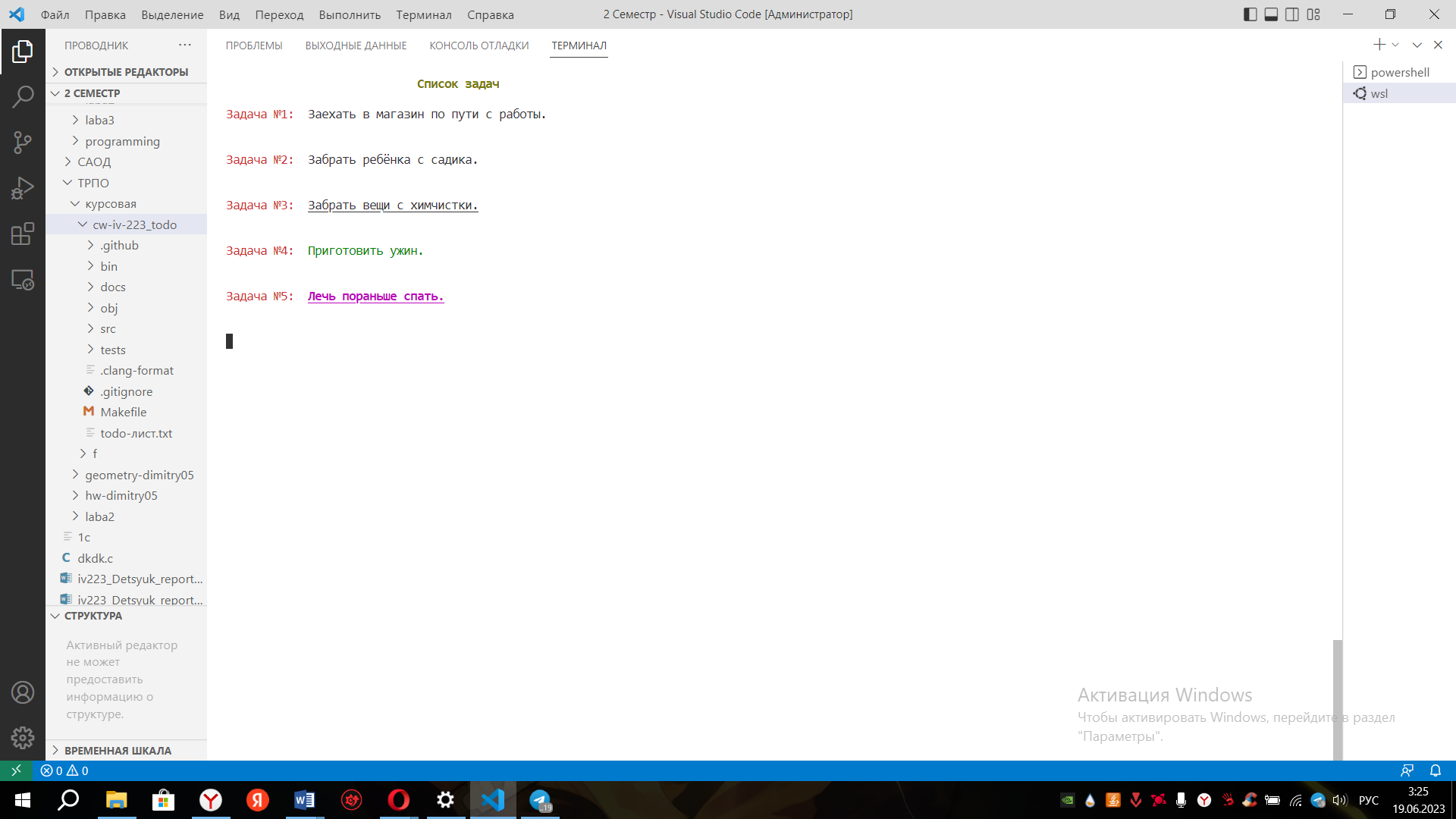
Здесь нам также выводится пример и предлагается ввести номер, который мы выберем. Выберем зелёный.



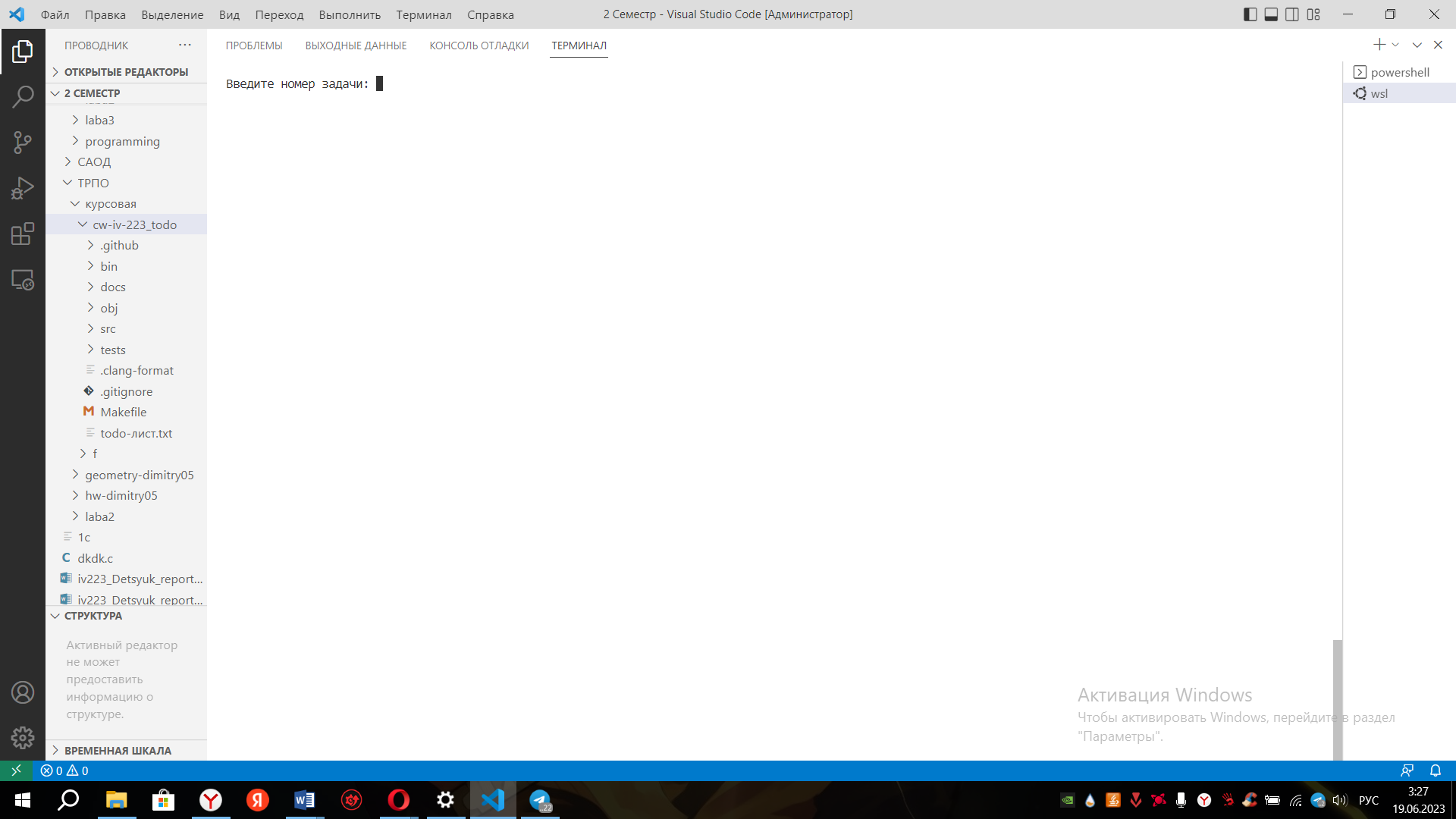
И также предлагается ввести номер задачи. Введём номер 4 и выведем полученный список задач.



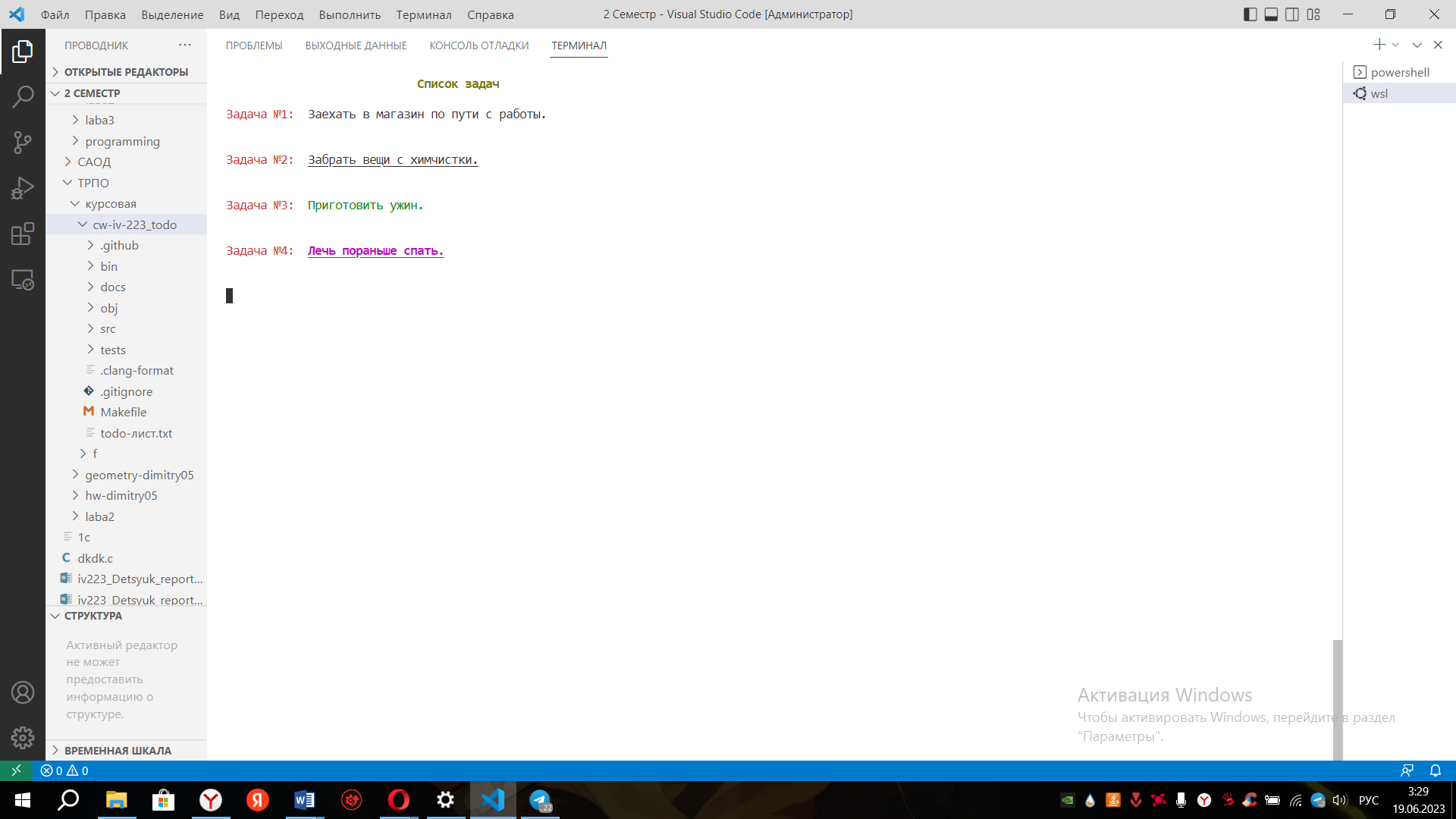
Мы видим, что 3 задача теперь с подчёркиванием, а 4 зелёного цвета. Также можно объединить форматирование, сделаем 5 задачу с подчёркиванием и ярко-фиолетового цвета.



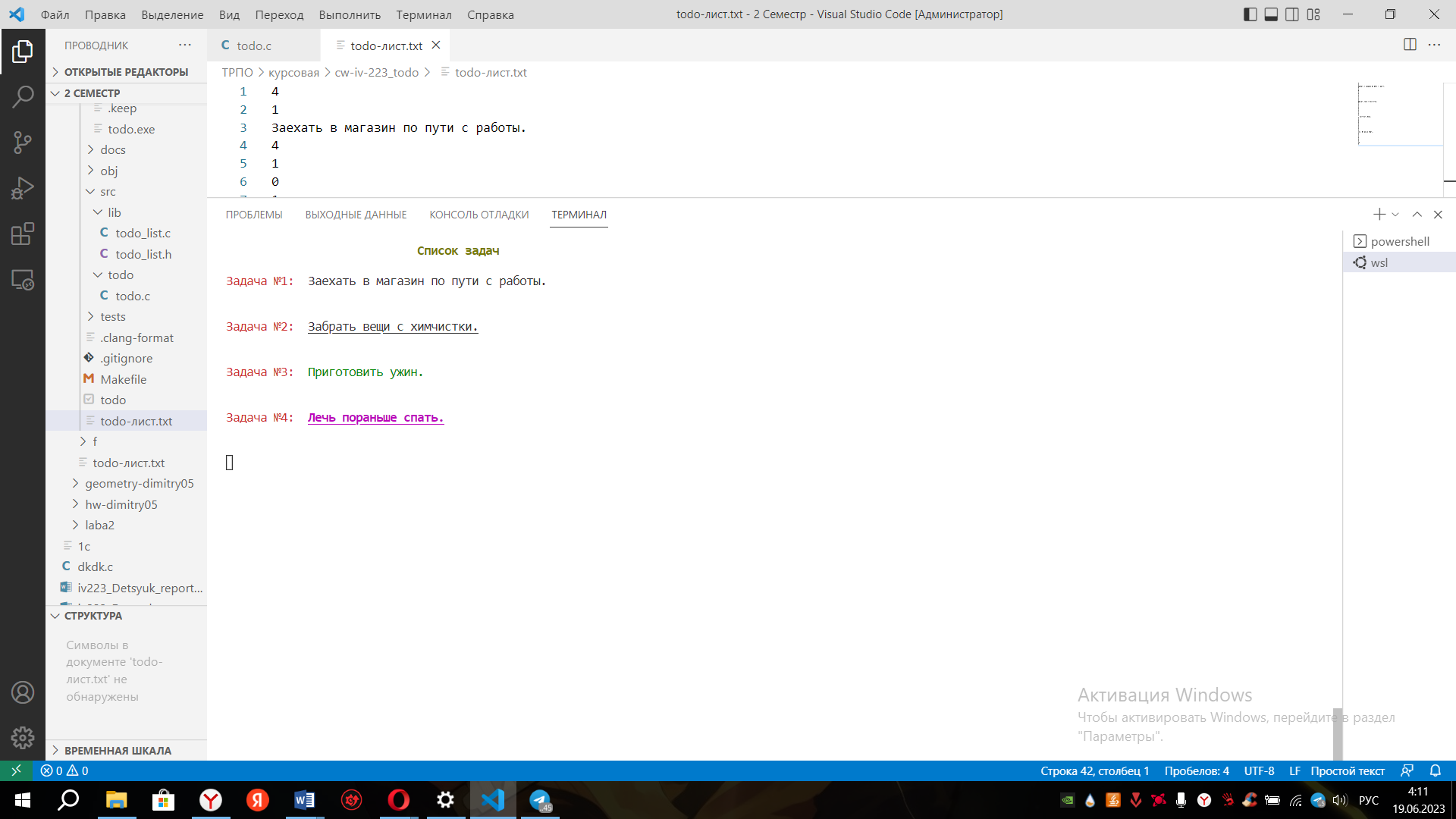
Теперь попробуем удалить вторую задачу. В меню выберем удалить задачу.



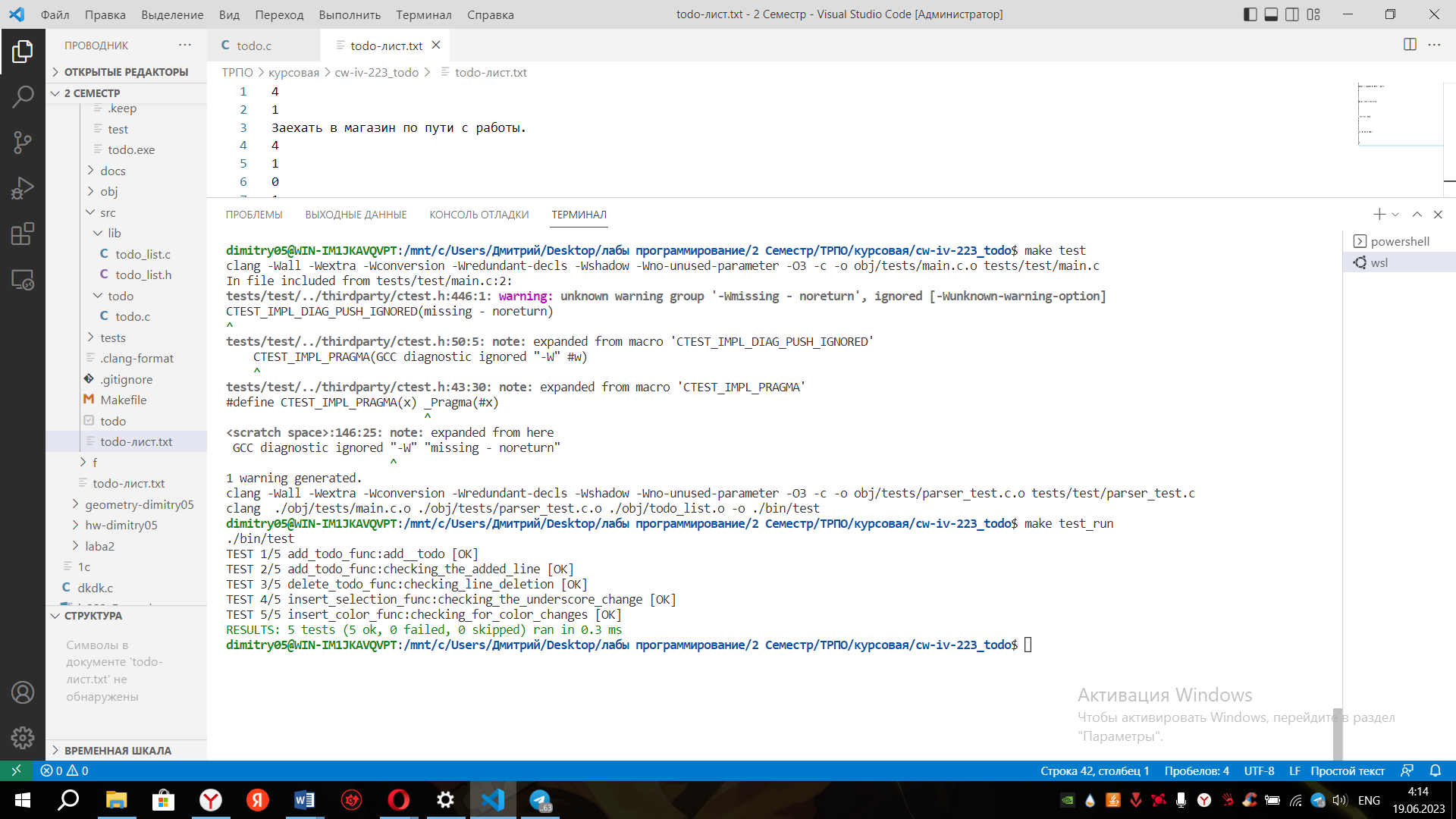
Здесь нам просто предлагается ввести задачу. Введём задачу и выведем список задач.



Мы видим, что задача удалилась и все последующие задачи сдвинулись. Завершим программу и снова запустим. Выведем снова список задач.

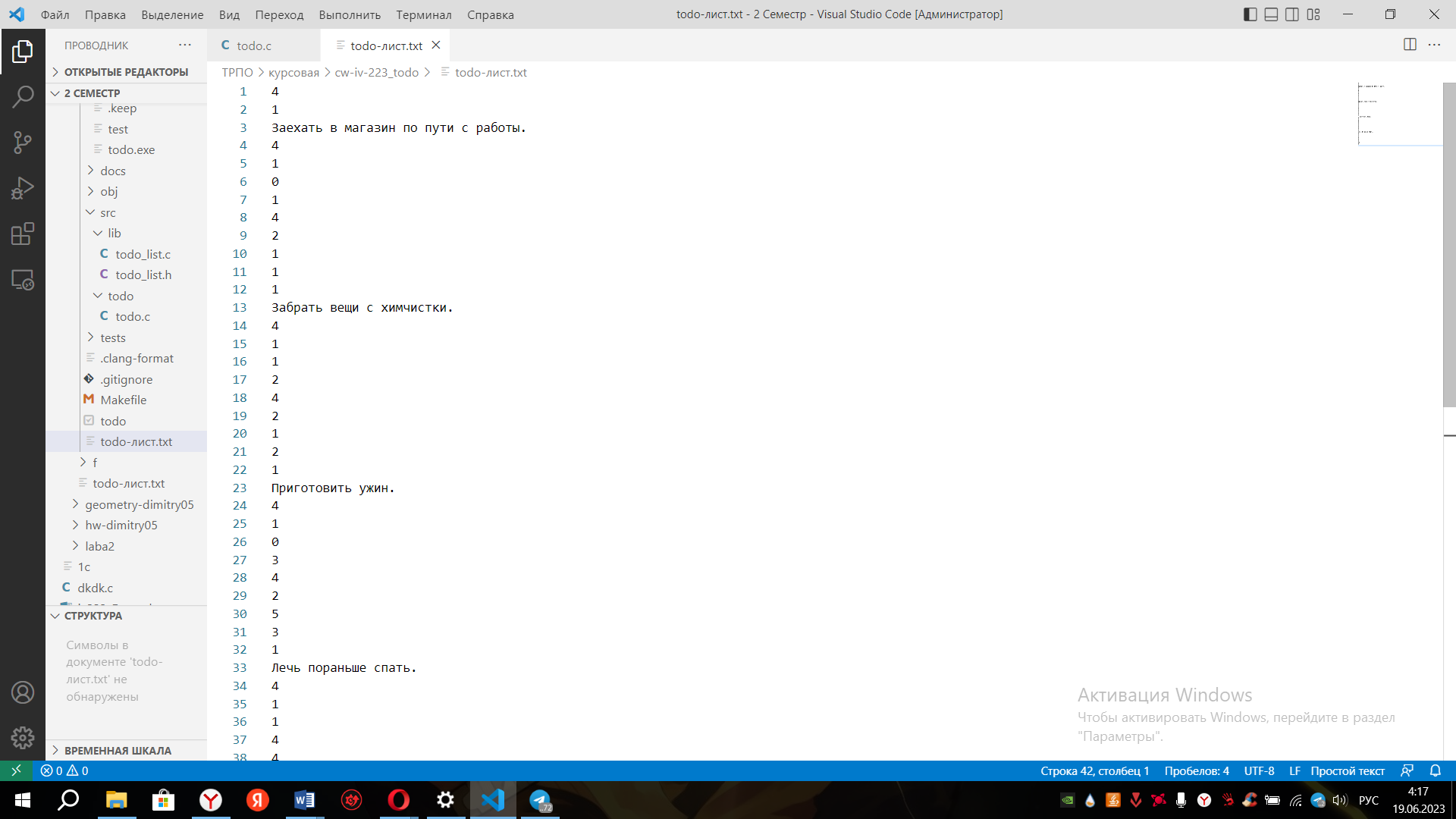


Мы видим, что ничего не поменялось, значит сохранение списка работает. Завершим работу и запустим тестирование функций.



Мы видим, что тестирование прошло без ошибок.

Пример записи последовательности команд задачи для дальнейшего их считывания:



**Описание функций:**

1. add\_todo() – функция добавления задачи в список задач, на вход подаётся указатель на список задач и строка задачи. В случае удачи возвращает 0, если строка не введена выводит ' Задача не введена', иначе возвращает -1.
2. delete\_todo() – функция удаления задачи из списка задач, на вход подаётся указатель на список задач и номер задачи. В случае удачи возвращает 0, если введён недопустимый номер строки, то выводит 'Недопустимый номер строки', иначе возвращает -1.
3. print\_todo\_list() – функция вывода списка задач.
4. insert\_selection() – функция изменения подчёркивания.
5. insert\_color() – функция изменения цвета.
6. print\_selection() – функция вывода примера подчёркивания.
7. print\_color() – функция вывода примера цвета.
8. menu() – функция вывода меню.
9. free\_todo\_list() – функция освобождения памяти списка задач.

# Приложение. Текст программы

**todo.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205 | #include "../lib/todo\_list.h"  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  int main()  {  todo\_list\* list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  FILE\* file = fopen("../todo-лист.txt", "r");  char\* g = malloc(**sizeof**(char) \* 1000);  **if** (file != NULL)  fgets(g, 1000, file);  **else** {  free(g);  g = NULL;  }  int count;  **if** (g != NULL && strlen(g) != 0)  count = atoi(g);  **else**  count = 0;  **while** (1) {  printf("\e[1;1H\e[2J");  menu();  char\* \_code = malloc(**sizeof**(char) \* 10);  **if** (count <= 0)  fgets(\_code, 10, stdin);  **else**  fgets(\_code, 10, file);  int code = atoi(\_code);  free(\_code);  printf("\e[1;1H\e[2J");  **if** (code == 1 || code == 2 || code == 3 || code == 4 || code == 5) {  **if** (code == 5) {  **break**;  }  **if** (code == 1) {  char\* todo = malloc(**sizeof**(char) \* 101);  int n = 1;  **while** (n != 0) {  printf("Введите задачу (не более 100 символов):**\n**");  **if** (count <= 0)  fgets(todo, 101, stdin);  **else**  fgets(todo, 101, file);  **if** (todo != NULL)  n = add\_todo(list, todo);  **else**  printf("Задача не введена. ");  printf("\e[1;1H\e[2J");  }  }  **if** (code == 2) {  int n = -1, num1;  char\* num = malloc(**sizeof**(char) \* 10);  **while** (n != 0) {  printf("Введите номер задачи: ");  **if** (count <= 0)  fgets(num, 10, stdin);  **else**  fgets(num, 10, file);  **if** (num != NULL)  num1 = atoi(num);  n = delete\_todo(list, num1);  printf("\e[1;1H\e[2J");  }  free(num);  }  **if** (code == 3) {  print\_todo\_list(list);  **if** (count <= 0)  getc(stdin);  **else**  getc(file);  printf("\e[1;1H\e[2J");  }  **if** (code == 4) {  **while** (1) {  printf("[1] - Подчёркивание**\n**");  printf("[2] - Цвет**\n**");  printf("[3] - Вернуться в меню**\n**");  int code1, l = -1;  char\* \_code1 = malloc(**sizeof**(char) \* 10);  **if** (count <= 0)  fgets(\_code1, 10, stdin);  **else**  fgets(\_code1, 10, file);  code1 = atoi(\_code1);  free(\_code1);  printf("\e[1;1H\e[2J");  **if** (code1 == 1 || code1 == 2 || code1 == 3) {  **if** (code1 == 3) {  printf("\e[1;1H\e[2J");  **break**;  }  **if** (code1 == 2) {  int n = -1;  char\* code2 = malloc(**sizeof**(char) \* 10);  **while** (n < 0 || n > 14) {  printf("Пример:**\n\n**");  print\_color();  printf("**\n**Введите код цвета: ");  **if** (count <= 0)  fgets(code2, 10, stdin);  **else**  fgets(code2, 10, file);  **if** (code2 != NULL)  n = atoi(code2);  printf("\e[1;1H\e[2J");  **if** (n < 0 || n > 14)  printf("Не верный код цвета. ");  }  free(code2);  l = n;  }  **if** (code1 == 1) {  int n = -1;  char\* code2 = malloc(**sizeof**(char) \* 10);  **while** (n < 0 || n > 1) {  printf("Пример:**\n\n**");  print\_selection();  printf("**\n**Введите код подчёркивания: ");  **if** (count <= 0)  fgets(code2, 10, stdin);  **else**  fgets(code2, 10, file);  **if** (code2 != NULL)  n = atoi(code2);  printf("\e[1;1H\e[2J");  **if** (n < 0 || n > 1)  printf("Не верный код подчёркивания. ");  }  free(code2);  l = n;  }  }  **if** (l != -1) {  **while** (l != -1) {  printf("Введите номер строки к которой хотите "  "применить форматирование вывода: ");  char\* num = malloc(**sizeof**(char) \* 10);  **if** (count <= 0)  fgets(num, 10, stdin);  **else**  fgets(num, 10, file);  **if** (num != NULL) {  int n = atoi(num);  **if** (n > 0 && n <= list->size) {  **if** (code1 == 1)  insert\_selection(list, n, l);  **if** (code1 == 2) {  insert\_color(list, n, l - 1);  **if** (count > 0)  count--;  }  **break**;  }  }  printf("\e[1;1H\e[2J");  printf("Недопустимый номер задачи.**\n**");  }  **break**;  }  }  }  }  }  fclose(file);  file = fopen("todo-лист.txt", "w");  char\* num = malloc(**sizeof**(char) \* 10);  sprintf(num, "%d", list->size);  fputs(num, file);  fprintf(file, "**\n**");  **for** (int i = 0; i < list->size; i++) {  fprintf(file, "1");  fprintf(file, "**\n**");  fprintf(file, "%s", list->todo[i]);  fprintf(file, "4");  fprintf(file, "**\n**");  fprintf(file, "1");  fprintf(file, "**\n**");  sprintf(num, "%d", list->selection[i]);  fprintf(file, "%s", num);  fprintf(file, "**\n**");  sprintf(num, "%d", i + 1);  fprintf(file, "%s", num);  fprintf(file, "**\n**");  fprintf(file, "4");  fprintf(file, "**\n**");  fprintf(file, "2");  fprintf(file, "**\n**");  sprintf(num, "%d", list->color[i] + 1);  fprintf(file, "%s", num);  fprintf(file, "**\n**");  sprintf(num, "%d", i + 1);  fprintf(file, "%s", num);  fprintf(file, "**\n**");  }  fclose(file);  free\_todo\_list(list);  } |

**todo-list.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170 | #include "todo\_list.h"  #define CSI "\x1B\x5B"  char colors[][5]  = {"0;30",  */\* Black \*/* "1;30", */\* Dark Gray \*/*  "0;31",  */\* Red \*/* "1;31", */\* Bold Red \*/*  "0;32",  */\* Green \*/* "1;32", */\* Bold Green \*/*  "0;33",  */\* Yellow \*/* "1;33", */\* Bold Yellow \*/*  "0;34",  */\* Blue \*/* "1;34", */\* Bold Blue \*/*  "0;35",  */\* Purple \*/* "1;35", */\* Bold Purple \*/*  "0;36",  */\* Cyan \*/* "1;36" */\*Bold Cyan \*/*};  int add\_todo(todo\_list\* list, char\* todo)  {  **if** (list == NULL)  **return** -1;  **if** (todo == NULL || strlen(todo) == 0) {  printf("Задача не введена. ");  **return** 1;  }  **if** (list->size == -1)  list->size++;  list->size++;  list->todo = realloc(list->todo, **sizeof**(char\*) \* list->size);  list->selection = realloc(list->selection, **sizeof**(int) \* list->size);  list->color = realloc(list->color, **sizeof**(int) \* list->size);  list->todo[list->size - 1] = todo;  **return** 0;  }  int delete\_todo(todo\_list\* list, int number\_todo)  {  **if** (list == NULL)  **return** -1;  **if** (number\_todo > list->size || number\_todo < 1) {  printf("Недопустимый номер задачи. ");  **return** 1;  }  **while** (number\_todo < list->size) {  list->todo[number\_todo - 1] = list->todo[number\_todo];  list->selection[number\_todo - 1] = list->selection[number\_todo];  list->color[number\_todo - 1] = list->color[number\_todo];  number\_todo++;  }  list->size--;  list->todo = realloc(list->todo, **sizeof**(char\*) \* list->size);  list->selection = realloc(list->selection, **sizeof**(int) \* list->size);  list->color = realloc(list->color, **sizeof**(int) \* list->size);  **return** 0;  }  int print\_todo\_list(todo\_list\* list)  {  **if** (list == NULL)  **return** -1;  printf("%s%sm Список задач %s0m **\n\n**",  CSI,  colors[7],  CSI);  **if** (list->size == 0 || list->size == -1) {  printf("%s%sm Задач нет.. %s0m **\n**", CSI, colors[2], CSI);  **return** 0;  }  **for** (int i = 0; i < list->size; i++) {  **if** (list->selection[i] == 1) {  **if** (list->color[i] == 0)  printf("%s%smЗадача №%d: %s0m %s4m%s%s0m**\n\n**",  CSI,  colors[2],  i + 1,  CSI,  CSI,  list->todo[i],  CSI);  **else**  printf("%s%smЗадача №%d: %s0m %s4m%s%sm%s%s0m**\n\n**",  CSI,  colors[2],  i + 1,  CSI,  CSI,  CSI,  colors[list->color[i]],  list->todo[i],  CSI);  } **else** {  **if** (list->color[i] == 0)  printf("%s%smЗадача №%d:%s0m %s%s0m**\n\n**",  CSI,  colors[2],  i + 1,  CSI,  list->todo[i],  CSI);  **else**  printf("%s%smЗадача №%d:%s0m%s%sm %s%s0m**\n\n**",  CSI,  colors[2],  i + 1,  CSI,  CSI,  colors[list->color[i]],  list->todo[i],  CSI);  }  }  **return** 0;  }  void insert\_selection(todo\_list\* list, int num\_todo, int selection)  {  list->selection[num\_todo - 1] = selection;  }  void insert\_color(todo\_list\* list, int num\_todo, int color)  {  list->color[num\_todo - 1] = color;  }  void print\_selection()  {  printf("[0] - Без подчёркивания%s0m**\n**", CSI);  printf("[1] - %s4mС подчёркиванием%s0m**\n**", CSI, CSI);  }  void print\_color()  {  printf("[0] - Белый**\n**");  printf("[1] - %s%sm Чёрный%s0m**\n**", CSI, colors[0], CSI);  printf("[2] - %s%sm Темно-серый%s0m**\n**", CSI, colors[1], CSI);  printf("[3] - %s%sm Красный%s0m**\n**", CSI, colors[2], CSI);  printf("[4] - %s%sm Ярко-красный%s0m**\n**", CSI, colors[3], CSI);  printf("[5] - %s%sm Зелёный%s0m**\n**", CSI, colors[4], CSI);  printf("[6] - %s%sm Ярко-зелёный%s0m**\n**", CSI, colors[5], CSI);  printf("[7] - %s%sm Жёлтый%s0m**\n**", CSI, colors[6], CSI);  printf("[8] - %s%sm Ярко-жёлтый%s0m**\n**", CSI, colors[7], CSI);  printf("[9] - %s%sm Синий%s0m**\n**", CSI, colors[8], CSI);  printf("[10] - %s%sm Ярко-синий%s0m**\n**", CSI, colors[9], CSI);  printf("[11] - %s%sm Фиолетовый%s0m**\n**", CSI, colors[10], CSI);  printf("[12] - %s%sm Ярко-фиолетовый%s0m**\n**", CSI, colors[11], CSI);  printf("[13] - %s%sm Голубой%s0m**\n**", CSI, colors[12], CSI);  printf("[14] - %s%sm Ярко-голубой%s0m**\n**", CSI, colors[13], CSI);  }  void menu()  {  printf("**\n\n**[1] - Добавить задачу**\n**[2] - Удалить задачу**\n**[3] - Выыести "  "список задач**\n**[4] - Изменить формат вывода задачи**\n**[5] - Завершить "  "программу**\n**");  }  void free\_todo\_list(todo\_list\* list)  {  **if** (list->size != -1)  **for** (int i = 0; i < list->size; i++) {  free(list->todo[i]);  }  free(list->color);  free(list->selection);  free(list->todo);  free(list);  } |

**todo-list.h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | #pragma once  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  **typedef** **struct** {  char\*\* todo;  int size;  int\* selection;  int\* color;  } todo\_list;  int add\_todo(todo\_list\* list, char\* todo);  int delete\_todo(todo\_list\* list, int number\_todo);  int print\_todo\_list(todo\_list\* list);  void insert\_selection(todo\_list\* list, int num\_todo, int selection);  void insert\_color(todo\_list\* list, int num\_todo, int color);  void print\_selection();  void print\_color();  void menu();  void free\_todo\_list(todo\_list\* list); |

**test/main.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #define CTEST\_MAIN  #include "../thirdparty/ctest.h"  int main(int argc, **const** char\*\* argv)  {  **return** ctest\_main(argc, argv);  } |

**parser\_test.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | #include "../thirdparty/ctest.h"  #include <string.h>  #include "../../src/lib/todo\_list.h"  CTEST(add\_todo\_func, add\_\_todo)  {  char todo[50] = "Hallo!";  todo\_list\* list = NULL;  int n = add\_todo(list, todo);  ASSERT\_EQUAL(-1, n);  list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  n = add\_todo(list, todo);  ASSERT\_EQUAL(0, n);  }  CTEST(add\_todo\_func, checking\_the\_added\_line)  {  todo\_list\* list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  char todo[50] = "Hallo!";  add\_todo(list, todo);  ASSERT\_STR(list->todo[0], "Hallo!");  }  CTEST(delete\_todo\_func, checking\_line\_deletion)  {  todo\_list\* list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  char todo[50] = "Hallo!";  add\_todo(list, todo);  ASSERT\_EQUAL(1, list->size);  delete\_todo(list, 1);  ASSERT\_EQUAL(0, list->size);  }  CTEST(insert\_selection\_func, checking\_the\_underscore\_change)  {  todo\_list\* list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  char todo[50] = "Hallo!";  add\_todo(list, todo);  insert\_selection(list, 1, 1);  ASSERT\_EQUAL(1, list->selection[0]);  }  CTEST(insert\_color\_func, checking\_for\_color\_changes)  {  todo\_list\* list = malloc(**sizeof**(\*list));  list->todo = (char\*\*)malloc(**sizeof**(char\*));  list->size = -1;  list->selection = (int\*)calloc(1, **sizeof**(int));  list->color = (int\*)calloc(1, **sizeof**(int));  char todo[50] = "Hallo!";  add\_todo(list, todo);  insert\_color(list, 1, 2);  ASSERT\_EQUAL(2, list->color[0]);  } |

**Makefile**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | TARGET = todo  CCFLAGS=-Wall -Wextra -Wconversion -Wredundant-decls -Wshadow -Wno-unused-parameter -O3  CC=clang  PREF\_BIN = ./bin/  PREF\_SRC\_LIB = ./src/lib/  PREF\_SRC\_PROG = ./src/todo/  PREF\_OBJ = ./obj/  PREF\_OBJ\_TESTS = ./obj/tests/  PREF\_TESTS\_TEST = ./tests/test/  PREF\_TESTS\_THIRDPARTY = ./tests/thirdparty/  all: $(PREF\_BIN)$(TARGET)  $(PREF\_BIN)$(TARGET): $(PREF\_OBJ)todo.o $(PREF\_OBJ)todo\_list.o  gcc -Wall -o $@.exe $(PREF\_OBJ)todo.o $(PREF\_OBJ)todo\_list.o  $(PREF\_OBJ)todo.o: src/todo/todo.c  gcc -c $(PREF\_SRC\_PROG)todo.c -o $@  $(PREF\_OBJ)todo\_list.o: src/lib/todo\_list.c  gcc -c $(PREF\_SRC\_LIB)todo\_list.c -o $@  run:  $(PREF\_BIN)$(TARGET).exe  clean:  rm $(PREF\_BIN)$(TARGET).exe $(PREF\_OBJ)\*.o $(PREF\_BIN)test $(PREF\_OBJ\_TESTS)\*.o  test: $(PREF\_OBJ\_TESTS)main.c.o $(PREF\_TESTS\_THIRDPARTY)ctest.h $(PREF\_OBJ\_TESTS)parser\_test.c.o $(PREF\_OBJ)todo\_list.o  $(CC) $(LDFLAGS) $(PREF\_OBJ\_TESTS)main.c.o $(PREF\_OBJ\_TESTS)parser\_test.c.o $(PREF\_OBJ)todo\_list.o -o $(PREF\_BIN)test  $(PREF\_OBJ\_TESTS)main.c.o: $(PREF\_TESTS\_TEST)main.c $(PREF\_TESTS\_THIRDPARTY)ctest.h  $(CC) $(CCFLAGS) -c -o $@ $<  $(PREF\_OBJ\_TESTS)parser\_test.c.o: $(PREF\_TESTS\_TEST)parser\_test.c $(PREF\_TESTS\_THIRDPARTY)ctest.h  $(CC) $(CCFLAGS) -c -o $@ $<  test\_run:  $(PREF\_BIN)test  clean\_test:  rm -f $(PREF\_BIN)test $(PREF\_OBJ\_TESTS)\*.o |