**Задание №6**

**Отладка программного кода**

Отладка (debugging) - этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, делают следующее:

- узнают текущие значения переменных;

- выясняют, по какому пути выполнялась программа.

Использование отладчика в VS позволяет в числе прочего пошагово выполнить программу: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.

Задание - написать программу вычисления корня квадратного уравнения. Для написания программы вспомним порядок действий (алгоритм) Работа уже выполнялась, но работу сделать еще раз!

Общий алгоритм такой:

1. Спросить у пользователя параметры уравнения: A, B, C;
2. Вычислить дискриминант по формуле D = B \* B – 4 \* A \* C;
3. В зависимости от значения дискриминанта вычислить корни или вывести, что вещественных корней нет.

*Приступим к решению поставленной задачи*

Выполнять будем всё в главной функции т.к. объем задачи не большой. И для начала создадим новый проект для выполнения этого задания, назовем его 2020\_PLF06\_01\_Фамилия.

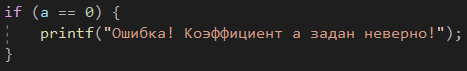
Можно выбрать пустой проект либо не пустой – кому как больше нравится.

1. Внутри функции main нам необходимо объявить шесть переменных – a, b, c, d, x1 и x2.

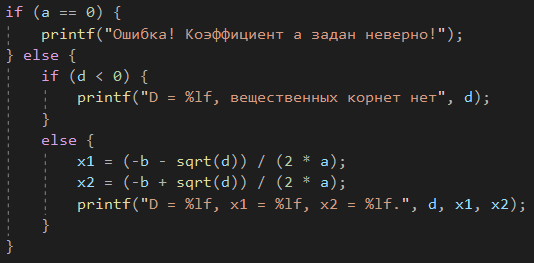
2. Далее спрашиваем у пользователя параметры уравнения по очереди: А, В, С.

3. Спросив у пользователя параметры уравнения, выведем его на экран в формате  
A \* x^2 + B \* x^2 + C = 0, где вместо А, В и С должны стоять введённые пользователем значения.

4. Анализируем параметр уравнения A. В случае, если параметр A равен 0, то выдадим сообщение пользователю и выйдем из программы.



5. В ветви else вычисляем дискриминант и в зависимости от дискриминанта посчитаем корни:



Вот и все проверяем что получилось! Если есть ошибки, устраните их.

Теперь приступаем к отладке, а именно к просмотру значений переменных на каждом шаге в процессе выполнения программы. Этот процесс позволяет проконтролировать, получают ли переменные нужные значения после выполнения операторов, верно ли выполняется выбор ветвей алгоритма в зависимости от условия и пр.

Задача – при параметрах уравнения a = 3, b = -6, c = 2. В пошаговом режиме отследить, как меняются значения переменных.

Точка останова (breakpoint) – место, где программа прекратит свое выполнение в штатном режиме и передаст управление пользователю.

Для начала сделаем точку останова в первой выполняемой строчке программы – строке с приглашением к вводу коэффициента уравнения A. Чтобы это сделать, нужно поставить курсор на эту строчку и нажать F9 (либо в меню «Отладка» выбрать пункт «Перейти к следующей точке останова»; либо щёлкнуть мышью слева от строки в сером поле), после чего должна появиться красная точка слева.



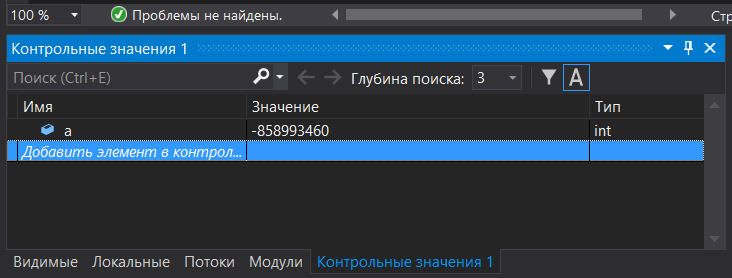
Теперь запустим программу в отладочном режиме, для этого нажмем F5 (либо меню «Отладка» - «Начать отладку»), раньше мы запускали программу без отладки (Ctrl + F5).

В итоге должны выполниться все строки программы до точки останова. В точке останова программа остановится и будет ждать указаний пользователя, что делать дальше. Это будет видно по желтой стрелочке слева:

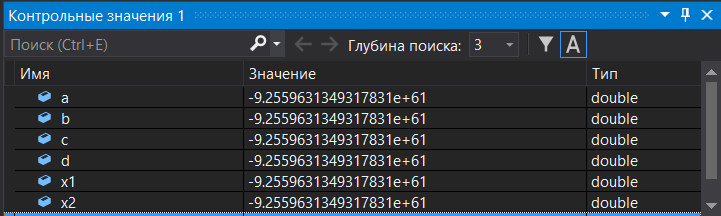


Просмотреть значения переменных можно разными способами. Со всеми способами отслеживания значения переменных предлагаю познакомиться самостоятельно. Опишу лишь один из способов отслеживания значений переменных.

Когда VS переходит в режим отладки, появляется окно «Контрольные значения», которое позволяет отобразить значение любой переменной или выражения с участием переменных. Для добавления переменной в окно «Контрольные значения», нужно выделить имя этой переменной и в её контекстном меню выбрать «Добавить контрольное значение» либо впечатать имя переменной в столбец «Имя». Переменная добавится в окно «Контрольные значения»:

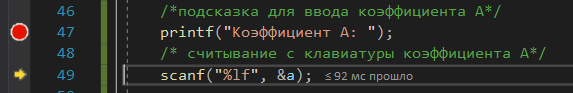


Добавьте в окно «Контрольные значения» все объявленные переменные.

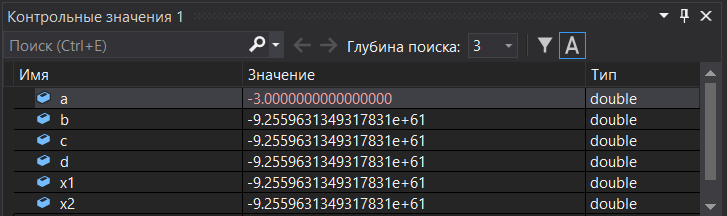


Для перехода к следующей строке программы есть два способа: с заходом в функцию и без захода в функцию. Если выбран заход в функцию, то программа войдёт в вызываемую функцию и начнёт по команде пользователя исполнять её оператор за оператором. Если переход осуществляется без захода в функцию, оператор вызова функции выполнится как простой оператор языка, далее курсор перейдёт к следующему оператору после оператора вызова функции.

Для перехода к следующему оператору без захода (с обходом) нужно нажать F10 (либо F11 для шага с заходом) либо соответствующий пункт меню «Отладка». Желтая стрелка перешла на следующую строку.



Смотрим значения переменных – не поменялись. Переходим к следующей строке. После выполнения этой строки у нас должна была измениться переменная A. Обратите внимание, что VC++ не перешел к следующей строке, переключитесь в окно программы и введите значение -3, нажмите Enter. VC++ перешел к следующей строке, значение переменной изменилось.



Запомнили этот факт. Переходим к следующей строке. Протрассируйте программу до конца (выполняя построчно - пошагово).

В итоге, полученные данные я свел в следующую таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Строка | A | B | C | D |
| printf("Введите параметр \A:"); | -1.07374e+008 | -1.07374e+008 | -1.07374e+008 | -9.2559631349317831e+061 |
| scanf("%f", &a); | -1.07374e+008 | -1.07374e+008 | -1.07374e+008 | -9.2559631349317831e+061 |
| printf("Введите параметр B:"); | 3.00000 | -1.07374e+008 | -1.07374e+008 | -9.2559631349317831e+061 |
| scanf("%f", &b); | 3.00000 | -1.07374e+008 | -1.07374e+008 | -9.2559631349317831e+061 |
| printf("Введите параметр C:"); | 3.00000 | -6.00000 | -1.07374e+008 | -9.2559631349317831e+061 |
| scanf("%f", &c); | 3.00000 | -6.00000 | -1.07374e+008 | -9.2559631349317831e+061 |
| if (a == 0) { | 3.00000 | -6.00000 | 2.00000 | -9.2559631349317831e+061 |
| d = b \* b - 4 \* a \* c; | 3.00000 | -6.00000 | 2.00000 | -9.2559631349317831e+061 |
| if (d < 0) { | 3.00000 | -6.00000 | 2.00000 | 12.000000000000000 |
| if (d == 0) { | 3.00000 | -6.00000 | 2.00000 | 12.000000000000000 |
| printf("Два корня\nДискриминант: %l0.3f\nКорень №1: %0.3lf\nКорень №2: %0.3lf", \  d, (-b + sqrt(d)) / (2 \* a), (-b - sqrt(d)) / (2 \* a)) | 3.00000 | -6.00000 | 2.00000 | 12.000000000000000 |
| } | 3.00000 | -6.00000 | 2.00000 | 12.000000000000000 |

В таблице указывается строка, где остановилась программа и значения переменных в **этой** строке. Строки, где программа **не** останавливалась **не пишутся**!

**1. Самостоятельно сделайте 4 таблицы с данными переменных для 4 случаев:**

1. Уравнение имеет 2 коня.
2. Уравнение имеет 1 корень.
3. Уравнение не имеет корней.
4. Параметр уравнения A задан не правильно.

Таблицы сделать в MS Word или MS Excel и показать для проверки.

**2. С помощью отладки разберитесь, почему данная программа работает неправильно. Устно объясните это преподавателю.**

printf(“Приветствуем вас в нашем казино!“);

printf(“В честью юбилея казино делает подарок выигравшим!”);

printf(“Для выигрышей до 10 тысяч ₽ включительно казино дарит 10% от выигрыша,”);

printf(“для выигрыша на сумму до 100 тысяч включительно - 20%,”);

printf(“для более 100 тысяч - 30%!”);

printf(“Сколько вы выиграли?”);

scanf(“%lf”, &money);

if (money <= 10000) {

money \*= 1.1;

}

if (money <= 100000) {

money \*= 1.2;

}

if (money > 100000) {

money \*= 1.3;

}

printf(“Вы получаете %lf ₽, поздравляем!”);