**Debugging**:

**Debugging in Visual Studio**:

Whenever you write C++ programs, inevitably you're going to have to debug the program. The series of steps will teach you how to debug a c++ program At first, we’ll look into the pre-requisites for debugging a c++ program in vs code. For that we require:

1)Visual Studio Code IDE.

2)Mingw for C++ compilation (g++ for windows, clang++ for macOS).

Step#01: Write a c++ program, save it or run .You may install code runner extension to run and get input from user. To simply run the program **enter: CTRL+ALT+N.**

#include <iostream>

#include<string>

using namespace std;

struct Fruit{

string name;

string color;

};

int main(){

Fruit f[2];

for(int i=0; i<2; i++){

cout << “Enter the fruits you want to add in the basket” << endl;

cin >> f[i].name;

cin >> f[i].color;

}

For(int i=0; i<2; i++){

Cout <<”The fruit is: “ << endl;

Cout << “color: “ << endl;

}

return 0;

}

**Step#02: Start the debugger**.

1) Set up the debugging mode by clicking on the run(run with debugging), then select the compiler ie.(g++ for gdb/ldb for windows, clang++ for macOS)

2) Press **F5** (**Debug > Start Debugging**) or the **Start Debugging** button Start Debugging in the Debug Toolbar. Since we haven't done anything special to examine the code, the app just loads and you see the console output.

3)Stop the debugger by pressing the red stop Stop Debugging button (**Shift** + **F5**).

4)In the console window, press a key and **Enter** to close the console window.

**Step#03: Set up a breakpoint and start the debugger again (enter F5)**

In the for loop of the main function, set a breakpoint by clicking the left margin of the following line of code:

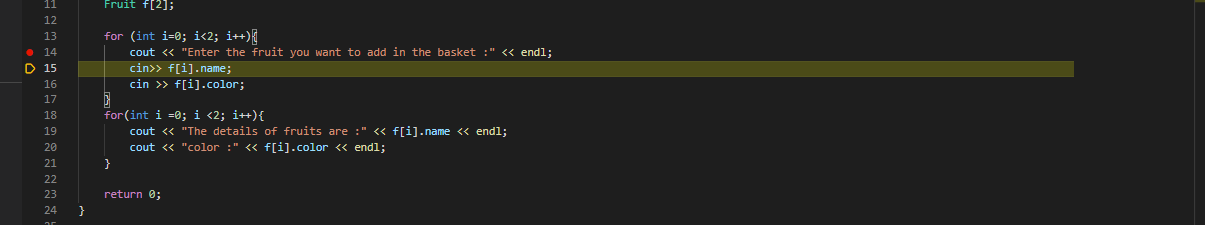
**cout << “Enter the fruit you want to add in the basket” << endl;**

A red circle Breakpoint appears where you set the breakpoint. Breakpoints are one of the most basic and essential features of reliable debugging. A breakpoint indicates where Visual Studio should suspend your running code so you can take a look at the values of variables, or the behavior of memory, or whether or not a branch of code is getting run.

**Step#04: Navigate code in the debugger using step command(Step Into)**

1. While paused in the for loop in the main method, press **F11** (or choose **Debug > Step Into**) twice to to advance to the next line of code.
2. After pressing **F11** twice, you should be at this line of code:

**cin >> f[i].name;**



Press **F11** one more time to step into the next line .The yellow pointer now points to following line of code:

**cin >> f[I].color**;

If you are done with examining the statement above,you can  **Step Out**  using Press **Shift** + **F11** (or **Debug > Step Out**) command

To resume, press F11 several times to get back in cin >> f[i].color .While paused at the method call, press **F10** (or choose **Debug > Step Over**) once.

**Additional Features:**

1. **Restart your program execution**:Click the **Restart** Restart App button in the Debug Toolbar (**Ctrl** + **Shift** + **F5**).
2. **Inspect variables with Autos and Locals windows**: Look at the **Autos** window at the bottom of the code editor or by choosing **Debug** > **Windows** > **Autos.**
3. Next, look at the **Locals** window, in a tab next to the **Autos** window.
4. Expand the letters variable to show the elements that it contains.
5. **Set a Watch:** In the main code editor window, right-click the name variable and choose **Add Watch**.
6. The watch window will show you how the value of specified variable/expression changes with program flow.
7. **Examine the Call Stack:** While paused in the for loop, click the Call Stack window, which is by default open in the lower right pane or choose **Debug** > **Windows** > **Call Stack**. The **Call Stack** window shows the order in which methods and functions are getting called