

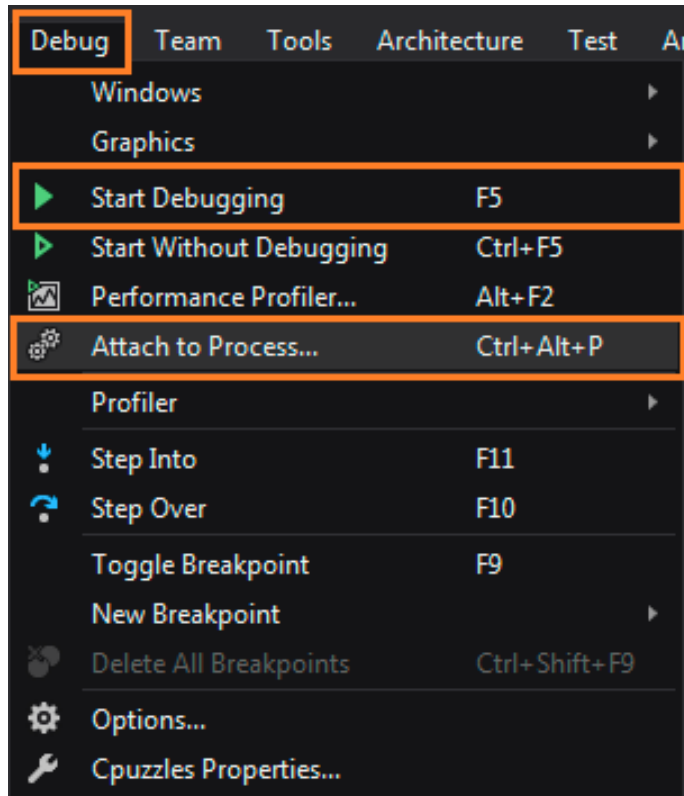
Отладка в Visual Studio

Отладка

Существуют две технологии отладки:

- Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран или в файл (создание логов).

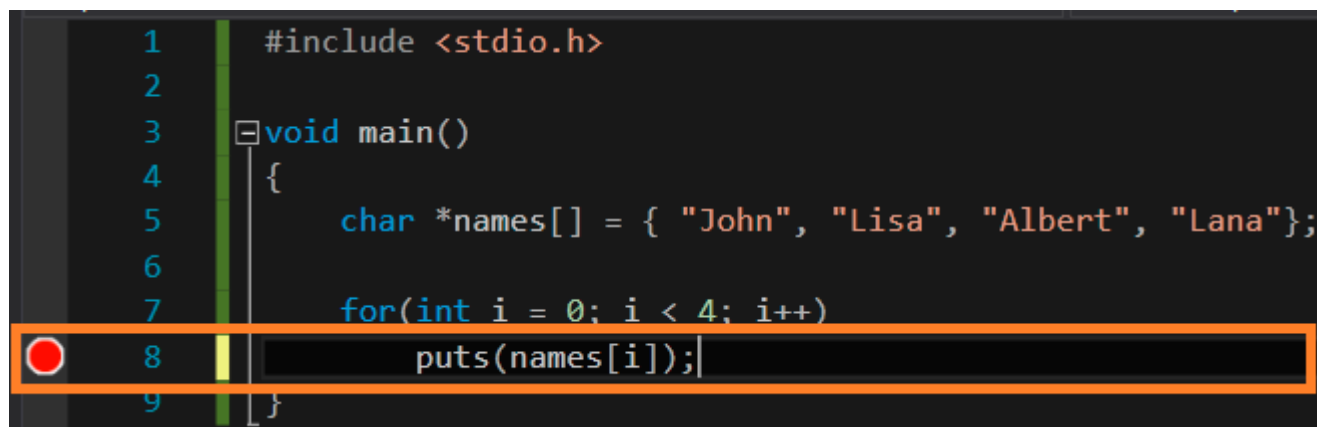
Как начать



Нажать **F5**.

Отладка начнется если
стоят точки останова
(**breakpoints**)

Точки останова (Breakpoints)



```
1  #include <stdio.h>
2
3  void main()
4  {
5      char *names[] = { "John", "Lisa", "Albert", "Lana"};
6
7      for(int i = 0; i < 4; i++)
8      puts(names[i]);
9  }
```

The image shows a code editor with a dark background. A C program is displayed, and a breakpoint (a red circle) is set on line 8, which contains the statement `puts(names[i]);`. The line is highlighted with a yellow background, and the entire line is enclosed in an orange rectangular box. The code is as follows:

- Точки останова используются чтобы показать , где отладчику необходимо остановиться.
- Точка ставится кликом на сайдбар слева от исходного кода, либо нажатием на **F9**.
- Точки останова обычно ставятся там, где есть сомнения в корректности кода.

Отладка с использованием точек останова (Debugging with Breakpoints)

- **Перешагнуть (Step Over)** **F10** – автоматически выполняет блок кода под курсором.
- **Зайти (Step Into)** **F11** – заходит в блок кода под курсором.
- **Выйти (Step Out)** **Shift + F11** – выходит из текущего блока.
- **Продолжить (Continue)** **F5** – переходит к следующей точке останова.

```
1  #include <stdio.h>
2  void function()
3  {
4      puts("Break Point in function()");
5  }
6
7  void main()
8  {
9      int num = 0;
10     function();
11     puts("We are in main()");
12 }
```

Нажать F10

```
1  #include <stdio.h>
2  void function()
3  {
4      puts("Break Point in function()");
5  }
6
7  void main()
8  {
9      int num = 0;
10     function();
11     puts("We are in main()");
12 }
```

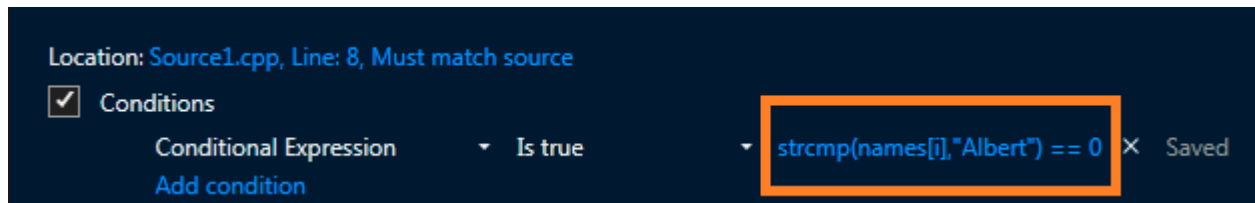
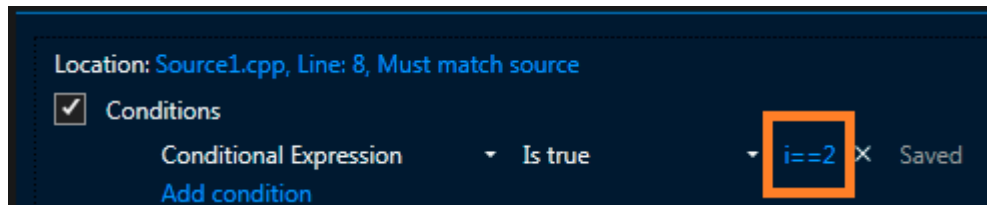
Нажать F11

Условные остановки (Conditional Breakpoint)



- В циклах может обрабатываться большое количество данных.
- Условная остановка нужна чтобы остановить выполнение кода в нужном месте

Условные остановки (Conditional Breakpoint)



```
{  
    char *names[] = { "John",  
    for(int i = 0; i < 4; i++)  
        puts(names[i]);  
}
```

```
{  
    char *names[] = { "John", "Lisa", "Albert",  
    for(int i = 0; i < 4; i++)  
        puts(names[i]);  
}
```

▶ names[i] 🔍 0x00f76b58 "Albert" ⇄

Количество остановок (Breakpoint Hit Count)

The screenshot shows the Visual Studio IDE with a C++ source file. A breakpoint is set at line 8 of Source1.cpp. The Breakpoint Settings window is open, showing the 'Conditions' tab. The 'Hit Count' is set to 3. The Autos window shows the current state of variables: i is 2, names is an array of 4 strings, and names[i] is 'Albert'. The Breakpoints window shows the breakpoint at Source1.cpp, line 8, with a hit count of 3.

```
4 {  
5     char *names[] = { "John", "Lisa", "Albert", "Lana"};  
6  
7     for(int i = 0; i < 4; i++)  
8         puts(names[i]);
```

Breakpoint Settings

Location: Source1.cpp, Line: 8, Must match source

☒ Conditions

Hit Count = 3 (Current: 3 Reset) X Saved

Add condition

132 %

Autos

Name	Value	Type
i	2	int
names	0x002df8ec {0x00f76b30 "John", 0x00f76b50 "Lisa", 0x00f76b58 "Albert", 0x00f76b5c "Lana"}	char *[4]
names[i]	0x00f76b58 "Albert"	char *

Breakpoints

Name	Labels	Condition	Hit Count
Source1.cpp, line 8	NAMES	(no condition)	when hit count is equal to 3 (currently 3)


Отслеживание сколько остановок отладчик сделает на конкретной точке останова









Подсказки (Data Tip)


```
#include <stdio.h>

void main()
{
    char *names[] = { "John", "Lisa", "Albert", "Lana"};

    for(int i = 0; i < 10; i++)
        puts(names[i]); ≤ 2ms elapsed
}
```

 i | 4

 (names)[0]	 - 0x00366b30 "John"
 (names)[1]	 - 0x00366b50 "Lisa"
 (names)[2]	 - 0x00366b58 "Albert"
 (names)[3]	 - 0x00366b60 "Lana"

 names[i] | 0xffffffff <Error reading characters of string.>

- Можно через подсказки менять значения

Окно просмотра данных (Watch Windows)

The screenshot shows a debugger interface with a C code editor and a Locals window. The code defines a 2D character array `names` with 5 rows and 10 columns, containing the names "John", "Lisa", "Albert", and "Lana". A `for` loop is shown, and the `puts` function is called. The `names[5][10]` expression in the code is highlighted with an orange box. The Locals window at the bottom shows the current state of variables: `i` is 0, and `names` is a 2D array of characters. The `names` variable is expanded, showing the first row `names[0]` containing the string "John". The `Names` tab in the bottom-left corner is highlighted with an orange box.

```
1  #include <stdio.h>
2
3  void main()
4  {
5      char names[5][10] = { "John", "Lisa", "Albert", "Lana"};
6
7      for(int i = 0; i < 4; i++)
8          puts(names[i]);
9  }
```

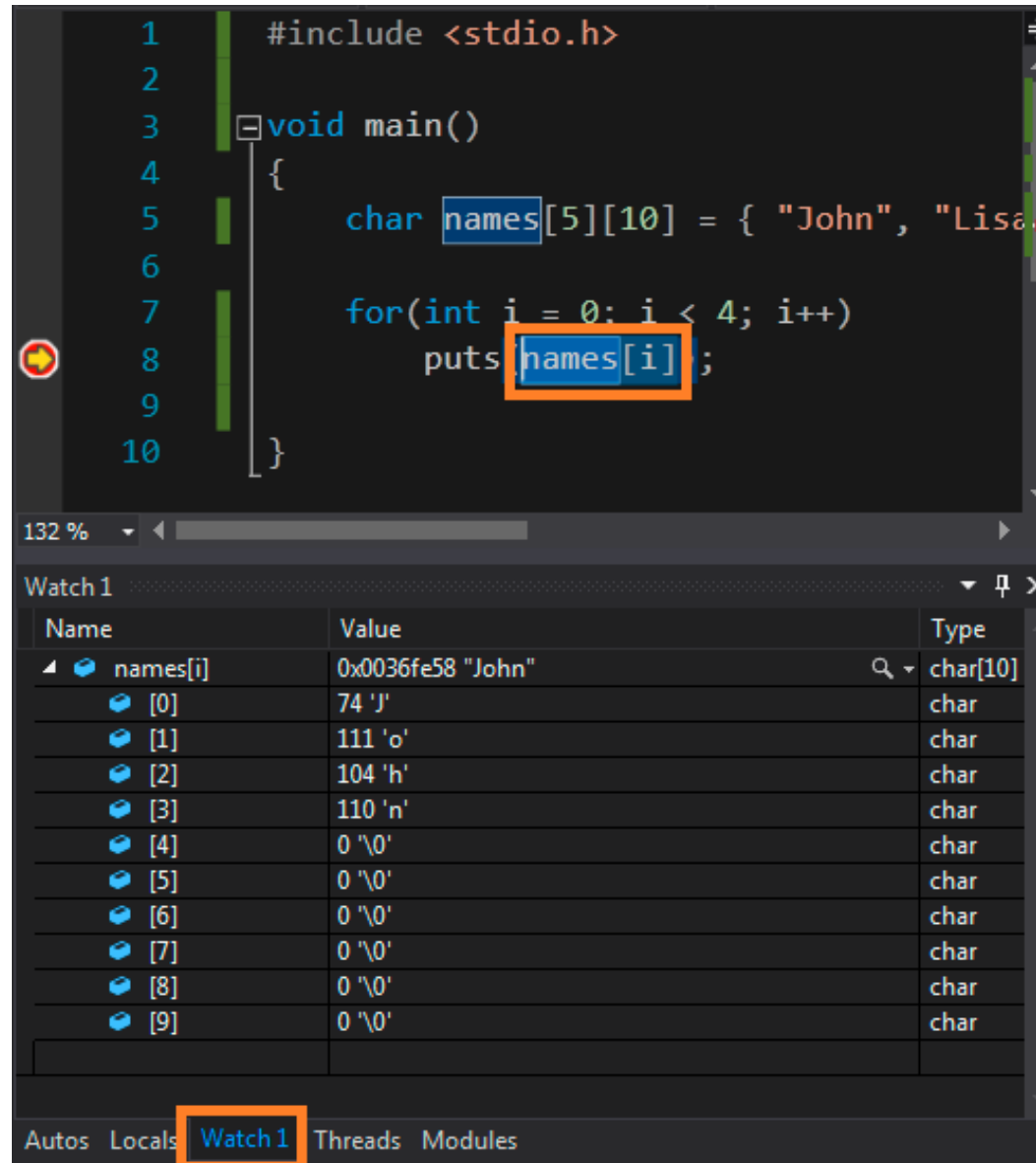
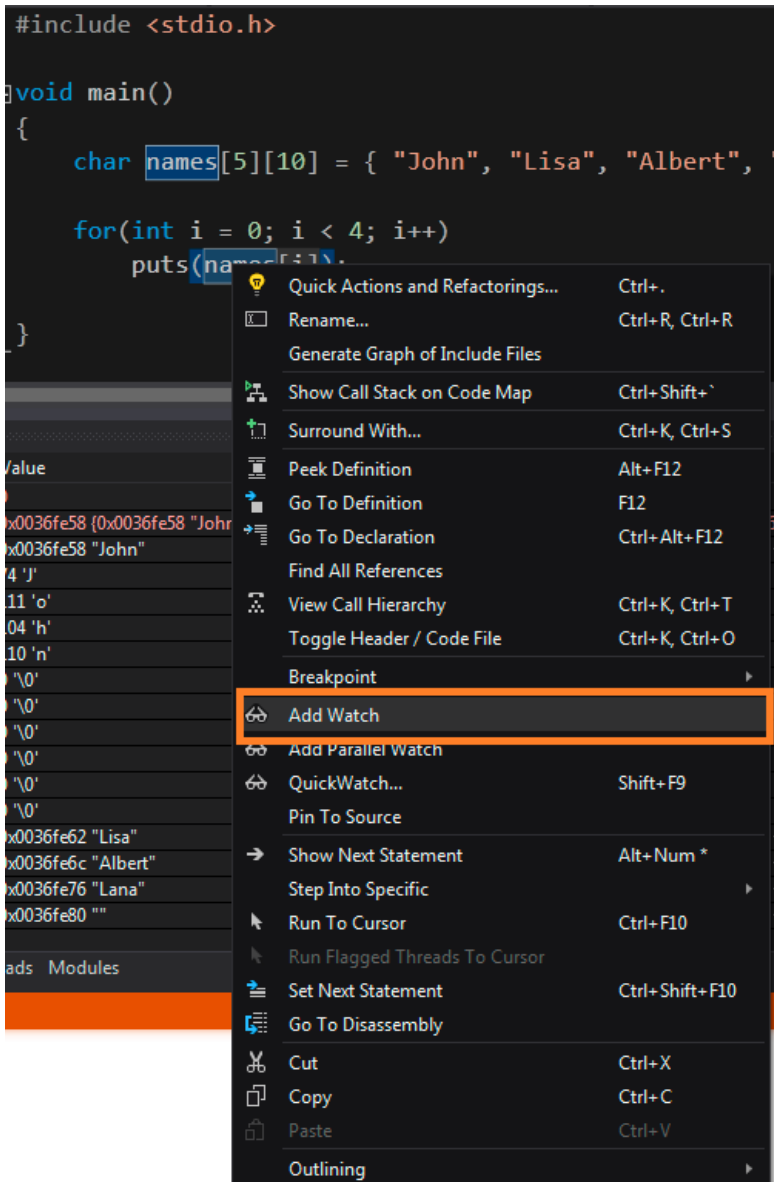
132 %

Locals

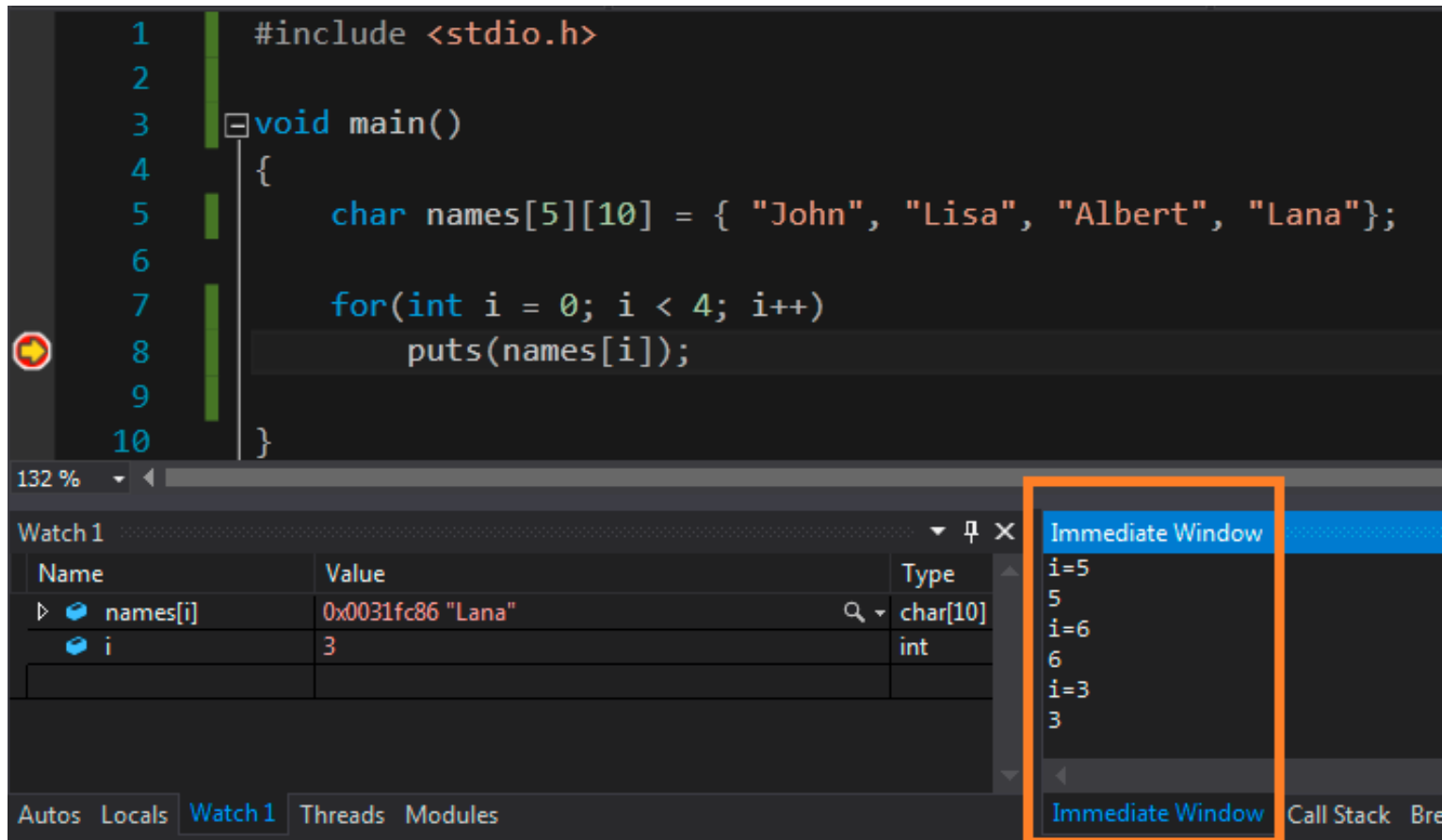
Name	Value	Type
i	0	int
names	0x0036fe58 {0x0036fe58 "John", 0x0036fe62 "Lisa", 0x0036fe6c "Albert", 0x0036fe76 "Lana", 0x0036fe80 ""}	char[5][10]
names[0]	0x0036fe58 "John"	char[10]
names[0][0]	74 'J'	char
names[0][1]	111 'o'	char
names[0][2]	104 'h'	char
names[0][3]	110 'n'	char
names[0][4]	0 '\0'	char
names[0][5]	0 '\0'	char
names[0][6]	0 '\0'	char
names[0][7]	0 '\0'	char
names[0][8]	0 '\0'	char
names[0][9]	0 '\0'	char
names[1]	0x0036fe62 "Lisa"	char[10]
names[2]	0x0036fe6c "Albert"	char[10]
names[3]	0x0036fe76 "Lana"	char[10]
names[4]	0x0036fe80 ""	char[10]

Autos **Names** Threads Modules

Окно просмотра данных (Watch Windows)



Оперативные изменения (Immediate Window)



- **Debug > Window > Immediate Window**
- Позволяет задавать значения выражений/переменных во время отладки